Department of Computer Science and Engineering

RGUKT – SRIKAKULAM

COMPUTER ORGANIZATION AND ARCHITECTURE
LECTURE NOTES

Prepared by
Mr. Dileep Kumar Koda *M. Tech., (Ph. D)*
Asst. Professor
Computer Science & Engineering

Rajiv Gandhi University of Knowledge Technologies

Srikakulam

Catering to the Educational Needs of Gifted Rural Youth of Andhra Pradesh

(Established by the Govt. of Andhra Pradesh and recognized as per Section 2(f) of UGC Act, 1956)

COMPUTER ORGANIZATION AND ARCHITECTURE

Course code	Course name Course	Category L-T-P	Credits
20CS2201	COMPUTER ORGANIZATION AND ARCHITECTURE	PCC 3-0-0	3

Course Content

UNIT-I

Basic Functional blocks of a computer: CPU, Memory, Input -Output Subsystems, Control Unit. **Data Representation:** Number Systems, Signed Number Representation, Fixed and Floating Point Representations, Character Representation.

UNIT-II

ALU: Computer Integer Arithmetic: addition, subtraction, multiplication, division, floating point arithmetic: Addition, subtraction, multiplication, division.

Instruction set architecture of a CPU registers, instruction execution cycle, RTL interpretation of instructions, addressing modes, instruction set. RISC and CISC architecture. Case study instruction sets of some common CPUs.

UNIT-III

CPU control unit design: Introduction to CPU design, Processor Organization, Execution of Complete Execution, Design of Control Unit: hardwired and micro-programmed control, Case study design of a simple hypothetical CPU.

UNIT-IV

Memory system design: Concept of memory: Memory hierarchy, SRAM vs DRAM, Internal organization of memory chips, cache memory: Mapping functions, replacement algorithms, Memory management, virtual memory.

UNIT-V

Input -output subsystems, I/O transfers: programmed I/O, interrupt driven and DMA. I/O Buses, Peripheral devices and their characteristics, Disk Performance

UNIT-VI

Performance enhancement techniques: Pipelining: Basic concepts of pipelining, through put and speedup, pipeline hazards.

Parallel processing: Introduction to parallel processing, Introduction to Network, Cache coherence

Text Books:

1. V. C. Hamacher, Z. G. Vranesic and S. G. Zaky, “Computer Organization,” 5/e, McGraw Hill, 2002.
2. William Stallings, “Computer Organization and Architecture”: Designing for Performance, 8/e, Pearson Education India. 2010.
3. **Morris Mano**, “Computer System Architecture”, Pearson Education India, Third edition. References: A. S. Tanenbaum, “Structured Computer Organization”, 5/e, Prentice Hall of India, 2009.
4. D. A. Patterson and J. L. Hennessy, “Computer Organization and Design,” 4/e, Morgan Kaufmann, 2008.
5. J. L. Hennessy and D. A. Patterson,” Computer Architecture: A Quantitative Approach",4/e, Morgan Kaufmann, 2006.

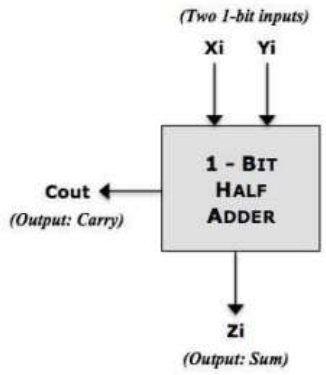
6. D. V. Hall, “Microprocessors and Interfacing”, 2/e, McGraw Hall, 2006 “8086 Assembler Tutorial for Beginners “By Prof. Emerson Giovani Carati.

UNIT – II

Integer Data Computation:

ONE BIT ADDITION: HALF ADDER

1) It is a simple 1-bit adder circuit.
2) It adds two 1-bit inputs Xi & Yi and produces a sum Zi and a Carry Cout.
3) As it does not consider any carry input, it can’t be combined to add large numbers.
4) Hence it is called a Half Adder.

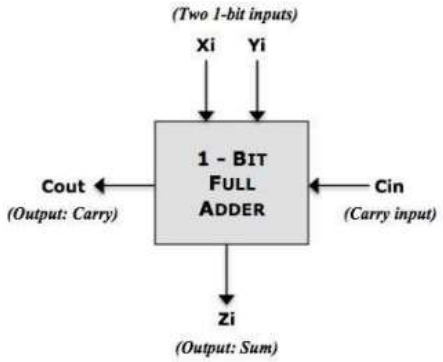


Inputs bits : X_i and Y_i.
Output (Sum) : Z_i
Output (Carry): Cout

Formula:
Sum (Zi) = X_i Ex-Or Y_i.
Carry (Cout) = $X_i.Y_i$

One-bit Addition: Full Adder

1. It is a 1-bit adder circuit.
2. It adds two 1-bit inputs Xi & Yi, along with a Carry Input Cin.
3. It produces a sum Zi and a Carry output Cout.
4. As it considers a carry input, it can be used in combination to add large numbers.
5. Hence it is called a Full Adder.



Inputs bits : X_i and Y_i.
Input Carry : C_{in}

Output (Sum) : Z_i
Output (Carry): C_{out}

Formula for Sum (Z_i):

$$Zi = Xi \div Yi \div Cin$$

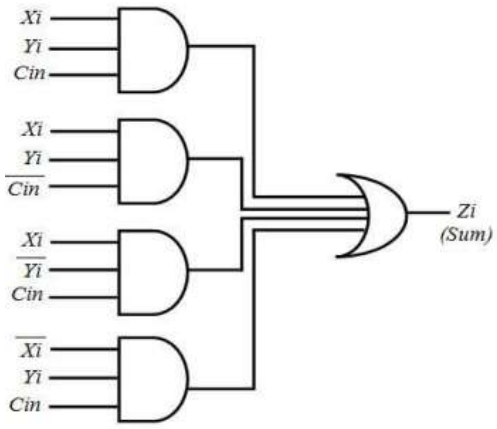
$$\therefore Zi = Xi \cdot Yi \cdot Cin + Xi \cdot Yi \cdot \overline{Cin} + Xi \cdot \overline{Yi} \cdot Cin + \overline{Xi} \cdot Yi \cdot Cin$$

Formula for Carry (C_{out}):

$$Cout = Xi \cdot Yi + Xi \cdot Cin + Yi \cdot Cin$$

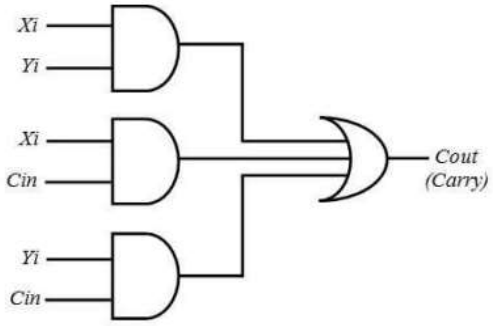
CIRCUIT FOR A FULL ADDER:

Circuit for Sum (Z):



Circuit for Carry (C_{out}):

Multiple Bit Addition: Serial Adder / Ripple Carry Adder:



1. A Full Adder can add two "1-bit" numbers with a Carry input.
2. It produces a "1-bit" Sum and a Carry output.
3. Combining many of these Full Adders, we can add multiple bits.
4. One such method is called Serial Adder.
5. Here, bits are added one-by-one from LSB.
6. The Carry of each stage is propagated (Rippled) into the next stage.
7. Hence, these adders are also called as Ripple Carry Adders.
8. Advantage: they are very easy to construct.
9. Drawback: As addition happens bit-by-bit, they are slow
10. Number of cycles needed for the addition is equal to the number of bits to be added

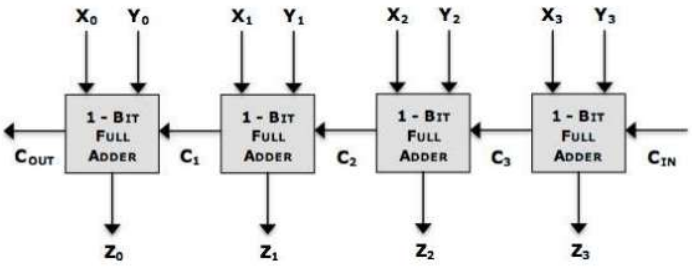
Inputs:
Assume X and Y are two "4-bit" numbers to be added, along with a Carry input C_{in}.

$X = X_0\ X_1\ X_2\ X_3$ (X_0 is the MSB ... X_3 is the LSB)
$Y = Y_0\ Y_1\ Y_2\ Y_3$ (Y_0 is the MSB ... Y_3 is the LSB)
C_{IN} = Carry Input

Outputs:
Assume Z to be a "4-bit" output, and COUT to be the output Carry

$Z = Z_0\ Z_1\ Z_2\ Z_3$ (Z_0 is the MSB ... Z_3 is the LSB)
C_{OUT} = Carry Output

Circuit for 4-bit Serial Adder/ Ripple Carry Adder:



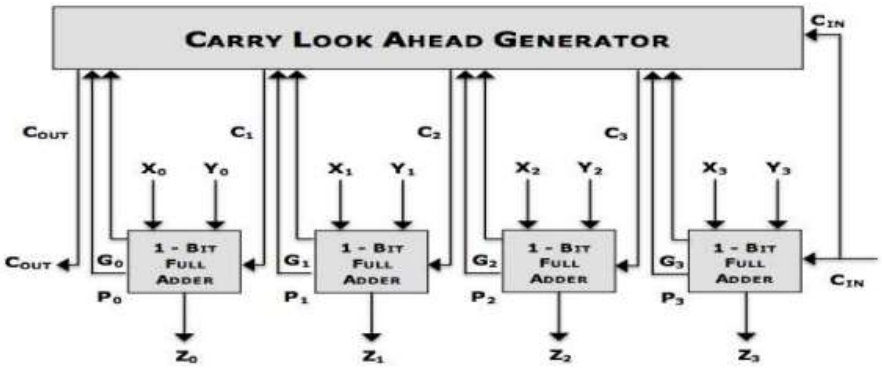
Multiple Bit Addition: Carry Look Ahead Adder / Parallel Adder:

1. It is used to add multiple bits simultaneously.
2. While adding multiple bits, the main issue is that of the intermediate carries.
3. In Serial Adders, we therefore added the bits one-by-one.
4. This allowed the carry at any stage to propagate to the next stage.
5. But this also made the process very slow.
6. If we "PREDICT" the intermediate carries, then all bits can be added simultaneously.
7. This is done by Carry Look Ahead generator.
8. Once all carries are determined beforehand, then all bits can be added simultaneously.
9. Advantage: This makes the addition process extremely fast.
10. Drawback: Circuit is Complex

Inputs:
Assume X and Y are two "4-bit" numbers to be added, along with a Carry input CIN.
$X = X_0\ X_1\ X_2\ X_3$ (X_0 is the MSB … X_3 is the LSB); $Y = Y_0\ Y_1\ Y_2\ Y_3$ & C_{IN} = Carry Input

Outputs:
Assume Z to be a "4-bit" output, and COUT to be the output Carry
$Z = Z_0\ Z_1\ Z_2\ Z_3$ & C_{OUT} = Carry Output

Circuit for 4-bit Carry Look Ahead Adder:



CALCULATIONS:
We can "Predict" (Look Ahead) all the intermediate carries in the following manner.

The Carry at any stage can be calculated as:

$C_i = X_i . Y_i + X_i . C_{in} + Y_i . C_{in}$
$C_i = X_i . Y_i + C_{in} (X_i + Y_i)$

$$\boxed{C_i = G_i + P_i . C_{IN}}$$

Here $Gi = X_i . Y_i$ ... (Generate)
And $Pi = X_i + Y_i$ ... (Propagate)

We need to predict the Carries: C_3, C_2, C_1 and C_0

$$C_3 = G_3 + P_3 C_{IN} \quad \text{... I}$$

$$C_2 = G_2 + P_2 C_3$$

Substituting the value of C3, we get:

$$C_2 = G_2 + P_2 G_3 + P_2 P_3 C_{IN} \quad \text{... II}$$

$$C_1 = G_1 + P_1 C_2$$

Substituting the value of C2, we get:

$$C_1 = G_1 + P_1 G_2 + P_1 P_2 G_3 + P_1 P_2 P_3 C_{IN} \quad \text{... III}$$

$$C_0 = G_0 + P_0 C_1$$

Substituting the value of C1, we get:

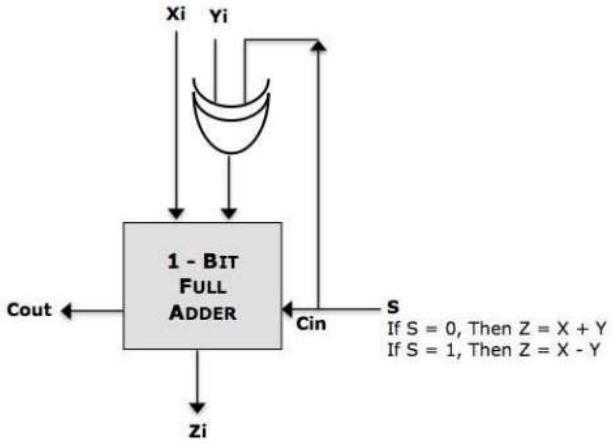
$$C_0 = G_0 + P_0 G_1 + P_0 P_1 G_2 + P_0 P_1 P_2 G_3 + P_0 P_1 P_2 P_3 C_{IN} \quad \text{... IV}$$

From the above four equations, it is clear that the values of all the four Carries (C3, C2, C1, C0) can be determined beforehand even without doing the respective additions. To do this we need the values of all G's (Xi.Yi) and all P's (Xi+Yi) and the original carry input CIN. This is done by the Carry Look Ahead Generator Circuit.

ADDER / SUBTRACTOR CIRCUIT:

1. Subtraction in binary numbers is simply performed by addition of two's complement.
2. That means, a special circuit for subtraction is not needed.
3. The same circuit that is used for Addition, can also be used for subtraction.
4. The following circuit is called Adder/ Subtractor circuit.
5. It can perform Addition as Z = X + Y.
6. It can also perform subtraction as Z = X + (2's Complement of Y)
7. The Variable "S" determines if Addition or Subtraction will be performed.
8. If S = 0, then Addition will be performed.
9. If S = 1, then Subtraction will be performed.

10. If S = 1, then the operation is Z = X + (1's Complement of Y) + 1. Hence Z = X – Y.



Unsigned Multiplication: Conventional Method / Pencil–Paper Method:

1. The Conventional (Pencil-Paper) method is used to multiply two unsigned numbers.
2. When we multiply two "N-bit" numbers, the answer is "2 x N" bits.
3. Three registers A, Q and M, are used for this process.
4. Q contains the Multiplier and M contains the Multiplicand.
5. A (Accumulator) is initialized with 0.
6. At the end of the operation, the Result will be stored in (A & Q) combined.
7. The process involves addition and shifting.

Algorithm

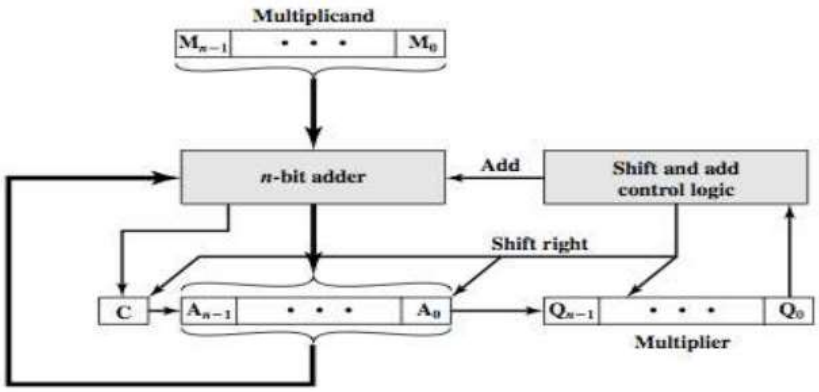
The number of steps required is equal to the number of bits in the multiplier.

1. At each step, examine the current multiplier bit starting from the LSB.
2. If the current multiplier bit is "1", then the Partial-Product is the Multiplicand itself.
3. If the current multiplier bit is "0", then the Partial-Product is the Zero.
4. At each step, ADD the Partial-Product to the Accumulator.
5. Now Right-Shift the Result produced so far (A & Q combined).

Repeat steps 1 to 5 for all bits of the multiplier.

The final answer will be in A & Q combined.

Circuit Diagram for Unsigned Multiplication:



Example: 7 x 6 = 42

				0	1	1	1	… **Multiplicand (7)**
		x		0	1	1	0	… **Multiplier (6)**
				0	0	0	0	… Partial-Product
			0	1	1	1	x	”
		0	1	1	1	x	x	”
+	0	0	0	0	x	x	x	”
	0	1	0	1	0	1	0	… **Result (42)**

SIGNED MULTIPLICATION: BOOTH'S ALGORITHM

1. Booth's Algorithm is used to multiply two SIGNED numbers.
2. When we multiply two "N-bit" numbers, the answer is "2 x N" bits.
3. Three registers A, Q and M, are used for this process.
4. Q contains the Multiplier and M contains the Multiplicand.
5. A (Accumulator) is initialized with 0.
6. At the end of the operation, the Result will be stored in (A & Q) combined.
7. The process involves addition, subtraction and shifting.

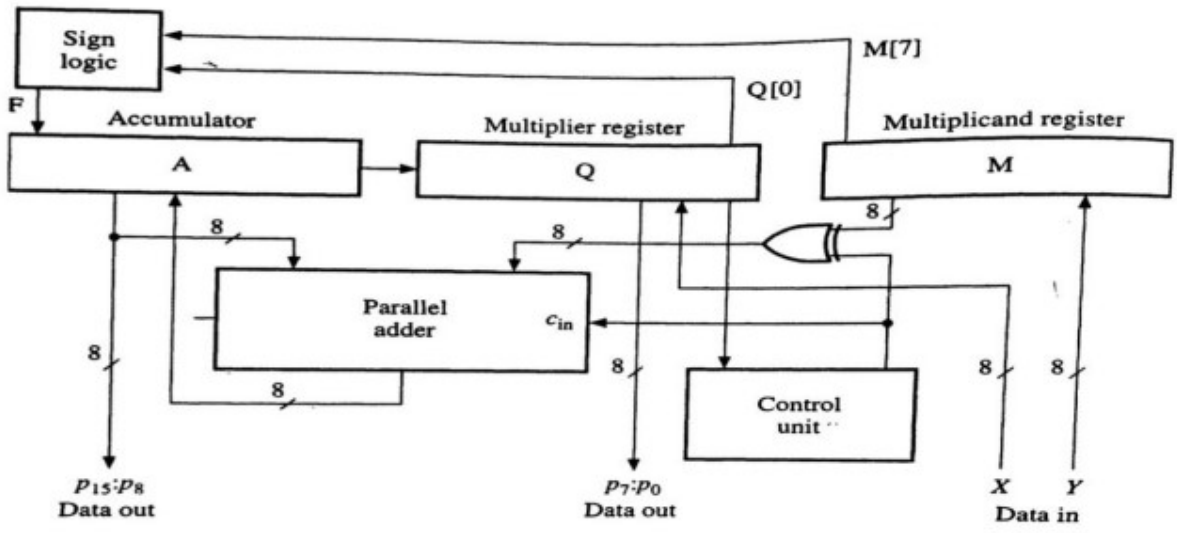
Algorithm:

The number of steps required is equal to the number of bits in the multiplier. At the beginning, consider an imaginary "0" beyond LSB of Multiplier

1. At each step, examine two adjacent Multiplier bits from Right to Left.
2. If the transition is from "0 to 1" then Subtract M from A and Right-Shift (A & Q) combined.
3. If the transition is from "1 to 0" then ADD M to A and Right-Shift.
4. If the transition is from "0 to 0" then simply Right-Shift.
5. If the transition is from "1 to 1" then simply Right-Shift. Repeat steps 1 to 5 for all bits of the multiplier.

The final answer will be in A & Q combined.

Circuit Diagram for Signed Multiplication using Booth's Algorithm


Sign logic
M[7]
Q[0]
F
Accumulator
Multiplier register
Multiplicand register
A
Q
M
8
8
8
Parallel adder
c_{in}
8
8
8
8
8
Control unit
p_{15}:p_8
Data out
p_7:p_0
Data out
X
Y
Data in