COMPUTER ORGANIZATION AND ARCHITECTURE (COA)

EET 2211

4TH SEMESTER – CSE & CSIT

CHAPTER 9,10,11; LECTURE 39

By Ms. Arya Tripathy

William Stallings Computer Organization and Architecture 10th Edition

Chapters - 9,10,11

CHAPTER 9 – THE DECIMAL SYSTEM

TOPICS TO BE COVERED

- The decimal system
- The Binary System
- Converting Between Binary and Decimal
- Hexadecimal Notation

LEARNING OBJECTIVES

- Understand the basic concepts and terminology of positional number systems.
- Explain the techniques for converting between decimal and binary for both integers and fractions.
- Explain the rationale for using hexadecimal notation.

Decimal Number System

- Base (also called radix) = 10
  - 10 digits { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
- Digit Position
  - Integer & fraction
- Digit Weight
  - Weight = $(Base)^{Position}$
- Magnitude
  - Sum of "*Digit* x *Weight*"
- Formal Notation

2	1	0	-1	-2
5	1	2	7	4
100	10	1	0.1	0.01
500	10	2	0.7	0.04

$$d_2*B^2+d_1*B^1+d_0*B^0+d_{-1}*B^{-1}+d_{-2}*B^{-2}$$

$$(512.74)_{10}$$

Binary Number System

- Base = 2
  - 2 digits { 0, 1 }, called *b*inary dig*its* or "*bits*"
- Weights
  - Weight = $(Base)^{Position}$
- Magnitude
  - Sum of "*Bit* x *Weight*"
- Formal Notation
- Groups of bits

4 bits = *Nibble*

4	2	1	1/2	1/4
1	0	1	0	1
2	1	0	-1	-2

$$1 * 2^{2} + 0 * 2^{1} + 1 * 2^{0} + 0 * 2^{-1} + 1 * 2^{-2}$$

$$= (5.25)_{10}$$

$$(101.01)_{2}$$

Hexadecimal Number System

- Base = 16
  - 16 digits { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F }
- Weights
  - Weight = $(Base)^{Position}$
- Magnitude
  - Sum of "*Digit* x *Weight*"
- Formal Notation

256	16	1	1/16	1/256
1	E	5	7	A
2	1	0	-1	-2

$$1 * 16^{2} + 14 * 16^{1} + 5 * 16^{0} + 7 * 16^{-1} + 10 * 16^{-2}$$

$$= (485.4765625)_{10}$$

$$(1E5.7A)_{16}$$

Decimal to Binary Conversion

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

Example: $(13)_{10}$

	Quotient	Remainder	Coefficient
$13 / 2 =$	6	1	$a_0 = 1$
$6 / 2 =$	3	0	$a_1 = 0$
$3 / 2 =$	1	1	$a_2 = 1$
$1 / 2 =$	0	1	$a_3 = 1$

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

MSB LSB

Decimal (*Fraction*) to Binary Conversion

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

Example: $(0.625)_{10}$

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

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

MSB LSB

REVIEW QUESTIONS

9.1 Count from 1 to 20_{10} in the following bases:

a. 8 **b.** 6 **c.** 5 **d.** 3

9.2 Order the numbers $(1.1)_2$, $(1.4)_{10}$, and $(1.5)_{16}$ from smallest to largest.

9.3 Perform the indicated base conversions:

a. 54_8 to base 5 **b.** 312_4 to base 7 **c.** 520_6 to base 7 **d.** 12212_3 to base 9

9.4 What generalizations can you draw about converting a number from one base to a power of that base; e.g., from base 3 to base 9 (3^2) or from base 2 to base 4 (2^2) or base 8 (2^3)?

9.5 Convert the following binary numbers to their decimal equivalents:

a. 001100 **b.** 000011 **c.** 011100 **d.** 111100 **e.** 101010

9.6 Convert the following binary numbers to their decimal equivalents:

a. 11100.011 **b.** 110011.10011 **c.** 1010101010.1

9.7 Convert the following decimal numbers to their binary equivalents:

a. 64 **b.** 100 **c.** 111 **d.** 145 **e.** 255

9.8 Convert the following decimal numbers to their binary equivalents:

a. 34.75 **b.** 25.25 **c.** 27.1875

Contd.

9.9 Prove that every real number with a terminating binary representation (finite number of digits to the right of the binary point) also has a terminating decimal representation (finite number of digits to the right of the decimal point).

9.10 Express the following octal numbers (number with radix 8) in hexadecimal notation:

a. 12 **b.** 5655 **c.** 2550276 **d.** 76545336 **e.** 3726755

9.11 Convert the following hexadecimal numbers to their decimal equivalents:

a. C **b.** 9F **c.** D52 **d.** 67E **e.** ABCD

9.12 Convert the following hexadecimal numbers to their decimal equivalents:

a. F.4 **b.** D3.E **c.** 1111.1 **d.** 888.8 **e.** EBA.C

9.13 Convert the following decimal numbers to their hexadecimal equivalents:

a. 16 **b.** 80 **c.** 2560 **d.** 3000 **e.** 62,500

9.14 Convert the following decimal numbers to their hexadecimal equivalents:

a. 204.125 **b.** 255.875 **c.** 631.25 **d.** 10000.00390625

9.15 Convert the following hexadecimal numbers to their binary equivalents:

a. E **b.** 1C **c.** A64 **d.** 1F.C **e.** 239.4

9.16 Convert the following binary numbers to their hexadecimal equivalents:

a. 1001.1111 **b.** 110101.011001 **c.** 10100111.111011

CHAPTER 10 – COMPUTER ARITHMETIC

TOPICS TO BE COVERED

- Integer Representation
- Integer Arithmetic
- Floating-Point Representation
- Floating-Point Arithmetic

LEARNING OBJECTIVES

- Understand the distinction between the way in which numbers are represented (the binary format) and the algorithms used for the basic arithmetic operations.
- Explain two's complement representation.
- Understand base and exponent in the representation of floating-point numbers.
- Present an overview of the IEEE 754 standard for floating-point representation.

Integer Representation

- Only have 0 & 1 to represent everything
- Positive numbers stored in binary

  e.g. 41=00101001
- No minus sign
- No period
- Sign-Magnitude
- Two's compliment

Sign-Magnitude

- Left most bit is sign bit
- 0 means positive
- 1 means negative

  +18 = 00010010

  -18 = 10010010

- Problems

  Need to consider both sign and magnitude in arithmetic

  Two representations of zero (+0 and -0)

Two's Complement

•It uses the MSB as a sign bit, making it easy to test whether an integer is positive or negative.

- +3 = 00000011
- +2 = 00000010
- +1 = 00000001
- +0 = 00000000
- -1 = 11111111
- -2 = 11111110
- -3 = 11111101

Range of Numbers

- 8 bit 2s compliment

  $+127 = 01111111 = 2^7 - 1$

  $-128 = 10000000 = -2^7$

- 16 bit 2s compliment

  $+32767 = 011111111\ 11111111 = 2^{15} - 1$

  $-32768 = 100000000\ 00000000 = -2^{15}$

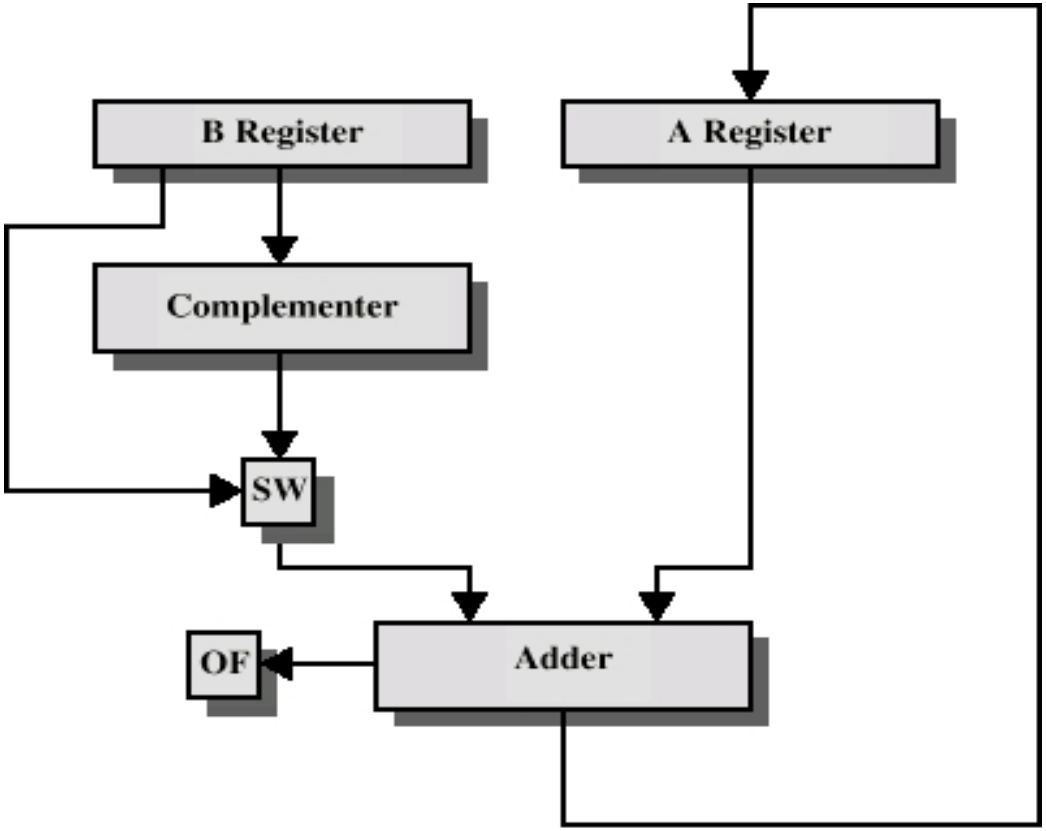
Addition and Subtraction

- Normal binary addition
- Monitor sign bit for overflow
- Take twos compliment of substahend and add to minuend

  i.e. a - b = a + (-b)

- So we only need addition and complement circuits

Hardware for Addition and Subtraction



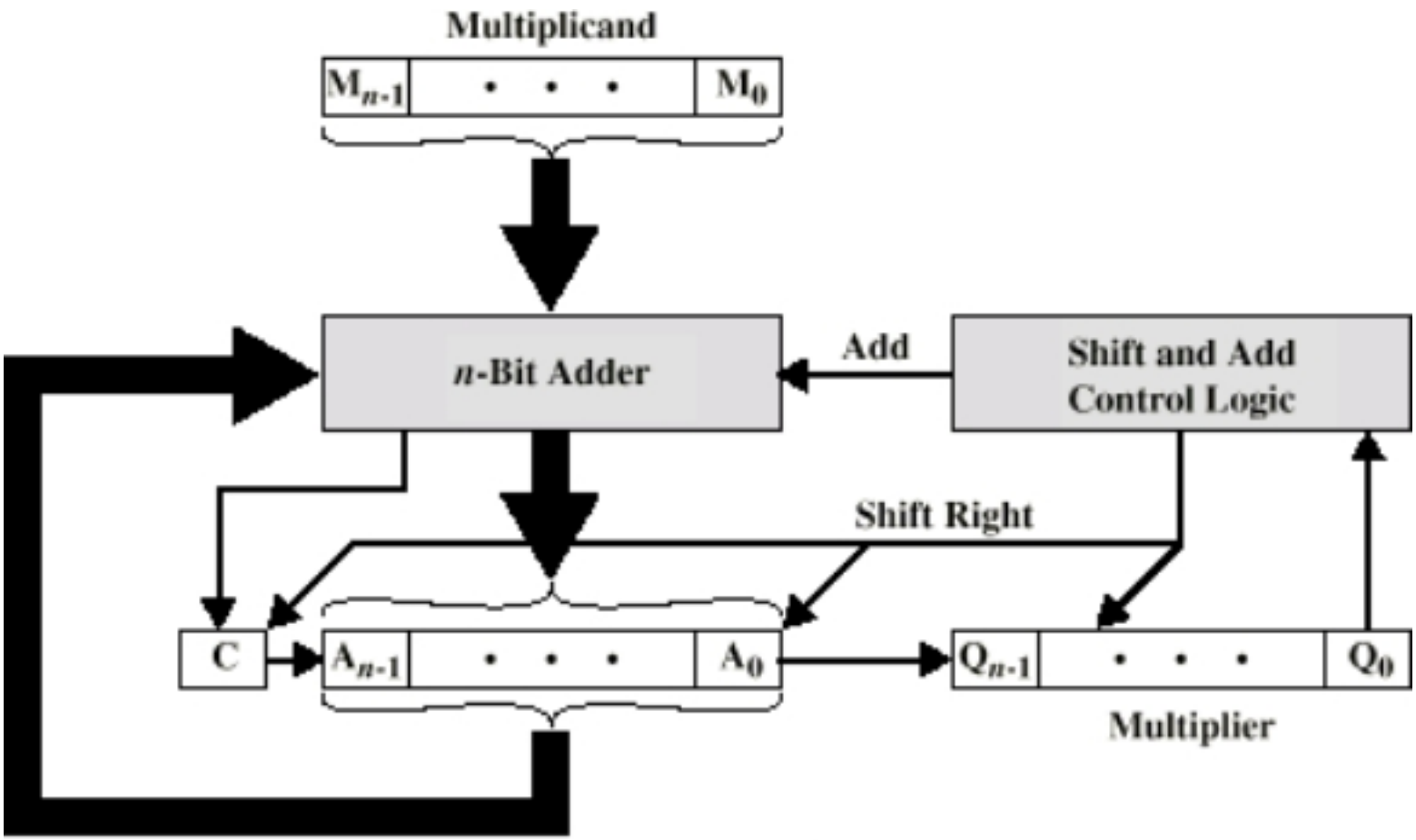
OF = overflow bit
SW = Switch (select addition or subtraction)

Multiplication

- ✓ It is a complex operation whether performed on hardware or software, compared with addition and subtraction.
- ✓ Important features of multiplication are:

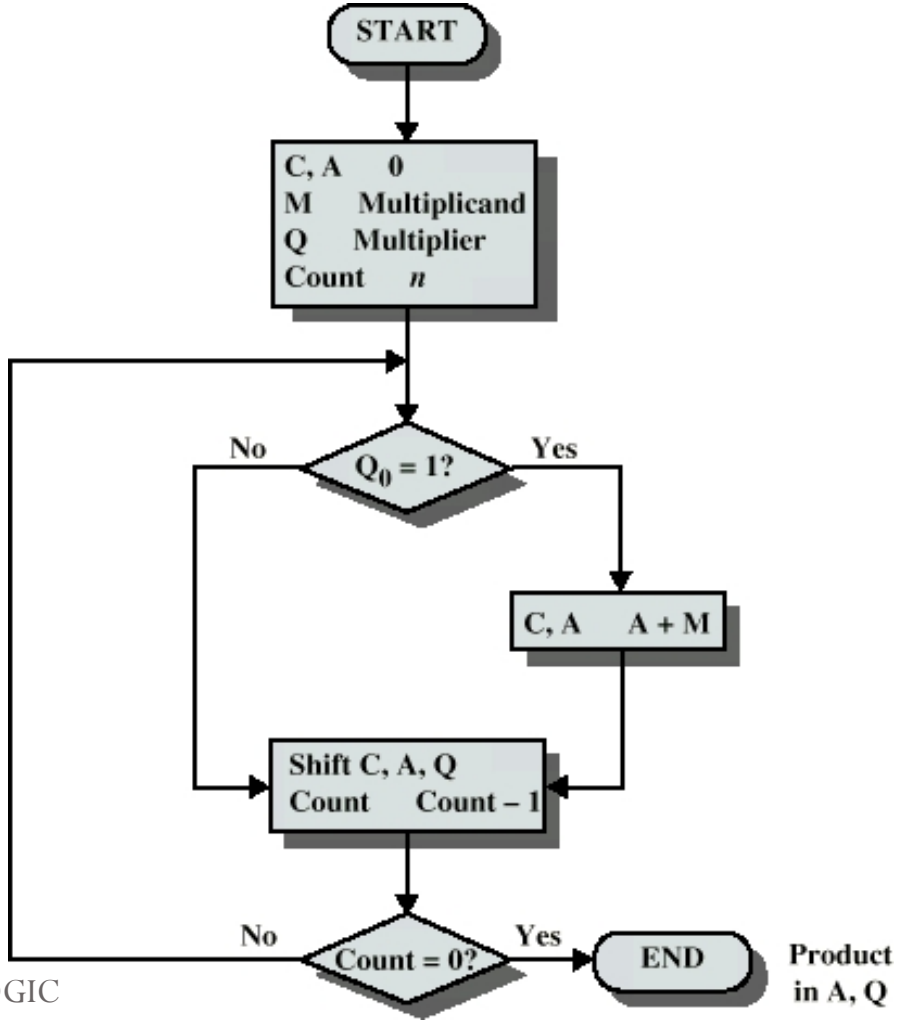
1. It involves the generation of partial products, one for each digit in the multiplier.
2. The partial products are easily defined.
3. The total product is produced by summing the partial products.
4. The multiplication of two n-bit binary integers results in a product of up-to 2n bits in length.

Unsigned Binary Multiplication



(a) Block Diagram

Flowchart for Unsigned Binary Multiplication



Multiplying Negative Numbers

- This does not work!
- Solution 1

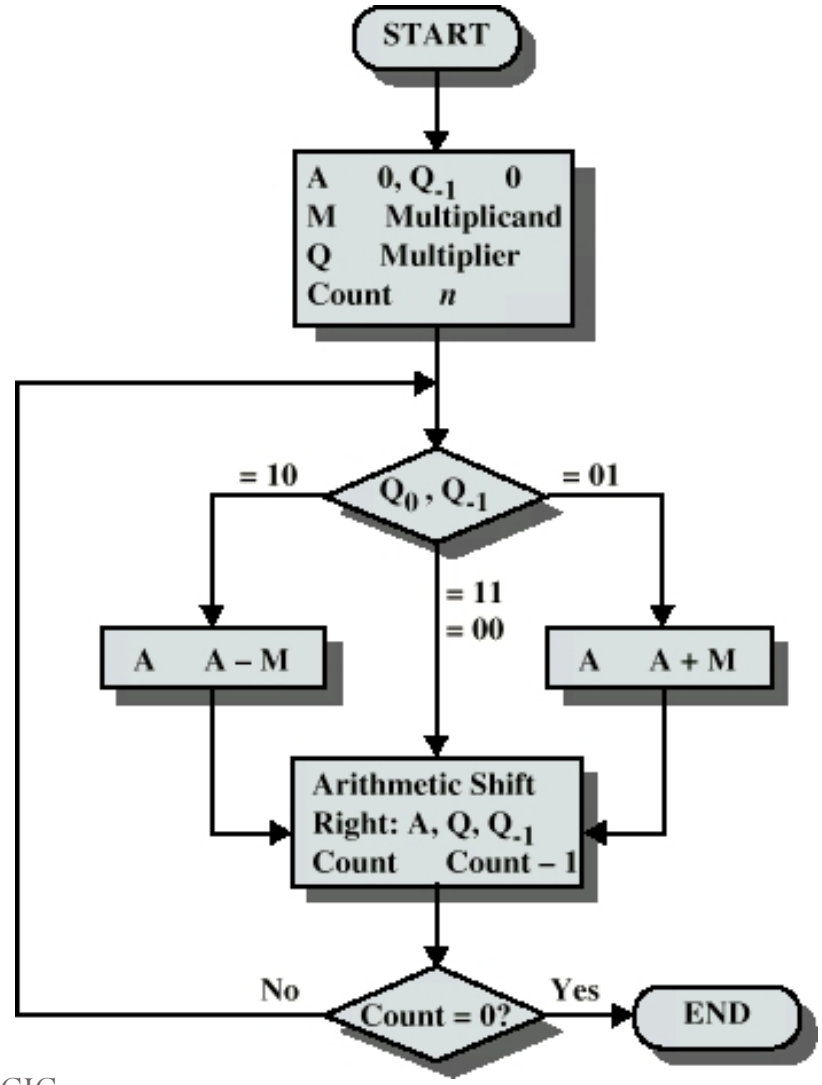
  Convert to positive if required

  Multiply as above

  If signs were different, negate answer
- Solution 2

  Booth's algorithm

Booth's Algorithm



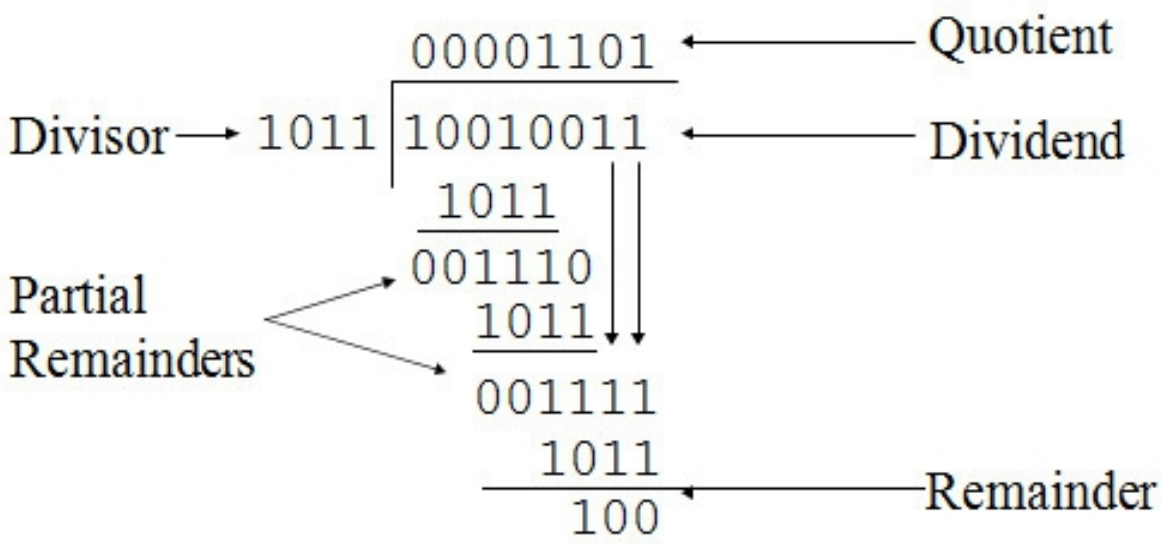
Example of Booth's Algorithm

A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial Values	
1001	0011	0	0111	A A - M	First Cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second Cycle
0101	0100	1	0111	A A + M	Third Cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth Cycle

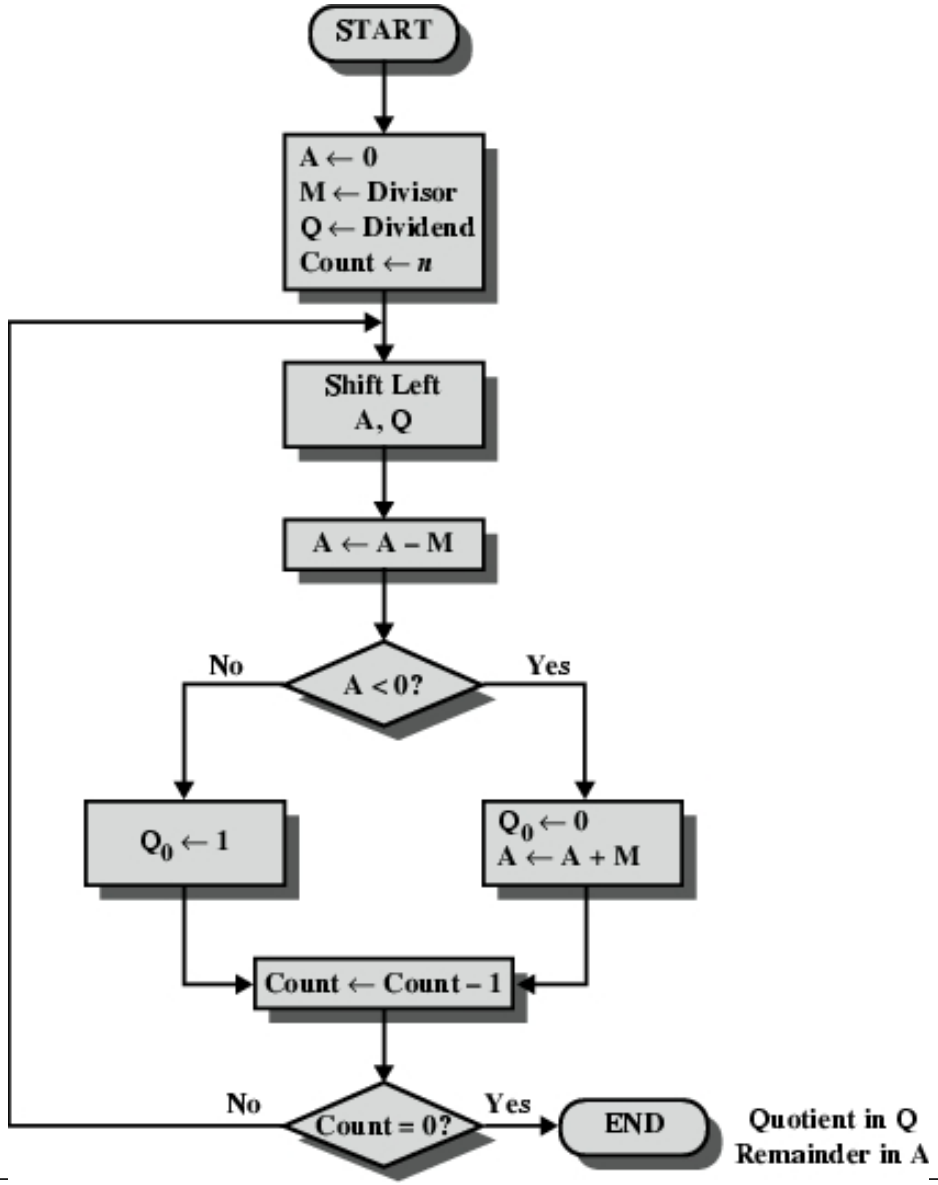
Division

- More complex than multiplication
- Negative numbers are really bad!
- Based on long division

Division of Unsigned Binary Integers



Flowchart for Unsigned Binary Division



Real Numbers

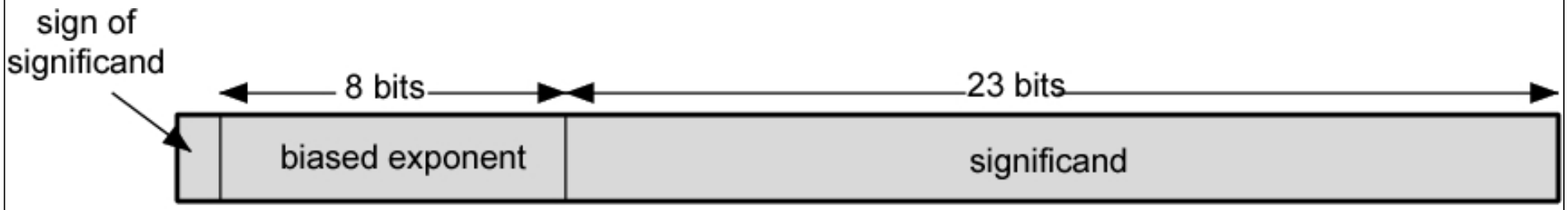
- Numbers with fractions
- Could be done in pure binary

  $1001.1010 = 2^4 + 2^0 + 2^{-1} + 2^{-3} = 9.625$
- Where is the binary point?
- Fixed?

  Very limited
- Moving?

  How do you show where it is?

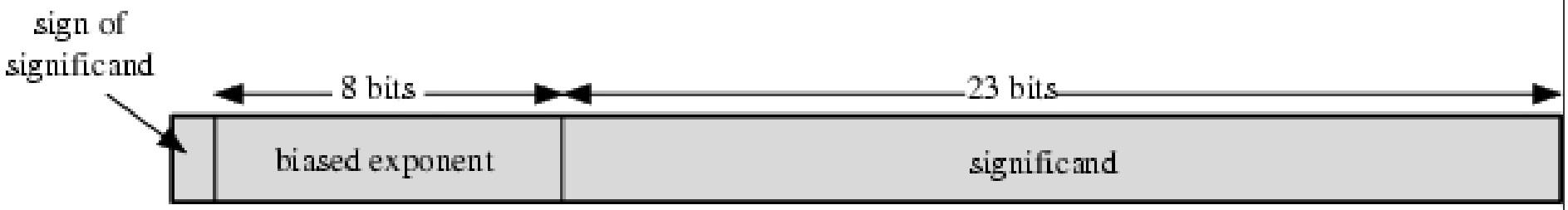
Floating Point



(a) Format

- +/- .significand x $2^{exponent}$
- Misnomer
- Point is actually fixed between sign bit and body of mantissa
- Exponent indicates place value (point position)

Floating Point Examples



(a) Format

$$1.1010001 \times 2^{10100} = 0\ 10010011\ 10100010000000000000000 = 1.638125 \times 2^{20}$$
$$-1.1010001 \times 2^{10100} = 1\ 10010011\ 10100010000000000000000 = -1.638125 \times 2^{20}$$
$$1.1010001 \times 2^{-10100} = 0\ 01101011\ 10100010000000000000000 = 1.638125 \times 2^{-20}$$
$$-1.1010001 \times 2^{-10100} = 1\ 01101011\ 10100010000000000000000 = -1.638125 \times 2^{-20}$$

(b) Examples

Signs for Floating Point

- Mantissa is stored in 2s compliment
- Exponent is in excess or biased notation

  e.g. Excess (bias) 128 means

  8 bit exponent field

  Pure value range 0-255

  Subtract 128 to get correct value

  Range -128 to +127

Normalization

- FP numbers are usually normalized

  i.e. exponent is adjusted so that leading bit (MSB) of mantissa is 1

- Since it is always 1 there is no need to store it

  (c.f. Scientific notation where numbers are normalized to give a single digit before the decimal point

  e.g. 3.123×10^3)

FP Ranges

- For a 32 bit number

  8 bit exponent

  $+/- 2^{256}$ 1.5×10^{77}

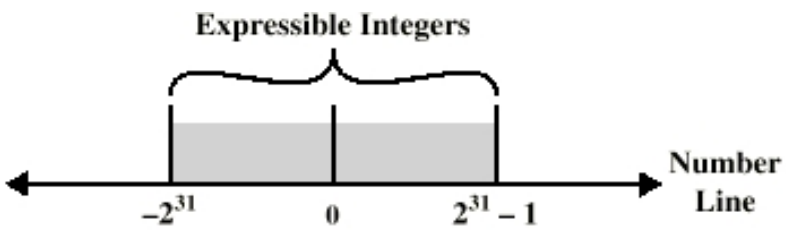
- Accuracy

  The effect of changing LSB of mantissa

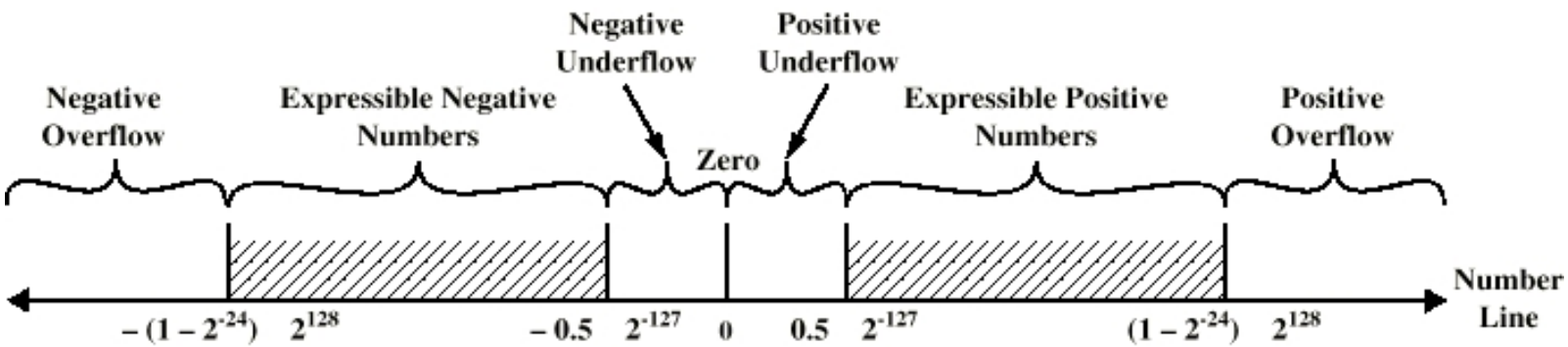
  23 bit mantissa 2^{-23} 1.2×10^{-7}

  About 6 decimal places

Expressible Numbers



(a) Twos Complement Integers



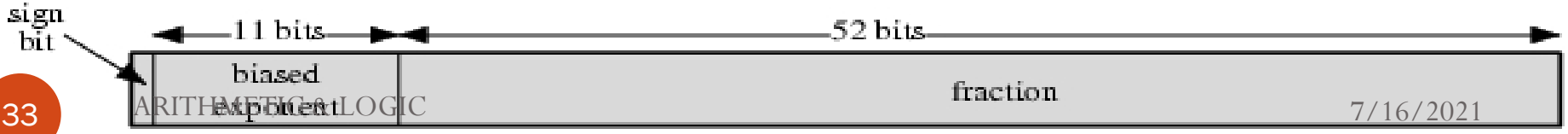
(b) Floating-Point Numbers

IEEE 754

- Standard for floating point storage
- 32 and 64 bit standards
- 8 and 11 bit exponent respectively
- Extended formats (both mantissa and exponent) for intermediate results

IEEE 754 Formats

(a) Single format

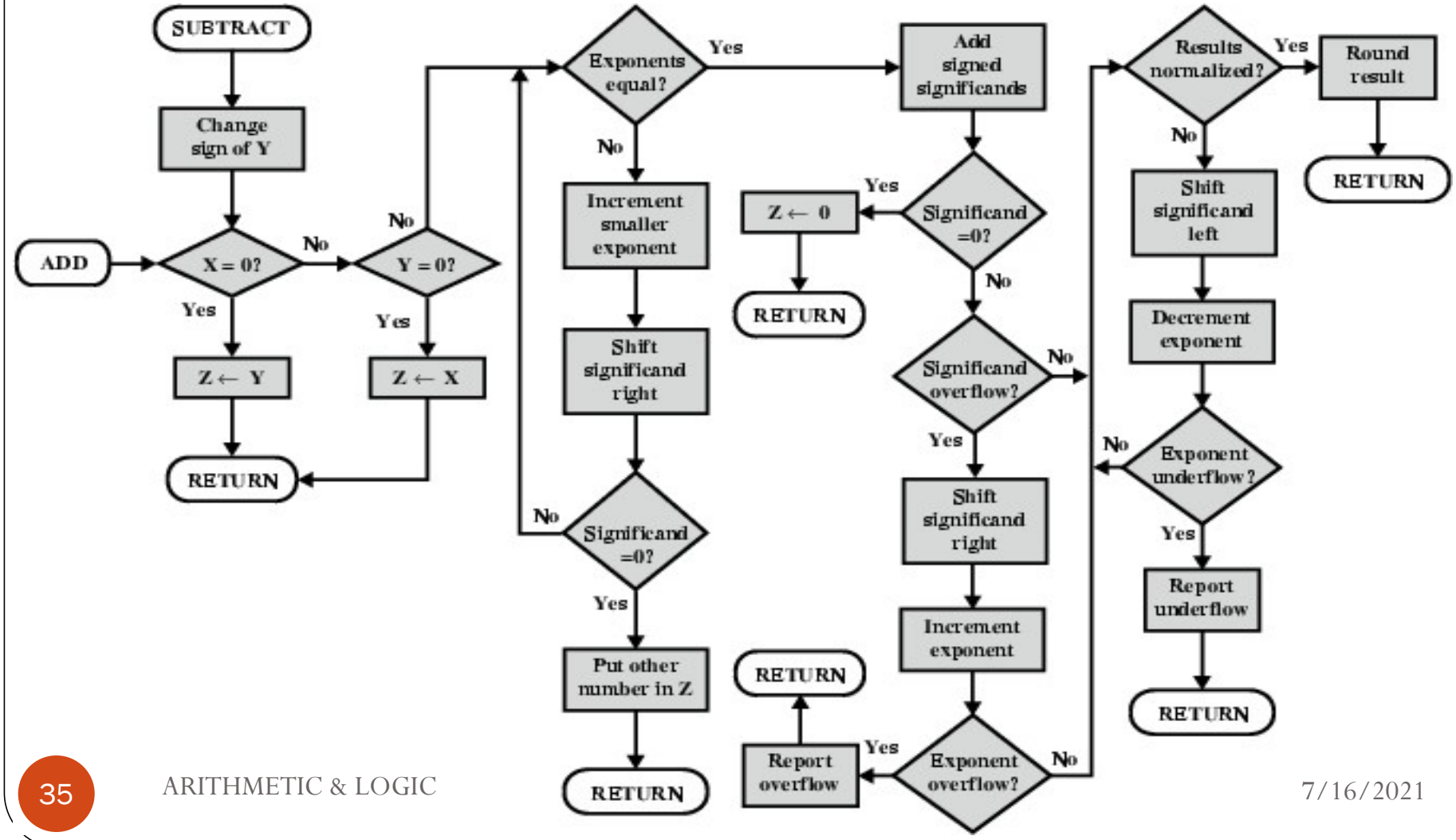


(b) Double format

FP Arithmetic +/-

- Check for zeros
- Align significands (adjusting exponents)
- Add or subtract significands
- Normalize result

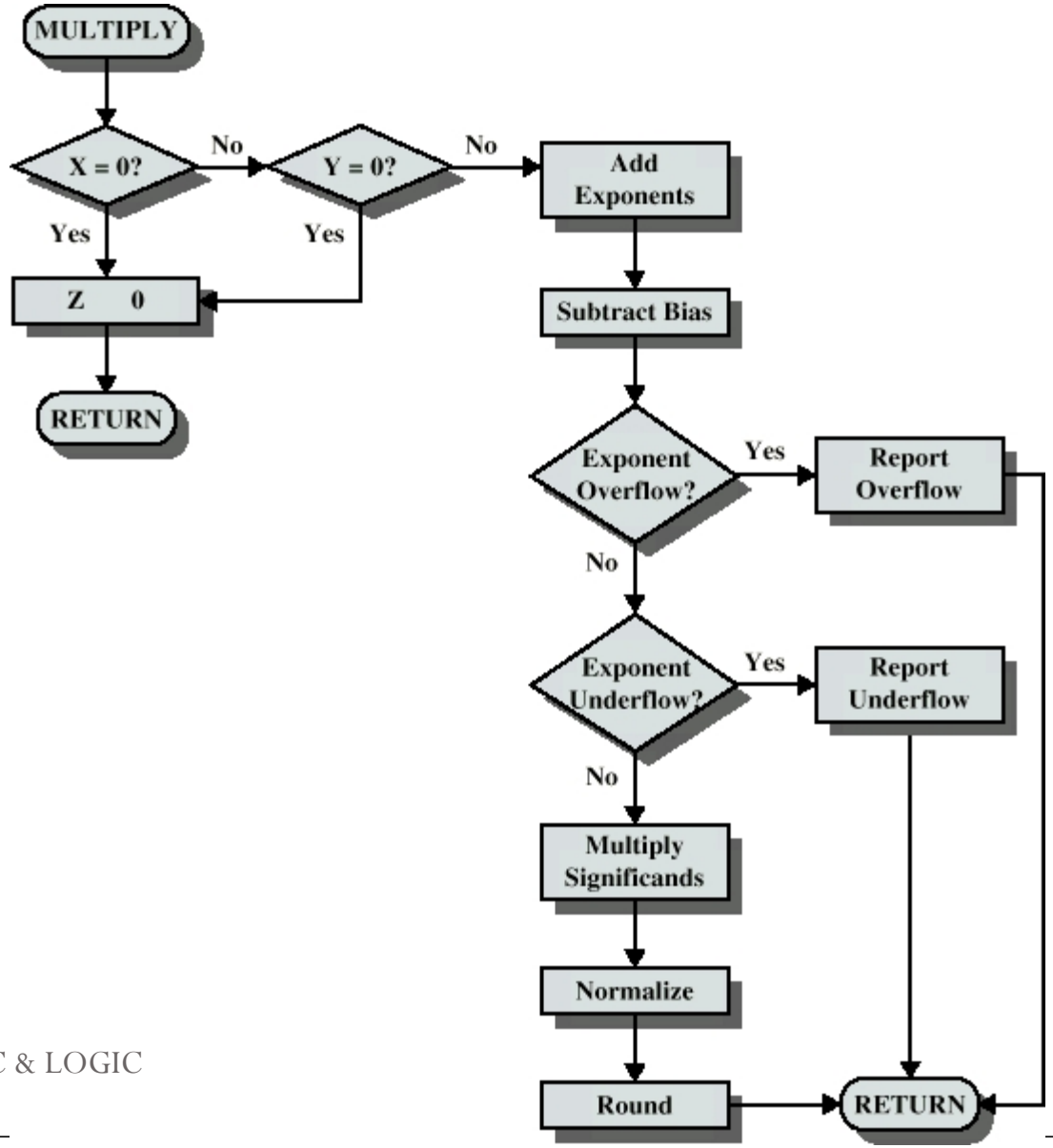
FP Addition & Subtraction Flowchart



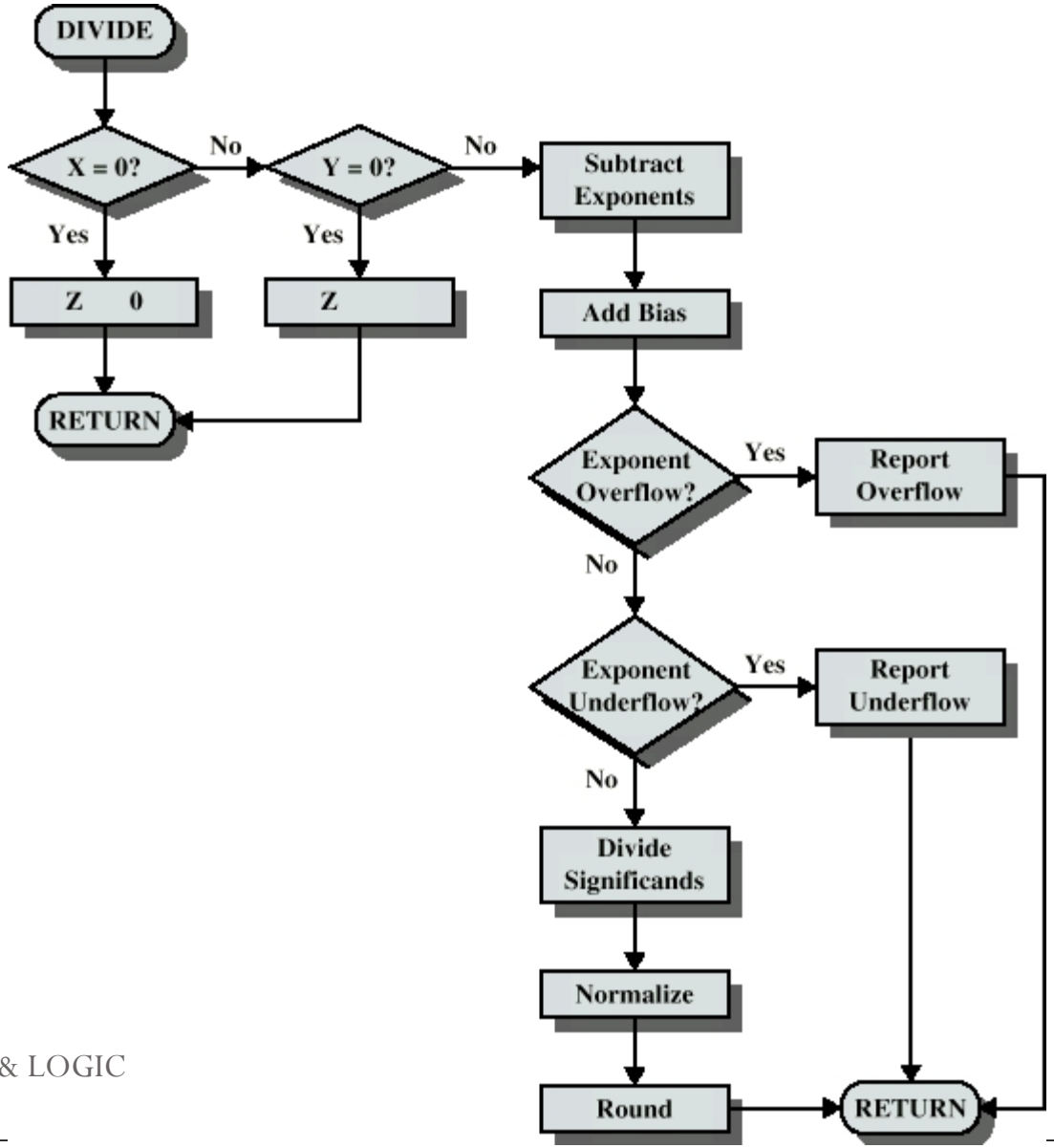
FP Arithmetic x/÷

- Check for zero
- Add/subtract exponents
- Multiply/divide significands (watch sign)
- Normalize
- Round
- All intermediate results should be in double length storage

Floating Point Multiplication



Floating Point Division



COMBINATIONAL CIRCUITS

- Combinational circuit is a circuit in which we combine the different gates in the circuit, for example encoder, decoder, multiplexer and demultiplexer.
- Some of the characteristics of combinational circuits are following:

✓ The output of combinational circuit at any instant of time, depends only on the levels present at inputterminals.

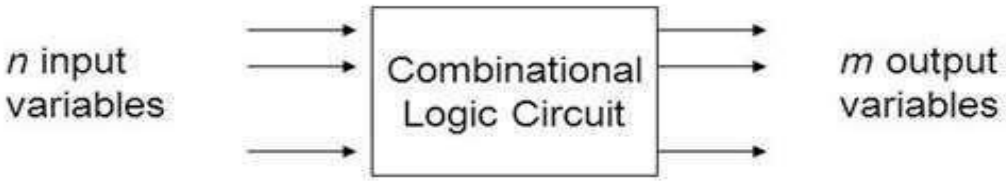
✓ The combinational circuit do not use any memory. The previous state of input does not have any effect on the present state of the circuit.

✓ A combinational circuit can have an n number of inputs and m numbe of outputs.

COMBINATIONAL CIRCUITS

- **Block diagram:**

2^n possible combinations of input values.



- Specific types of combinational circuits: Adders, subtractors, multiplexers, comprators, encoder, decoder.

ANALYSIS PROCEDURE

Analysis procedure

To obtain the output Boolean functions from a logic diagram, proceed as follows:

- Label all gate outputs that are a function of input variables with arbitrary symbols. Determine the Boolean functions for each gate output.
- Label the gates that are a function of input variables and previously labeled gates with other arbitrary symbols. Find the Boolean functions for these gates.
- Repeat the process outlined in step 2 until the outputs of the circuit are obtained.

DESIGN PROCEDURE

- The problem is stated.
- The number of available input variables and requiredoutput variables is determined.
- The input and output variables are assigned lettersymbols.
- The truth table that defines the required relationship between inputs and outputs is derived.
- The simplified Boolean function for each output is obtained.
- The logic diagram is drawn.

BINARY ADDERS

Full Adder

The full-adder adds the bits A and B and the carry from the previous column called the carry-in C_{in} and outputs the sum bit S and the carry bit called the carry-out C_{out}.

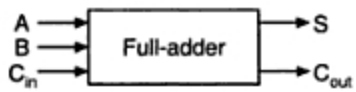


Fig 3: block diagram

Inputs			Sum	Carry
A	B	C_{in}	S	C_{out}
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

Fig 4:Truth table

$$S = \overline{A}\,\overline{B}C_{in} + \overline{A}B\overline{C}_{in} + A\overline{B}\,\overline{C}_{in} + ABC_{in}$$

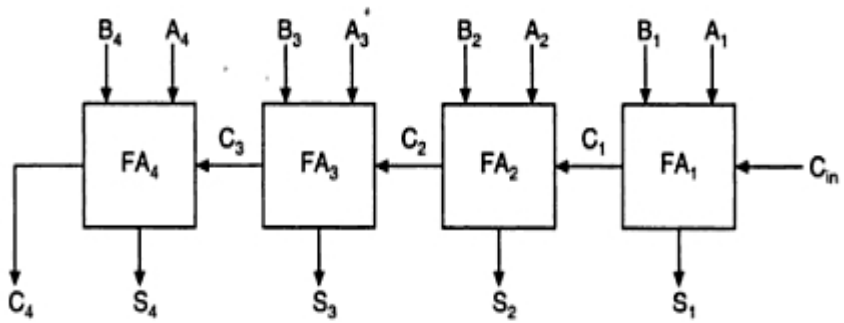
$$C_{out} = \overline{A}BC_{in} + A\overline{B}C_{in} + AB\overline{C}_{in} + ABC_{in}$$

$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AC_{in} + BC_{in} + AB$$

PARALLEL ADDER AND SUBTRACTOR

A binary parallel adder is a digital circuit that adds two binary numbers in parallel form and produces the arithmetic sum of those numbers in parallel form



parallel adder

DECODER

- A binary decoder is a combinational logic circuit that converts binary information from the **n** coded inputs to a maximum of 2^n unique outputs.
- We have following types of decoders 2x4,3x8,4x16….

2x4 decoder

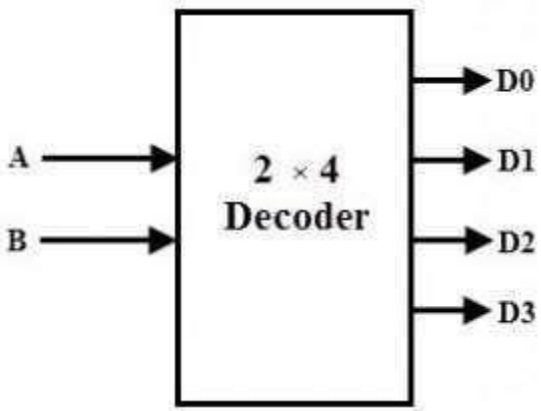


Fig 1: Block diagram

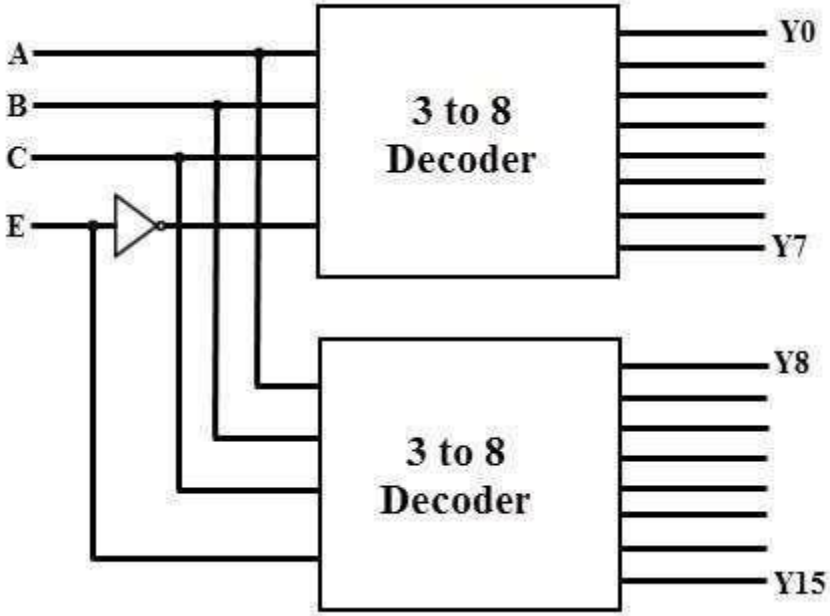
Inputs		Output			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
0	1	0	0	1	0
1	1	0	0	0	1

Fig 2:Truth table

DECODERS

Higher order decoder implementation using lower order.

Ex:4x16 decoder using 3x8 decoders



MULTIPLEXERS

- Multiplexer is a combinational circuit that has maximum of 2^n data inputs, 'n' selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines.
- We have different types of multiplexers 2x1,4x1,8x1,16x1,32x1……

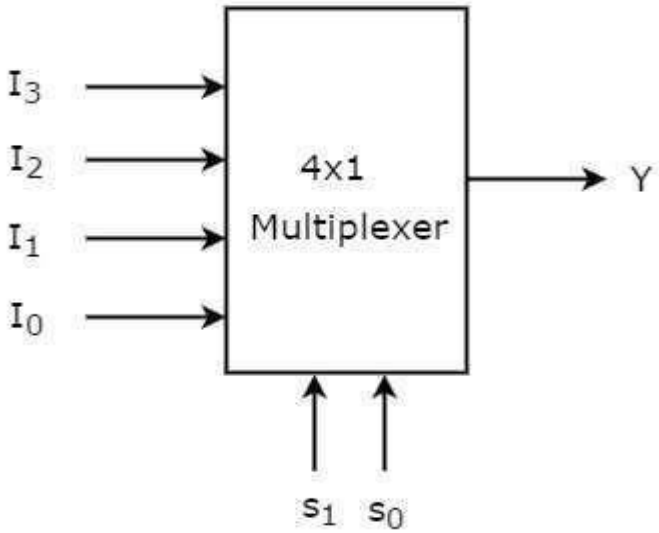


Fig 1: Block diagram

Selection Lines		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Fig 2: Truth table

MULTIPLEXERS

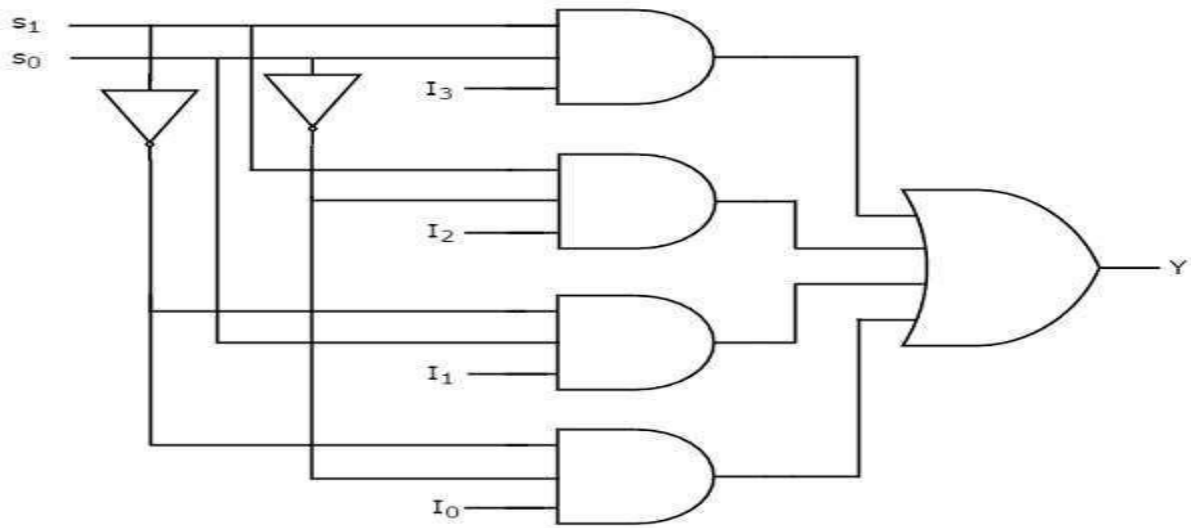


Fig 3: Logic diagram

SEQUENTIAL LOGIC CIRCUITS

Sequential logic circuit consists of a combinational circuit with storage elements connected as a feedback to combinational circuit

- output depends on the sequence of inputs (past and present)
- stores information (state) from past inputs

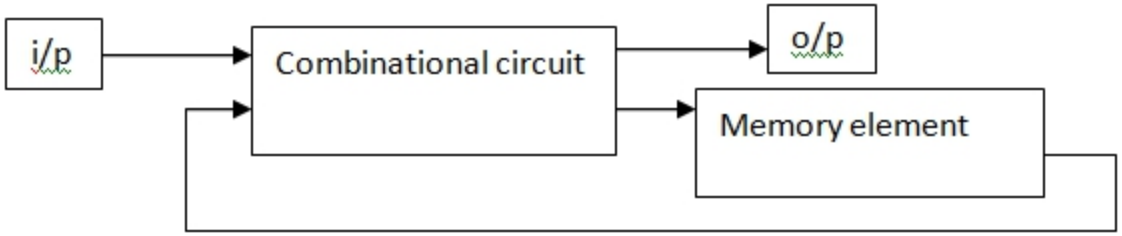
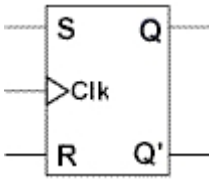


Figure 1: Sequential logic circuits

FLIPFLOPS:EXCITATION FUNCTIONS

SR Flip flop



FLIP-FLOPSYMBOL

S	R	$Q_{(next)}$
0	0	Q
0	1	0
1	0	1
1	1	?

CHARACTERISTIC TABLE

$$Q_{(next)} = S + R'Q$$

$$SR = 0$$

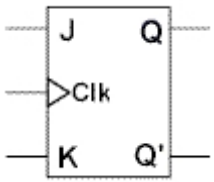
CHARACTERISTIC EQUATION

Q	$Q_{(next)}$	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

EXCITATION TABLE

FLIPFLOPS:EXCITATION FUNCTIONS

JK Flip flop



FLIP-FLOPSYMBOL

J	K	Q(next)
0	0	Q
0	1	0
1	0	1
1	1	Q'

CHARACTERISTIC TABLE

$$Q_{(next)} = JQ' + K'Q$$

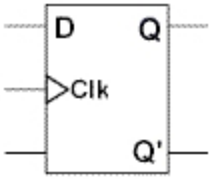
CHARACTERISTIC EQUATION

Q	Q(next)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

EXCITATION TABLE

FLIPFLOPS:EXCITATION FUNCTIONS

D Flip flop



FLIP-FLOPSYMBOL

D	Q(next)
0	0
1	1

CHARACTERISTIC TABLE

$$Q_{(next)} = D$$

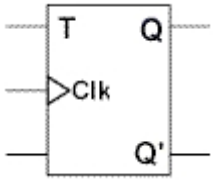
CHARACTERISTIC EQUATION

Q	Q(next)	D
0	0	0
0	1	1
1	0	0
1	1	1

EXCITATION TABLE

FLIPFLOPS:EXCITATION FUNCTIONS

T Flip flop



FLIP-FLOPSYMBOL

T	Q(next)
0	Q
1	Q'

CHARACTERISTIC TABLE

$$Q_{(next)} = TQ' + T'Q$$

CHARACTERISTIC EQUATION

Q	Q(next)	T
0	0	0
0	1	1
1	0	1
1	1	0

EXCITATION TABLE

CONVERTION OF ONE FLIP FLOP TO ANOTHER FLIP FLOP

CONVERTION OF SR FLIP FLOP TO JK FLIPFLOP

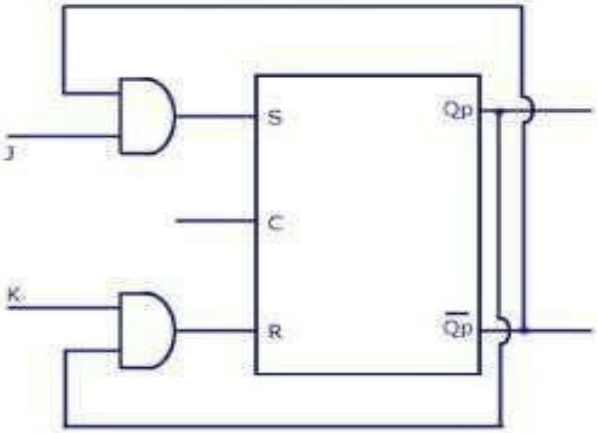
J and K will be given as external inputs to S and R. As shown in the logic diagram in next slide, S and R will be the outputs of the combinational circuit. The truth tables for the flip flop conversion are given . The present state is represented by Qp and Qp+1 is the next state to be obtained when the J and K inputs are applied. For two inputs J and K, there will be eight possible combinations. For each combination of J, K and Qp, the corresponding Qp+1 states are found. Qp+1 simply suggests the future values to be obtained by the JK flip flop after the value of Qp. The table is then completed by writing the values of S and R required to get each Qp+1 from the corresponding Qp. That is, the values of S and R that are required to change the state of the flip flop from Qp to Qp+1 are written.

CONVERTION OF ONE FLIP FLOP TO ANOTHER FLIP FLOP

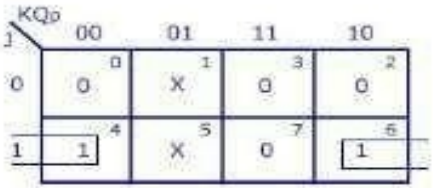
Conversion Table

J-K Inputs		Outputs		S-R Inputs	
J	K	Qp	Qp+1	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

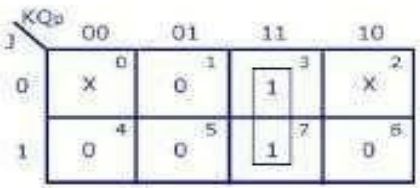
Logic Diagram



SR Flip Flop to JK Flip Flop



$S = \overline{J}Q_p$



$R = KQ_p$

K-Map

www.CircuitsToday.com

SR Flip Flop to JK Flip Flop

SHIFT REGISTERS

Introduction :

Shift registers are a type of sequential logic circuit, mainly for storage of digital data. They are a group of flip-flops connected in a chain so that the output from one flip-flop becomes the input of the next flip-flop. Most of the registers possess no characteristic internal sequence of states. All the flip-flops are driven by a common clock, and all are set or reset simultaneously. Shift registers are divided into four types.

1. SISO SR (serial in – serial out shift register)
2. SIPO SR (serial in – parallel out shift register)
3. PISO SR (Parallel in – serial out shift register)
4. PIPO SR (parallel in – parallel out shift register)

PROGRAMMABLE LOGIC DEVICES

Programmable Logic Array

- A programmable logic array (PLA) is a type of logic device that can be programmed to implement various kinds of combinational logic circuits.
- The device has a number of AND and OR gates which are linked together to give output or further combined with more gates or logic circuits.

PROGRAMMABLE LOGIC ARRAY

Programmable LogicArray

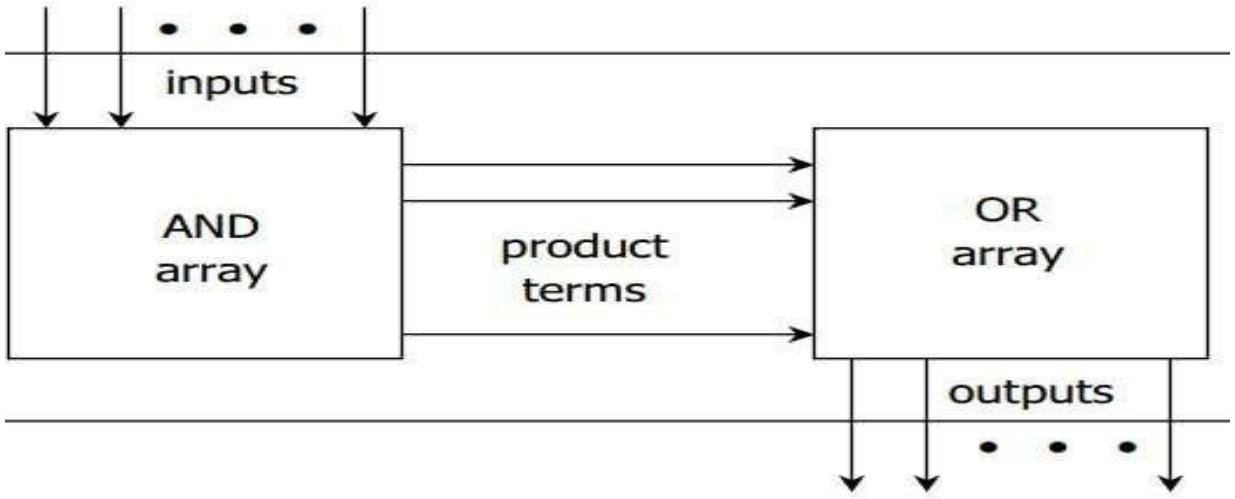


Fig 1: Block diagram of PLA

PROGRAMMABLE LOGIC ARRAY

PLA

F1 =AB’+AC+A’BC’

F2 = (AC+BC)’

PLA Programming Table

		Inputs			Outputs (T)	Outputs (C)
	Product Term	A	B	C	F_1	F_2
AB′	1	1	0	–	1	–
AC	2	1	–	1	1	1
BC	3	–	1	1	–	1
A′BC′	4	0	1	0	1	–

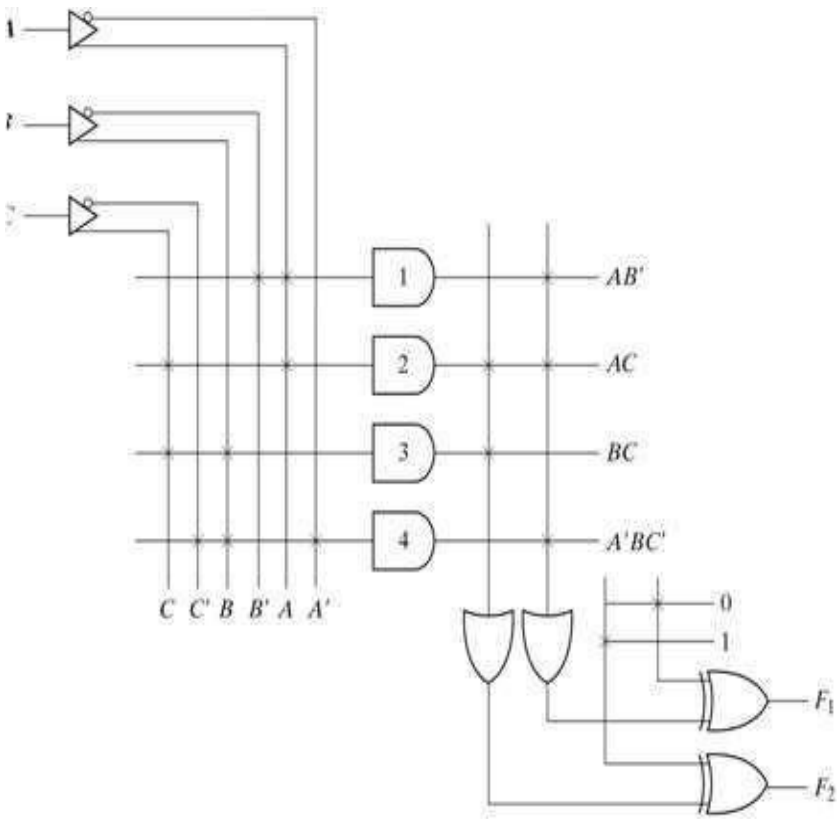


Fig 2: PLA with 3-inputs 4 product terms and 2 outputs

PROGRAMMABLE LOGIC ARRAY

Simplification of PLA

- Careful investigation must be undertaken in order to reduce the number of distinct product terms, PLA has a finite number of AND gates.
- Both the true and complement of each function should be simplified to see which one can be expressed with fewer product terms and which one provides product terms that are common to other functions.

PROGRAMMABLE LOGIC ARRAY

Example

Implement the following two Boolean functions with a PLA:

$$F_1(A, B, C) = \sum (0, 1, 2, 4)$$

$$F_2(A, B, C) = \sum (0, 5, 6, 7)$$

The two functions are simplified in the maps of given figure

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

$$F_1 = A'B' + A'C' + B'C'$$

$$F_1 = (AB + AC + BC)'$$

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

$$F_2 = AB + AC + A'B'C'$$

$$F_2 = (A'C + A'B + AB'C')'$$

PROGRAMMABLE LOGIC ARRAY

PLA table by simplifying the function

- Both the true and complement of the functions are simpl f products.
- We can find the same terms from the group terms of the f F_1', F_2 and F_2' which e minimum terms.

F1 = (AB +AC + BC)'

F2 = AB + AC + A'B'C'

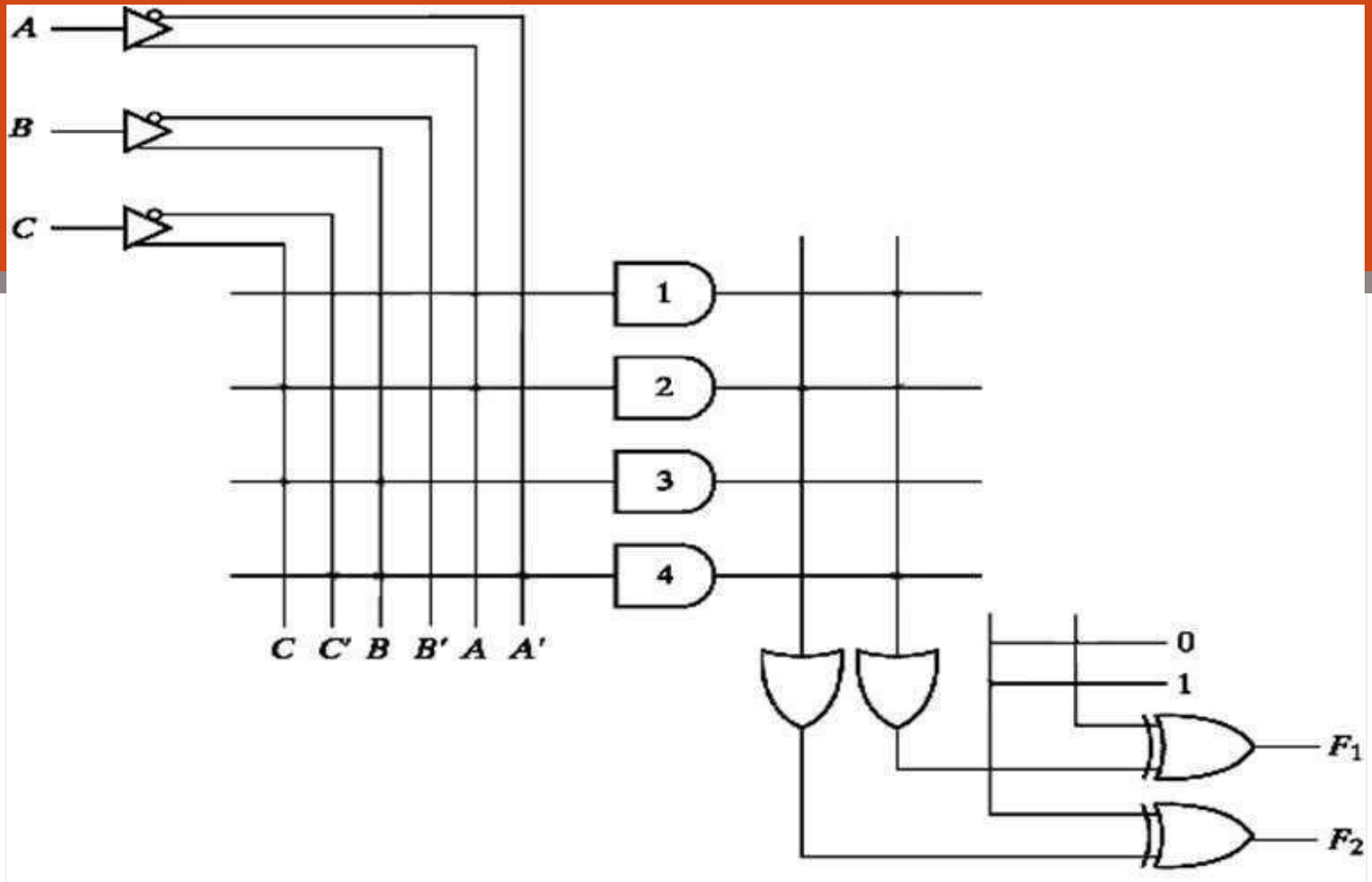
PLA programming table

	Product term	Inputs A	B	C	Outputs (C) F_1	(T) F_2
AB	1	1	1	–	1	1
AC	2	1	–	1	1	1
BC	3	–	1	1	1	–
A'B'C'	4	0	0	0	–	1

Fig 1: Solution to example

PROGRAMMABLE LOGIC ARRAY

PLA implementation



THANKYOU