

Logical Design

CS 221

Prof.Dr. Mohamed Osama Khozium





Prof.Dr. Mohamed Osama Khozium



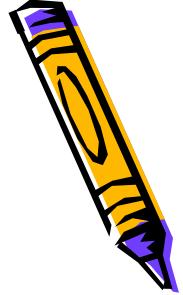
- Ph. D. I.S. "Faculty of Computers & Information" Cairo University
- Title :

*Intelligent Information System for Jamming
and Anti-Jamming Applications of
Electromagnetic Spectrum*

- ◆ Ms.C.I. Institute of Statistical Studies & Research Cairo University

- ◆ Title :
Computer – Based System For Unmanned Air Vehicle Mission Planning





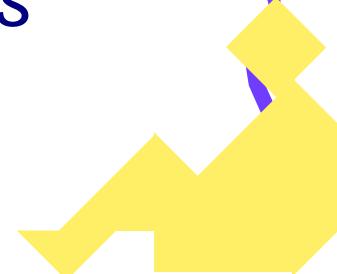
Dr. Mohamed Osama Khozium

- Ms. M.Sc. "Aviation Academy"
- Title :

*Laser Applications in Electronic Warfare to
Secure Egyptian Air Force Missions*

- ◆ D.C.Sc.I.Institute of Statistical Studies & Research
Cairo University

- ◆ Title :
Data base system for World Countries

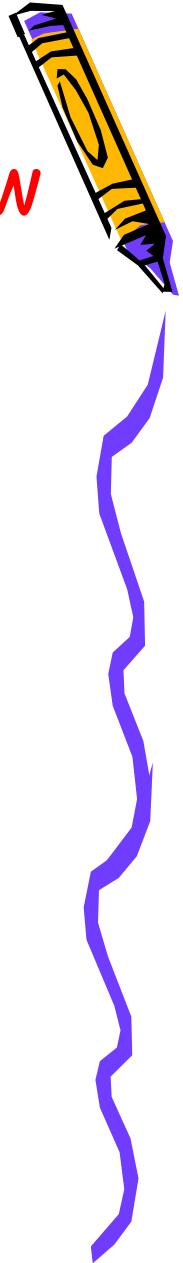




Lecture 1 : Logic Design Course overview

Agenda

- Logic design Background
- Course Team
- Required Textbook
- Course Outlines
- Course Requirements
- Course Time





Logic design Background

- Logic design is also known by other names such as : digital design, digital logic, switching circuits and digital systems





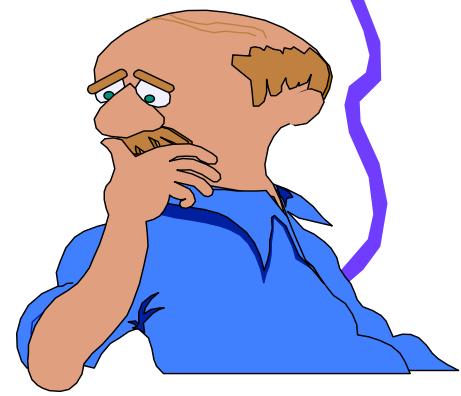
Course Team

Instructor: Prof.Dr Mohamed Osama Khozium

E-mail: osama@khozium.com

Assist :

E-mail :





Required Textbook

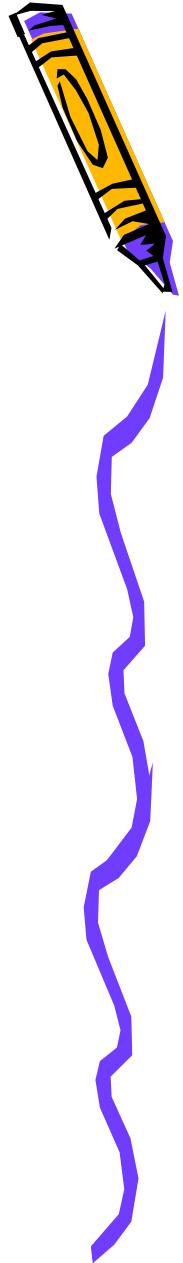
DIGITAL DESIGN

By

M.MORRIS MANO

Published by

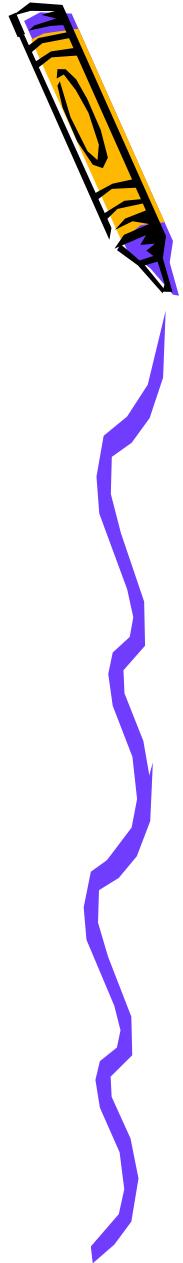
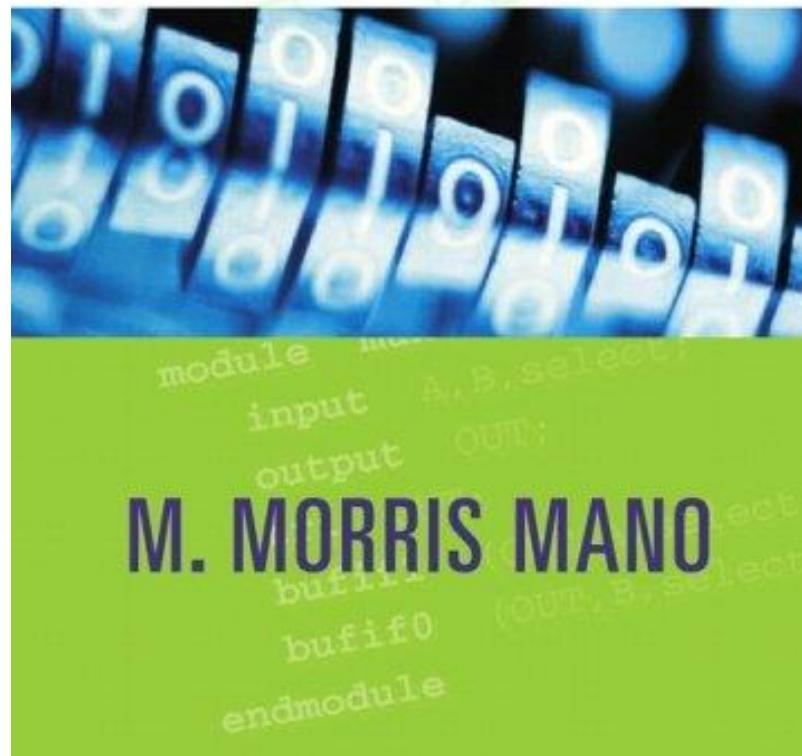
Prentice – Hall International Editions





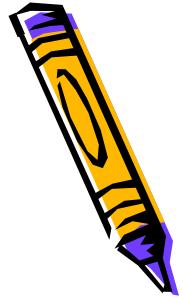
DIGITAL DESIGN

THIRD EDITION





Course Outlines



1 - Numbering systems.

- a - Overview and definitions.
- b - Decimal numbering systems.
- c - binary numbering systems.
- d - Octal numbering systems.
- e - Hexadecimal numbering system.

2 - conversions between numbering systems including fractions

3 - Binary arithmetic

- a - Binary additions
- b - binary subtraction (one's and two's complements of binary numbers).
- c - Binary tools





Course Outlines

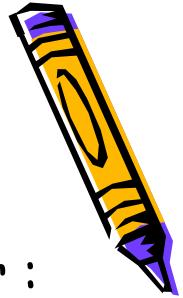
4 - Simple logic circuits

Logic circuits (standard), truth tables, and Boolean expressions for :

- a - OR gate
- b - AND gate
- c - NOT gate
- d - NAND gate
- e - NOR gate
- f - Exclusive – OR gate (XOR – gate)
- g - Exclusive – NOR gate (XNOR – gate)

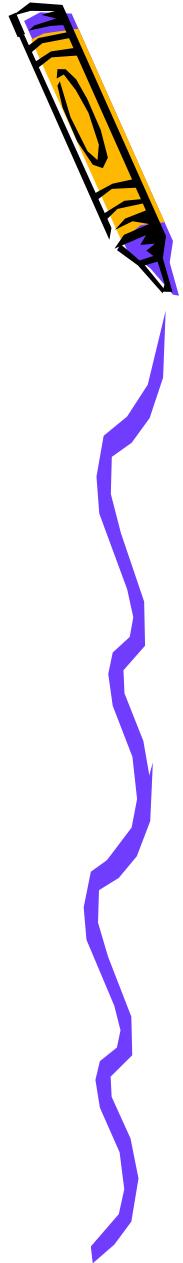
5 - Rules of Boolean algebra

- a - Basic rules
- b - The Boolean expression for a logic circuit
- c - Implementation of a logic circuit using a Boolean expression
- d - Implementation of a logic circuit via a truth table
- e - Converting Boolean expression to a truth table.
- F - Simplification of Boolean expression using Boolean algebra





Course Outlines



6 – Demorgan's theorems

- a - NAND gate as universal element
- b - NOR gate as universal element
- c - Sum of Product Σ
- d - Product of Sum Π

7 - Karnaugh Map

Simplification using Karnaugh map

8 - combinational logic circuits

- a - Half adder
- b - Full adder





Course Outlines

9 - Decoder, encoder and multiplexer

10- Sequential logic circuits

 flip – flop types

 a - set – reset FF

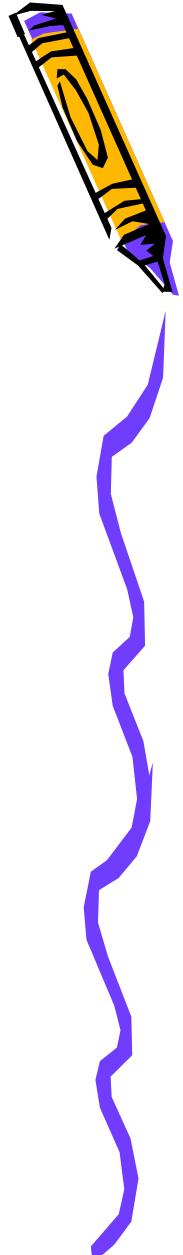
 b - J-K FF

 c - Delay FF

 d -Trigger FF

11- Flip Flop applications

Registers – Counters

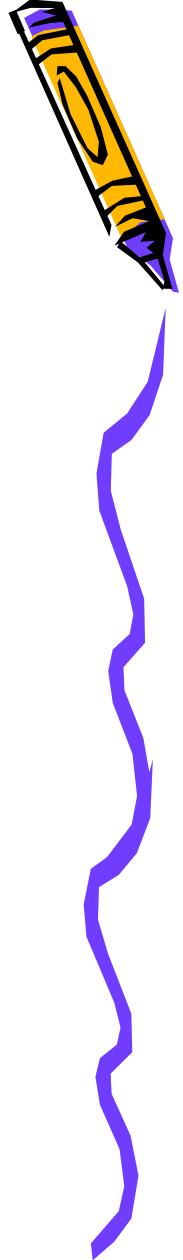
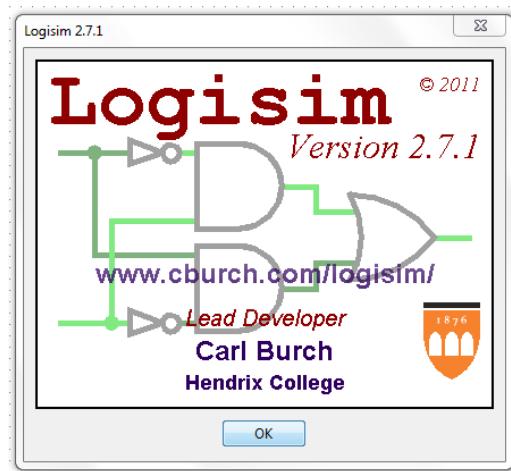


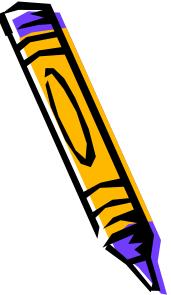


Practical Part

Training for :

- * conversations
- * simplifications
- * Implementations
- * Simulation soft ware (LOGISIM)

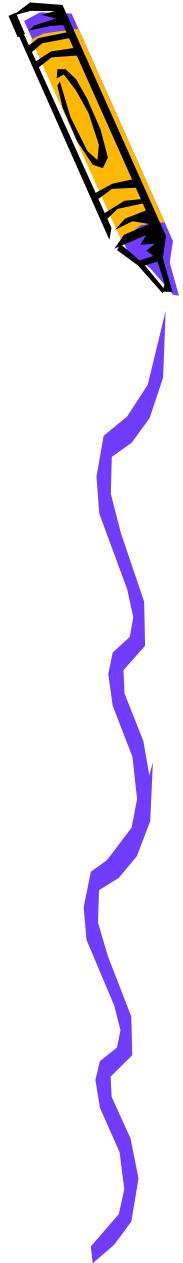




Course Requirements

- Ten lectures plus
 - Lecture in the beginning (overview)
 - Lecture in the end (any required questions)
- One week for mid term exam
- First quiz will be conducted during the beginning of sixth week.
- Midterm exam will be conducted during the beginning of the eighth week.





Term Assessment:

Class activity: [25%]

participation and attendance, homework: 10%

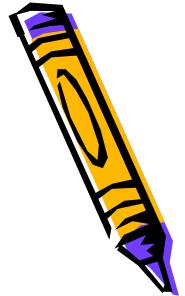
quizzes → 5%

Lab and exercise activities → 10%

Midterm Exam: 15%

Final written exam : 60%





Late Policy

- Late assignments will receive a 10% penalty per day,
 - * unless a Doctor's note is provided.
-
- * Thursdays & Fridays count as late days as well





Course Time

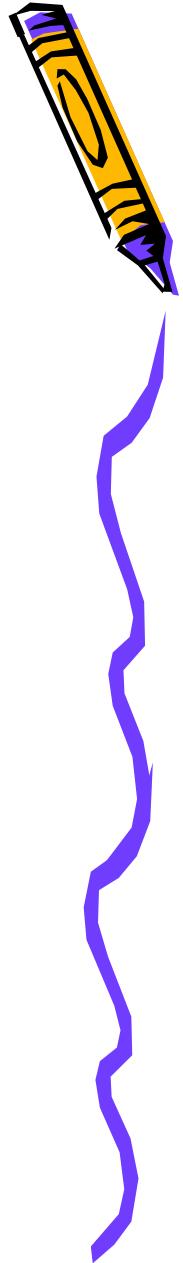
Lectures :

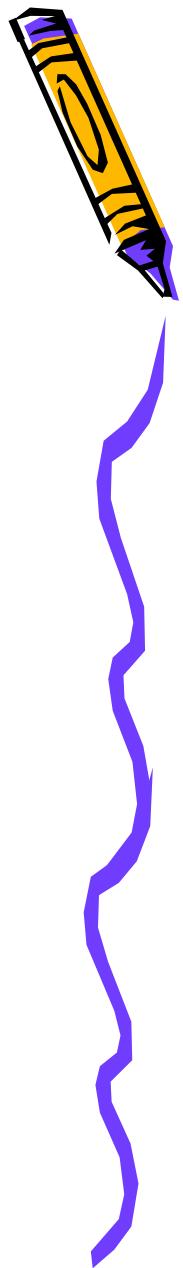
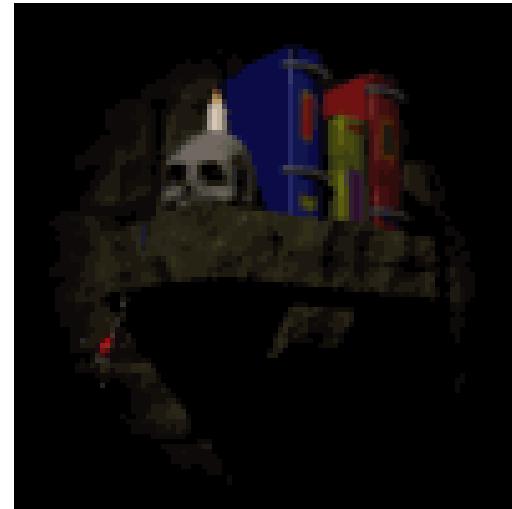
9-11 AM, 11-1PM, 1-3PM

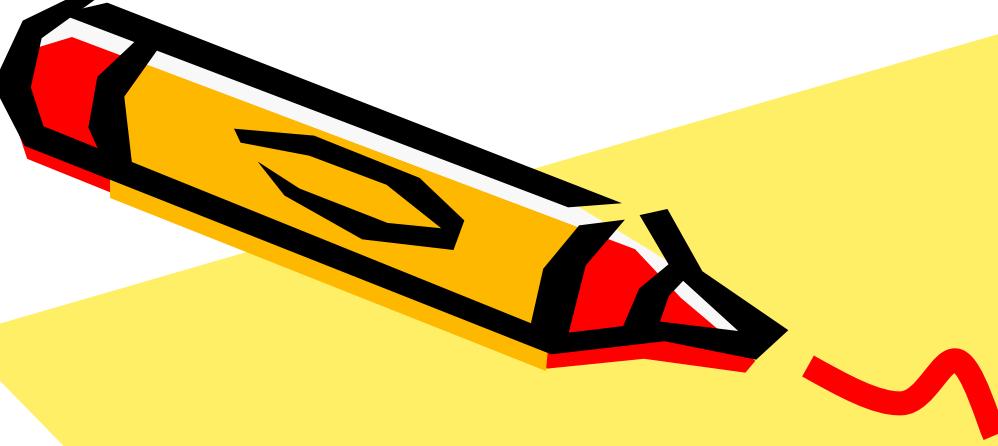
9-11 AM, 11-1PM, 1-3PM

o'clock Saturday

o'clock Tuesday.





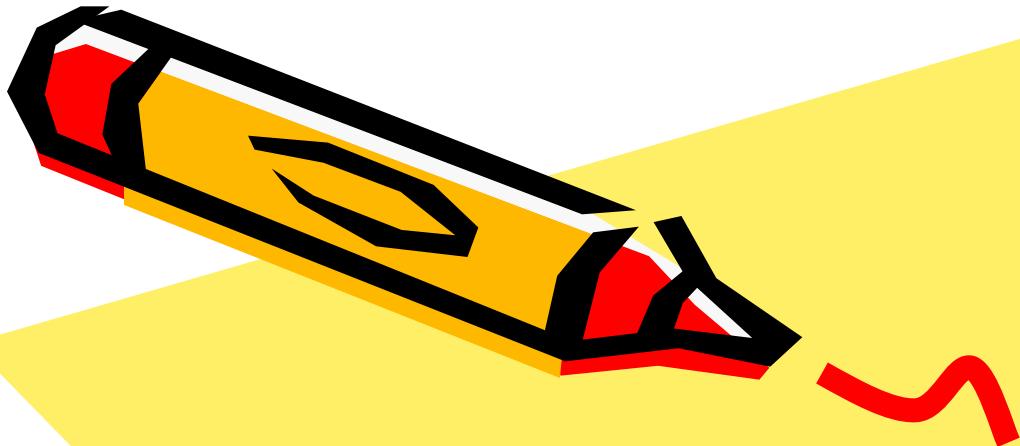


Logical Design

CS 221

Prof.Dr. Mohamed Osama Khozium





1. Number Systems

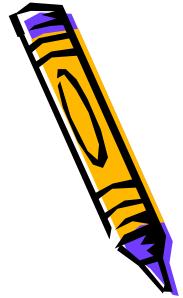
Location in
course textbook



Chapt. 1

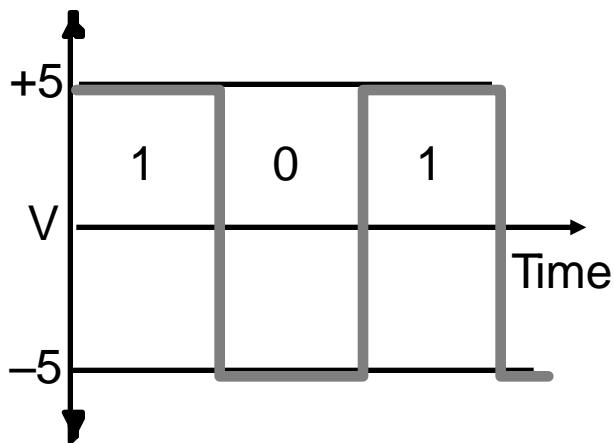


Digital Hardware Systems

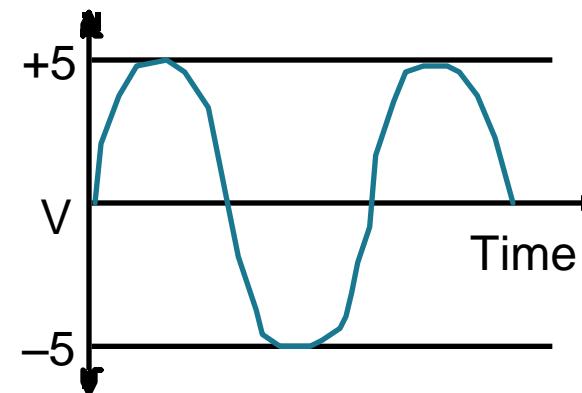


Digital Systems

Digital vs. Analog Waveforms



Digital:
only assumes discrete values



Analog:
values vary over a broad range
continuously





Digital Hardware Systems



- Digital Binary System
 - Two discrete values:
 - yes, on, 5 volts, current flowing, "1"
 - no, off, 0 volts, no current flowing, "0"
 - Advantage of binary systems:
 - Rigorous (exact) mathematical foundation based on logic
 - it's easy to implement

IF the garage door is open
AND the car is running
THEN the car can be backed out of the garage

both the door must be open and the car running before I can back out

the preconditions must be true to imply the conclusion





Binary Bit and Group Definitions

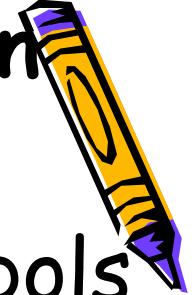


- Bit - a single binary digit
- Nibble - a group of four bits
- Byte - a group of eight bits
- Word - depends on processor; 8, 16, 32, or 64 bits
- LSB - Least Significant Bit (on the right)
- MSB - Most Significant Bit (on the left)





Binary Representation of Information



- Information divided into groups of symbols
 - 26 English letters
 - 10 decimal digits
 - 50 states in USA
- Digital systems manipulate information as 1's & 0's
- The mapping of symbols to binary value is known as a “code”
- The mapping must be unique





Common Number Systems



System	Base	Symbols	Used by humans?	Used in computers?
Decimal	10	0, 1, ... 9	Yes	No
Binary	2	0, 1	No	Yes
Octal	8	0, 1, ... 7	No	No
Hexa-decimal	16	0, 1, ... 9, A, B, ... F	No	No





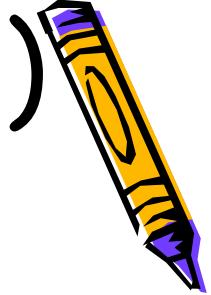
Quantities/Counting (1 of 3)

Decimal	Binary	Octal	Hexa-decimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7





Quantities/Counting (2 of 3)



Decimal	Binary	Octal	Hexa-decimal
8	1000	10	8
9	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

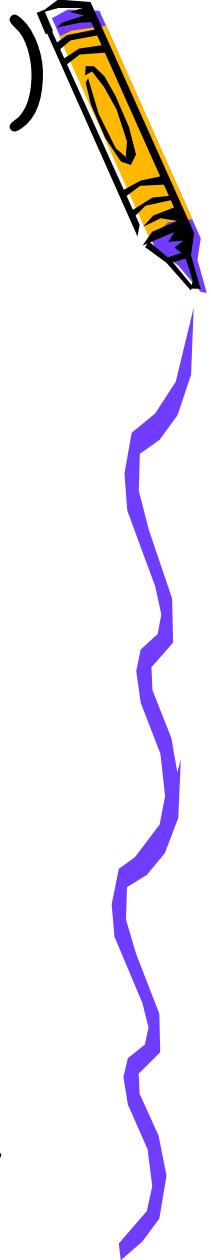




Quantities/Counting (3 of 3)

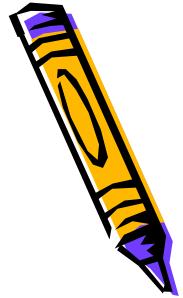
Decimal	Binary	Octal	Hexa-decimal
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17

Etc.

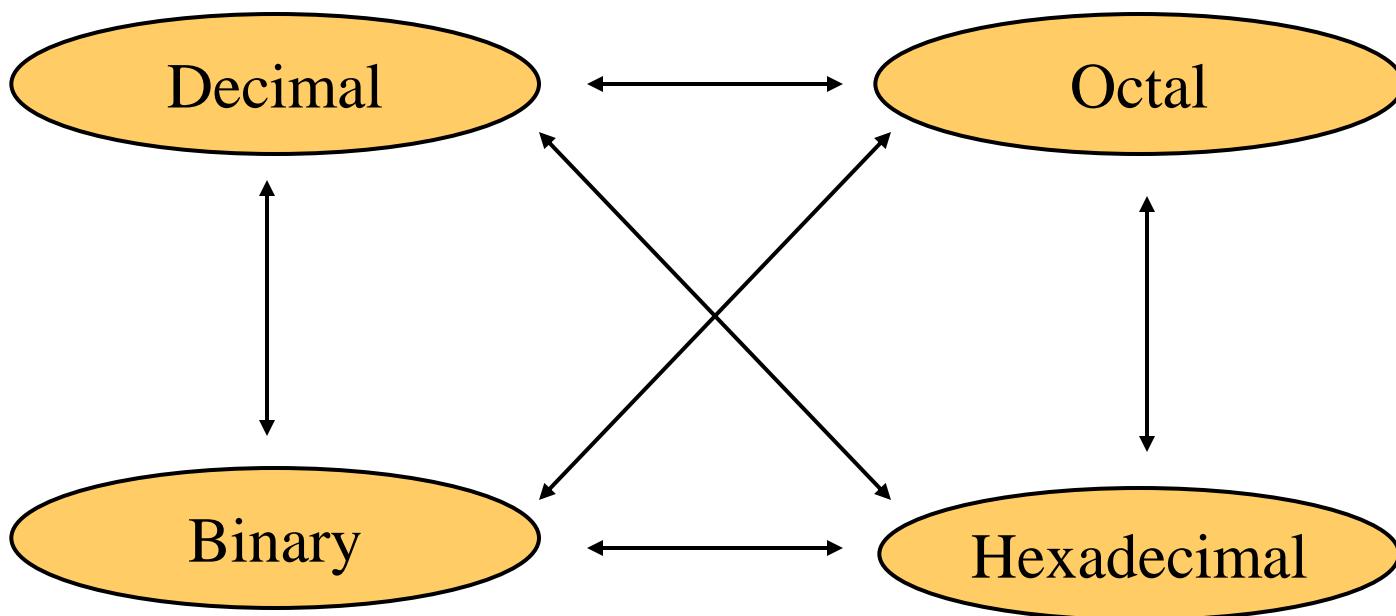




Conversion Among Bases



- The possibilities:

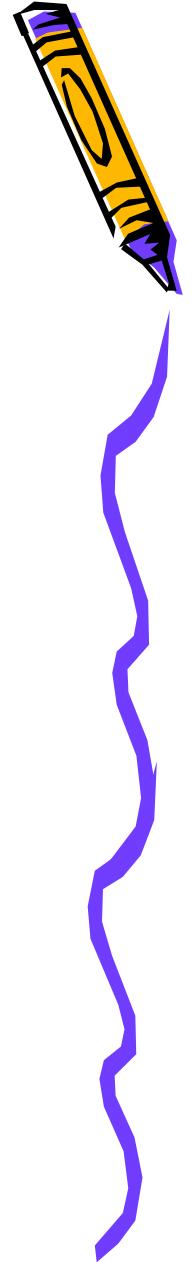




Quick Example

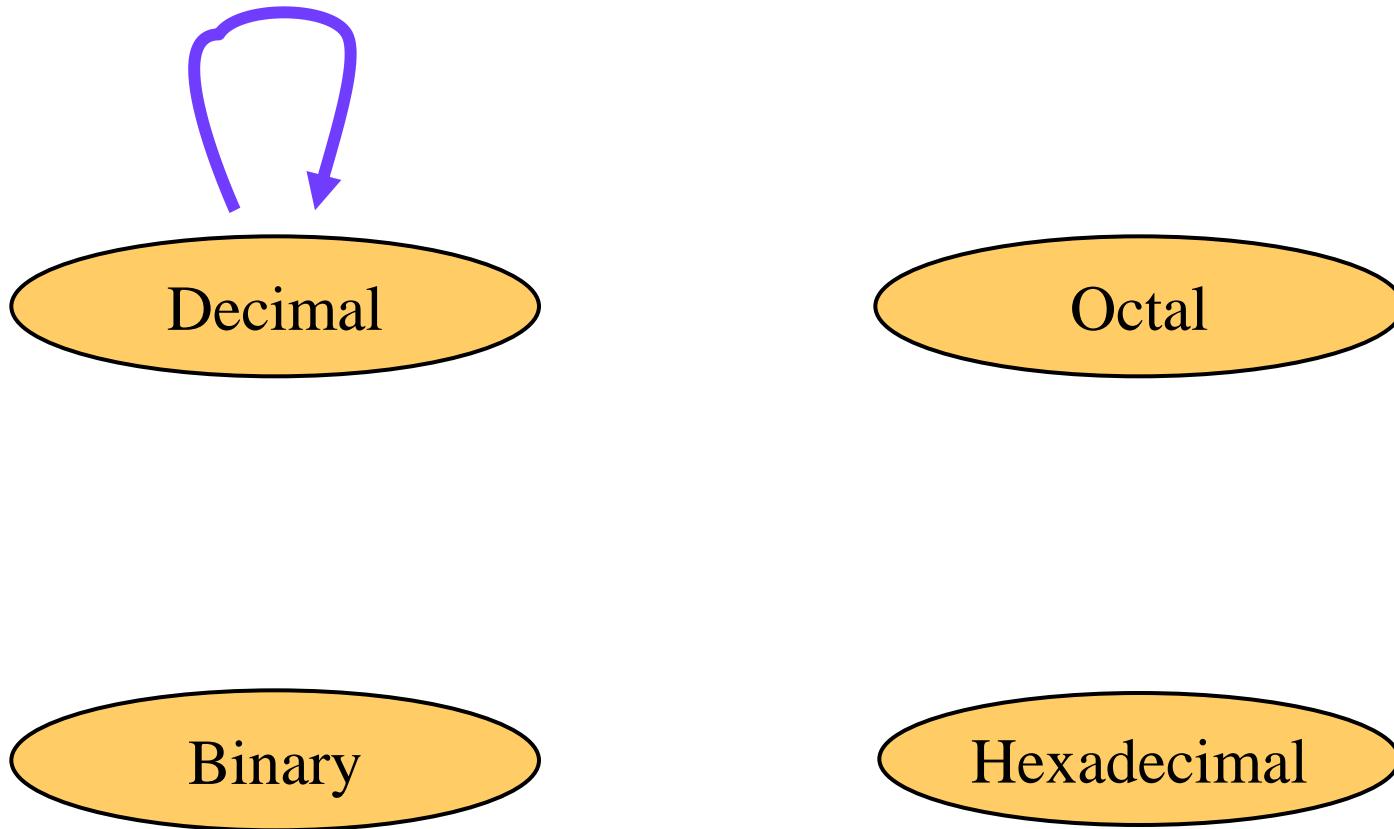
$$25_{10} = 11001_2 = 31_8 = 19_{16}$$

Base

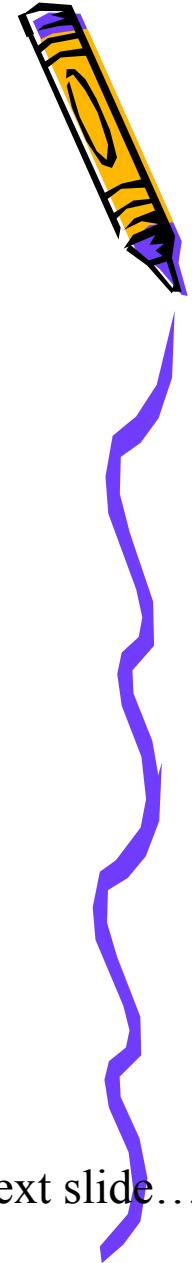


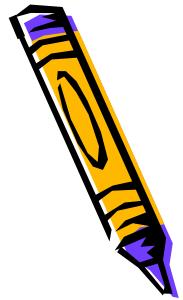


Decimal to Decimal (just for fun)



Next slide...





Weight

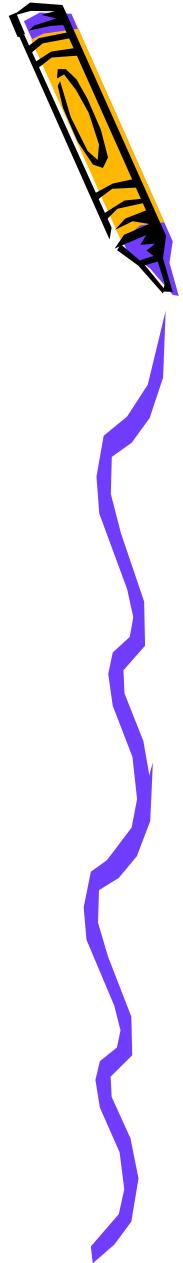
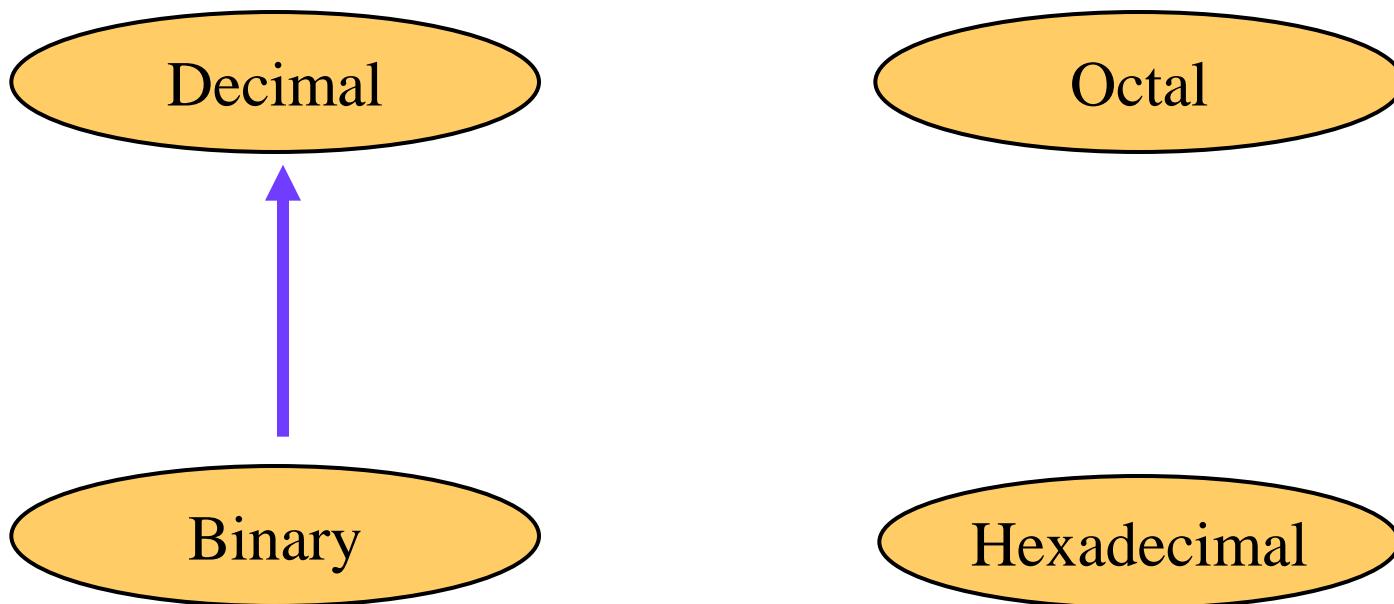
$$\begin{array}{rcl} 125_{10} \Rightarrow & 5 \times 10^0 & = 5 \\ & 2 \times 10^1 & = 20 \\ & 1 \times 10^2 & = \underline{100} \\ & & 125 \end{array}$$

Base



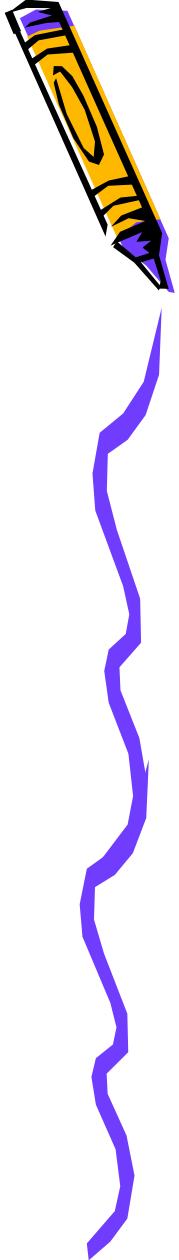


Binary to Decimal





Binary to Decimal

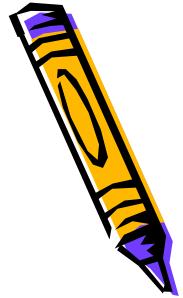


- Technique
 - Multiply each bit by 2^n , where n is the “weight” of the bit
 - The weight is the position of the bit, starting from 0 on the right
 - Add the results





Example



Bit “0”

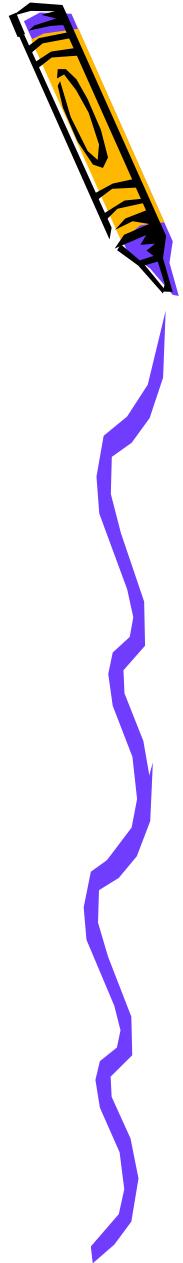
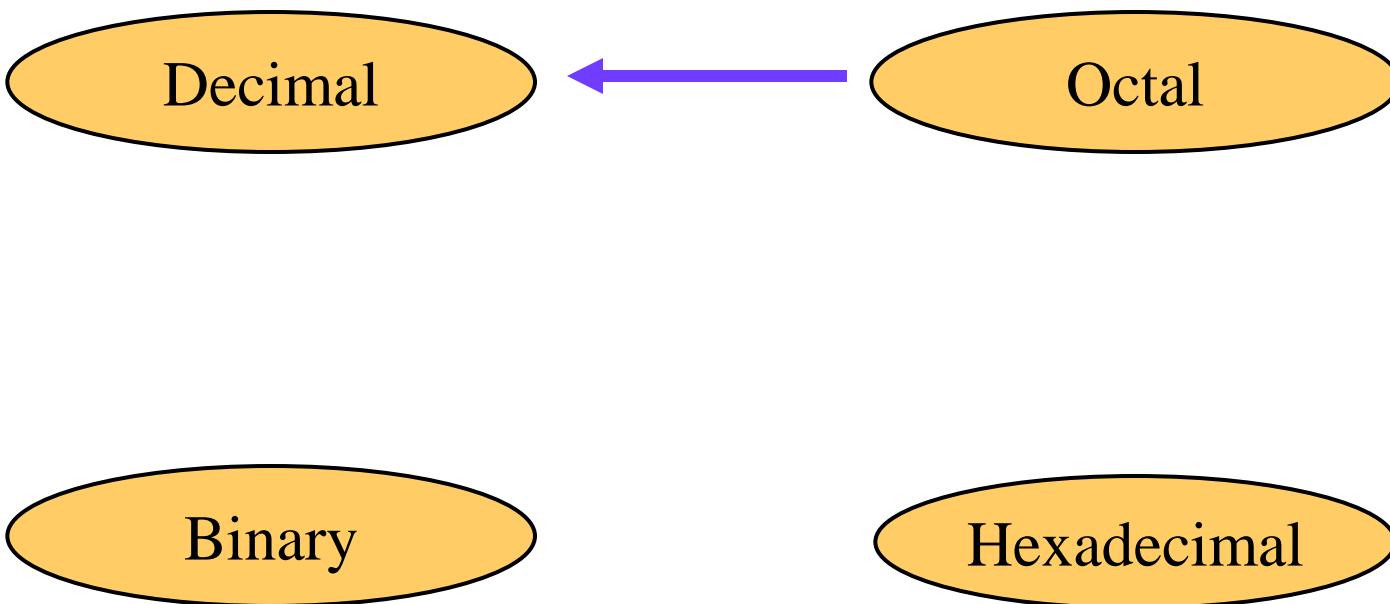
$101011_2 \Rightarrow$

$$\begin{array}{rcl} 1 & \times & 2^0 = 1 \\ 1 & \times & 2^1 = 2 \\ 0 & \times & 2^2 = 0 \\ 1 & \times & 2^3 = 8 \\ 0 & \times & 2^4 = 0 \\ 1 & \times & 2^5 = 32 \\ & & \hline & & 43_{10} \end{array}$$



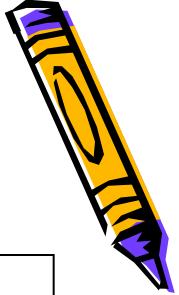


Octal to Decimal





Octal to Decimal



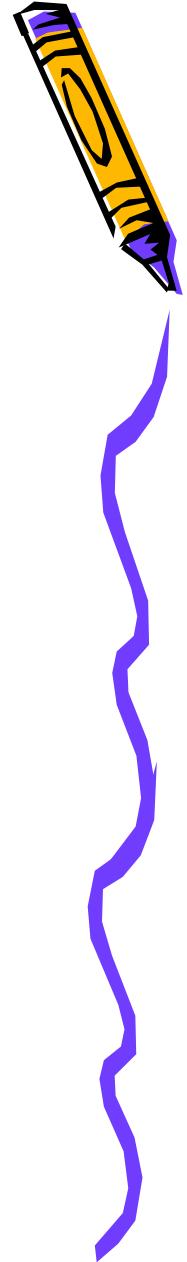
- Technique
 - Multiply each bit by $\underline{8}^n$, where n is the “weight” of the bit
 - The weight is the position of the bit, starting from 0 on the right
 - Add the results





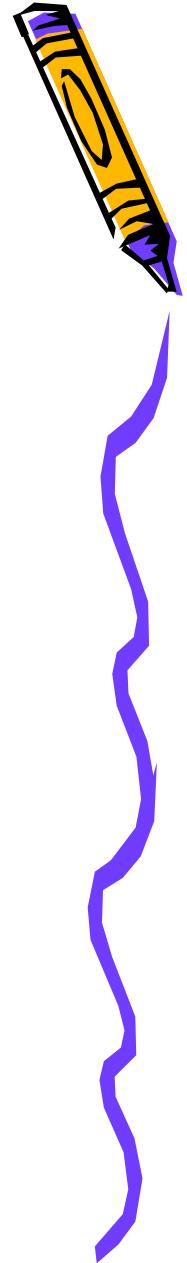
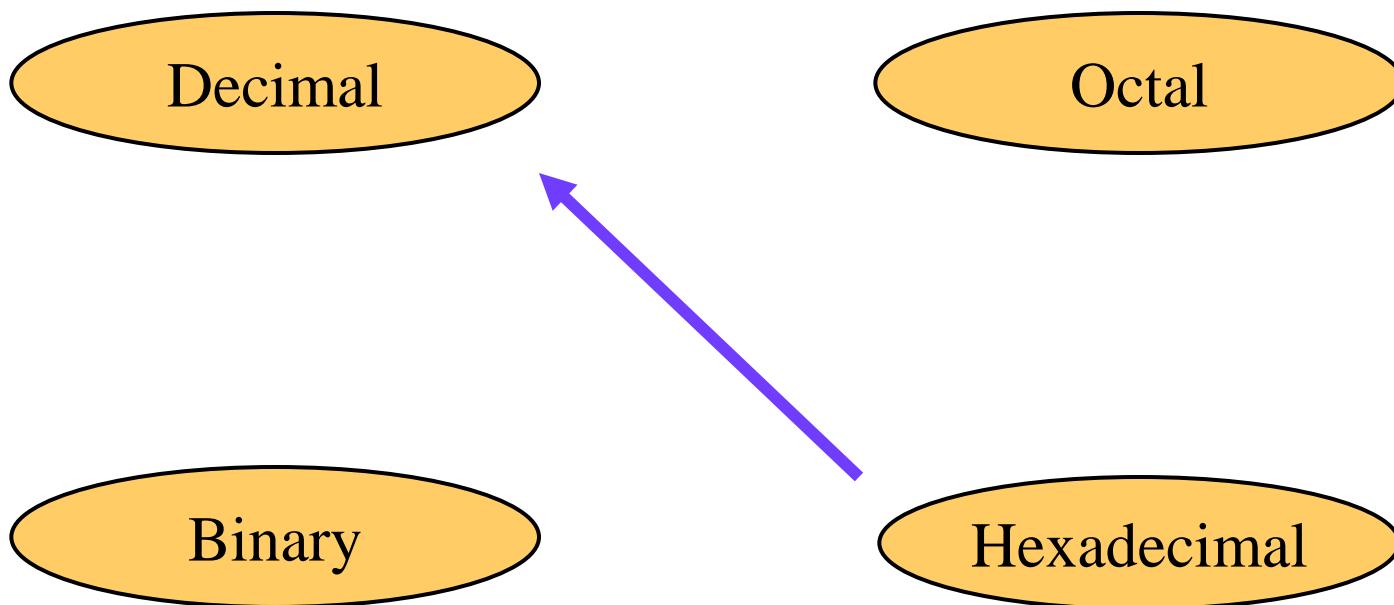
Example

$$724_8 \Rightarrow \begin{array}{rcl} 4 & \times & 8^0 = 4 \\ 2 & \times & 8^1 = 16 \\ 7 & \times & 8^2 = 448 \\ & & \hline & & 468_{10} \end{array}$$





Hexadecimal to Decimal





Hexadecimal to Decimal

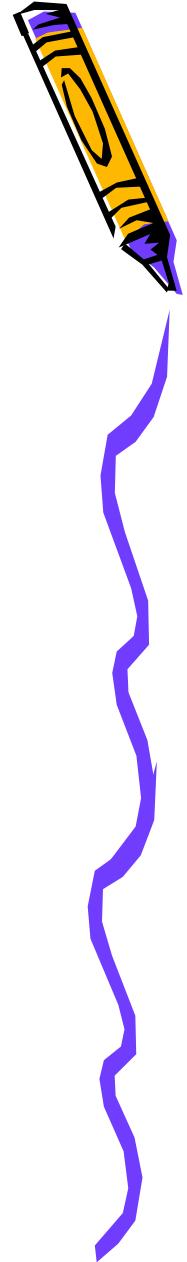


- Technique
 - Multiply each bit by 16^n , where n is the “weight” of the bit
 - The weight is the position of the bit, starting from 0 on the right
 - Add the results





Example

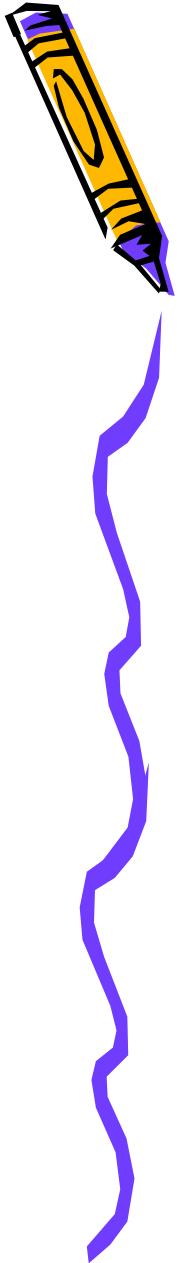
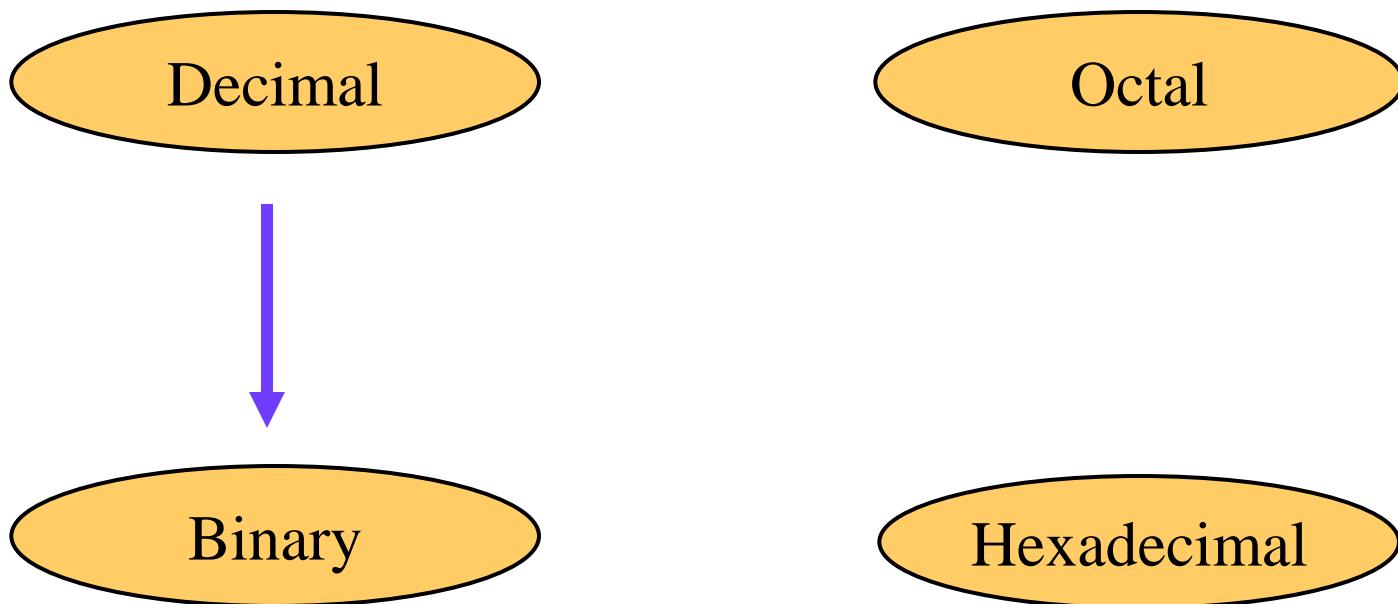


$$\begin{array}{rcl} \text{ABC}_{16} \Rightarrow & C \times 16^0 = 12 \times 1 = 12 \\ & B \times 16^1 = 11 \times 16 = 176 \\ & A \times 16^2 = 10 \times 256 = \underline{2560} \\ & & 2748_{10} \end{array}$$





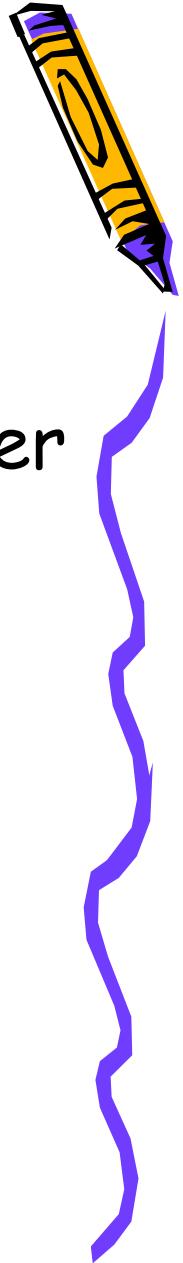
Decimal to Binary





Decimal to Binary

- Technique
 - Divide by two, keep track of the remainder
 - First remainder is bit 0 (LSB, least-significant bit)
 - Second remainder is bit 1
 - Etc.



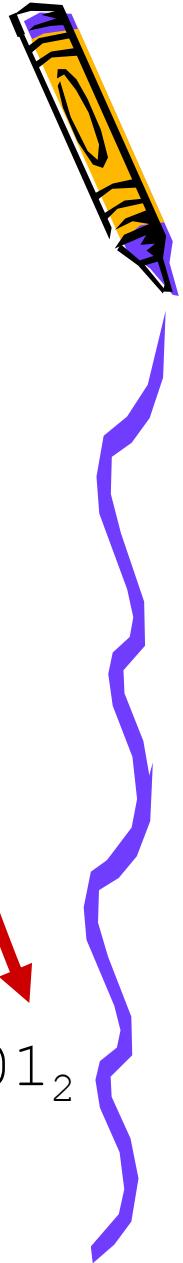


Example

$$125_{10} = ?_2$$

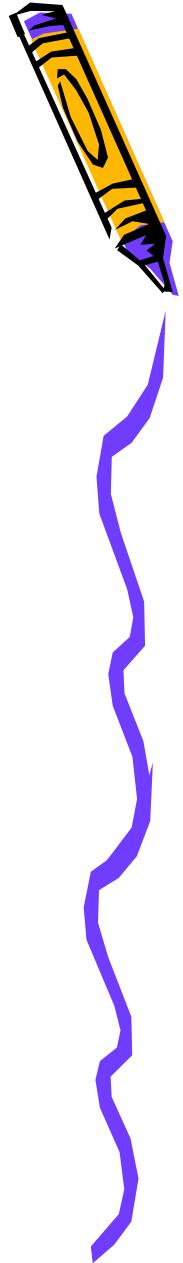
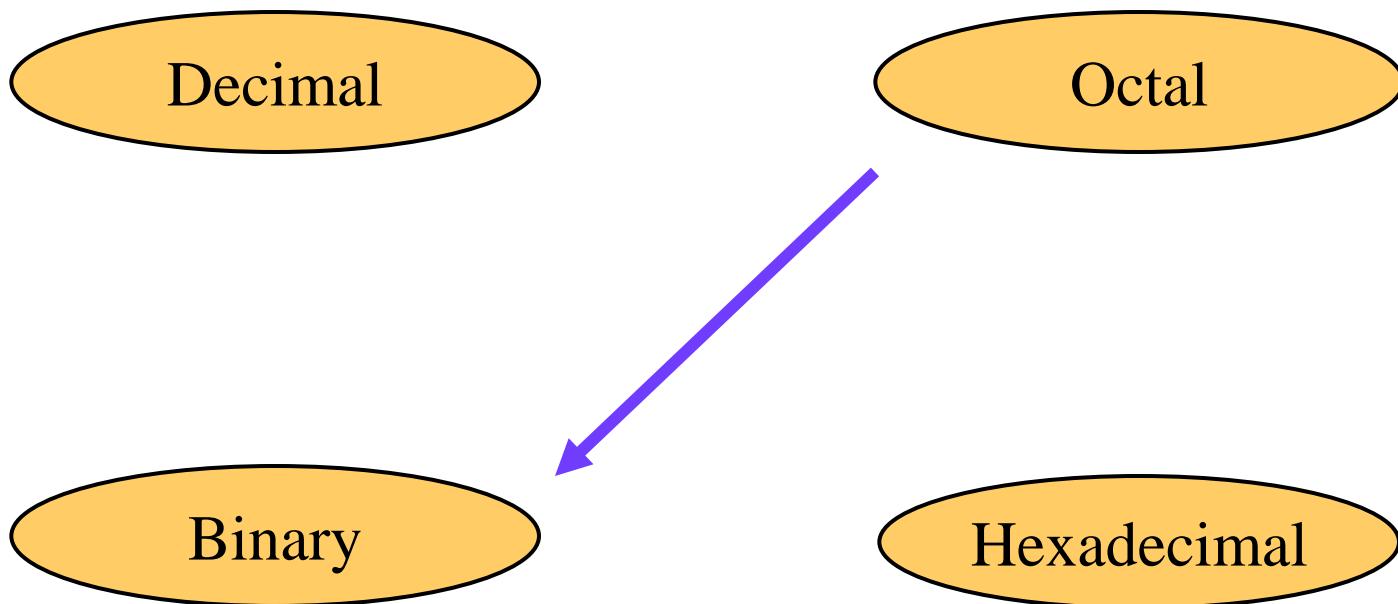
$$\begin{array}{r} 2 \longdiv{125} \\ 2 \longdiv{62} & 1 \\ 2 \longdiv{31} & 0 \\ 2 \longdiv{15} & 1 \\ 2 \longdiv{7} & 1 \\ 2 \longdiv{3} & 1 \\ 2 \longdiv{1} & 1 \\ 0 & \end{array}$$

$$125_{10} = 1111101_2$$





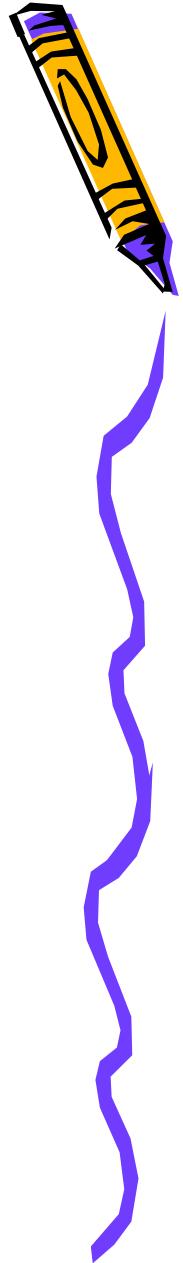
Octal to Binary





Octal to Binary

- Technique
 - Convert each octal digit to a 3-bit equivalent binary representation



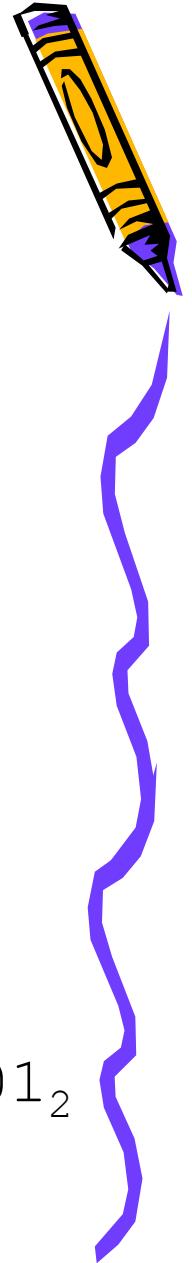


Example

$$705_8 = ?_2$$

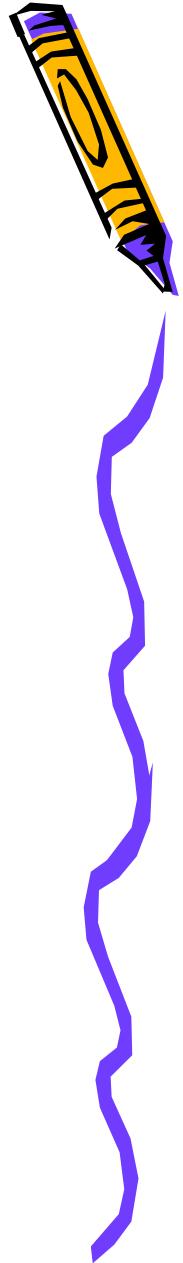
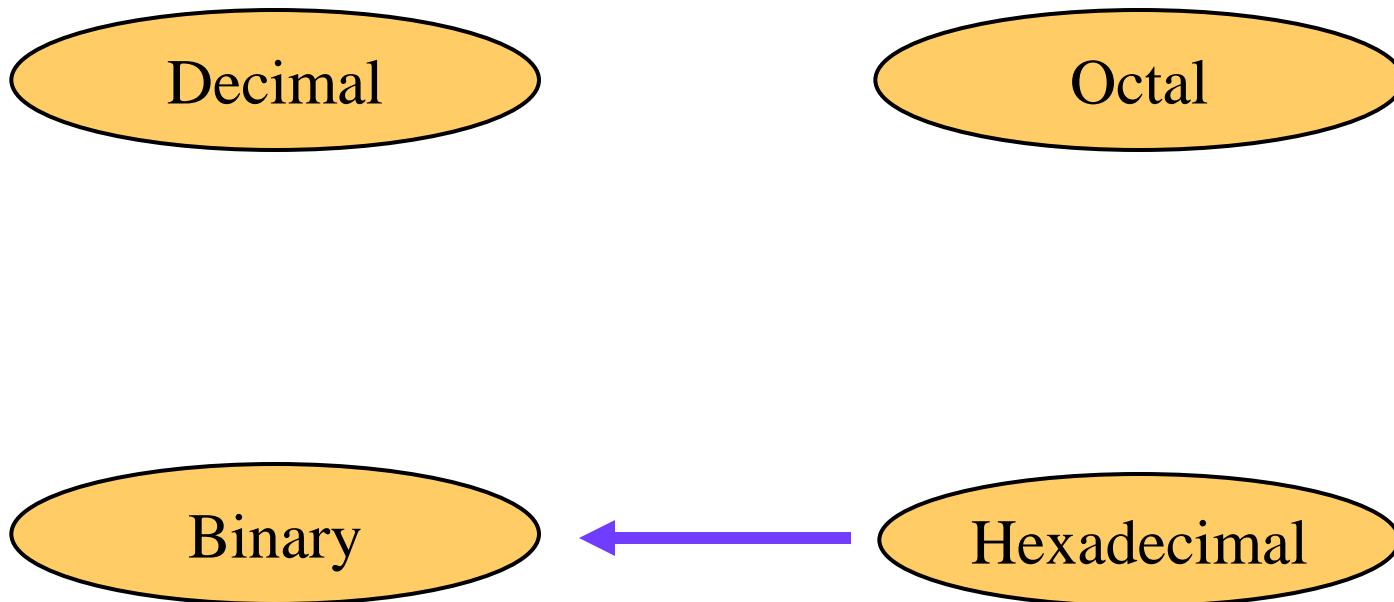
7 0 5
↓ ↓ ↓
111 000 101

$$705_8 = 111000101_2$$





Hexadecimal to Binary





Hexadecimal to Binary



- Technique
 - Convert each hexadecimal digit to a 4-bit equivalent binary representation



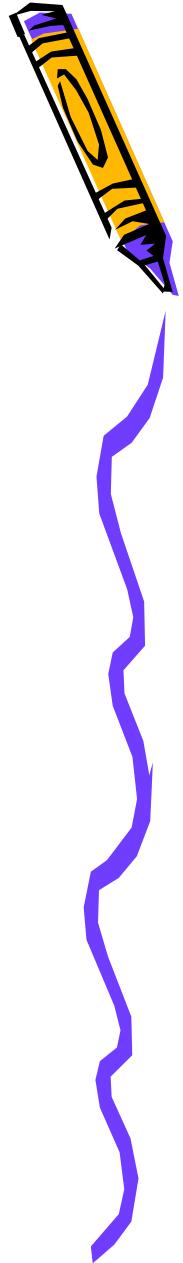


Example

$$10AF_{16} = ?_2$$

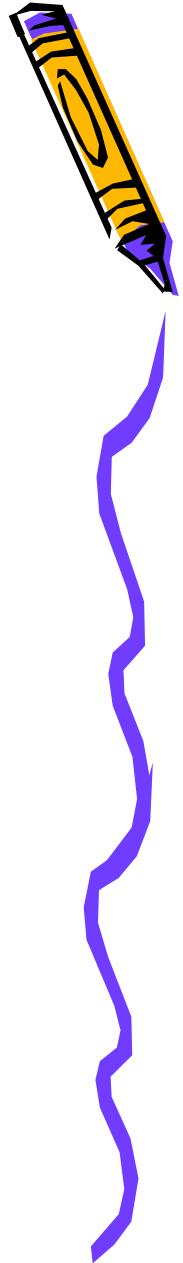
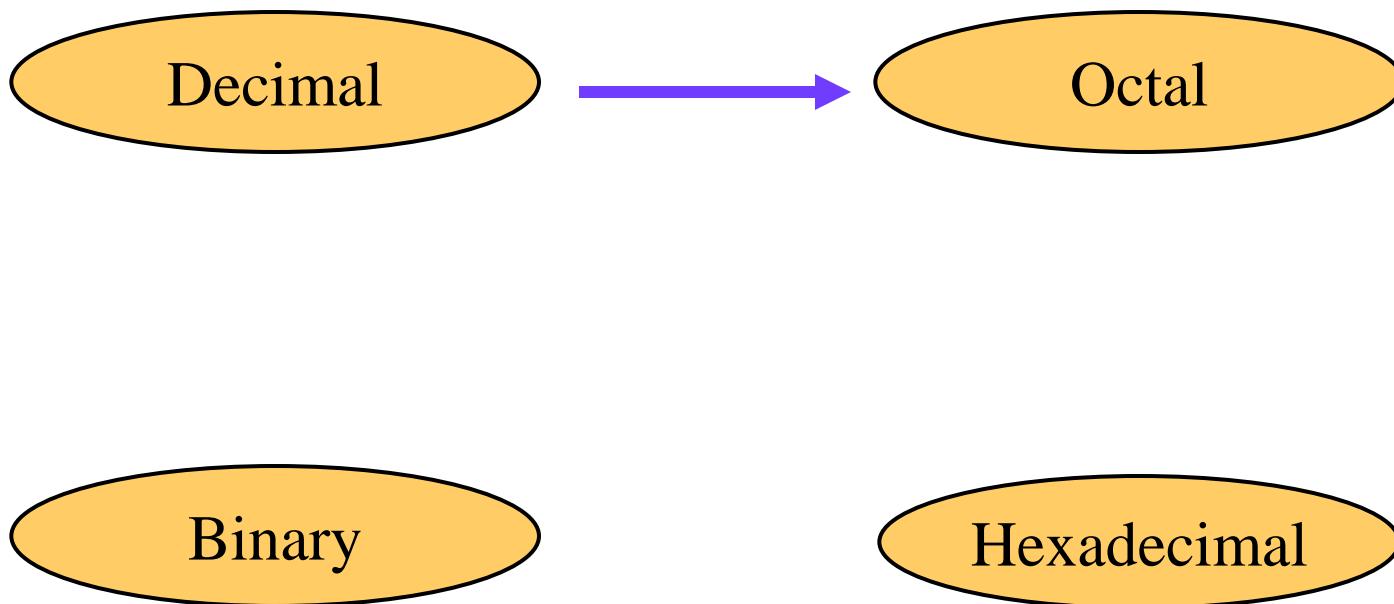
1 0 A F
↓ ↓ ↓ ↓
0001 0000 1010 1111

$$10AF_{16} = 0001000010101111_2$$





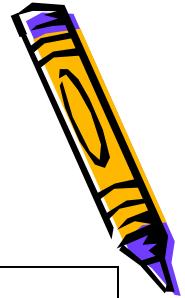
Decimal to Octal





Decimal to Octal

- Technique
 - Divide by 8
 - Keep track of the remainder



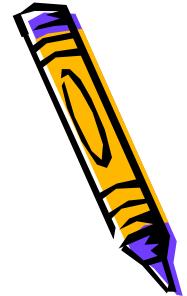


Example

$$1234_{10} = ?_8$$

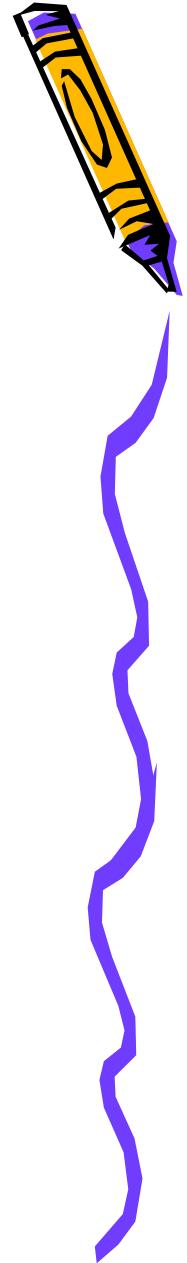
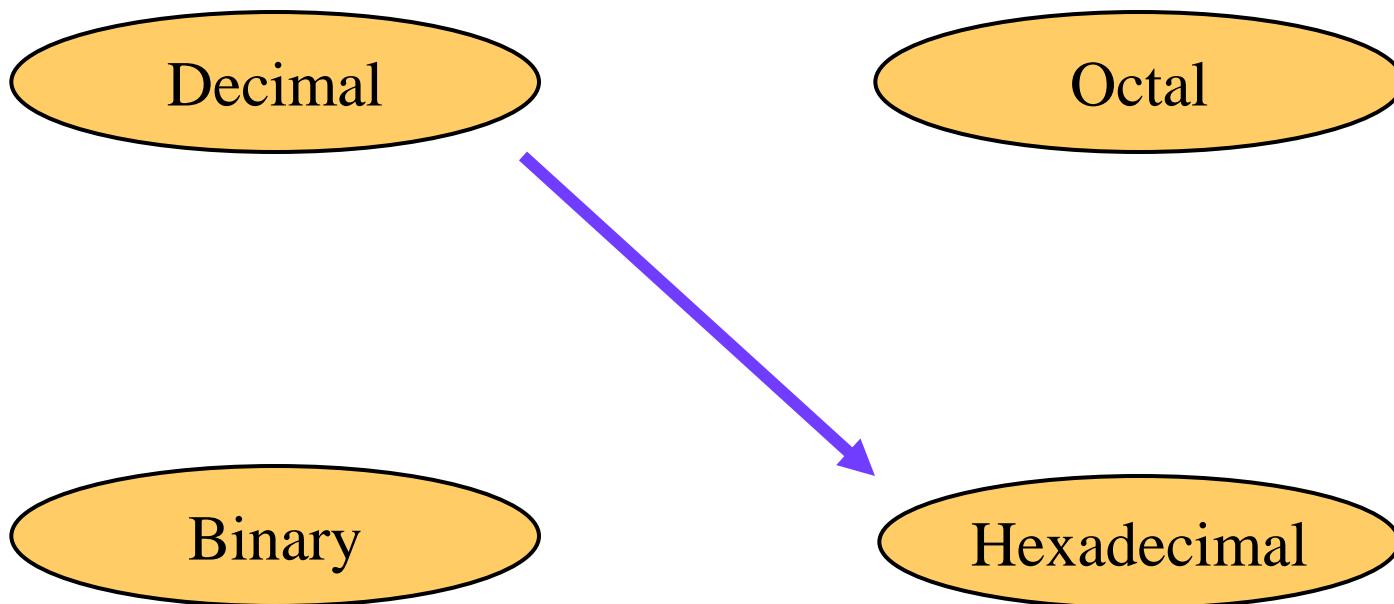
$$\begin{array}{r} 1234 \\ \hline 8 | 154 \\ \hline 8 | 19 \\ \hline 8 | 2 \\ \hline 0 \end{array} \quad \begin{array}{r} 2 \\ 2 \\ 3 \\ 2 \end{array}$$

$$1234_{10} = 2322_8$$



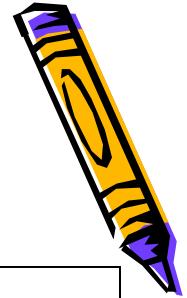


Decimal to Hexadecimal





Decimal to Hexadecimal



- Technique
 - Divide by 16
 - Keep track of the remainder



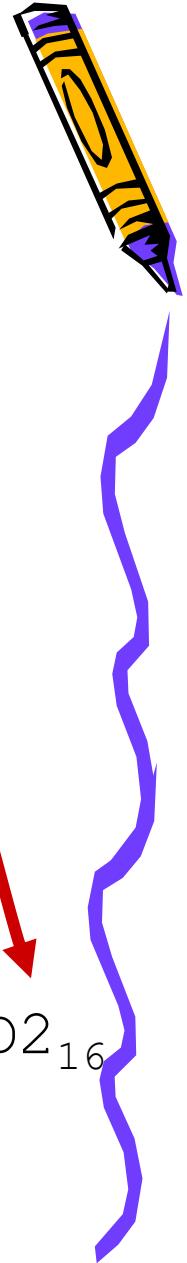


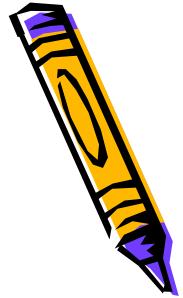
Example

$$1234_{10} = ?_{16}$$

$$\begin{array}{r} 16 \longdiv{1234} \\ 16 \quad \quad \quad 2 \\ \hline 77 \\ 16 \quad \quad \quad 13 = D \\ \hline 4 \\ 16 \quad \quad \quad 4 \\ \hline 0 \end{array}$$

$$1234_{10} = 4D2_{16}$$



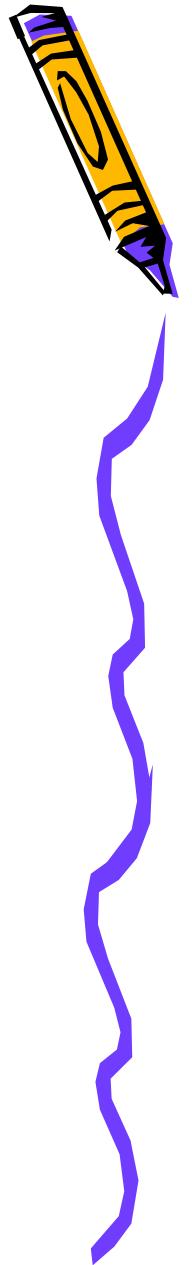


End of the first week

H A K

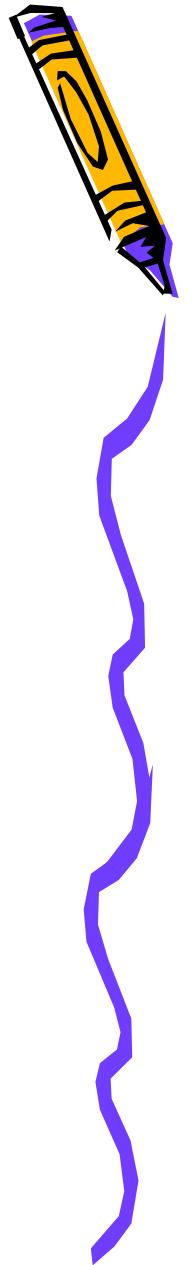
T N S

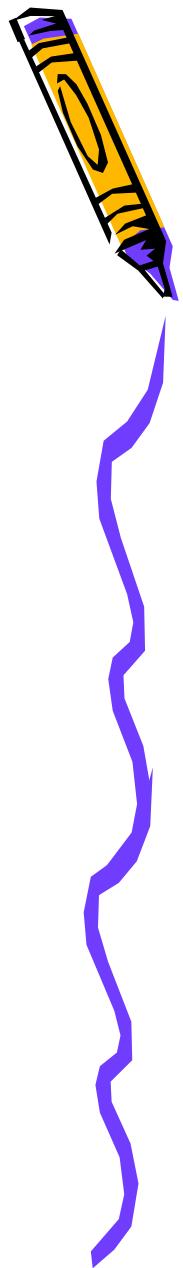


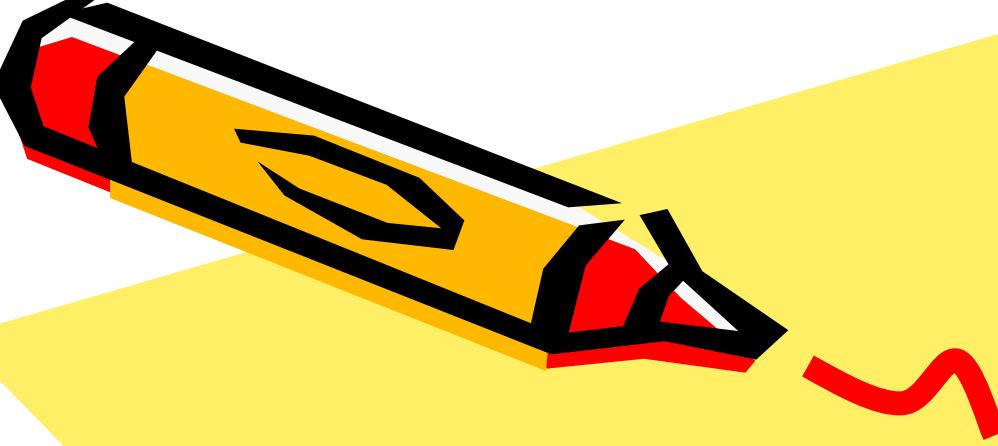


Thank you







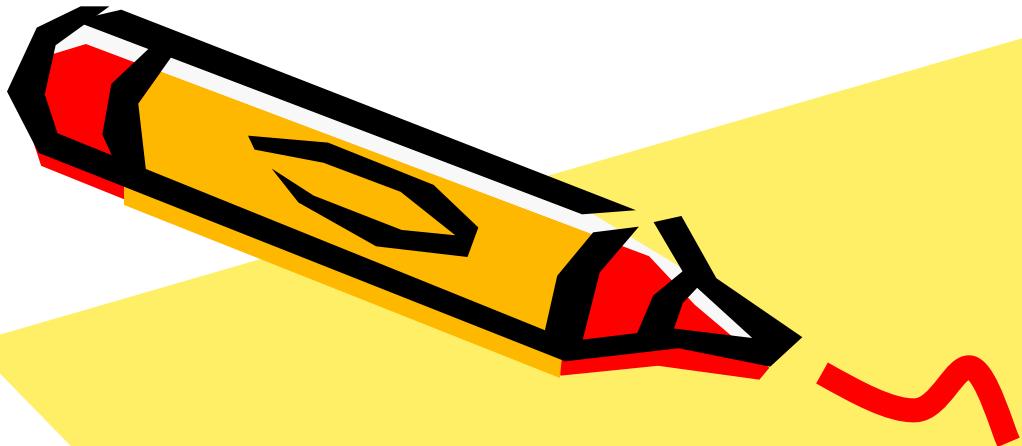


Logical Design

CS 221

Prof.Dr. Mohamed Osama Khozium





2. Number Systems

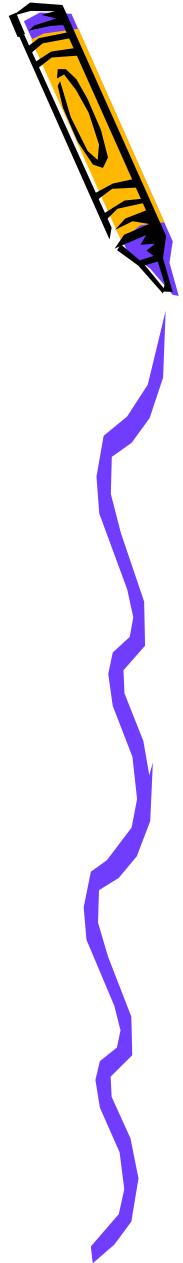
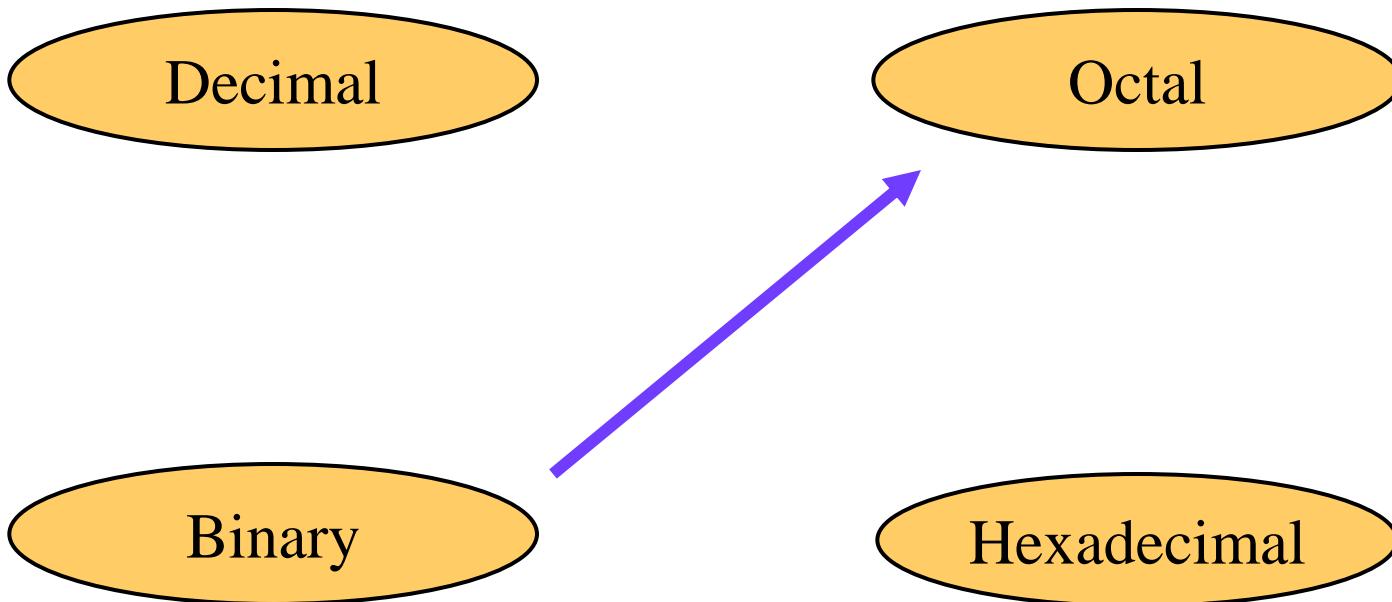
Location in
course textbook



Chapt. 1

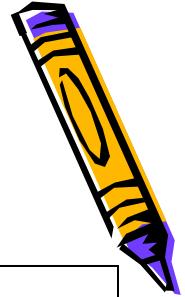


Binary to Octal





Binary to Octal



- Technique
 - Group bits in threes, starting on right
 - Convert to octal digits



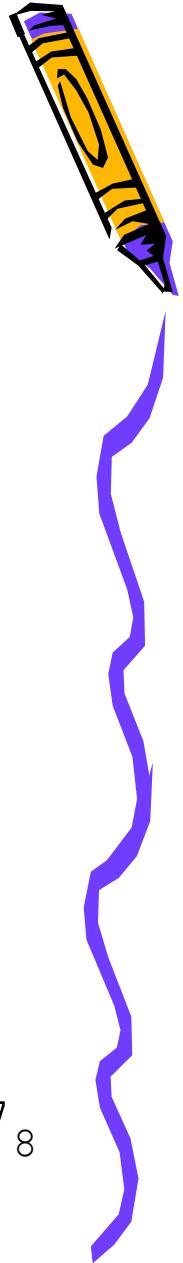


Example

$$1011010111_2 = ?_8$$

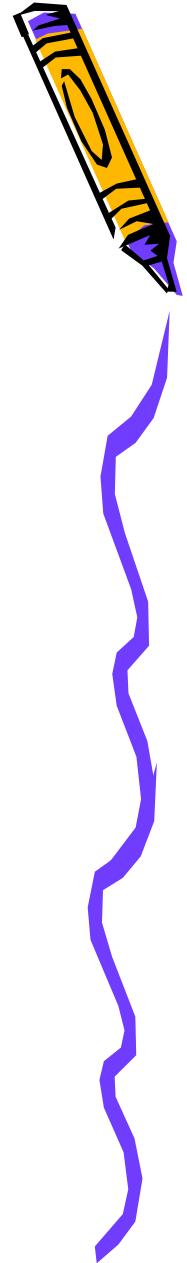
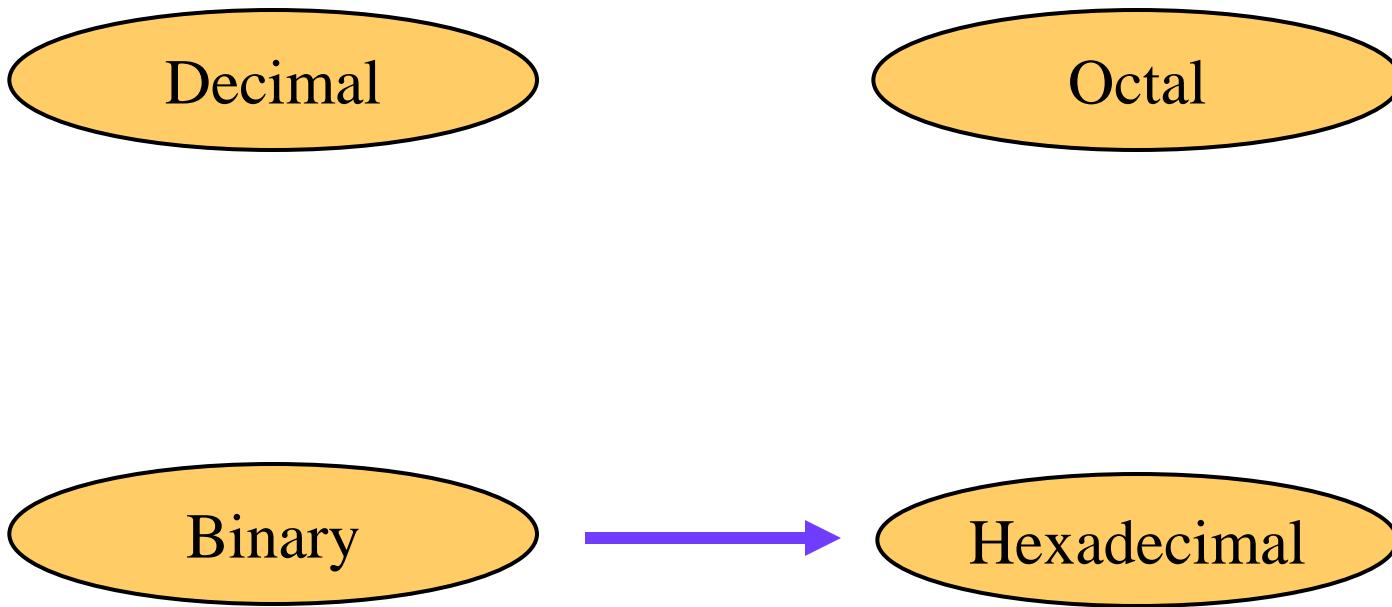
1 011 010 111
↓ ↓ ↓ ↓
1 3 2 7

$$1011010111_2 = 1327_8$$



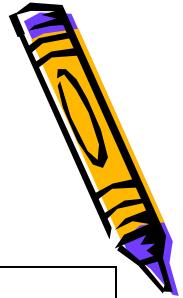


Binary to Hexadecimal





Binary to Hexadecimal



- Technique
 - Group bits in fours, starting on right
 - Convert to hexadecimal digits



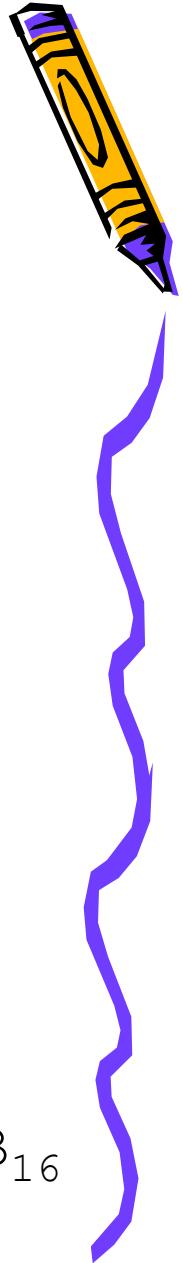


Example

$$1010111011_2 = ?_{16}$$

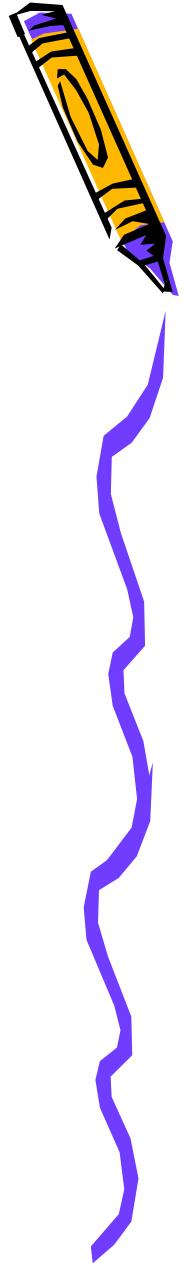
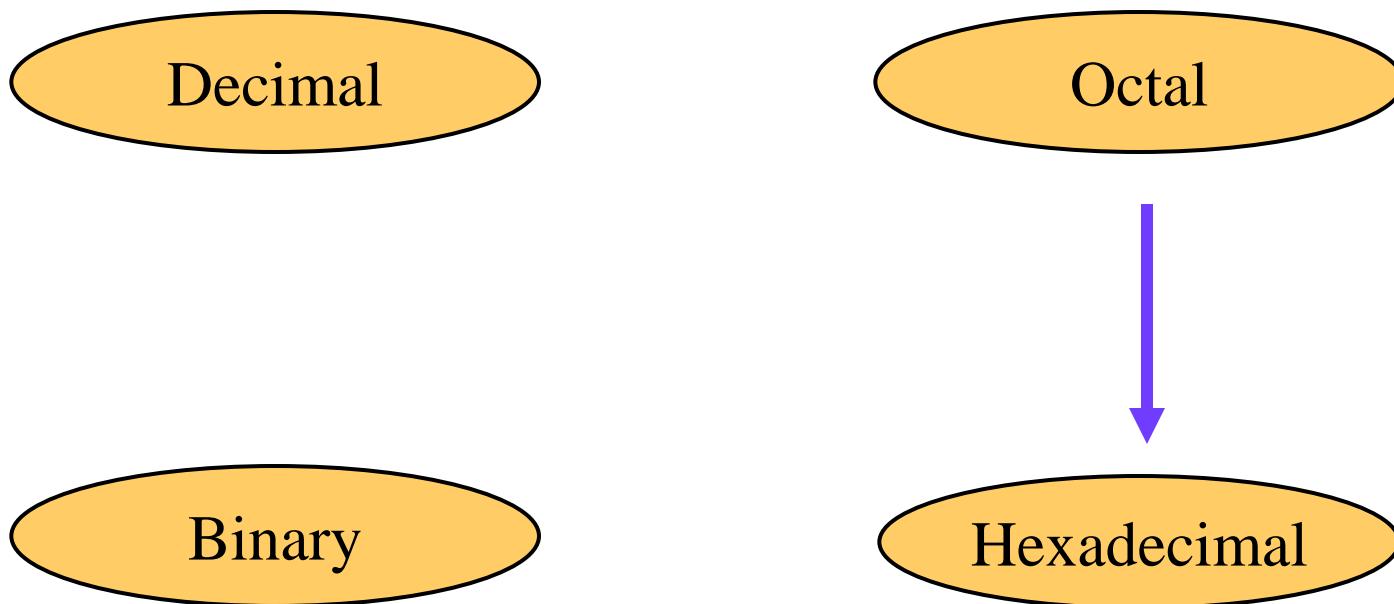
10 1011 1011
↓ ↓ ↓
2 B B

$$1010111011_2 = 2BB_{16}$$





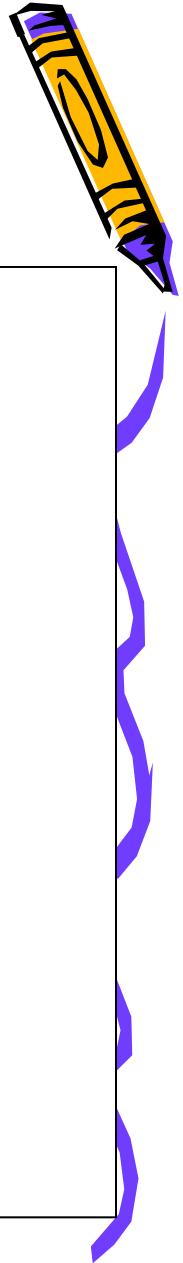
Octal to Hexadecimal





Octal to Hexadecimal

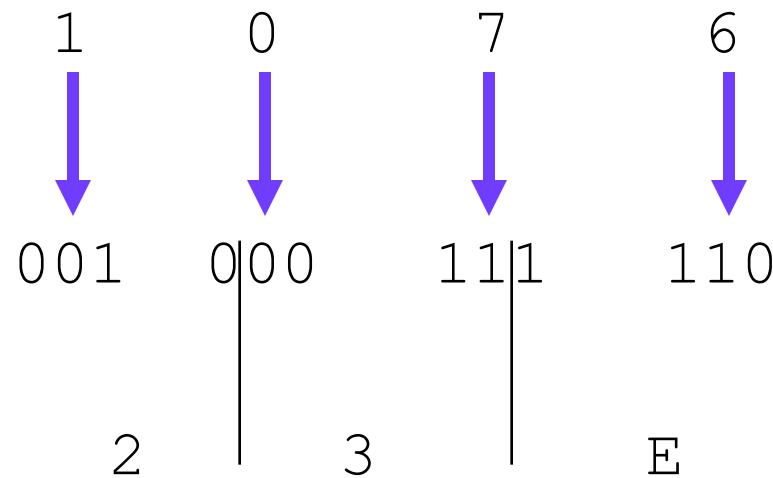
- Technique
 - Use binary as an intermediary



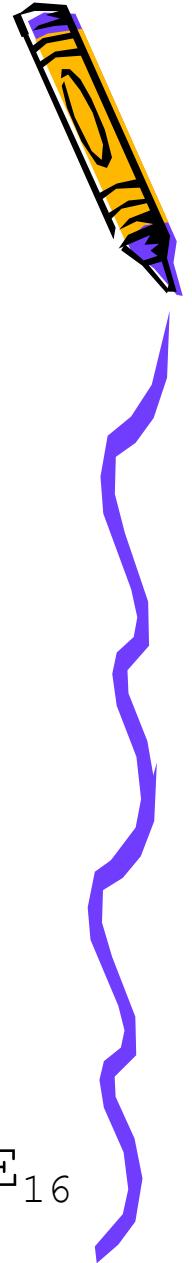


Example

$$1076_8 = ?_{16}$$

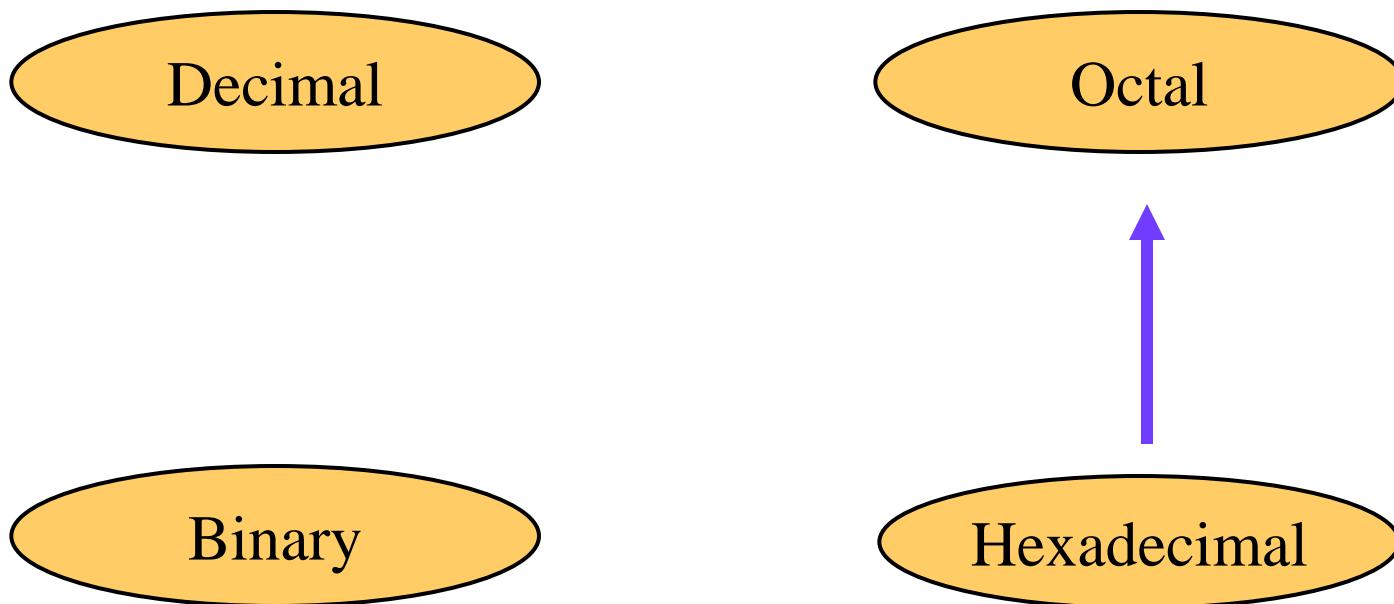


$$1076_8 = 23E_{16}$$



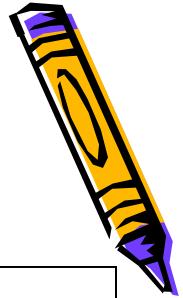


Hexadecimal to Octal





Hexadecimal to Octal



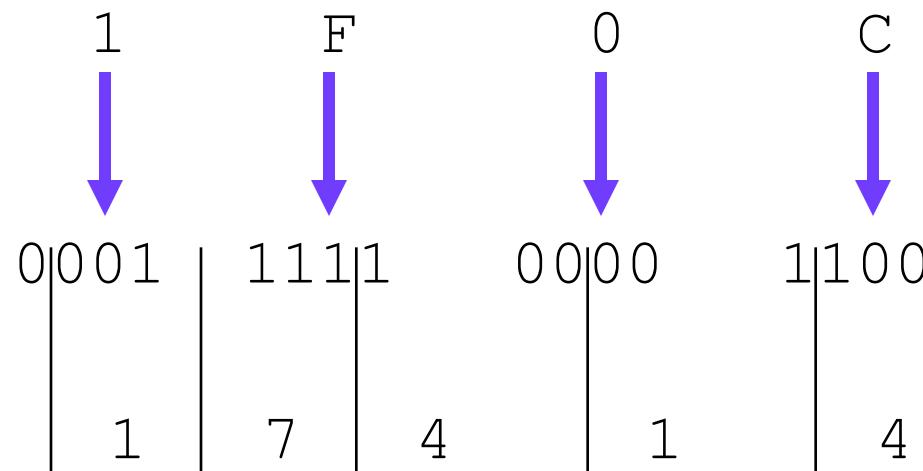
- Technique
 - Use binary as an intermediary





Example

$$1F0C_{16} = ?_8$$

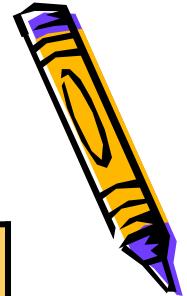


$$1F0C_{16} = 17414_8$$





Exercise – Convert ...



Decimal	Binary	Octal	Hexa-decimal
33			
	1110101		
		703	
			1AF

Don't use a calculator!

Skip answer

Answer





Exercise – Convert ...

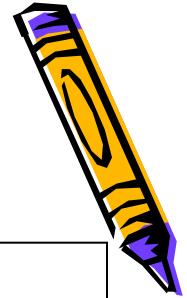
Answer

Decim al	Binary	Octal	Hexa-decimal
33	100001	41	21
117	1110101	165	75
451	111000011	703	1C3
431	110101111	657	1AF





Common Powers (1 of 2)



- Base 10

Power	Preface	Symbol	Value
10^{-12}	pico	p	.00000000001
10^{-9}	nano	n	.000000001
10^{-6}	micro	μ	.000001
10^{-3}	milli	m	.001
10^3	kilo	k	1000
10^6	mega	M	1000000
10^9	giga	G	1000000000
10^{12}	tera	T	1000000000000



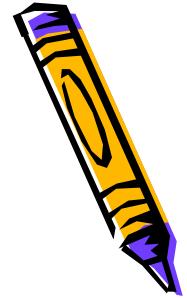


Common Powers (2 of 2)

- Base 2

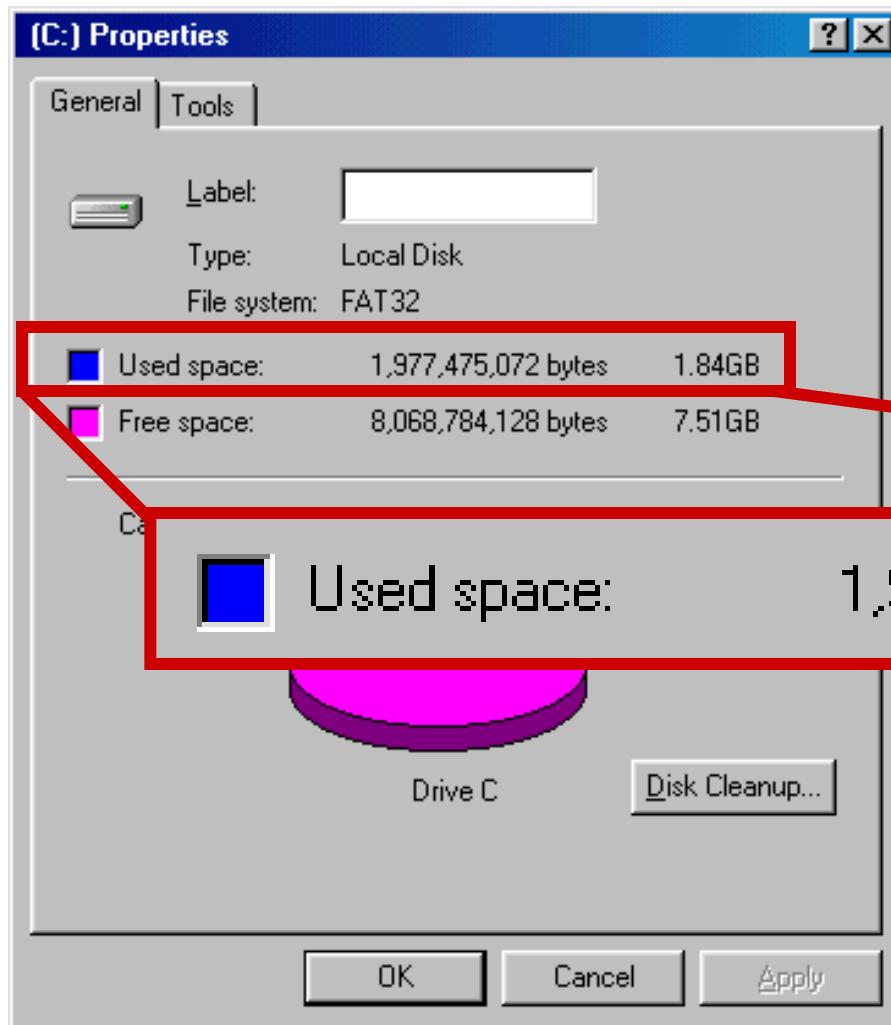
Power	Preface	Symbol	Value
2^{10}	kilo	k	1024
2^{20}	mega	M	1048576
2^{30}	Giga	G	1073741824

- What is the value of “k”, “M”, and “G”?
- In computing, particularly w.r.t. memory, the base-2 interpretation generally applies





Example

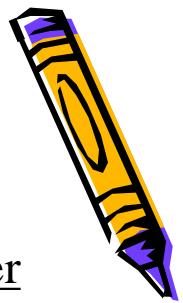


In the lab...

1. Double click on My Computer
2. Right click on C:
3. Click on Properties

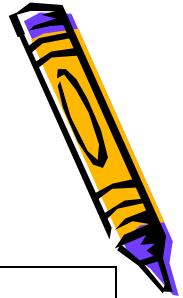
Used space: 1,977,475,072 bytes 1.84GB

$$/ 2^{30} =$$





Exercise – Free Space



- Determine the “free space” on all drives on a machine in the lab

Drive	Free space	
	Bytes	GB
A:		
C:		
D:		
E:		
etc.		





Review – multiplying powers



- For common bases, add powers

$$a^b \times a^c = a^{b+c}$$

$$2^6 \times 2^{10} = 2^{16} = 65,536$$

or...

$$2^6 \times 2^{10} = 64 \times 2^{10} = 64k$$



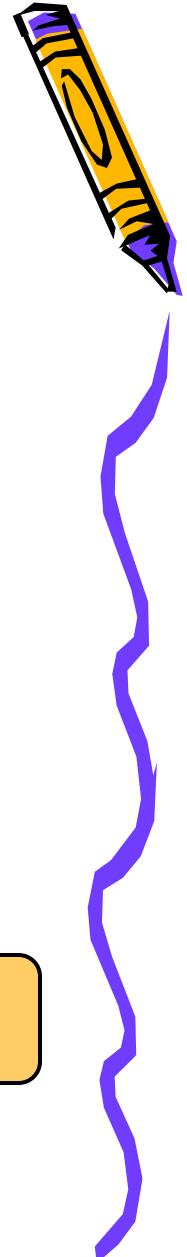


Binary Addition (1 of 2)

- Two 1-bit values

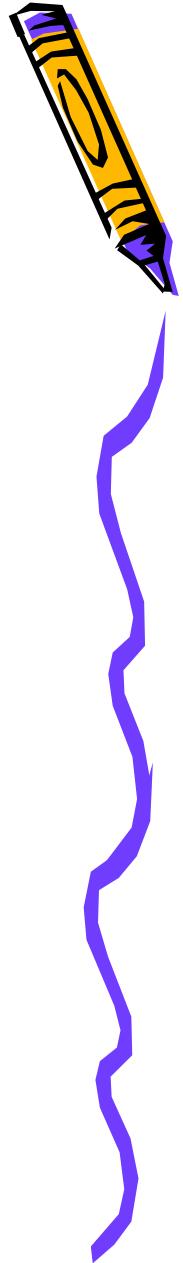
A	B	$A + B$
0	0	0
0	1	1
1	0	1
1	1	10

“two”





Binary Addition (2 of 2)



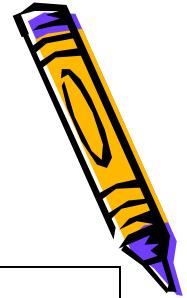
- Two n -bit values
 - Add individual bits
 - Propagate carries
 - E.g.,

$$\begin{array}{r} & \overset{1}{} & \overset{1}{} \\ & 10101 & \\ + & 11001 & \\ \hline & 101110 & \end{array} \qquad \begin{array}{r} & 21 \\ + & 25 \\ \hline & 46 \end{array}$$





Multiplication (1 of 3)



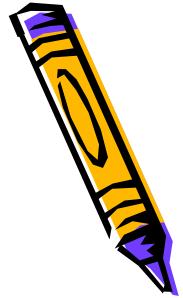
- Decimal (just for fun)

$$\begin{array}{r} 35 \\ \times 105 \\ \hline 175 \\ 000 \\ \hline 35 \\ \hline 3675 \end{array}$$





Multiplication (2 of 3)



- Binary, two 1-bit values

A	B	$A \times B$
0	0	0
0	1	0
1	0	0
1	1	1





Multiplication (3 of 3)



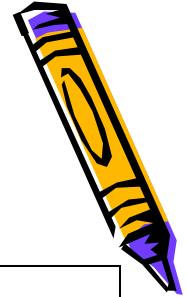
- Binary, two n -bit values
 - As with decimal values
 - E.g.,

$$\begin{array}{r} 1110 \\ \times 1011 \\ \hline 1110 \\ 1110 \\ 0000 \\ \hline 10011010 \end{array}$$





Fractions



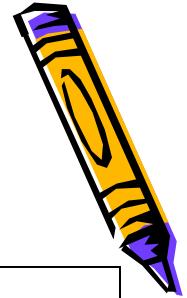
- Decimal to decimal (just for fun)

$$\begin{array}{rcl} 3.14 \Rightarrow & 4 \times 10^{-2} = 0.04 \\ & 1 \times 10^{-1} = 0.1 \\ & 3 \times 10^0 = \underline{\underline{3}} \\ & & 3.14 \end{array}$$





Fractions



- Binary to decimal

10.1011 =>

$$1 \times 2^{-4} = 0.0625$$

$$1 \times 2^{-3} = 0.125$$

$$0 \times 2^{-2} = 0.0$$

$$1 \times 2^{-1} = 0.5$$

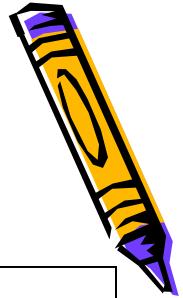
$$0 \times 2^0 = 0.0$$

$$\begin{array}{r} 1 \times 2^1 = \underline{2.0} \\ 2.6875 \end{array}$$





Fractions



- Decimal to binary

3.14579

11.001001...

.14579

$$\begin{array}{r} \times \\ 0.29158 \end{array} \quad 2$$

$$\begin{array}{r} \times \\ 0.58316 \end{array} \quad 2$$

$$\begin{array}{r} \times \\ 1.16632 \end{array} \quad 2$$

$$\begin{array}{r} \times \\ 0.33264 \end{array} \quad 2$$

$$\begin{array}{r} \times \\ 0.66528 \end{array} \quad 2$$

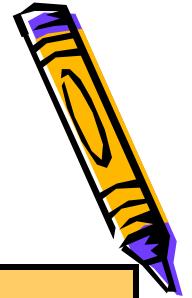
$$\begin{array}{r} \times \\ 1.33056 \end{array} \quad 2$$

etc.





Exercise – Convert ...



Decimal	Binary	Octal	Hexa-decimal
29.8			
	101.1101		
		3.07	
			C.82

Don't use a calculator!

Skip answer

Answer





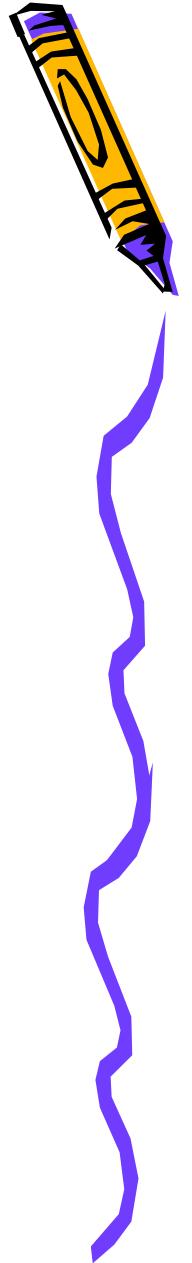
Exercise – Convert ...



Answer

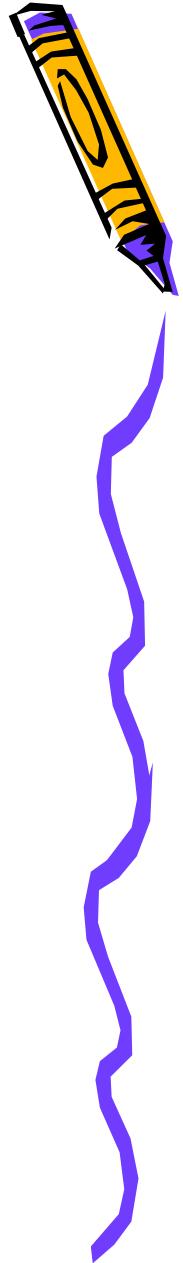
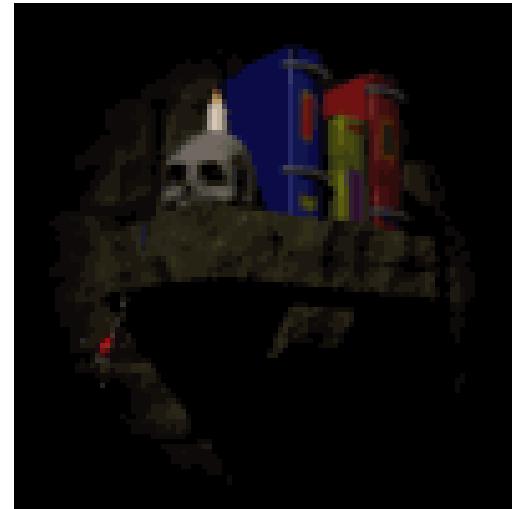
Decimal	Binary	Octal	Hexa-decimal
29.8	11101.110011...	35.63...	1D.CC...
5.8125	101.1101	5.64	5.D
3.109375	11.000111	3.07	3.1C
12.5078125	1100.10000010	14.404	C.82

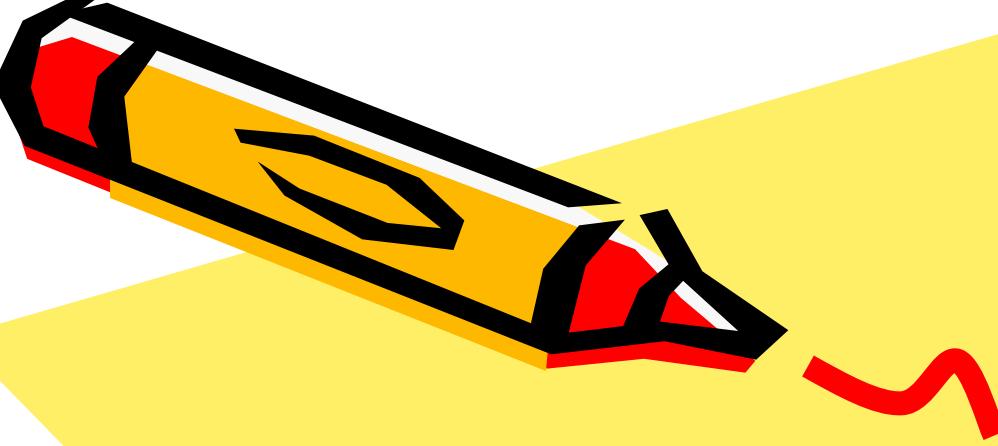




Thank you





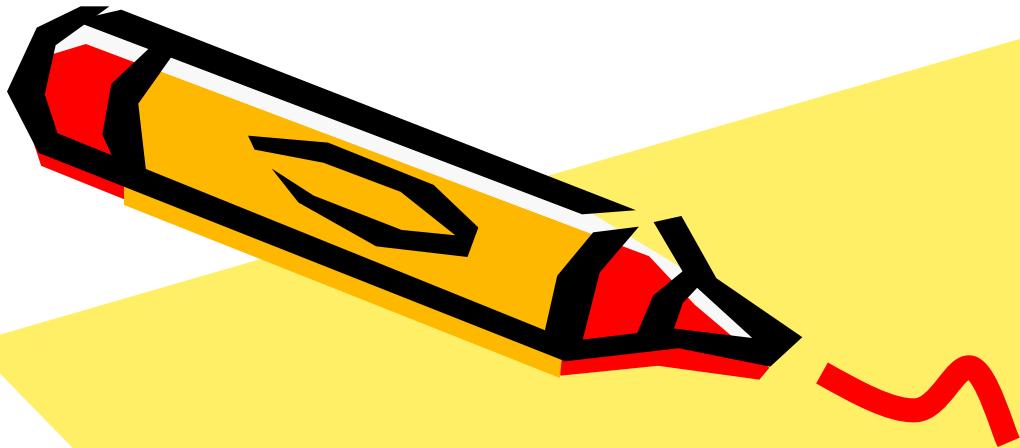


Logical Design

CS 221

Prof.Dr. Mohamed Osama Khozium





3. Number Systems

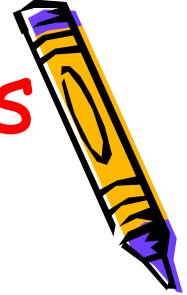
Location in
course textbook



Chapt. 1



Representation of Signed Numbers



- Positive number representation same in most systems
- Major differences are in how negative numbers are represented
- Three major schemes:
 - sign and magnitude
 - ones complement
 - twos complement





Negative Number Representation

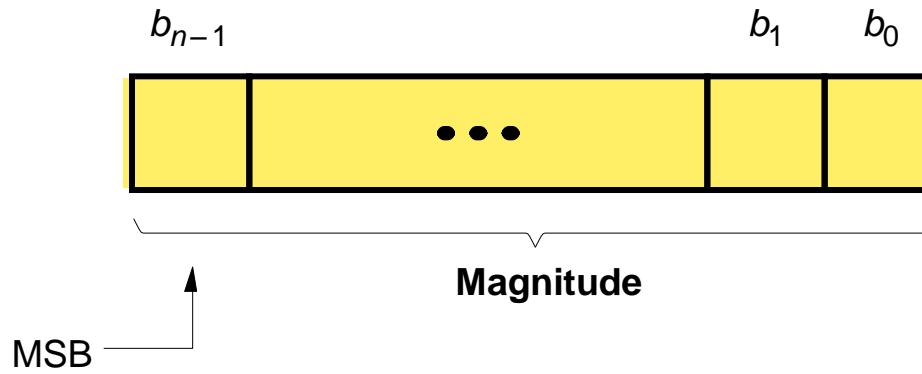


- Assumptions:
 - we'll assume a 4 bit machine word
 - 16 different values can be represented
 - roughly half are positive, half are negative
 - sign bit is the MSB; 0 = plus, 1 = minus

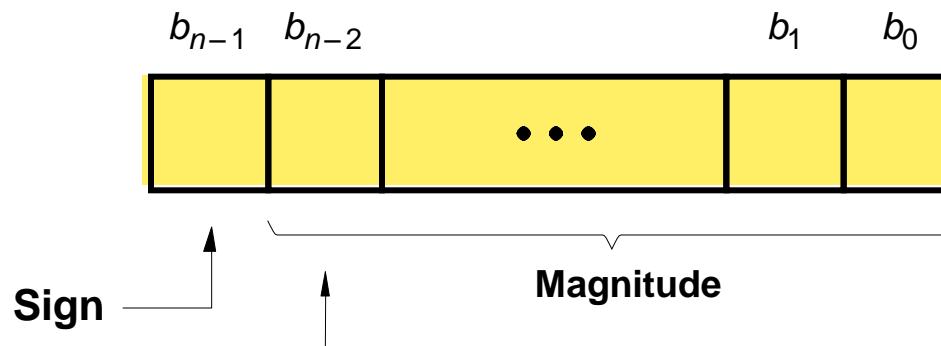




Representation of Negative Numbers

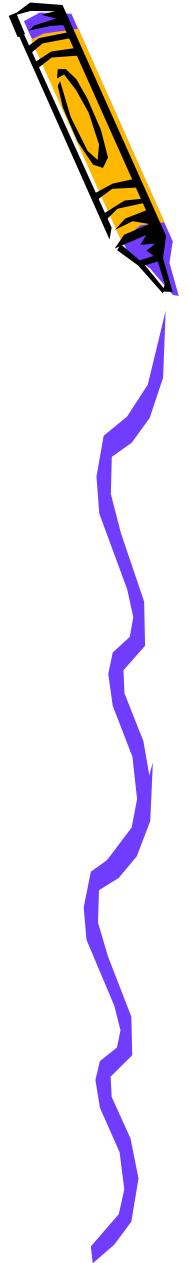


(a) Unsigned number



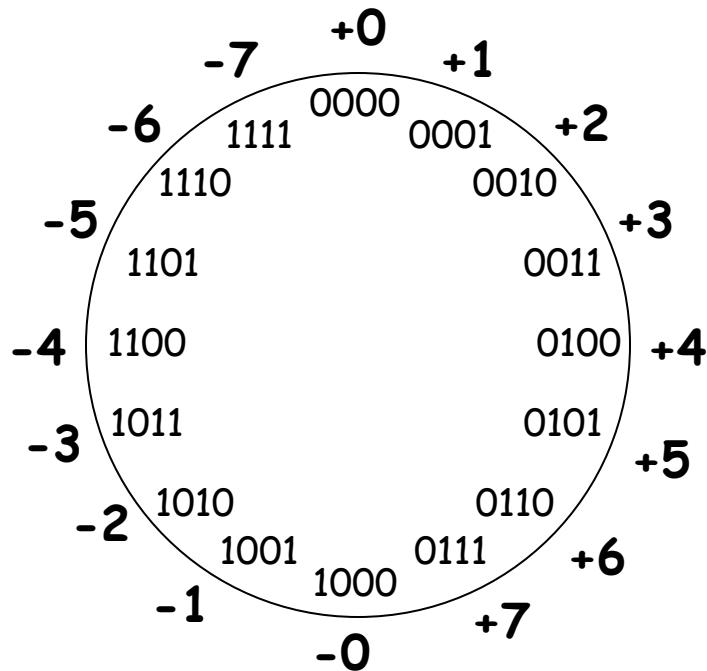
0 denotes +
1 denotes -

(b) Signed number





Sign-Magnitude Representation



High order bit is sign: 0 = positive (or zero), 1 = negative

Three low order bits is the magnitude: 0 (000) thru 7 (111)

Number range for n bits = $\pm 2^{n-1} - 1$

Two representations for 0

The major disadvantage is that we need separate circuits to both add and subtract

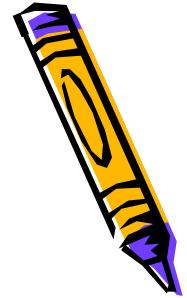
$$0\ 100 = +4$$

$$1\ 100 = -4$$

Number magnitudes need to be compared to get the right result



Subtraction of Numbers using Ones Complement



Example :

Subtract 1 from 1000 (binary sys.)

$$\begin{array}{r} 1000 \text{ (8)} \\ - 1 \text{ (1)} \\ \hline \end{array}$$

1- Complete the subtrahend by zeros to be the same digits like the minuend
 $1 \rightarrow 0001$

2 - Get the ones complement for the subtrahend
 $0001 \rightarrow 1110$

3 - Add the ones complement to minuend

$$\begin{array}{r} 1000 \\ + 1110 \\ \hline 10110 \end{array}$$

If there is a carry then add 1 and neglect the carry ...then $10110 \rightarrow 0111(7)$
If there is no carry then get the 1's complement and add (-) it will be negative





Subtraction of Numbers using Ones Complement

Another Example :

1010100 (84)

- 1000011 (67)

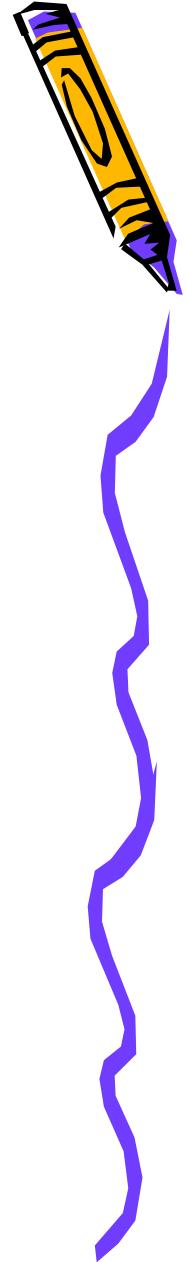
- Get the ones complement for the subtrahend

1000011 → 0111100

- Add the ones complement to minuend

$$\begin{array}{r} 1010100 \\ + 0111100 \\ \hline 10010000 \end{array}$$

- If there is a carry then add 1 and neglect the carry
- then 001000 → 001001 (17)





Subtraction of Numbers using Ones Complement

Another Example :

010011 (19)

- 010101 (21)

- Get the ones complement for the subtrahend

010101 → 101010

- Add the ones complement to minuend

010011

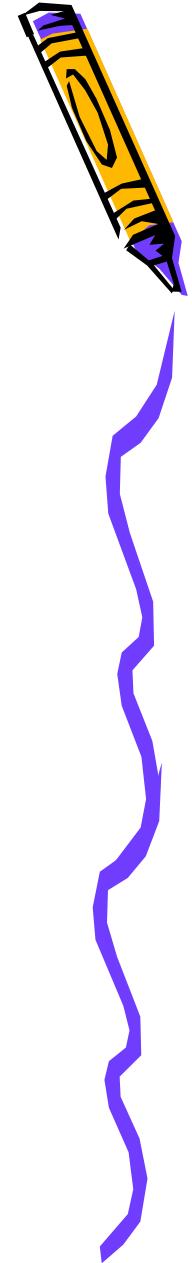
+

101010

111101

- No carry ... then get the 1's complement and put (-)

111101 → 000010 → - 000010 (- 2)





Subtraction of Numbers using Ones Complement

Another Example :

1000011 (67)

- 1010100 (84)

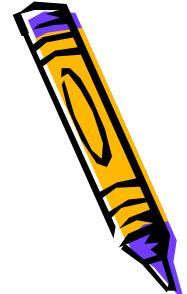
- Get the ones complement for the subtrahend

1010100 → 0101011

- Add the ones complement to minuend

$$\begin{array}{r} 1000011 \\ + 0101011 \\ \hline 1101110 \end{array}$$

- No carry ... then get the 1's complement and put (-)
1101110 → 0010001 → - 0010001 (- 17)



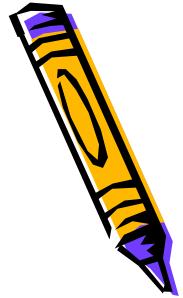


Twos Complement

- Most common scheme of representing negative numbers in computers
- Affords natural arithmetic (no special rules!)
- To represent a negative number in 2's complement notation...
 1. Decide upon the number of bits (n)
 2. Find the binary representation of the +ve value in n -bits
 3. Flip all the bits (change 1's to 0's and vice versa)
 4. Add 1

Shortcut ...start from right put each zero the same and the first one then change each zero by one and each one by zero

0110100 → 1001100





Twos Complement Example



- Represent -5 in binary using 2's complement notation
 1. Decide on the number of bits
6 (for example)
 2. Find the binary representation of the +ve value in 6 bits

000101

+5

3. Flip all the bits

111010

4. Add 1

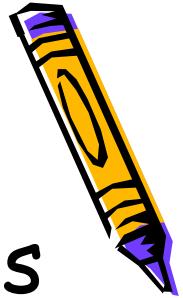
$$\begin{array}{r} 111010 \\ + \quad \quad 1 \\ \hline 111011 \end{array}$$

-5



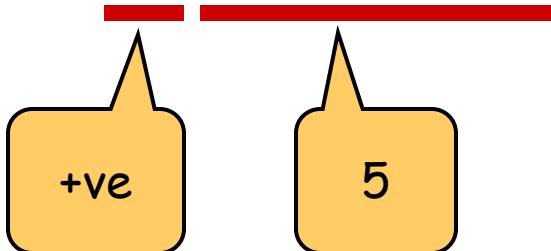


Sign Bit

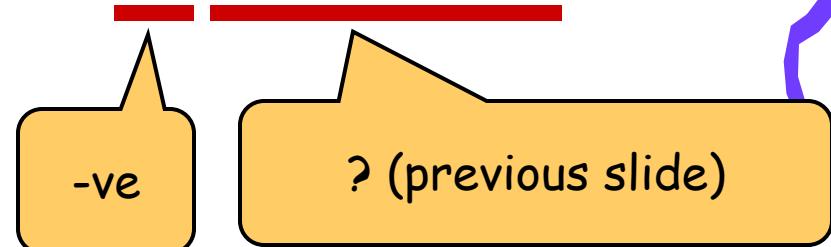


- In 2's complement notation, the MSB is the sign bit (as with sign-magnitude notation)
 - 0 = positive value
 - 1 = negative value

+5: 0 0 0 1 0 1

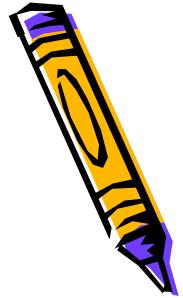


-5: 1 1 1 0 1 1

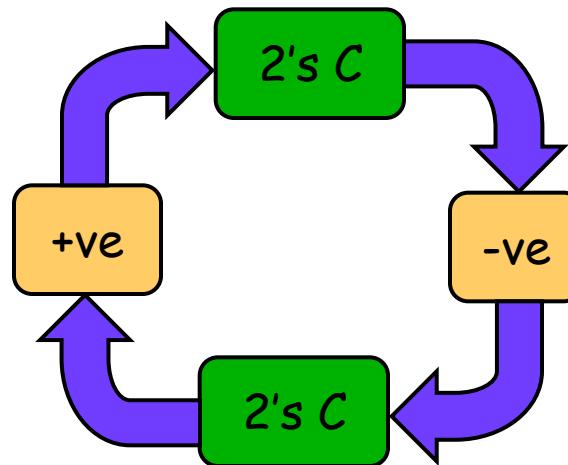




"Complementary" Notation

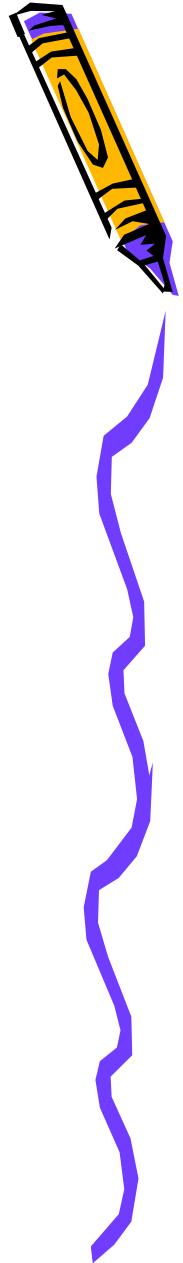
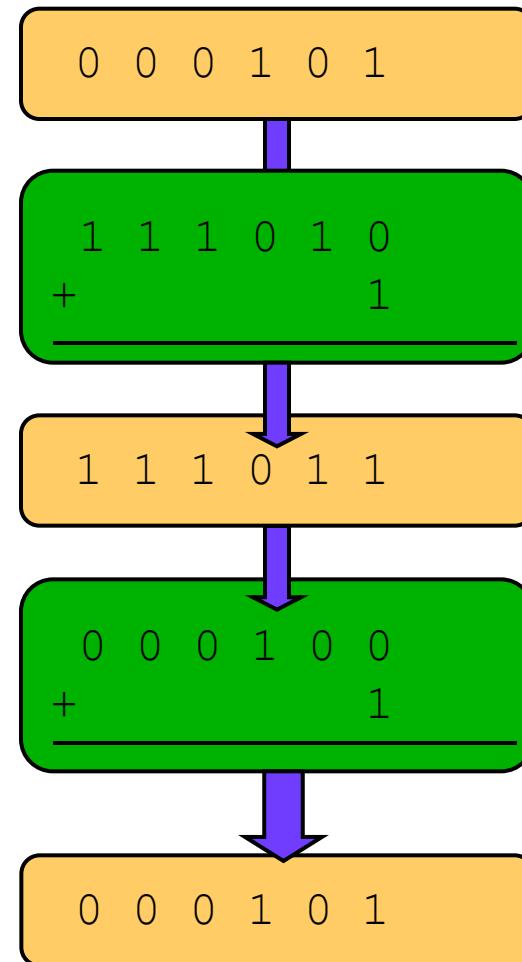
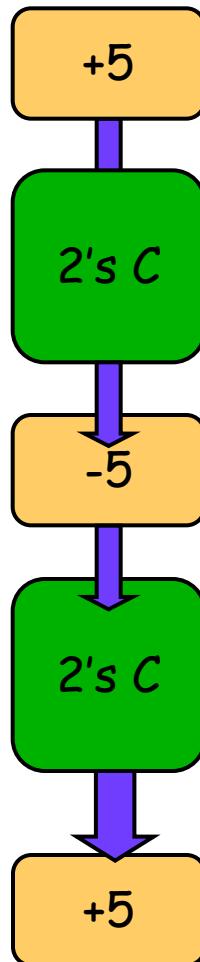


- Conversions between positive and negative numbers are easy
- For binary (base 2)...



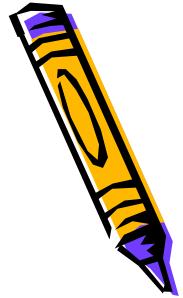


Example





Exercise - 2's C conversions



- What is -20 expressed as an 8-bit binary number in 2's complement notation?
 - Answer: _____
- 1100011 is a 7-bit binary number in 2's complement notation. What is the decimal value?
 - Answer: _____

Skip answer

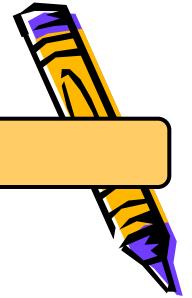
Answer





Exercise - 2's C conversions

Answer

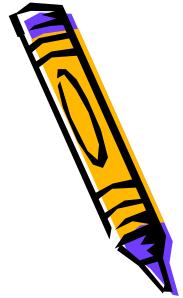


- What is -20 expressed as an 8-bit binary number in 2's complement notation?
 - Answer: 11101100
- 1100011 is a 7-bit binary number in 2's complement notation. What is the decimal value?
 - Answer: -29

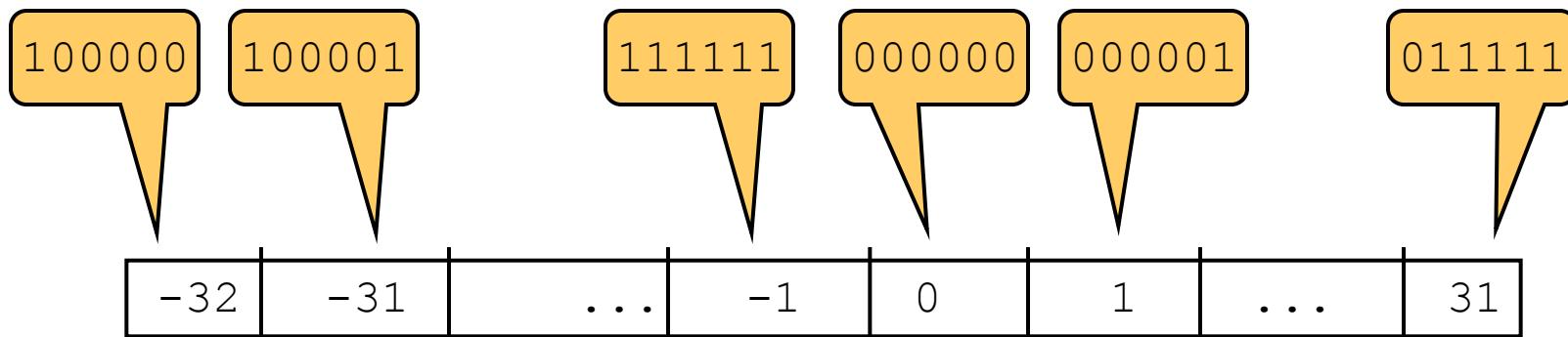




Range for 2's Complement



- For example, 6-bit 2's complement notation



Negative, sign bit = 1

Zero or positive, sign bit = 0





One's Complement subtraction

01010 (10)

-

00111 (7)

One's complement for subtrahend
then add

01010

+

11000

100010

Neglect carry and add 1

00011 (+3)

Two's Complement subtraction

01010 (10)

-

00111 (7)

Two's complement for subtrahend
then add

01010

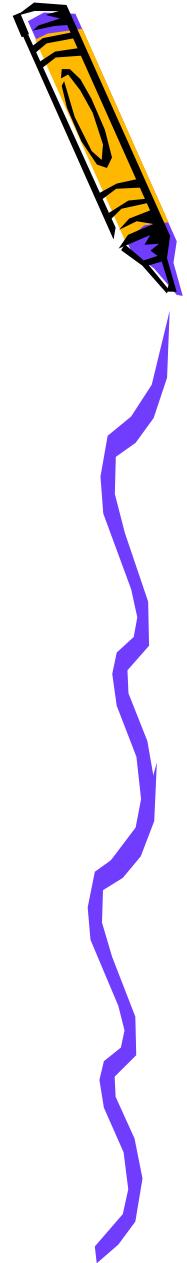
+

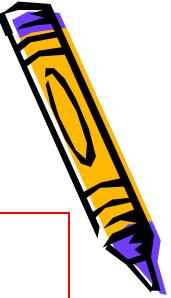
11001

100011

Neglect carry

00011 (+3)





One's Complement subtraction

00111 (7)

-

01010 (10)

One's complement for subtrahend
then add

00111

+

10101

11100

No carry and get one's
complement again...put -
-00011 (-3)

Two's Complement subtraction

00111 (7)

-

01010 (10)

Two's complement for subtrahend then
add

00111

+

10110

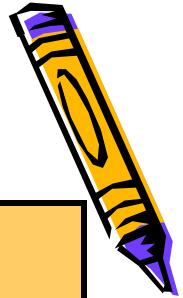
11101

No carry Two's complement again
- 00011 (-3)





Ranges (revisited)

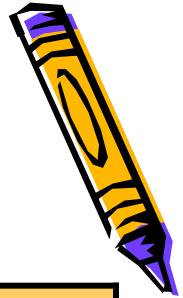


No. of bits	Binary					
	Unsigned		Sign-magnitude		2's complement	
	Min	Max	Min	Max	Min	Max
1	0	1				
2	0	3	-1	1	-2	1
3	0	7	-3	3	-4	3
4	0	15	-7	7	-8	7
5	0	31	-15	15	-16	15
6	0	63	-31	31	-32	31
Etc.						





In General (revisited)



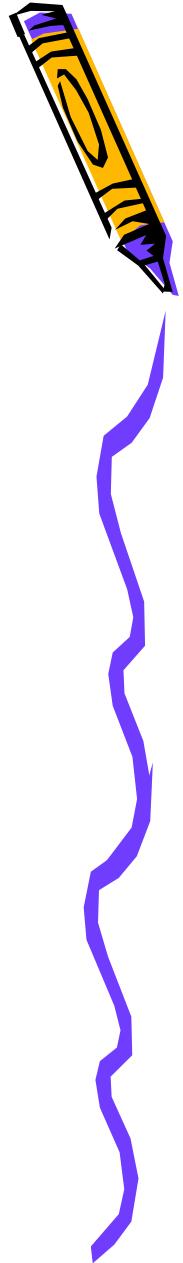
No. of bits	Binary					
	Unsigned		Sign-magnitude		2's complement	
	Min	Max	Min	Max	Min	Max
n	0	$2^n - 1$	$-(2^{n-1} - 1)$	$2^{n-1} - 1$	-2^{n-1}	$2^{n-1} - 1$





2's Complement Addition

- Easy
- No special rules
- Just add





What is -5 plus +5?

- Zero, of course, but let's see

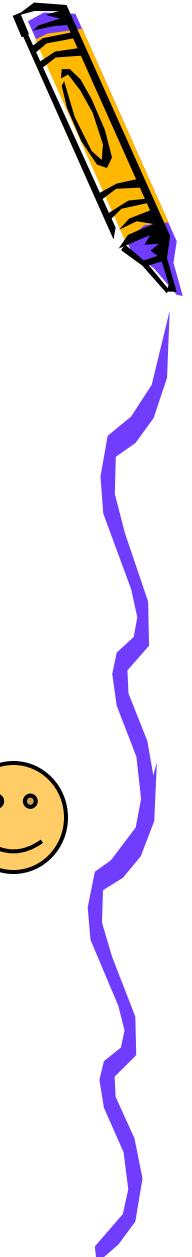
Sign-magnitude

$$\begin{array}{r} -5: \quad 10000101 \\ +5: \quad +00000101 \\ \hline 10001010 \end{array}$$



Twos-complement

$$\begin{array}{r} \text{1 1 1 1 1 1 1 1} \\ -5: \quad 11111011 \\ +5: \quad +00000101 \\ \hline 00000000 \end{array}$$





2's Complement Subtraction



- Easy
- No special rules
- Just subtract, well ... actually ... just add!

$$A - B = A + (-B)$$

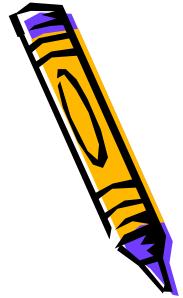
add

2's complement of B





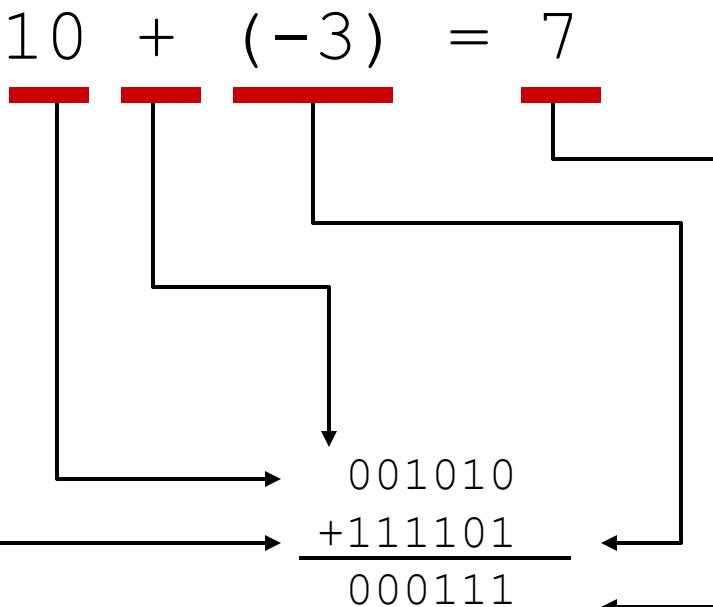
What is 10 subtract 3?



- 7, of course, but...
- Let's do it (we'll use 6-bit values)

$$10 - 3 = 10 + (-3) = 7$$

+3: 000011
1s C: 111100
+1:
-3: 111101





What is 10 subtract -3?

$$(-(-3)) = 3$$

- 13, of course, but...
- Let's do it (we'll use 6-bit values)

$$10 - (-3) = \underline{10} + \underline{(-(-3))} = 13$$

-3: 111101

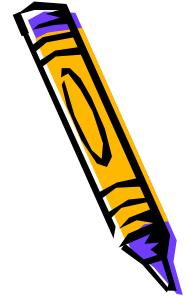
1s C: 000010

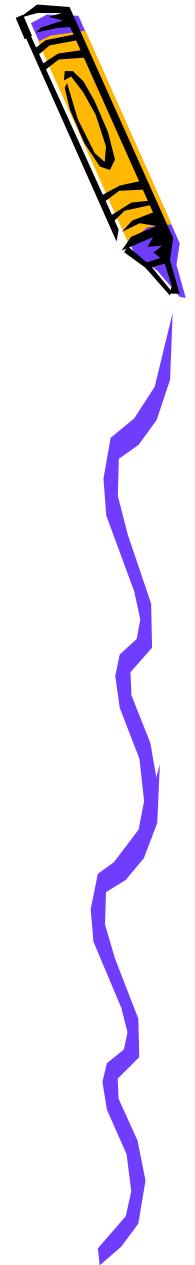
+1: 1

+3: 000011

→

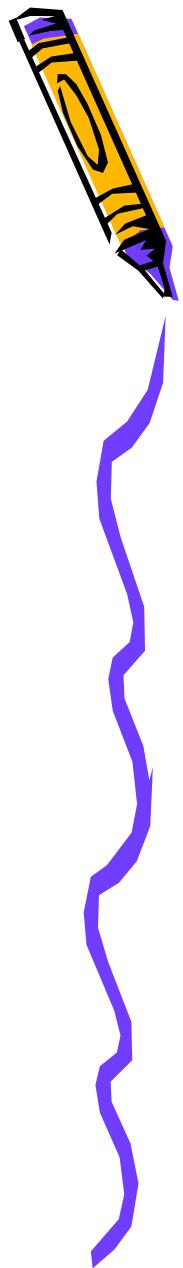
$$\begin{array}{r} 001010 \\ +000011 \\ \hline 001101 \end{array}$$

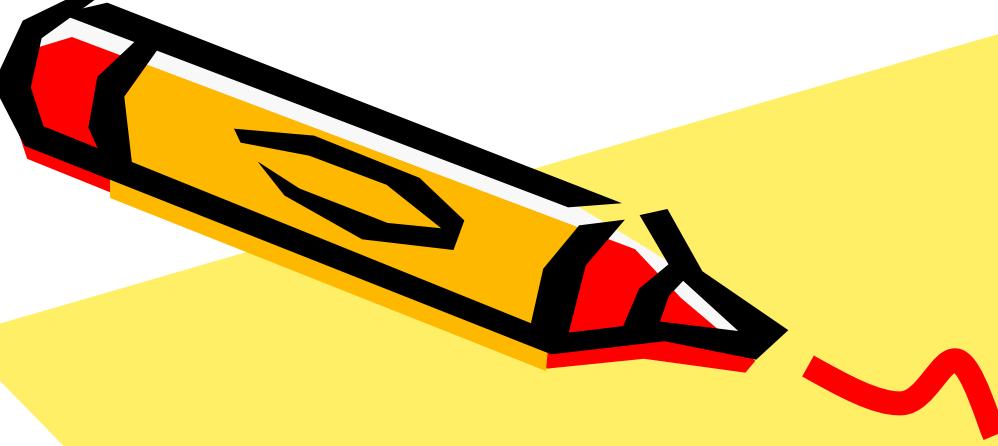




Thank You





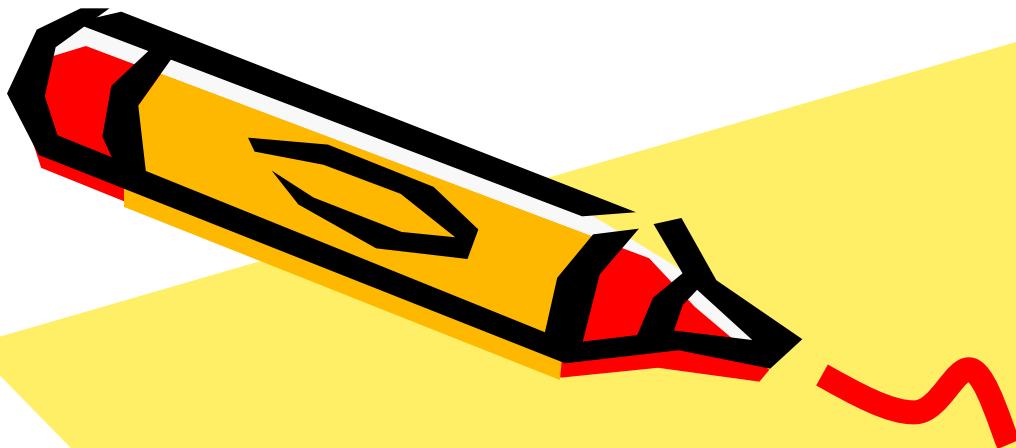


Logical Design

CS 221

Prof.Dr. Mohamed Osama Khozium





Binary Tools





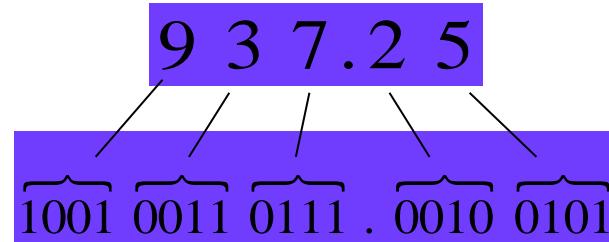
Computers deals with the binary systems through some tools (codes)....

For examples BCD, Gray, and excess-3 code
We will discuss these three codes in the following slides

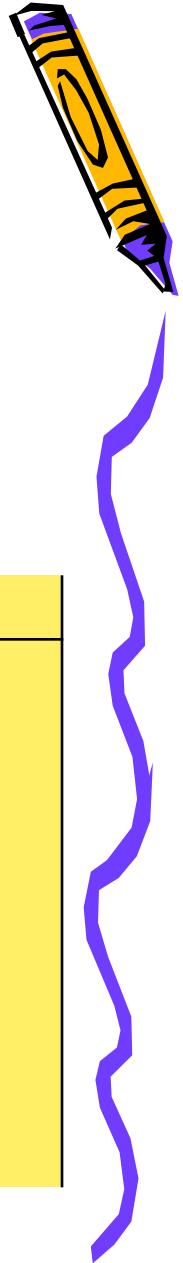




Binary Codes

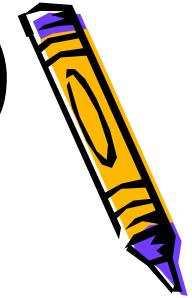


Decimal Digit	8-4-2-1 Code (BCD)	Gray Code	Excess-3 Code
0	0000	0000	0011
1	0001	0001	0100
2	0010	0011	0101
3	0011	0010	0110
4	0100	0110	0111
5	0101	0111	1000
6	0110	0101	1001
7	0111	0100	1010
8	1000	0100	1011
9	1001	1101	1100





BCD (binary coded decimal)

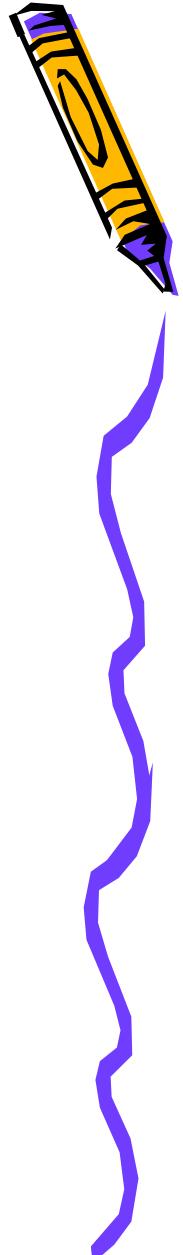


Decimal Digit	8-4-2-1 Code (BCD)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001





BCD conversion



- Code = 4 bit
- No of combination = $2^4 = 16$

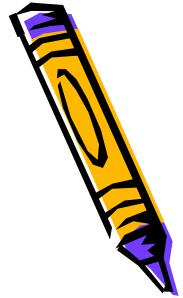
Ex 1 : convert each of the following decimal to BCD code :

- a) 35
- b) 98
- c) 170
- d) 2469





BCD conversion



Solution

a) 35 3 5
 0011 0101

Then $35 \rightarrow 00110101$

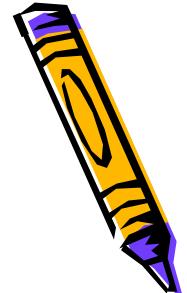
b) 98 9 8
 1001 1000

Then $98 \rightarrow 10011000$





BCD conversion



c) 170 1 7 0
 0001 0111 0000

Then 170 → 000101110000

d) 2469 2 4 6 9
 0010 0100 0110 1001

Then 2469 → 0010010001101001





BCD conversion

Ex2 Convert each of the following BCD code to decimal :

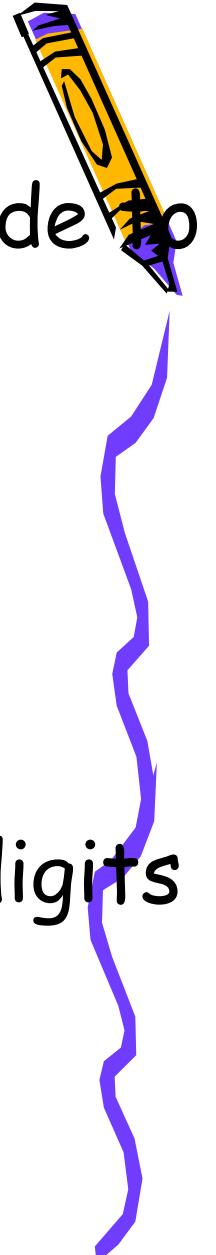
- a) 10000110 b) 001101010001
- c) 1001010001110000

Solution

a) Start from right and group each four digits

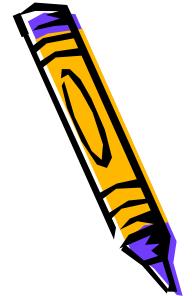
1000	0110
8	6

Then $10000110 \rightarrow 86$





BCD conversion



b) 001101010001

0011 0101 0001

3 5 1 → 351

c) 1001 0100 0111 0000

9

4

7

0

→ 9470

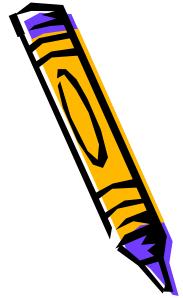




BCD addition

1 - use binary addition rules

2 - if the 4-bit sum is greater than 9 then
it is not a BCD valid numberadd
6(0110) to the 4-bit sum.





BCD addition

Add the following BCD numbers

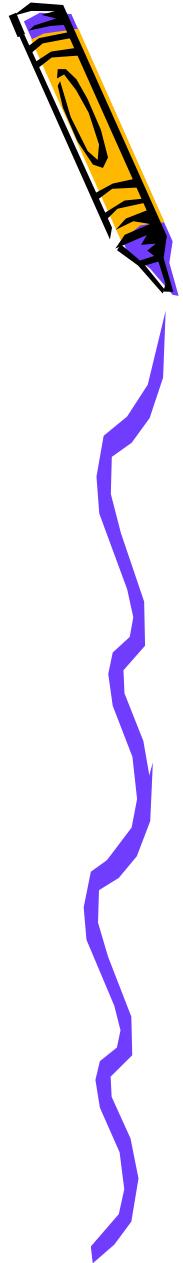
a) $0011 + 0100$

b) $001000111 + 00010101$

c) $1001 + 0100$

d) $00010110 + 00010101$

e) $01100111 + 01010011$





BCD addition

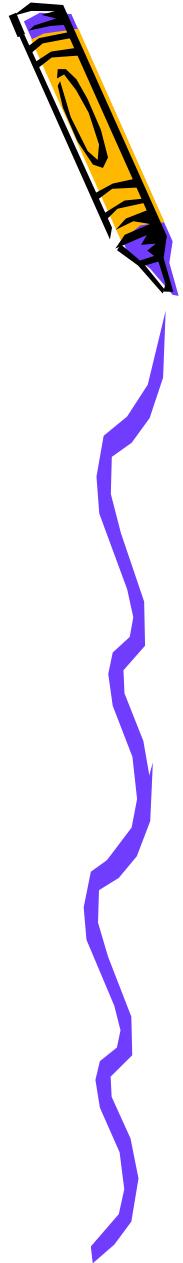
Solutions

a) 0011 → 3
 + 0100 → 4

 0111 → 7

b) 00100011 → 23
 + 00010101 → 15

 00111000 → 38 (each number < 9)





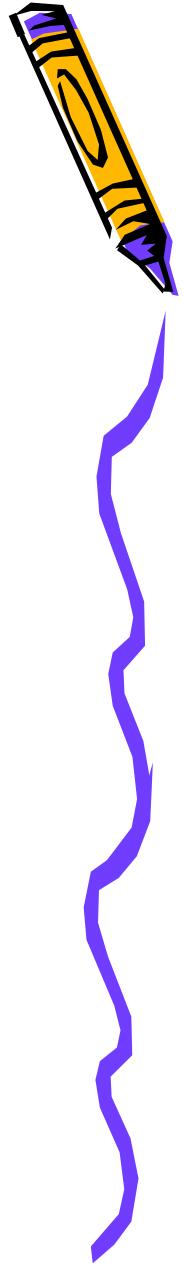
BCD addition

Solutions

$$\begin{array}{r} c) \quad 1001 \rightarrow 9 \\ + \quad 0100 \rightarrow 4 \\ \hline \end{array}$$

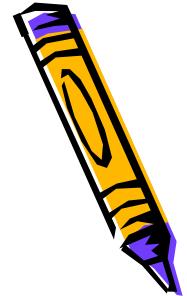
$$\begin{array}{r} 1101 \rightarrow 13 \rightarrow \text{invalid BCD number} > 9 \\ + \quad 0110 \rightarrow \text{Add } 6 \text{ (0110)} \\ \hline \end{array}$$

$$10011 \rightarrow 0001 \ 10011 \rightarrow 13 \text{ in BCD}$$





BCD addition



Solutions

d) 00010110 → 1 6
 + 00010101 → 1 5 $6+5 = 11 > 9$

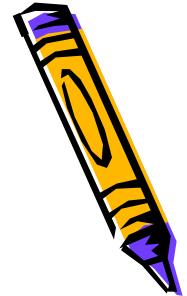
00101011 → 1011 > 9 then add 6(0110)
+ 0110 → Add 6 (0110)

00110001 → 0011 00001 → 31 in BCD





BCD addition



Solutions

e) 01100111 → 6 7

$$+ \quad 01010011 \rightarrow \quad 5 \quad 3 \quad 6+5 = 11 > 9 \text{ & } 7+3 > 9$$

$$\underline{\underline{1011 \quad 1010}} \rightarrow 1011 > 9 \text{ then add } 6(0110)$$

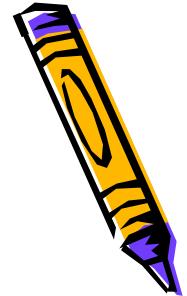
$$+ \quad 0110 \quad 0110 \rightarrow \text{Add } 6 \text{ (0110) \& } 6(0110)$$

$$1 \ 0010 \ 0000 \rightarrow 0001 \ 0010 \ 0000 \rightarrow 120$$





Gray Code



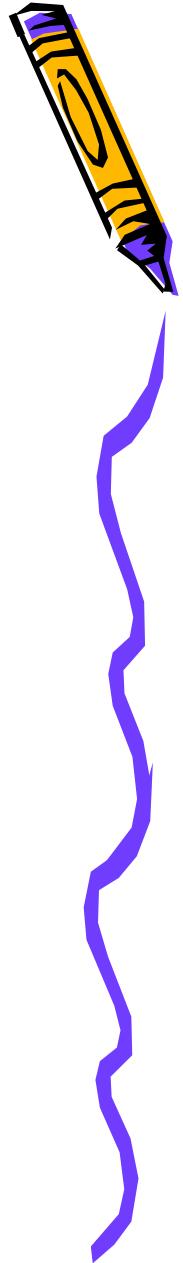
Decimal Digit	8-4-2-1 Code (BCD)	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101





Gray Code

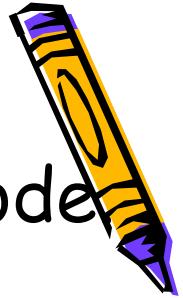
It is unweighted code and it is not arithmetic code





Binary to Gray Code

Convert the binary number 11000110 to Gray code



Sol.

Binary $1 + 1 + 0 + 0 + 0 + 1 + 1 + 0$

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

Gray 1 0 1 0 0 1 0 1

Shortcut

The first number will be the same...the second number in gray = first + second in binary..... the third in gray = second + third in binary and go on
...neglect carry.





Gray Code to binary



Convert the gray code number 10100101 to binary.

Sol.

Gray	1	0	1	0	0	1	0	1
	↓	↓	↓	↓	↓	↓	↓	↓
Binary	1 + 0 = 1	+ 1 = 0	+ 0 = 0	0	0	1	1	0

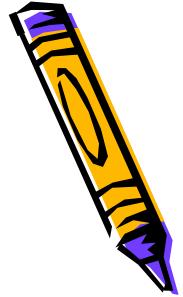
Shortcut

The first number will be the same...the second number in binary = first (binary) + second (gray)..
the third in binary = second (binary)+ third (gray)
and go on ...neglect carry.





Excess - 3 code



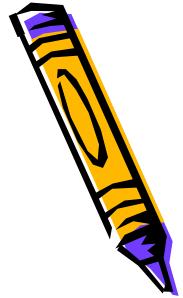
It is a digital code related to BCD derived by adding 3 to each decimal digit ...then converting the result to 4-bit binary ..

It is unweighted code.





Excess - 3 code



Convert each of the following decimal numbers to excess-3 code. a) 13 b) 430

a)

$$\begin{array}{r} 1 \quad 3 \\ + \quad 3 \quad + \quad 3 \\ \hline 4 \quad 6 \end{array}$$

0100 0110 → excess-3 code



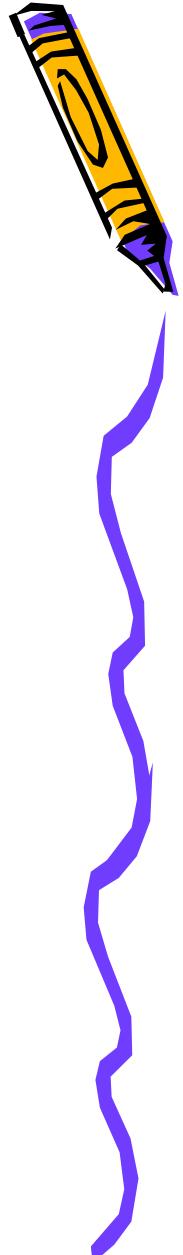


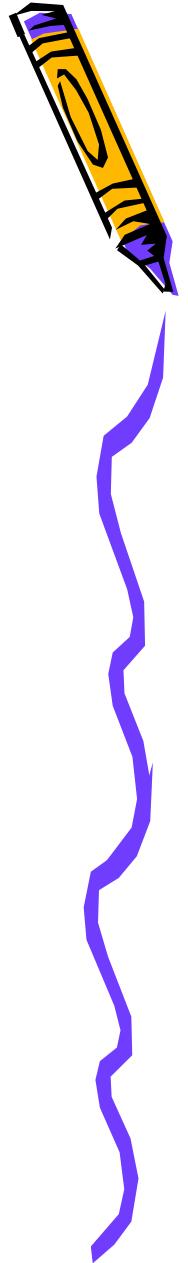
Excess - 3 code

a)

$$\begin{array}{r} 4 & 3 & 0 \\ + & 3 & + 3 & + 3 \\ \hline 7 & 6 & 3 \end{array}$$

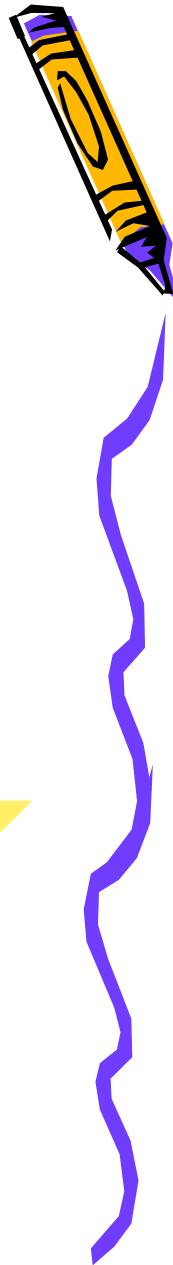
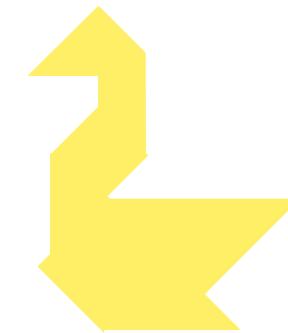
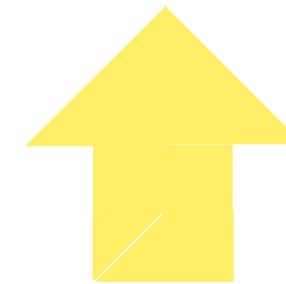
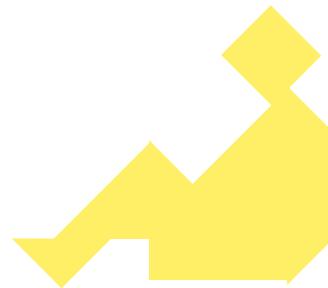
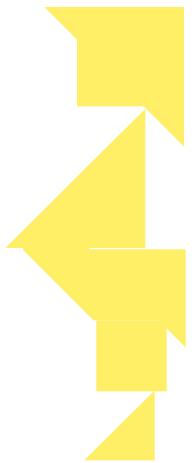
0111 0110 0011 → excess-3 code

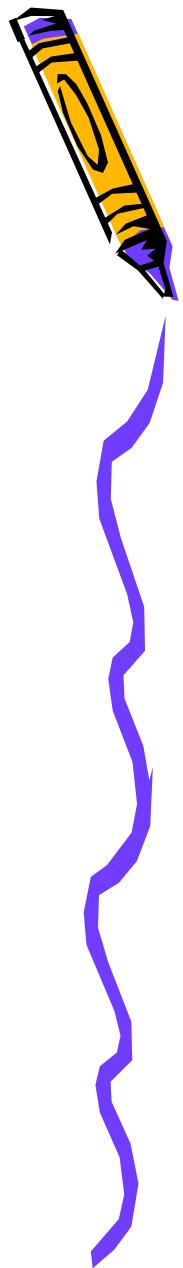


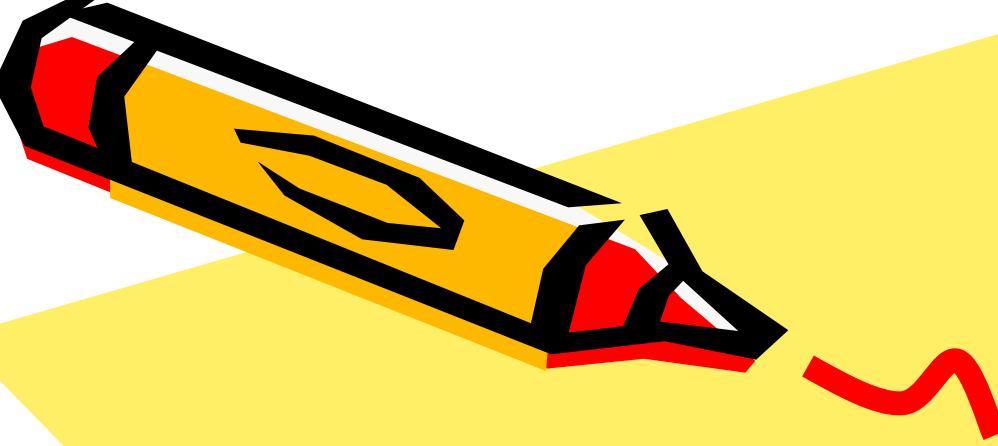


Thank you







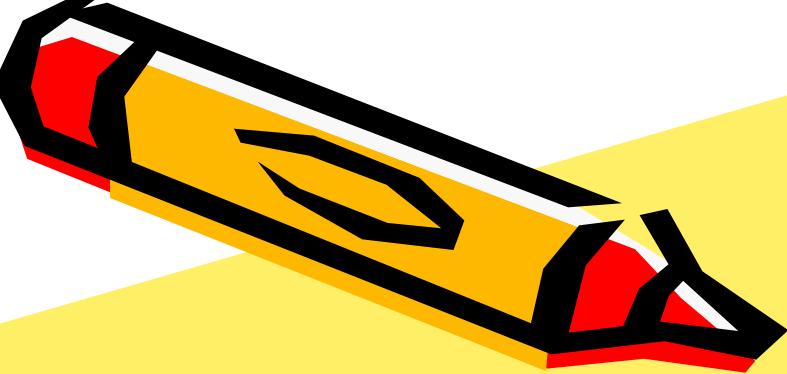


Logical Design

CS 221

Prof.Dr. Mohamed Osama Khozium





\sim

Simple Logic Gates & Boolean algebra





Module Outline



- Binary Logic and Basic Gates
- Boolean Algebra
- The Boolean expression for a logic circuit
- Implementation of a logic circuit using B.E
- Implementation of a logic circuit via truth table
- Converting a Boolean expression to a truth table
- Simplification of Boolean Functions





Binary Logic



- Deals with variables that take on only two discrete values and operations of mathematical logic that are applied to these variables
- The two values are known by different names:
 - HIGH and LOW
 - TRUE and FALSE
 - 0 and 1
- Variables may be denoted by any name but single letter character names such as A, B, C, X, Y, Z , etc. are common and easy to use





Logical Operations



- AND
 - Represented by a dot \cdot or sometimes \wedge or \cap
 - Binary operator, i.e. operates on two variables at a time
- OR
 - Represented by a + or sometimes \vee or \cup
 - Binary operator, i.e. operates on two variables at a time
- NOT
 - Represented by a bar over the variable, e.g. \bar{A} or A'
 - Unary operator, i.e. operates on a single variable
 - Also referred to as COMPLEMENT operator





Binary Logic and Basic Gates



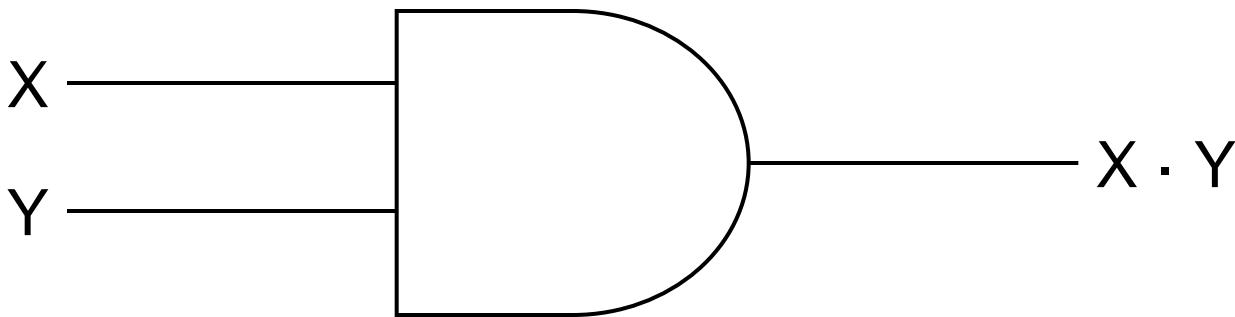
- Digital Circuits and Systems are complex
- However, the basic building blocks of all digital circuits and systems are logic gates
- Logic gates are therefore called the basic primitives of digital systems
- Logic gates in reality are implemented using electronic components such as transistors
- However, we are not concerned with their internal electronic properties but rather their external logic behavior



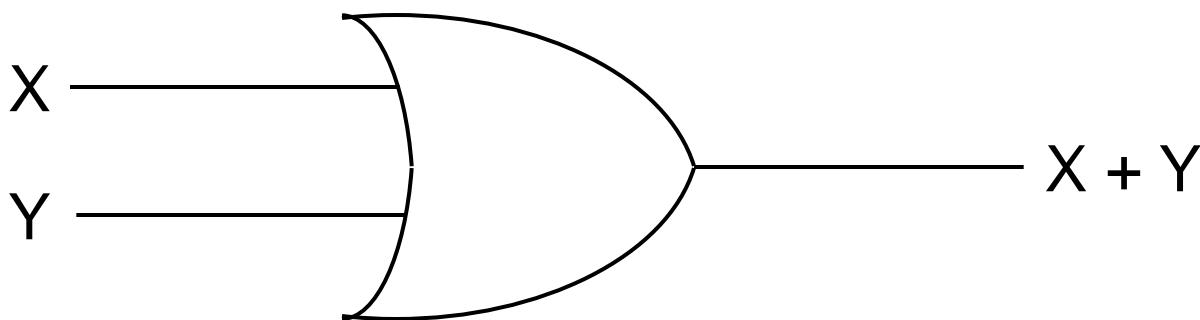


Standard Symbols

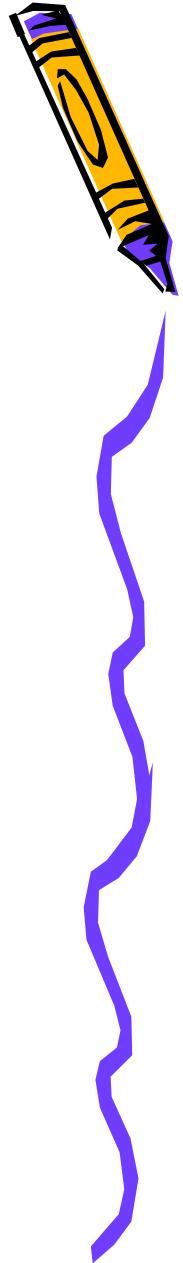
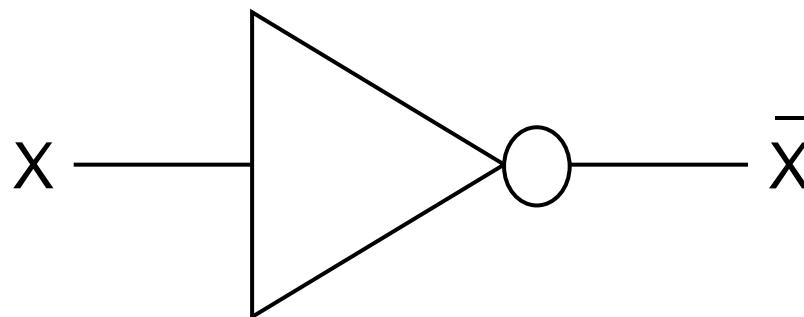
AND



OR



NOT





Truth Table



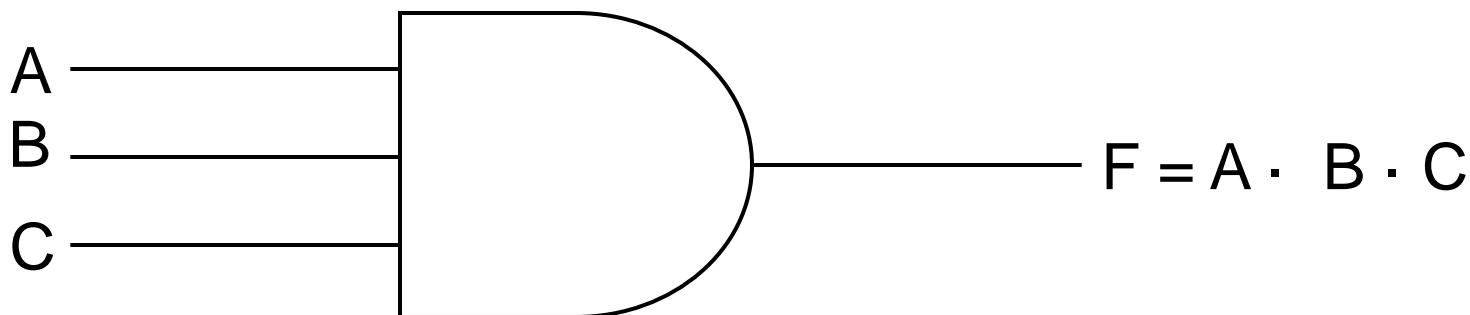
- A table of combinations of the binary variables showing the relationship between the values the variables take on and the values of the result of the operation

AND		OR		NOT	
X	Y	Z = X · Y	X	Y	Z = X + Y
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	1	0	1
1	1	1	1	1	1

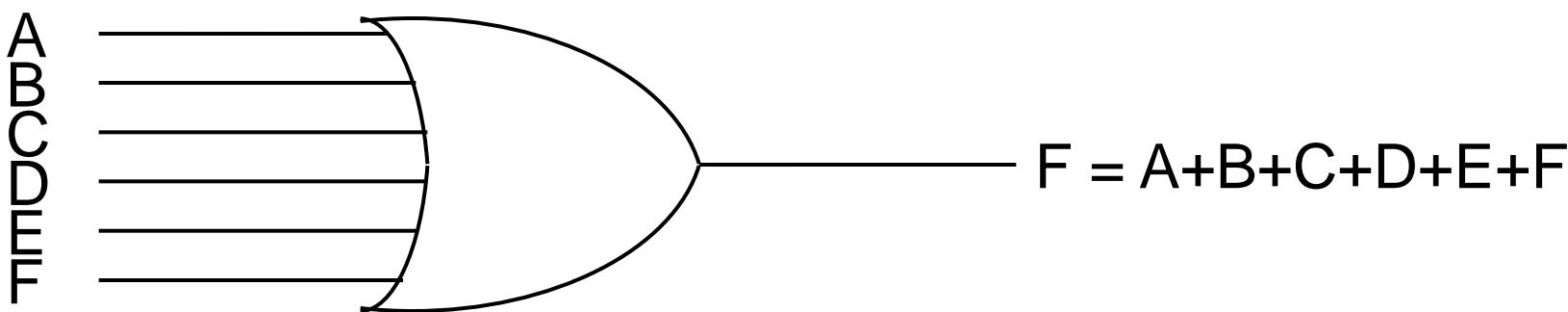




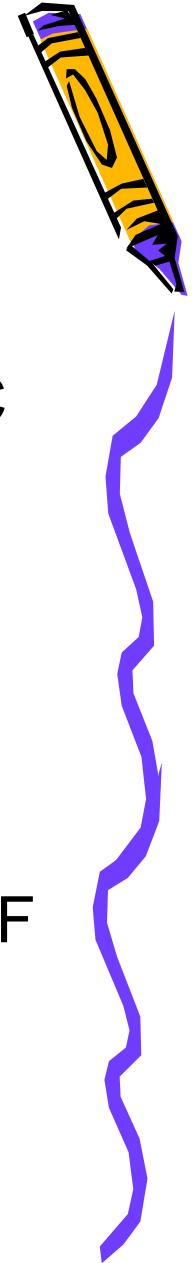
Gates with More Than Two Inputs



3-Input AND gate

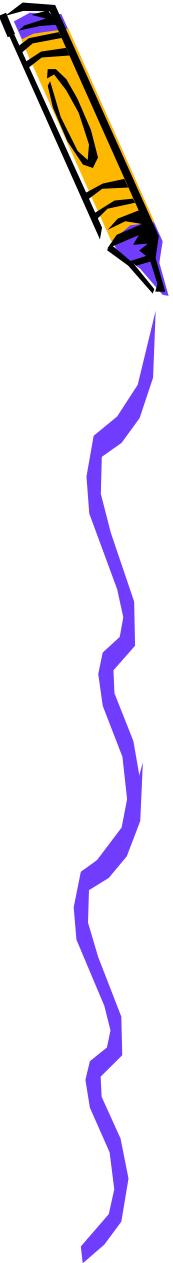


6-Input OR gate





Truth Table



AND

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





Boolean Algebra



- Defines rules of operations on Binary Logic and Logic Functions
- Sometimes similar to Binary Arithmetic but sometimes different - because binary logic variables can only take on two possible values 0 and 1
- A Boolean function consists of a binary variable denoting the function, an equals sign, and an algebraic expression formed by using binary variables. For example

$$F = X + Y'Z$$





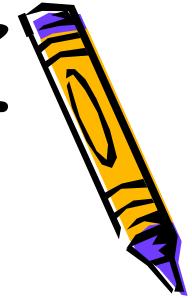
Truth Tables for Boolean Functions

- Truth Table can be constructed for every boolean function
- A boolean function of n -variables will have 2^n rows in its truth table
- These 2^n inputs are formed by counting from 0 to $2^n - 1$
- For example $G(W,X,Y,Z)$ will have 16 rows in the truth table and $F(A,B)$ will have 4 rows





Truth Table for $F = X + Y'Z$

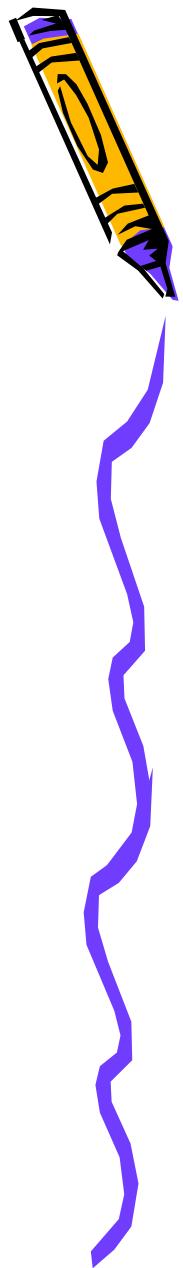
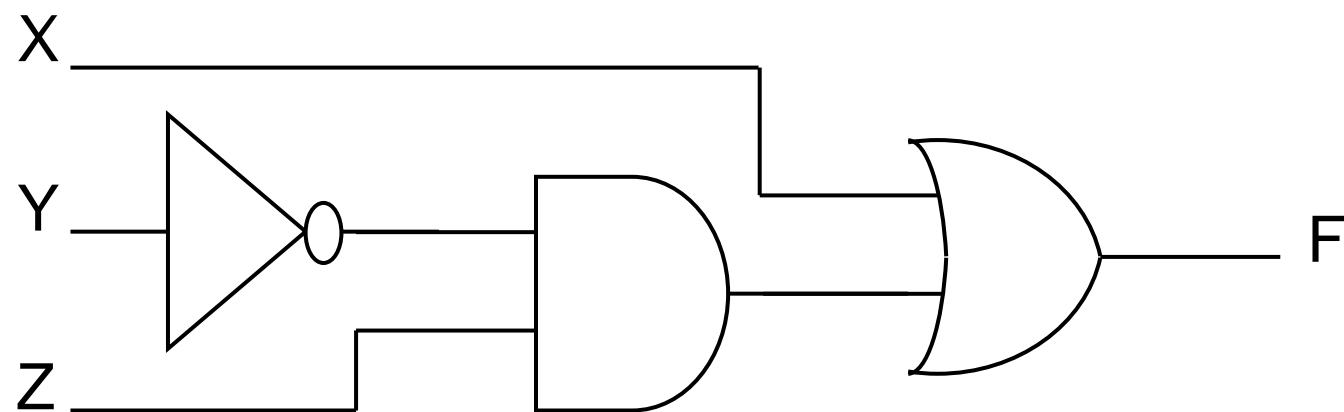


x	y	z	y'	$y'z$	$x + y'z$
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	0	0	1





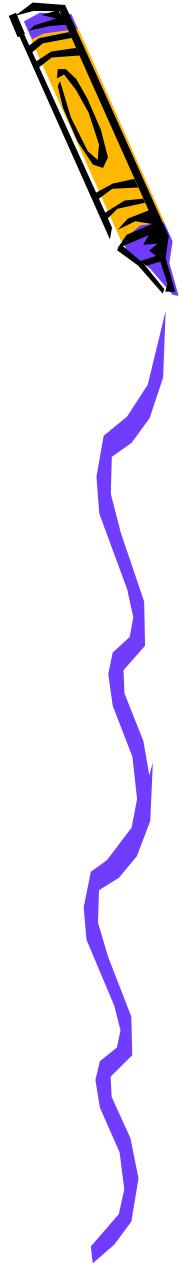
Logic Circuit Diagram for $F = X + Y'Z$

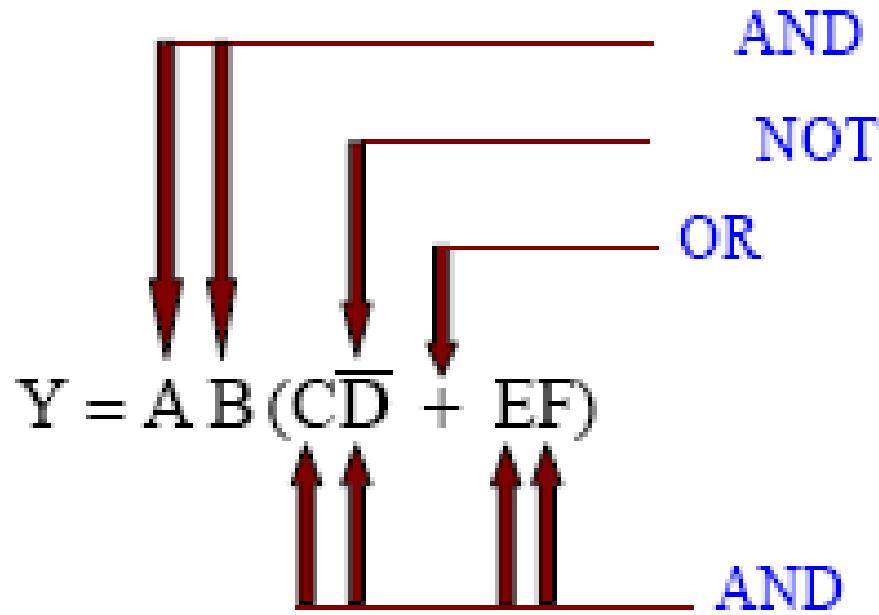
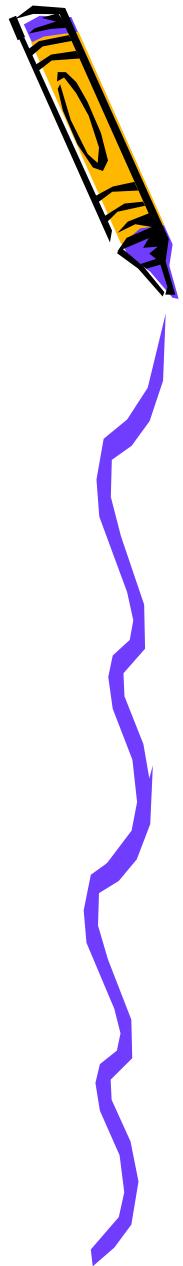


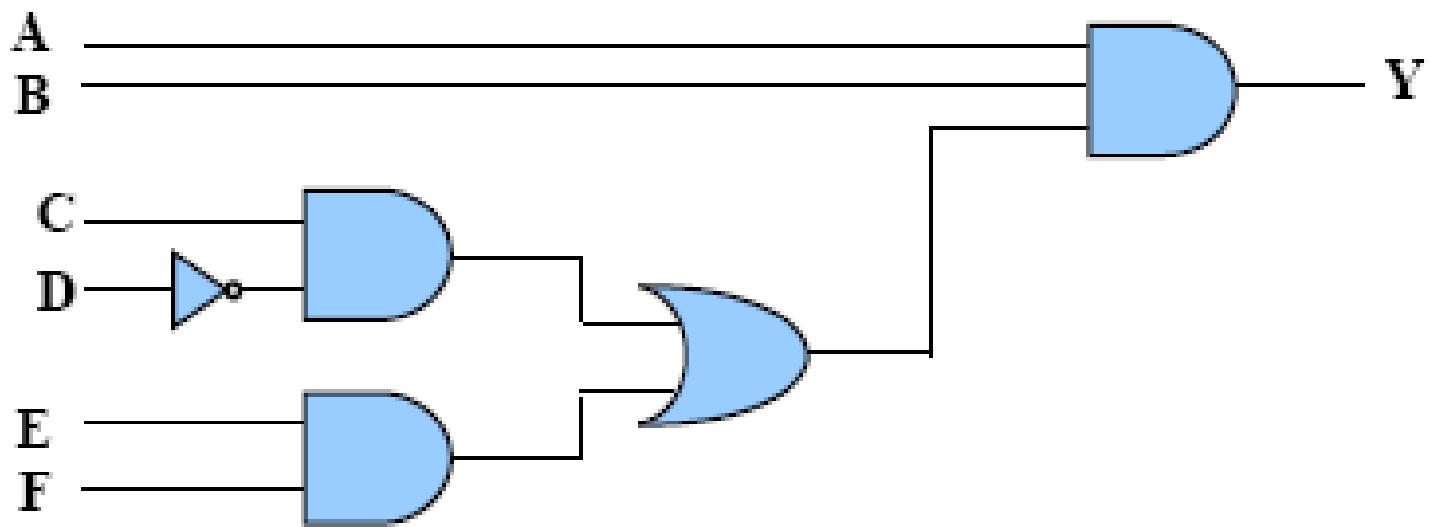
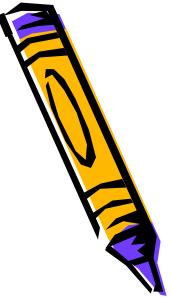


Implementation of a logic circuit using a boolean expression

$$Y = AB(CD' + EF)$$







$$Y = AB(CD' + EF)$$

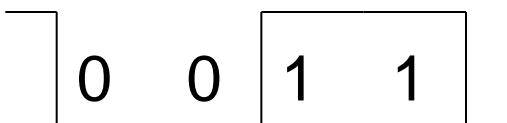




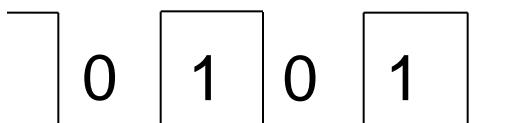
Timing Diagram



X



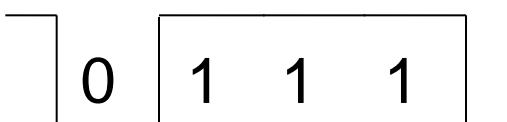
Y



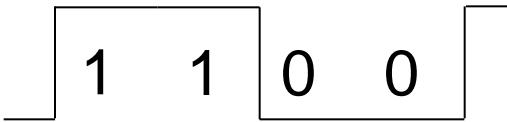
AND



OR



NOT(X)



- Horizontal axis represents time
- Vertical axis shows the value of the signal
- High voltage level represents 1 and low voltage level represents 0
- Timing diagrams are very important in digital systems design and verification





Basic Identities of Boolean Algebra

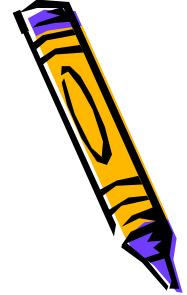


1. $X + 0 = X$
2. $X \cdot 1 = X$
3. $X + 1 = 1$
4. $X \cdot 0 = 0$
5. $X + X = X$
6. $X \cdot X = X$
7. $X + X' = 1$
8. $X \cdot X' = 0$
9. $(X')' = X$





Basic Identities of Boolean Algebra



$$1. \quad X + Y = Y + X$$

$$2. \quad XY = YX$$

$$3. \quad X + (Y + Z) = (X + Y) + Z$$

$$4. \quad X(YZ) = (XY)Z$$

$$5. \quad X(Y+Z) = XY + XZ$$

$$6. \quad X + YZ = (X+Y)(X+Z)$$

$$7. \quad (X + Y)' = X'Y'$$

$$8. \quad (XY)' = X' + Y'$$

Commutative

Associative

Distributive

DeMorgan's





OR

1 $x + 1 = 1$

2 $x + x' = 1$

3 $x + x = x$

4 $x + 0 = x$

5 $(x')' = x$

6 $x + y = y + x$

7 $x + (y + z) = (x + y) + z$

8 $x \cdot (y + z) = x \cdot y + x \cdot z$

9 $(x + y)' = x' \cdot y'$

10 $x + (x \cdot y) = x$

AND

1 $x \cdot 1 = x$

2 $x \cdot x' = 0$

3 $x \cdot x = x$

4 $x \cdot 0 = 0$

5 $(x')' = x$

6 $x \cdot y = y \cdot x$

7 $x \cdot (y \cdot z) = (x \cdot y) \cdot z$

8 $x + y \cdot z = (x + y) \cdot (x + z)$

9 $(x \cdot y)' = x' + y'$

10 $x \cdot (x + y) = x$

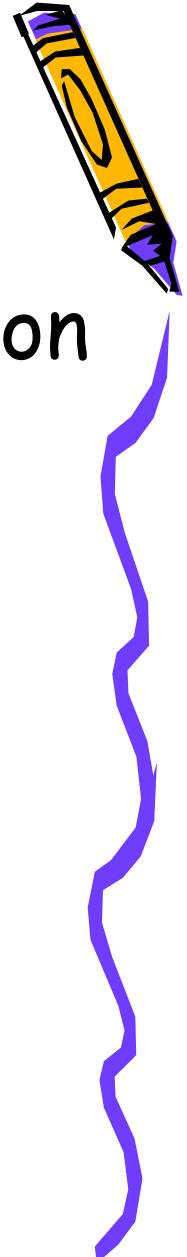




Algebraic Manipulation

- Using basic identities a boolean function can be simplified
- For example

$$F = X'YZ + X'YZ' + XZ$$





Algebraic Manipulation



- Using basic identities a boolean function can be simplified
- For example

$$\begin{aligned} F &= X'YZ + X'YZ' + XZ \\ &= X'Y(Z + Z') + XZ \\ &= X'Y \cdot 1 + XZ \\ &= X'Y + XZ \end{aligned}$$





Algebraic Manipulation Examples



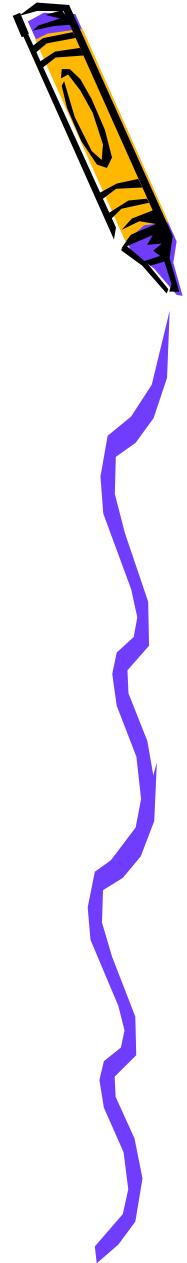
1. $X + XY =$
2. $XY + XY' =$
3. $X + X'Y =$
4. $X(X + Y) =$
5. $(X + Y)(X + Y') =$

6. $X(X' + Y) =$





Algebraic Manipulation Examples

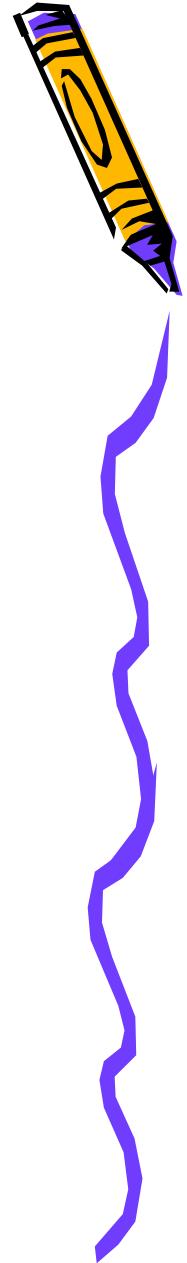


$$1. \ X + XY = X(1+Y) = X$$





Algebraic Manipulation Examples



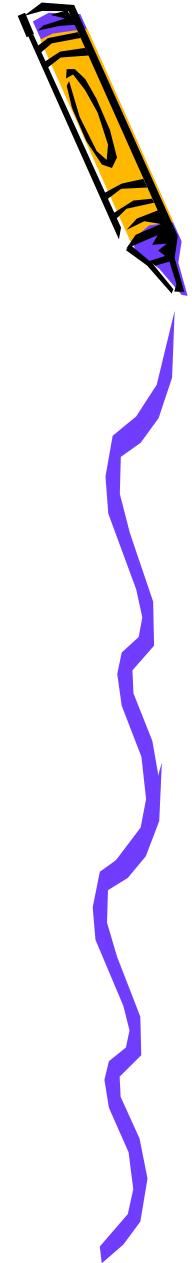
$$2 \cdot XY + XY' = X(Y + Y') = X$$





Algebraic Manipulation Examples

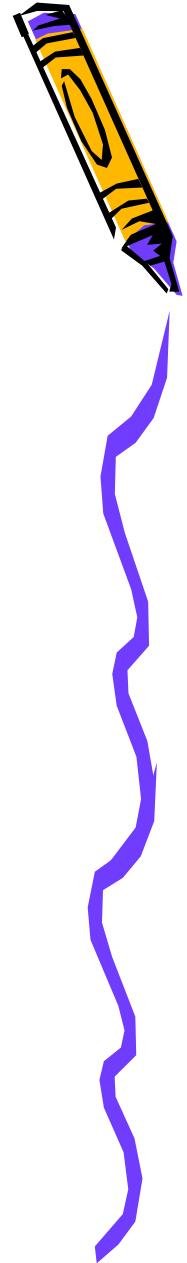
$$3 . \ X + X'Y = (X + X') (X + Y) = X + Y$$





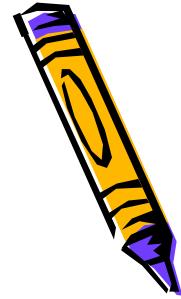
Algebraic Manipulation Examples

$$4 \cdot X (X + Y) = X + XY = X (1 + Y) = X$$





Algebraic Manipulation Examples



$$1. \quad X + XY = X(1+Y) = X$$

$$2. \quad XY + XY' = X(Y + Y') = X$$

$$3. \quad X + X'Y = (X + X')(X + Y) = X + Y$$

$$4. \quad X(X + Y) = X + XY = X(1 + Y) = X$$

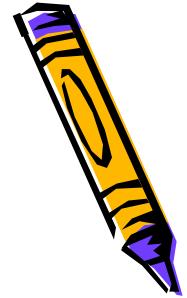
$$\begin{aligned} 5. \quad (X + Y)(X + Y') &= X + XY' + XY + YY' \\ &= X(1 + Y') + XY \\ &= X \cdot X + XY \\ &= X + XY \\ &= X(1 + Y) \\ &= X \end{aligned}$$

$$6. \quad X(X' + Y) = XX' + XY = XY$$





Algebraic Manipulation Examples



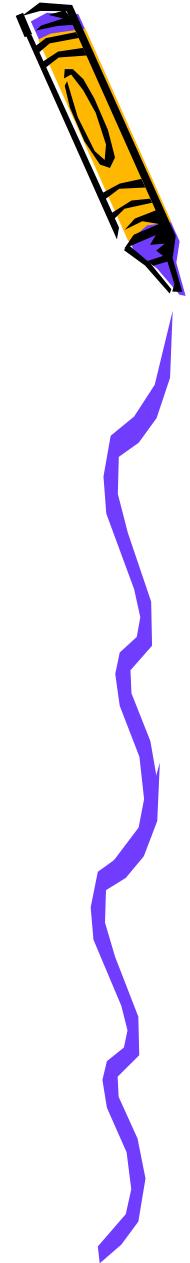
$$\begin{aligned}5 \cdot (X + Y)(X + Y') &= X + XY' + XY + YY' \\&= X(1 + Y') + XY \\&= X \cdot 1 + XY \\&= X + XY \\&= X(1 + Y) \\&= X\end{aligned}$$





Algebraic Manipulation Examples

$$6 \cdot X(X' + Y) = \mathbf{XX'} + XY = XY$$





Algebraic Manipulation Examples



$$1. \quad X + XY = X(1+Y) = X$$

$$2. \quad XY + XY' = X(Y + Y') = X$$

$$3. \quad X + X'Y = (X + X')(X + Y) = X + Y$$

$$4. \quad X(X + Y) = X + XY = X(1 + Y) = X$$

$$\begin{aligned} 5. \quad (X + Y)(X + Y') &= X + XY' + XY + YY' \\ &= X(1 + Y') + XY \\ &= X \cdot 1 + XY \\ &= X + XY \\ &= X(1 + Y) \\ &= X \end{aligned}$$

$$6. \quad X(X' + Y) = XX' + XY = XY$$

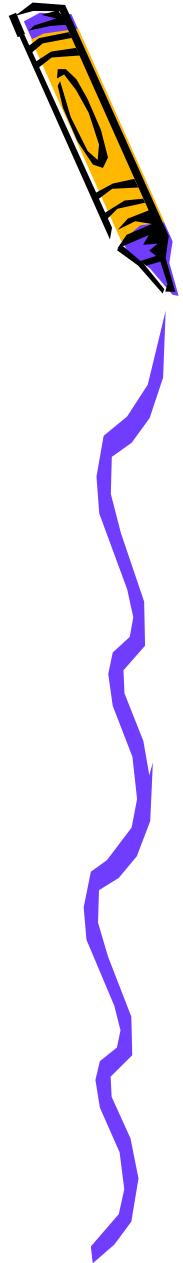




Example

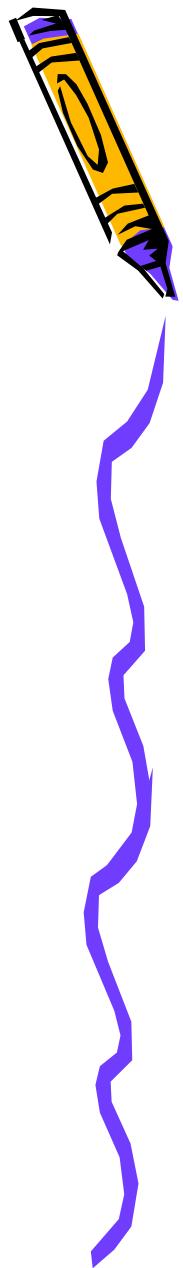
- Simplify the following Boolean expression and implement logic circuit for it

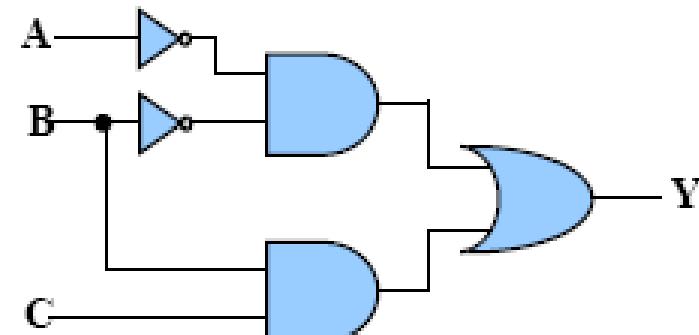
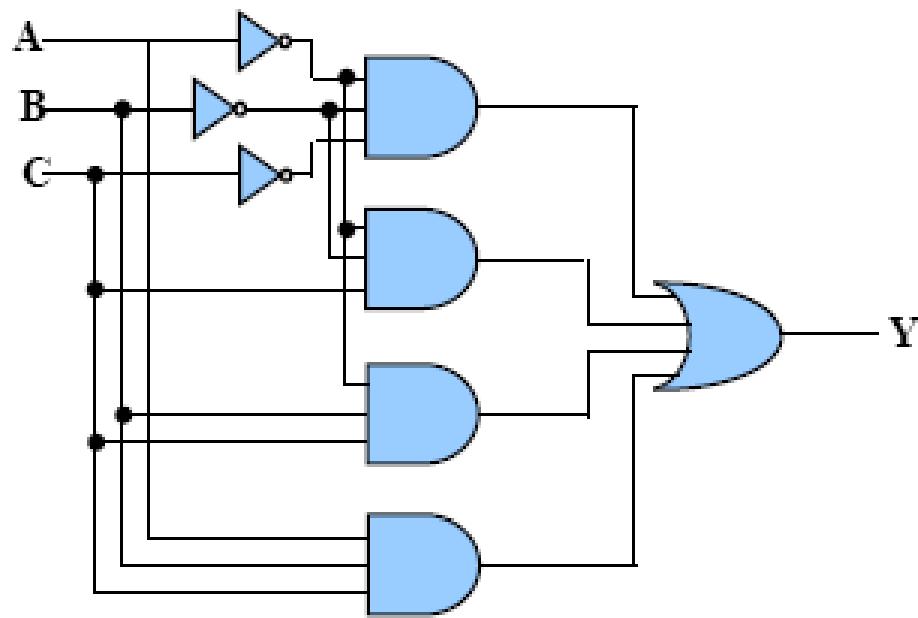
$$Y = A'B'C' + A'B'C + A'BC + ABC$$

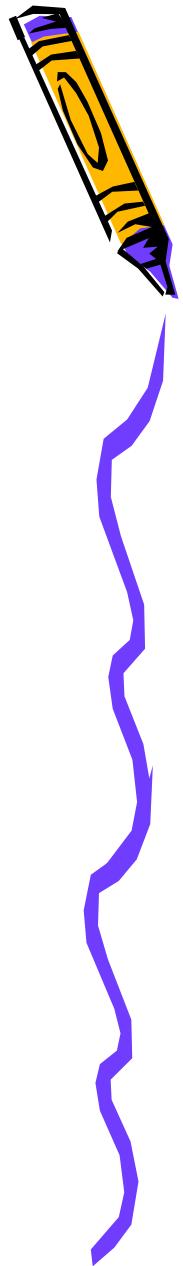




$$\begin{aligned}Y &= A'B'C' + A'B'C + A'BC + ABC \\&= A'B'(C'+C) + BC(A'+A) \\&= A'B' + BC\end{aligned}$$





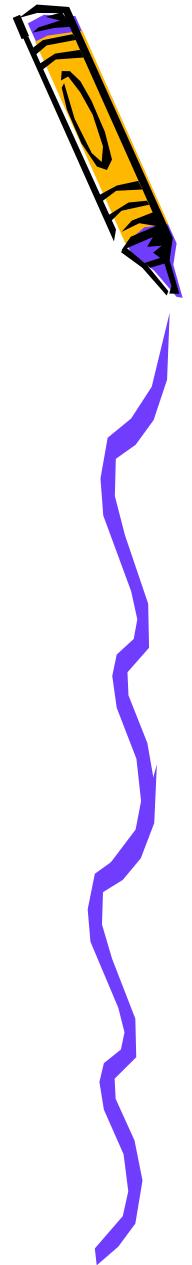


Combinational logic circuits



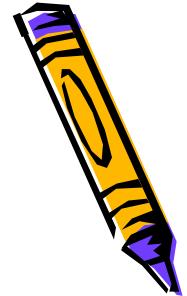


NOR & NAND





Basic Identities of Boolean Algebra



$$1. \quad X + Y = Y + X$$

$$2. \quad XY = YX$$

$$3. \quad X + (Y + Z) = (X + Y) + Z$$

$$4. \quad X(YZ) = (XY)Z$$

$$5. \quad X(Y+Z) = XY + XZ$$

$$6. \quad X + YZ = (X+Y)(X+Z)$$

$$7. \quad (X + Y)' = X'Y'$$

$$8. \quad (XY)' = X' + Y'$$

Commutative

Associative

Distributive

DeMorgan's





$$(A + B)' = A'B'$$

$$(AB)' = A' + B'$$

$$\overline{A + B} = \overline{A} \bullet \overline{B}$$

$$\overline{A \bullet B} = \overline{A} + \overline{B}$$

نظريّة ديمورجان الأولى:

نظريّة ديمورجان الثانية:





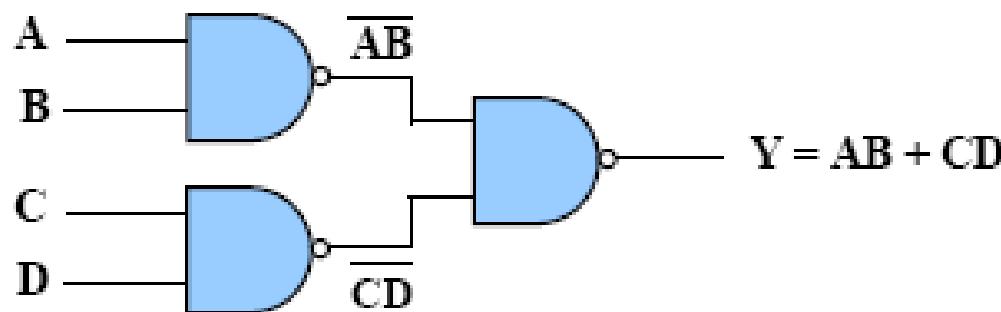
$$\overline{A \bullet B} = \overline{A} + \overline{B}$$

NAND

Negative-OR



Example



$$Y = (\overline{AB})(\overline{CD})$$

From Demorgan Theorem

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

Then $Y = AB + CD$

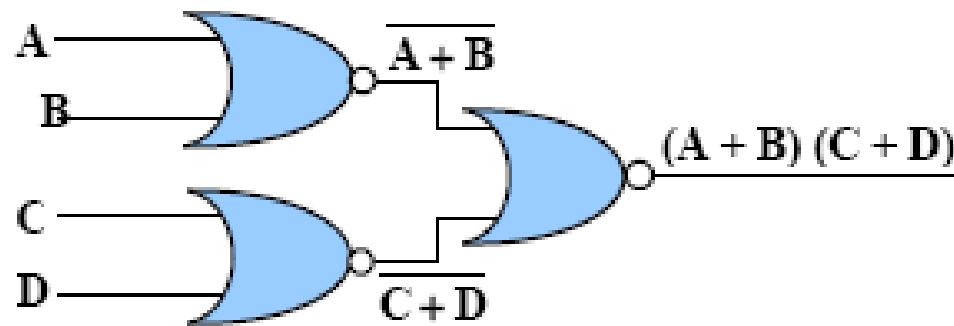




$$\overline{A + B} = \overline{A} \bullet \overline{B}$$

NOR

Negative-AND

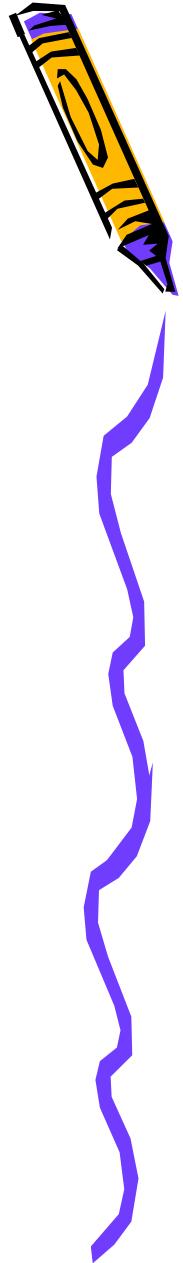


$$Y = \overline{\overline{(A + B)} + \overline{(C + D)}}$$

From Demorgan Theorem

$$Y = \overline{(A + B)} \bullet \overline{(C + D)}$$

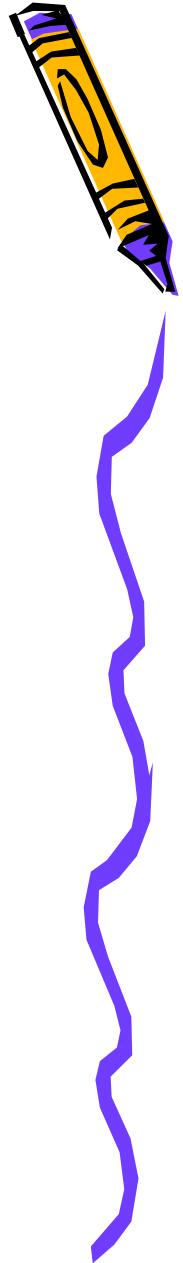
$$Y = (A + B) \bullet (C + D)$$





طبق نظريات ديمورجان على التعبير البوليني التالي:

$$Y = \overline{(A + \overline{B} + \overline{C}) \bullet (\overline{A} + B + \overline{C})}$$





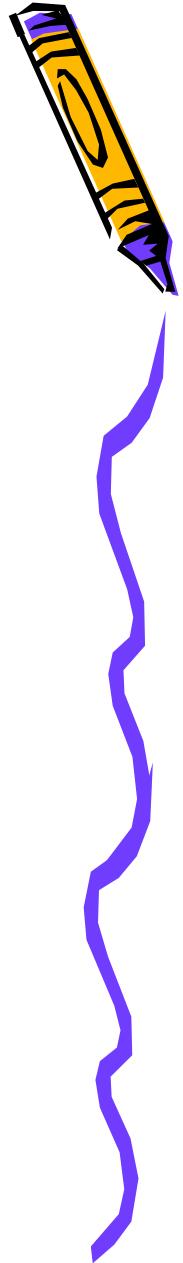
$$\begin{aligned} Y &= \overline{(A + \bar{B} + \bar{C})} \cdot (\bar{A} + B + \bar{C}) \\ &= \overline{(A + \bar{B} + \bar{C})} + \overline{\bar{A} + B + \bar{C}} \\ &= \overline{A} \bar{B} C + \bar{A} \overline{B} C = \overline{A} B C + A \overline{B} C \end{aligned}$$





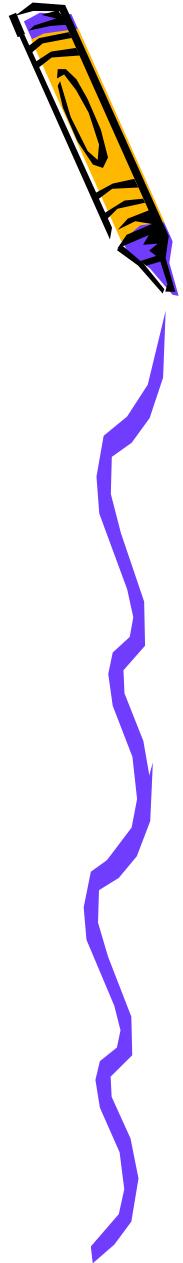
طبق نظريات ديمورجان على التعبير البوليني التالي:

$$Y = \overline{(A + B) + CD}$$





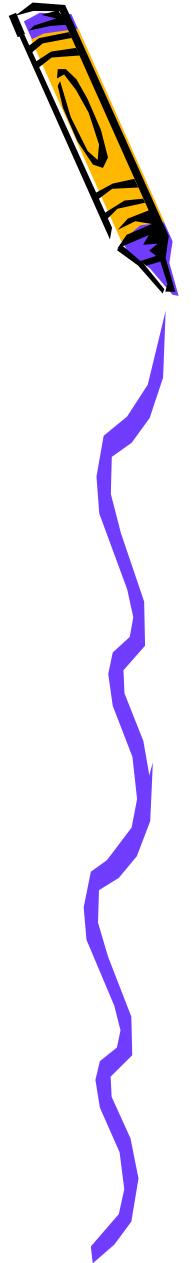
$$\begin{aligned} Y &= \overline{\overline{(A + B)} + \overline{CD}} \\ &= \overline{\overline{(A + B)} \cdot \overline{CD}} \\ &= \overline{\overline{A} \cdot \overline{B}} (\overline{C} + \overline{D}) \\ &= \overline{AB} (\overline{C} + \overline{D}) \end{aligned}$$

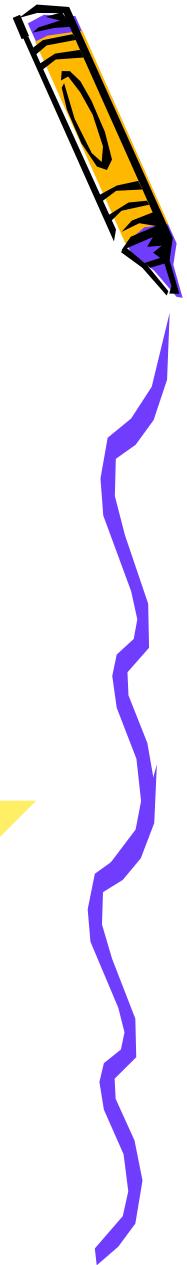
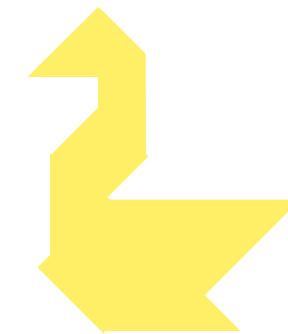
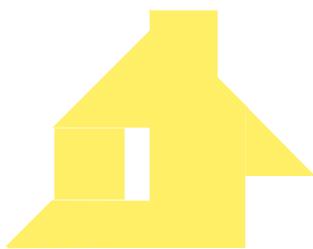
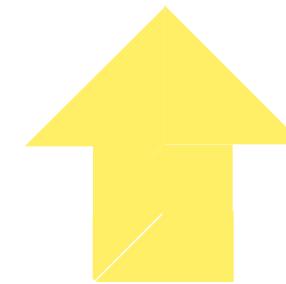
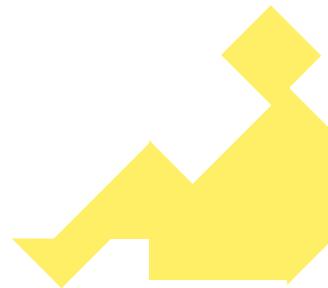
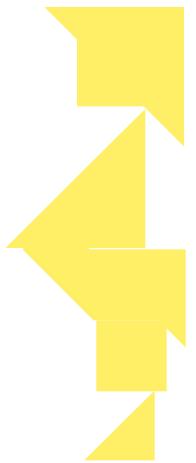


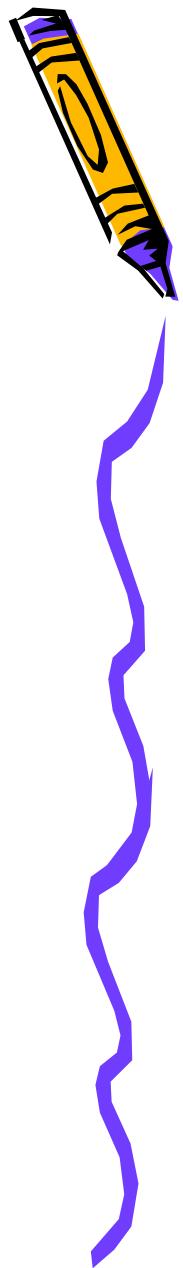


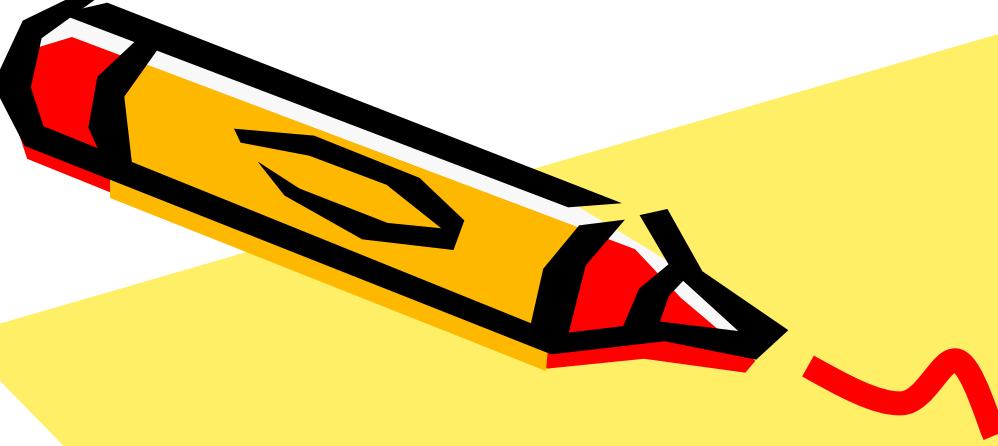


Thank you









Logical Design

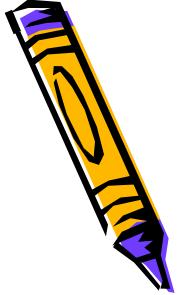
CS 221

Prof.Dr. Mohamed Osama Khozium





Complement of a Function



- Obtained by interchange of 1's to 0's and 0's to 1's for the values of F in the truth table
- Can be derived algebraically by applying DeMorgan's theorem
- Complement of an expression is obtained by interchanging AND and OR and complementing each variable and constant
- Example:

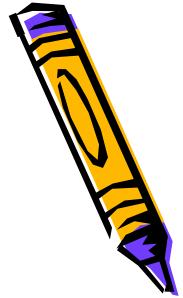
$$F = X'YZ' + X'Y'Z$$

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





Standard Forms

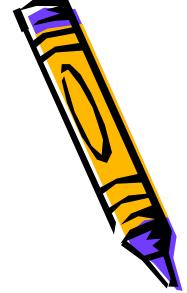


- Variety of possible ways of writing a boolean function algebraically
- Hard to see unambiguously that it is the same function we are talking about
- Standard forms have been developed
- Standard forms also generate more desirable logic circuits
- Product terms and Sum terms
 - Product: $XY'Z$
 - Sum: $X + Y' + Z$





Canonical and standard forms

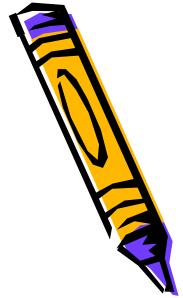


- Two Standard forms
 - 1) Sum of product .. SOP ...minterms .. Σ
 - 2) Product of sum .. POS ... maxterms.. Π





Minterms and maxterms of three binary variables



X	Y	Z	SOP Term	Designation	POS Term	Designation
0	0	0	$X'Y'Z'$	m0	$X+Y+Z$	M0
0	0	1	$X'Y'Z$	m1	$X+Y+Z'$	M1
0	1	0	$X'YZ'$	m2	$X+Y'+Z$	M2
0	1	1	$X'YZ$	m3	$X+Y'+Z'$	M3
1	0	0	$XY'Z'$	m4	$X'+Y+Z$	M4
1	0	1	$XY'Z$	m5	$X'+Y+Z'$	M5
1	1	0	XYZ'	m6	$X'+Y'+Z$	M6
1	1	1	XYZ	m7	$X'+Y'+Z'$	M7



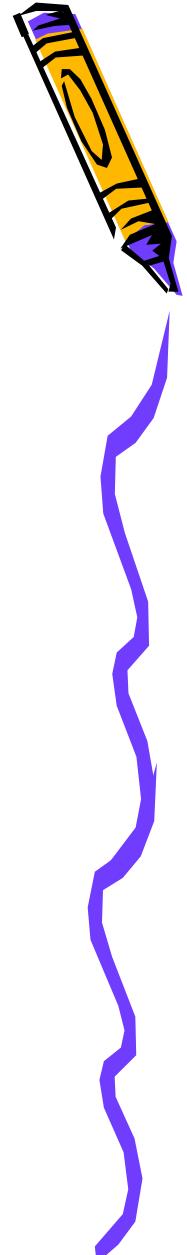


Canonical standard SOP form : each minterm is having all the variable in normal or complimented form " Normally from the truth table "

$$\text{Ex } F = \overline{A} B + A B + \overline{A} \overline{B}$$

Minimal SOP form : Each minterm does not have all the variables in normal or complimented form ...

$$\text{Ex } G = A + \overline{B} C$$





Que. For the given truth table, minimize the SOP expression

B	A	Y
0	0	0
1	0	1
0	1	0
1	1	1

Sol.

Note in SOP $1 = A \& 0 = \bar{A}$

We see when $Y = 1$ and write the expression

Then $Y = \bar{A}B + AB$ This is the standard or canonical form

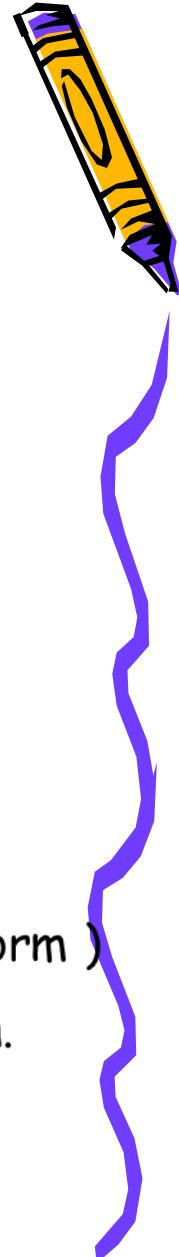
(each minterm is having all the variable in normal or complimented form)

We will convert the standard form to minimal form by simplification.

$$Y = \bar{A}B + AB = B(\bar{A} + A) = B * 1 = B$$

Then $Y = B$ Minimal form

In the truth table $Y = B$





Que. Simplify the expression for
 $Y(A, B) = \sum m(0, 2, 3)$

Sol. from the truth table

Then
$$Y = \overline{A} \overline{B} + A \overline{B} + AB$$

$$\begin{aligned} Y &= \overline{B} (\overline{A} + A) + AB \\ &= \overline{B} + AB = (\overline{B} + A) (\overline{B} + B) \end{aligned}$$

m0 means 0 0 $\overline{A} \overline{B}$

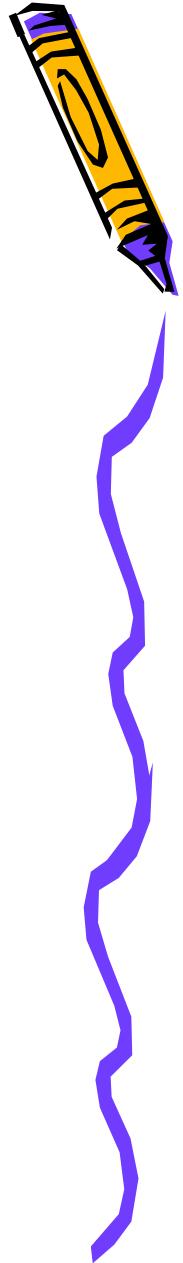
m2 means 1 0 $A \overline{B}$

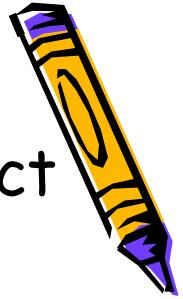
m3 means 1 1 AB

canonical / standard SOP Form

minimal SOP form

A	B	Y from TT	\overline{B}	$Y = A + \overline{B}$
0	0	1	1	1
0	1	0	0	0
1	0	1	1	1
1	1	1	0	1





Express the Boolean function $f = A + B'C$ in sum of product

We have two methods ..algebraic & truth table

1) Algebraic method (two approaches)

a) Algebraic first approach

$$F = A + B'C$$

$$A = A * 1 = A(B+B') = AB + AB' = AB + AB'(C+C')$$

$$\rightarrow A = ABC + ABC' + AB'C + AB'C'.$$

$$B'C = B'C * 1 = B'C(A+A') = B'CA + B'CA' \rightarrow AB'C + A'B'C$$

$$\dots F = ABC + ABC' + \underline{AB'C} + AB'C' + \underline{AB'C} + A'B'C$$

$$F = ABC + ABC' + AB'C + AB'C' + A'B'C$$

$$F = m_1 + m_4 + m_5 + m_6 + m_7 \rightarrow \text{look the previous slide}$$

$$\dots F(A, B, C) = \sum(1, 4, 5, 6, 7)$$





b) Algebraic second approach

$F = A + B'C$ (complete each part by all the rest available symbols)

$$F = A (BC + B'C + BC' + B'C') + B'C (A + A')$$

$$\rightarrow F = ABC + \underline{AB'C} + ABC' + AB'C' + \underline{AB'C} + A'B'C.$$

Erase the repeated symbols

$$\dots F = ABC + ABC' + AB'C + AB'C' + A'B'C$$

$$F = m_1 + m_4 + m_5 + m_6 + m_7 \rightarrow \text{look the previous slide}$$

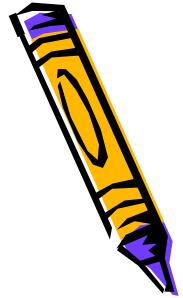
$$\dots F (A, B, C) = \sum (1, 4, 5, 6, 7)$$





2) Truth table method

$$F = A + B'C$$



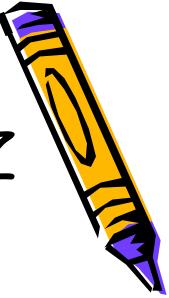
A	B	C	B'	B'C	A+B'C	Designation
0	0	0	1	0	0	m0
0	0	1	1	1	1	m1
0	1	0	0	0	0	m2
0	1	1	0	0	0	m3
1	0	0	1	0	1	m4
1	0	1	1	1	1	m5
1	1	0	0	0	1	m6
1	1	1	0	0	1	m7

Solution

$$F(A, B, C) = \sum(1, 4, 5, 6, 7)$$

means at these points results = 1





Example : express the boolean function $F = XY + X'Z$ in product of sum

1 - Algebraic Method

$$\begin{aligned} F = XY + X'Z &= (XY + X')(XY + Z) \\ &= (X' + X)(X' + Y)(Z + X)(Z + Y) \\ &= (X' + Y)(Z + X)(Y + Z) \end{aligned}$$

MISSING Z Y X

$$X' + Y = X' + Y + 0 = X' + Y + ZZ' = (X' + Y + Z)(X' + Y + Z')$$

$$Z + X = X + Z + 0 = X + Z + YY' = (X + Z + Y)(X + Z + Y')$$

$$Y + Z = Y + Z + 0 = Y + Z + XX' = (X + Y + Z)(X' + Y + Z)$$

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

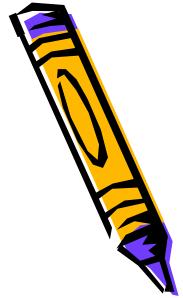
$$F(x, y, z) = XY + X'Z = m_0 m_2 m_4 m_5 = \Pi(0, 2, 4, 5)$$





Truth table method

$$F = XY + X'Z$$

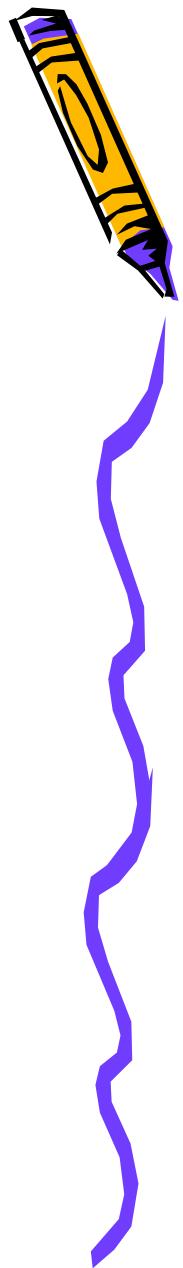


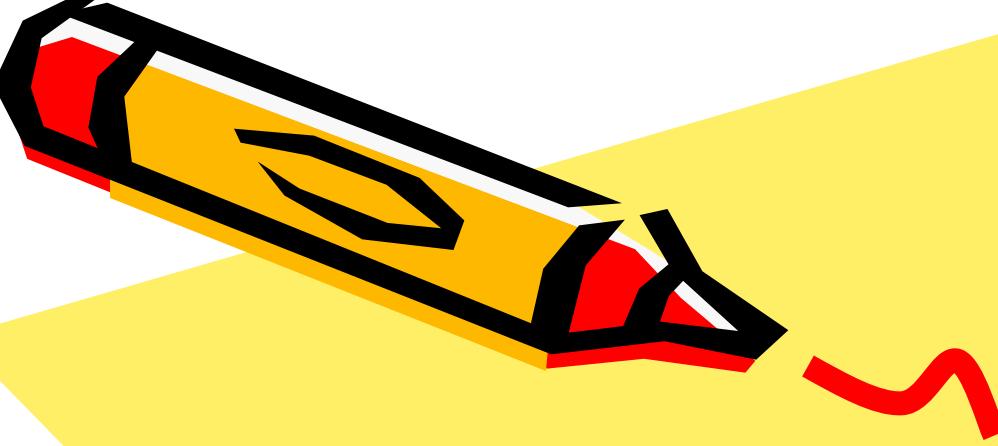
X	Y	Z	X'	XY	X'Z	XY+X'Z	Designation
0	0	0	1	0	0	0	m0
0	0	1	1	0	1	1	m1
0	1	0	1	0	0	0	m2
0	1	1	1	0	1	1	m3
1	0	0	0	0	0	0	m4
1	0	1	0	0	0	0	m5
1	1	0	0	1	0	1	m6
1	1	1	0	1	0	1	m7

$$F(x, y, z) = XY + X'Z = m0 \ m2 \ m4 \ m5 = \Pi(0, 2, 4, 5)$$

Means at these points results = zeros





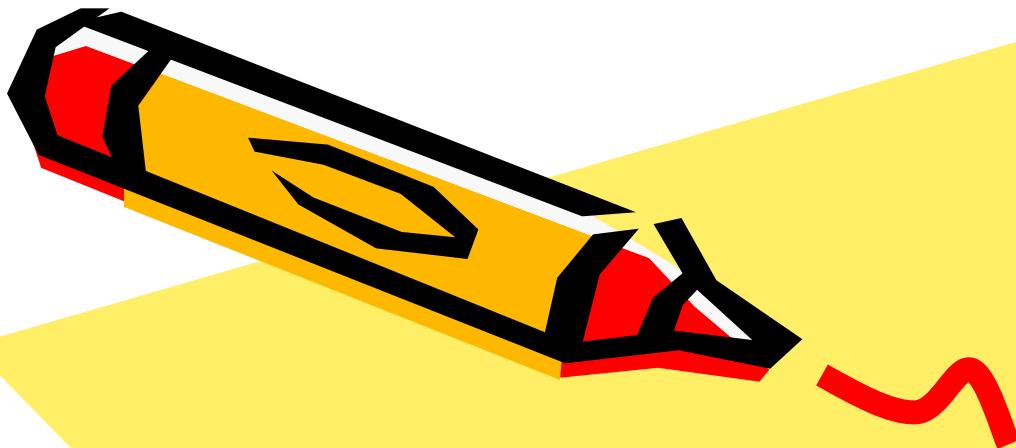


Logical Design

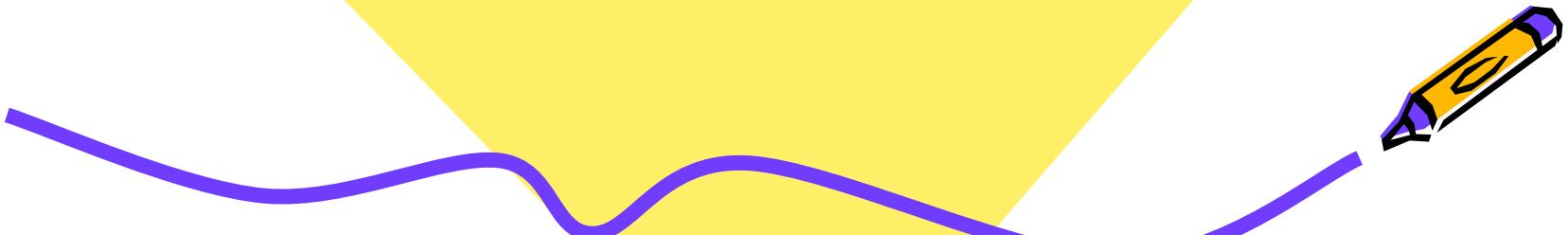
CS 221

Prof.Dr. Mohamed Osama Khozium



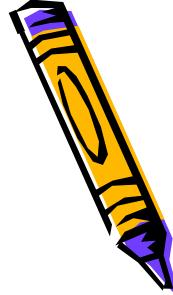


Karnaugh Maps





Karnaugh Maps

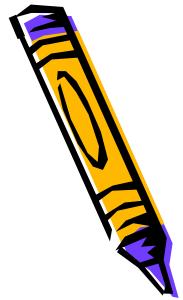


- Boolean expressions may be simplified by using algebraic operations
- But there is not set method to predict the steps to take
- That means not amenable to automated techniques
- Karnaugh Maps to the rescue
- Generates simplified expressions that are in SOP or POS form
- Produce two-level implementation with a minimum number of gates and a minimum number of inputs to the gates
- Sometimes two or more expressions that satisfy the simplification criteria

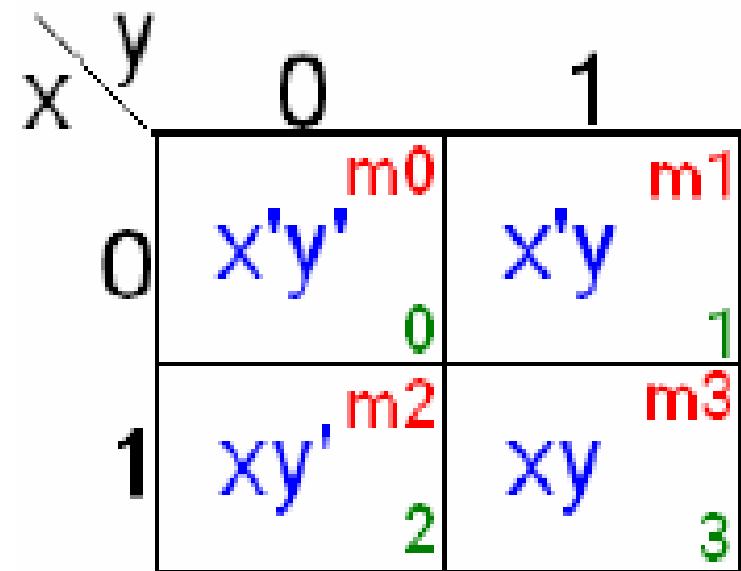




Two-Variable Map



X	Y	Minterms	
0	0	$x'y'$	m0
0	1	$x'y$	m1
1	0	xy'	m2
1	1	xy	m3

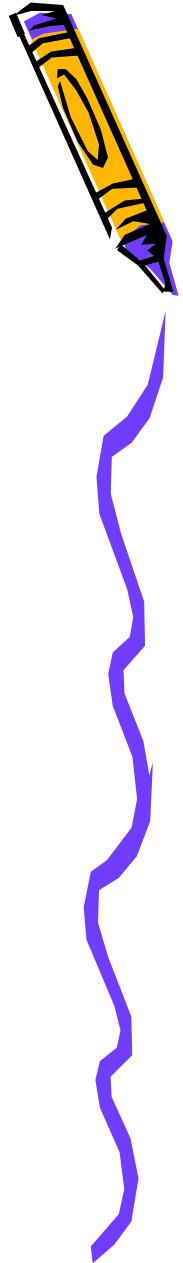




$$F(x,y) = xy + x'y$$

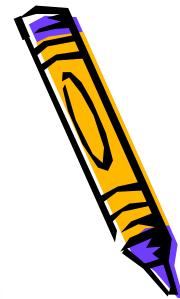
	x	y	
0		0	
1		0	
0	1	1	1

$$F(x,y) = y$$





Three Variables Map



A Karnaugh map for three variables (x, y, z) with four minterms labeled m0 through m7.

		yz	x	00	01	11	10
		0	0	$x'y'z'$ 0	$x'y'z$ 1	$x'yz$ 3	$x'yz'$ 2
		1	1	$xy'z'$ 4	$xy'z$ 5	xyz 7	xyz' 6
m0	m1	m3	m2	m4	m5	m7	m6



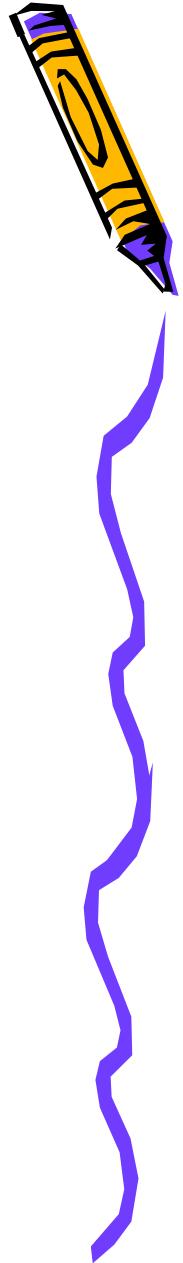


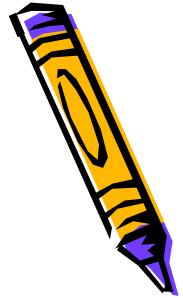
Simplify the following Boolean function

$$F(x,y,z) = \Sigma(3,4,6,7)$$

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

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





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

Express it in sum of minterms

Find the minimal sum of products expression

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

$$F(x,y,z) = \Sigma(1,2,3,5,7)$$

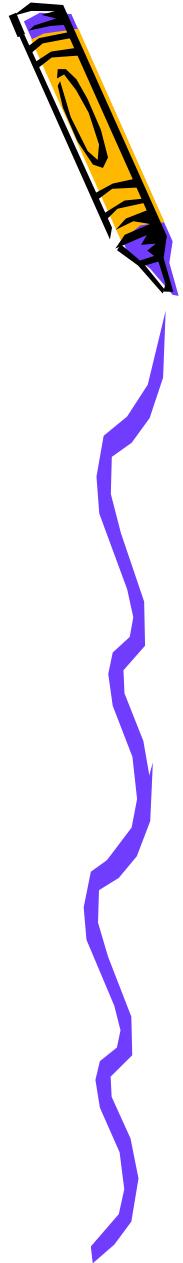
$$F(X,Y,Z) = Z + X'Y$$

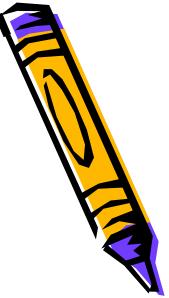




Four Variables Map

		yz	00	01	11	10
		wx	m0 $w'x'y'z'$ 0	m1 $w'x'y'z$ 1	m3 $w'x'yz$ 3	m2 $w'x'yz'$ 2
		00	m4 $w'xy'z'$ 4	m5 $w'xy'z$ 5	m7 $w'xyz$ 7	m6 $w'xyz'$ 6
		11	m12 $wxy'z'$ 12	m13 $wxy'z$ 13	m15 $wxyz$ 15	m14 $wxyz'$ 14
		10	m8 $wx'y'z'$ 8	m9 $wx'y'z$ 9	m11 $wx'yz$ 11	m10 $wx'yz'$ 10





Simplify the following Boolean function

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

wx\yz	00	01	11	10
00	1	1		1
01	1	1		1
11	1	1		1
10	1	1		

$$F(w,x,y,z) = y' + w'z' + xz'$$



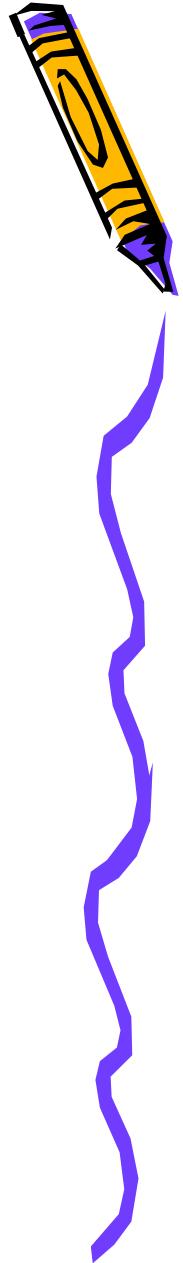


Simplify the following Boolean function

$$F(w,x,y,z) = \Sigma(0,2,3,5,7,8,9,10,11,13,15)$$

wx \ yz	00	01	11	10
00	1		1	1
01		1	1	
11		1	1	
10	1	1	1	1

$$F(w,x,y,z) = wx' + yz + xz + x'z'$$





Don't Care Conditions

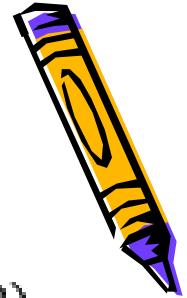
Simplify the following Boolean function

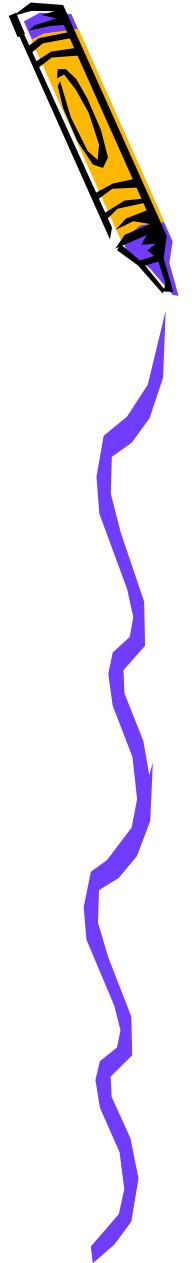
$$F(w,x,y,z) = \Sigma(1,3,7,11,15)$$

Which has the don't care conditions $d(w,x,y,z) = \Sigma(0,2,5,8)$

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

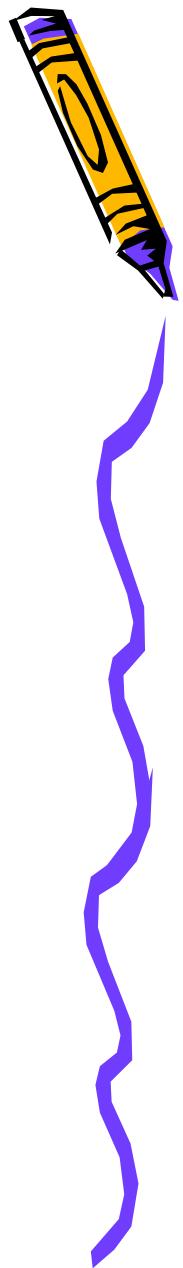
$$F(w,x,y,z) = w'x' + yz$$





Thank you





Digital Logic and Design

Karnaugh Map

Recap: Boolean Algebra

- Simplification of Boolean expressions are done through Boolean algebra.
- The method wasn't effective.
- Solving a problem through Boolean algebra was a bit tricky
- Need skill of applying rules and law.
- Doesn't guarantee simplest form of expression

Karnaugh Map

- K-map provides a systematic method
 - For simplifying Boolean expressions and minimizing expressions.
 - If properly used, will produce the simplest SOP and POS.
 - Similar to truth table, presents all possible values of input variables and resulting output for each value.

Karnaugh Map

- Karnaugh map is an array of cells in which each cell represents a binary value of input variables.
- The cells are managed in a way so that simplification of a given expression is simply a matter of properly grouping the cells.
- Karnaugh maps can be used for expression with two, three, four and five variables, but we will discuss only 3-variable and 4-variable situations to illustrate the principles.

Karnaugh Map

The number of cells in a **Karnaugh map** is equal to the total number of possible input variable combinations as is the number of rows in a truth table.

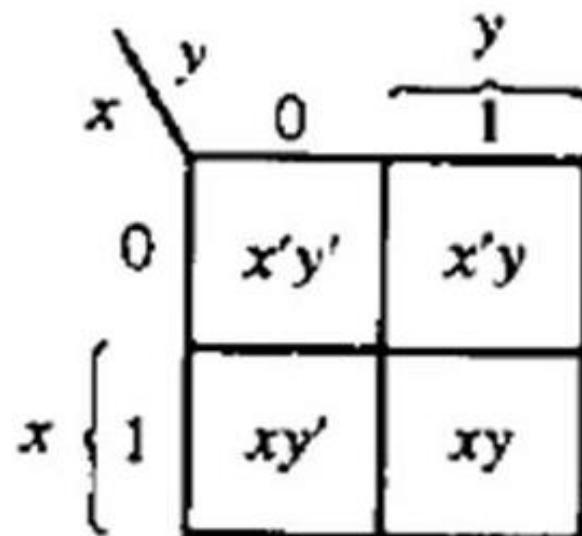
- For two variables, the number of cells is $2^2 = 4$.
- For three variables, the number of cells is $2^3 = 8$.
- For four variables, the number of cells are $2^4 = 16$.

2-variable K-map

A two variable has four minterms, hence it has 4 squares one for each term, as shown below,

m_0	m_1
m_2	m_3

(a)



(b)

3-variable karnaugh map

- The 3-variable karnaugh map is an array of eight cells.
- In this case A, B and C are used for the variables although other letters could be used.
- Binary values of A is along the left side and the value of B and C is across the top.

The 3-variable Karnaugh map

- The value of a given cell is the binary values of A at the left in the same row combined with the value of B and C at the top in the same column.
- For example: the cell in the upper left corner has a binary value of 000 and the cell in the lower right corner has a binary value of 110.

3-Variable K-map

AB\C	0	1
00	0	1
01	2	3
11	6	7
10	4	5

A\BC	00	01	11	10
0	0	1	3	2
1	4	5	7	6

The 4-variable karnaugh map

- The 4-variable karnaugh map is an array of sixteen cells
- Binary values of A and B are along the left side and the values of C and D are across the top.
- The value of a given cell is the binary value of A and B at the left in the same row combined with the binary values of C and D at the top of the same column.

4-Variable K-map

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

Grouping & Adjacent Cells

- K-map is considered to be wrapped around
- All sides are adjacent to each other
- Adjacent cells which has 1's in SOP can be grouped together in 2's power.
 - 2 adjacent cells can be grouped (pair)
 - 4 adjacent cells can be grouped (Quads)
 - 8 adjacent cells can be grouped (octets)
- Groups can be row, column, square or rectangular.
- Groups of diagonal cells are not allowed

Mapping of Standard SOP expression

- Selecting n-variable K-map
- 1 marked in cell for each minterm

Mapping of Standard SOP expression

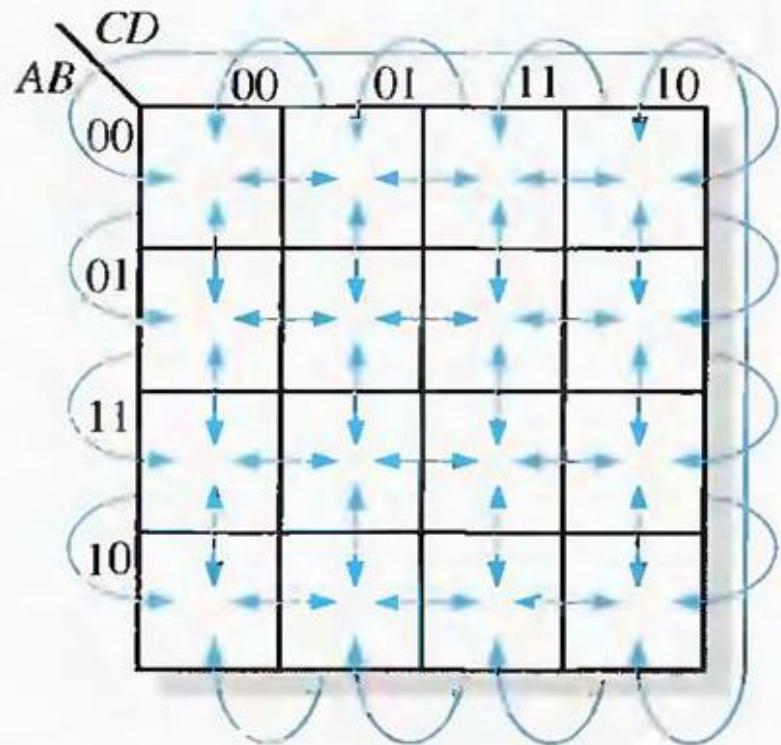
- SOP expression $ABC\bar{C} + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C}$

A\BC	00	01	11	10
0	0	0	0	1
1	1	0	0	1

Cell Adjacency

- The cells in a karnaugh map are arranged so that there is only a single-variable change between adjacent cells.
- Adjacency is defined by a single variable change.
- Cells with values that differ by more than one variable are not adjacent.

Cell Adjacency



Adjacent cells on a Karnaugh map are those that differ by only one variable. Arrows point between adjacent cells.

Continued (simplification through K maps)

- Karnaugh map is used for simplifying Boolean expressions to their minimum form.
- A minimized SOP expression contains the fewest possible terms with the fewest possible variables per term.
- Generally a minimum SOP expression can be implemented with fewer logic gates than a standard expression.

Mapping a Standard SOP Expression

- For an SOP expression in standard form, a 1 is placed on the **Karnaugh map** for each product term in the expression.
- Each 1 is placed in a cell corresponding to the value of a product term. For example, for the product term $AB'C$, a 1 goes in the 101 cell on a 3-variable map.

Mapping SOP expression

When an SOP expression is completely mapped, there will be a number of 1's on the Karnaugh map equal to the number of product terms in the standard SOP expression.

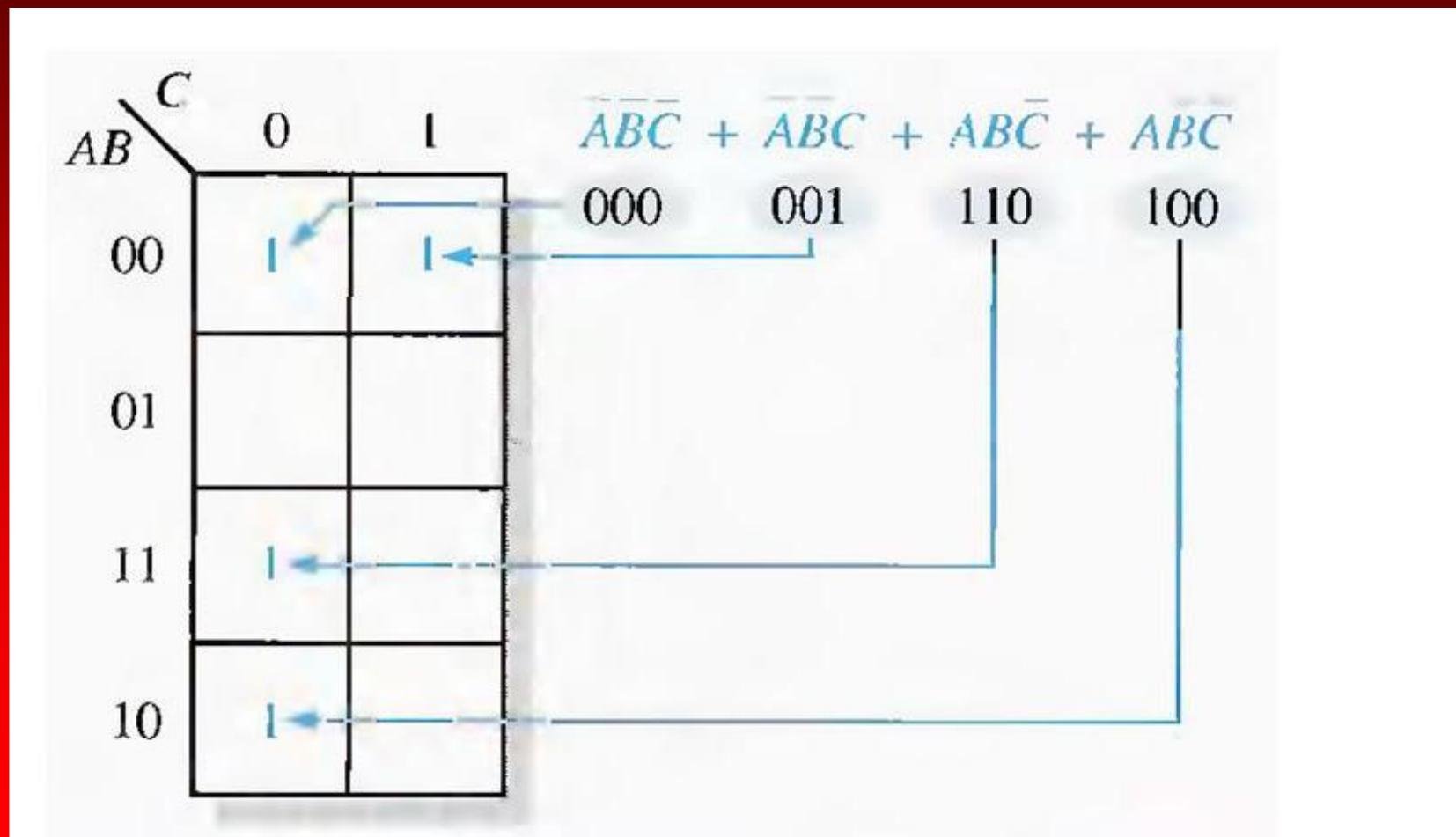
The cells that do not have a 1 are the cells for which the expression is 0.

Usually, when working with SOP expressions, the 0's are left off the map.

The following steps are used for mapping process.

- **Step 1-** Determine the binary value of each product term in the standard SOP expression. After some practice, you can usually do the evaluation of terms mentally.
- **Step 2.** As each product term is evaluated, place a 1 on the Karnaugh map in the cell having the same value as the product term.

Mapping entries in K-map



Class activity

Map the following standard SOP expression on a Karnaugh map:

$$\mathbf{A'B'C + A'BC' + ABC' + ABC}$$

Solution

$$A'B'C + A'BC' + ABC' + ABC$$

- Solution:

$$A'B'C + A'BC' + ABC' + ABC$$

$$001 + 010 + 110 + 111$$

A \ BC	00	01	11	10
0	0	1	0	1
1	0	0	1	1

- Map the standard SOP expression

$$A'BC + AB'C + AB'C'$$

Self Assessment

Problem:

- Map the standard SOP expression

$$\mathbf{A' BC + AB' C + AB' C'}$$

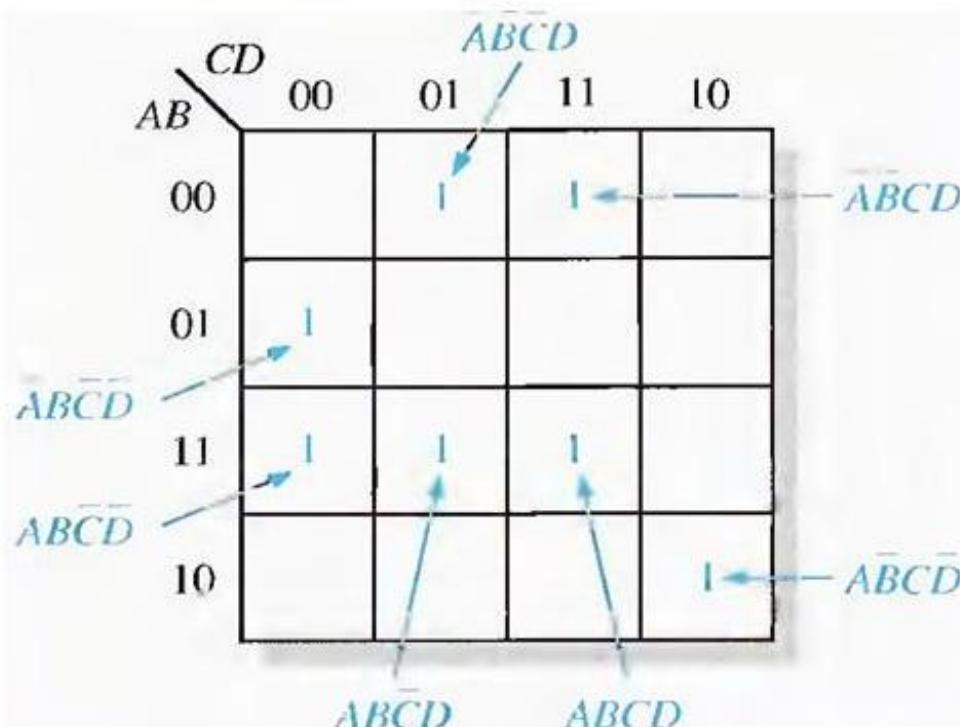
on a Karnaugh map just in 3 minutes, hurry up.



Map the following standard SOP expression on a Karnaugh map:

- $A' B' C D + A' B C' D' + A B C' D + A B C D + A B C' D'$
 $+ A' B' C' D + A B' C D'$

Solution:



Self Assessment

Problem:

- Map the following standard SOP expression on a Karnaugh map:

$$A' BCD' + ABCD' + ABC' D' + ABCD$$

Map the above expression on a K-Map in just 2 Minutes, Hurry Up.



Mapping a non standard SOP expression

A Boolean expression must first be in standard form before you use a **Karnaugh map**.

If an expression is not in standard form, then it must be converted to standard form.

It can also be done through numerical expansion.
Numerical expansion is probably the most efficient approach.

Numerical Expansion of a Nonstandard Product Term:

- Recall that a nonstandard product term has one or more missing variables.
- Assume that one of the product term in a certain 3-variable SOP expression is AB' .
- First, write the binary value of the two variables and attach a 0 for the missing variable C' : 100.
- Next, write the binary value of the two variables and attach a 1 for the missing variable C : 101

Mapping a non standard SOP expression

As another example, assume that one of the product terms in a 3-variable expression is B.

This term can be expanded numerically to standard form as follows.

Write the binary value of the variable; then attach all possible values for the missing variables A and C as follows:

B
010
011
110
111

Note:

The four resulting binary numbers are the values of the standard SOP terms are $A' BC'$, $A' BC$, ABC' , and ABC .

Map the following SOP expression on a Karnaugh map:

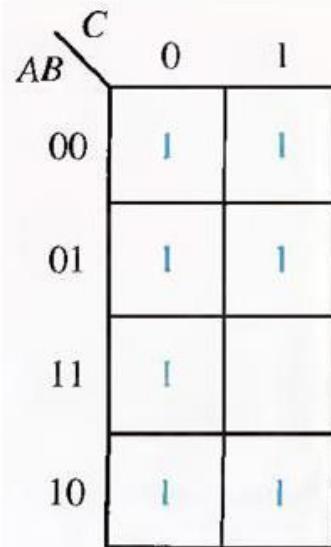
$$\mathbf{A' + AB' + ABC'}$$

The first term is missing two variables, the second term is missing one variable, and the third term is standard.

Numerical expansion of the given expression can be done as follows

Mapping non standard SOP into K-map

Now map each of the resulting binary values by placing a 1 in the appropriate cell of the 3-variable Karnaugh map



$$\begin{array}{c} \bar{A} + A\bar{B} + ABC \\ \hline 000 & 100 & 110 \\ 001 & 101 & \\ 010 & & \\ 011 & & \end{array}$$

Problem

Map the SOP expression
 $BC + A' C'$

on a Karnaugh map.

Map the following SOP expression on a Karnaugh map:

$$B' C' + AB' + ABC' + AB' CD' + A' B' C' D + AB' CD$$

The SOP expression is obviously not in standard form because each product term does not have four variables.

The first and second terms are both missing two variables, the third term is missing one variable, and the rest of the terms are standard.

Continued...

First expand the terms by including all combinations of the missing variables numerically as follows:

$\bar{B}\bar{C}$	$A\bar{B}$	+	ABC	$A\bar{B}CD$	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}\bar{B}CD$
0000	1000		1100	1010	0001	1011
0001	1001		1101			
1000	1010					
1001	1011					

1. Map each of the resulting binary values by placing a 1 in the appropriate cell of the 4-variable Karnaugh map.
2. Notice that some of the values in the expanded expression are redundant.

Continued...

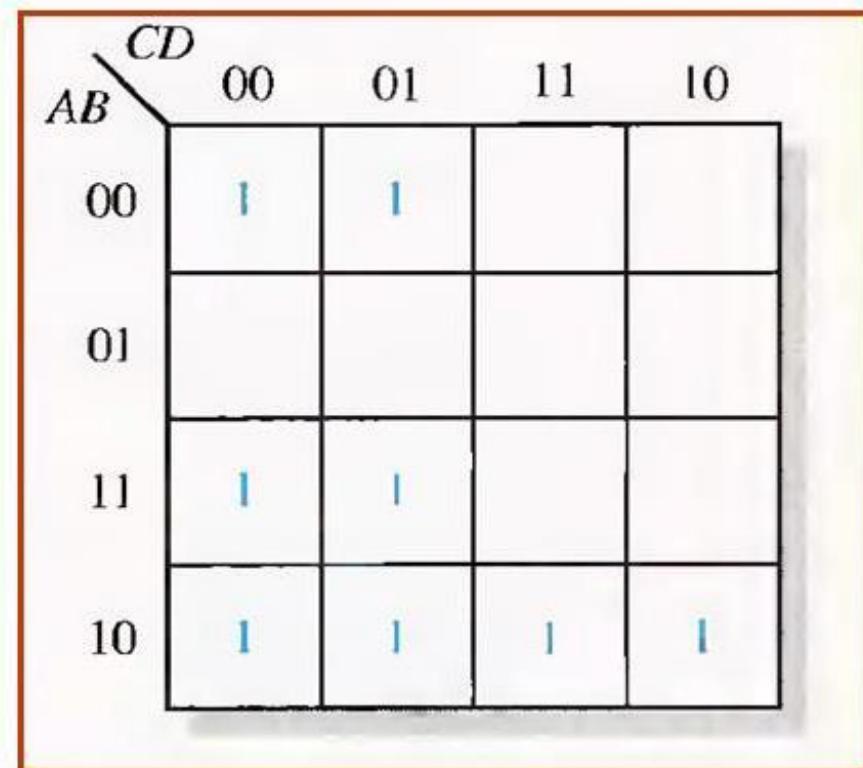
After mapping the expression the resulting 4 variable k map look like this,

Related Problem:

Map the expression

$$A + C' D + ACD' + A' BCD'$$

on a Karnaugh map?



K-Map Simplification

After an SOP expression has been mapped, a minimum SOP expression is obtained by grouping the Is and determining the minimum SOP expression from the map.

You can group Is on the Karnaugh map according to the following rules by enclosing those adjacent cells containing 1s.

The goal is to maximize the size of the groups and to minimize the number of groups.

Steps for Grouping

- A group must contain either 1, 2, 4, 8, or 16 cells, which are all powers of two. In the case of a 3-variable map, $2^3 = 8$ cells is the maximum group.
- Each cell in a group must be adjacent to one or more cells in that same group. but all cells in the group do not have to be adjacent to each other.
- Always include the largest possible number of 1's in a group in accordance with rule 1.
- Each 1 on the map must be included in at least one group. The Is already in a group can be included in another group as long as the overlapping groups include noncommon 1's.

For Example

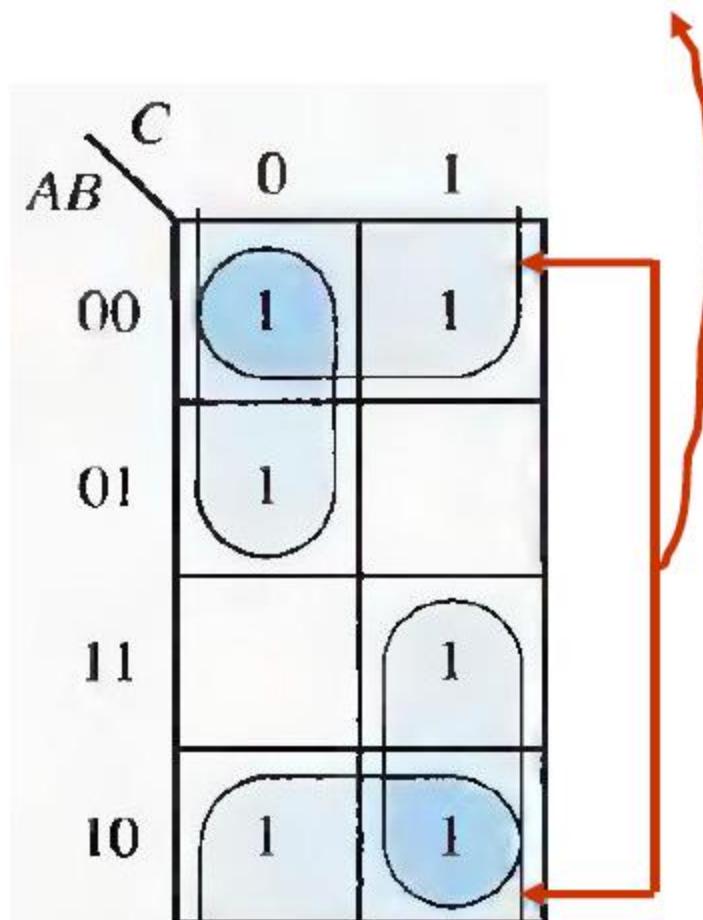
AB	C	0	1
00		1	
01			1
11		1	1
10			

AB	C	0	1
00		1	
01			1
11		1	1
10			

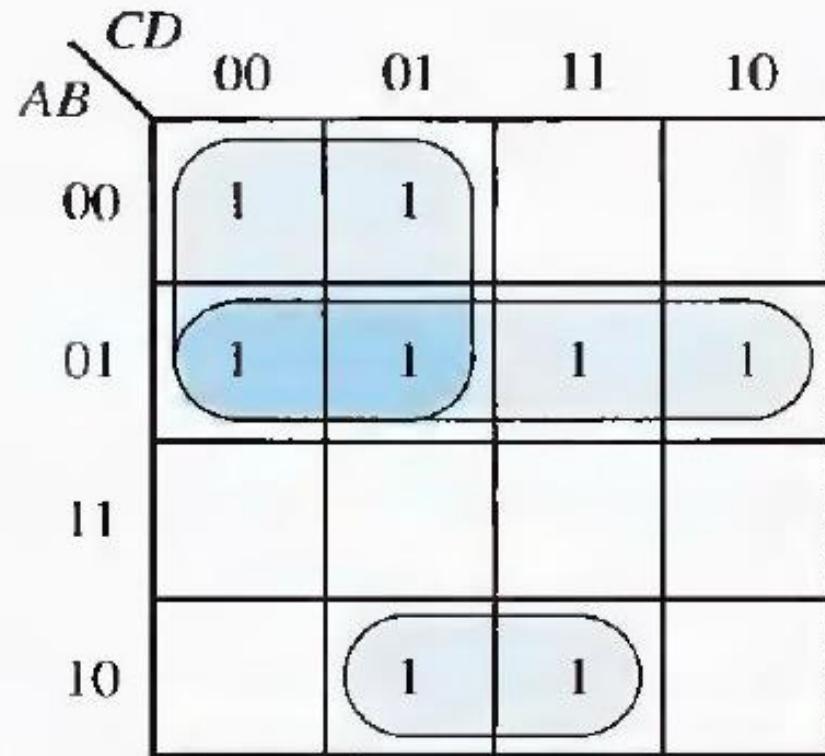
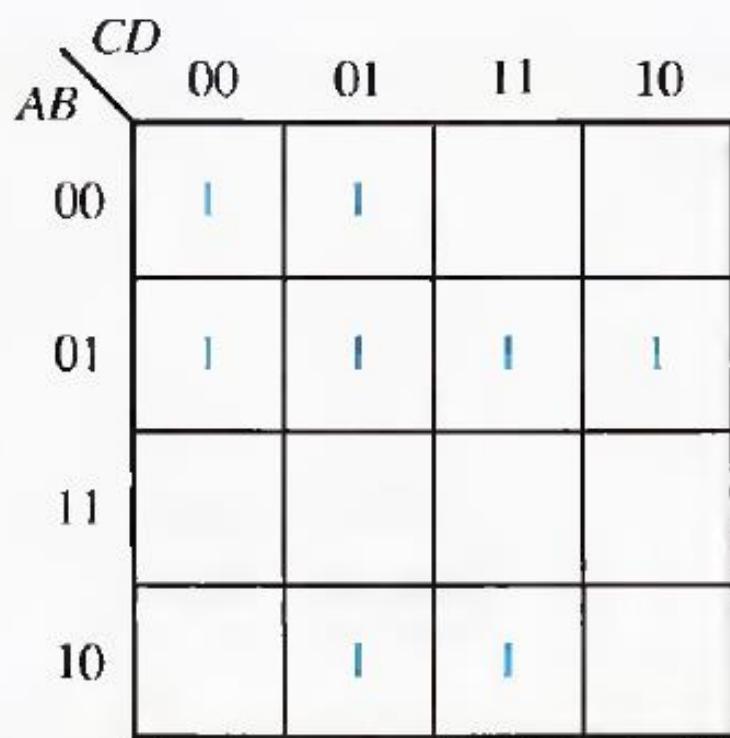
Example

Wrap around adjacency

	AB	C
	0	1
00	1	
01	1	
11		1
10	1	1



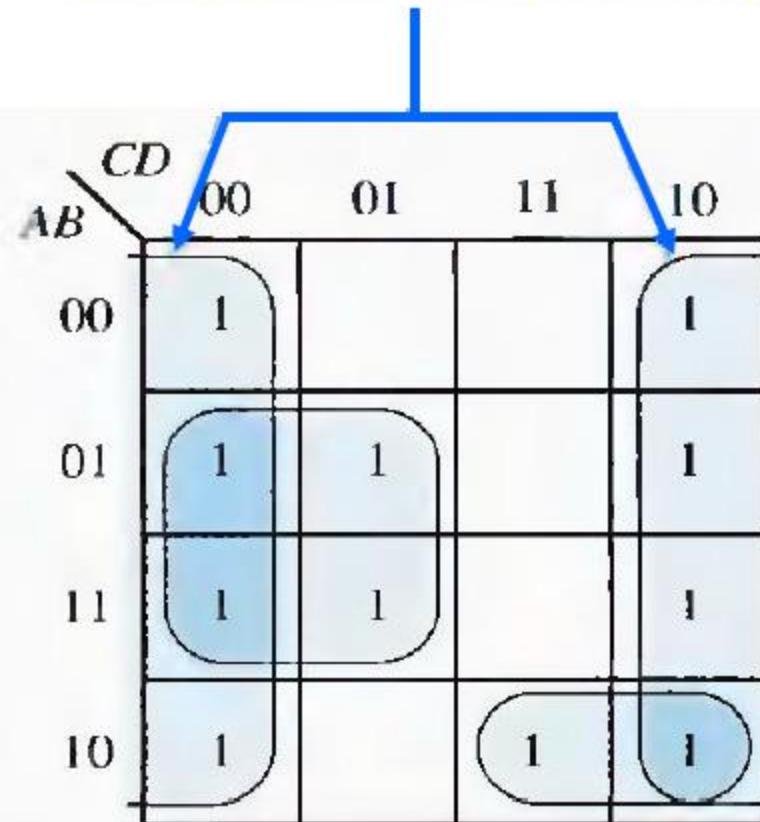
K-Maps



K-Maps

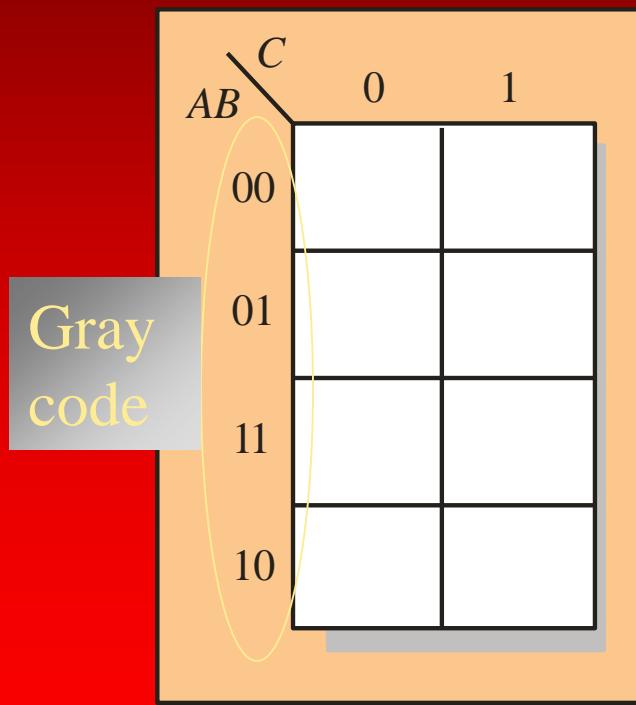
Wrap around adjacency

$AB \backslash CD$	00	01	11	10
00	1			1
01	1	1		1
11	1	1		1
10	1		1	1



Karnaugh maps

Cells are usually labeled using 0's and 1's to represent the variable and its complement.



The numbers are entered in gray code, to force adjacent cells to be different by only one variable.

Ones are read as the true variable and zeros are read as the complemented variable.

Determining the Minimum SOP Expression from the Map.

- When all the 1's representing the standard product terms in an expression are properly mapped and grouped, the process of determining the resulting minimum SOP expression begins.

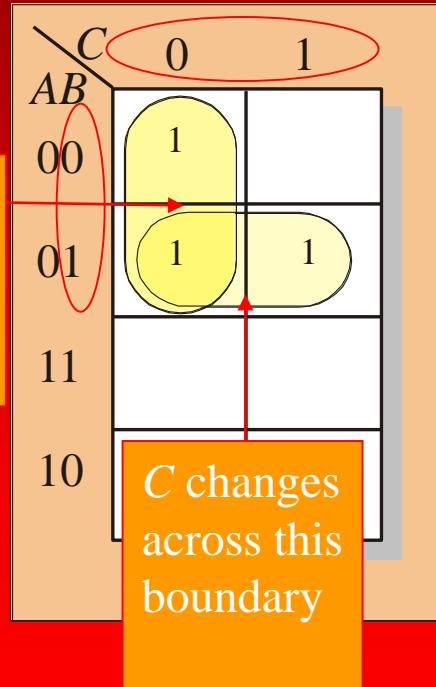
Rules

- Group the cells that have 1's. Each group of cells containing 1's creates one product term composed of all variables that occur in either form within the group.
- Variables that occur both uncomplemented and complemented within the group are eliminated. These are called contradictory variables.

Karnaugh maps

K-maps can simplify combinational logic by grouping cells and eliminating variables that change.

Example Group the 1's on the map and read the minimum logic.



B changes
across this
boundary

C changes
across this
boundary

Solution

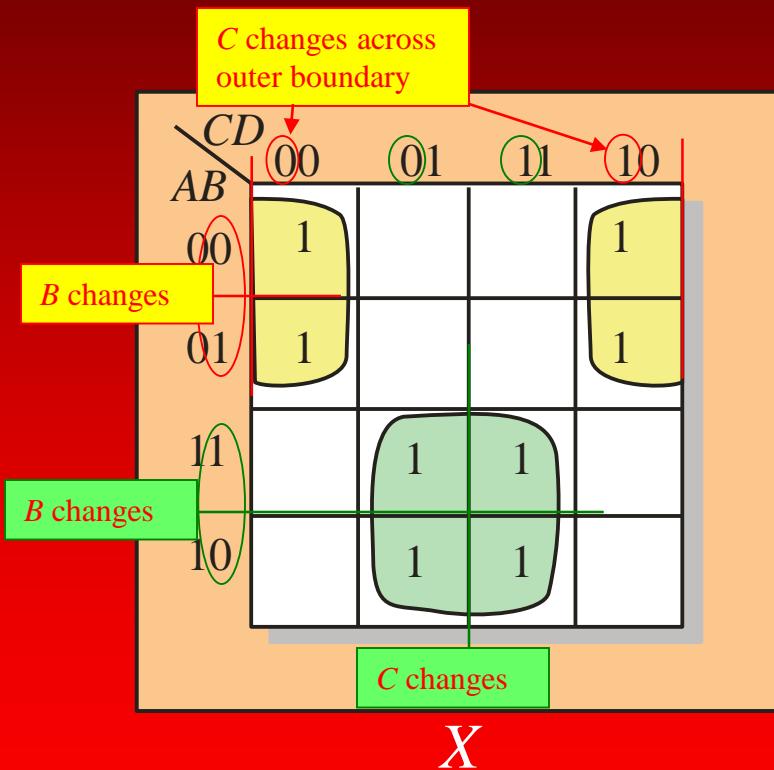
1. Group the 1's into two overlapping groups as indicated.
2. Read each group by eliminating any variable that changes across a boundary.
3. The vertical group is read $\bar{A}\bar{C}$.
4. The horizontal group is read $\bar{A}B$.

$$X = \bar{A}\bar{C} + \bar{A}B$$

Karnaugh maps

Example

Group the 1's on the map and read the minimum logic.



Solution

1. Group the 1's into two separate groups as indicated.
2. Read each group by eliminating any variable that changes across a boundary.
3. The upper (yellow) group is read as $\bar{A}\bar{D}$.
4. The lower (green) group is read as AD .

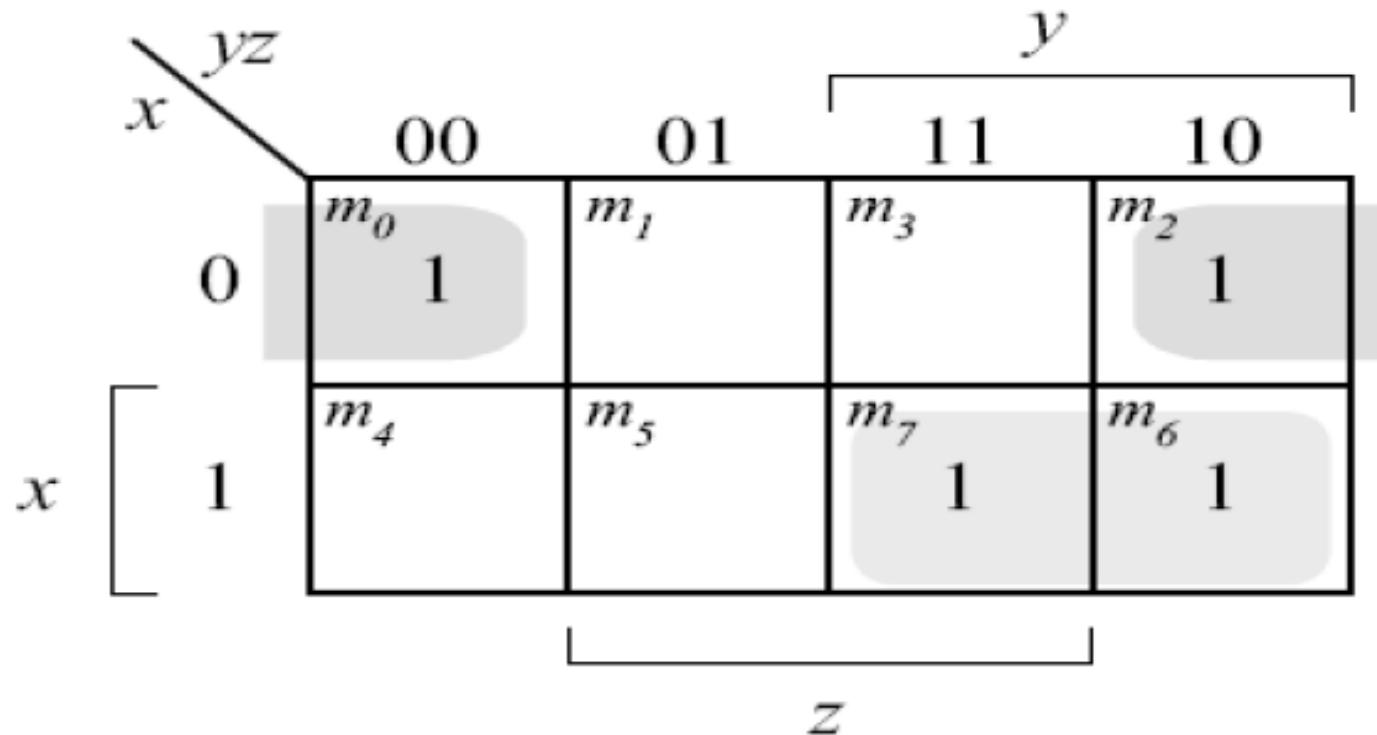
$$X = \bar{A}\bar{D} + AD$$

Related Example

- Minimize the following SOP expression,
- $F(x, y, z) = \Sigma(0,2,6,7)$
- $F(x, y, z) = \Sigma(0,2,3,4,6)$
- $F(x, y, z) = \Sigma(0,1,2,3,7)$
- $F(x, y, z) = \Sigma(3,5,6,7)$
- $F(x, y, z) = \Sigma(0,1,5,7)$
- $F(x, y, z) = \Sigma(0,1,6,7)$
- $F(x, y, z) = \Sigma(1,2,3,6,7)$

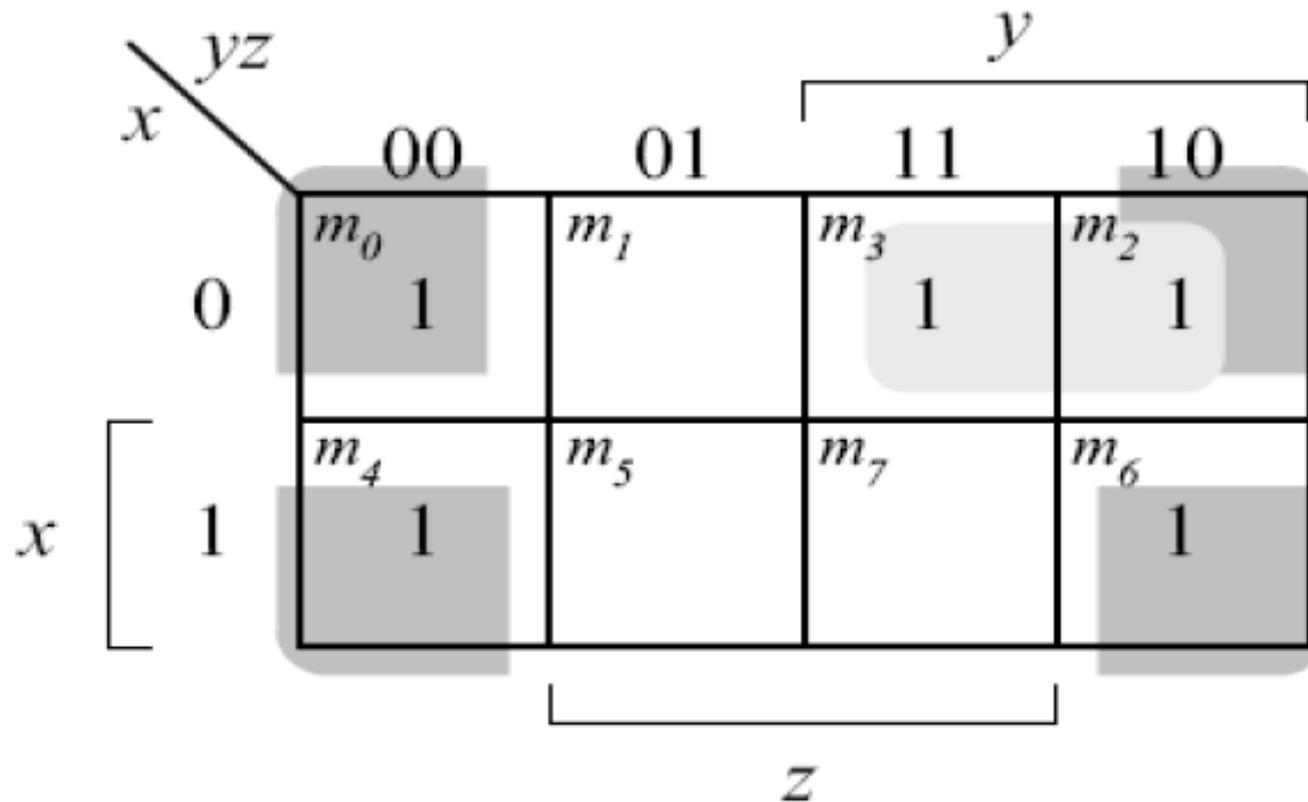
SOLUTION(1-7)

$$F(x, y, z) = \Sigma(0, 2, 6, 7) \rightarrow 1$$



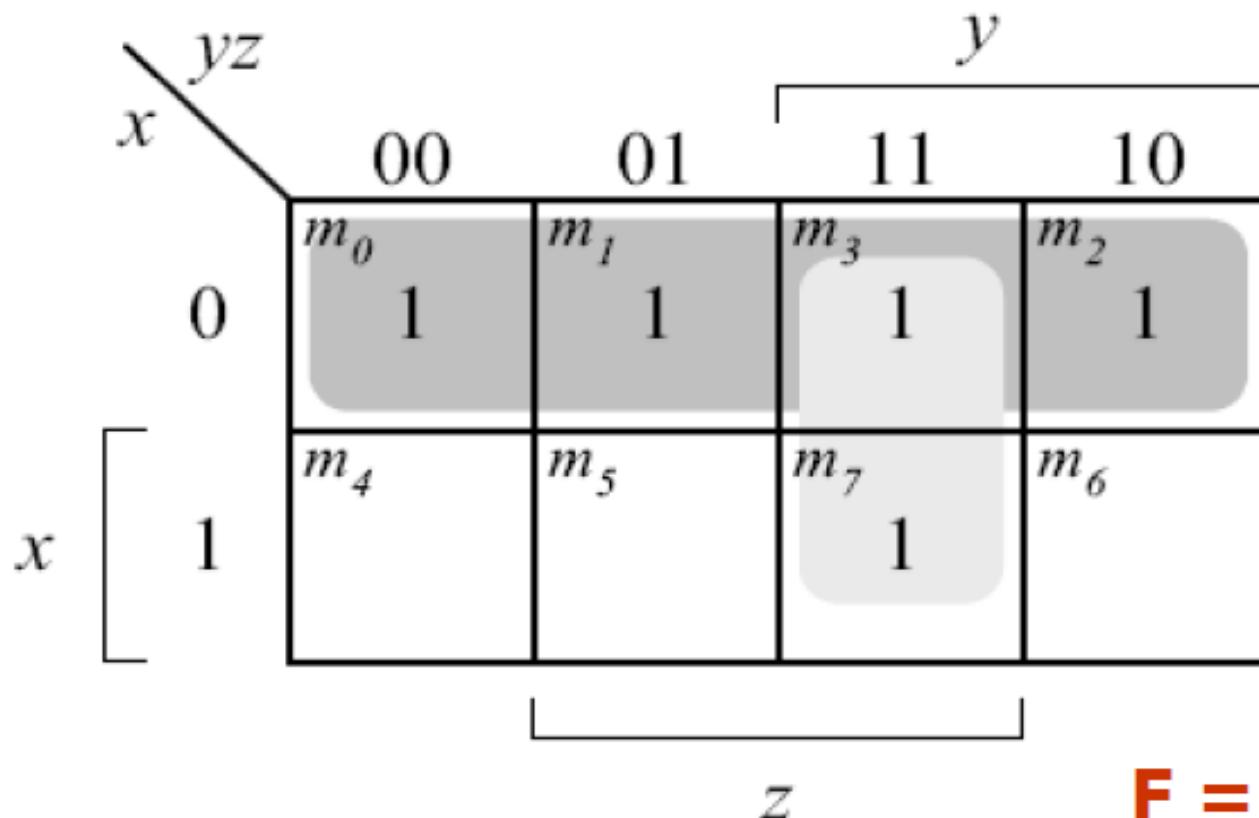
$$F = xy + x'z'$$

$$F(x, y, z) = \Sigma(0, 2, 3, 4, 6) \quad \rightarrow \quad 2$$

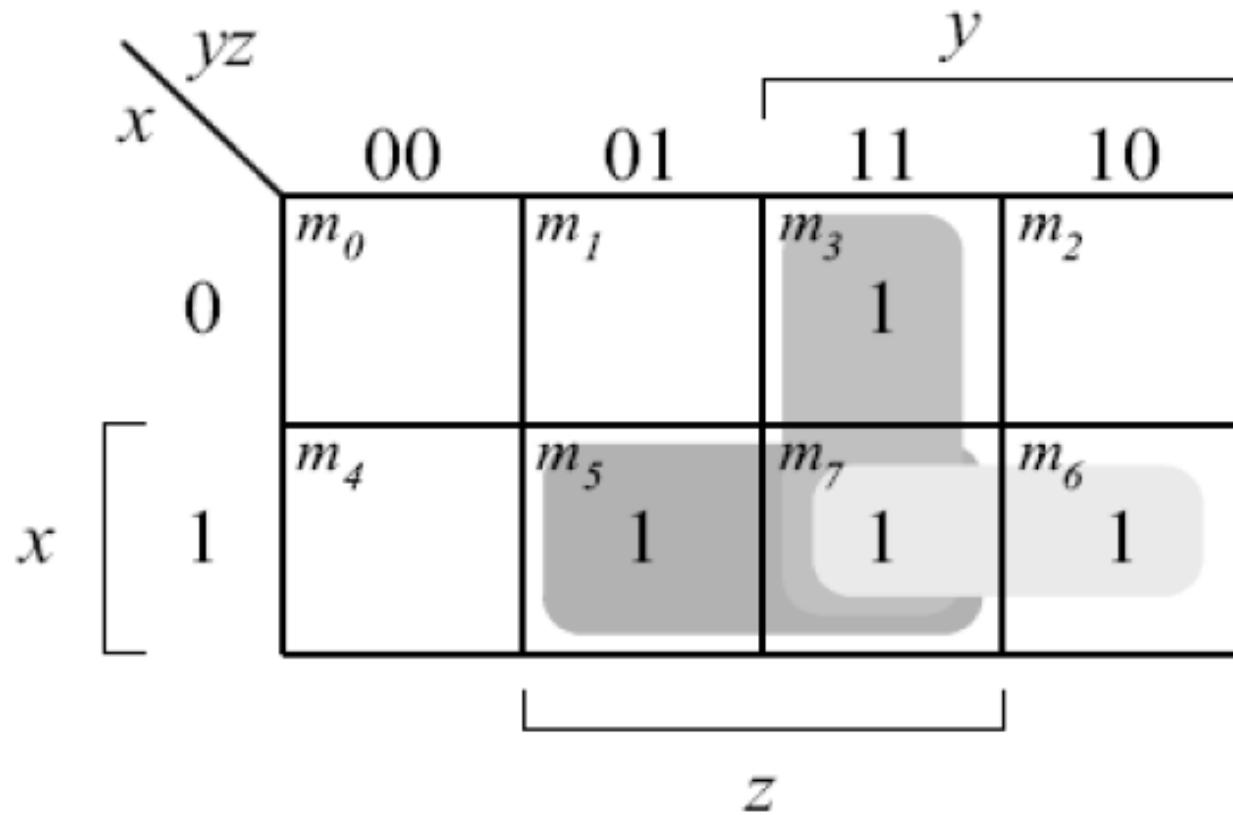


$$F = z' + x'y$$

$$F(x, y, z) = \Sigma(0, 1, 2, 3, 7) \quad \rightarrow \quad 3$$

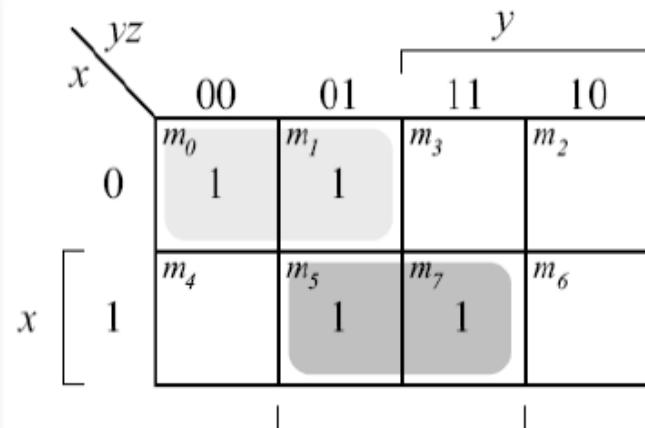


$$F(x, y, z) = \Sigma(3, 5, 6, 7) \quad \rightarrow \quad 4$$

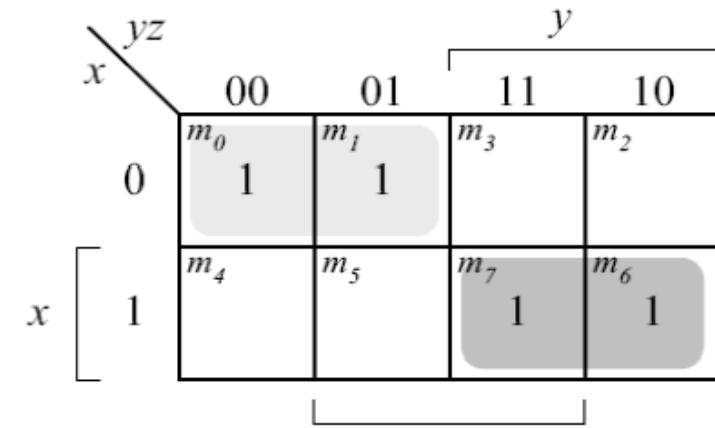


$$F = xy + xz + yz$$

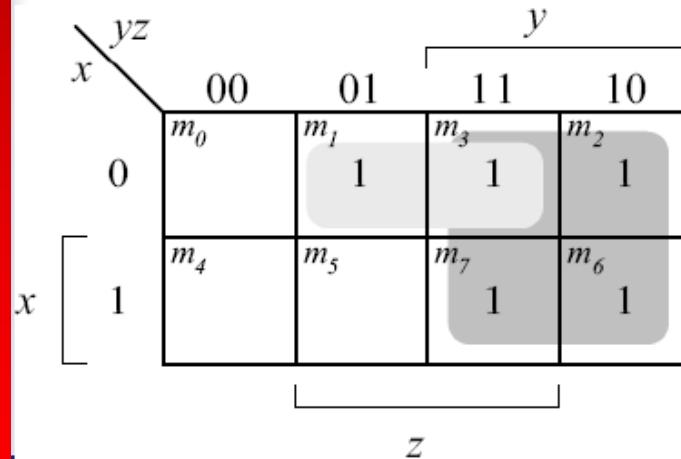
Solution of 5, 6, 7



$$5. F = \overline{x}^z y' + x z$$



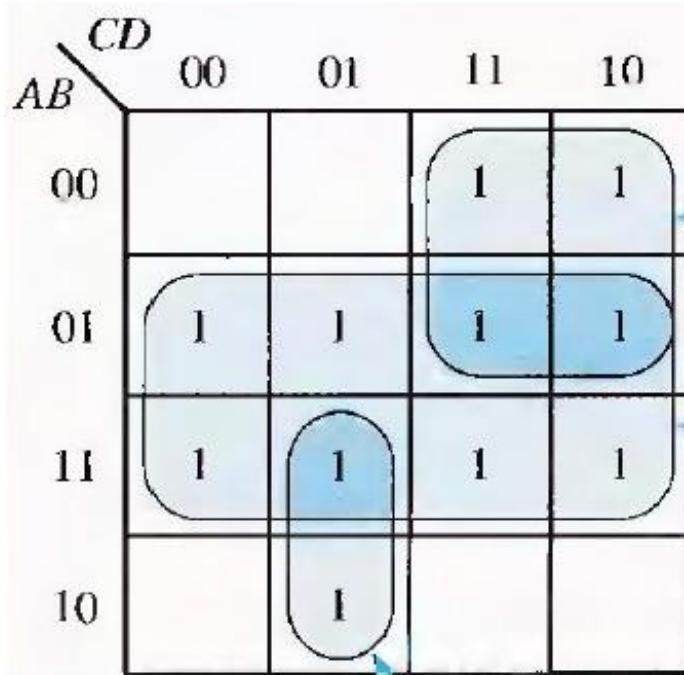
$$6. F = \overline{x}^z y' + x y$$



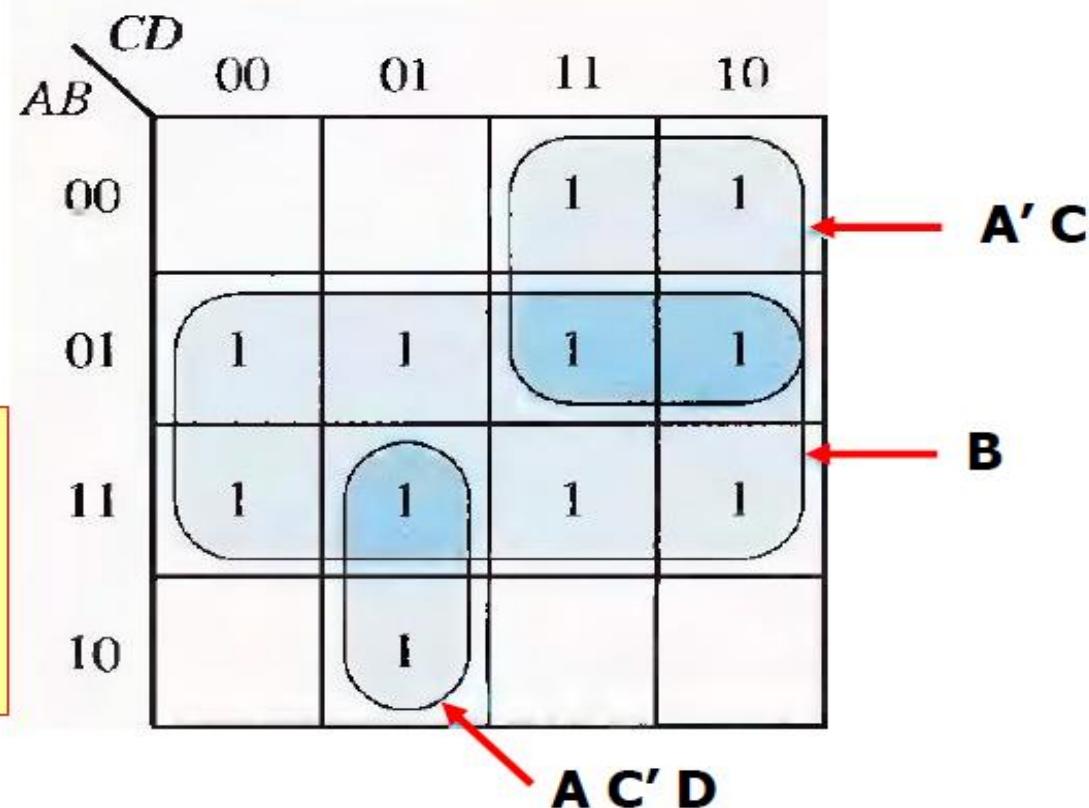
$$7. F = y + x^z$$

Example

- Determine the product terms for the Karnaugh map given and write the resulting minimum SOP expression?



Solution



The resulting minimum SOP expression is,

$$B + A' C + AC' D$$

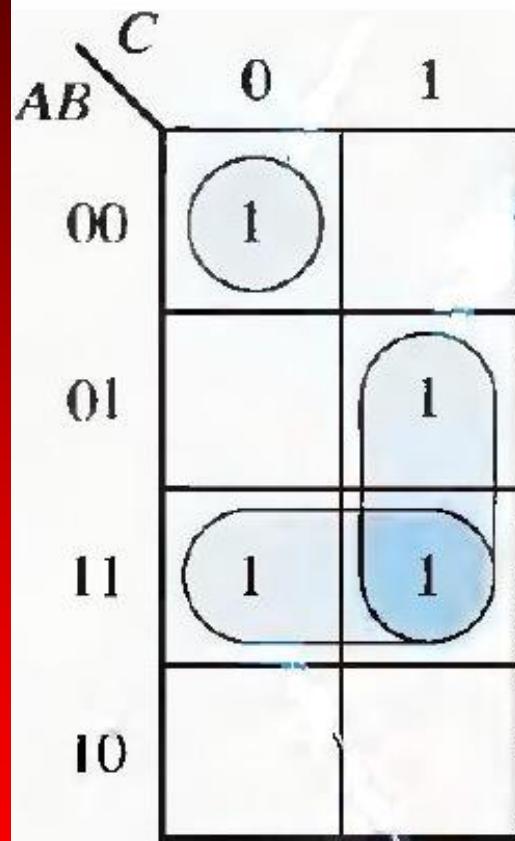
the sum of these product terms:

Self Assessment

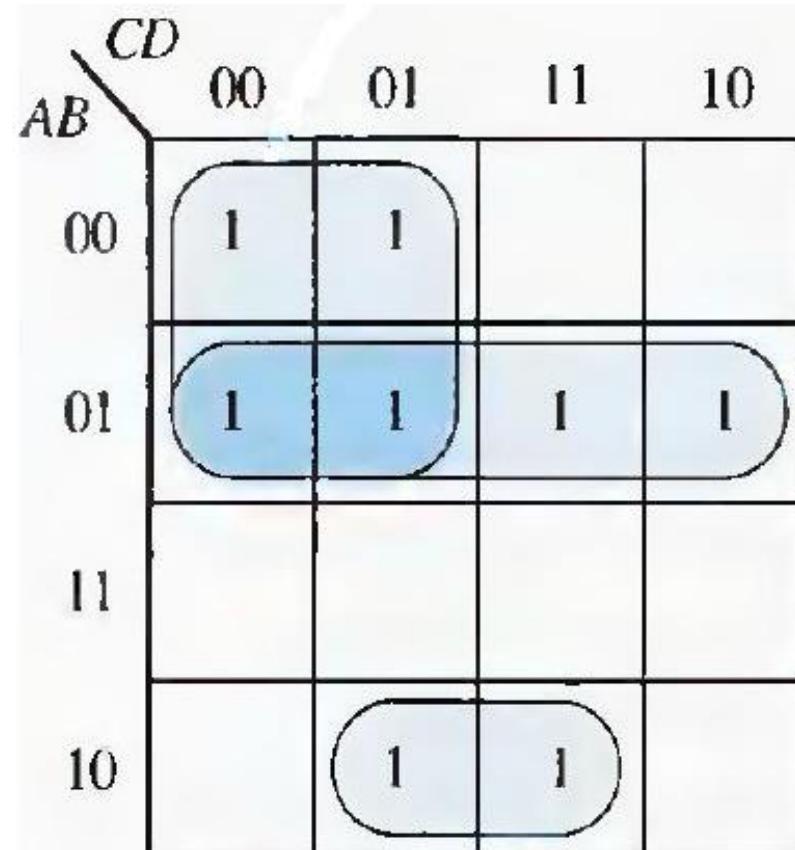
Problem: For the Karnaugh map on the previous slide, add a 1 in the lower right cell (1010) and determine the resulting SOP expression.



Example 1

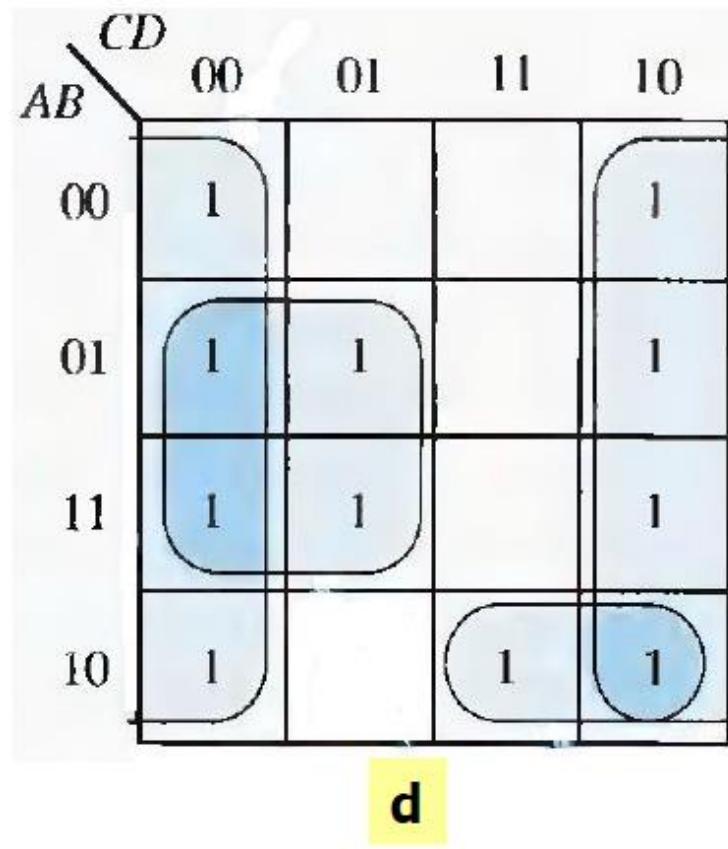
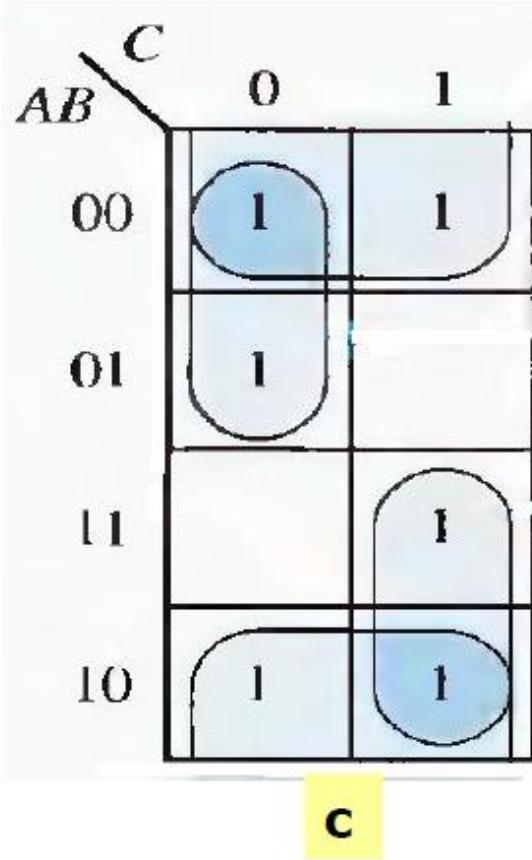


a



b

Example 2



Solution

The resultant SOP expression against a, b, c and d are,

- a. $AB + BC + A' B' C'$
- b. $A' B + A' C' + AB' D$
- c. $B' + A' C' + AC$
- d. $D' + AB' C + BC'$

Class Assignment

Using k-map simplify the following expression,

$$F(A,B,C,D) = \Sigma(0,2,3,4,6,8,10,11,12,14)$$

Just do it in 3min. Only.



The resulting minimum SOP expression is

$$D' + B' C$$

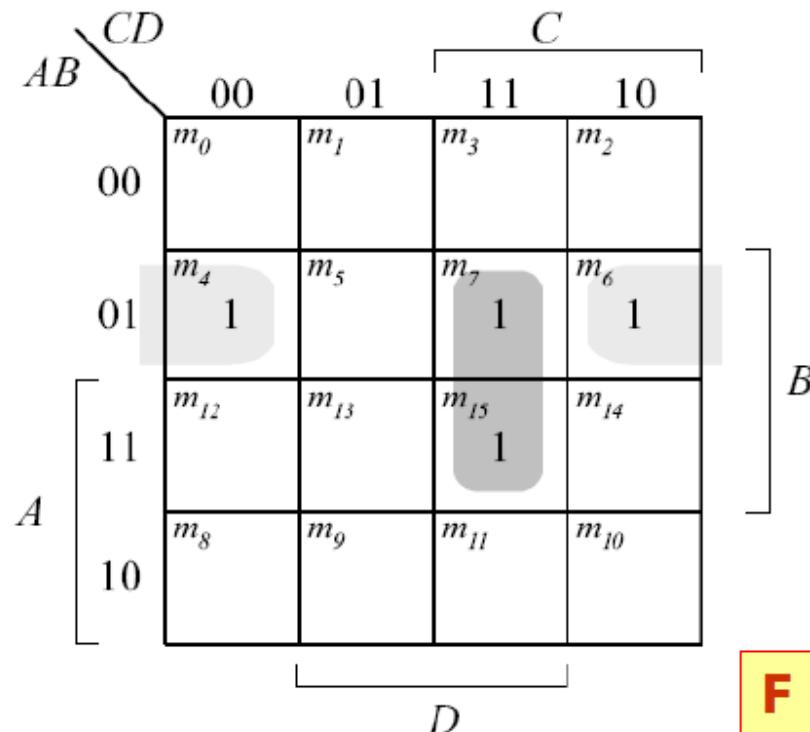
Examples

Find the minimized SOP expression against the following,

1. $F(A,B,C,D) = \Sigma(4,6,7,15)$
2. $F(A,B,C,D) = \Sigma(3,7,11,13,14,15)$
3. $F(A,B,C,D) = \Sigma(0,1,5,8,9)$
4. $F(A,B,C,D) = \Sigma(1,4,5,6,12,14,15)$

Solution

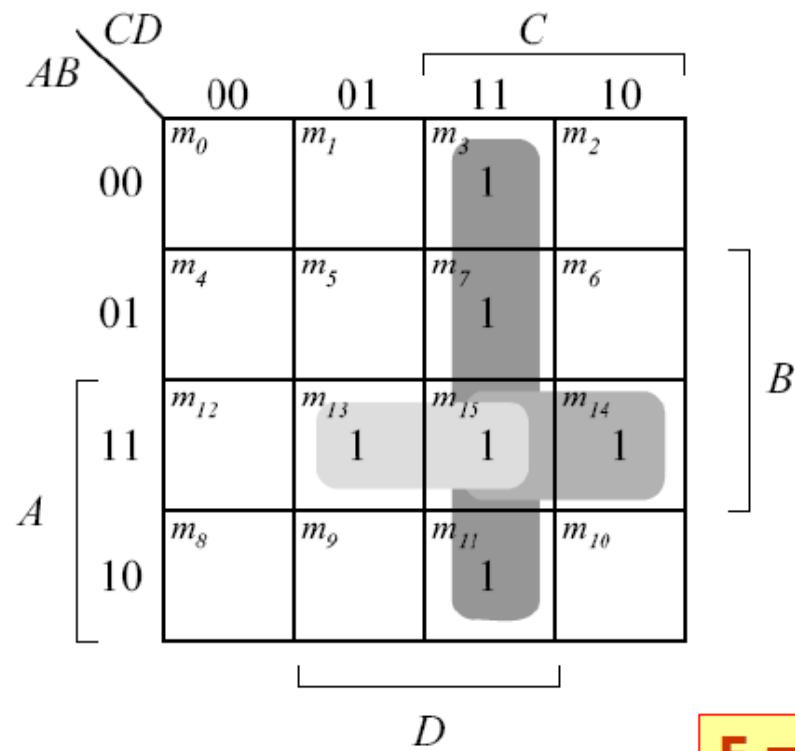
$$1. F(A,B,C,D) = \Sigma(4,6,7,15)$$



$$F = BCD + A'BD'$$

Solution

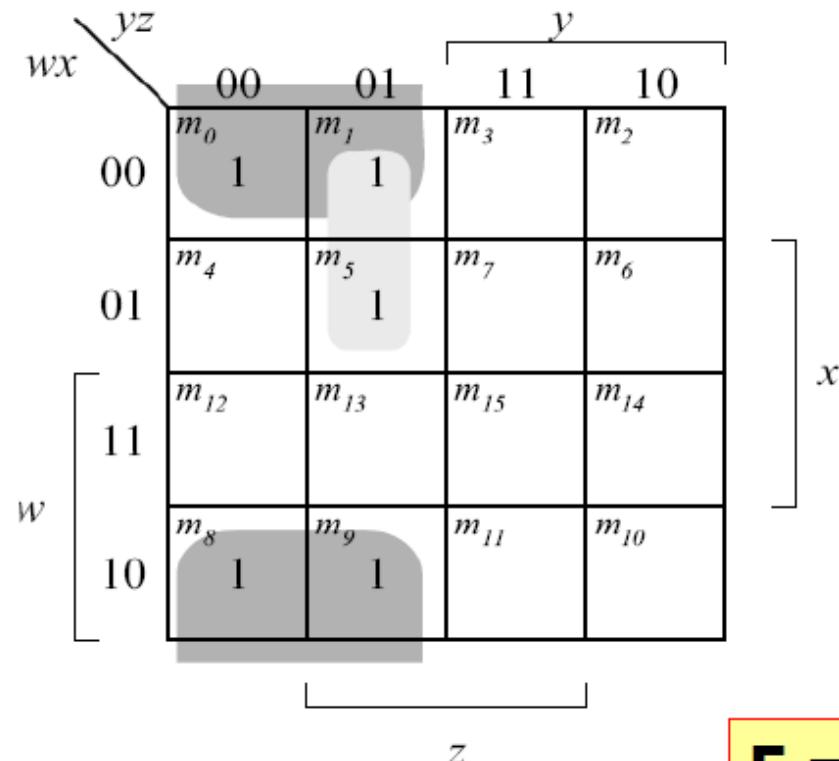
$$2. F(A,B,C,D) = \Sigma(3,7,11,13,14,15)$$



$$F = CD + ABD + ABC$$

Solution

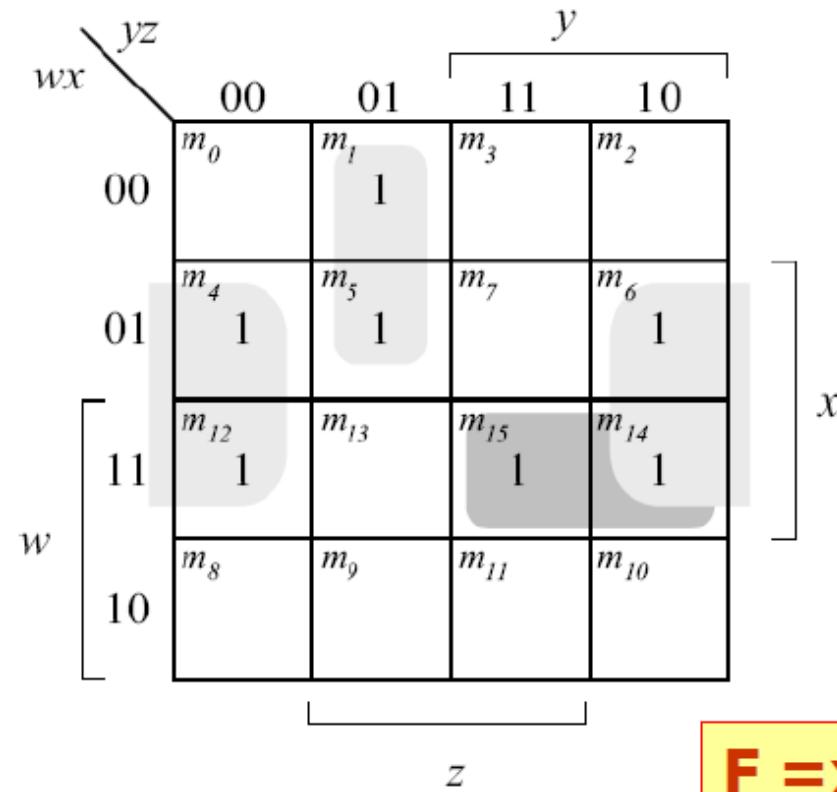
$$3. F(A,B,C,D) = \Sigma(0,1,5,8,9)$$



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

Solution

$$4. F(A,B,C,D) = \Sigma(1,4,5,6,12,14,15)$$



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

Don't care conditions

Sometimes a situation arises in which some input variable combinations are not allowed. For example, recall that in the BCD code.

- There are six invalid combinations: 1010, 1011, 1100, 1101, 1110, and 1111.
- Since these unallowed states will never occur in an application involving the BCD code, they can be treated as "don't care" terms with respect to their effect on the output.
- The "don't care" terms can be used to advantage on the Karnaugh map.

Cont...

For each "don't care" term, an X is placed in the cell.

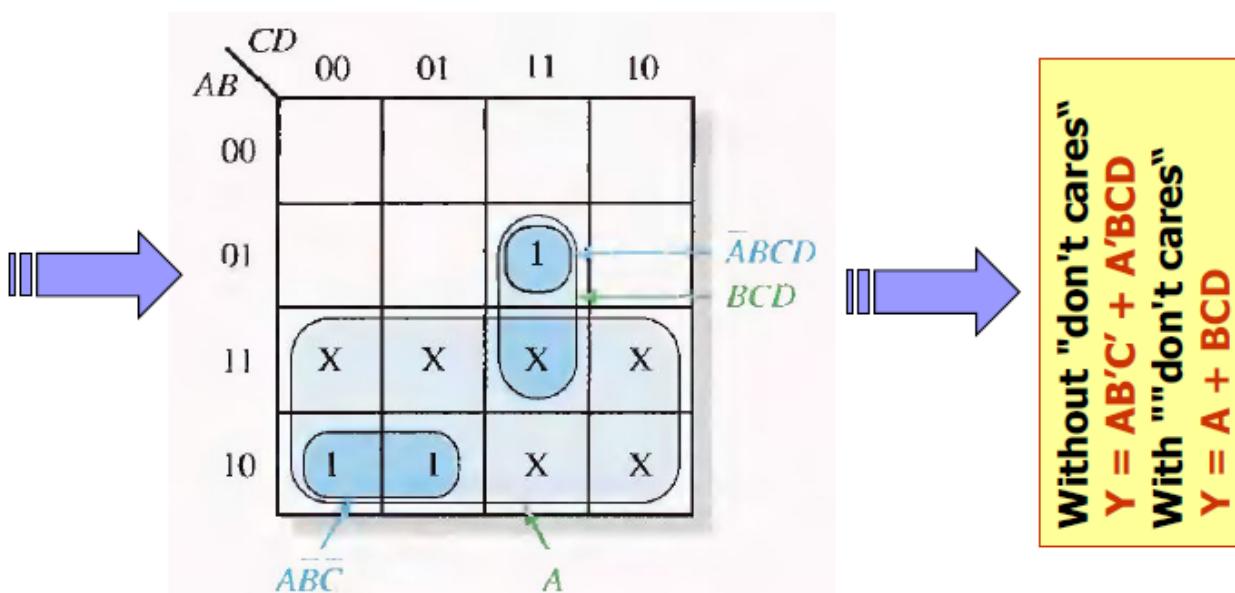
- When grouping the 1's, the X's can be treated as 1's to make a larger grouping or as 0's if they cannot be used to advantage.
- The larger a group, the simpler the resulting term will be.

Inputs <i>A B C D</i>	Output <i>Y</i>
0 0 0 0	0
0 0 0 1	0
0 0 1 0	0
0 0 1 1	0
0 1 0 0	0
0 1 0 1	0
0 1 1 0	0
0 1 1 1	1
1 0 0 0	1
1 0 0 1	1
1 0 1 0	X
1 0 1 1	X
1 1 0 0	X
1 1 0 1	X
1 1 1 0	X
1 1 1 1	X

DON'T care conditions

The truth table in Figure below describes a logic function that has a 1 output only when the BCD code for 7, 8, or 9 is present on the inputs. If the "don't cares" are used as 1s, the resulting expression for the function is $A + BCD$.

Inputs $ABCD$	Output Y
0 0 0 0	0
0 0 0 1	0
0 0 1 0	0
0 0 1 1	0
0 1 0 0	0
0 1 0 1	0
0 1 1 0	0
0 1 1 1	1
1 0 0 0	1
1 0 0 1	1
1 0 1 0	X
1 0 1 1	X
1 1 0 0	X
1 1 0 1	X
1 1 1 0	X
1 1 1 1	X



Without "don't cares"
 $Y = AB'C' + A'BCD$
With ""don't cares"
 $Y = A + BCD$

Related Problem

Simplify the Boolean functions

$$F(w,x,y,z) = \Sigma(1,3,7,11,15)$$

Don't care conditions are,

$$d(w,x,y,z) = \Sigma(0,2,5)$$

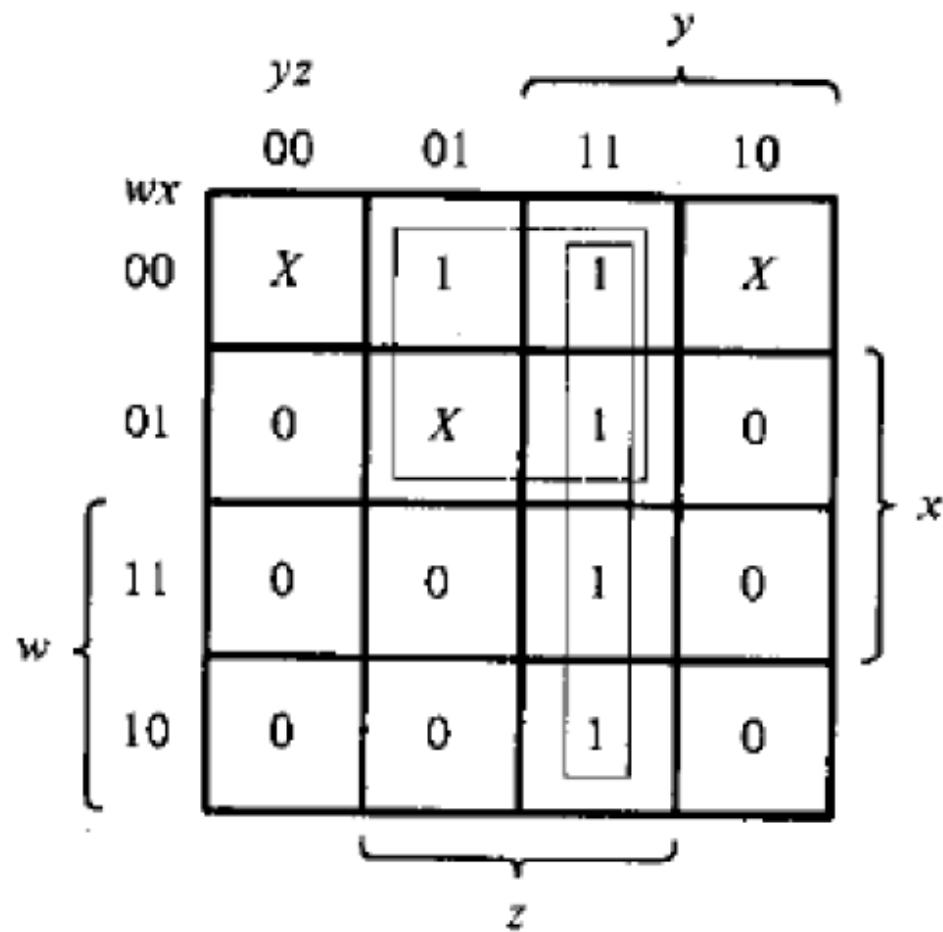
Solution

$$F = yz + w'x'$$

		yz		y	
		00	01	11	10
wx		00	1	1	x
w	00	X	1	1	X
	01	0	X	1	0
	11	0	0	1	0
	10	0	0	1	0

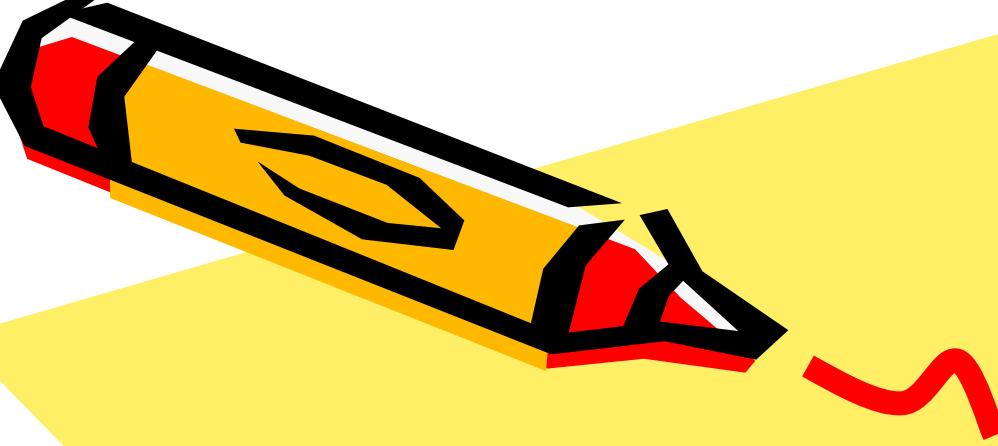
y
 x
 z

Example



$$(b) \quad F = yz + w'z$$

END OF LECTURE

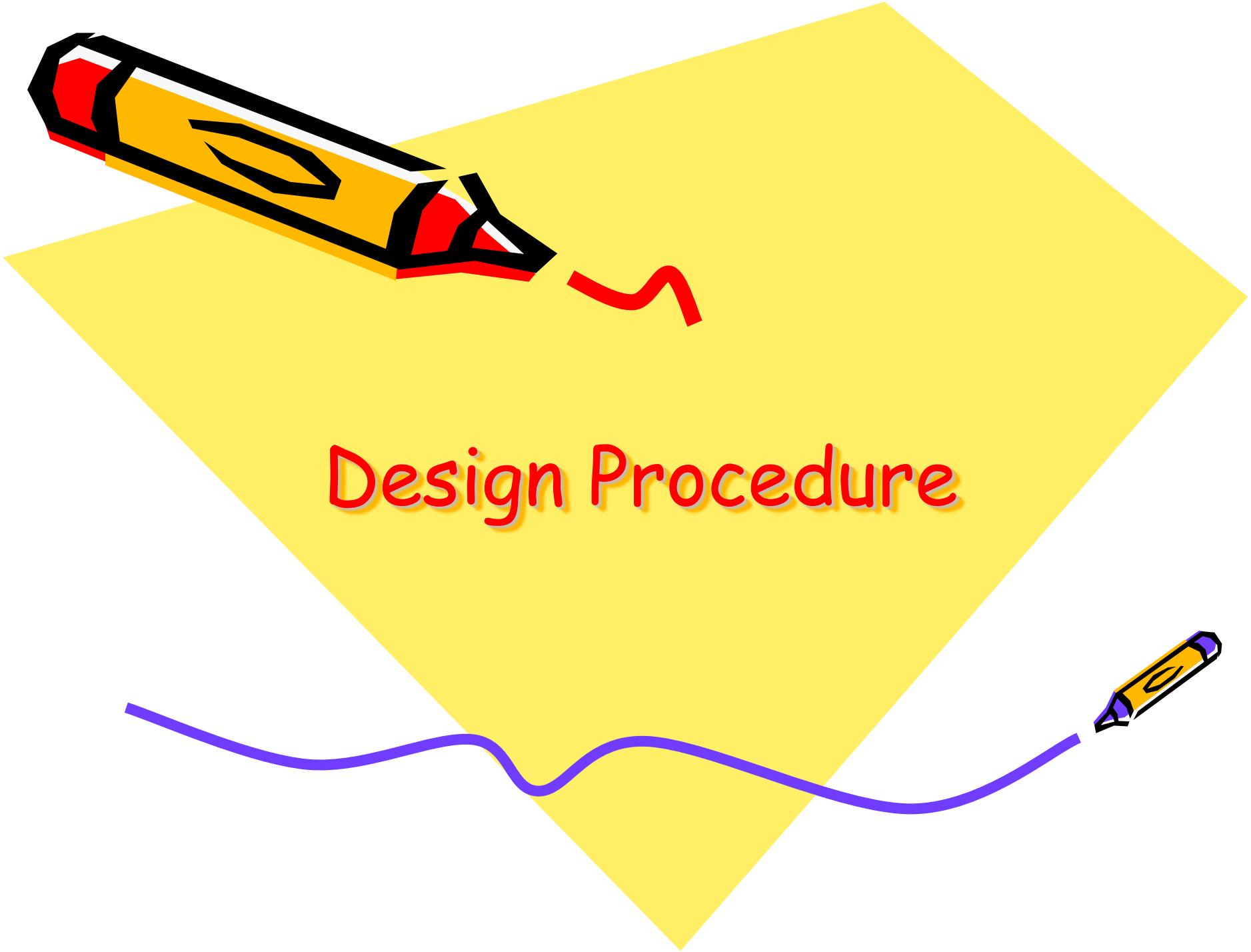


Logic Design

CSE 221

Dr. Mohamed Osama Khozium



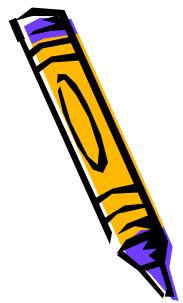


Design Procedure



Design Procedure

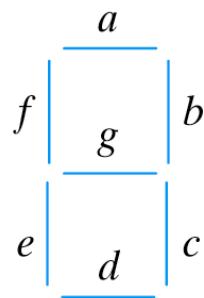
1. From the specifications of the circuit, determine the required number of inputs and outputs and assigned a letter symbol to each
2. Derive the truth table that defined the relationship between inputs and outputs
3. Obtain the simplified Boolean functions for each output as a function of the input variables
4. Draw the logic diagram
5. Verify the correctness of the design





Some Design Examples

- BCD to Excess-3 code converter
- BCD to Seven Segment Decoder



(a) Segment designation



(b) Numerical designation for display

Fig. P4-9





BCD to Excess-3 code converter

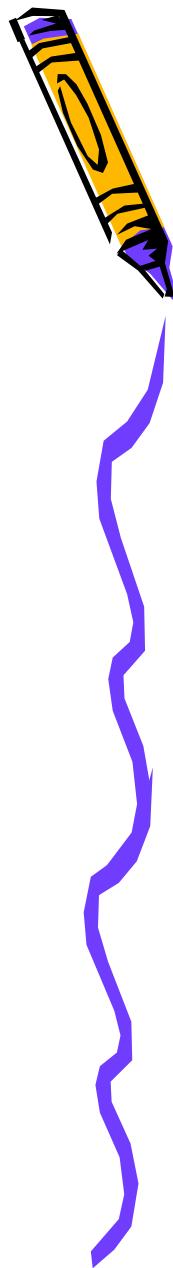
□ TABLE 3-2
Truth Table for Code Converter Example

Decimal Digit	Input BCD				Output Excess-3			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

› 9 (in) → X (don't care) in output till 15



InPuts				OutPuts			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x





BCD to Excess-3 code converter K-Maps

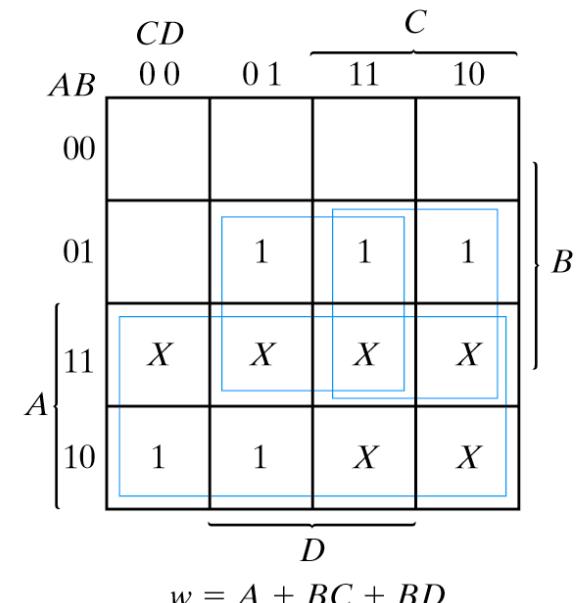
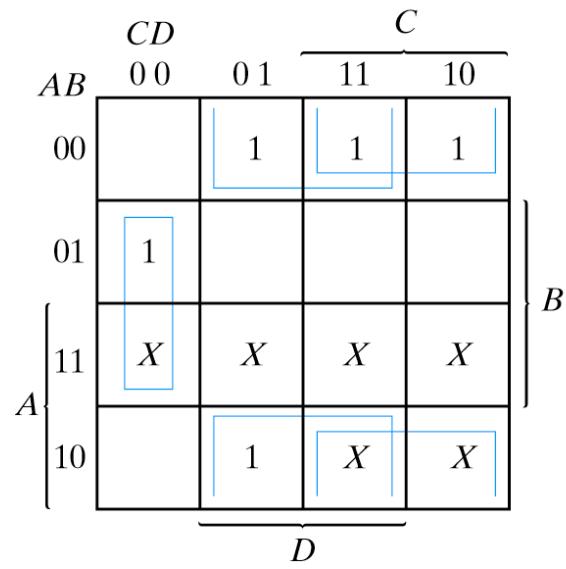
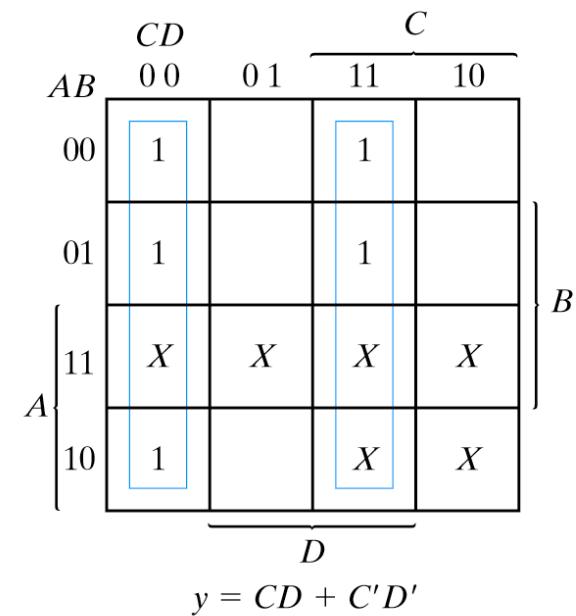
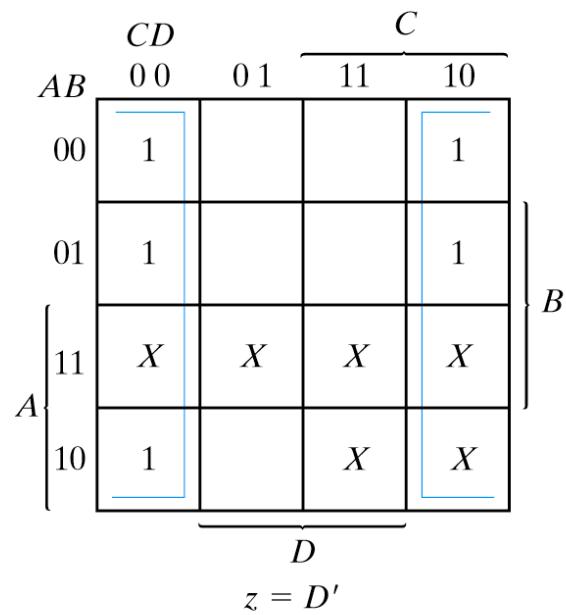
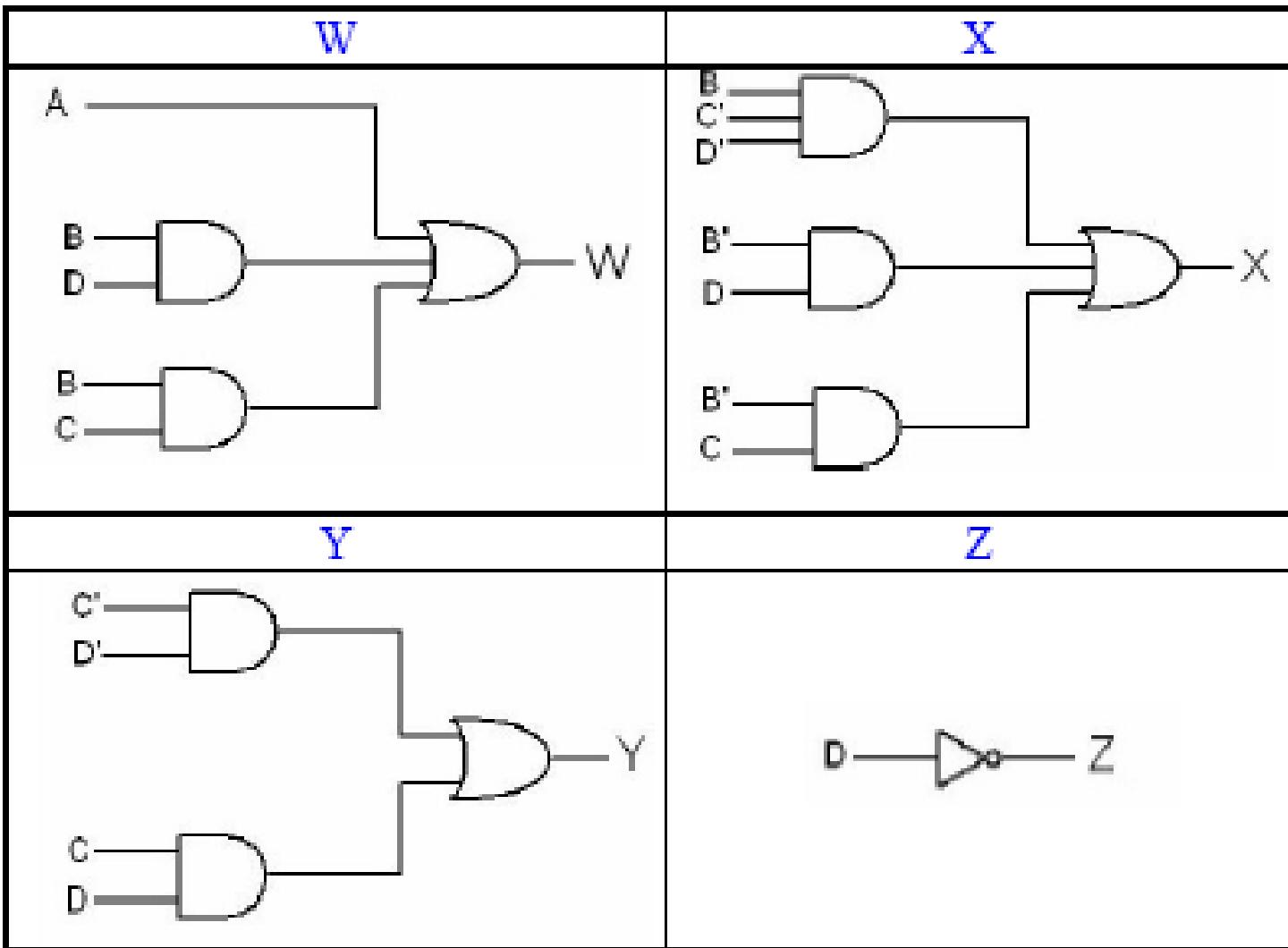
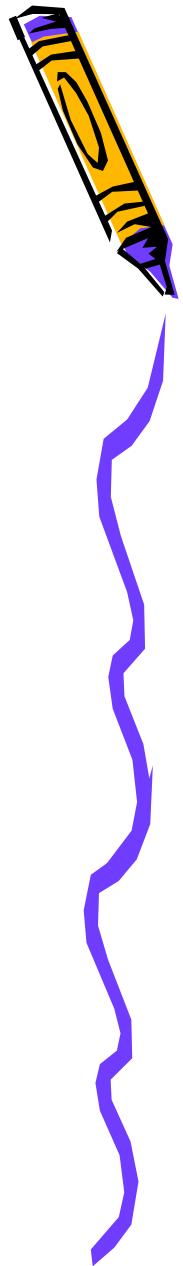


Fig. 4-3 Maps for BCD to Excess-3 Code Converter







BCD to Excess-3 code converter

Logic Diagram

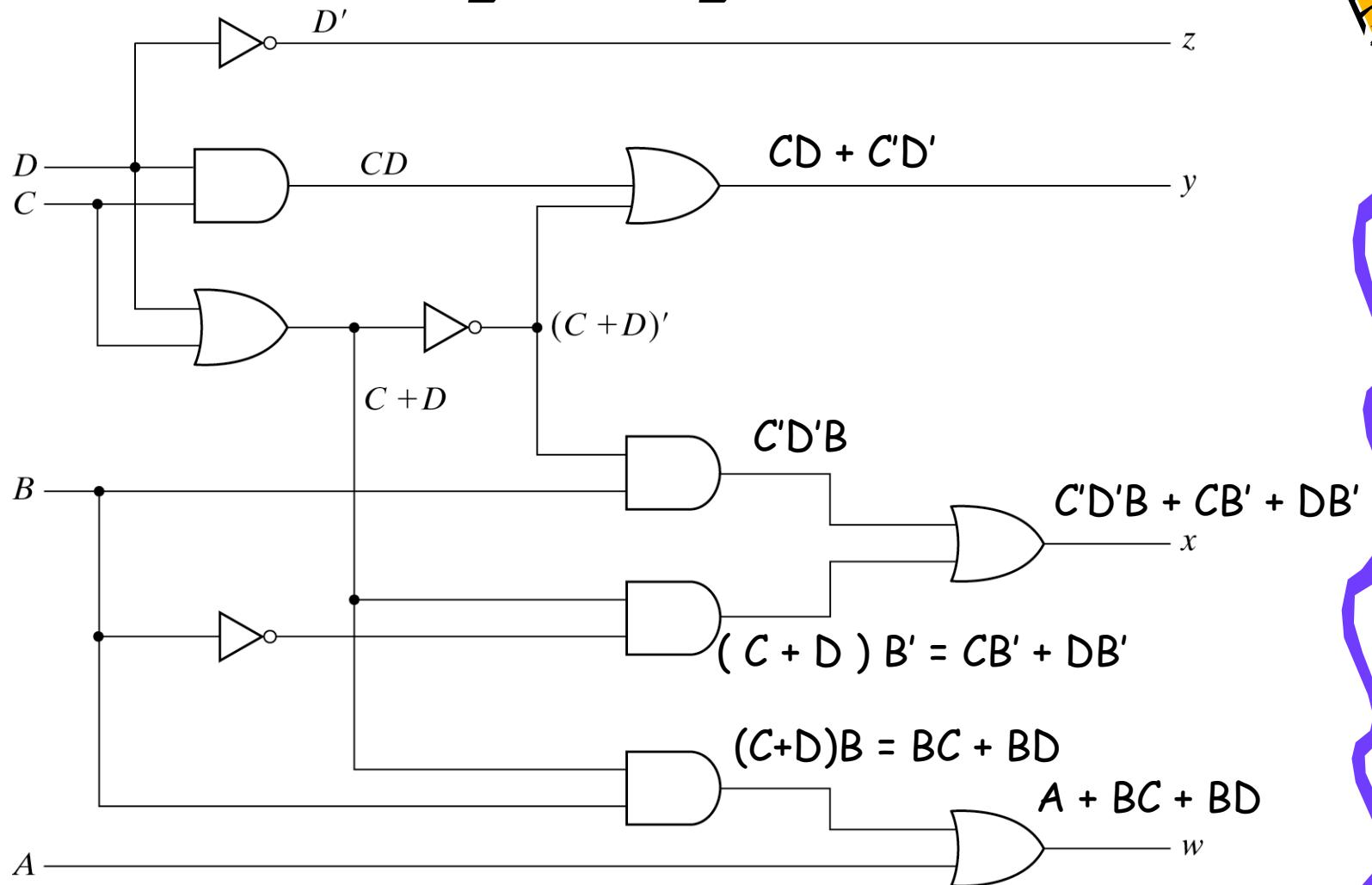
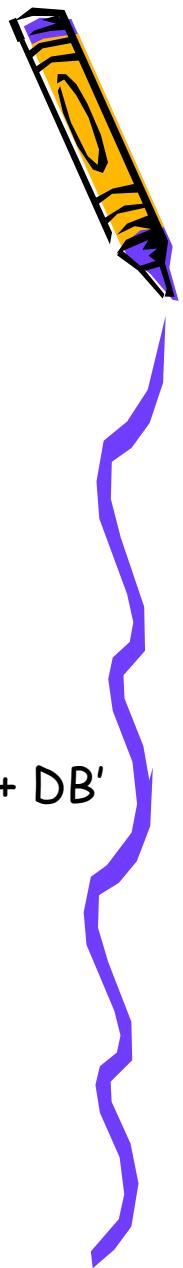
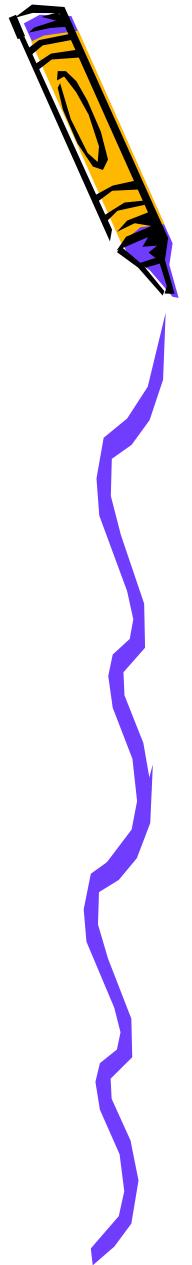


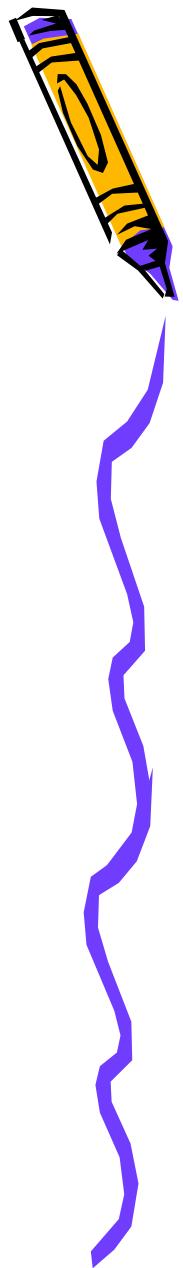
Fig. 4-4 Logic Diagram for BCD to Excess-3 Code Converter

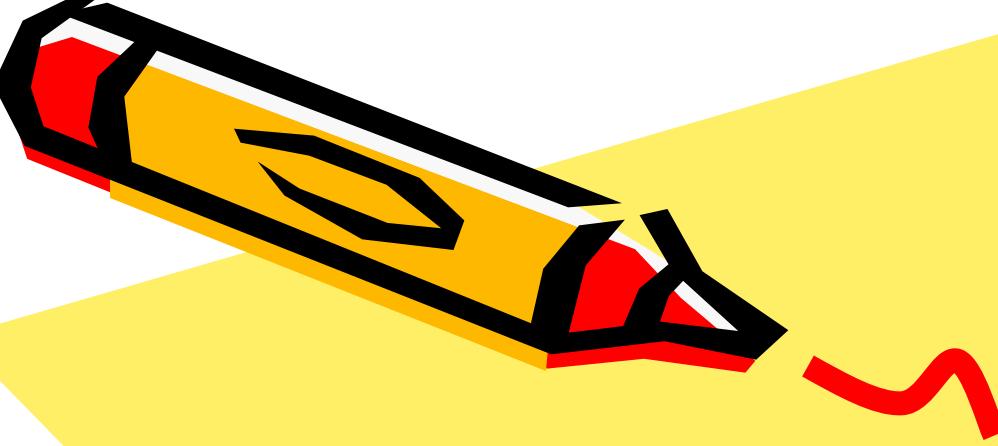




Thank you





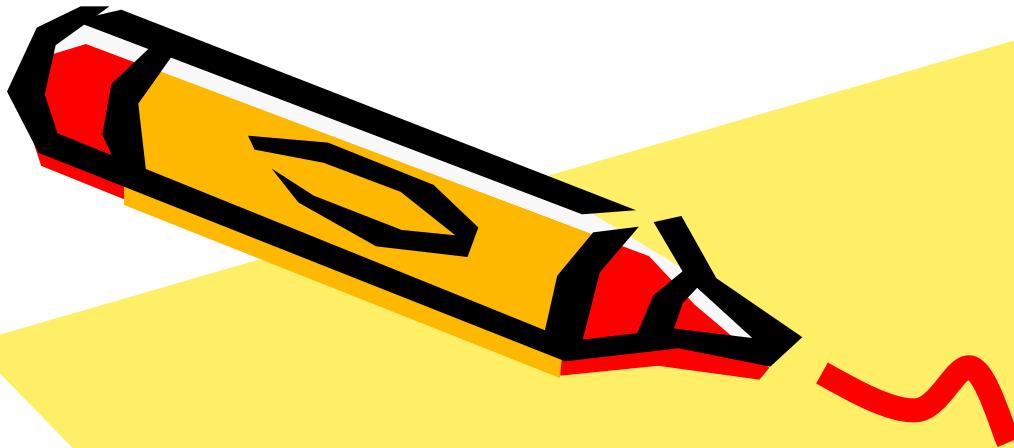


Logical Design

CS 221

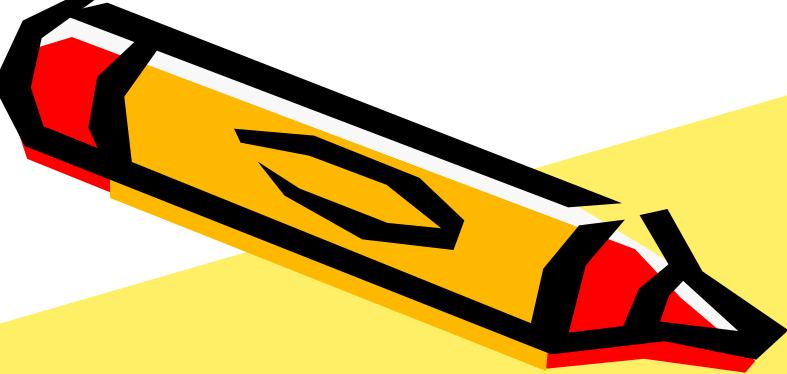
Prof.Dr. Mohamed Osama Khozium





Combinational logic circuits

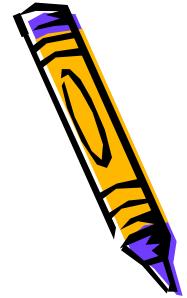




NOR & NAND
XOR & XNOR
Half & Full Adder
Decoder
Multiplexer



AND = AB ,.... NAND = \overline{AB} ,.... Negative AND = $\overline{A} \overline{B}$



A	B	A'	B'	AB	\overline{AB}	$\overline{A} \overline{B}$
0	0	1	1	0	1	1
0	1	1	0	0	1	0
1	0	0	1	0	1	0
1	1	0	0	1	0	0





$OR = A+B$, $NOR = \overline{A+B}$, Negative OR = $\overline{A} + \overline{B}$, $XOR = A \oplus B$, .. $XNOR = A \otimes B$

A	B	A'	B'	$A+B$	$\overline{A+B}$	$\overline{A} + \overline{B}$	$A \oplus B$	$A \otimes B$
0	0	1	1	0	1	1	0	1
0	1	1	0	1	0	1	1	0
1	0	0	1	1	0	1	1	0
1	1	0	0	1	0	0	0	1

$$XOR = A \oplus B = A'B + AB'$$

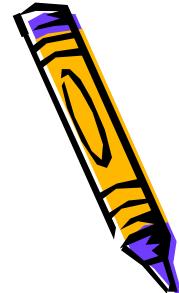
$$XNOR = A \otimes B = AB + \overline{A}\overline{B}$$





AND = AB ,.... NAND = \overline{AB} ,..... Negative AND = $\overline{A} \overline{B}$

A	B	A'	B'	AB	\overline{AB}	$\overline{A} \overline{B}$
0	0	1	1	0	1	1
0	1	1	0	0	1	0
1	0	0	1	0	1	0
1	1	0	0	1	0	0



OR = $A+B$, NOR = $\overline{A + B}$, Negative OR = $\overline{A} + \overline{B}$, XOR = $A \oplus B$, .. XNOR = $A \otimes B$

A	B	A'	B'	$A+B$	$\overline{A + B}$	$\overline{A} + \overline{B}$	$A \oplus B$	$A \otimes B$
0	0	1	1	0	1	1	0	1
0	1	1	0	1	0	1	1	0
1	0	0	1	1	0	1	1	0
1	1	0	0	1	0	0	0	1

$$\text{XOR} = A \oplus B = A'B + AB'$$

$$\text{XNOR} = A \otimes B = AB + \overline{A} \overline{B}$$



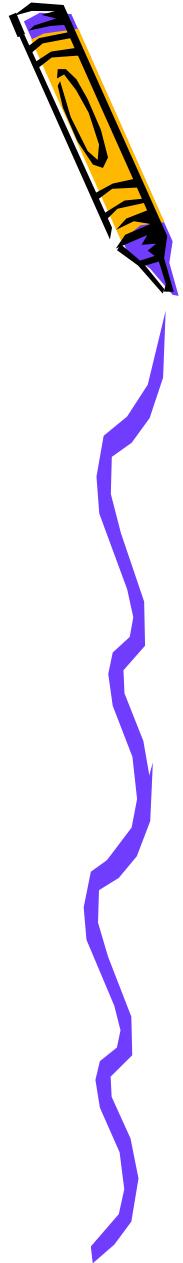


Prove $A \otimes B = \overline{A \oplus B}$

$$A \otimes B = AB + A'B' = [(AB)'(A'B')']' =$$

$$[(A'+B')(A+B)]' = [A'A + A'B + AB' + BB']'$$

$$= [A'B + AB']' = \overline{A \oplus B}$$

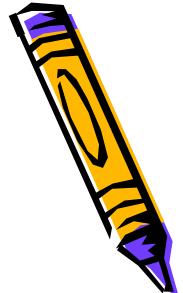




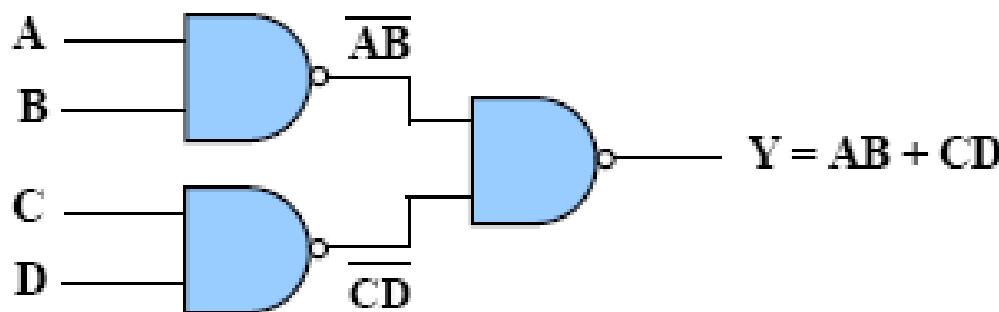
$$\overline{A \bullet B} = \overline{A} + \overline{B}$$

NAND

Negative-OR



Example



$$Y = (\overline{AB})(\overline{CD})$$

From Demorgan Theorem

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

Then $Y = AB + CD$

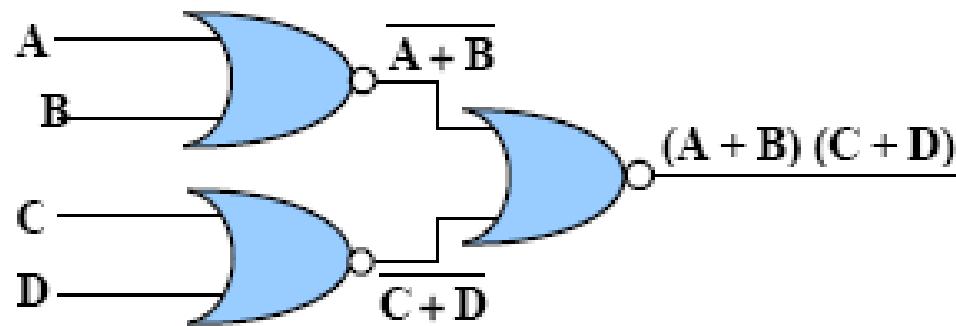




$$\overline{A + B} = \overline{A} \bullet \overline{B}$$

NOR

Negative-AND

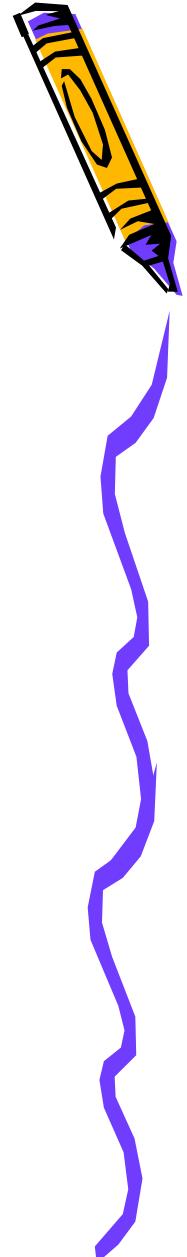


$$Y = \overline{\overline{(A + B)} + \overline{(C + D)}}$$

From Demorgan Theorem

$$Y = (\overline{A + B}) \bullet (\overline{C + D})$$

$$Y = (A + B) \bullet (C + D)$$



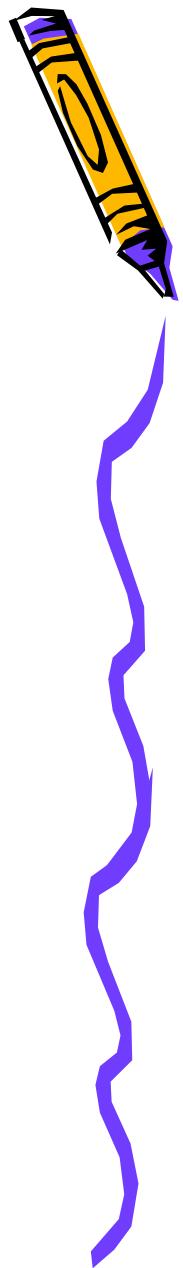


Exclusive-OR Gate XOR-gate

IN		OUT	
A	B	Y	
0	0	0	$A'B'$
0	1	1	$A'B$
1	0	1	AB'
1	1	0	AB

$$Y = \overline{A}B + A\overline{B}$$

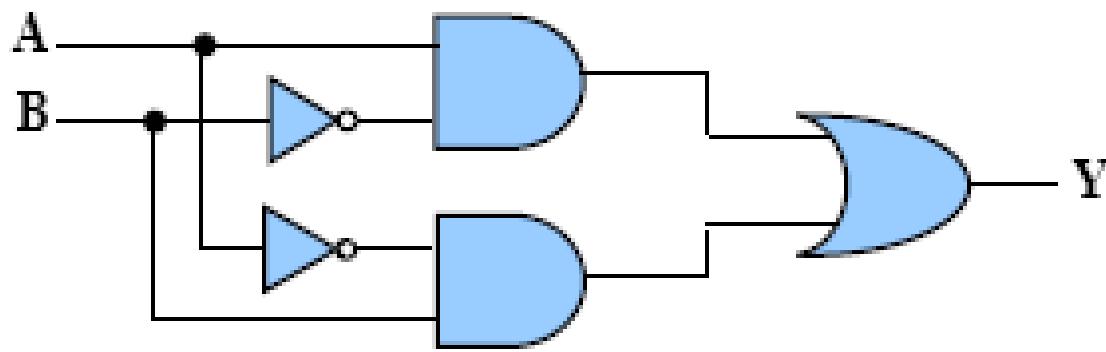
$$Y = A \oplus B$$



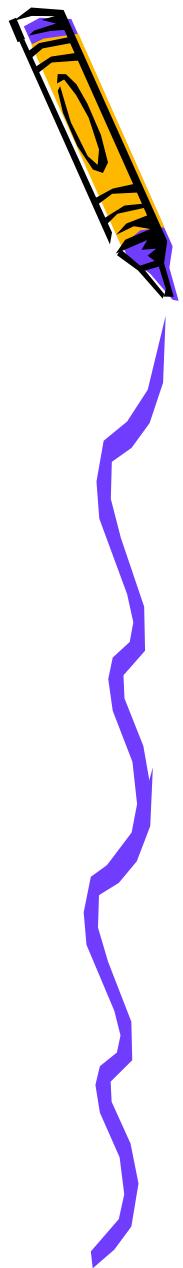


$$Y = \overline{A}B + A\overline{B}$$

$$Y = A \oplus B$$

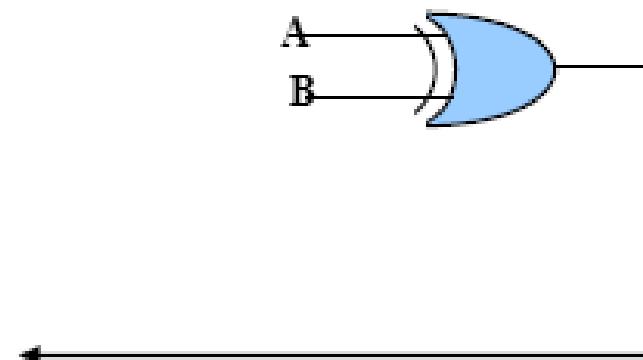
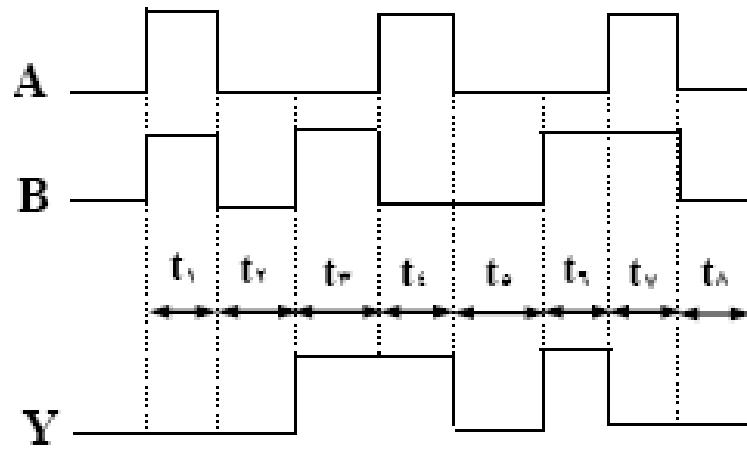


XOR represented by AND & OR & NOT



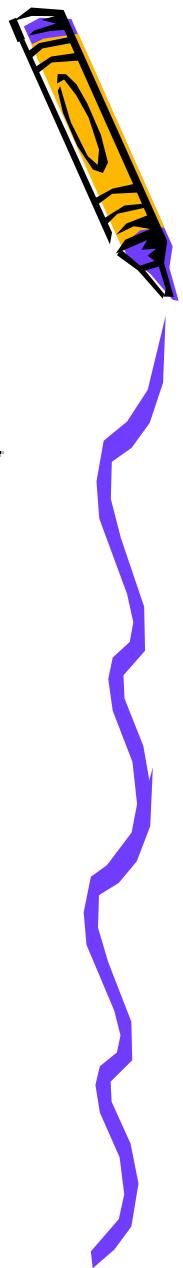


Timing Diagram



$$Y = A \oplus B$$

Y



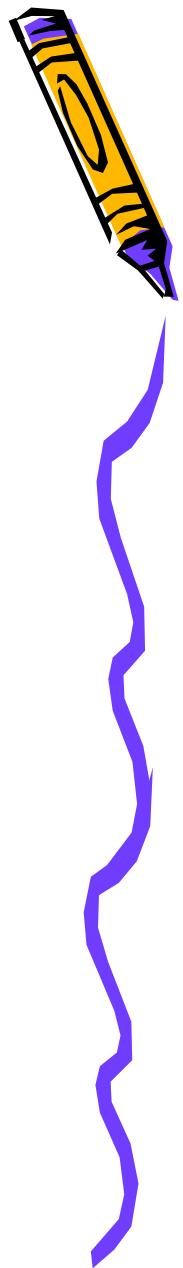


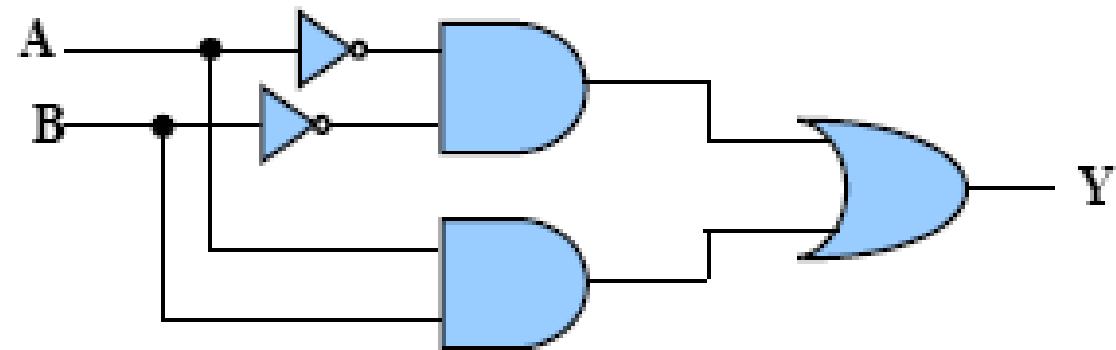
Exclusive-NOR Gate XNOR-gate

IN		OUT	
A	B	Y	
0	0	1	$A'B'$
0	1	0	$A'B$
1	0	0	AB'
1	1	1	AB

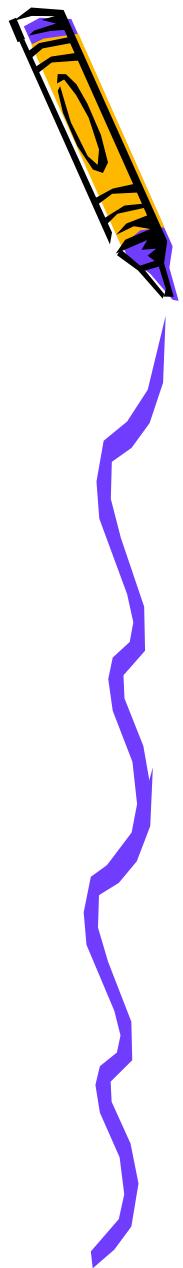
$$Y = AB + \overline{AB}$$

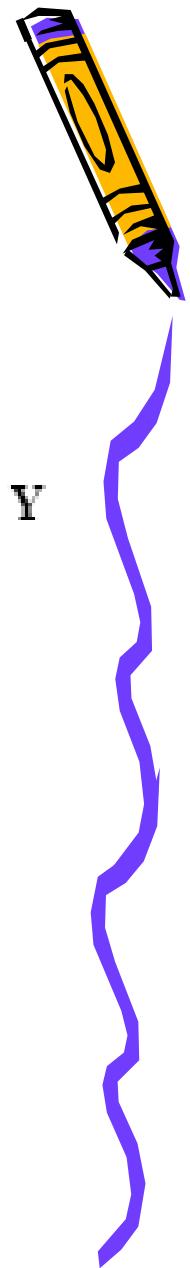
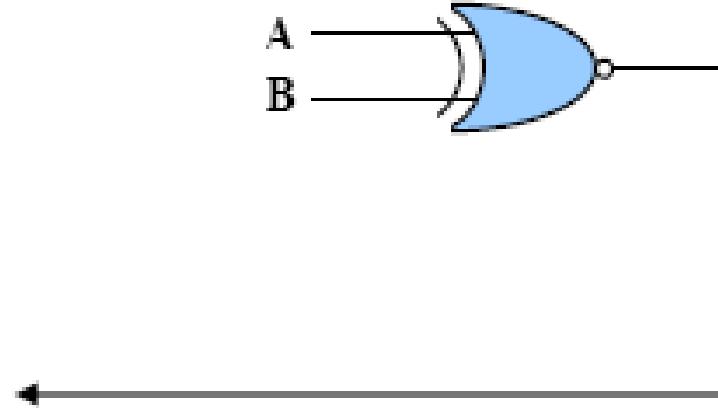
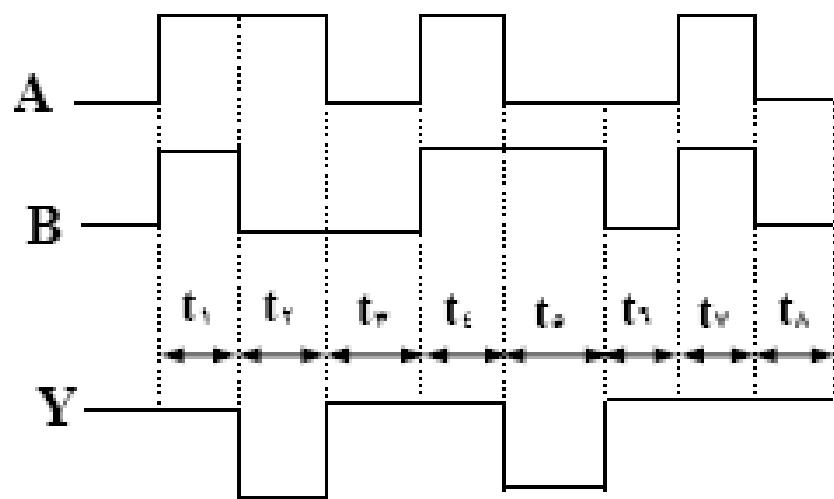
$$Y = A \odot B$$

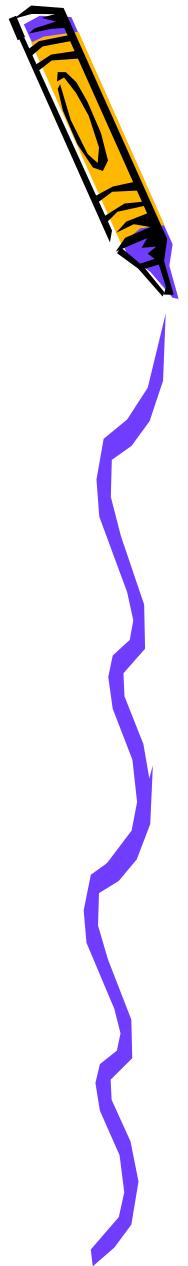




XNOR represented by AND & OR & NOT







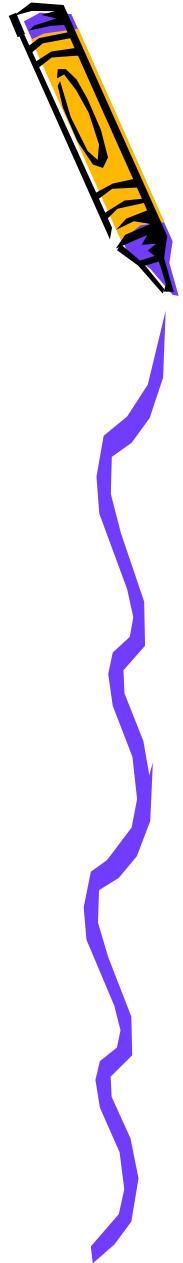
Half Adder





The Design Procedures

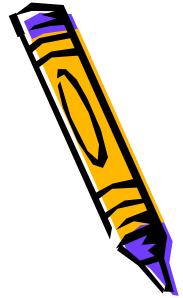
- (1) Determine inputs
 Determine outputs
- (2) Derive the truth table
- (3) Obtain the simplified Boolean functions
- (4) Draw the logic diagram
- (5) Verify the correctness of the design





Half Adder

Follow The Design Procedures



(1) Determine inputs : A & B
Determine outputs : S & C

(2) Derive the truth table :

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(3) Obtain the simplified Boolean functions :

$$S = A' B + A B' = A \oplus B$$

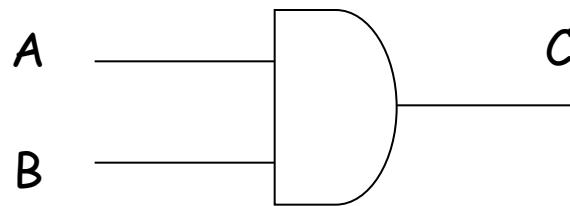
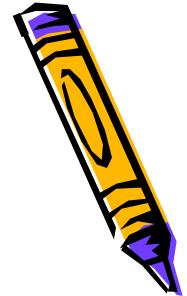
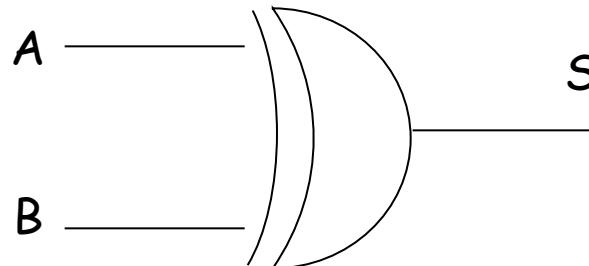
$$C = A B$$





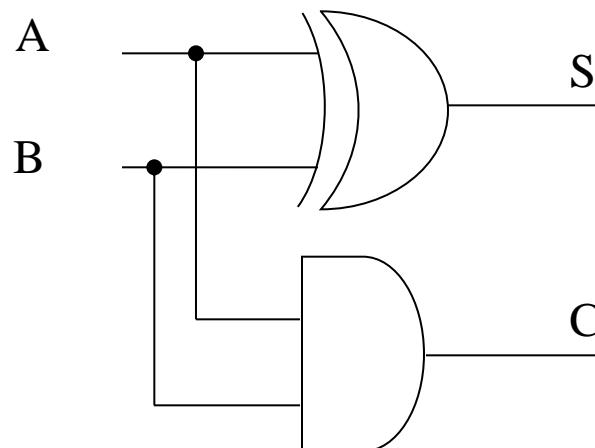
(4)

Draw the logic diagram :



(5)

Verify the correctness of the design :





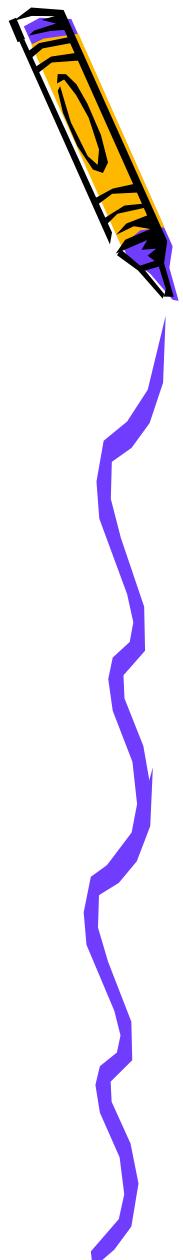
Full Adder

(1) Inputs : A, B, Cin
 Output : S, C

(2)

A	B	Cin	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(3) $S = A' B' Cin + A' B Cin' + A B' Cin' + A B Cin$
 $C = A' B Cin + A B' Cin + A B Cin' + A B Cin$



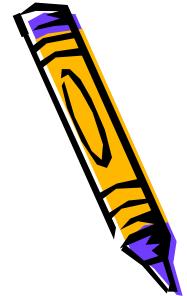
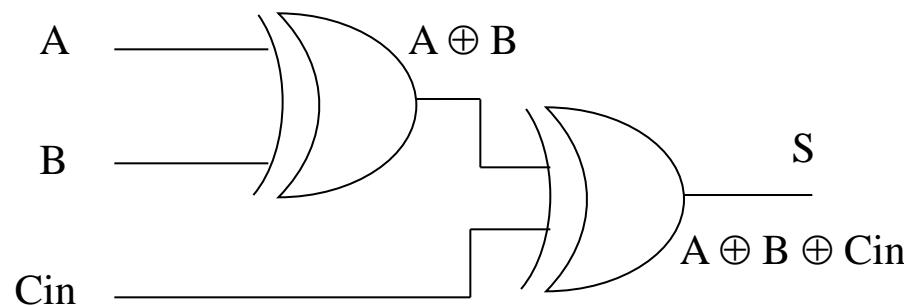


(3)

$$S = A' B' Cin + A' B Cin' + A B' Cin' + A B Cin$$
$$C = A' B Cin + A B' Cin + A B Cin' + A B Cin$$
$$S = (A' B + AB') Cin' + (A' B' + A B) Cin$$
$$= (A \oplus B) Cin' + (A \ominus B) Cin$$
$$= (A \oplus B) Cin' + (\overline{A \oplus B}) Cin$$
$$= A \oplus B \oplus Cin$$

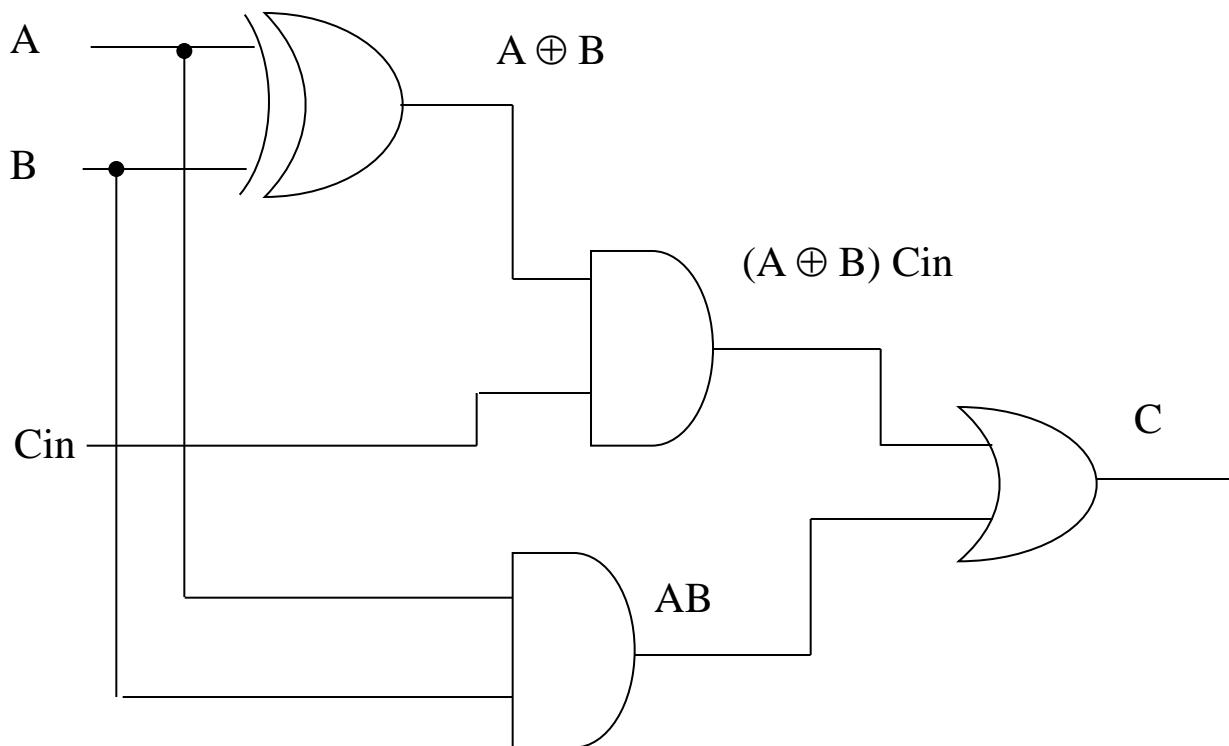
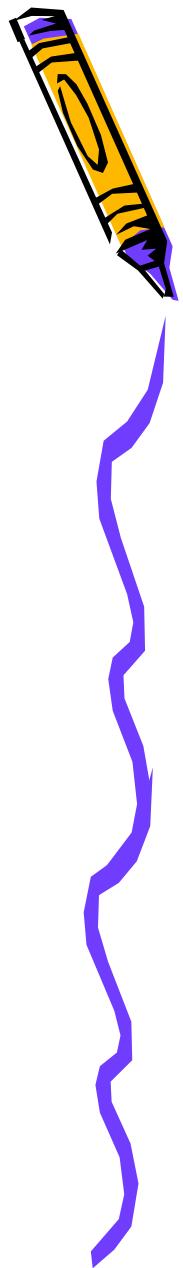
$$Cin = Y ; A \oplus B = X$$

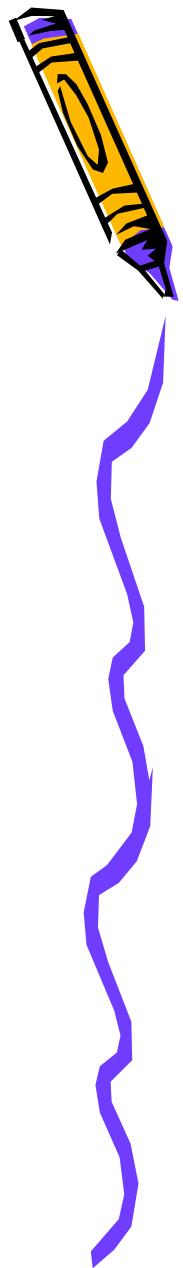
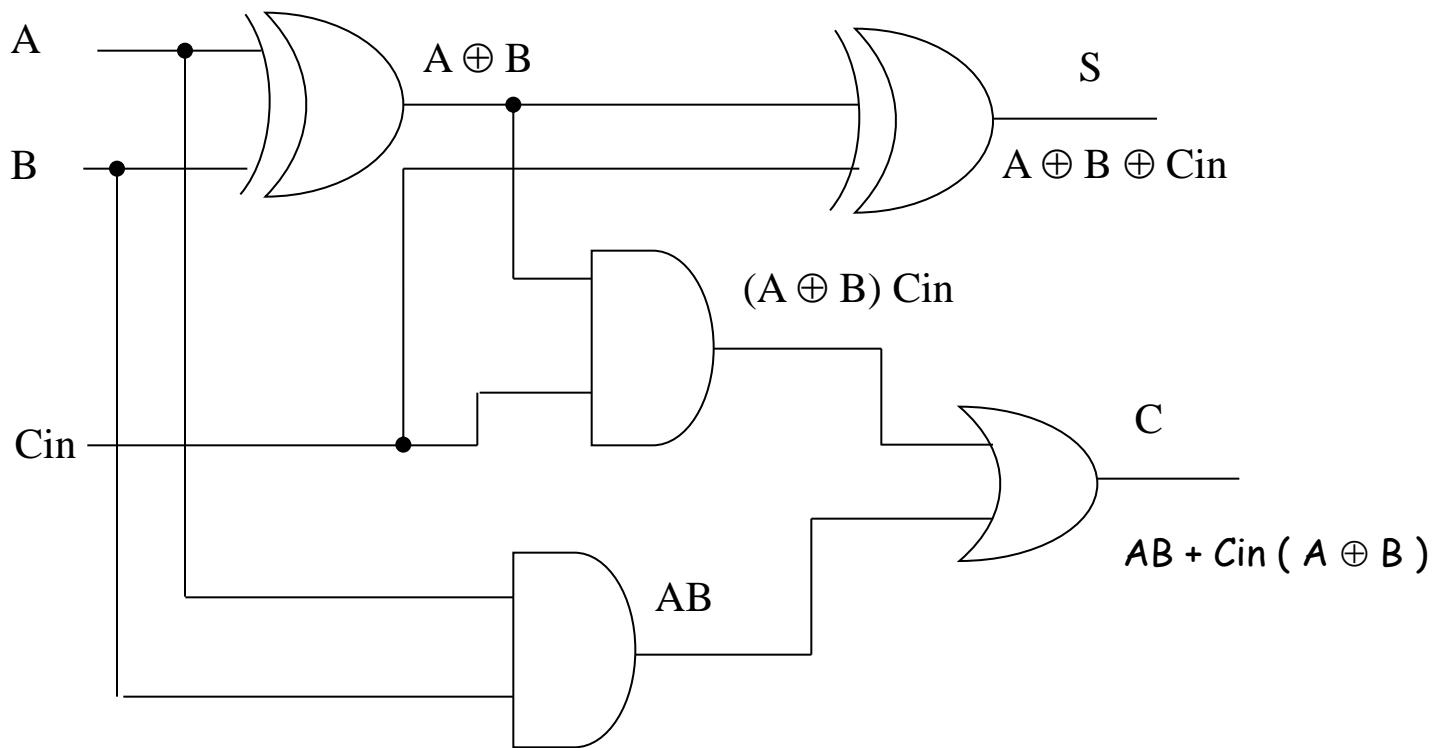
$$\therefore S = XY' + X'Y \Rightarrow S = X \oplus Y$$





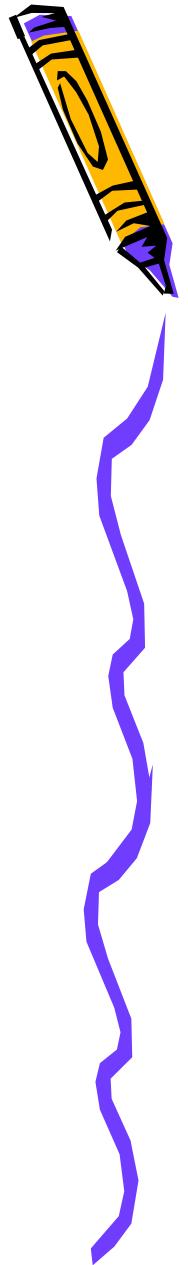
$$\begin{aligned}C &= AB (Cin + Cin') + Cin (A'B + AB') \\&= AB + Cin (A \oplus B)\end{aligned}$$

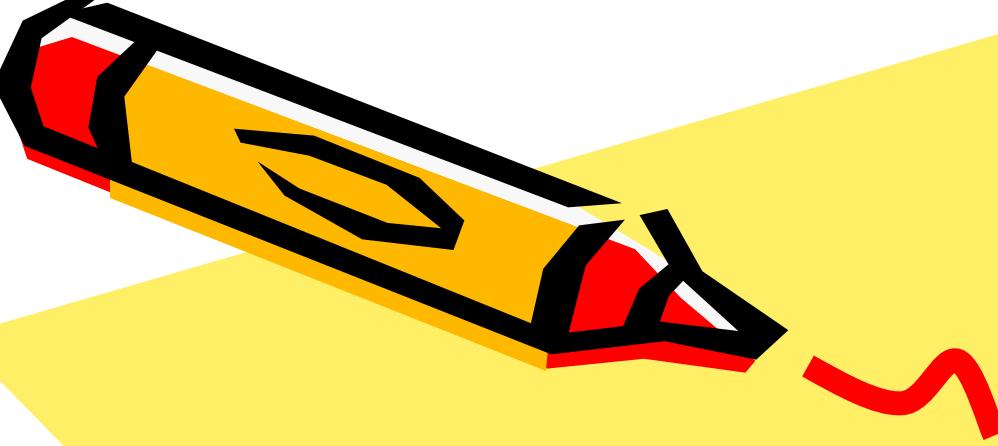






Thank you



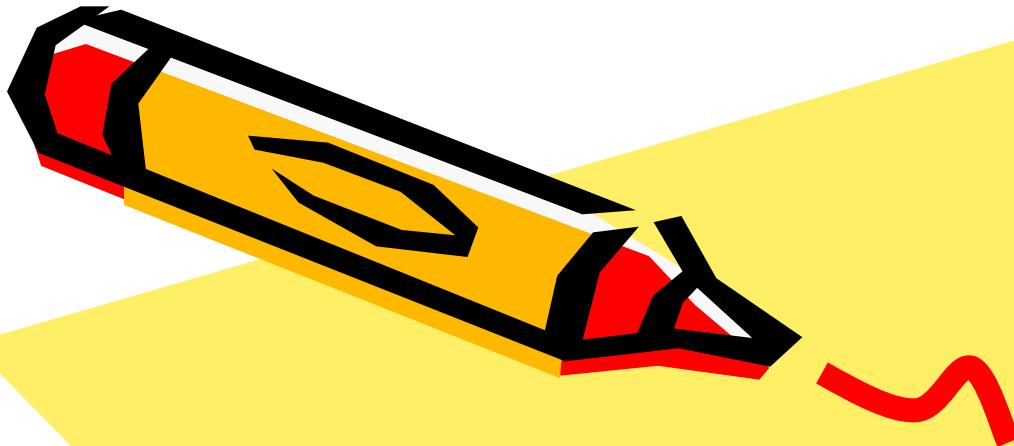


Logical Design

CS 221

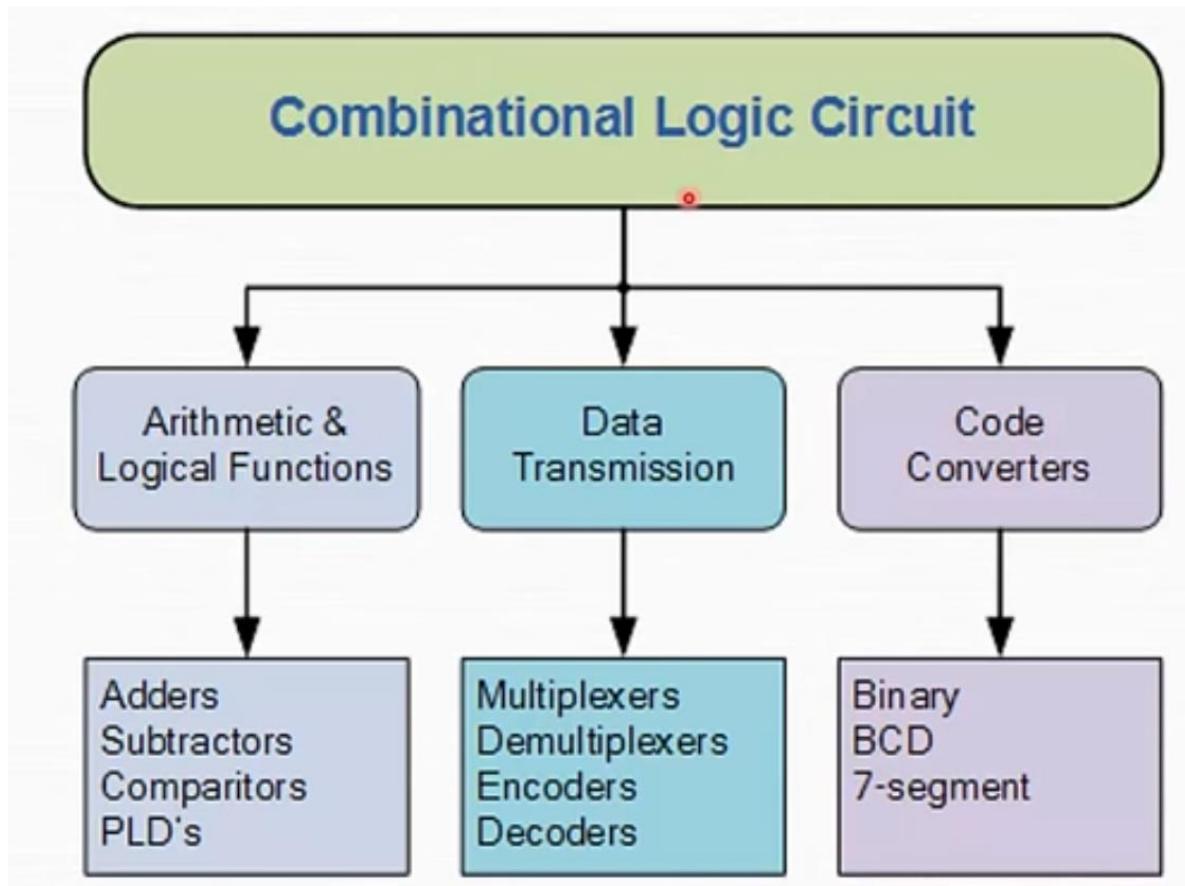
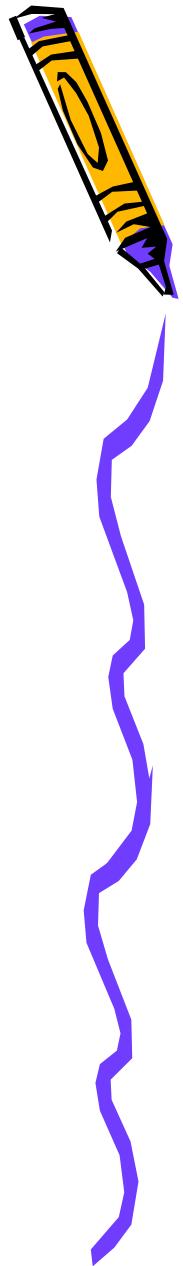
Prof.Dr. Mohamed Osama Khozium





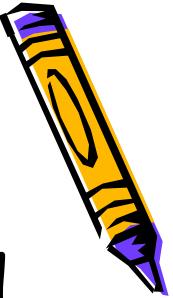
Combinational logic circuits







Encoders and Decoders

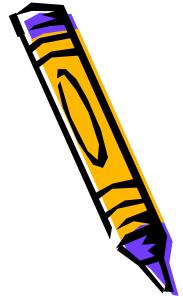


Binary code of N digits can be used to store 2^N distinct elements of coded information. This is what encoders and decoders are used for. **Encoders** convert 2^N lines of input into a code of N bits and **Decoders** decode the N bits into 2^N lines.





Encoders

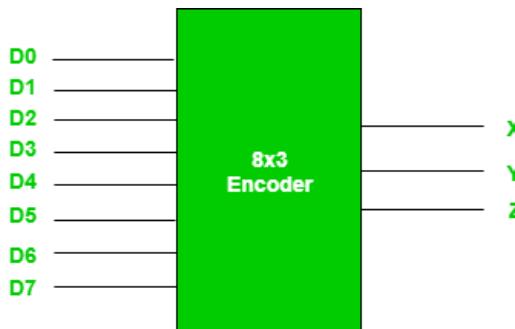


An encoder is a combinational circuit that converts binary information in the form of a 2^N input lines into N output lines, which represent N bit code for the input. For simple encoders, it is assumed that only one input line is active at a time.





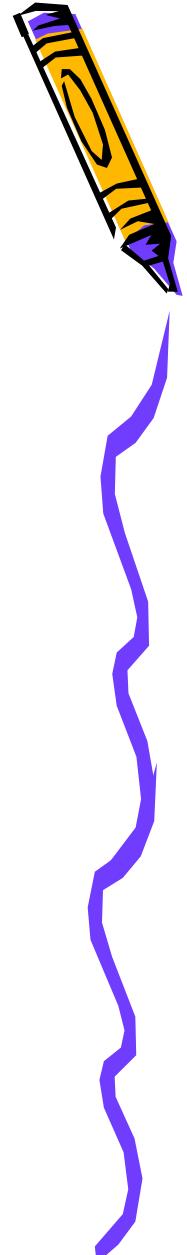
As an example, let's consider **Octal to Binary** encoder. As shown in the following figure, an octal-to-binary encoder takes 8 input lines and generates 3 output lines.



Block diagram for an encoder 8x3

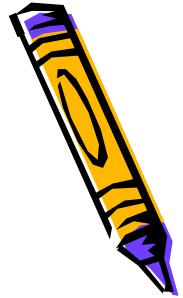
Truth Table

D7	D6	D5	D4	D3	D2	D1	D0	X	Y	Z
0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1





As seen from the truth table, the output is 000 when D0 is active; 001 when D1 is active; 010 when D2 is active and so on.



Implementation -

From the truth table, the output line Z is active when the input octal digit is 1, 3, 5 or 7. Similarly, Y is 1 when input octal digit is 2, 3, 6 or 7 and X is 1 for input octal digits 4, 5, 6 or 7. Hence, the Boolean functions would be:

$$X = D_4 + D_5 + D_6 + D_7$$

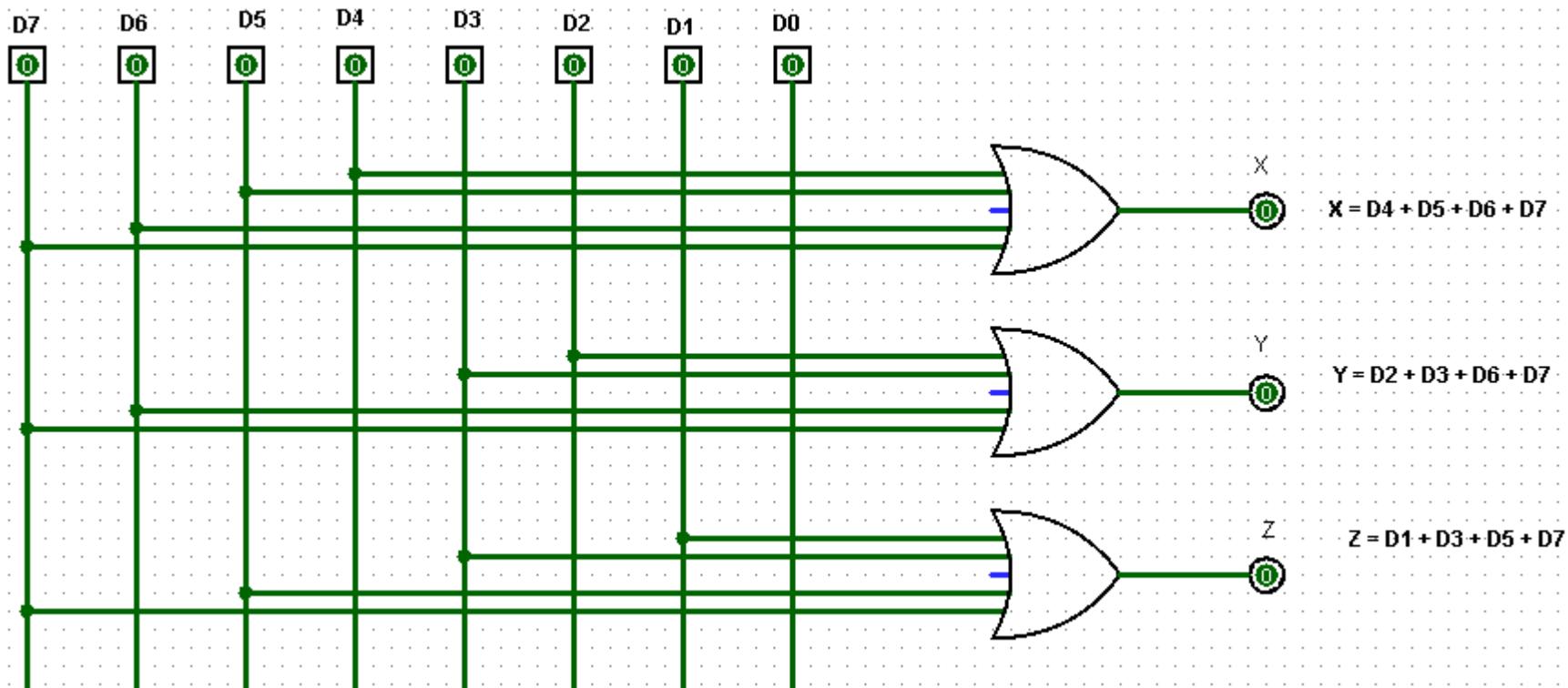
$$Y = D_2 + D_3 + D_6 + D_7$$

$$Z = D_1 + D_3 + D_5 + D_7$$

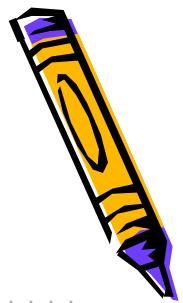


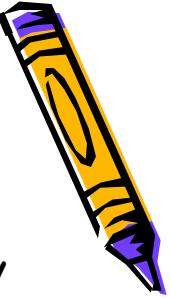


Hence, the encoder can be realized with OR gates as follows:



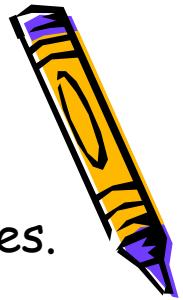
Logic circuit for 8 X 3 Encoder





One limitation of this encoder is that only one input can be active at any given time. If more than one inputs are active, then the output is undefined. For example, if D6 and D3 are both active, then, our output would be 111 which is the output for D7. To overcome this, we use Priority Encoders. Another ambiguity arises when all inputs are 0. In this case, encoder outputs 000 which actually is the output for D0 active. In order to avoid this, an extra bit can be added to the output, called the valid bit which is 0 when all inputs are 0 and 1 otherwise.





Priority Encoder -

A priority encoder is an encoder circuit in which inputs are given priorities.

Truth Table

D3	D2	D1	D0	X	Y	V
0	0	0	0	x	x	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1





Decoders

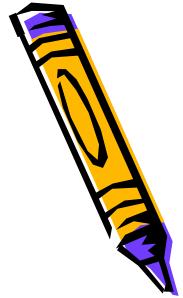
Decoders are important type of combinational circuit.

Address decoders with n inputs can select any of 2^n locations.

They are useful in selecting a memory location according a binary value place don the address lines of a memory bas.



Block diagram for a decoder



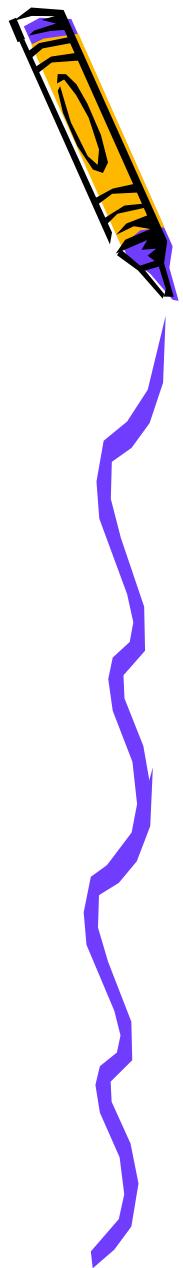


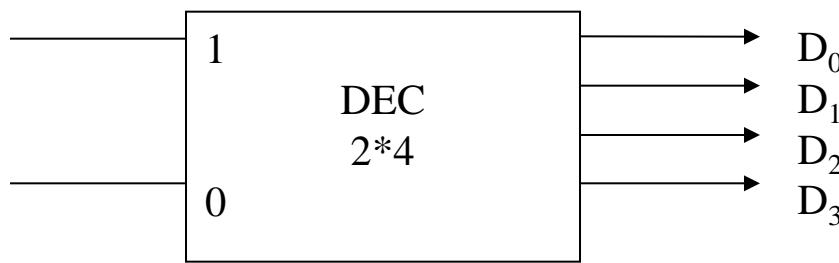
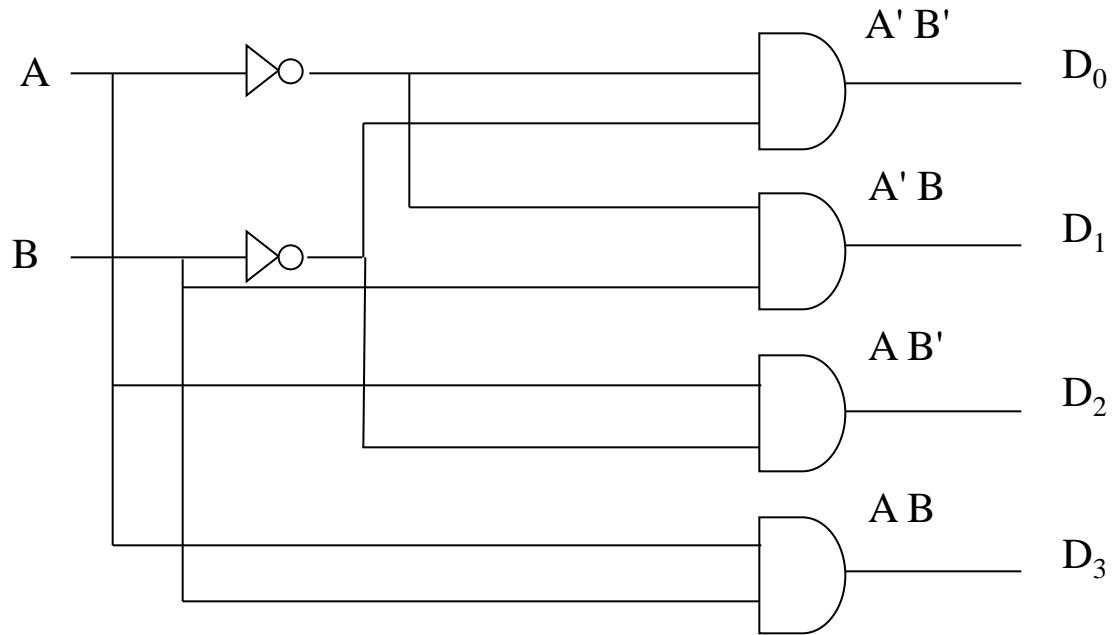
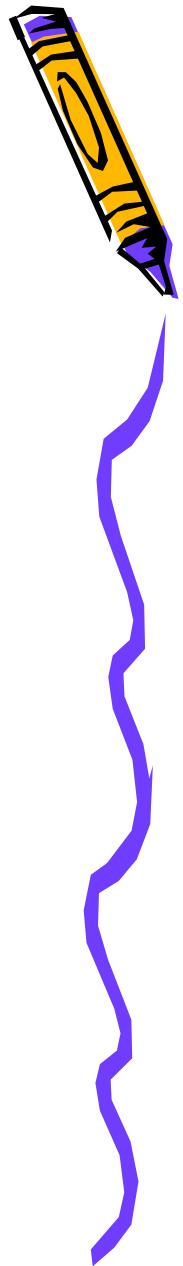
Decoder 2*4

Inputs		Outputs				
	A	B	D3	D2	D1	D0
0	0	0	0	0	0	1
1	0	1	0	0	1	0
2	1	0	0	1	0	0
3	1	1	1	0	0	0

Inputs = 2

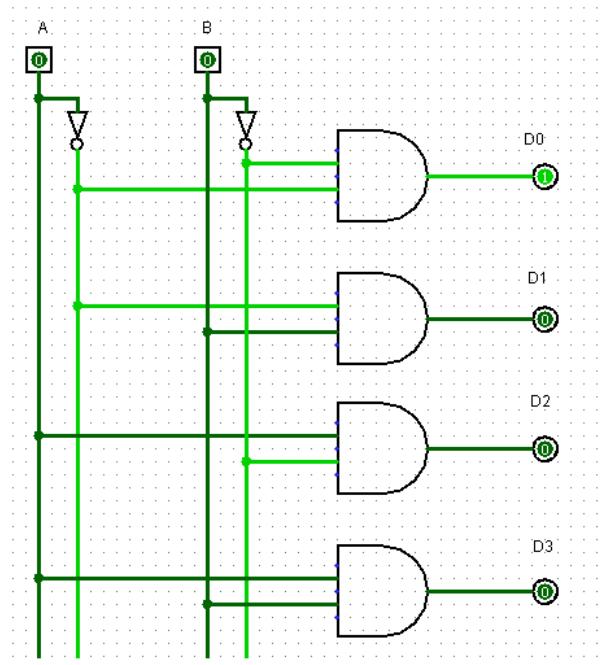
Outputs = 4



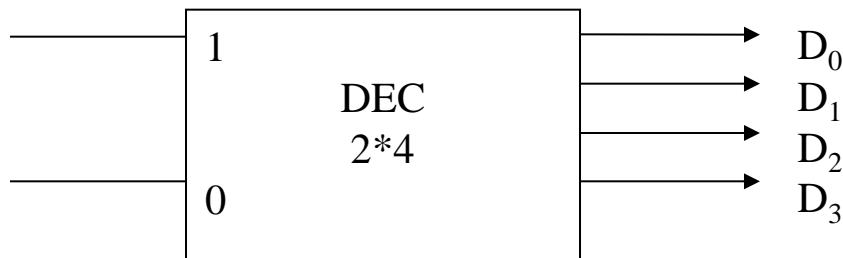


Decoder Block Diagram 2×4

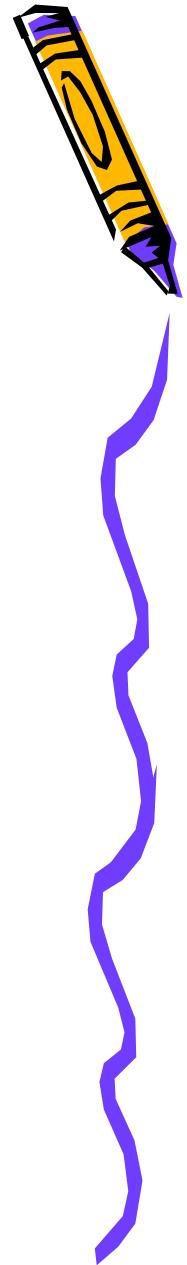




Logic diagram for DEC 2×4

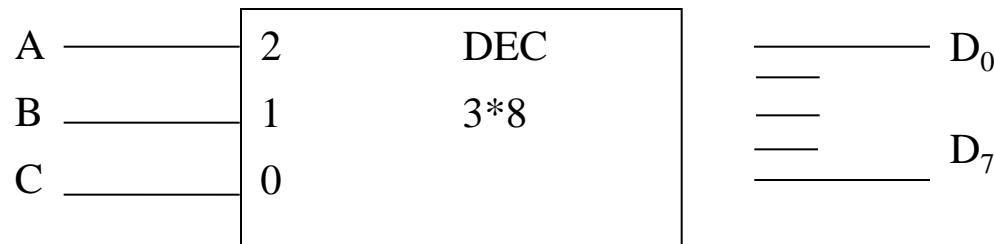


Decoder Block Diagram 2×4

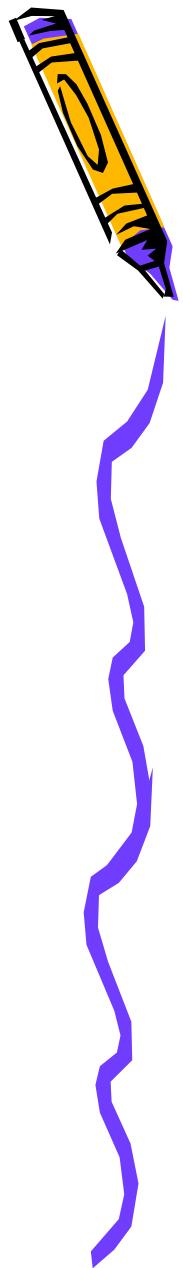
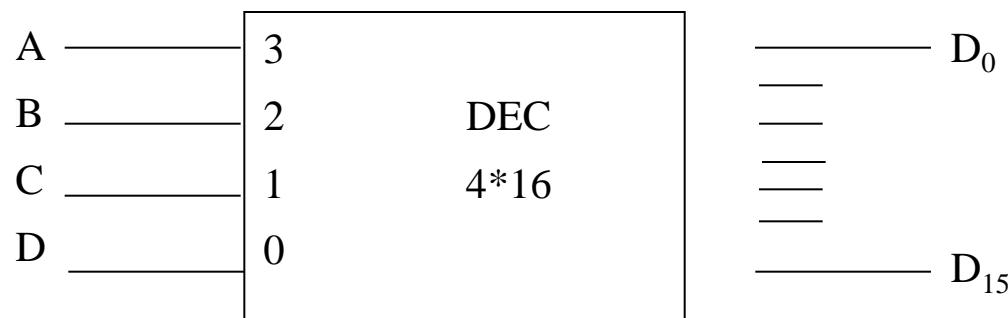




Decoder 3*8



Decoder 4*16

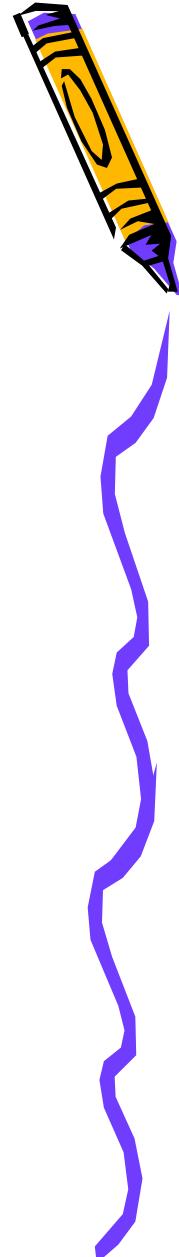




Decoder with enable

Decoder works when enable = 1

Decoder doesn't work when enable = 0



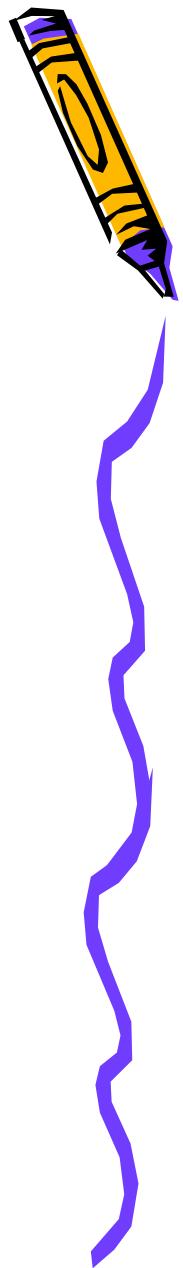
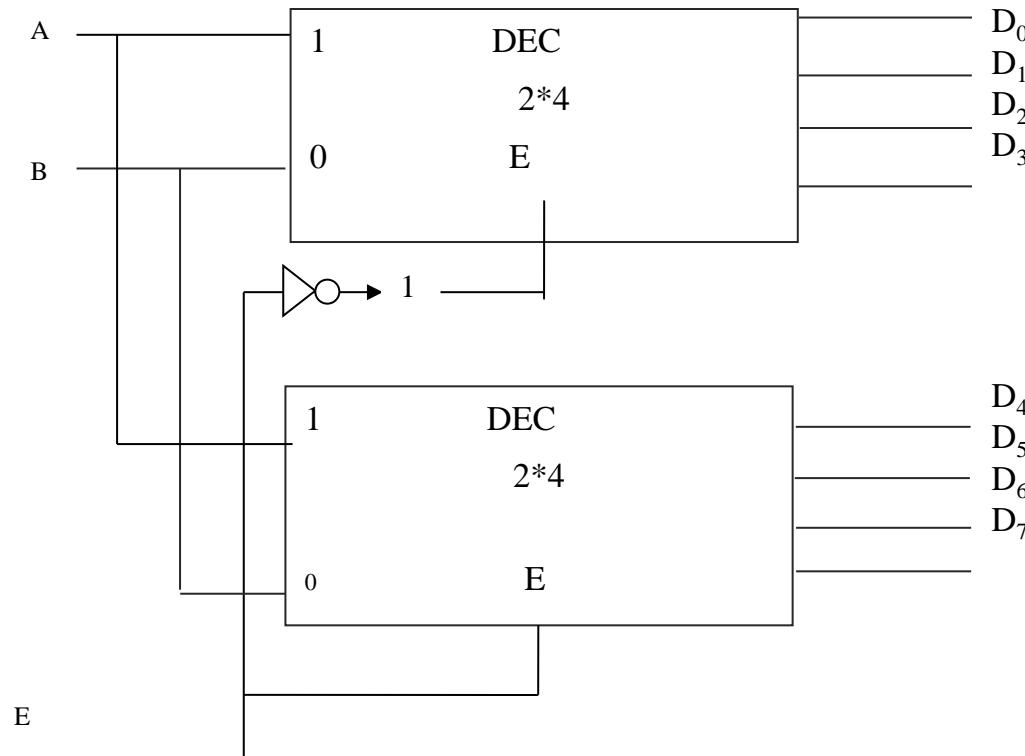
Inputs			Outputs			
E	A	B	D3	D2	D1	D0
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Inputs			Outputs			
E	A	B	D3	D2	D1	D0
0	×	×	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0



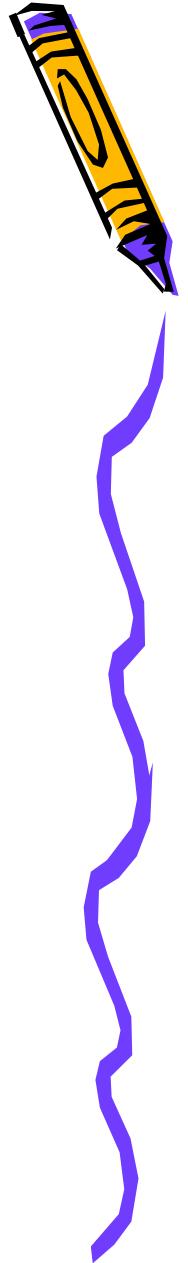
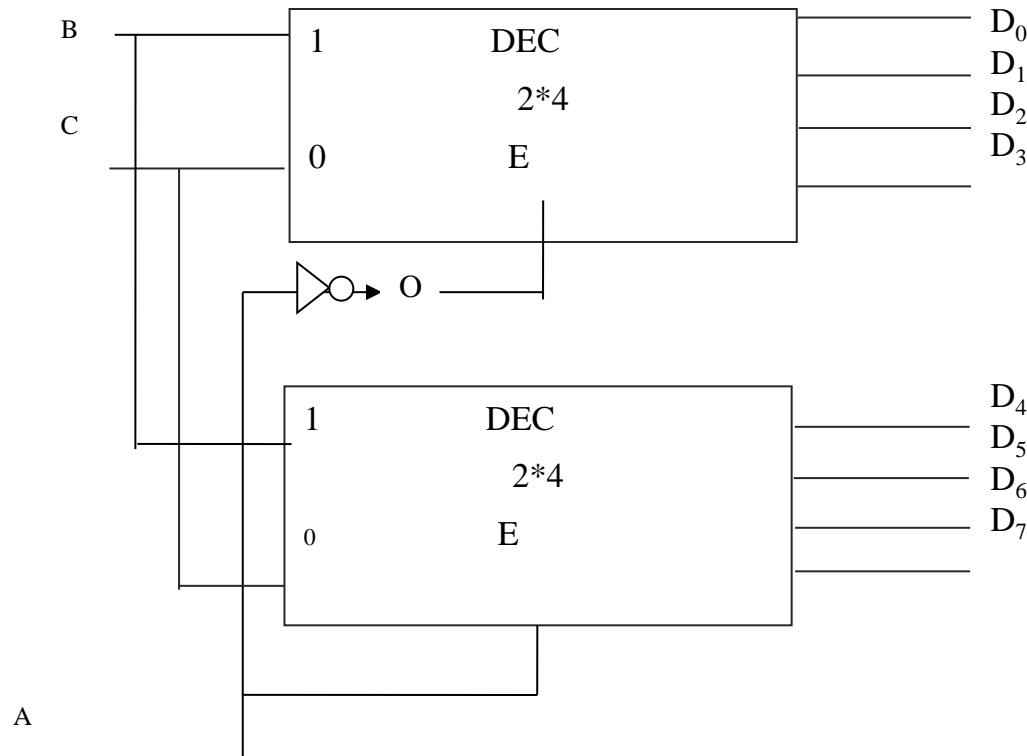


Designing 3*8 Decoder using 2*4 decoder and enable.





Designing 3*8 Decoder using 2*4 decoder and enable.





Multiplexer

A multiplexer has 2^n inputs, 1 outputs and n selections

$$N = 3$$

$$= 8 \text{ المدخلات} \Rightarrow 2^n$$

$$= 1 \text{ المخرجات}$$

$$= 3 \text{ الاختيارات}$$

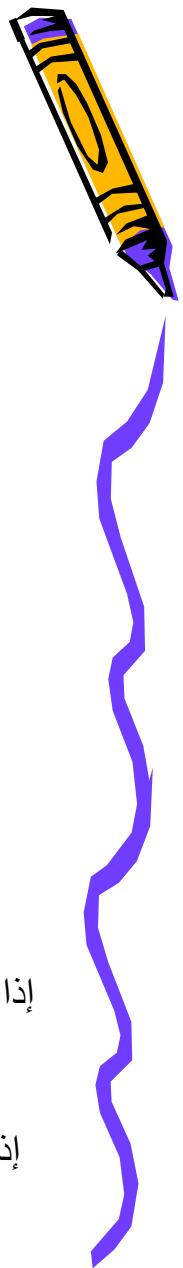
Multiplexer 2*1

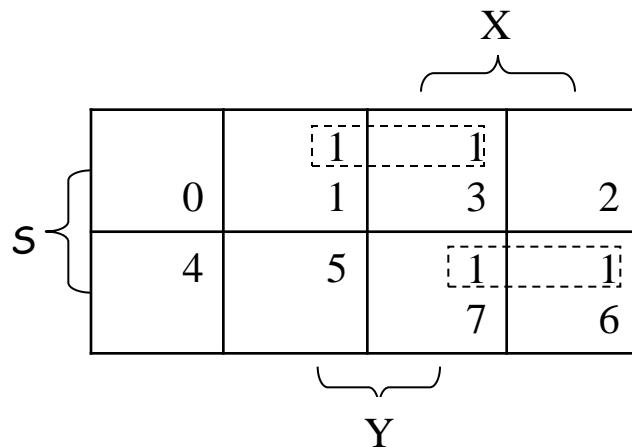
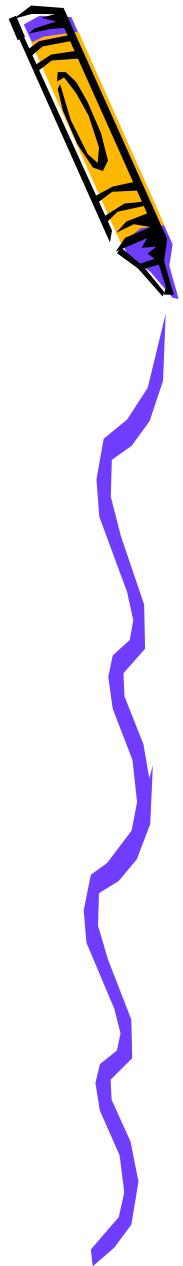
2 inputs 1 outputs 1 selection

selection	Inputs		Outputs
S	X	Y	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

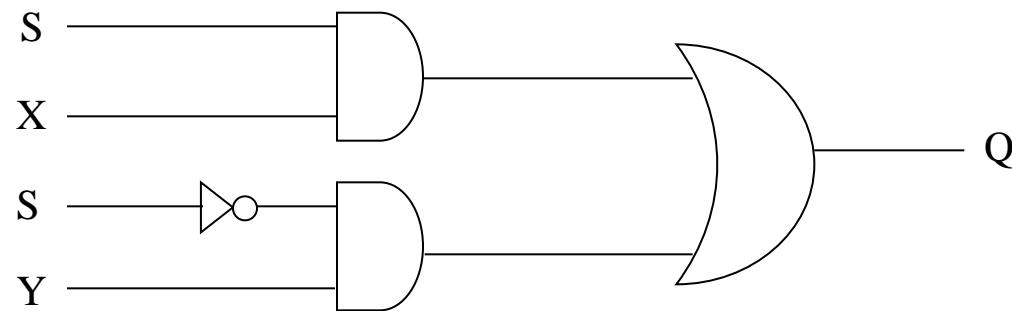
إذا كانت قيمة S = 0 فإن قيمة المخرج Q هي قيمة Y

إذا كانت قيمة S = 1 فإن قيمة المخرج Q هي قيمة X

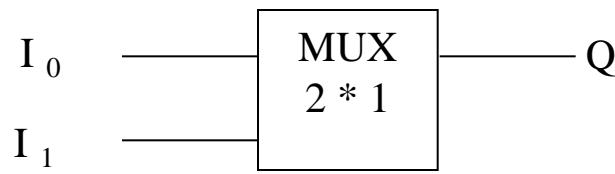




$$Q = Y \ S' + X \ S$$

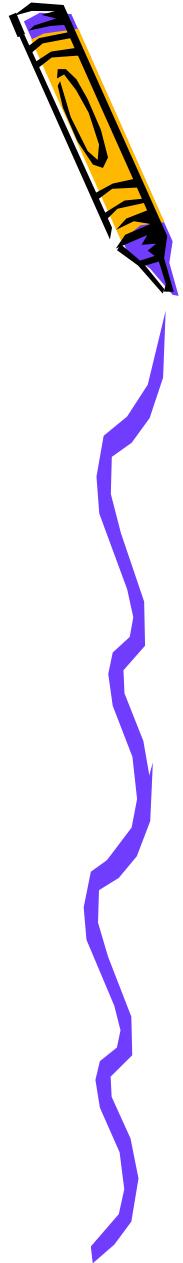
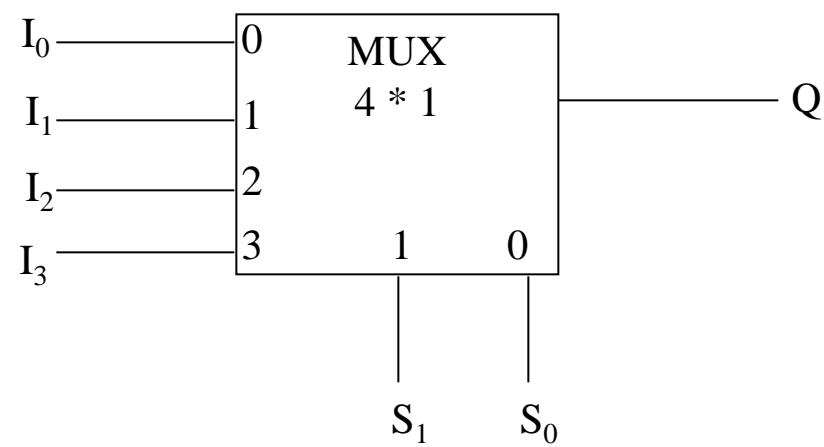


Block D





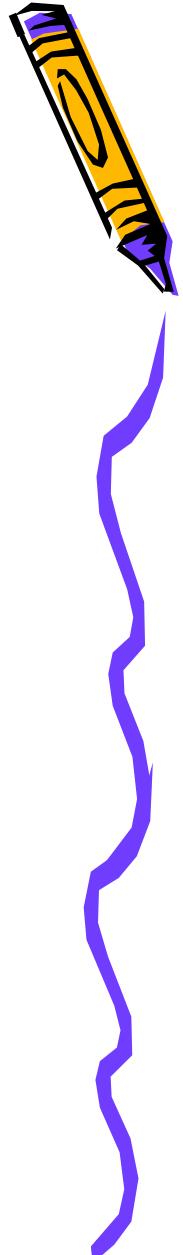
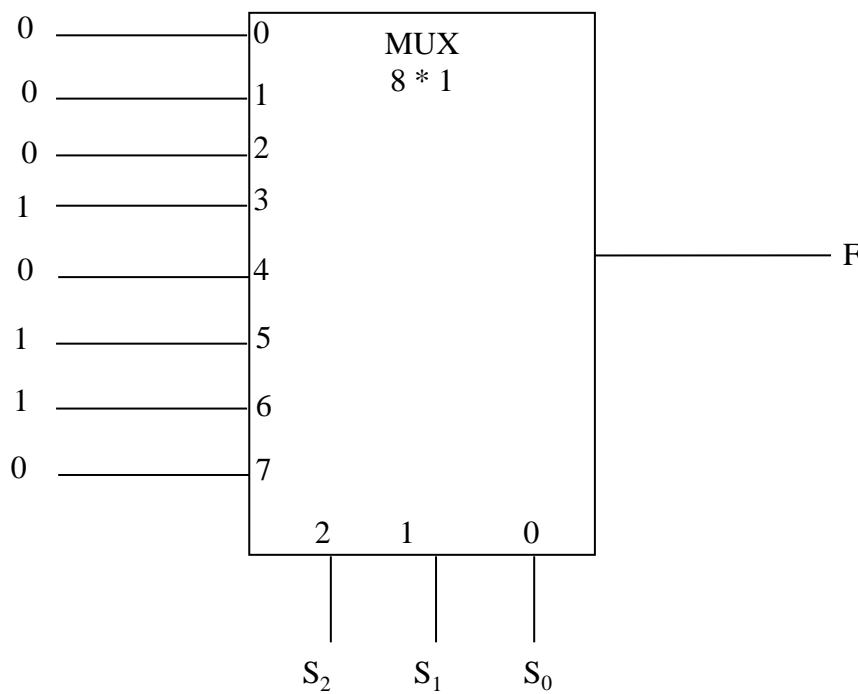
Multiplexer $4 * 1$

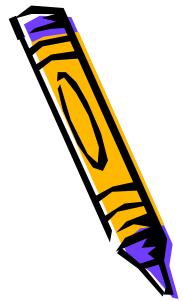




Example :

Implement the following function $F(X,Y,Z) = \sum (3,5,6)$
using an $8 * 1$ Multiplexer

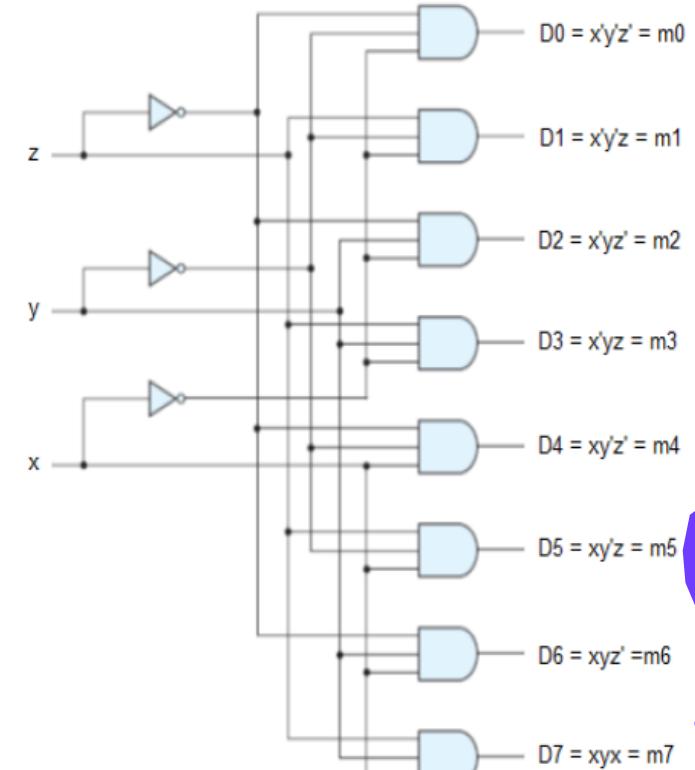
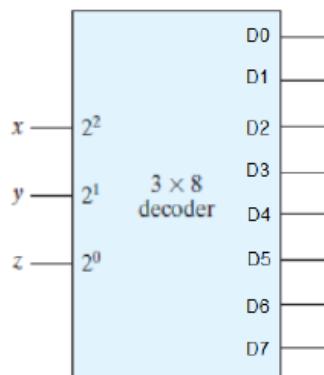


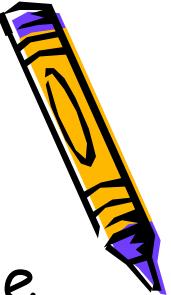


Three-to-Eight Line Decoder

Truth Table of a Three-to-Eight-Line Decoder

Inputs			Outputs							
x	y	z	D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

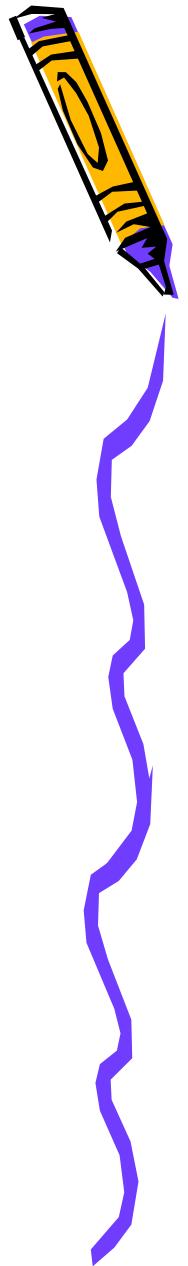




Design Procedure

1. From the specifications of the circuit, determine the required number of inputs and outputs and assigned a letter symbol to each
2. Derive the truth table that defined the relationship between inputs and outputs
3. Obtain the simplified Boolean functions for each output as a function of the input variables
4. Draw the logic diagram
5. Verify the correctness of the design





Thank you



Logical Design

CS 221

* Prof.Dr. Mohamed Osama Khozium

*



Sequential Circuits

Logic circuits is classified into two types :

1 – Combinational logic circuits

Outputs are entirely dependent on the current input

Basic units is the logic gates

يعتمد الخرج فى أى لحظة على المدخلات الموجودة فى تلك اللحظة
وحدة البناء الأساسية البوابات المنطقية

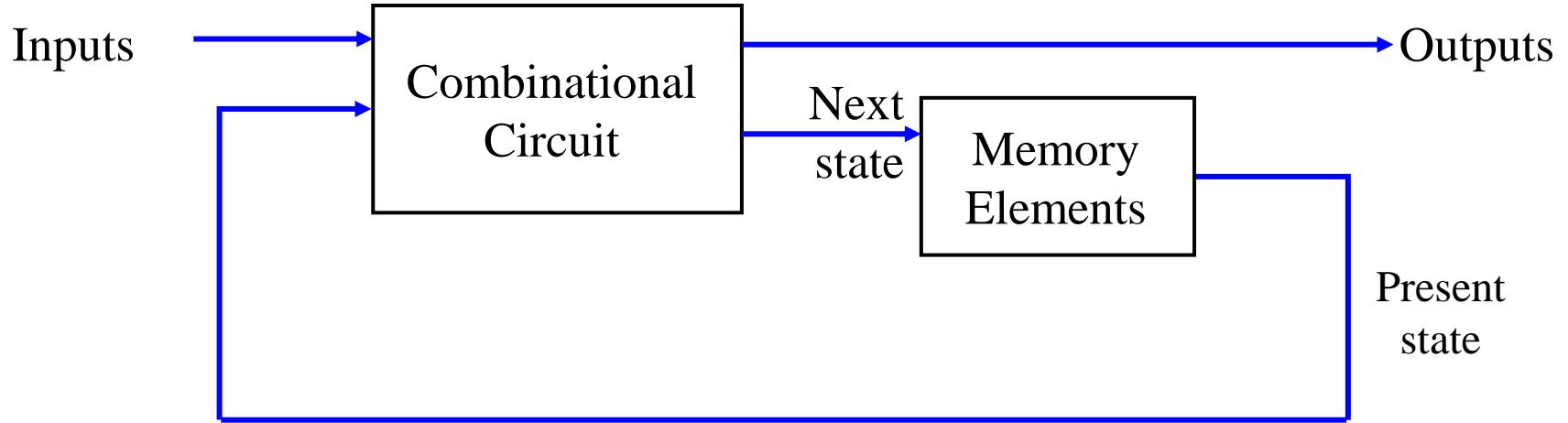


2 – Sequential Circuits

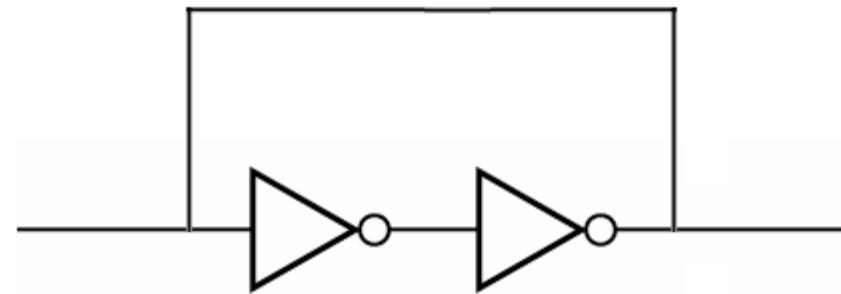
It consists of a combinational circuit to which storage elements are connected to form a feedback path.

The storage elements are devices capable of storing binary information. The basic units is flip flop circuit

يتميز هذا النوع من الدوائر بوجود ذاكرة (memory)
ووحدة البناء الأساسية هي دوائر النطاط .



Block Diagram of Sequential Circuit



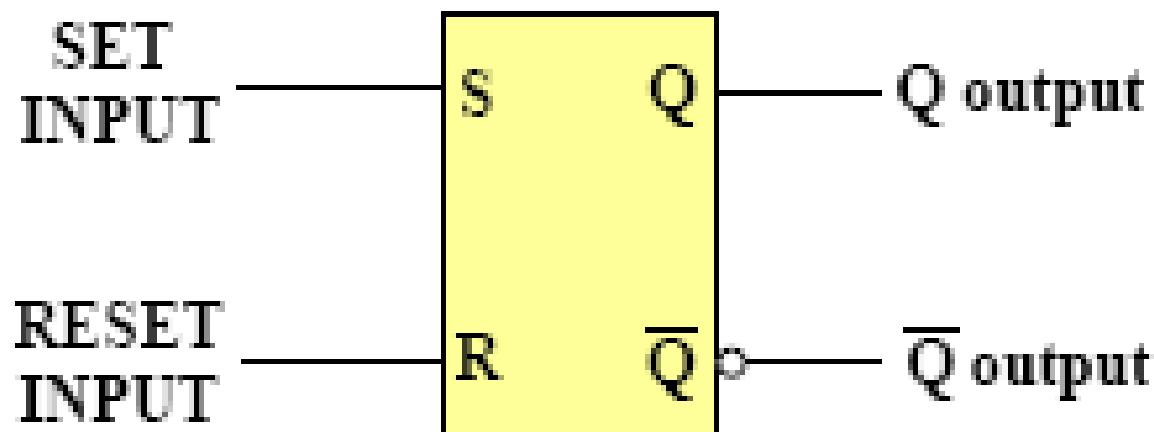
Simple example for memory unit



Latches (المساکات)

Memory element differs from the flip flop in the way used to change its stability condition

عنصر من عناصر التخزين ويختلف عن النطاط في الطريقة المستخدمة للتغيير فقط.

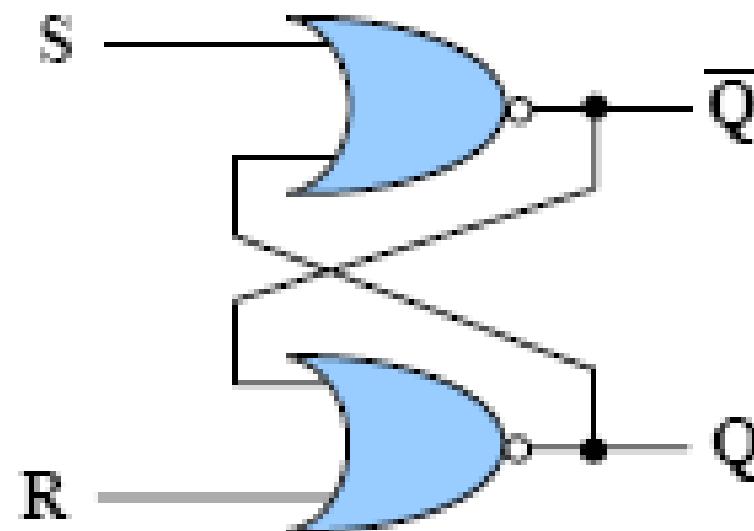


SR Latch Circuit



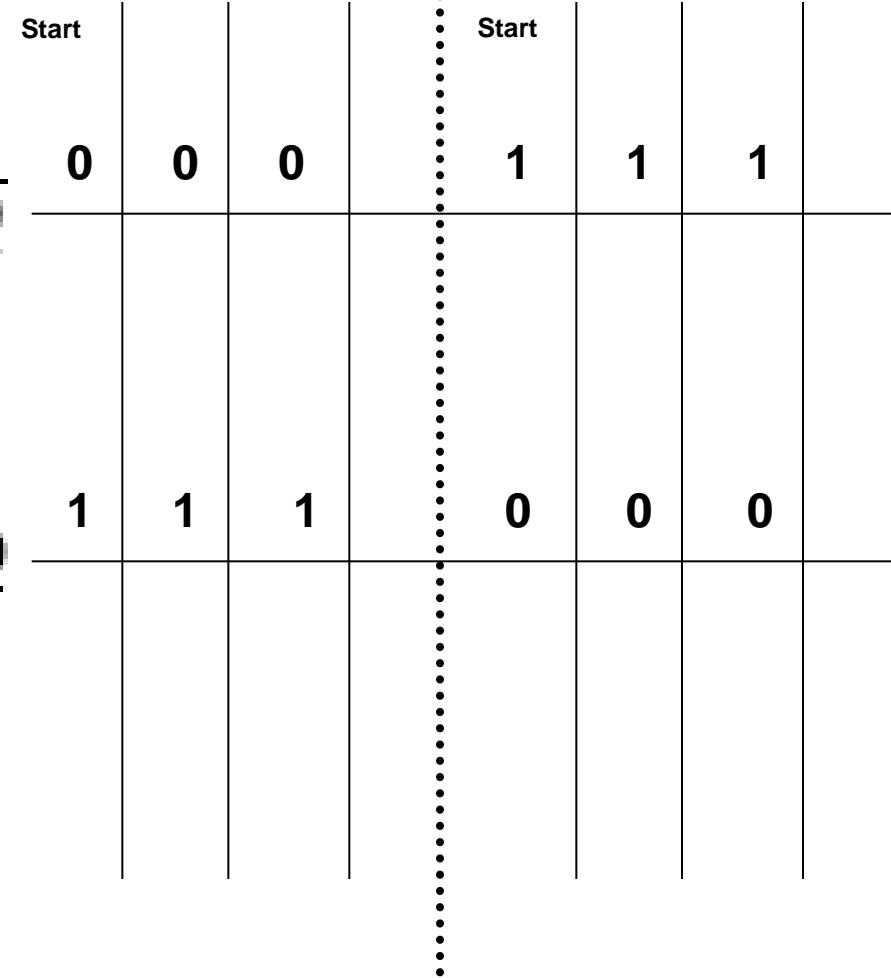
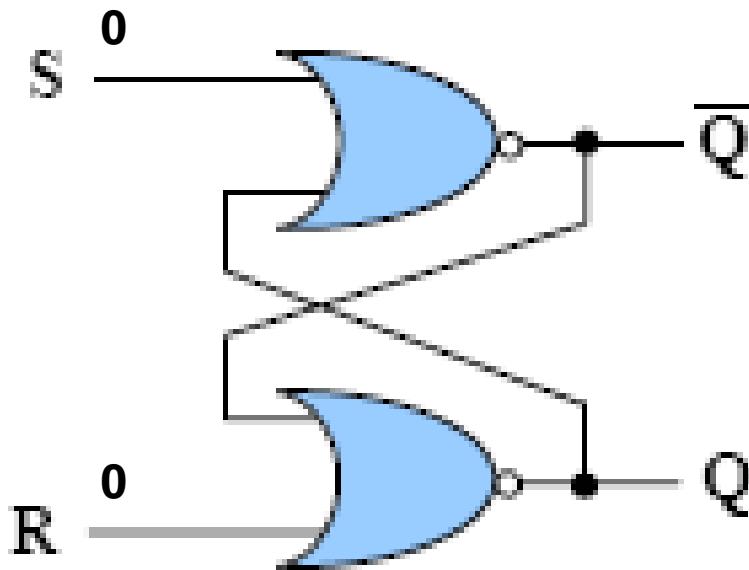
SR Latch Circuit

Could be built by NOR gates using feed back (Active High Inputs)



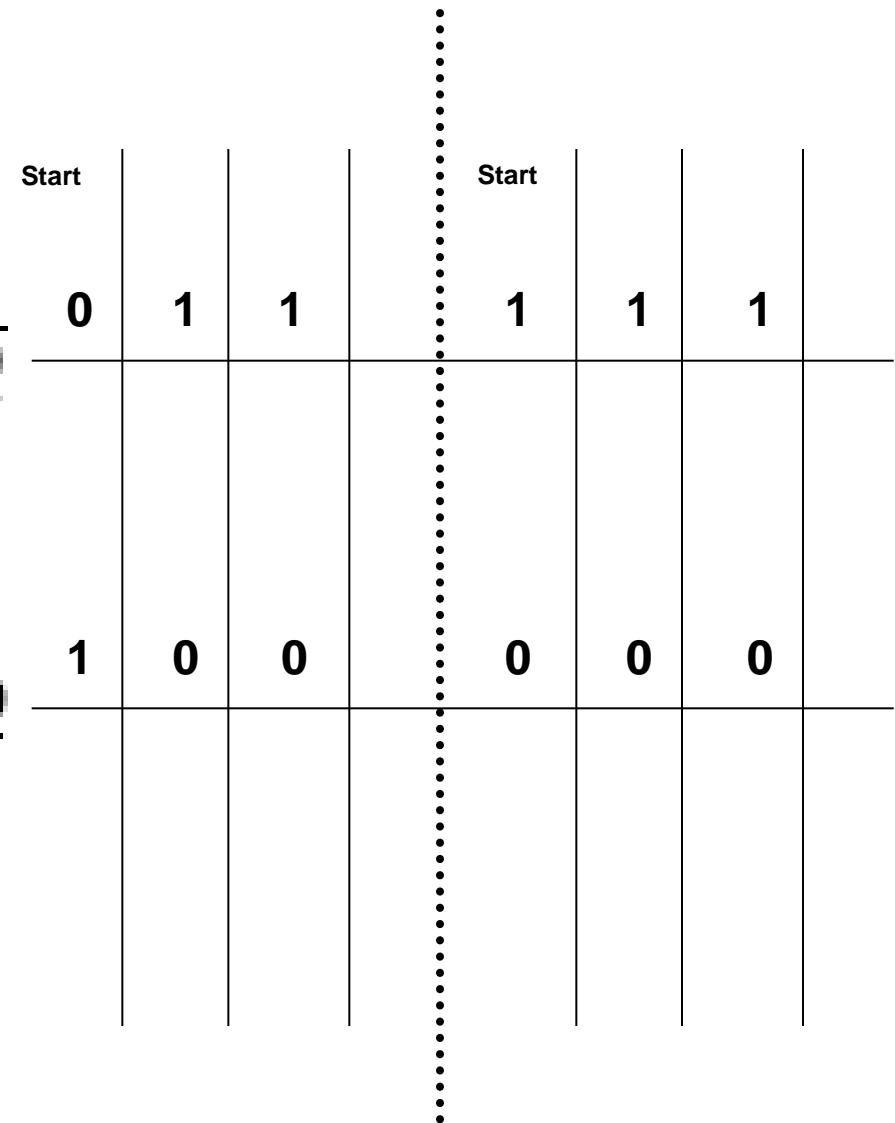
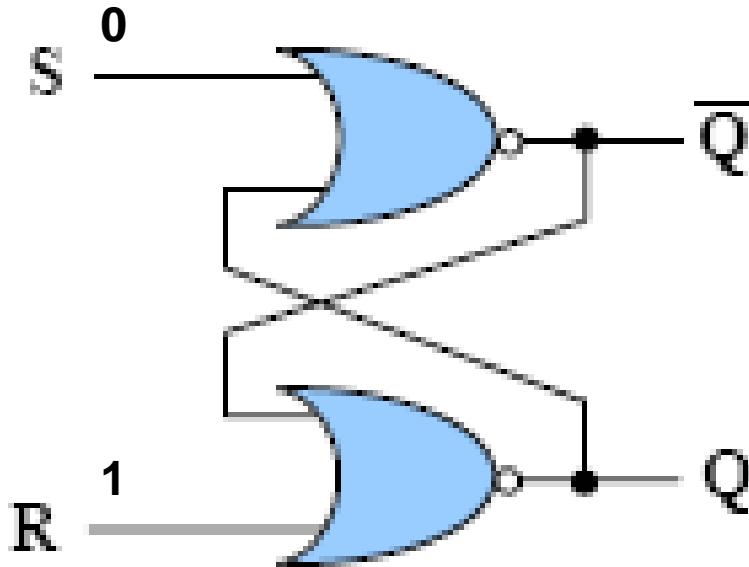


الدخلات		الخرج	وضع التشغيل (Mode of Operation)
S	R	Q	
*	*	Q.	وضع الإمساك (عدم التغيير) No Change
*	٠	*	الوضع الغير فعال Latch RESETS
٠	*	*	الوضع الفعال Latch SETS
٠	٠	?	وضع الخطأ أو وضع غير مسموح به Invalid condition



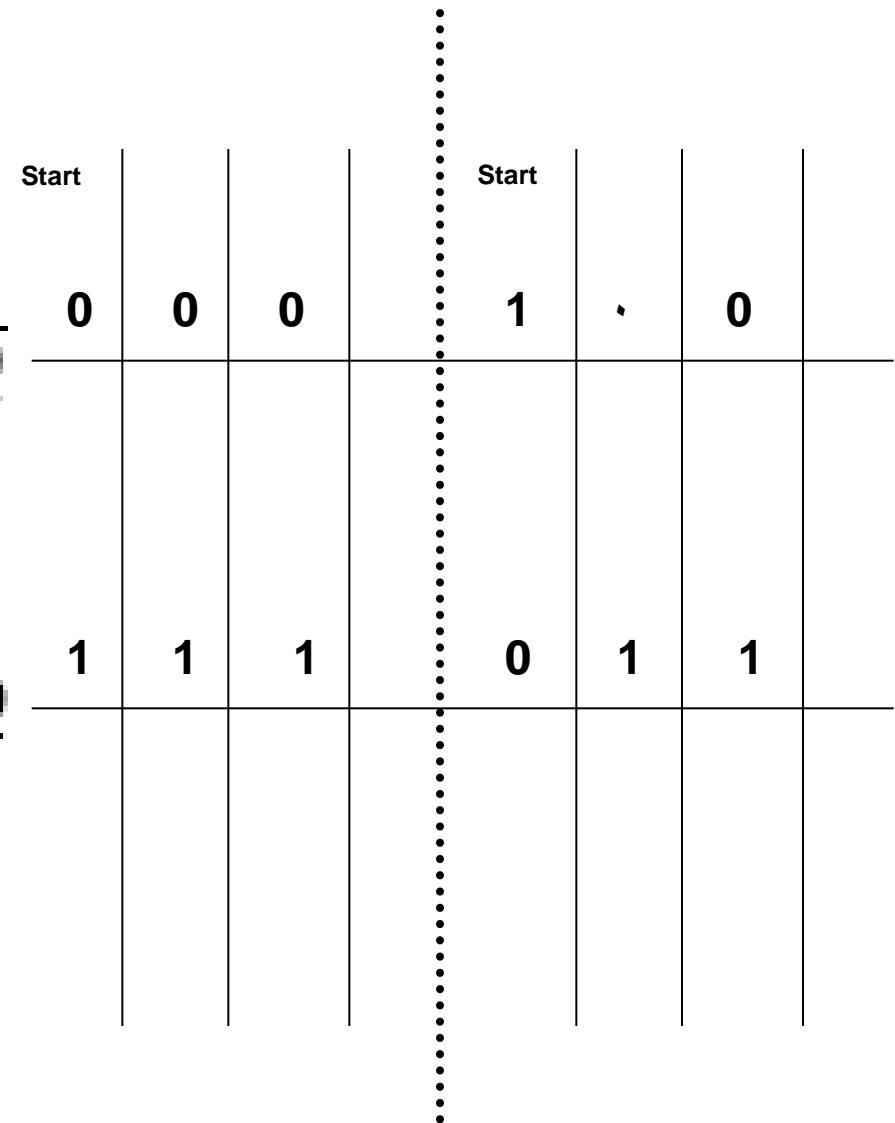
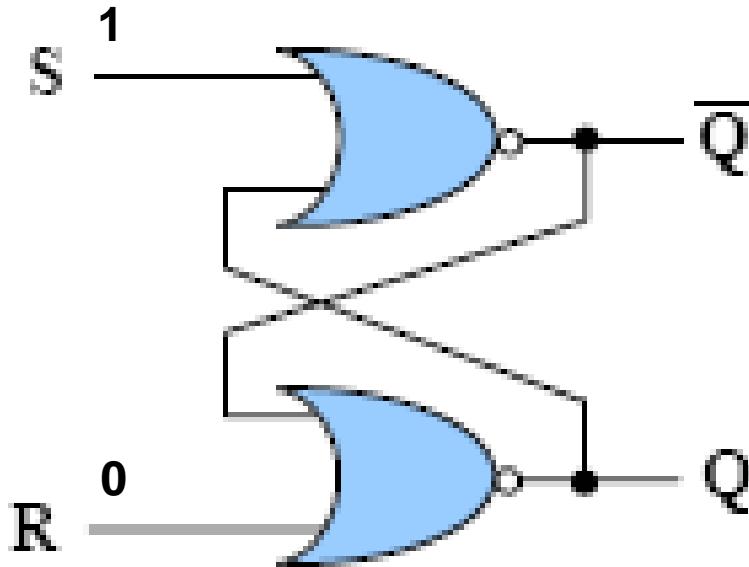
When $S = 0 \text{ & } R = 0$

Condition = No change



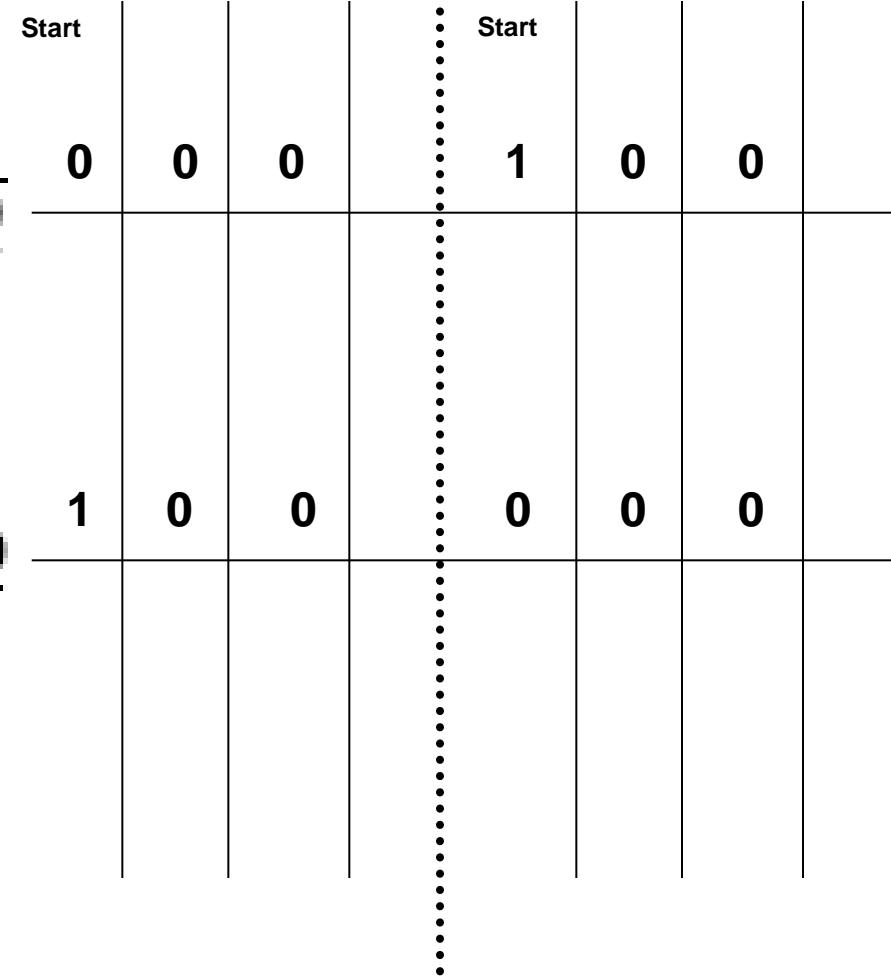
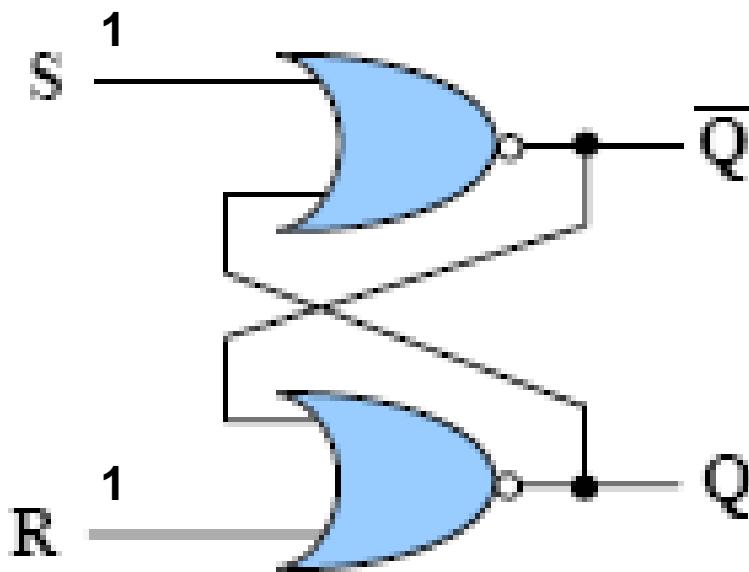
When $S = 0 \text{ & } R = 1$

Condition = Reset ($Q = 0 \text{ & } Q' = 1$)



When $S = 1$ & $R = 0$

Condition = Set ($Q = 1$ & $Q' = 0$)



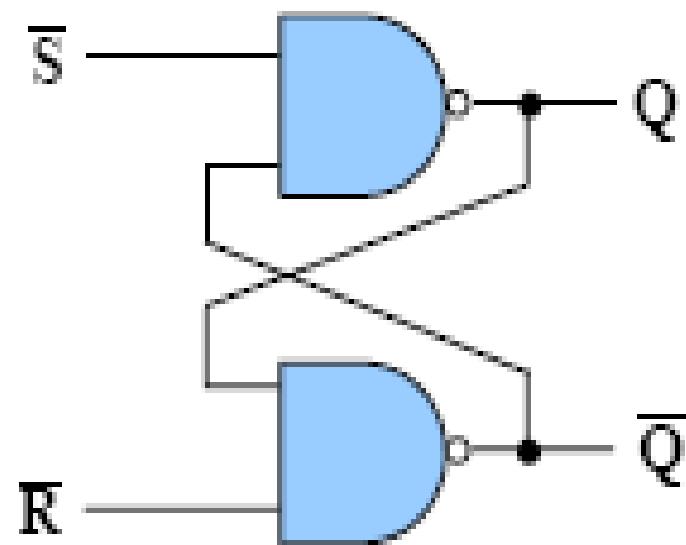
When $S = 1$ & $R = 1$

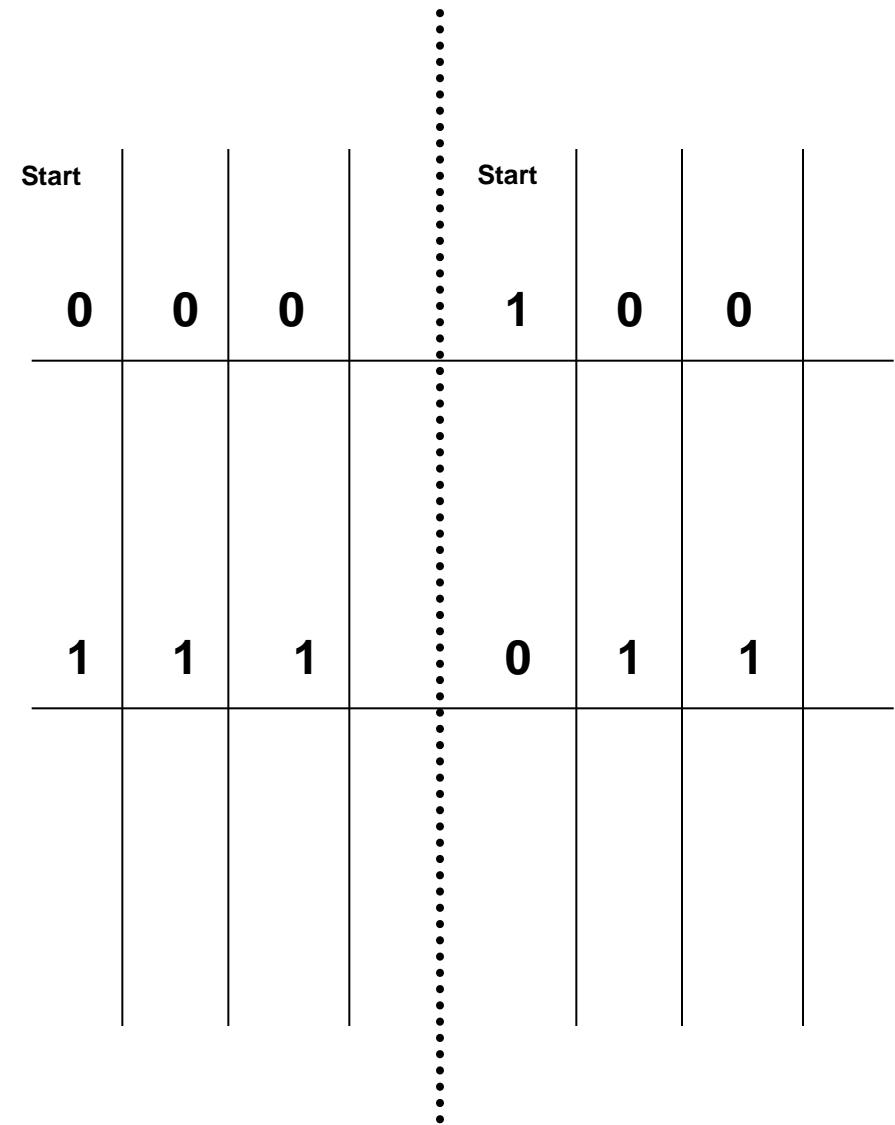
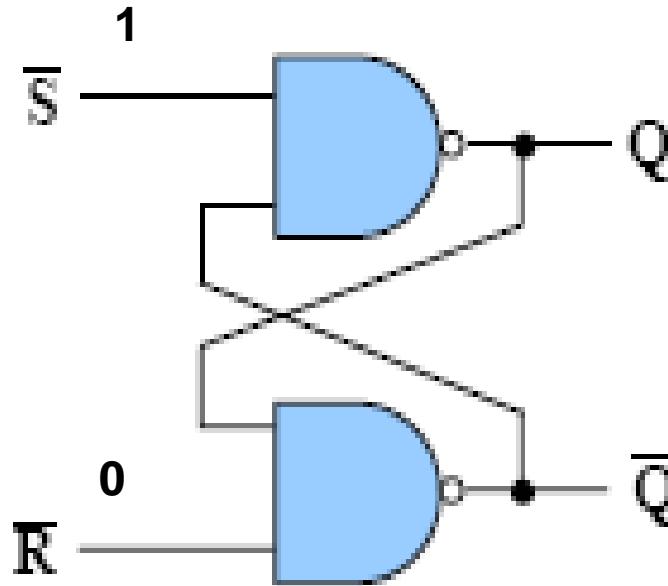
Condition = invalid condition



Using Nand Gate for SR Latch (Active low inputs)

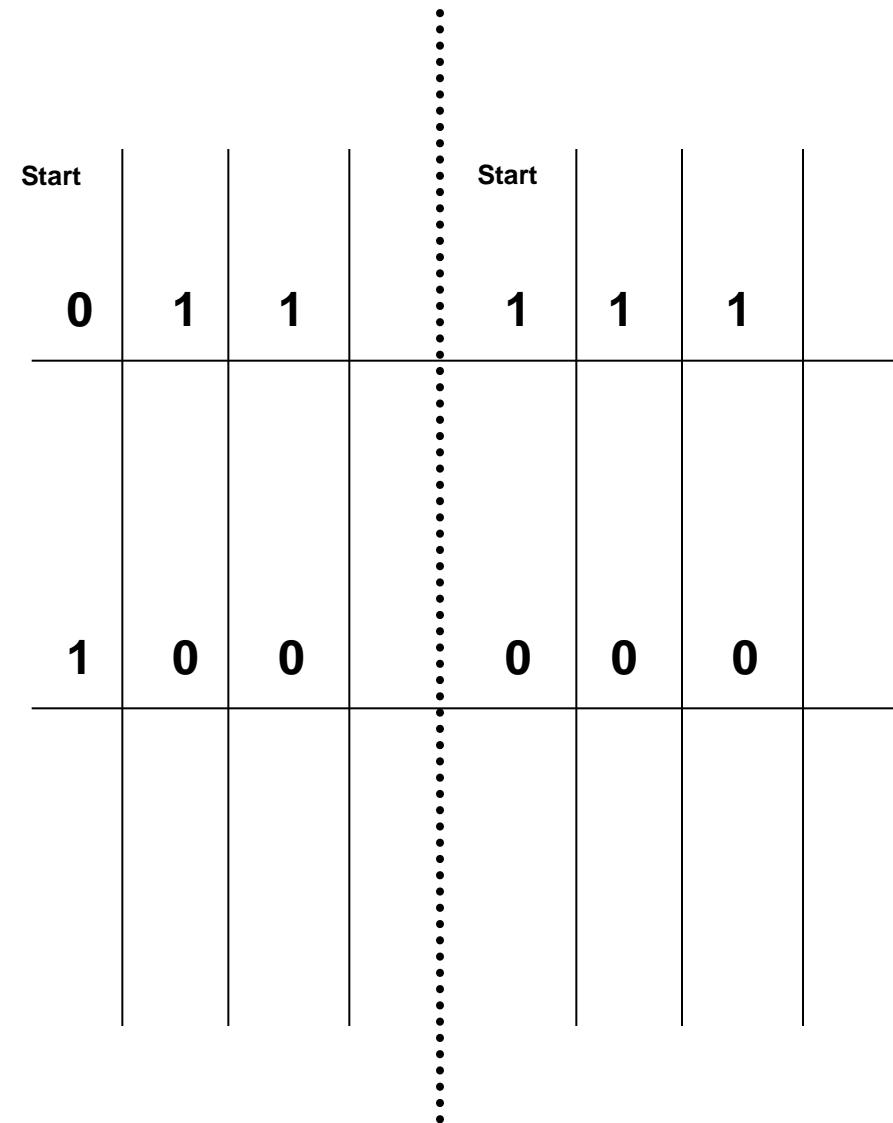
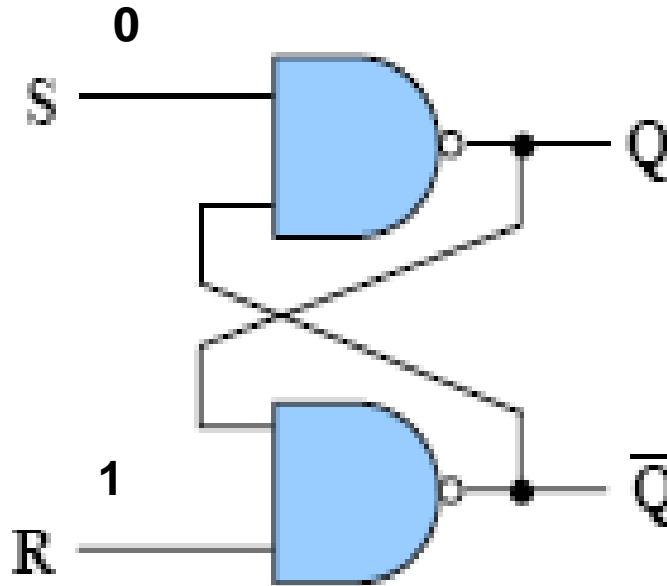
لاحظ أن المستوى الفعال لدائرة ال (NAND) هو الصفر وبالتالي الخطأ يكون عند $S = 0$ & $R = 0$





When $S = 1$ & $R = 0$

Condition = Reset ($Q = 0$ & $Q' = 1$)

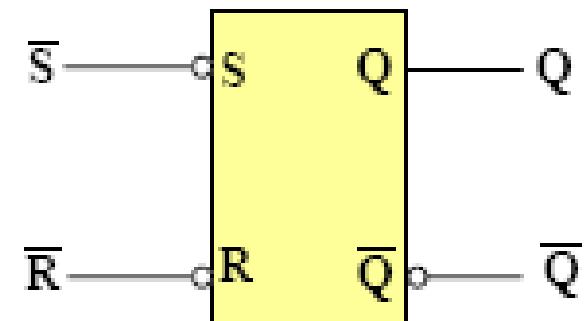
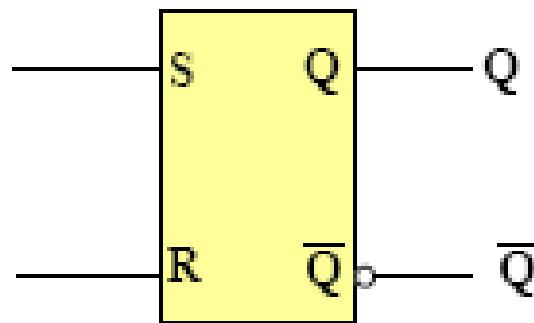


When $S = 1 \text{ & } R = 0$

Condition = Reset ($Q = 0 \text{ & } Q' = 1$)

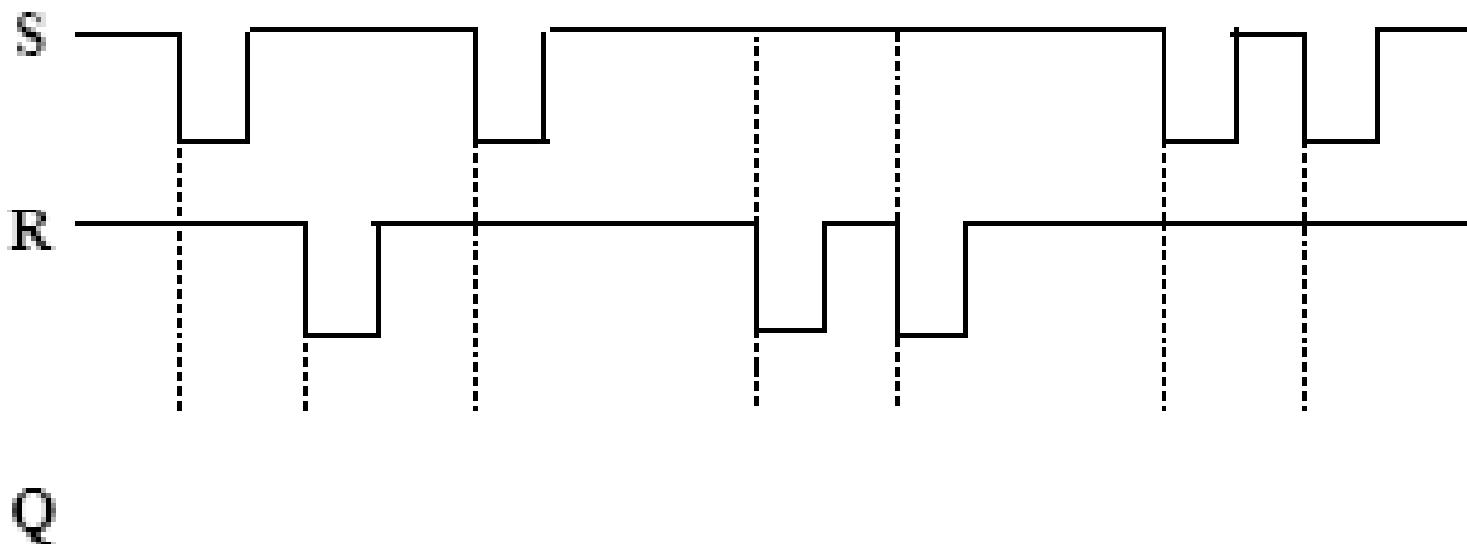


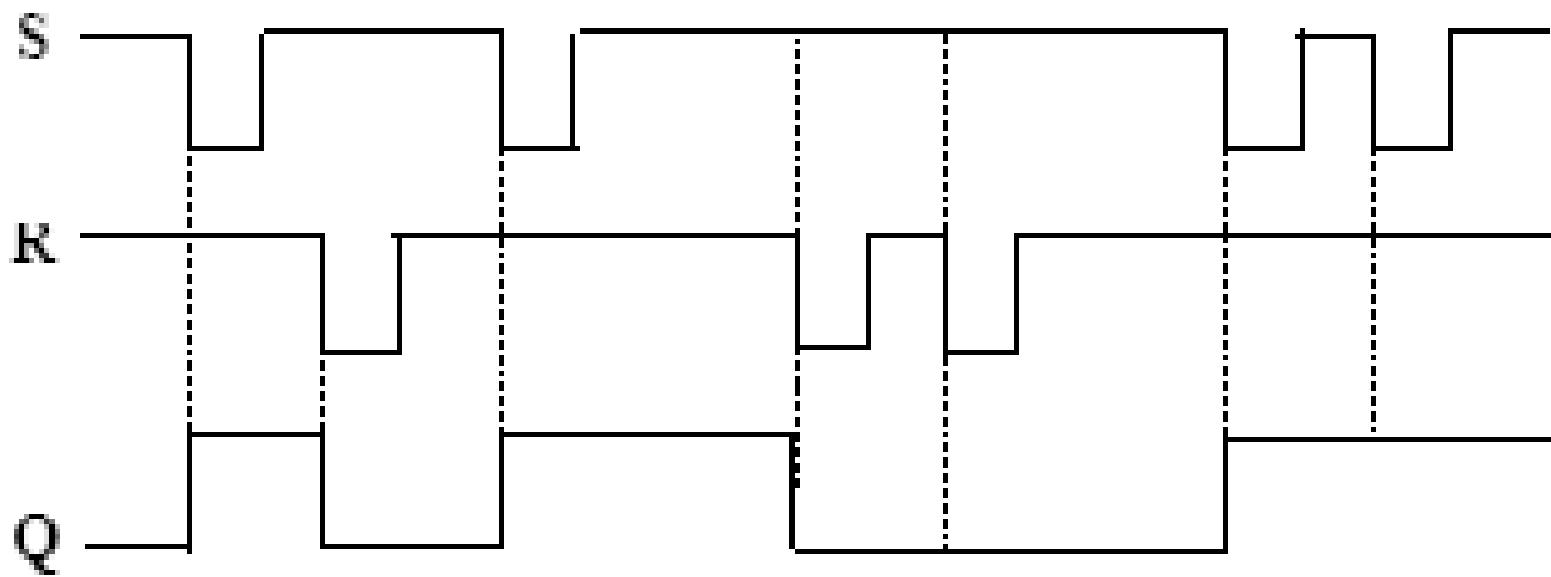
الدخلات		الخرج	وضع التشغيل (Mode of Operation)
\bar{S}	\bar{R}	Q	
*	*	?	وضع الخاطئ أو وضع غير مسموح به Invalid condition
*	٠	١	الوضع الفعال Latch SETS
١	*	*	الوضع غير الفعال Latch RESETS
١	١	$Q_{\text{.}}$	وضع الإمساك (عدم التغير) No Change





The following is a timing diagram for SR (Active low inputs)
Draw the timing diagram for Q (Initial Q = 0) ... Note the
timing clock.

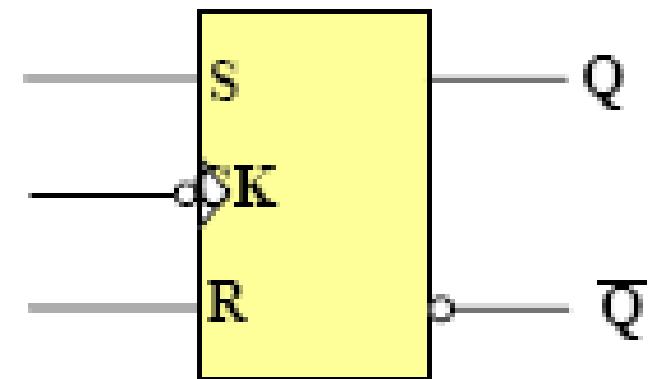
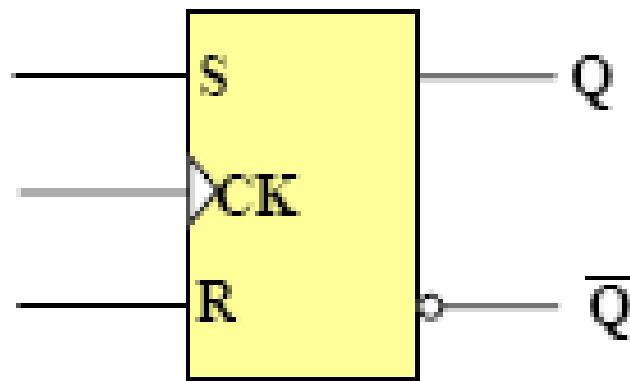






S-R Flip Flop

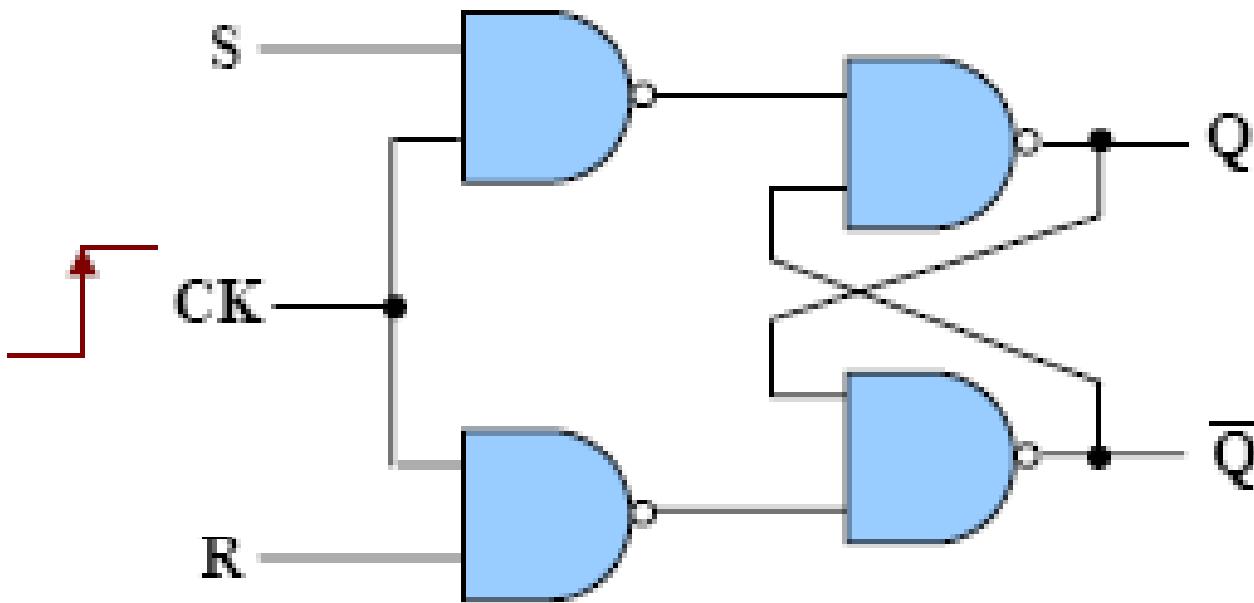
Block diagram for clocked S-R Flip Flop







We add another (Two Nand gates)

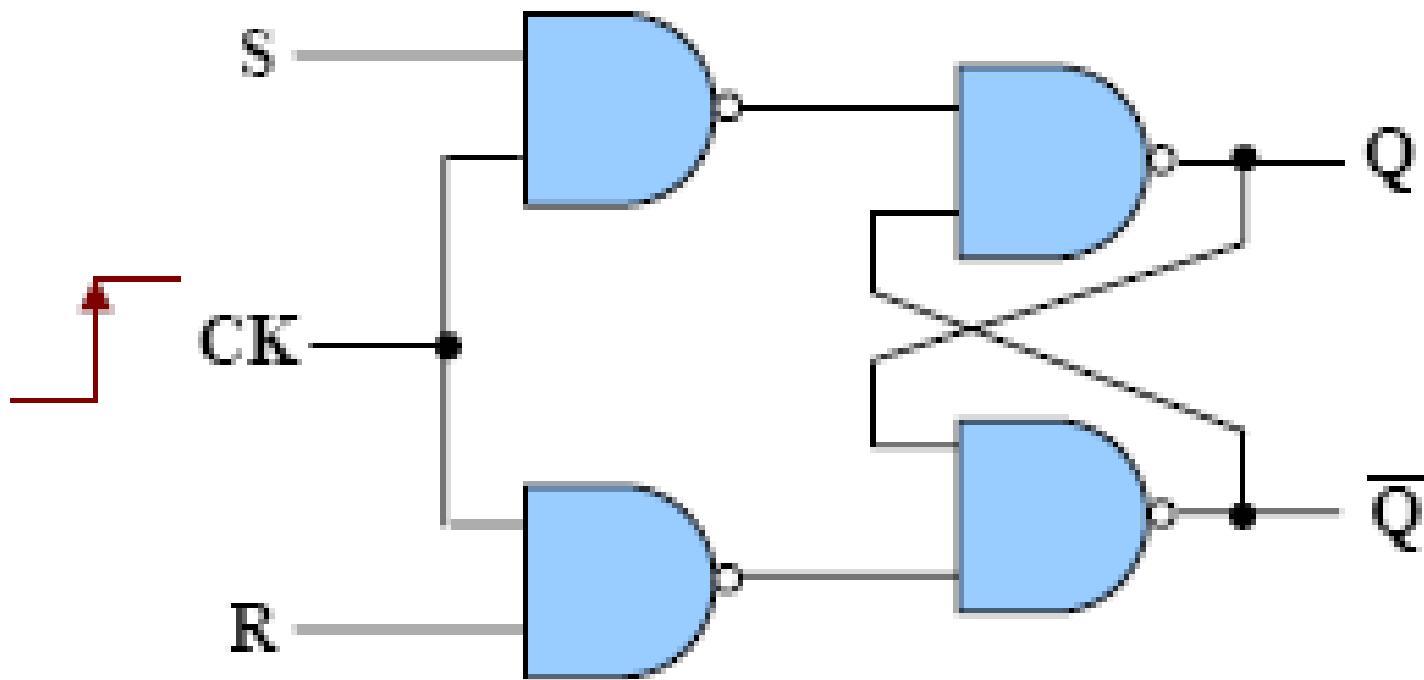


Logic diagram for SR Latch (Flip flop) with control input

(لو فاکر (Active low inputs) مع ال Nand و هو عکس ال Or وبالتالي ال 0 & 0 المشكّلة
هنا مع وجود ال CK و ال Nand الأولى نرجع لل Active high وبالتالي تكون 1 & 1 هي المشكّلة .



المدخلات			الخرج	وضع التشغيل (Mode of Operation)
S	R	CK	Q	
*	*	X	Q.	وضع الإمساك (عدم التغيير) No Change
*	1	↑	.	الوضع غير الفعال Latch RESETS
1	*	↑	1	الوضع الفعال Latch SETS
1	1	↑	?	وضع الخطأ أو وضع غير مسموح به Invalid condition

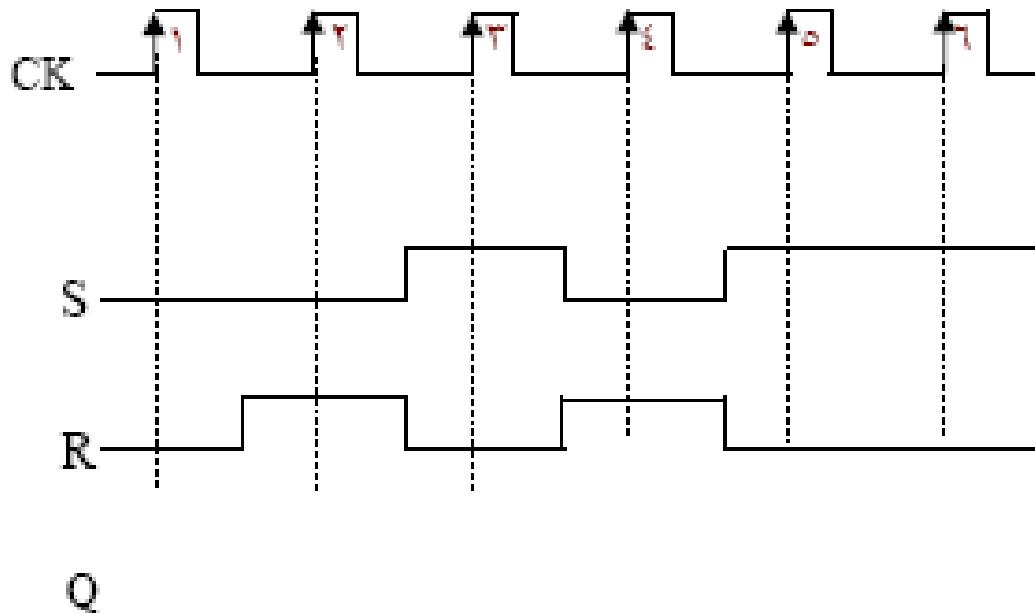


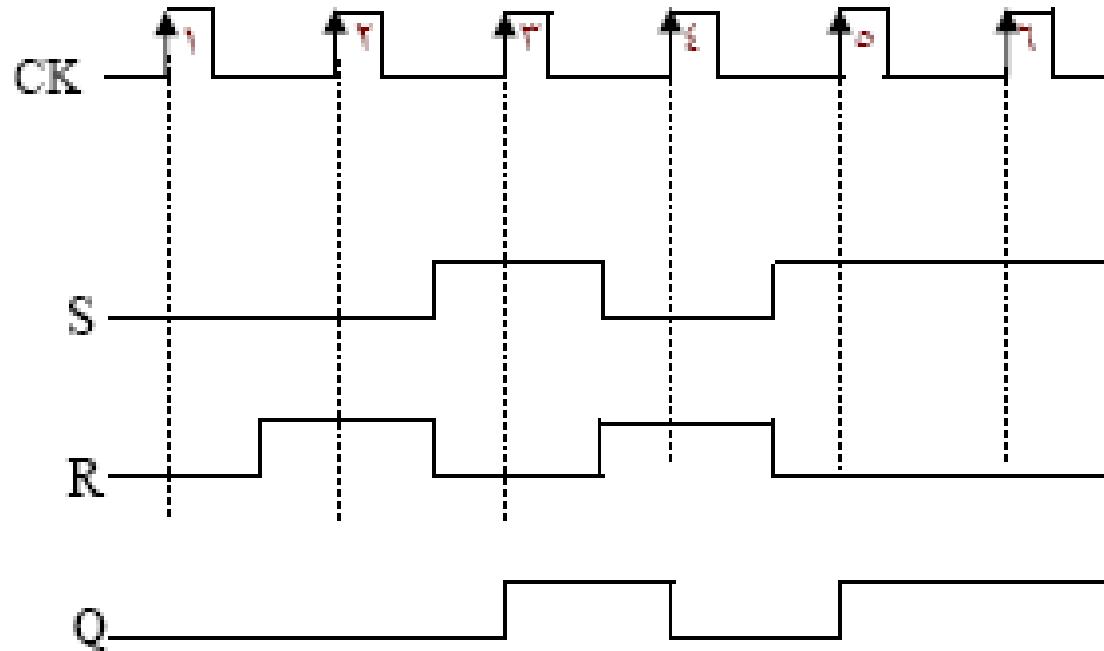


The following is timing diagram for S R flip flop with control input (S , R , K)

Draw the timing diagram for Q (initial Q = 0) ...

Note the timing clock.



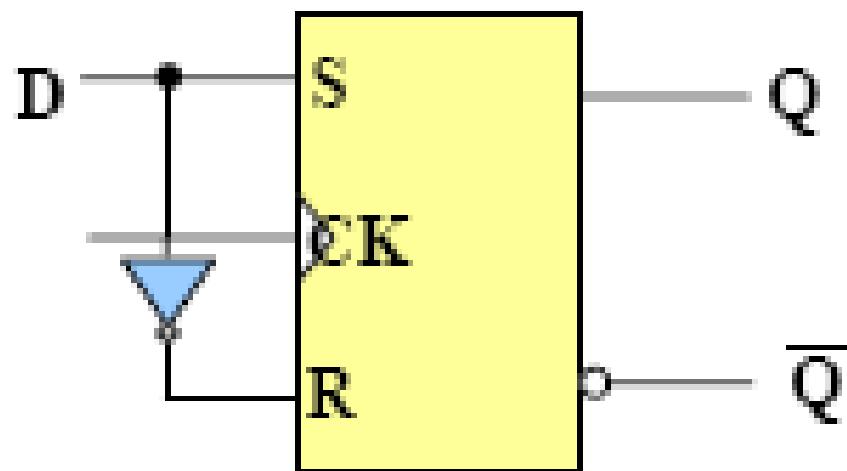


- 1 – First pulse $S = 0$ & $R = 0$ Then $Q = 0$
- 2 – Second pulse $S = 0$ & $R = 1$ Then Q still 0
- 3 – Third pulse $S = 1$ & $R = 0$ then $Q \rightarrow 1$
- 4 – Fourth pulse $S = 0$ & $R = 1$ then $Q \rightarrow 0$
- 5 – Fifth pulse $S = 1$ & $R = 0$ then $Q \rightarrow 1$
- 6 – Sixth pulse $S = 1$ & $R = 0$ then Q still 1



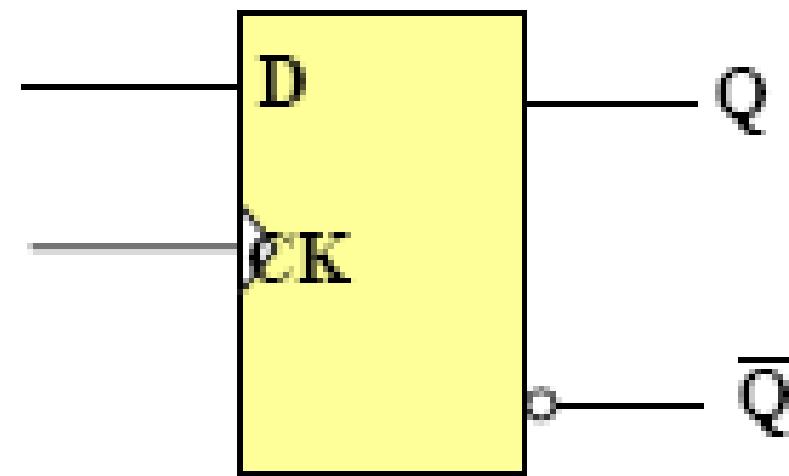
D type flip flop

Memory unit for single bitZero or One

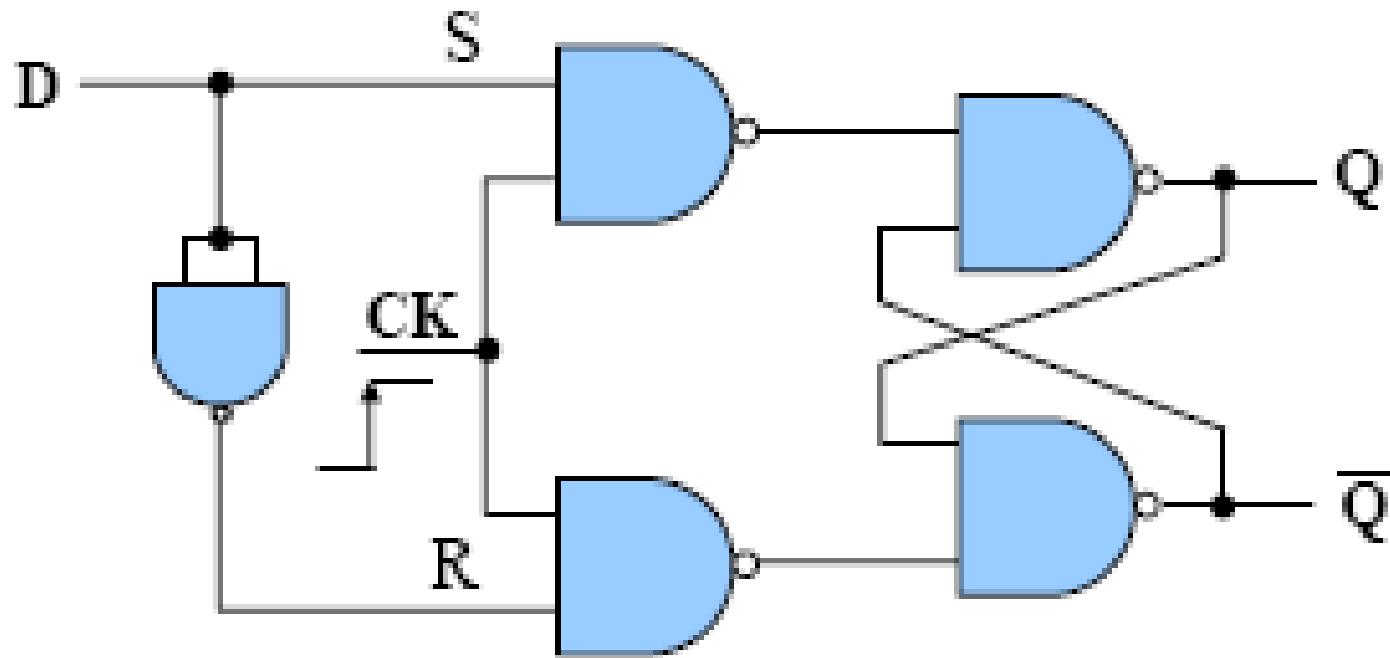




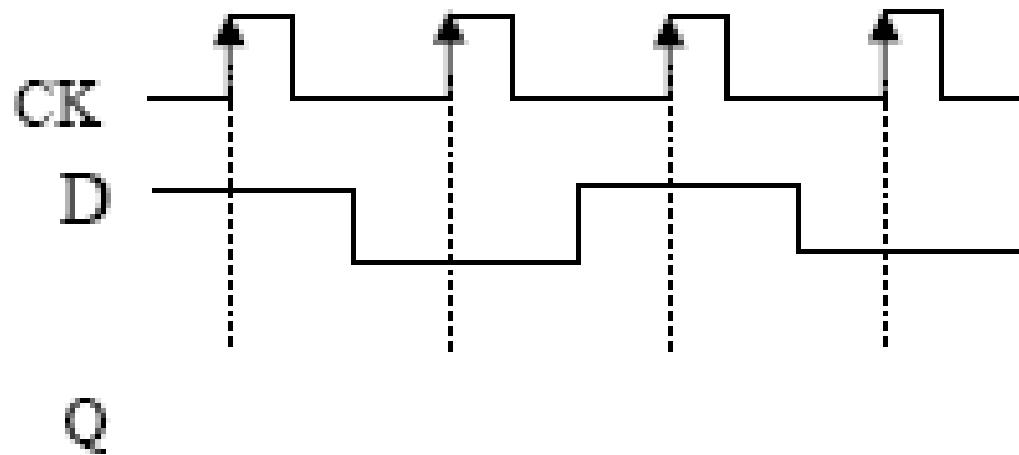
الدخلات		الخرج	وضع التشغيل (Mode of Operation)
D	CK	Q	
1		1	الحالة الفعالة (SET) (stores a 1)
0		0	الحالة غير فعالة (RESET) (stores a 0)

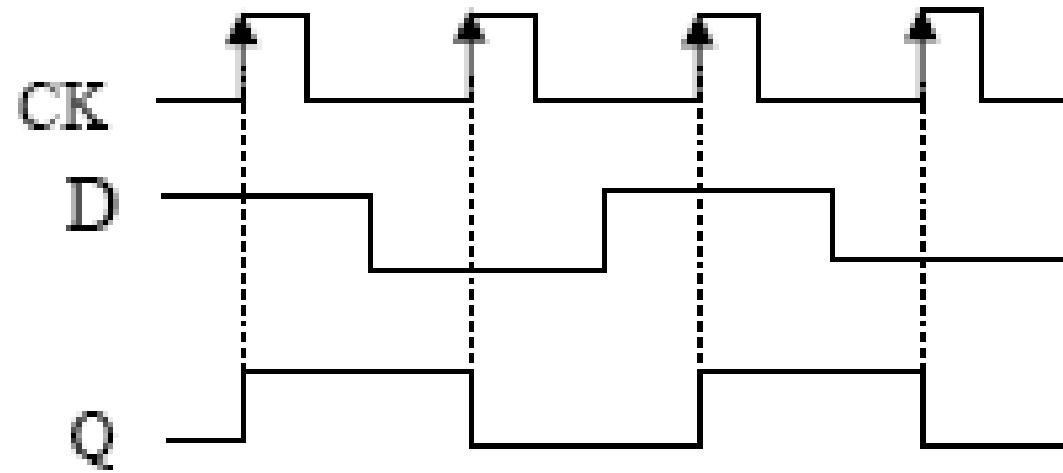


Block diagram for D flip Flop



Logic diagram for D flip Flop



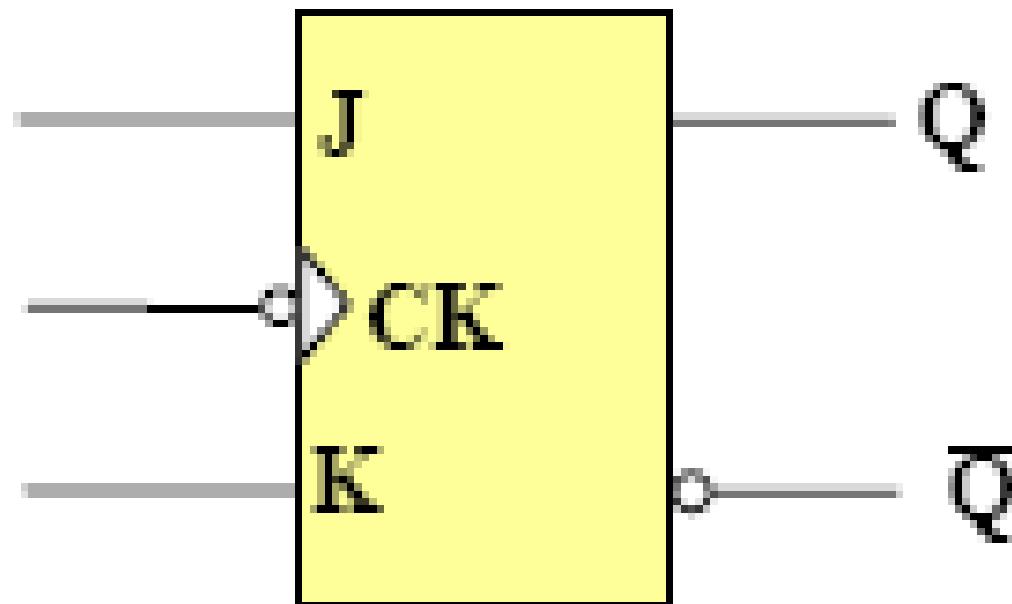




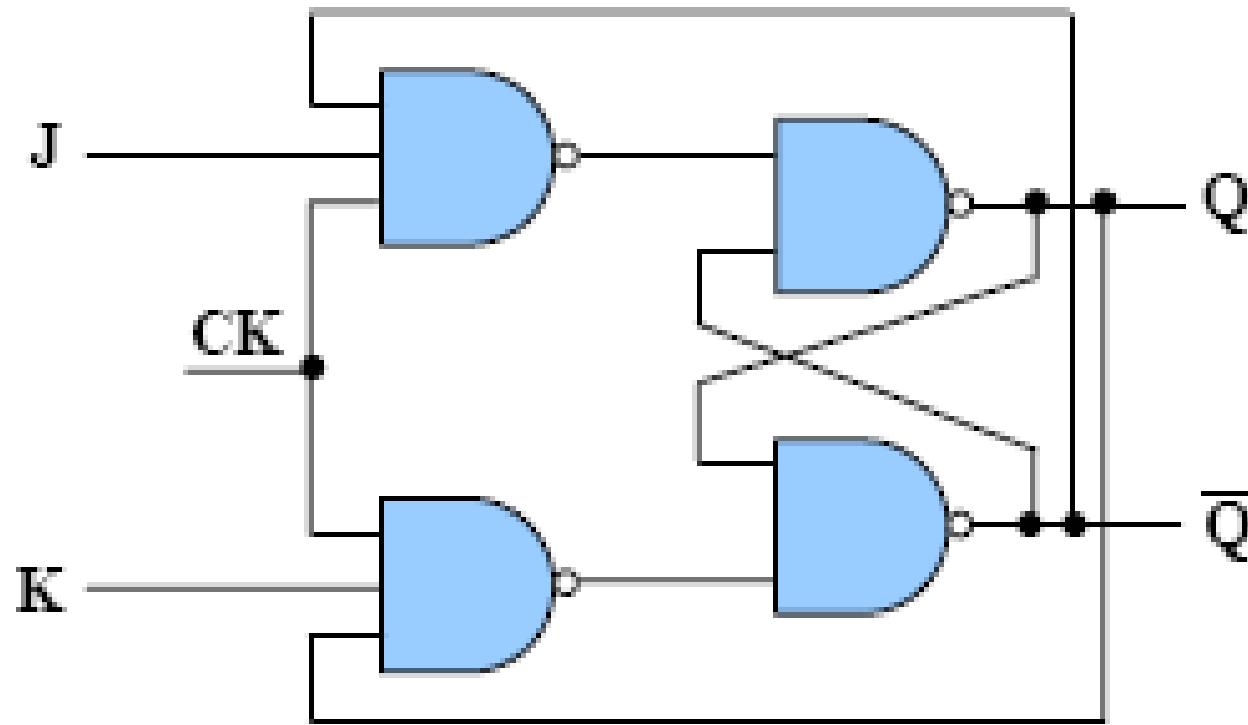
J-K Flip flop

J & K are just letters

J-K flip flop is the same like SR except Q & Q' are used as inputs again ...
then there is no invalid condition.... when J & K = 1 it called toggle condition
where Q becomes Q' and vice versa



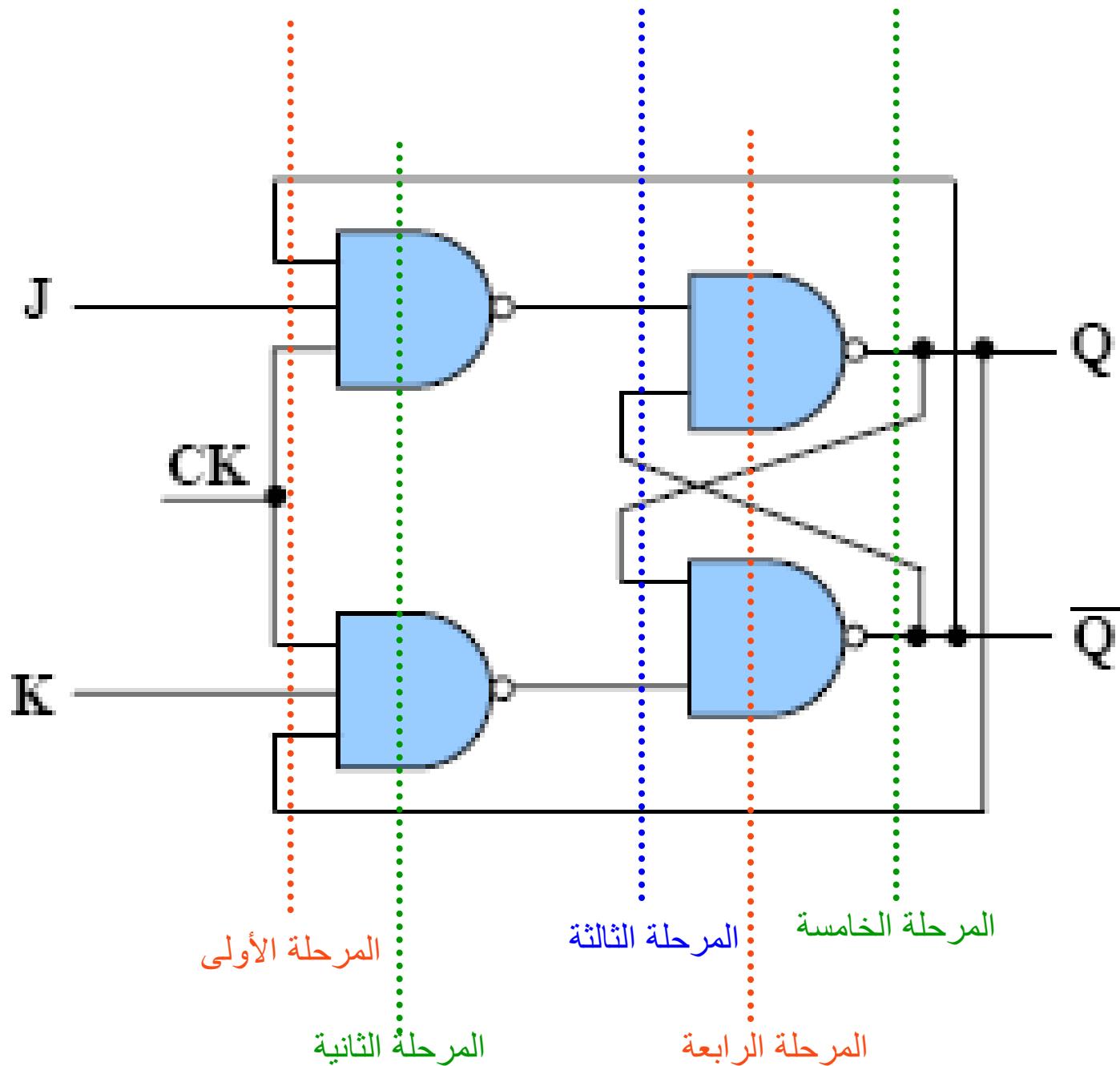
Block Diagram for J – K Flip Flop



Logic diagram for J – K Flip Flop



الدخلات			الخرج	وضع التشفير (Mode of Operation)
J	K	CK	Q	
.	.	↓	Q.	وضع الإمساك (عدم التغيير) No Change
.	1	↓	.	الوضع غير الفعال (RESET)
1	.	↓	1	الوضع الفعال (SET)
1	1	↓	\bar{Q}_0	وضع التبديل Toggle





1

J

CK

K

0

المرحلة الأولى

المرحلة الثانية

المرحلة الثالثة

المرحلة الرابعة

Start

0

1

1

Start

1

1

1

1

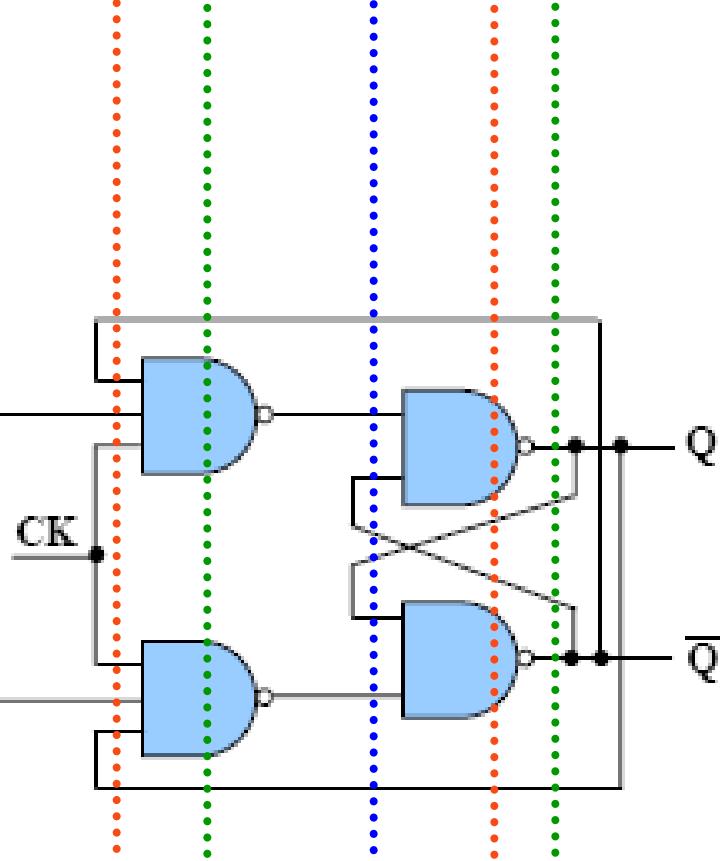
0

0

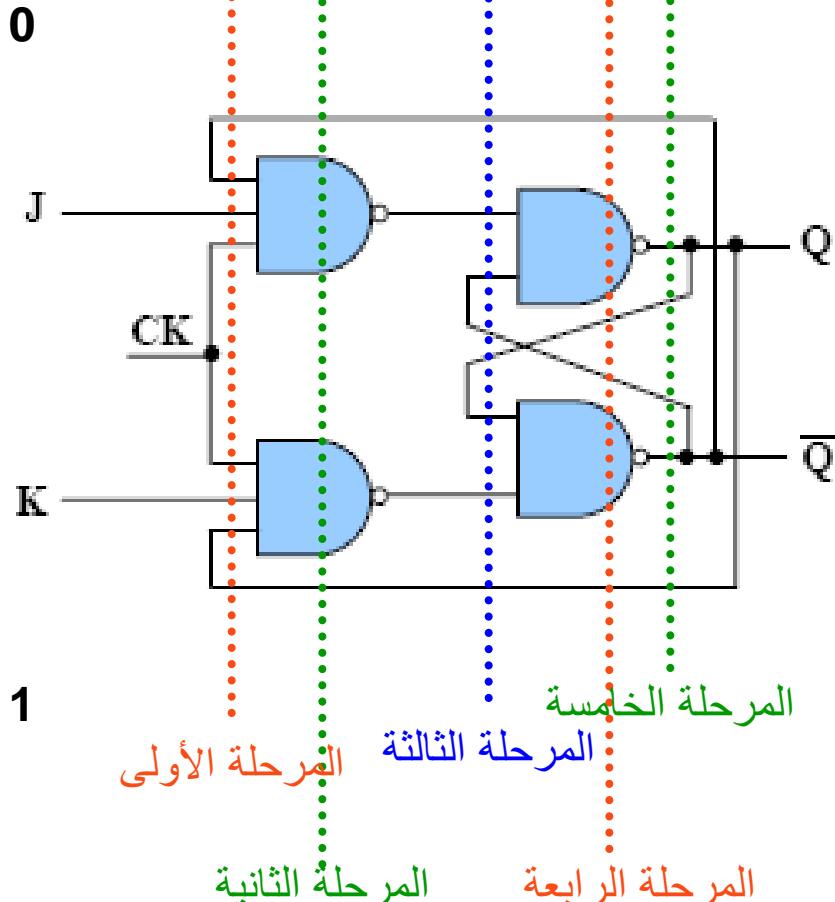
0

0

0

When $J = 1 \& K = 0$ Condition = Set ($Q = 1 \& Q' = 0$)

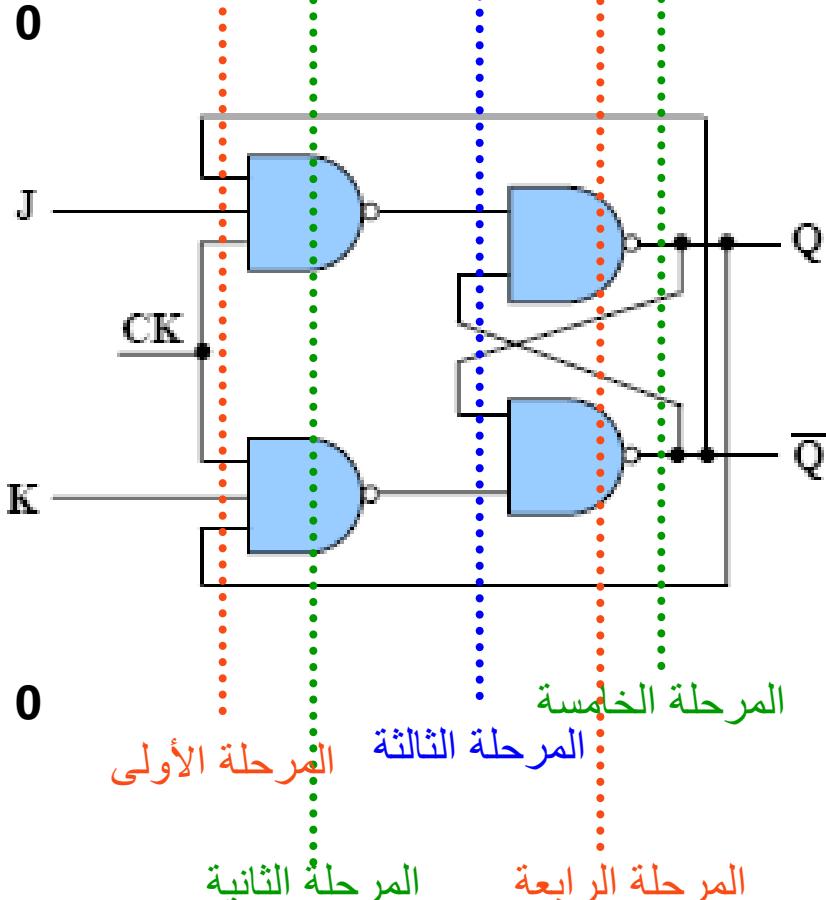
المرحلة الخامسة



When $J = 0 \& K = 1$

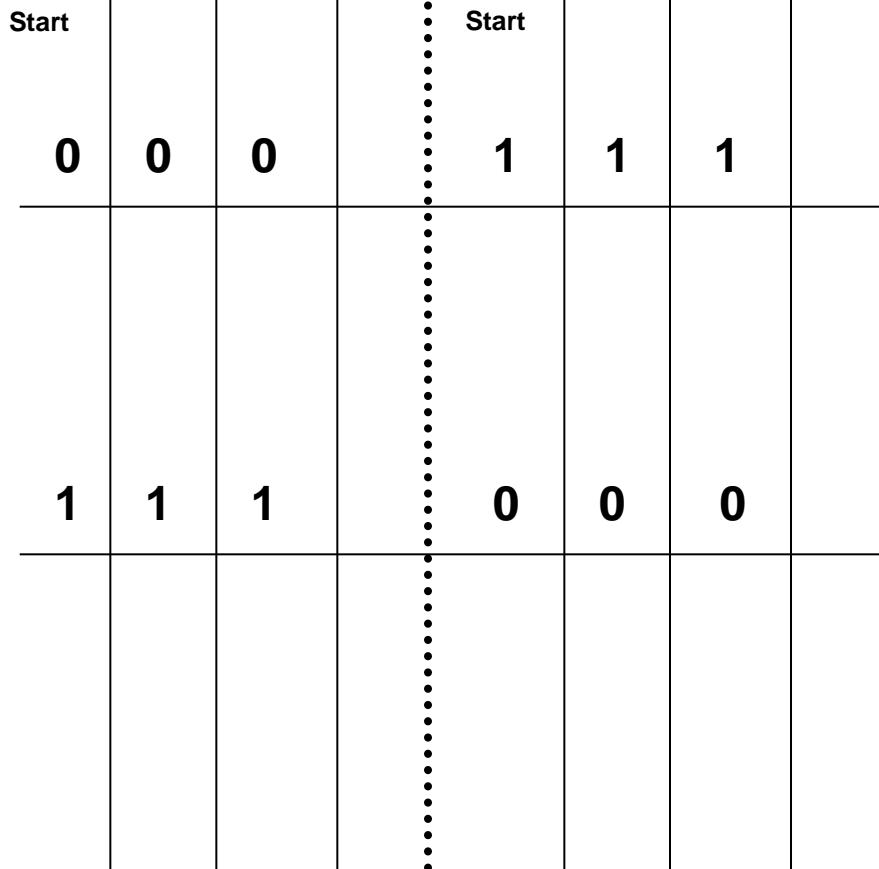
Condition = Reset ($Q = 0 \& Q' = 1$)

Start	0	0	0	Start	1	0	0
	1	1	1		0	1	1



When $J = 0$ & $K = 0$

Condition = No change





1

J

CK

K

1

المرحلة الأولى

المرحلة الثانية

المرحلة الثالثة

المرحلة الرابعة

المرحلة الخامسة

Q

 \bar{Q}

Start

0

1

1

Start

1

0

0

1

0

0

0

1

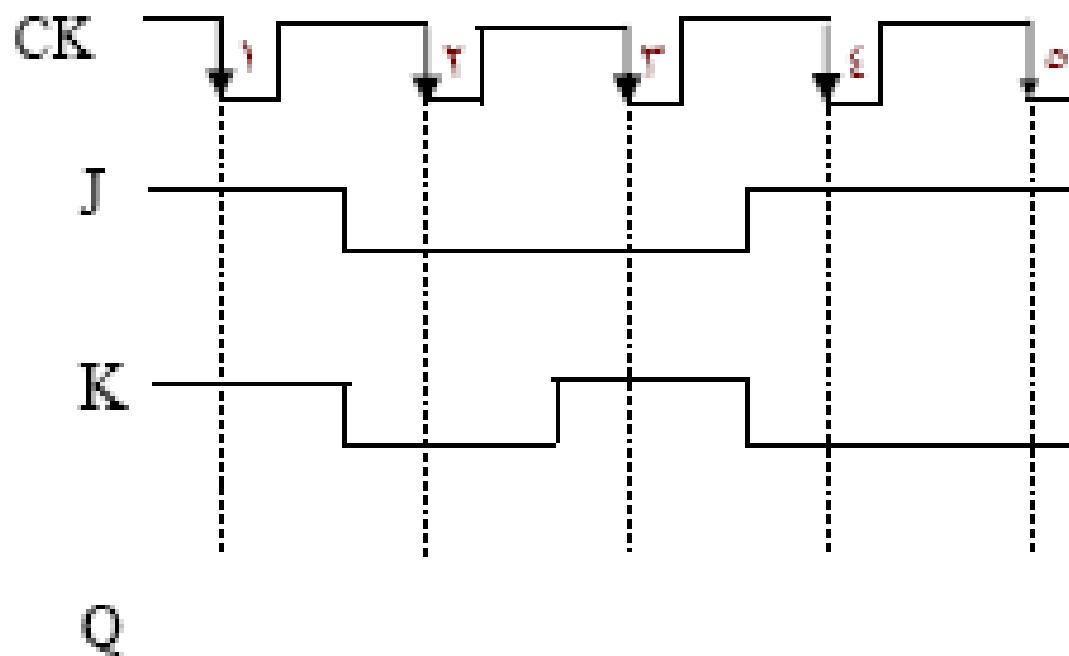
1

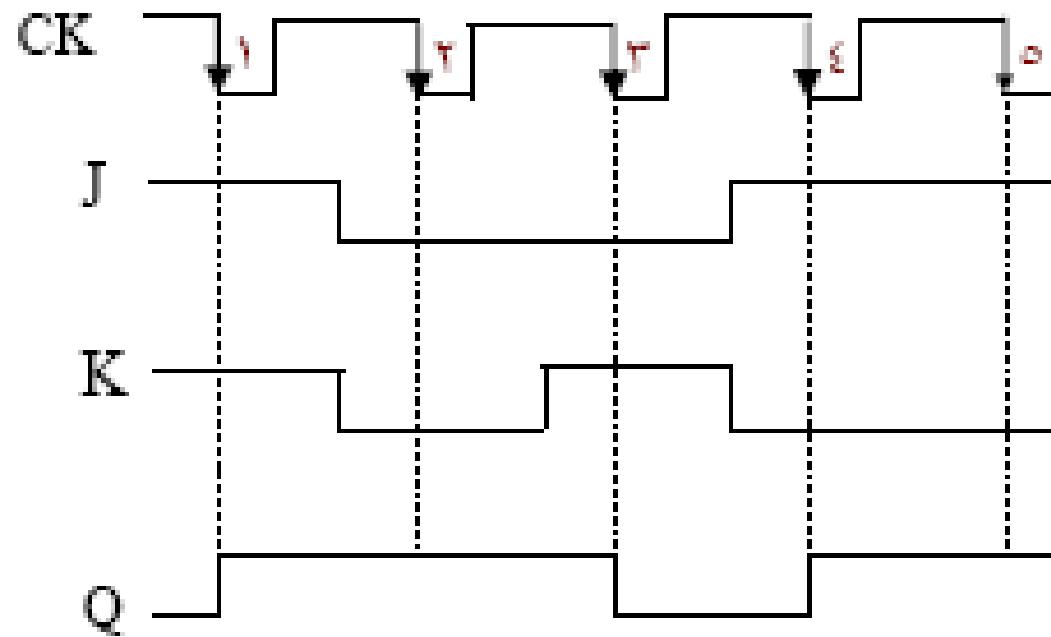
When $J = 1 \& K = 1$

Condition = Toggle



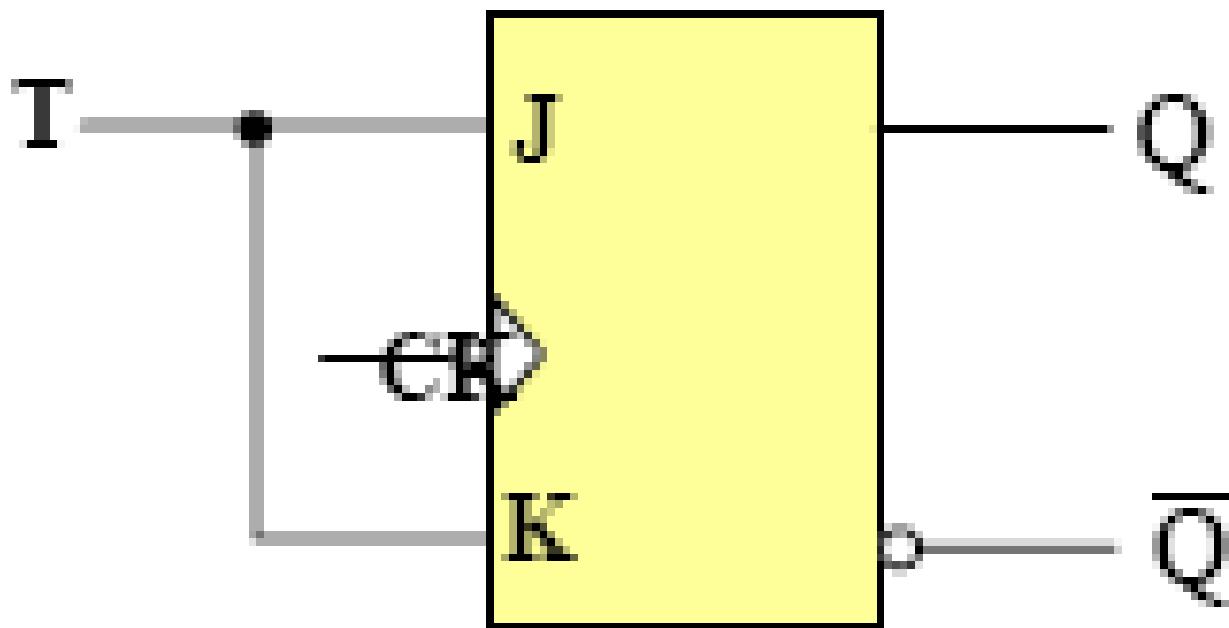
The following is timing diagram for J K flip flopDraw the pulse shape for the output Q (initial Q = 0) clock pulse from (1) to (0) ↓







T – Type flip flop



From J-K flip flop connect J with K



الدخلات		الخرج	وضع التشغيل (Mode of Operation)
T	CK	Q	
.	↓	Q.	وضع الامساك (عدم التغير) No Change
١	↓	\overline{Q}_0	وضع التبديل Toggle

