LOGICAL EXPRESSION SIMPLICATION

Two basic methods

- ✧ Algebraic manipulation
  - Use Boolean laws to simplify the expression
  - Difficult to use
  - Don’t know if you have the simplified form
- ✧ Karnaugh map (K-map) method
  - Graphical method Easy to use
  - Can be used to simplify logical expressions with a few variables

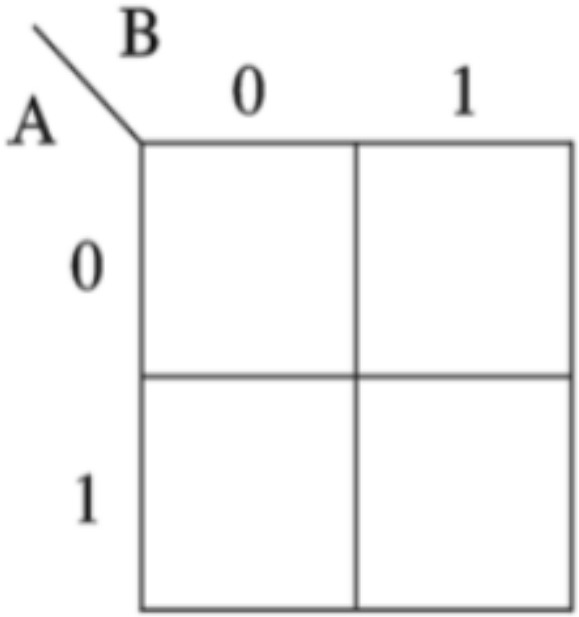
K-MAP

- The K-map method of solving the logical expressions is referred to as the graphical technique of simplifying Boolean expressions.
- K-maps are also referred to as 2D truth tables.
- K-map can be used to simplify expressions having up to 5 variables
- The number of cells in the K-map is determined by the number of input variables(n), given by 2^n
- We can rearrange the Minterms of the truth table into Karnaugh map
- To simplify a function with two inputs, we require a K-map with 4 (=2^2) cells. A four-input function requires 16 (=2^4) cells in the K-map.

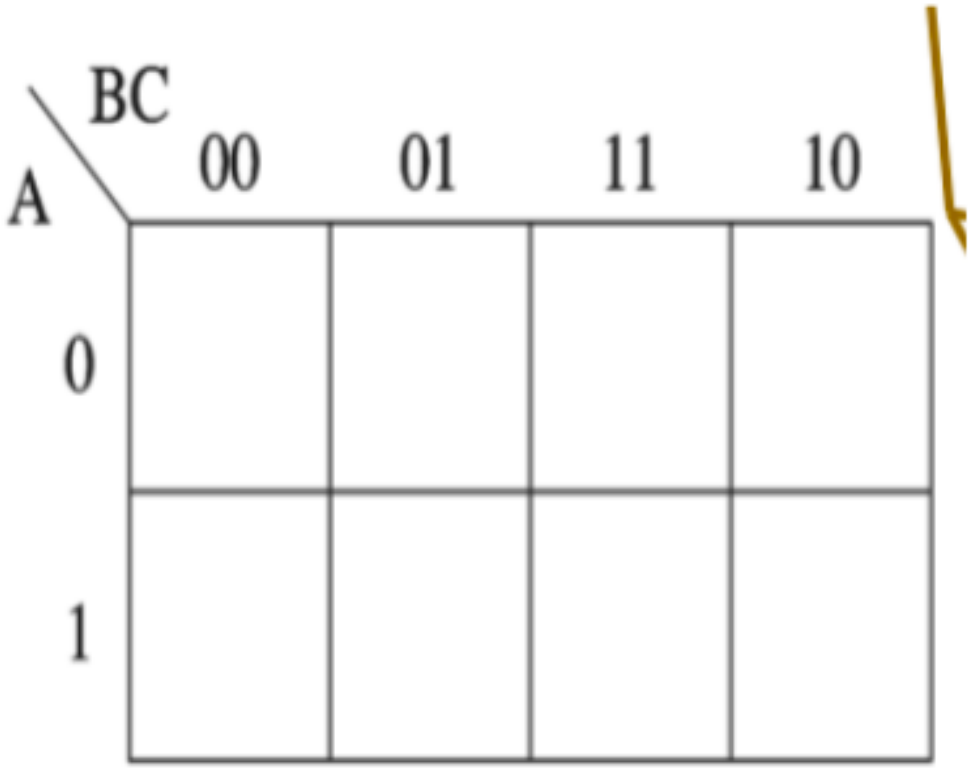
K-MAP ARRANGEMENT OF CELL

- The arrangement of cells within the K-map follows an encoding technique known as Gray code
- The rows and the columns in the K-map uses Gray code-labeling.
- In Gray code labeling the adjacent code values differ only by a single bit.
- For example for a given code word 01, the next or previous code word could either be 11 or 00, but not 10

K-MAP REPRESENTATION

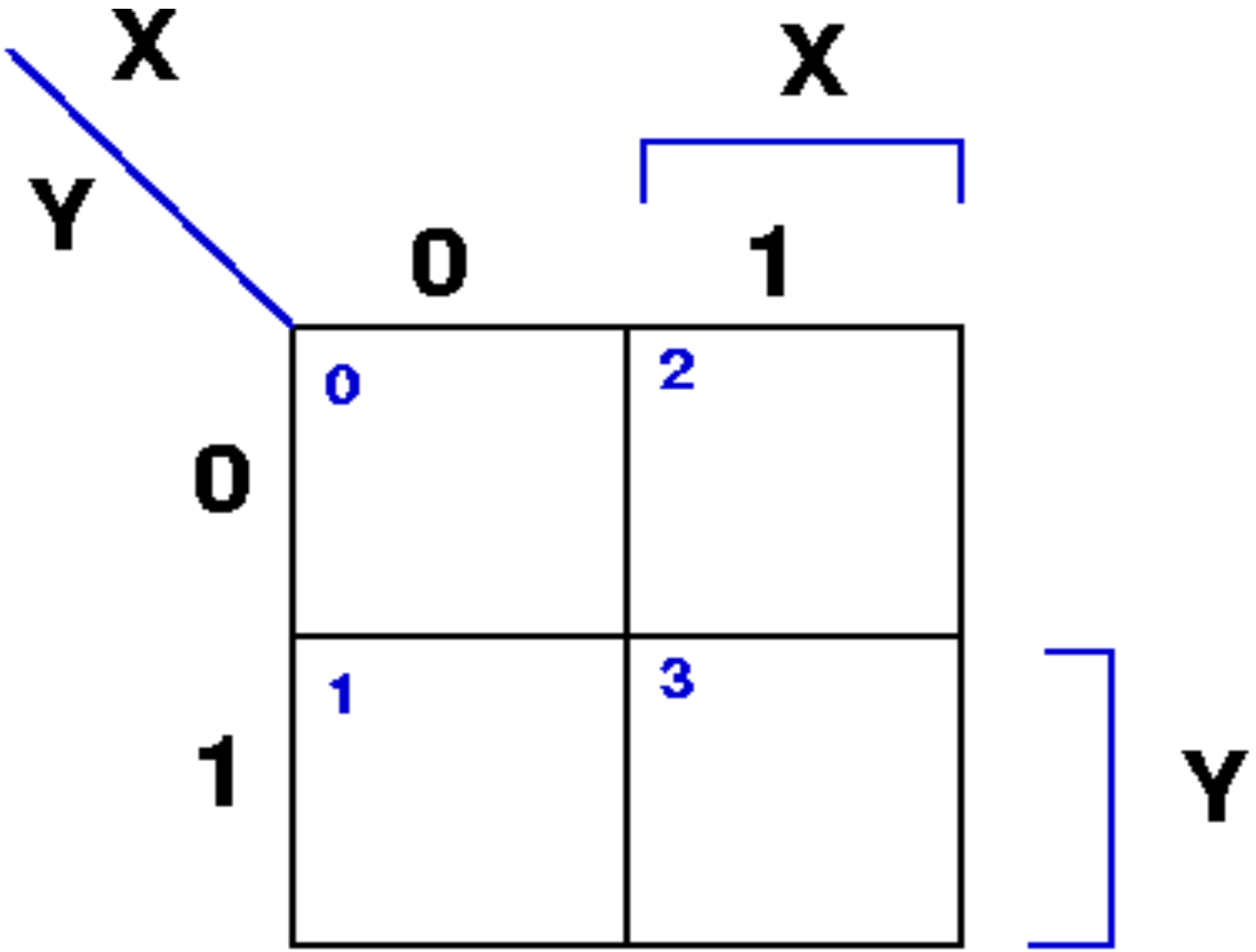


(a) Two-variable K-map



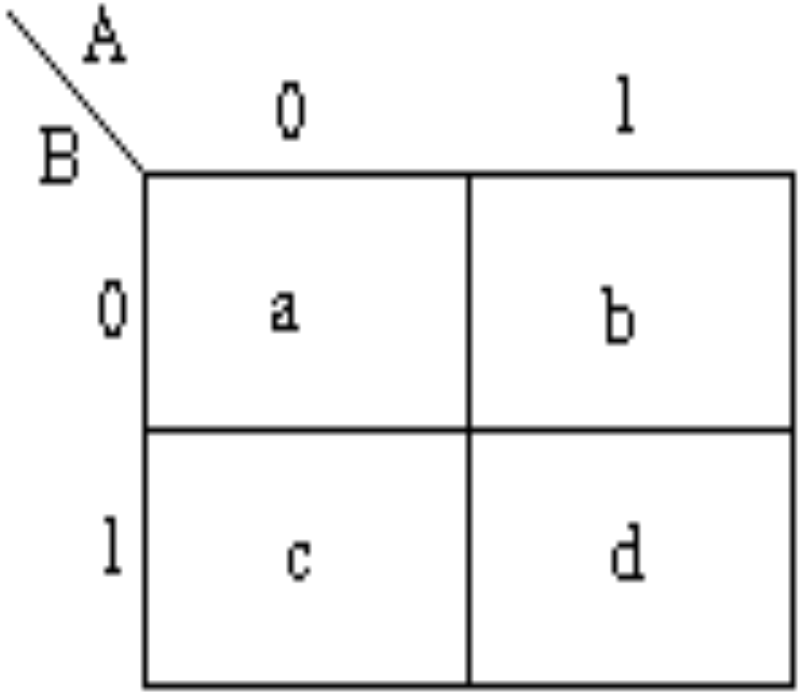
(b) Three-variable K-map

2-input variable k-map

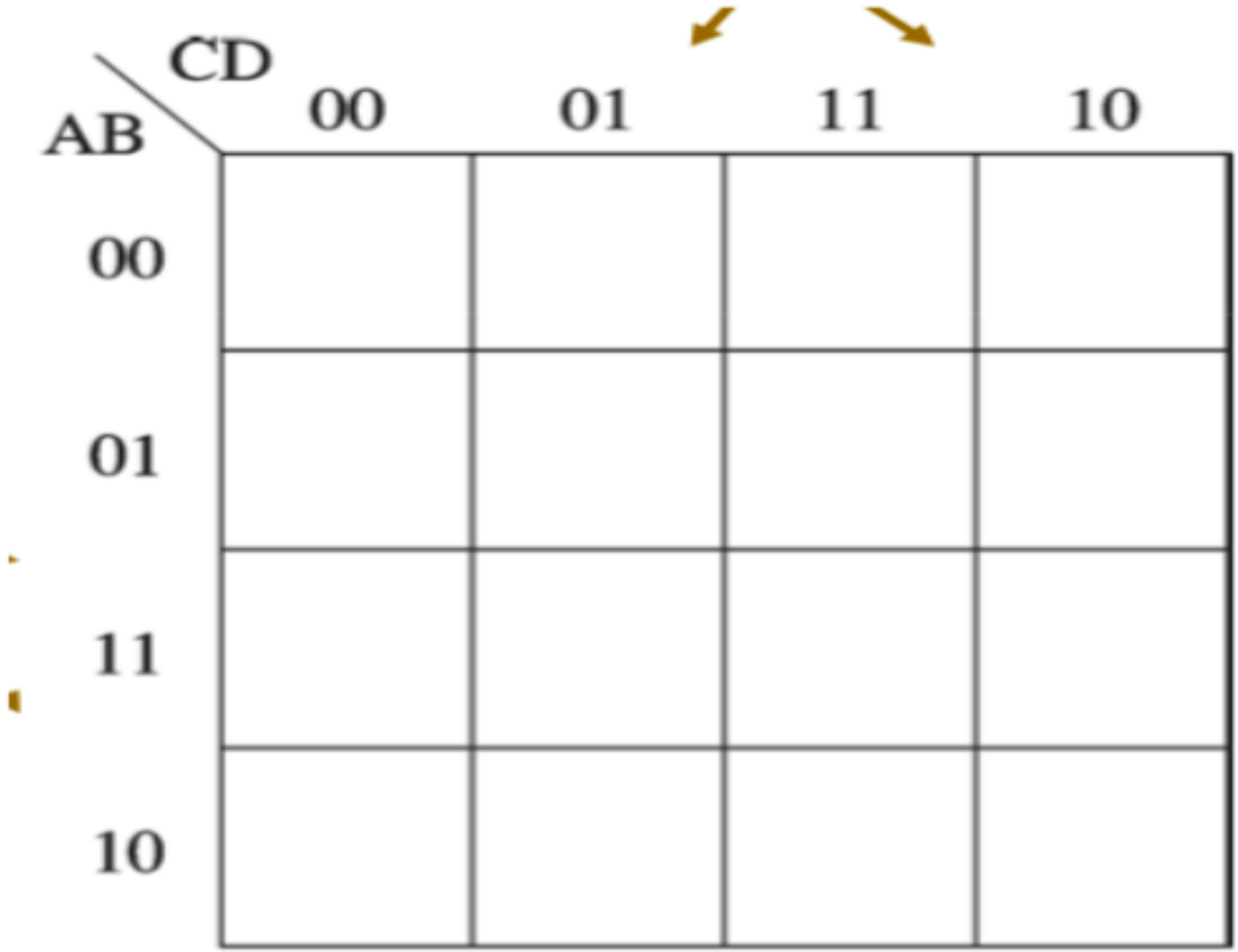


A	B	F
0	0	a
0	1	b
1	0	c
1	1	d

Truth Table.



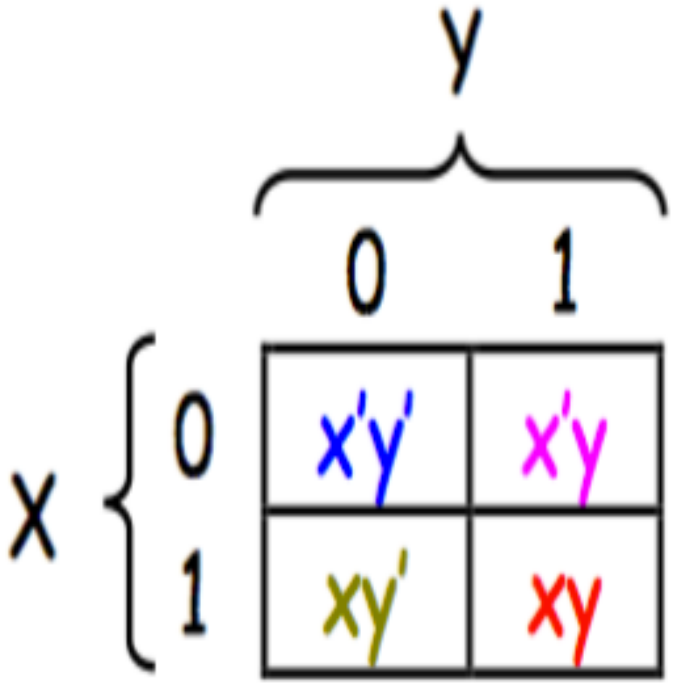
F.



(c) Four-variable K-map

RE-ARRANGING MINTERMS IN THE TRUTH TABLE TO FORM K-MAP

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



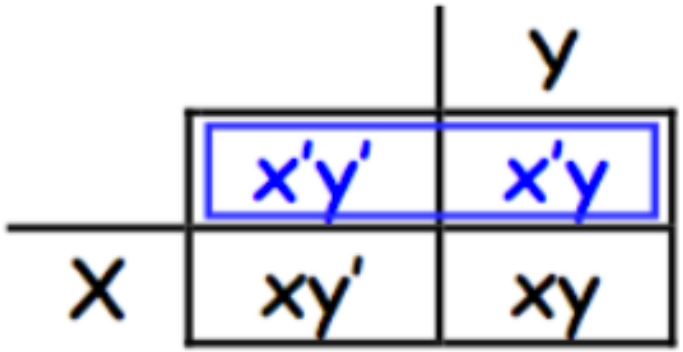
Now we can easily see which minterms contain common literals

– Minterms on the left and right sides contain y' and y respectively

– Minterms in the top and bottom rows contain x' and x respectively

Imagine a two-variable sum of minterms: x'y' + x'y

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



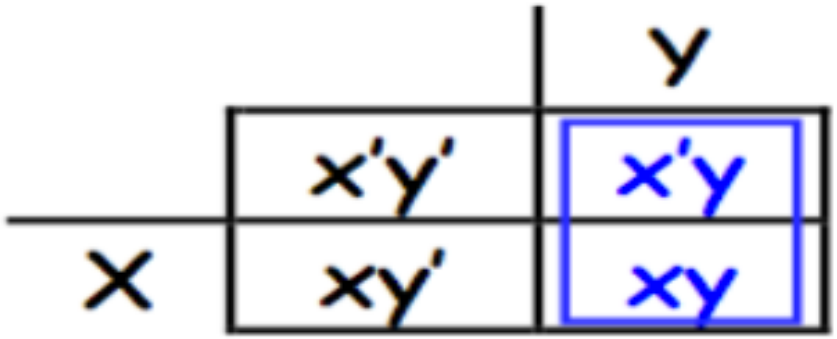
What happens if we simplify with Boolean algebra

$$x'y' + x'y = x'(y' + y) \quad [\text{ Distributive }]$$
$$= x' \bullet 1 \quad [\, y + y' = 1 \,]$$
$$= x' \quad [\, x \bullet 1 = x \,]$$

Another example expression is ***x'y + xy***

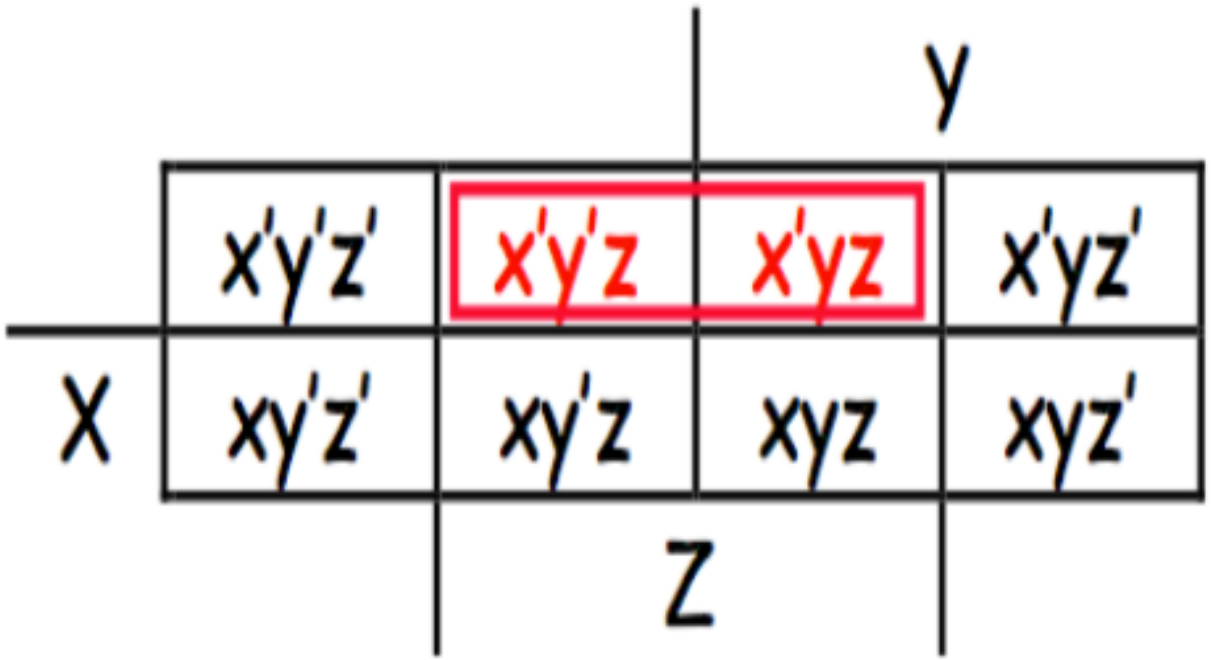
– Both minterms appear in the right side, where y is uncomplemented

– Thus, we can reduce ***x'y + xy*** to just ***y***



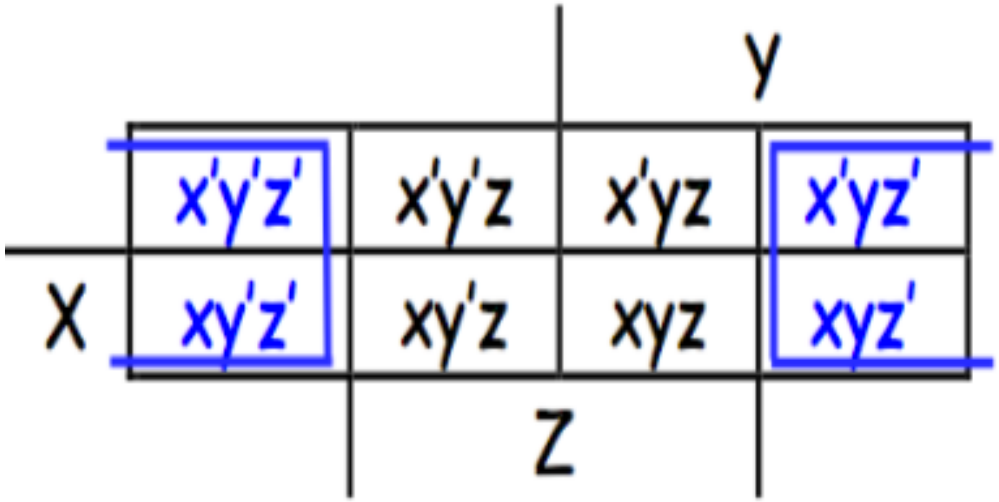
Simplify ***x'y' + x'y + xy*** using the K-map above

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



$$x'y'z + x'yz$$
$$= x'z(y' + y)$$
$$= x'z \bullet 1$$
$$= x'z$$

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



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

THE RULES OF K-MAP SIMPLIFICATION

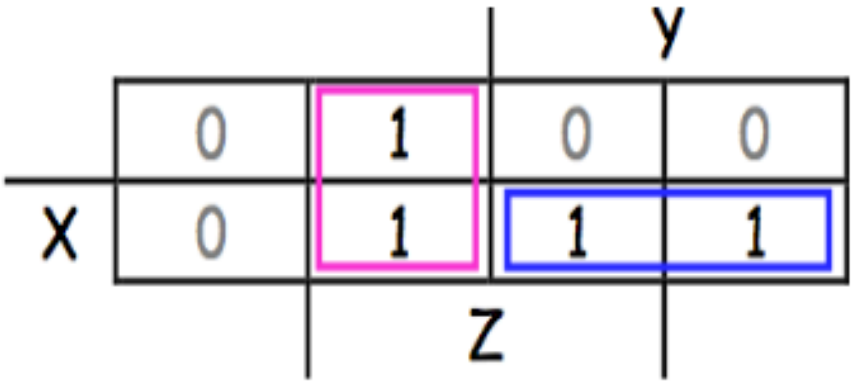
1. Groupings can contain only 1s; no 0s
2. Groups can be formed only at right angles; diagonal groups are not allowed
3. The number of 1s in a group must be a power of 2 – even if it contains a single 1
4. The groups must be made as large as possible
5. Groups can overlap and wrap around the sides of the K-map

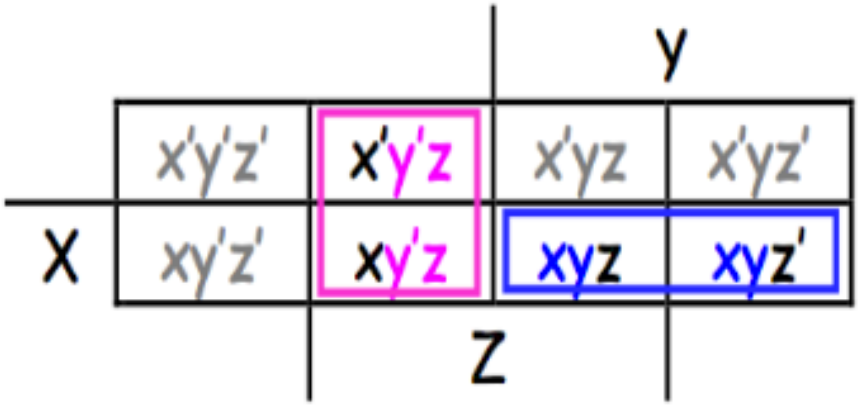
THE RULES OF K-MAP SIMPLIFICATION

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

READING THE MINIMAL SOP

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

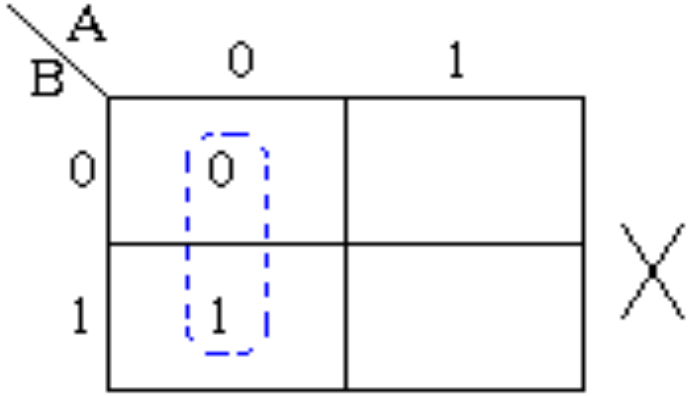


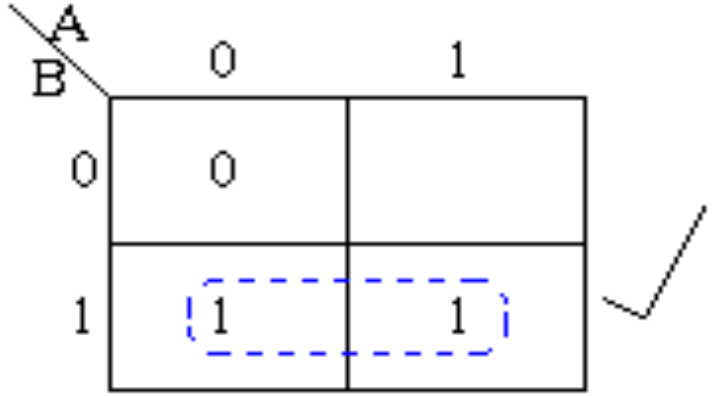


- For our example, we find that xy + y'z + xz = y'z + xy. (This is one of the additional algebraic laws from last time.)

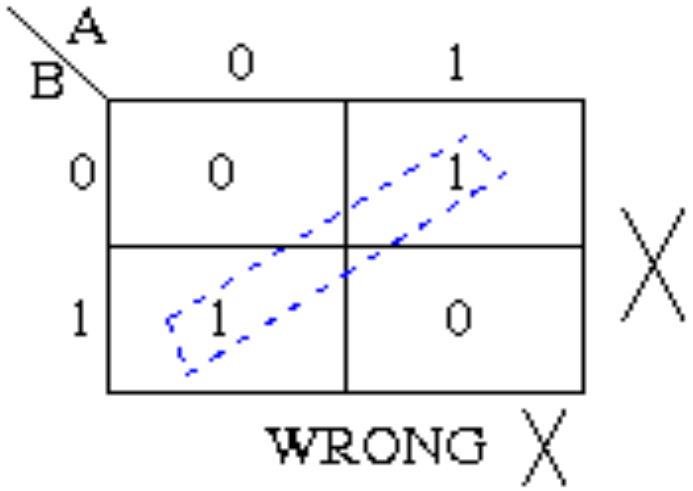
MORE ON GROUPING

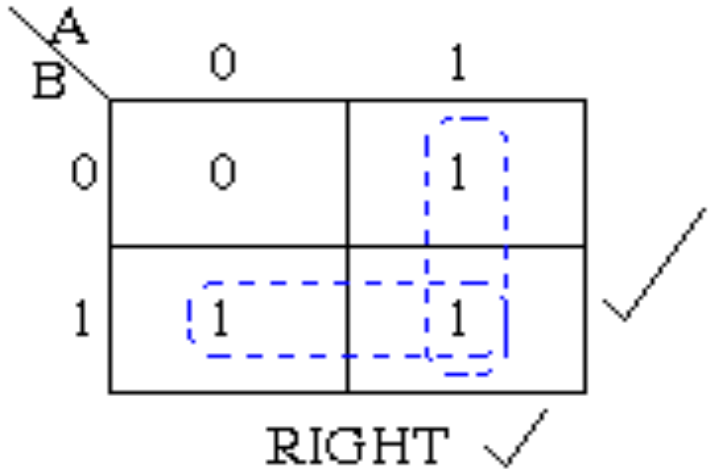
- **Groups may not include any cell containing a zero**





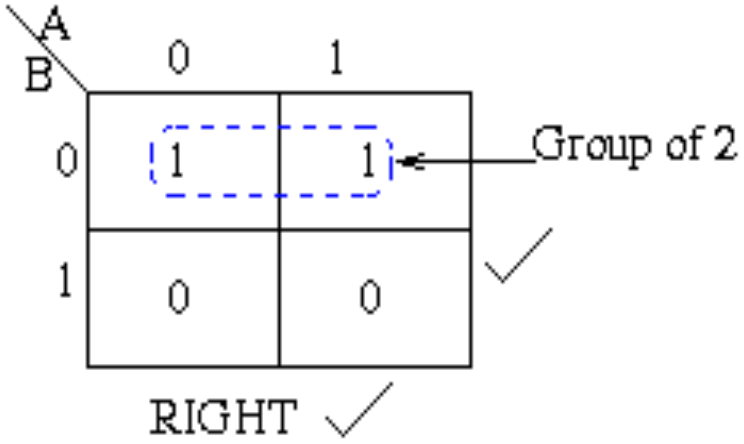
- **Groups may be horizontal or vertical, but not diagonal.**



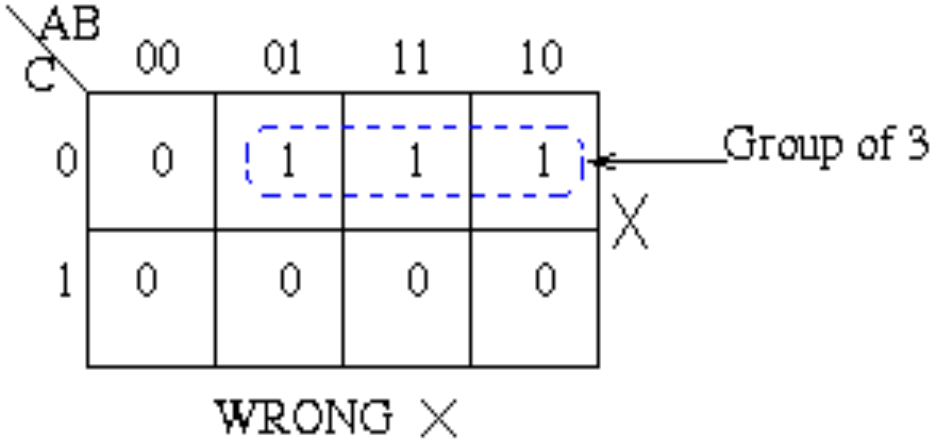


MORE ON GROUPING

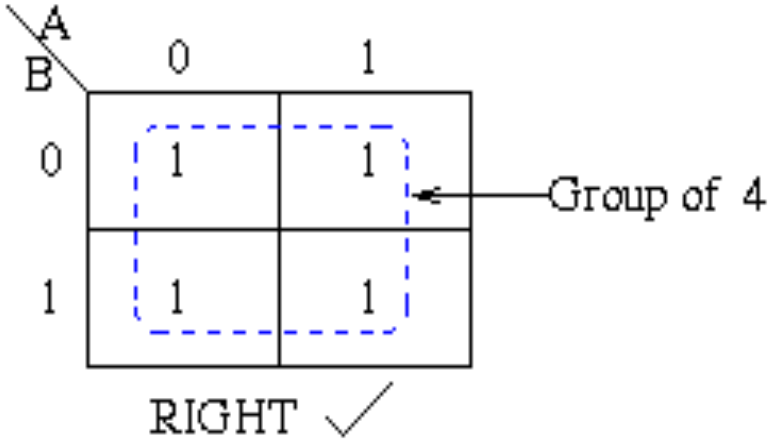
- **Groups must contain 1, 2, 4, 8, or in general 2^n cells.**
  That is if n = 1, a group will contain two 1's since $2^1 = 2$.
  If n = 2, a group will contain four 1's since $2^2 = 4$.



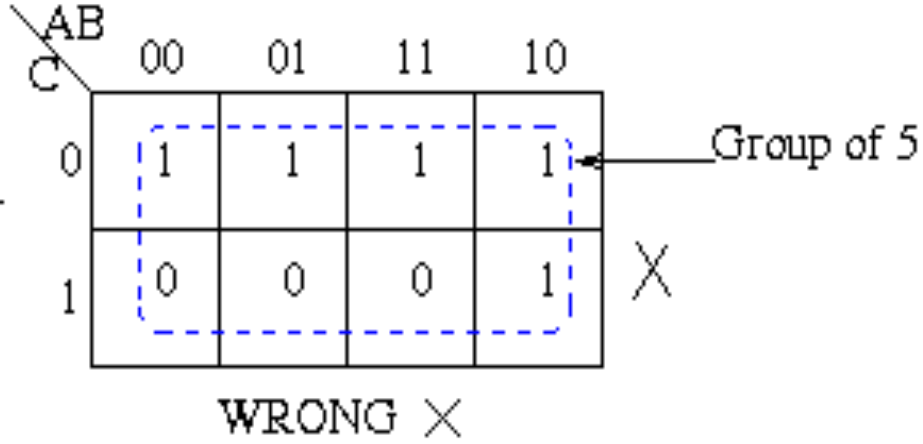
RIGHT ✓



WRONG ✗



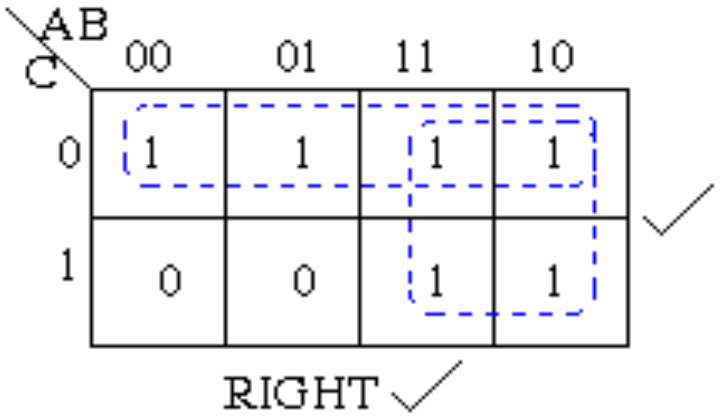
RIGHT ✓



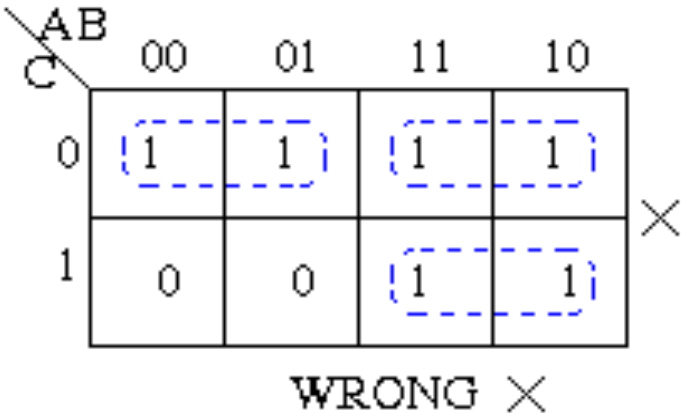
WRONG ✗

MORE ON GROUPING

- **Each group should be as large as possible.**



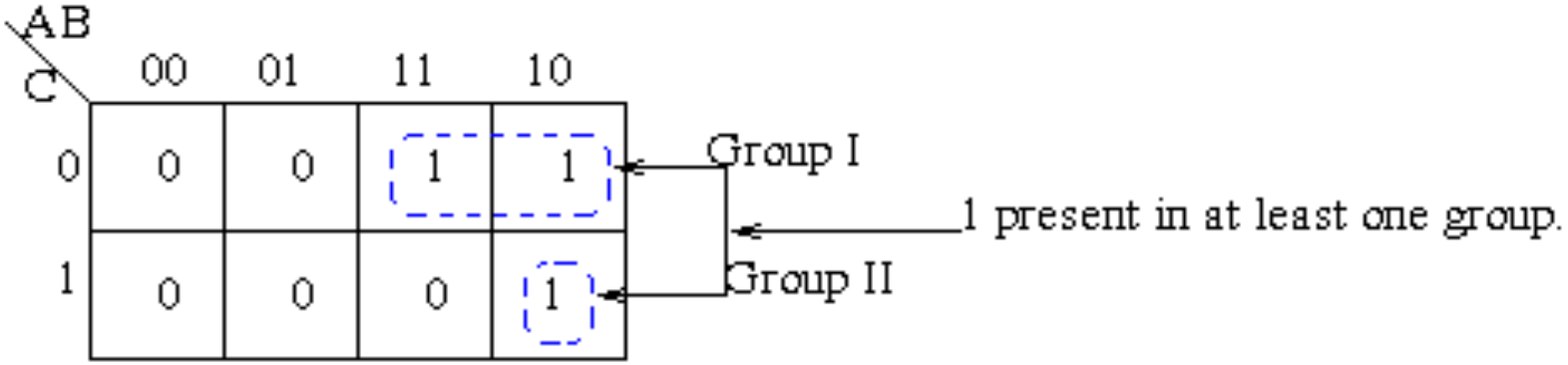
RIGHT ✓



WRONG ×

(Note that no Boolean laws broken, but not sufficiently minimal)

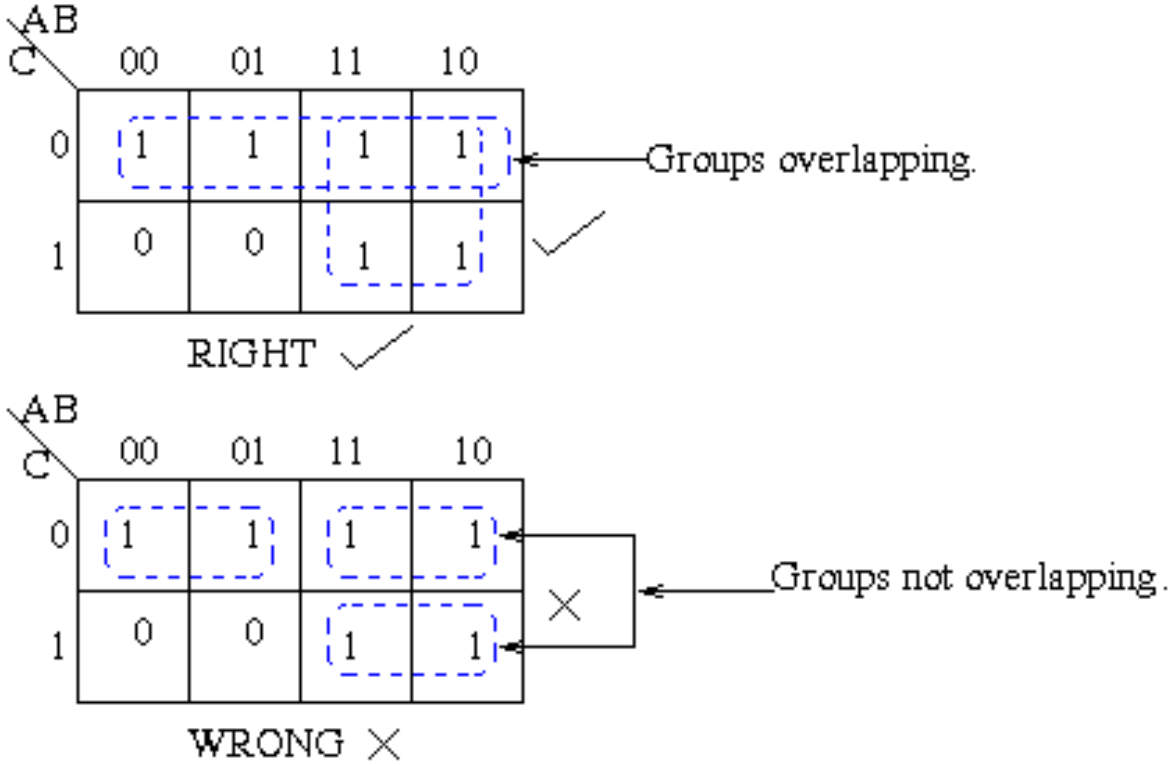
- **Each cell containing a *one* must be in at least one group.**



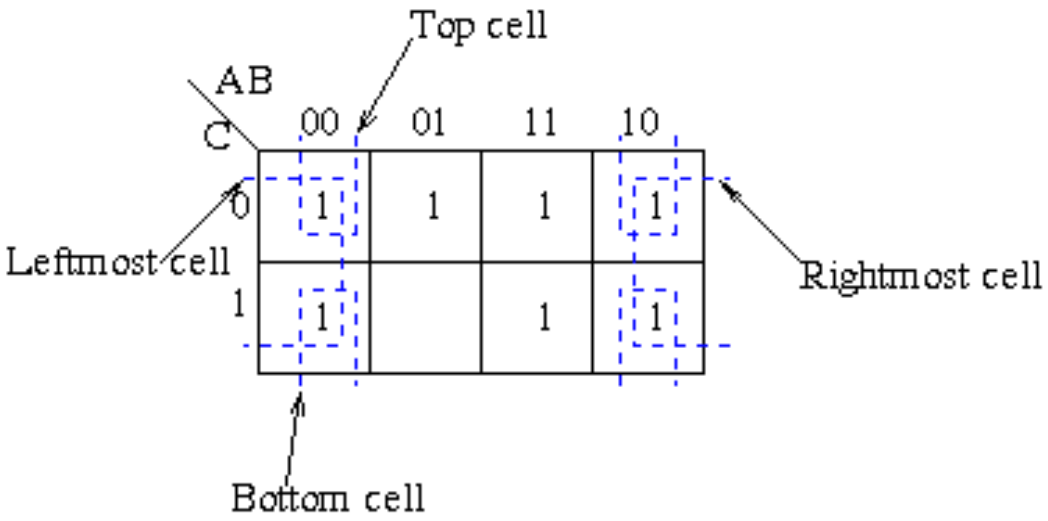
STILL MORE GROUPING

DWUMFOUR ABDULLAI ABDUL-AZIZ

- **Groups may overlap.**



- **Groups may wrap around the table. The leftmost cell in a row may be grouped with the rightmost cell**



LETS SUMMARIZE

- No zeros allowed.
- Only ones (1's) are grouped.
- No diagonals.
- Only power of 2 number of cells in each group.
- Groups should be as large as possible.
- Every one must be in at least one group.
- Overlapping is allowed.
- Wrap around is allowed.
- Fewest number of groups possible.

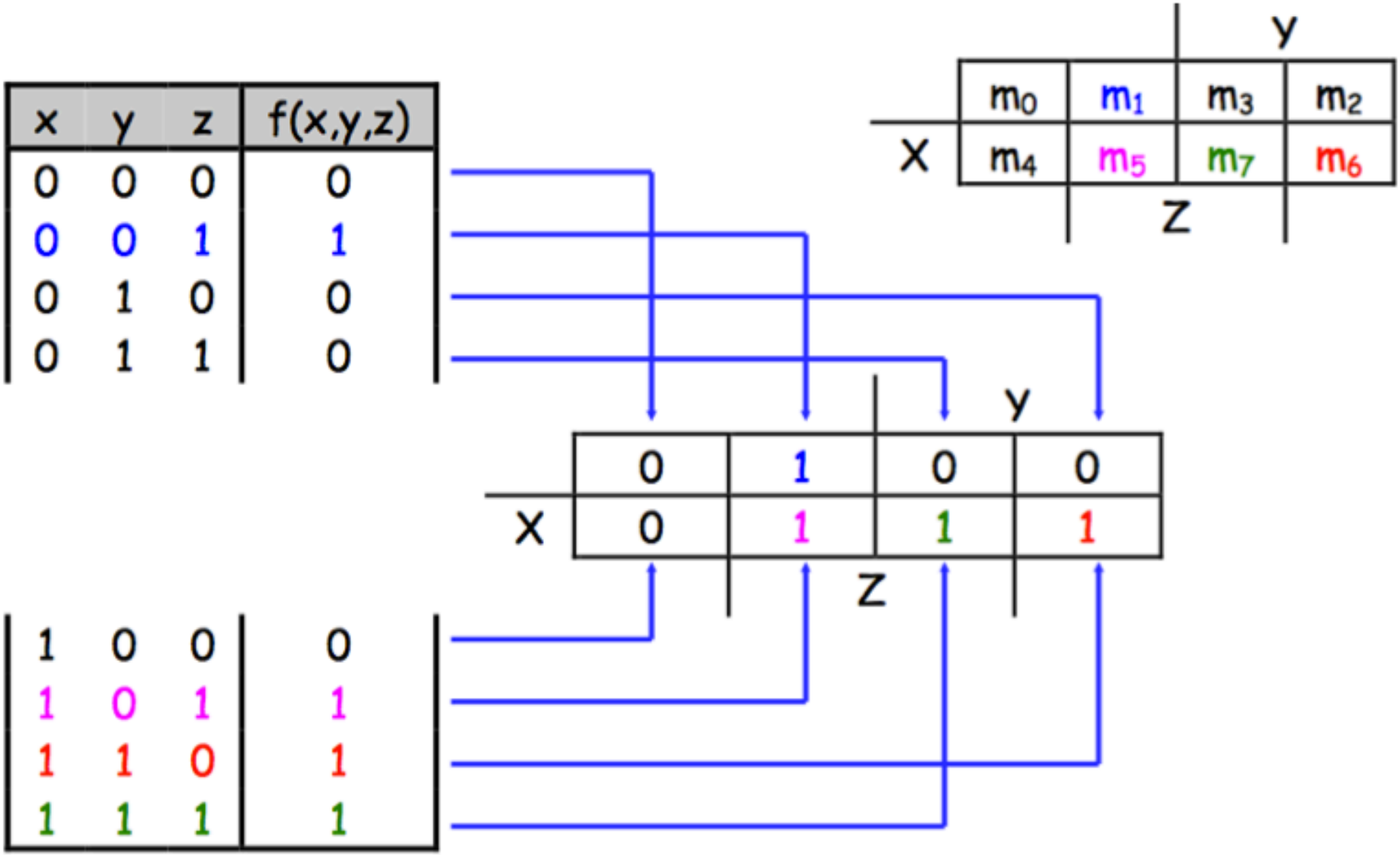
- simplify f(x,y,z) = ***xy + y'z + xz*** using the k-map
- First, you should convert the expression into a sum of minterms form, if it's not already
  - The easiest way to do this is to make a truth table for the function, and then read off the minterms
  - You can either write out the literals or use the Minterm shorthand

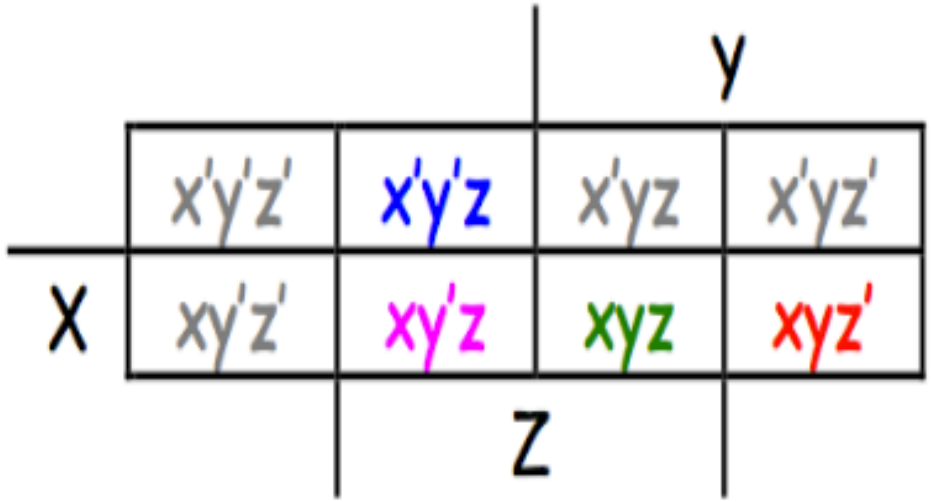
- Here is the truth table and sum of minterms for our example:

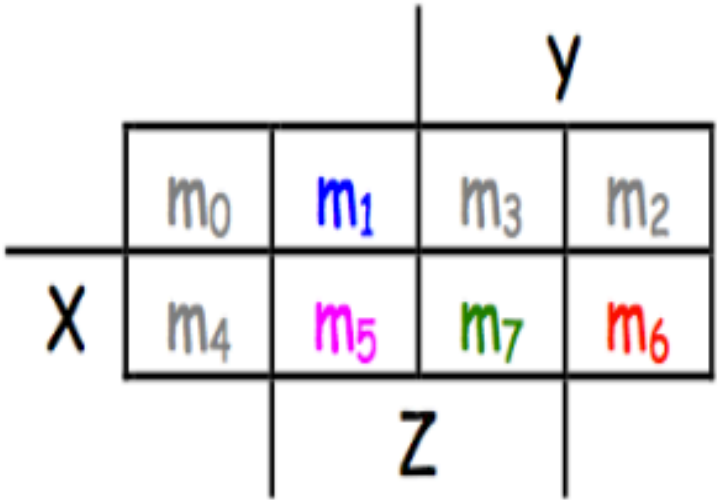
x	y	z	f(x,y,z)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

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

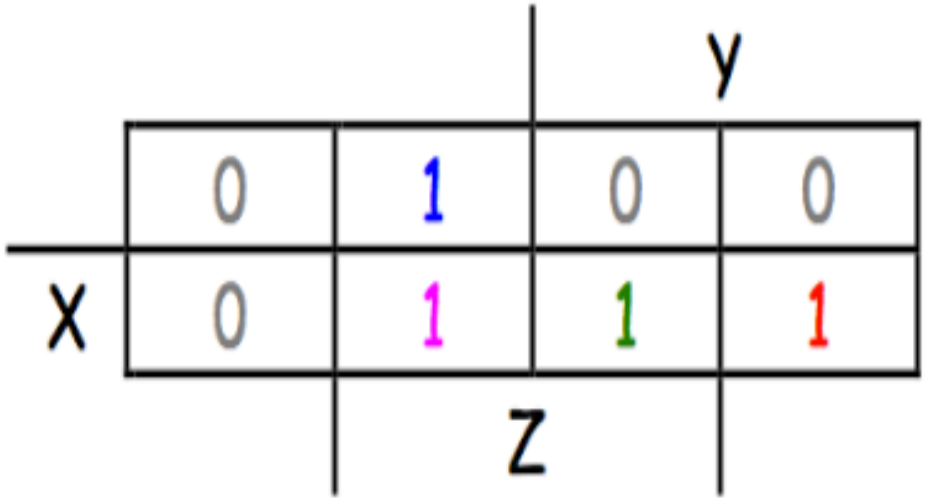
K-MAP FROM TRUTH TABLE







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



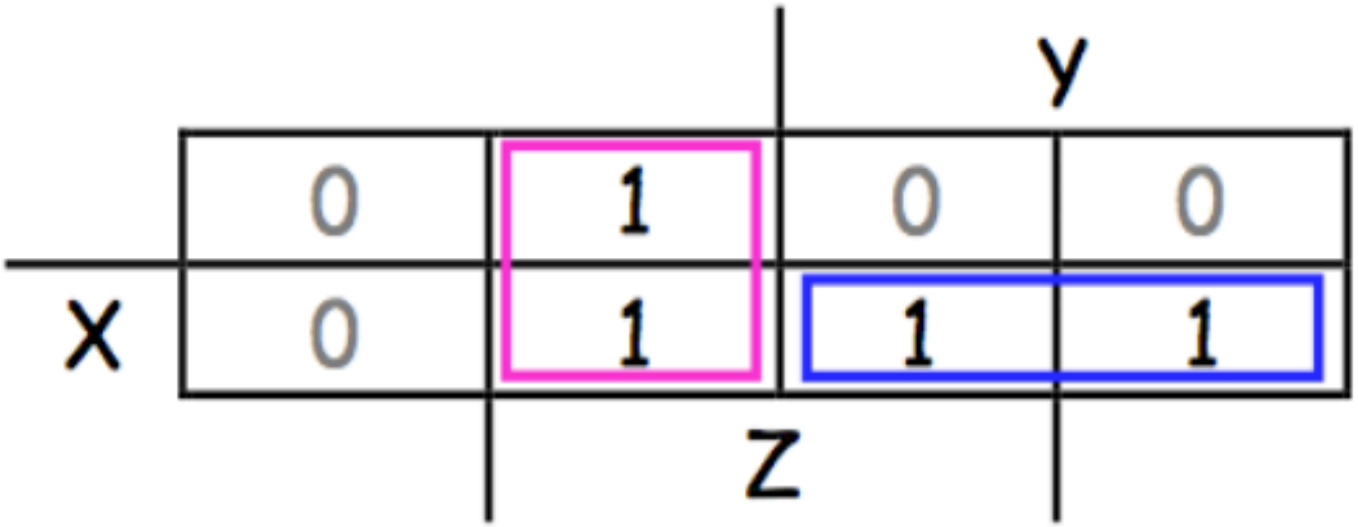
GROUPING THE MINTERMS

The most difficult step is grouping together all the 1s in the K-map

– Make rectangles around groups of one, two, four or eight 1s

– All of the 1s in the map should be included in at least one rectangle

– Do not include any of the 0s



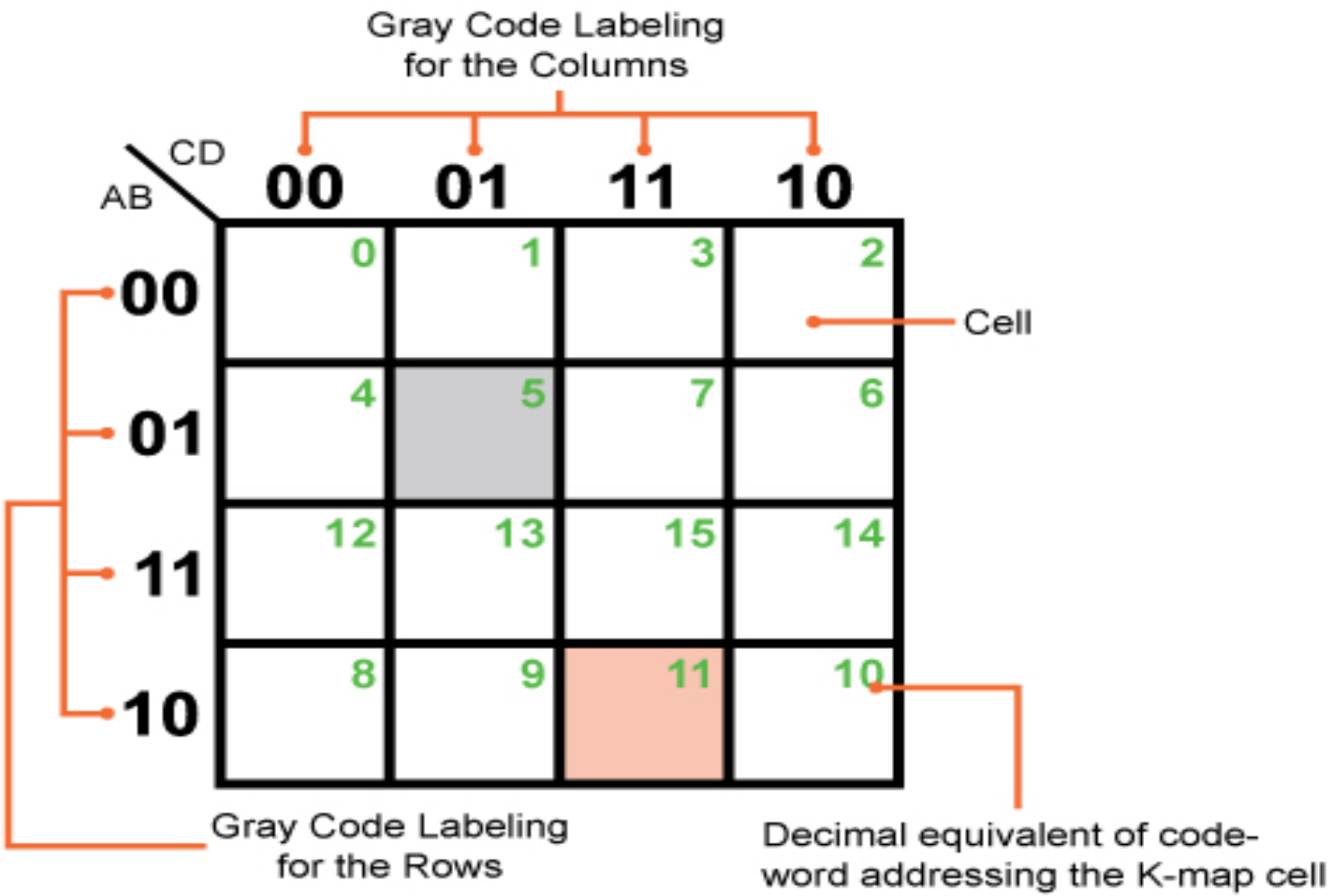
Simplify the expression using the k-map

Kmap - Three Variable (SOP Form): Example 1

- Consider the function:

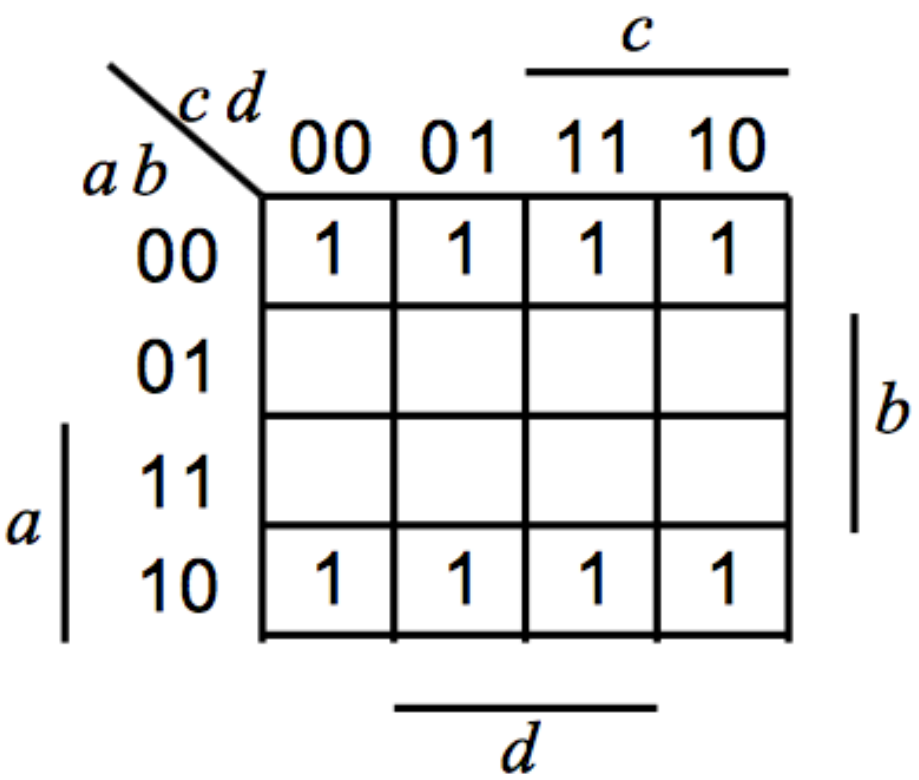
$$F(X,Y,Z) = \bar{X}\bar{Y}Z + \bar{X}YZ + X\bar{Y}Z + XYZ$$

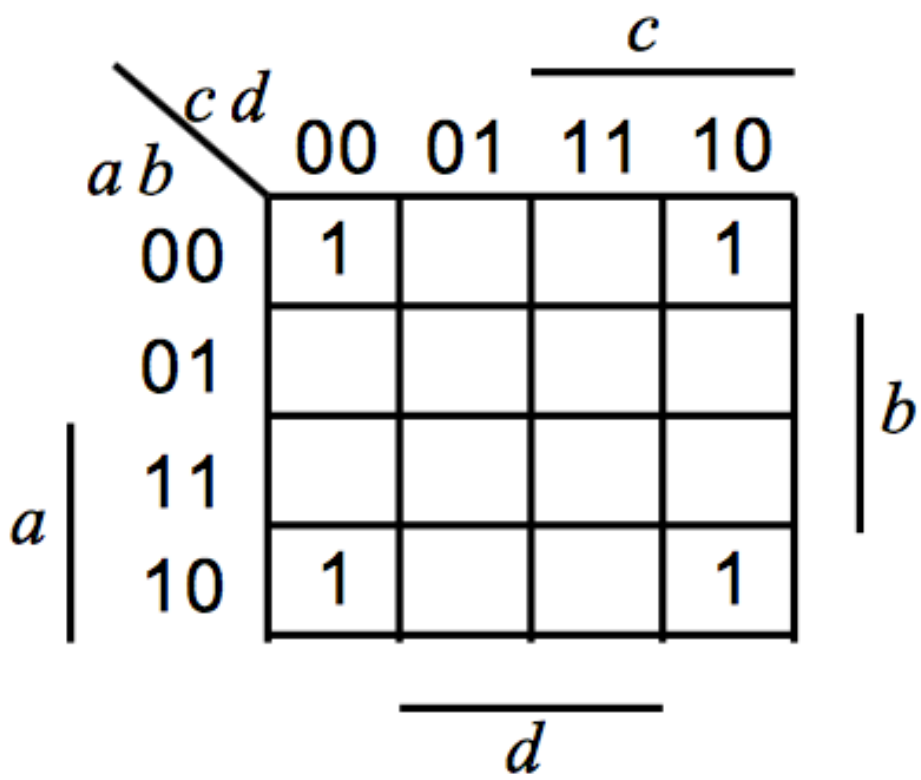
4 –VARIABLE K-MAP



K-maps – 4 variables

- For example, plot

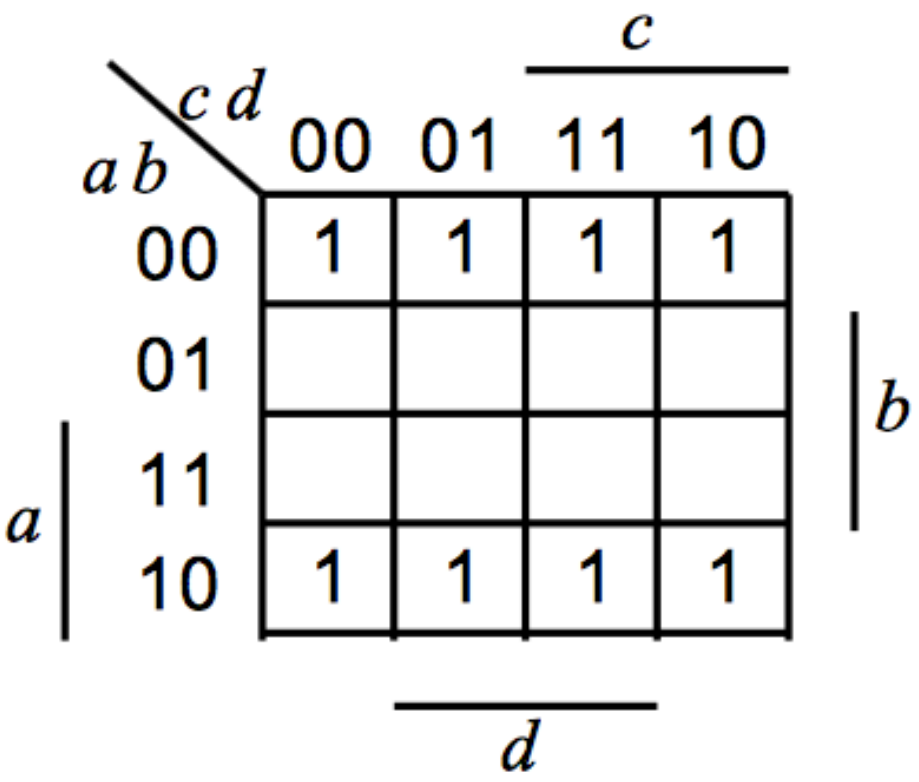




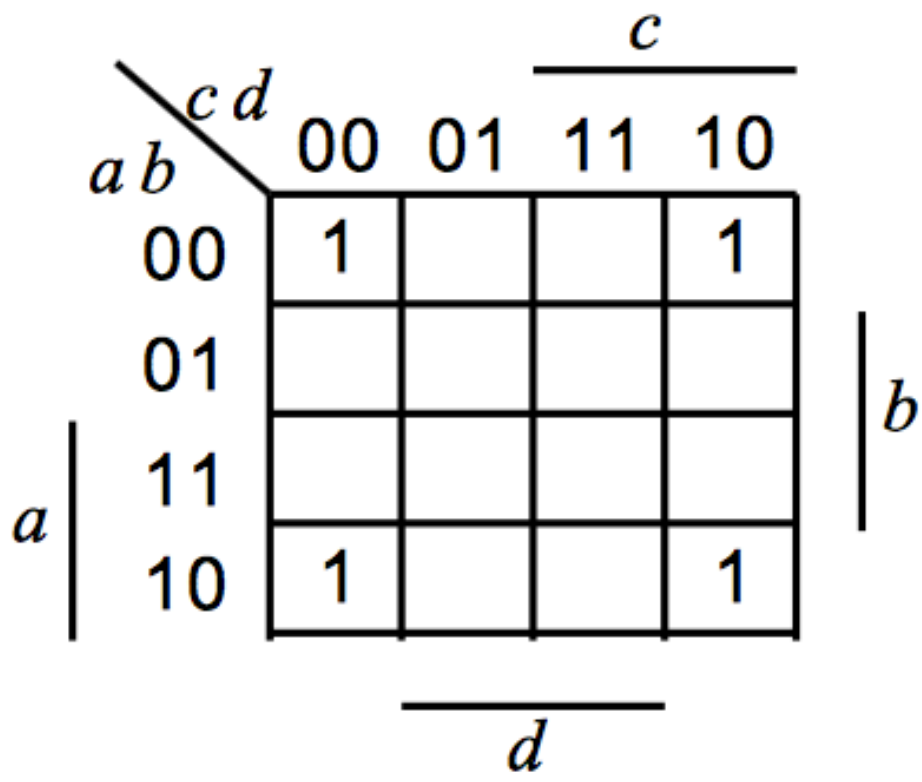
K-maps – 4 variables

- For example, plot

$$f = \overline{b}$$



$$f = \overline{b}.\overline{d}$$



ASSIGNMENT

Q1. Explain the following concepts in relation to K-map.

For each concept, give at least two(2) examples that make use of it.

-Don't care condition

-Implicants

-Prime Implicants

-Essential prime Implicants

Submission date: Thursday, October 25, 2018

TYPES OF DIGITAL CIRCUIT DESIGN

DIGITAL CIRCUITS

As discussed earlier, logic gates are the fundamental building blocks of digital circuits.

Digital circuits are used in the design of digital system.

There are two types of digital circuit design:

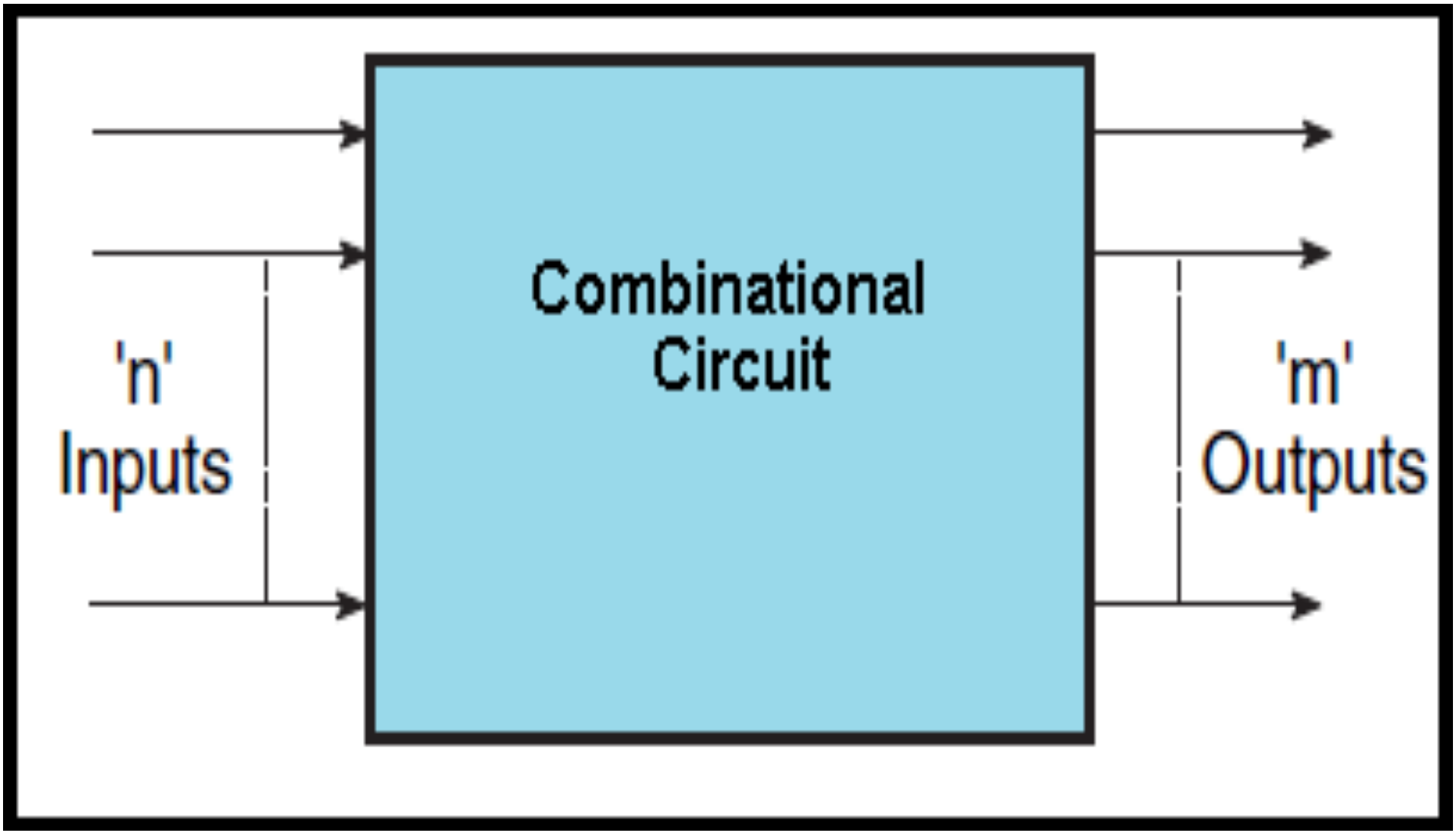
- combinational circuit
- Sequential circuit

COMBINATIONAL LOGIC

- Combinational circuit combines different gates to produce digital circuit.
- A combinational circuit interconnects set of gates whose output at any time is a function only of the input at that time.
- The combinational circuit do not use any memory. The previous state of input does not have any effect on the present state of the circuit.
- combinational circuit could be use to design Adders, encoders, decoders, multiplexers and demultiplexers.

- Using combinations of logic gates, complex operations can be performed.
- In theory, there is no limit to the number of gates that can be arrayed together in a circuit.
- In practice, there is a limit to the number of gates that can be packed into a given physical space.
- Arrays of logic gates are found in digital integrated circuits (ICs).
- combinational circuit can have an *n* number of inputs and *m* number of outputs.

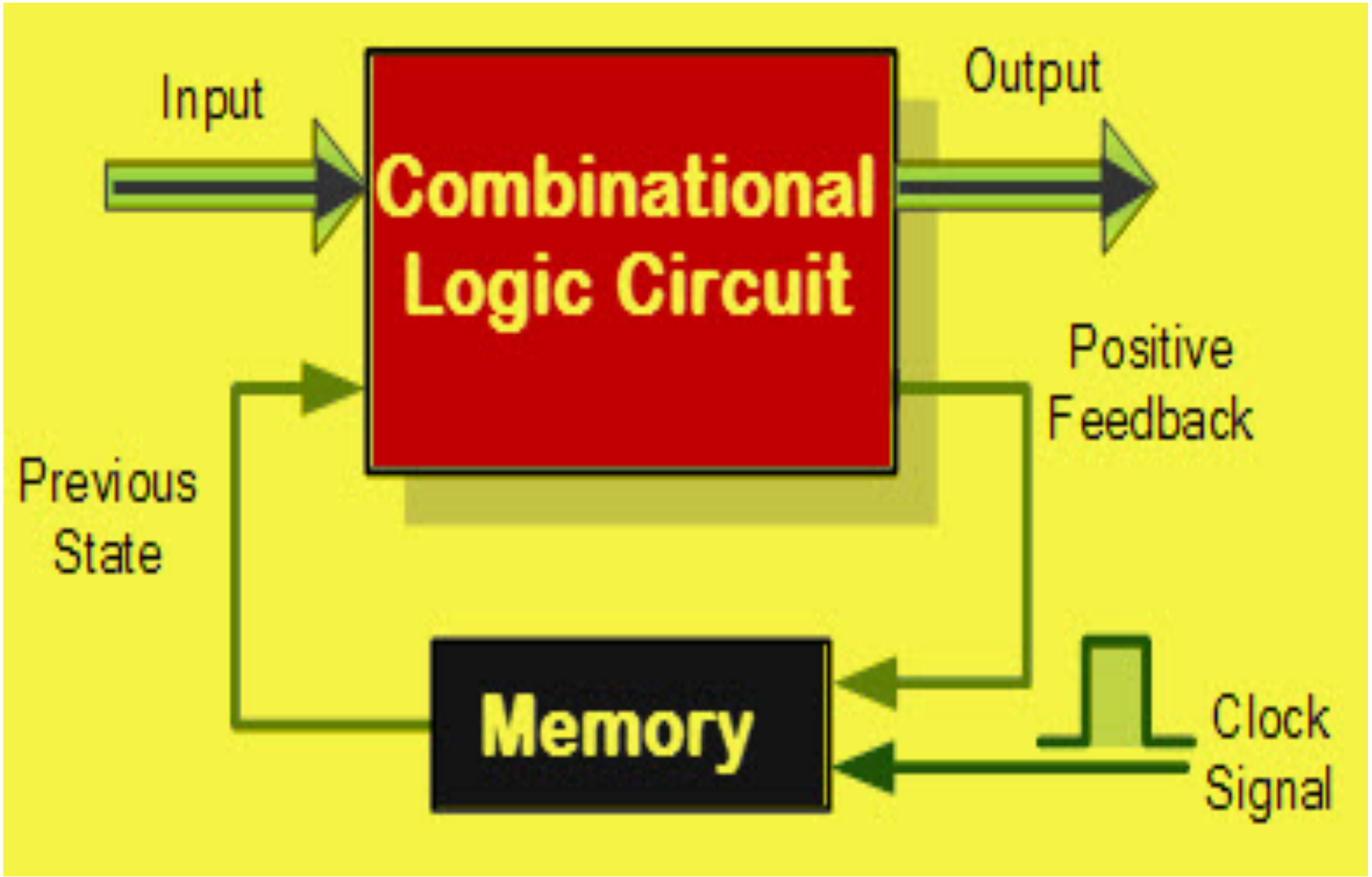
COMBINATIONAL LOGIC CIRCUIT



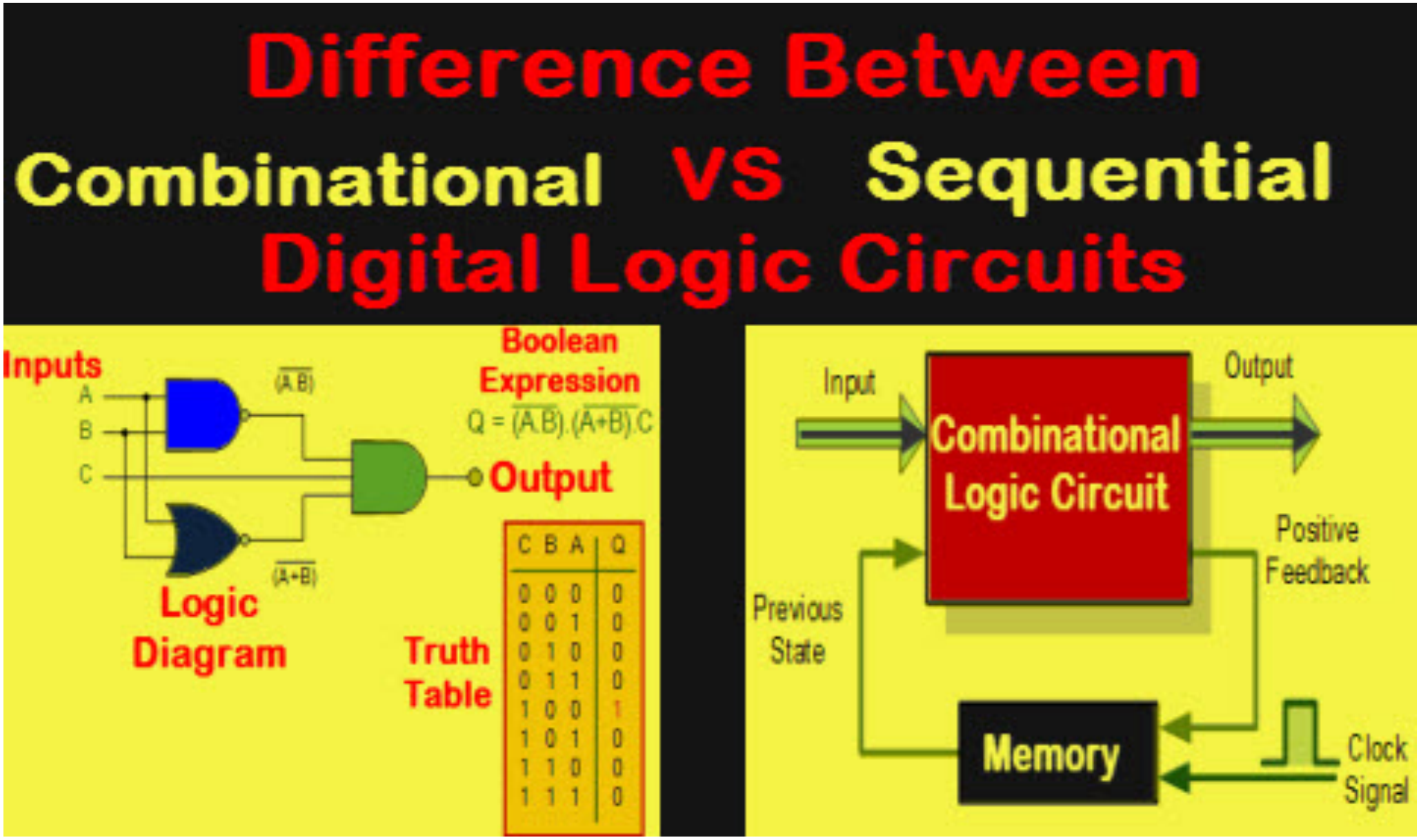
SEQUENTIAL CIRCUITS

- Sequential logic is a type of logic circuit whose output depends not only on the present value of its input signals but on the sequence of past inputs.
- sequential circuits are basically combinational circuits with the additional properties of storage (to remember past inputs) and feedback
- Sequential logic circuits contain memory elements
- Example: counters, register, flip flops etc.

SEQUENTIAL CIRCUITS



SEQUENTIAL vs COMBINATIONAL CIRCUITS

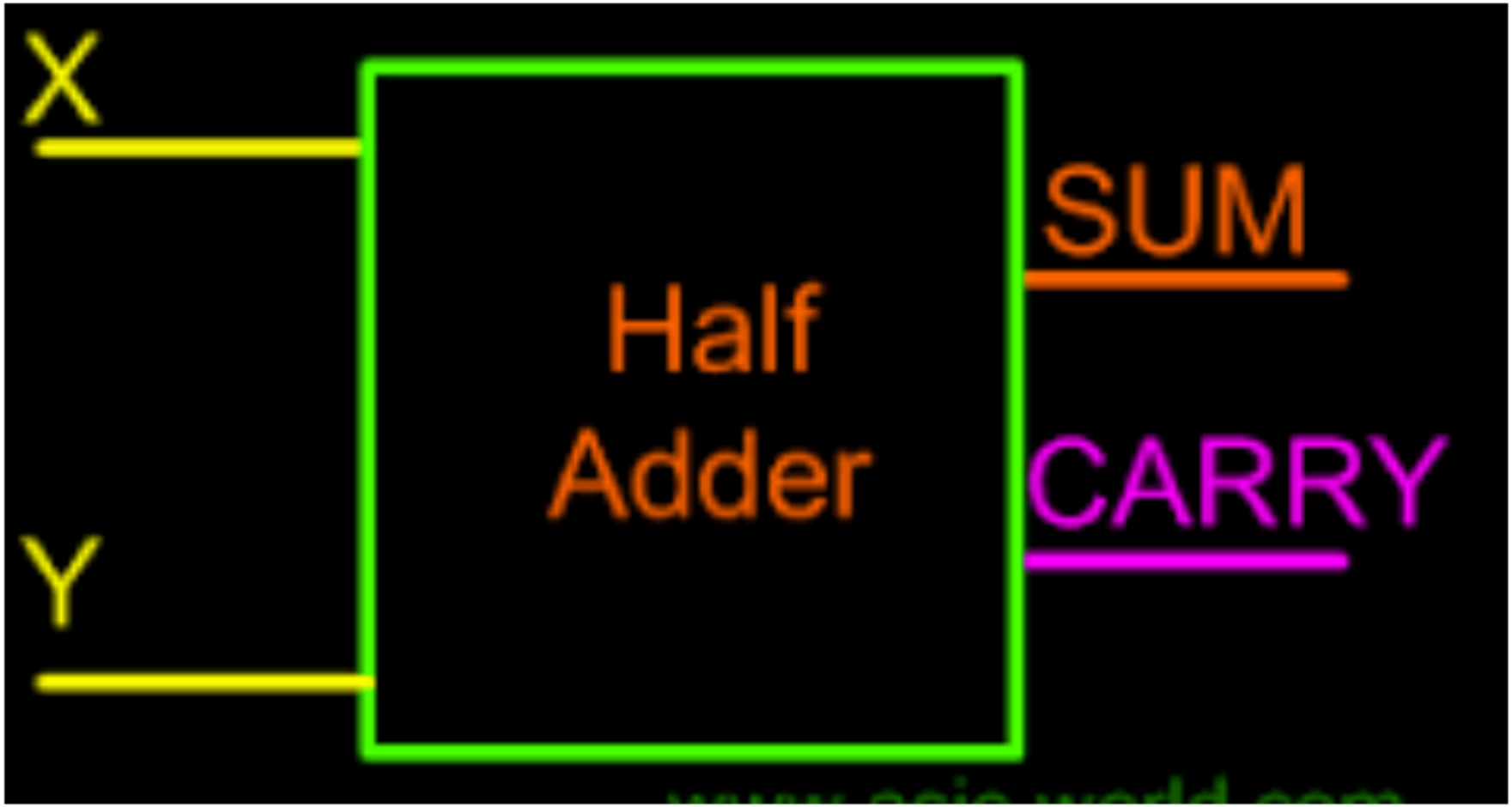


IMPLEMENTATION OF COMBINATIONAL LOGIC CIRCUIT

HALF ADDER

- Half adder is a combinational logic circuit with two inputs and two outputs.
- The half adder circuit is designed to add two single bit binary number A and B.
- It is the basic building block for addition of two **single** bit numbers.
- This circuit has two outputs **carry** and **sum**.

HALF ADDER

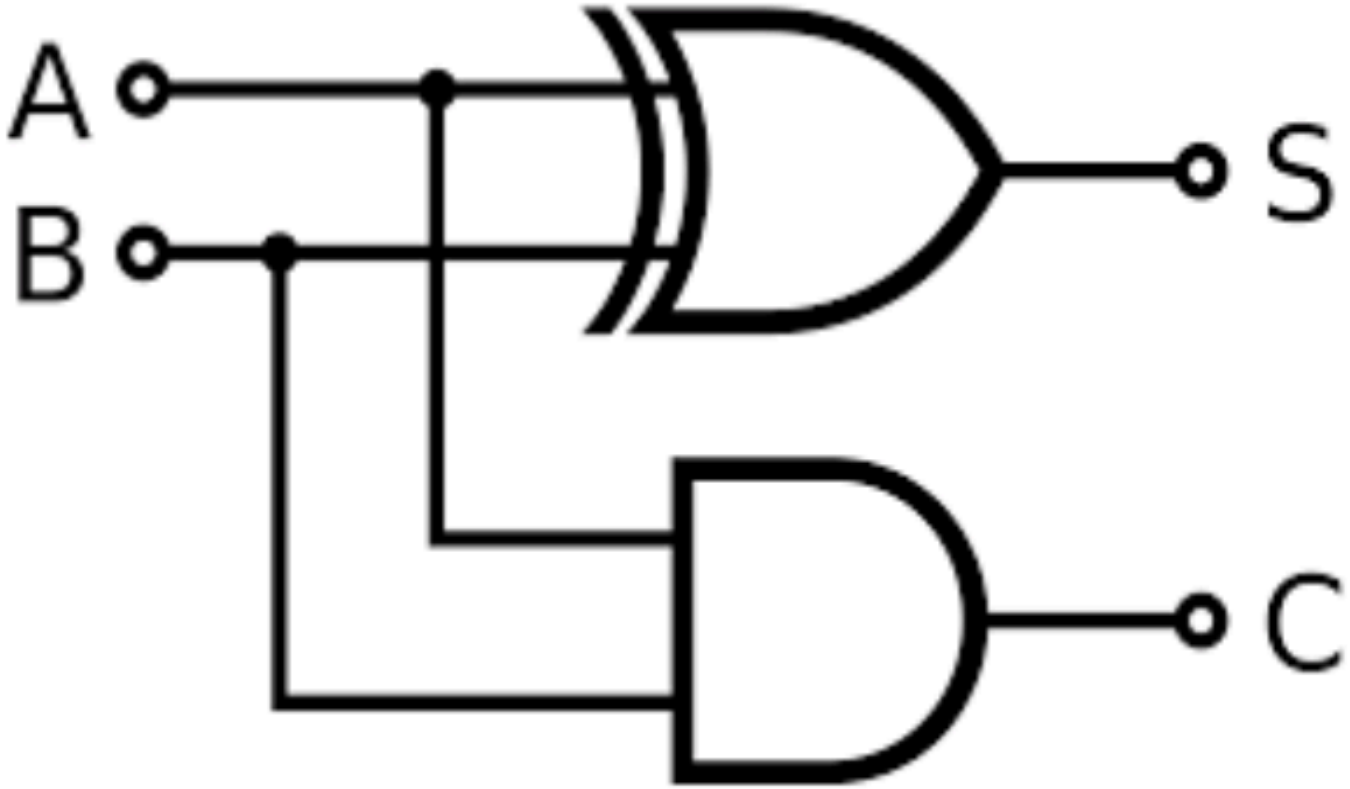


BLOCK DIAGRAM

HALF ADDER

Truth Table			
Input		Output	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

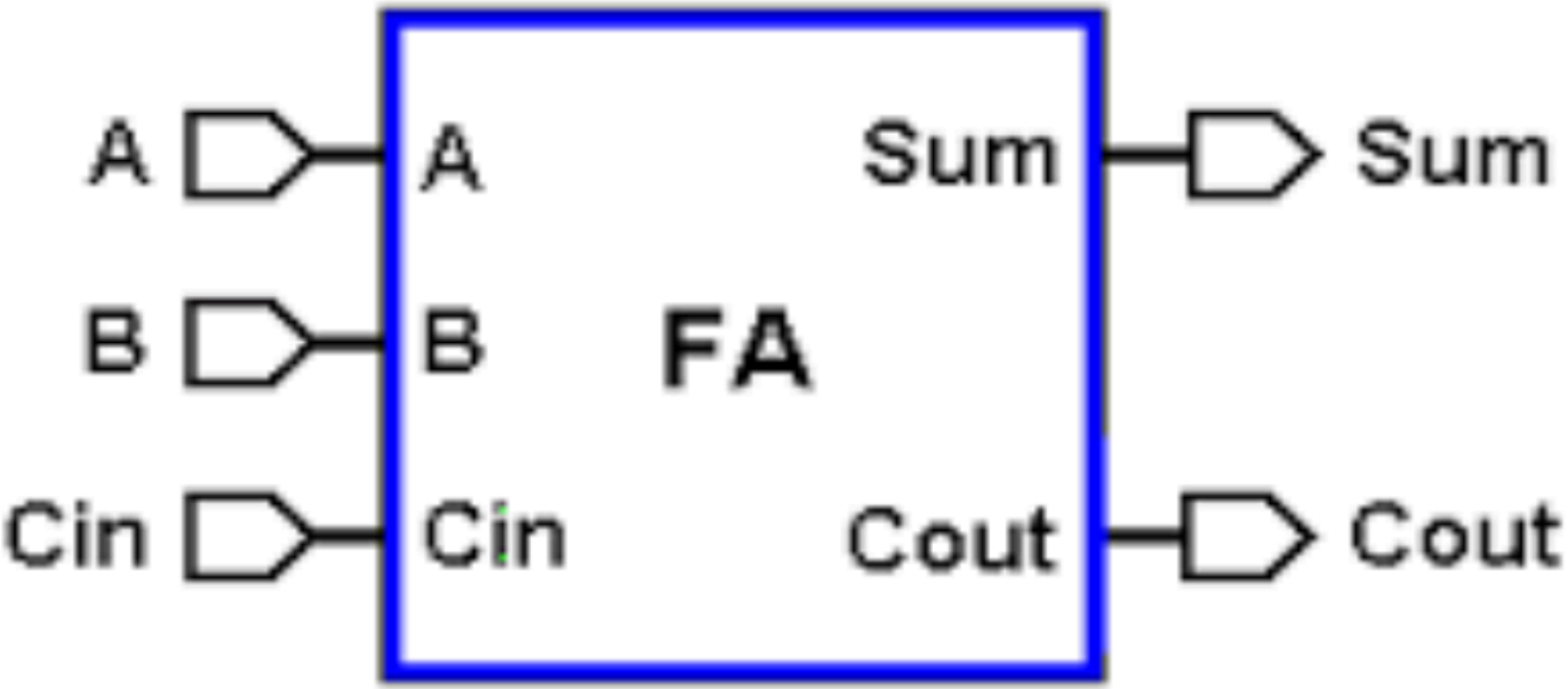
HALF ADDER



FULL ADDER

- Full adder is developed to overcome the drawback of Half Adder circuit.
- It can add two one-bit numbers A and B, and carry c.
- The full adder is a three input and two output combinational circuit.

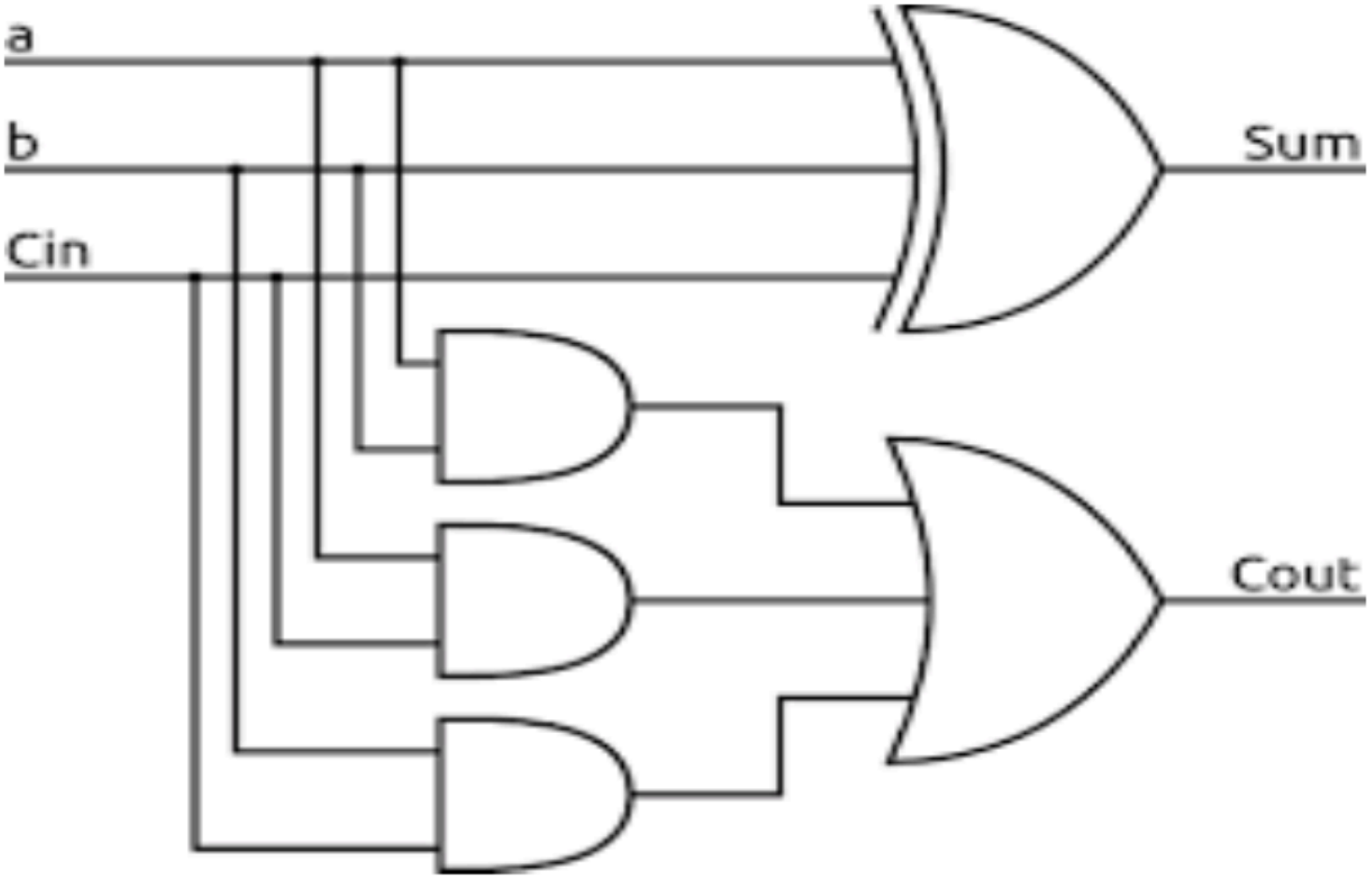
FULL ADDER



FULL ADDER

Full Adder –Truth Table				
Input			Output	
A	B	Carry in	Sum	Carry
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

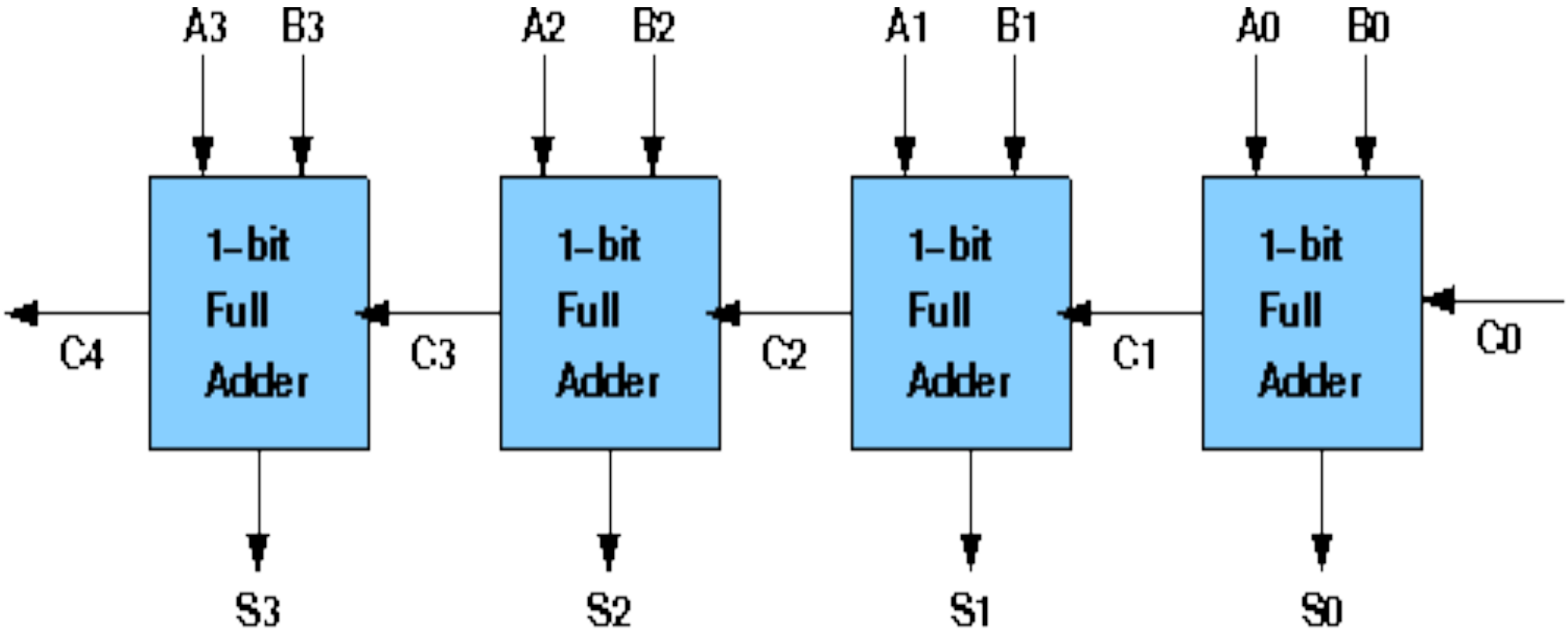
FULL ADDER



$$sum = a \oplus b \oplus Cin \qquad C_{out} = a.b + b.C_{in} + a.C_{in}$$

RIPPLE CARRY ADDER

- The problem of adding two multidigit binary numbers has the following form.
- Two *n-bit* binary numbers are available, with all digits being presented in parallel.
- The addition is performed by using a full adder to add each corresponding pair of digits, one from each number.
- The full adders are connected in tandem so that the carry out from one stage becomes the carry into the next stage



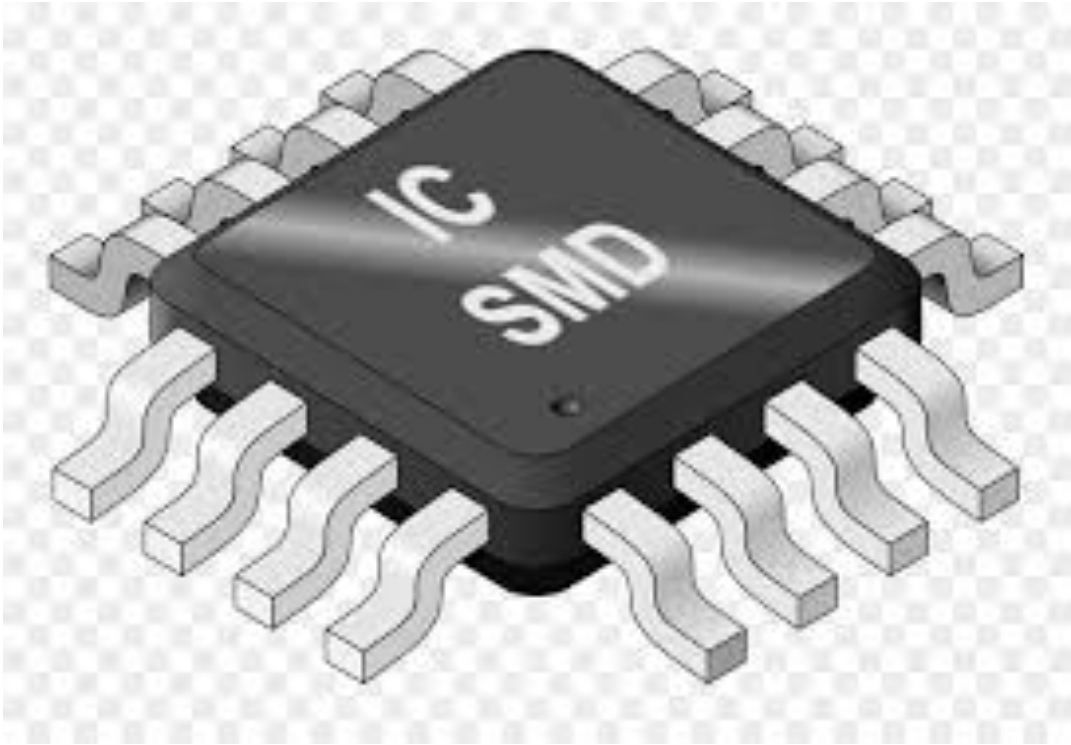
RIPPLE CARRY ADDER

EXERCISE

Design a full subtractor using the concept of full adder.

You need to show truth tables, logical expressions and logic circuit diagram

CLASSIFICATION OF LOGIC GATES AND INTEGRATED CIRCUIT



DWUMFOUR ABDULLAI ABDUL-AZIZ

CLASSIFICATION OF INTEGRATED CIRCUIT (IC)

- As discussed earlier, Individual logic gates can be connected together to form combinational or sequential circuits, or larger logic gate functions, called integrated circuits.
- Standard commercially available digital logic gates are available in two basic families or forms,
  - **TTL**=> *Transistor-Transistor Logic* technology
  - **CMOS** => *Complementary Metal-Oxide- Semiconductor* technology

TTL IC's use NPN and PNP type Bipolar Junction Transistors

CMOS logic IC's use complementary MOSFET or JFET

Bipolar Junction Transistors

Other logic families of integrated circuit include:

- ✓ RTL(Resistor Transistor Logic)
- ✓ DTL(Diode Transistor Logic)
- ✓ ECL(Emitter Coupled Logic)

However, DTL and RTL families of transistors are rarely used in commercial production environment

TECHNOLOGICAL DEVELOPMENT

- Computer technology has shown an unprecedented rate of improvement.
- This includes the development of processors and memories.
- The integration of number of transistors into a single chip has increased from a few hundred to millions.
- This impressive increase has been made possible by the advances in the fabrication technology of transistors.
- This phenomenon has been predicted by a renowned researcher called Gordon Moore

MOORE'S LAW

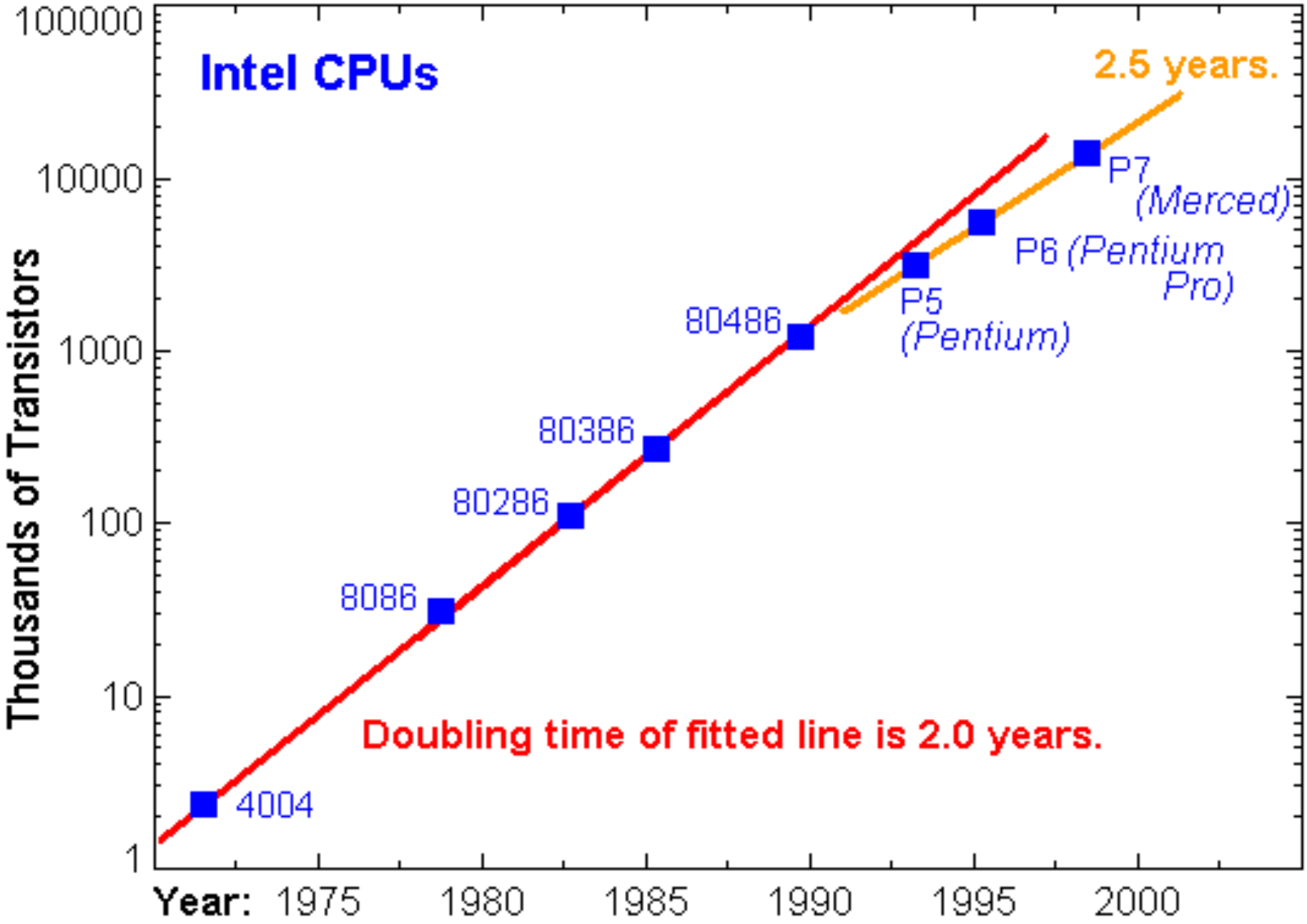
- In 1965, Gordon Moore co-founder of the Intel corporation predicted that "*The number of transistors and resistors on a single chip will double every 2 years* " regarding the development of semiconductor gate technology.

- When Gordon Moore made his famous comment way back in 1965 there were approximately only 60 individual transistor gates on a single silicon chip or die.

MOORE'S LAW

- The worlds first microprocessor in 1971 was the Intel 4004 contained about 2,300 transistors on a single chip,
- Today, the Intel Corporation have placed 1.2 Billion individual transistor gates onto its new Quad-core i7
- The on-chip transistor count is still rising, as newer faster microprocessors and micro-controllers are developed.

Moore's Law


Intel CPUs
2.5 years.
Thousands of Transistors
100000
10000
1000
100
10
1
Year: 1975
1980
1985
1990
1995
2000
4004
8086
80286
80386
80486
P5 (Pentium)
P6 (Pentium Pro)
P7 (Merced)
Doubling time of fitted line is 2.0 years.

CLASSIFICATION OF INTEGRATED CIRCUIT

- Small Scale Integration or (SSI) – Contain up to 10 transistors or a few gates within a single package such as AND, OR, NOT gates.
- Medium Scale Integration or (MSI) – between 10 and 100 transistors or tens of gates within a single package and perform digital operations such as adders, decoders, counters, flip-flops and multiplexers.
- Large Scale Integration or (LSI) – between 100 and 1,000 transistors or hundreds of gates and perform specific digital operations such as I/O chips, memory, arithmetic and logic units

CLASSIFICATION OF INTEGRATED CIRCUIT

- Very-Large Scale Integration or (VLSI) – between 1,000 and 10,000 transistors or thousands of gates and perform computational operations such as processors, large memory arrays and programmable logic devices.
- Super-Large Scale Integration or (SLSI) – between 10,000 and 100,000 transistors within a single package and perform computational operations such as microprocessor chips, micro-controllers, basic PICs and calculators.
- Ultra-Large Scale Integration or (ULSI) – more than 1 million transistors – used in computers CPUs, GPUs, video processors, micro-controllers, FPGAs and complex PICs.

PERFORMANCE MEASUREMENT

- There are various facets to the performance of a computer.
  - User measures performance based on the time taken to execute a given job (program)
  - a laboratory engineer measures the performance of his system by the total amount of work done in a given time
- How fast does computer system execute a program?
  - To determine this, we need to find out the time taken by the computer system to execute a given job.
  - The time required to execute a job by a computer is often expressed in terms of clock cycles.
  - Clock cycle is the distance between two successive rising edges of periodic clock signal

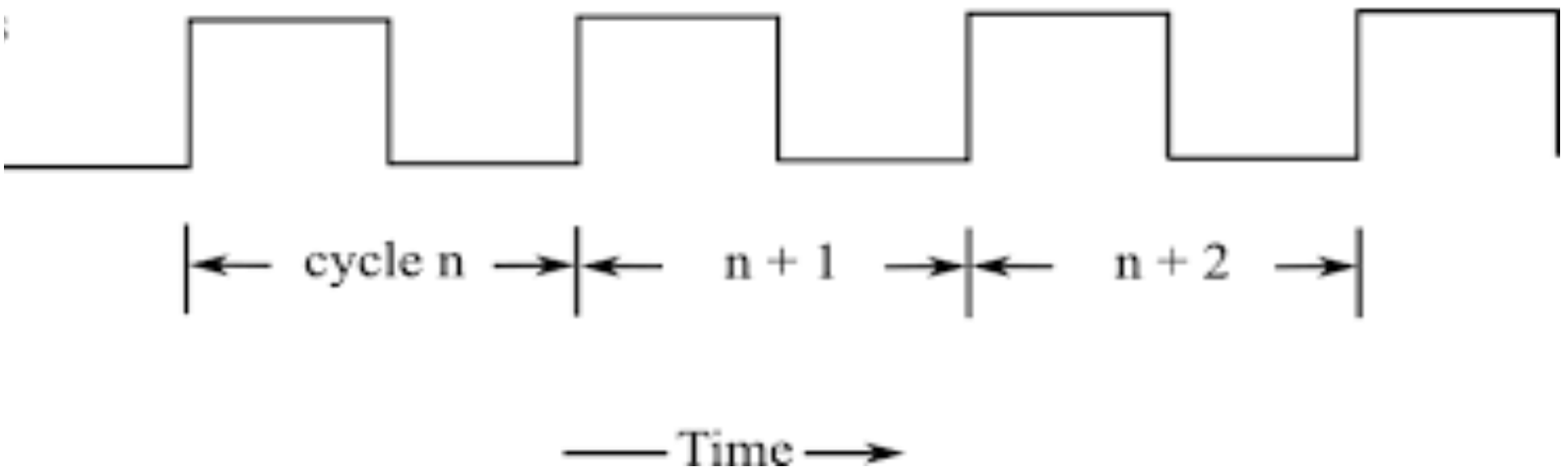
PERFORMANCE MEASUREMENT

- We define the clock cycle time as the time between two successive rising edges of a periodic clock signal.
- Clock cycle count is the number of clock cycles for executing a job
- Clock rate (frequency)or clock speed is the number of clock cycle per second
- Clock rate is inversely proportional to clock cycle time

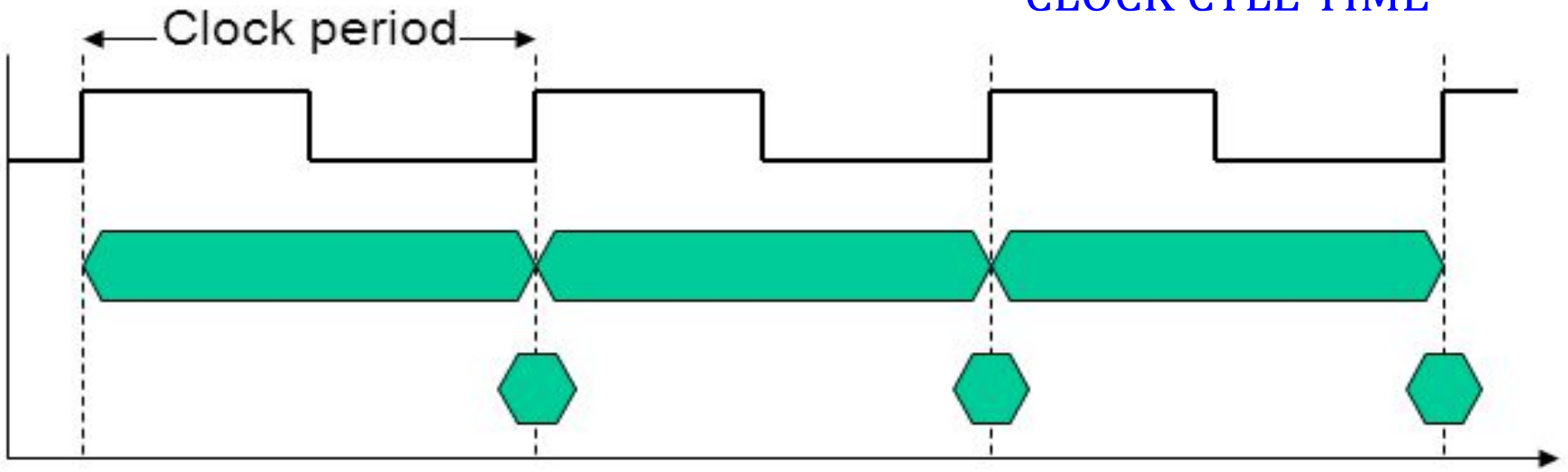
$$CR=1/CT \Leftrightarrow CT=1/CR$$

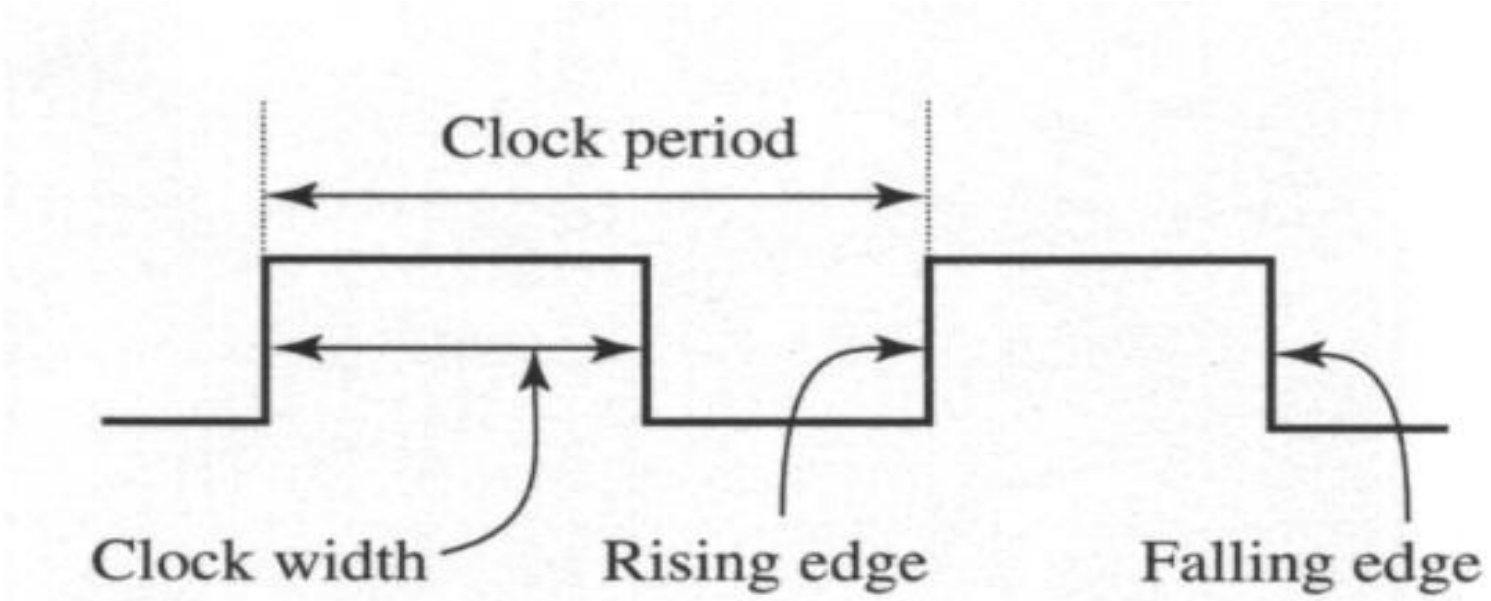
CLOCK CYCLS

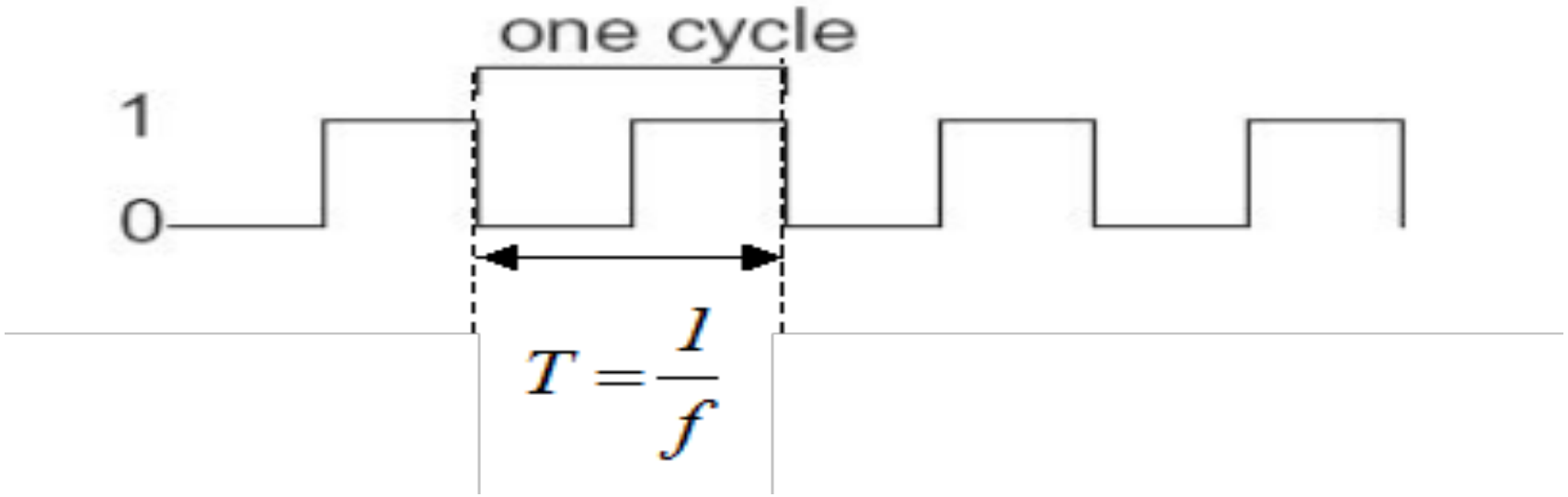
NUMBER OF CLOCK CYCLE



CLOCK CYLE TIME




Clock period
Clock width
Rising edge
Falling edge


one cycle
1
0
$T = \frac{1}{f}$

CPU TIME CALCULATION

We denote clock cycles count by CC,

Cycle Time by CT

Clock frequency or Clock rate is given by F=>1/CT

The time taken by the CPU to execute a job is given by:

$$CPU_{time} = CC * CT \quad \text{but} \quad F = \frac{1}{CT} \Rightarrow CT = \frac{1}{F}$$

$$CPU_{time} = \frac{CC}{F}$$

It may be easier to count the number of instructions executed in a given program as compared to counting the number of CPU clock cycles needed for executing that program

CPU TIME CALCULATION USING NUMBER OF INSTRUCTION

The average number of clock cycles per instruction (CPI) has been used as an alternate performance measure.

$$CPI = \frac{clock\ cycle\ count\ \ for\ that\ \ program(CC)}{instruction\ count(IC)}$$

$$CC = CPI * IC$$

$$CPU_{time} = CC * CT. \qquad but\ CC = CPI * IC$$

$$CPU_{time} = CPI * IC * CT. \qquad but\ CT = I / F$$

$$CPU_{time} = \frac{CPI * IC}{F}$$

$$F\ is\ the\ clock\ rate(CR)$$

CPI COMPUTATION EXAMPLE

Compute the Clock Cycle Per Instruction for a program which is executed in 125 clock cycle with a total of 98 instructions.

CPI=CC/IC

=>125/98

CPI=1.28

What is the cycle time for a computer with the following features

- ✓ 1GHz processor
- ✓ 4GHz processor

A Program is running on a specific machine (CPU) with the following parameters:

- ✓ Total executed instruction count: 10,000,000 instructions
- ✓ Average CPI for the program: 2.5 cycles/instruction.
- ✓ – CPU clock rate: 200 MHz. (clock cycle = C = $5x10^{-9}$ seconds)

what is the execution time of the program

$CPU_{TIME}=IC*CPI*CC$

$CPU_{TIME}=10000000*2.5*(5*10^{-9})$

$CPU_{TIME}=0.125seconds$

COMPUTING CPI FOR A DIFFERENT INSTRUCTION CATEGORIES

- A given instruction set of a given machine may consists of a number of instruction categories:
- ALU (assignment, arithmetic and logic instructions),
- load, store, branch, and so on.
- In the case that the CPI for each instruction category is known, the overall CPI can be computed as

$$CPI = \frac{\sum_{i=1}^{N} CPI_i * I_i}{instruction\ count}$$

COMPUTING CPI FOR A DIFFERENT INSTRUCTION CATEGORIES

- where I_i is the number of times an instruction of type i is executed in the program and CPI_i is the average number of clock cycles needed to execute such instruction.

Example

Consider computing the overall CPI for a machine A for which the following performance measures were recorded when executing a set of benchmark programs.

Assume that the clock rate of the CPU is 200 MHz.

Instruction type	Percentage of occurrence	Cycles Per Instruction (CPI_i)
ALU	**38**	**1**
LOAD&STORE	**15**	**3**
BRANCH	**42**	**4**
OTHERS	**5**	**5**

COMPUTING CPI

Instruction type	Percentage of occurrence	Cycles per instruction
ALU	**38**	**1**
LOAD&STORE	**15**	**3**
BRANCH	**42**	**4**
OTHERS	**5**	**5**
TOTAL	**100**	

$$\sum_{i=1}^{4} CPI_i * I_i = 38*1+15*3+42*4+5*5 = 38+45+168+25 = 276$$

$$CPI = \frac{276}{100} = 2.76$$

Compute the CPU time required to execute this program?

PERFORMANCE MEASUREMENT WITH MIPS

MIPS (Million Instruction Per Second) is a method of measuring the raw speed of a computer's processor

MIPS is defined as the number of machine instructions (in millions) that a processor can execute in one second.

$$MIPS = \frac{Clock\ rate}{Cycle\ Per\ Instruction(CPI) * 10^6} = \frac{cycle/seconds}{cycle/instruction} = \frac{million\ instruction}{second}$$

PERFORMANCE MEASUREMENT WITH MIPS

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{CPU time} \times 10^6} = \frac{\text{Instruction count}}{\text{IC} \times \text{CPI} \times \text{Clock cycle time} \times 10^6} = \frac{\text{IC} \times \text{Clock rate}}{\text{IC} \times \text{CPI} \times 10^6}$$

$$\text{so MIPS} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

PERFORMANCE MEASUREMENT WITH MIPS

A different performance measure that has been given a lot of attention in recent years is MIPS (million instructions-per-second (the rate of instruction execution per unit time)), which is defined as

$$MIPS = \frac{instruction\ count}{execution\ time * 10^6} = \frac{clock\ rate}{CPI * 10^6}$$

Consider the problem above, which run the program on machine A. compute the MIPS

Recall : CPI=2.76, clock rate=200MHz

$$MIPS_a = \frac{200 * 10^6}{2.76 * 10^6} = 72.46$$

Million floating-point instructions per second, MFLOP (rate of floating-point instruction execution per unit time) has also been used as a measure for machines' performance.

It is defined as

$$MFLOPS = \frac{\text{Number of floating-point operations in a program}}{execution\ time * 10^6}$$

While MIPS measures the rate of average instructions, MFLOPS is only defined for the subset of floating-point instructions