UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Computer Organization & Architecture** Subject Code: **MCA101**
Year:1st **Year** Semester: **First**

Module Number	Topics	Number of Lectures
1	**Introduction to digital computer organization**	**7L**
	1. Concept of basic components of a digital computer, High level view of computer. 2. Commonly used number systems 3. Fixed and floating point representation of a number 4. Booth's Algorithm – restoring and non restoring.	6L
	5. What are the H/W resources that we will need in computer 6. Conversion of high level code to m/c level language.	1L
2	**CPU design**	**3L**
	1. Basic organization of the stored program computer and operation sequence for execution of a program. 2. Description of ALU, Design of circuit of a small scale CPU	2L
	3. Fetch, decode and execute cycle, Concept of operator, operand, registers and storage, Instruction format. Instruction sets and addressing modes.	1L
3	**CPU design - Timing and control**	**2L**
	1. Timing diagram and control design	2L
4	**Micro programmed control**	**5L**
	1. Concepts of Micro operations 2. Horizontal and vertical micro-program 3. Optimization of hardware resources for designing of micro-programmed control unit	5L
5	**Pipeline concept**	**3L**
	1. Instruction and Arithmetic Pipelining, 2. Synchronous and Asynchronous pipeline 3. Solving problems on pipeline speed-up, efficiency, throughput	2L
6	**Memory Organization**	**5L**
	1. Static and dynamic memory, Memory hierarchy, Associative memory, Bare machine	3L
	2. Memory unit design with special emphasis on implementation of CPU-memory interfacing.	2L

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Computer Organization** Subject Code: **CS304**
Year: **2nd Year** Semester: **Third**

7	**Cache memory Architecture**	**4L**
	1. Cache memory organizations, Set associative cache	1L
	2. Techniques for reducing cache misses	1L
	3. Discussion on Buffer cache	2L
8	**RAM architecture**	**2L**
	1. Basic concepts of architecture of RAM	2L
9	**Discussion on DRAM & SRAM**	**3L**
	1. Architecture of Static Ram and Dynamic Ram	2L
	2. Difference between SRAM and DRAM	1L
10	**I-O subsystem organization**	**3L**
	1. Concept of handshaking, Polled I/O	1L
	2. Interrupt and DMA	2L
Total Number Of Hours = 37L		

Faculty In-Charge HOD, CSE Dept.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Computer Organization & Architecture** Subject Code: **MCA101**
Year:1st **Year** Semester: **First**

Assignments:-

Unit 1:-

1. What is the role of operating system?
2. Discuss the basic organization of digital computer.
3. Convert (1B3F)16 → (?)8
4. Why do we need 2's compliment method?
5. Briefly explain IEEE 754 standard format for floating point representation in single precision.
6. Using Booth's algorithm multiply (– 12) and (+ 6).
7. Explain various assembler directives used in assembly language program.
8. Write +710 in IEEE 754 floating point representation in double precision.

Unit 2 + Unit 3 + Unit 4:-

1. What are the advantages of micro programming control over hardwired control?
2. Write down the control sequence for Move (R1), R2.
3. Define hardwired control.
4. Discuss the principle of operation of a micro programmed control.
5. Write the control sequence for execution of the instruction Add(R3), R1.
6. Give the organization of typical hardwired control unit and explain the functions performed by the various blocks.
7. Explain the multiple bus organization in detail.
8. What is microprogrammed sequencer?
9. Design a basic ALU which can perform 8 different arithmetic and logical operations.
10. Design the basic block diagram of Intel microprocessor 8085 and discuss the working principle of the different parts of it.

Unit 5 :-

1. What is pipelining?
2. What are the major characteristics of a pipeline?
3. What is a pipeline hazard?
4. What is the use of pipelining?
5. What are the remedies commonly adopted to overcome/minimize these hazards.
6. What is the ideal speedup expected in a pipelined architecture with n stages. Justify your answer.
7. Draw the structure of two stage instruction pipeline.

Unit 6 + Unit 7 + Unit 8 + Unit 9 :-

1. Define Memory Access time for a computer system with two levels of caches.
2. How to construct an 8M * 32 memory using 512 K * 8 memory chips.
3. Write two advantages of MOS device.

4. List the factors that determine the storage device performance.
5. What will be the width of address and data buses for a 512K * 8 memory chip?
6. Define memory cycle time.
7. What is RAM?
8. What is cache memory?
9. Explain virtual memory.
10. List the various semiconductors RAMs?
11. Define DRAM's.
12. What is ROM?
13. Give the format for main memory address using direct mapping function for 4096 blocks in main memory and 128 blocks in cache with 16 blocks per cache.
14. Give the format for main memory address using associative mapping function for 4096 blocks in main memory and 128 blocks in cache with 16 blocks per cache.
15. Give the format for main memory address using set associative mapping function for 4096 blocks in main memory and 128 blocks in cache with 16 blocks per cache.
16. Define Hit and Miss rate?
17. What are the enhancements used in the memory management?
18. Define latency time.
19. A computer system has a main memory consisting of 16 M words. It also has a 32Kword cache organized in the block-set-associative manner, with 4 blocks per set and 128 words per block.
20. How will the main memory address look like for a fully associative mapped cache?
21. A digital computer has a memory unit of 64K*16 and a cache memory of 1K words. The cache uses direct mapping with a block size of four words. How many bits are there in the tag, index, block and word fields of the address format? How many blocks can the caches accommodate?
22. Define the terms "spatial locality" and "temporal locality", and explain how caches are used to exploit them for a performance benefit. Be specific in the different ways that caches exploit these two phenomena.
23. Suppose physical addresses are 32 bits wide. Suppose there is a cache containing 256K words of data (not including tag bits), and each cache block contains 4 words. For each of the following cache configurations,
    a. direct mapped
    b. 2-way set associative
    c. 4-way set associative
    d. fully associative
    specify how the 32-bit address would be partitioned. For example, for a direct mapped cache, you would need to specify which bits are used to select the cache entry and which bits are used to compare against the tag stored in the cache entry.
24. Cache misses can be characterized as one of the following: compulsory misses, capacity misses, and conflict misses. Describe how each of these kinds of misses can be addressed in the hardware.
25. Why virtual memory is called virtual ? What are the different address spaces ? Explain with example how logical address is converted into physical address and also explain how page

replacements take place. Explain the instruction cycle with a neat diagram. Explain the disadvantages of stored program computer.

26. A three-level memory system having cache access time of 5 nsec and disk access time of 40 nsec, has a cache hit ratio of 0.96 and main memory hit ratio of 0.9. What should be the main memory access time to achieve an overall access time of 16 nsec ?
27. According to the following information, determine size of the subfields (in bits) in the address for Direct Mapping and Set Associative Mapping cache schemes :
    We have 256 MB main memory and 1 MB cache memory
    The address space of the processor is 256 MB
    The block size is 128 bytes
    There are 8 blocks in a cache set.

Unit 10 :-

1. What are the functions of I/O interface?
2. How does the processor handle an interrupt request?
3. What are the necessary operations needed to start an I/O operation using DMA?
4. What is the advantage of using interrupt initiated data transfer?
5. Why do you need DMA?
6. What is the difference between subroutine and interrupt service routine?
7. Why I/O devices cannot be directly be connected to the system bus?
8. What is polling?
9. State the differences between memory mapped I/O and I/O mapped I/O.
10. Explain the functions to be performed by a typical I/O interface with a typical input output interface.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Computer P**rogramming** with C Subject Code: **MCA102**
Year: **1st Year** Semester: FIRST

Module Number	Topics	Number of Lectures
1	**Introduction to Computers**	**2**
	1. Generations, Classifications, Applications, Basic Organization, Input and output devices	1
	2. Basic concept of Computer memory, Computer software and networks	1
2	**Number system**	**4**
	1. Decimal, Binary, Octal, Hexa-decimal, Conversion of numbers, Addition and subtraction of two numbers, Two's compliment	2
	2. Multiplication and division of binary numbers, Working with fractions, signed number representation in binary form	1
	3. Logic gates	1
3	**Introduction to C**	**4**
	1. compiling and executing C programs, using comments, keywords, identifiers, Data type, variables, constants	1
	2. input/output statements in C, operators in C	2
	3. type conversion and type casting.	1
4	**Decision Control and looping statements**	**6**
	1. conditional branching statement	2
	2. iterative statements	2
	3. nested loops, break and continue statements, goto statement	2
5	**Arrays & Strings**	**6**
	1. Declaration, accessing elements of array, storing values	1
	2. calculating the length of array, two dimensional arrays	1
	3. reading and writing strings, suppressing input, string taxonomy	1
	4. string operations – using and without using library function	2
	5. array of strings	1
6	**Functions**	**5**
	1. Declaration, prototype, definition, function call	1
	2. return statement, passing parameters to the function	1
	3. scope of variable, storage classes	1
	4. recursive functions	2
7	**Pointers**	**7**
	1. introduction, declaration, Pointer expression and arithmetic	1
	2. null pointer, generic pointer, passing arguments to functions using pointer	1
	3. pointers and arrays, passing an array to function, difference between array name and pointer	2
	4. pointers and strings, array of pointers	1
	5. function pointers, pointers to pointers	1
	6. dynamic memory allocation, drawbacks	1

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Basic computation and principles of computer programming**
Subject Code: **CS101**
Year: **1st Year** Semester: 1st

8	**Structure-union, Files, Preprocessor directives**	**4**
	1. Structure, nested structure, array of structure 2. union, array of union variable, unions inside structure 3. Files – Reading –writing etc 4. Preprocessor directives	1 1 1 1
Total Lecture Hours – 37 l.h.		

Faculty In-Charge HOD, CSE Dept.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Computer P**rogramming** with C Subject Code: **MCA102**
Year: **1st Year** Semester: FIRST
Assignments :

Unit 1:- (Introduction to Computers)

1. How is OCR technology better than ordinary image scanner?
2. How does MICR technology help to detect fraud in check payment?
3. Which factors will you consider while purchasing a monitor for your personal computer?
4. What is a head crash? How does it occure?
5. What is USB flash drive?
6. What is a BIOS? Which kind of memory is preferred in it, and why?
7. How is application software is different from system software?
8. Classify the operating systems based on their capabilities.
9. In what situation must the network have a gateway?
10. Which device will you prefer to form a network – hub or a switch? Justify your answer.

Unit 2 :- (Number system)

1. How can two numbers be subtracted only through addition? Explain with example.
2. Convert the following –
   a) (10110101)2 – (?)10
   b) (5674)8 – (?)10
   c) (A1E2)16 – (?)10
   d) (289)10 – (?)8
   e) (593)10 – (?)2
   f) (36541)8 – (?)16
3. 10101011 – 01111 = ?
4. 11100001 + 11111110111 = ?
5. 1111000 / 100 = ?
6. Why NAND gate is known as universal gate?

Unit 3 :- (Introduction to C)

1. State true or false – a> $amount is a valid identifier in C,
   b> The equality operators have higher precedence than the relational operators.
   c> Signed variables can increase the maximum positive range.
   d> printf("%d", scanf("%d",&num)); is valid statement
   e> The modulus operators can be used only with integers
2. Write a program to prepare a grocery bill. For that enter the name of the items purchased, quantity in which it is purchased, and its price per unit. Then display the bill in the following format

```
****************************BILL********************************
  Item            Quantity              Price              Amount

_________________________________________________________________
Total amount to be paid
_________________________________________________________________
```

3. Write a program to read two floating point numbers. Add these numbers and assign the result to an integer. Finally display the value of all the three variables.
4. Find the error(s) –
   ```
   Int n;
   float a b;
   double = a, b;
   complex a b;
   a,b : INTEGER
   long int a;b;
   ```
5. Find the error(s) –
   ```
   int a = 9;
   float y =2.0;
   a = b% a;
   printf("%d", a);
   ```
6. Write the output of the code →
   ```
   int main()
   {
   int a = 2,b =3, c = 4;
   a=b=c;
   printf("a=%d",a);
   return 0;
   }
   ```
7. Evaluate the expression – (x > y) + ++a || !c
8. Write a program to count the number of vowel in a given text.
9. Why do we include <stdio.h> in our program?
10. Write a program to calculate simple and compound interest .

Unit 4 :- (Decision Control and looping statements)

1. Change the following for loop into do-while loop
   ```
   int i;
   for(i=10 ; i>0 ; i--)
   printf("%d", i);
   ```
2. WAP to accept any number and print the number of digits in that program.
3. WAP to print the sum of all odd numbers from 1 to 100.
4. WAP that displays all the numbers from 1 to 100 that are not divisible by 2 as well as by 3.
5. Write down the output
   ```
   #include<stdio.h>
   main()
   { int num = 10;
    for(;;--num)
   ```

```
printf("%d", num);
}
```

6. Find errors

```
#include<stdio.h>
main() {
int i,j;
for (i=1,j=0 ; i+j <=10 ; i++)
printf("%d", i);
j+=2;
}
```

7. WAP to print the pattern →

```
a> $ * * * *
   *$  * * *
   * * $ * *
   * * * $ *
   * * * * $
b>  @
    @ @
    @ @ @
    @ @ @ @
    @ @ @ @ @
    @ @ @ @
    @ @ @
    @ @
    @
```

8. WAP to print the sum of the following series →
   a> 1 + (1+2) – (1+2+3) + (1+2+3+4) - ....
   b> $-x + x^2 - x^3 + x^4$ .....

Unit 5 :- (Arrays)

1. How are multi dimensional arrays useful?
2. What happens when an array is initialized with (a) fewer initializers as compared to its size? (b) more initializers as compared to its size?
3. For an array declared as int arr[50], calculate the address of arr[35], if base(arr) = 1000 and w =2.
4. Write a program that reads a square matrix of size n x n. Write a function int isUpperTraingular(int[][],int n) that returns 1 if the matrix is upper triangular.
5. WAP to read two floating point arrays. Merge these arrays and display the resultant array.
6. WAP to display the word HELLO in the following format

```
H
H E
H E L
H E L L
H E L L O
```

7. Wap to count the number of charecters, words and lines in the given text.
8. WAP to find the last instance of occurrence of a given string.
9. Write a program to display a list of candidates. Prompt 100 users to cast their vote. Finally display the winner in the election.
10. In a class there are 20 students. Each student is supposed to appear in three tests and two quizzes throughout the year. Make an array that stores the names of all these 20 students. Make five arrays that stores marks of three subjects as well as scores of two quizzes for all the students. Calculate the average and total marks of each student. Display the result.

Unit 6 :- (Functions)

1. How many types of storage classes does the C language support? Why do we need different types of such classes?
2. What is the difference between formal and actual parameters?
3. Write a function to reverse a string using recursion.
4. What will happen when the actual parameters are less than the formal parameter in a function?
5. WAP to compute F(x , y) where F(x , y) = F(x – y , y) + 1 if y <= x and F(x , y) = 0 if x < y.
6. Write a function to draw the following pattern on the screen

```
* * * * * * * *
!             !
!             !
!             !
* * * * * * * *
```

7. Write a function to print a table of bionomial coefficients which is given by the formula

   B(m , x) = m ! / (x ! (m – x)!) where m > x ,

   hint : B (m , 0) = 1 , B (0 ,0) = 1, and B (m, x) = B (m, x-1) * [(m – x + 1) / x]

8. Write a program to swap two integers using Call by Value method of passing arguments to a function.
9. Find the output –

```
 #include<stdio.h>
 int prod (int x, int y)

{
 return (x * y);
}
main()
{
```

```
int x = 2, y = 3, z;
z = prod (x, prod(x,y) );
printf("%d",z);
return 0;
}
```

10. Find the output
```
#include <stdio.h>
int a;
static int func()
{
return  a++;
}
main()
{
a = 10;
printf("%d", func() );
a*= 10;
printf("%d", func() );
return 0;
}
```

Unit 7 :- (Pointers)

1. Explain the result of the following code
```
int num1 = 2, num2 =3;
int *p = &num1, *q = &num2;
*p++ =*q++;
```
2. What do you understand by null pointer? Discuss with example.
3. Differentiate between ptr++ and *ptr++.
4. Can we subtract two pointer variables?
5. Can array names appear on the left side of the assignment operator? Why ?
6. WAP to print the lowercase letters into uppercase characters and the vice versa in the given string – "HappY NeW YeaR.".
7. What is a dangling pointer?
8. What do you mean by wild pointer?
9. With the help of an example, explain how pointers can be used to dynamically allocate space for 2 D and 3D arrays.
10. State true or false—
    a. Only one call to free() is enough to release the entire array allocated using calloc().
    b. Ragged arrays consumes less memory space.
    c. *ptr++ will add 1 to the value pointed by ptr.
    d. Pointer constants cannot be changed.

e. Adding 1 to a pointer variable will make it point 1 byte ahead of the memory location to which it is currently pointing.

Unit 8 :- (Structure, union, Files, Preprocessor directives)

1. What do you mean by nested structure? Discuss.
2. Differentiate between structure and union.
3. WAP to create a structure with information given below. Then read and print the data.
   Employee[10]
   (a) Emp_id
   (b) Name
      (i) First name
      (ii) Middle Name
      (iii) Last Name
   (c) Address
      (i) Area
      (ii) City
      (iii) State
   (d) Age
   (e) Salary
   (f) Designation.
4. WAP to read data from the keyboard and write it to a file. Read the contents stored in the file and display it on the screen.
5. WAP to copy a file using feof().
6. Can we have a C program that does not use any pre-processor directive?
7. What happens when the argument passed to the macro has multiple white space characters?

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Business Systems & Applications **Subject Code-MCA103**
Year: 1st Year **Semester: First**

Module Number	Topics	Number of Lectures
1	**Introduction:**	**5L**
	Use of computers for managerial applications, Technology issues and data processing in organisations, Introduction to Information Systems, shift in Information system thinking, latest trends in Information Technology.	5
2	**Information System:**	**4L**
	Computer Based Information Systems- office automation systems. Decision making and MIS, transaction processing systems.	4
3	**Decision Support System:**	**4L**
	Decision support system, Group Decision Support, Executive Information systems, DSS generator.	4
4	**Modern computation:**	**4L**
	Introduction to: Artificial Intelligence Based Systems, End user computing, Distributed data processing.	4
5	**IS architecture:**	**4L**
	Deciding on IS architecture, IT leadership & IS strategic planning.	4
6	**IS strategy:**	**3L**
	Introduction to: IS strategy and effects of IT on competition.	3
7	**ERP:**	**5L**
	Introduction to: ERP, re-engineering work processes for IT applications, Business Process Redesign.	5
8	**Knowledge Based:**	**3L**
	Knowledge engineering and data warehouse.	3
Total Number Of Hours = 32		

Assignment:

Module-1:
1. Describe the use of computer in managerial applications.

Module-2:
1. Explain Management Information System with examples.

Module-3:
1. Discuss Decision support system.

Module-4:
1. What do you mean by artificial intelligence (AI)? How AI involves in business?

Module-5:

1. Describe IS architecture with proper diagram.

Module-6:

1. Discuss about various IS strategies.

Module-7:

1. Define Enterprise resource planning. What do you mean by re-engineering?

Module-8:

1. Define Data warehouse? How we can Mine data?

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Discrete Mathematical Structure **Subject Code-M101**
Year: 1STYear **Semester: FIRST**

Module Number	Topics	Number of Lectures
1.	**Introduction to Propositional Calculus:**	**10L**
	Propositions, Logical Connectives, Conjunction, Disjunction, Negation and their truth table	2
	Conditional Connectives, Implication, Converse, Contrapositive, Inverse, Biconditional statements with truth table	2
	Logical. Equivalence, Tautology,Normal forms-CNF, DNF; Predicates and Logical Quantifications of Propositions and related examples.	6
2.	**Theory of Numbers:**	**10L**
	Well Ordering Principle, Divisibility theory and properties of divisibility	2
	Fundamental theorem of Arithmetic; EuclideanAlgorithm for finding G.C.D and some basic properties of G.C.D with simple examples	2
	Order, Relation and Lattices: POSET, Hasse Diagram, Minimal , Maximal, Greatest and Least elements in a POSET, Lattices and its properties, Principle of Duality, Distributive and Complemented Lattices	6
3.	**Counting Techniques:**	**10L**
	Permutations, Combinations, Binomial coefficients, Pigeon- hole Principle	2
	Principles of inclusion and exclusions; Generating functions, Recurrence Relations and their solutions using generating function, Recurrence relation of Fibonacci numbers and it's solution	4
	Divide-and-Conquer algorithm and its recurrence relation and its simple application in computer	4
4.	**Graph Coloring:**	**6L**
	Chromatic Numbers and its bounds, Independence and Clique Numbers	1
	Perfect Graphs-Definition and examples, Chromatic polynomial and its determination, Applications of Graph Coloring.	2
	Matchings: Definitions and Examples of Perfect Matching, Maximal and Maximum Matching, Hall's Marriage Theorem (Statement only) and related problems	3

Assignment:
Module-1:

1. Represent as propositional expressions:
   Tom is a math major but not computer science major
   P: Tom is a math major
   Q: Tom is a computer science major

Use De Morgan's Laws to write the negation of the expression, and translate the negation in English

2. Let
   P = "John is healthy"
   Q = "John is wealthy"
   R = "John is wise"
   Represent:
   John is healthy and wealthy but not wise: $P \wedge Q \wedge \neg R$
   John is not wealthy but he is healthy and wise: $\neg Q \wedge P \wedge R$
   John is neither healthy nor wealthy nor wise: $\neg P \wedge \neg Q \wedge \neg R$

   .
3. Translate the sentences into propositional expressions:
   "Neither the fox nor the lynx can catch the hare if the hare is alert and quick."
4. Given a conditional statement in English,
   i. translate the sentence into a logical expression
   ii. write the negation of the logical expression and translate the negation into English
   iii. write the converse of the logical expression and translate the converse into English

Module-2:

1. Prove that $\mathbf{3^n > n^2}$ for n = 1, n = 2 and use the mathematical induction to prove that $3^n > n^2$ for n a positive integer greater than 2.
2. Prove that for any positive integer number n , $n^3 + 2n$ is divisible by 3

3. Use mathematical induction to prove that

   $1^3 + 2^3 + 3^3 + ... + n^3 = n^2 (n + 1)^2 / 4$

   for all positive integers n.
4. In a room of 50 people whose dresses have either red or white color, 30 are wearing red dress, 16 are wearing a combination of red and white. How many are wearing dresses that have only white color?

Module-3:

1. Out of 7 consonants and 4 vowels, how many words of 3 consonants and 2 vowels can be formed?
2. In a group of 6 boys and 4 girls, four children are to be selected. In how many different ways can they be selected such that at least one boy should be there?
3. From a group of 7 men and 6 women, five persons are to be selected to form a committee so that at least 3 men are there on the committee. In how many ways can it be done?
4. In how many different ways can the letters of the word 'OPTICAL' be arranged so that the vowels always come together?
5. In how many different ways can the letters of the word 'CORPORATION' be arranged so that the vowels always come together?

Module-4:

1. Prove that any planar graph has an edge coloring of at most three colors in which adjacent edges of the same color are allowed but cycles of edges of the same color are not.
2. If G is a graph and H is any subgraph of G, $\chi(G) \geq \chi(H)$.
3. State and prove four coloring theorem.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Course Description

Title of Course: Business English & Communication
Course Code: HU101
L-T Scheme: 3-1 **Course Credits: 4**

Introduction:

This course can enhance the drafting and understanding skills of engineering students.

Objectives:

1. This Course has been designed to impart advanced skills of Technical Communication in English through Language Lab. Practice Sessions to 1STSemester UG students of Engineering &Technology.
2. To enable them to communicate confidently and competently in English Language in all spheres.

Learning Outcomes:

Knowledge:

1. This course will help the students to learn English very easily. Even the Hindi medium students can translates easily.
2. The technical communication will help the students to improve their speaking skills and drafting skill for engineering students.

Course Contents:

Unit 1: ENGLISH LANGUAGE GRAMMAR-Correction of Errors in Sentences Building Vocabulary Word formation Single Word for a group of Words Fill in the blanks using correct Words Sentence Structures and Transformation Active &Passive Voice Direct &Indirect Narration (MCQ Practice during classes).

Unit 2: READING COMPREHENSION-Strategies for Reading Comprehension Practicing Technical & Non Technical Texts for Global/Local/Inferential/Referential comprehension; Précis Writing

Unit 3: TECHNICAL COMMUNICATION-the Theory of Communication–Definition & Scope Barriers of Communication Different Communication Models Effective Communication (Verbal/Nonverbal) Presentation / Public Speaking Skills (MCQ Practice during classes)

Unit 4: MASTERING TECHNICAL COMMUNICATION- Technical Report (formal drafting) Business Letter (formal drafting) Job Application (formal drafting) Organizational

Unit 5: GROUP DISCUSSION–Principle & Practice

Text Books

1. Board of Editors: Contemporary Communicative English for Technical Communication Pearson Longman, 2010
2. Technical Communication Principle sand Practice by Meenakshi Raman, Sangeeta Sharma (Oxford Higher Education)
3. Effective Technical Communication by Barun K. Mitra (Oxford Higher Education).
4. P C WREN & H.MARTIN (English language & grammar)

References

1. D.Thakur: Syntax Bharati Bhawan, 1998
2. Longman Dictionary of Contemporary English (New Edition) for Advanced Learners
3. Internet

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Title of Course: Micro Programming & Architecture Lab
Course Code: MCA191
L-T-P Scheme: 0-0-3 **Course Credits: 2**

Objective:
To learn the fundamental aspects of computer architecture design and analysis. This lab course provides a comprehensive introduction to understand the underlying of VHDL (VHSIC Hardware Description Language) which is a hardware description language used to describe a logic circuit by function. In particular defined data flow, behaviour or structure. It can also be used as a general purpose parallel programming language i.e. commands, which correspond to logic gates, are executed (computed) in parallel, as soon as a new input arrives. The emphasis of the course will be placed on understanding HDL programming using xilinx to implement different type of circuit.

Learning Outcomes:
Students can understand the functions, structures and history of VHDL programming. Understand the data flow model, behaviaral model, structural model.

Course Contents:

Unit –I: Implement AND NOT ,OR Gate, Implement basic gates using data flow model Behavioral model and Structural model

Unit –II: Implement NAND, NOR ,XOR Gate, Implement Few gates using data flow model Behavioral model and Structural model. Individually find each and every gates out put.

Unit –III: Implement Half Adder and Full Adder, Write the code for the same circuit in different type like data flow, behavioral and structural method.

Unit –IV: Implement Half Subtractor and Full Subtractor, Write the code for the same circuit in different type like data flow, behavioral and structural method.

Unit –V: Implement Flip-Flop, S-R Flip Flop, J-K Flip Flop, D Flip Flop, T Flip Flop

Text Book:

1. "Essential of Computer Architecture", Douglas E. Corner, Pearson

References:

1. "Computer Organization and Design" David A. Patterson, John L. Hennessy, Elsevier.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Experiment – 1

Aim:-

To implement **And&Xor** Gate in Behavioral& Structural method using Xilinx 7.1i

Tools/Apparatus Used:-

(a) Xilinx 7.1i
(b) Windows 8.1
(c) Other necessary requirements

Procedure:-

(a) Make the VHDL Module with required port specification.
(b) Calculate the value of "c" (c <= a& b) for data flow and similar calculation in behavioral model using if-else statements.
(c) Make the TBW (Teat Bench Waveforms) and check the output for a given input.

Source Code:-

Data Flow Model:-

AND GATE-

```
begin
c<= a and b;
```

XOR GATE-

```
begin
c<=a xor b;
```

Behavioral Model:-

AND GATE-

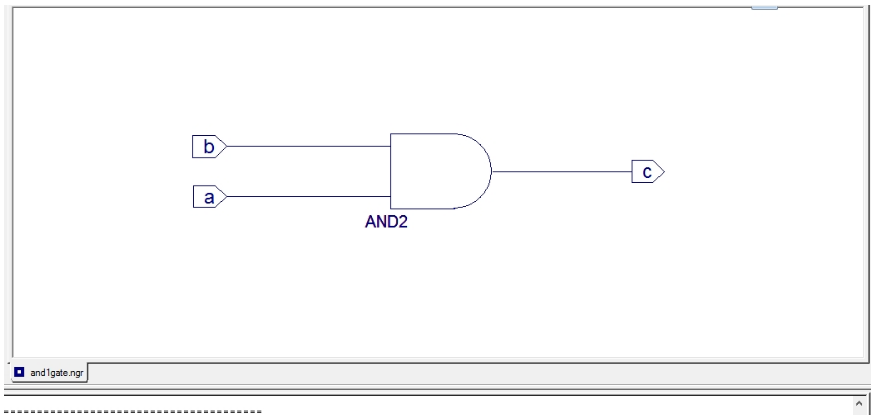
```
begin
p: process (a, b)
                begin
                if (a='1' and b='1')then
                 c<='1';
                else
                 c<='0';
                end if;
                end process;
end Behavioral;
```

XOR GATE-

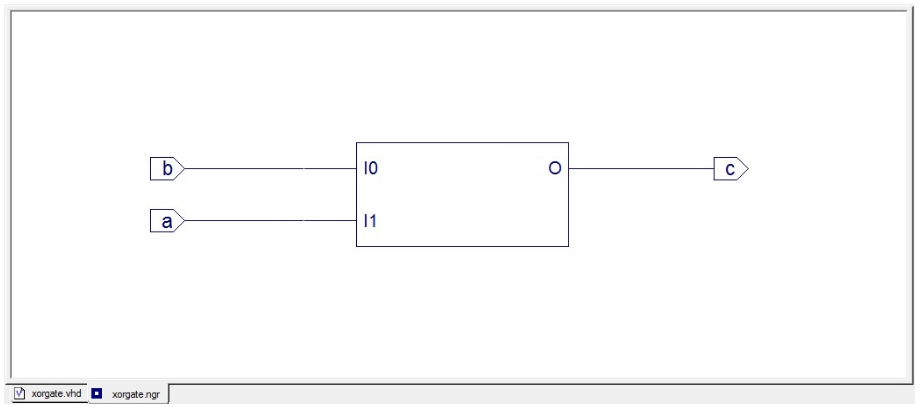
```
begin
p:process(a,b)
        begin
        If (a=b) then
                 c<='0';
                else
                 c<='1';
                end if;
                end process;
end Behavioral;
```

RTL Schematic:-

For AND-Gate:-

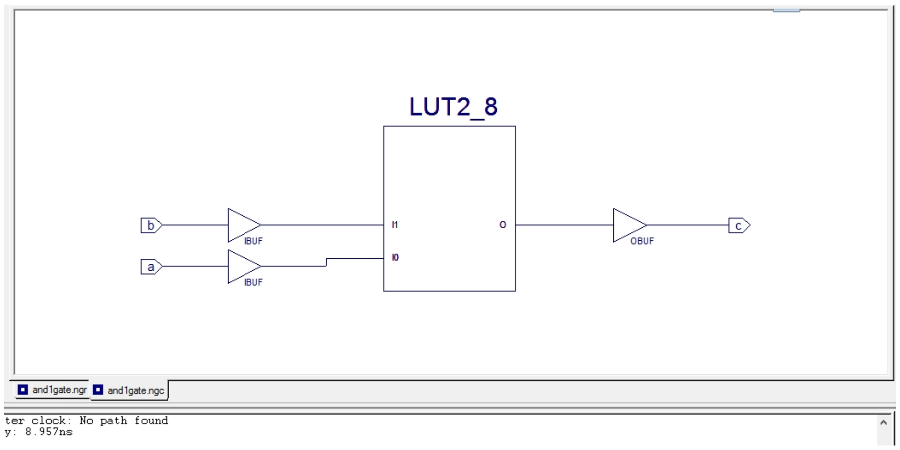


For XOR-Gate:-



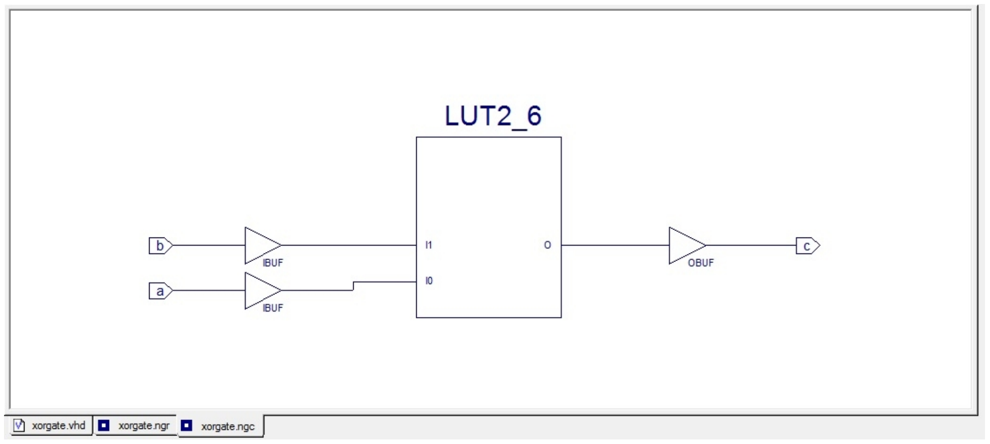
Technology Schematic:-

For AND-Gate:-



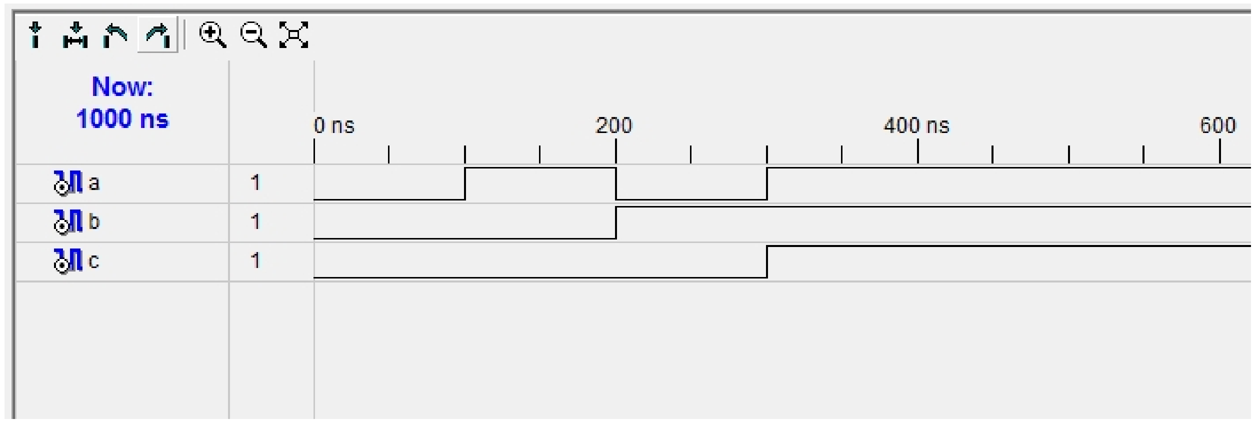
```
ter clock: No path found
y: 8.957ns
```

For XOR-Gate:-

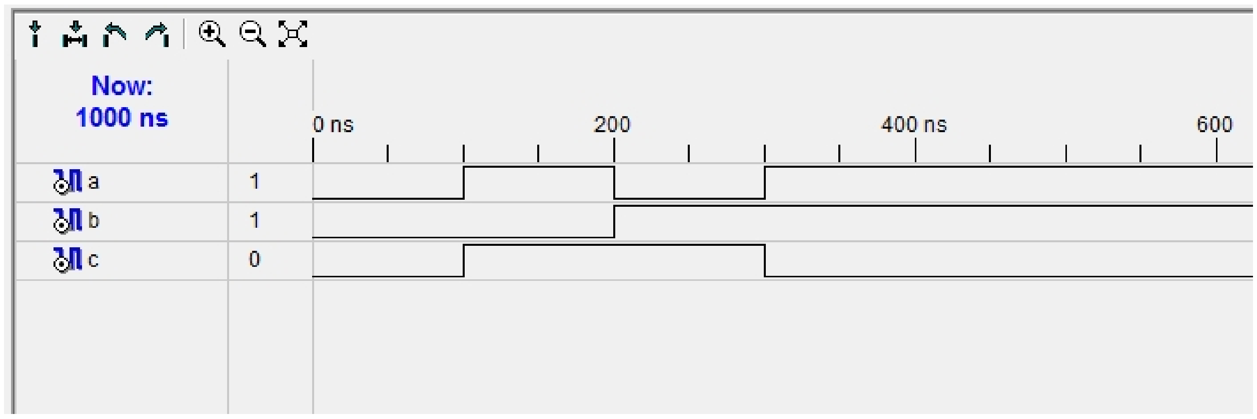


Test Bench Waveform:-

For AND-Gate:-



For XOR-Gate:-



Conclusion:-

The Desired output is received along with the desired Circuitry as per the TBW & RTL Schematic.

Experiment – 2

Aim:-
To implement **OR** Gate in Behavioral & Structural method using Xilinx 7.1i

Tools/Apparatus Used:-

(a) Xilinx 7.1i.
(b) Windows 8.1
(c) Other necessary requirements

Procedure:-

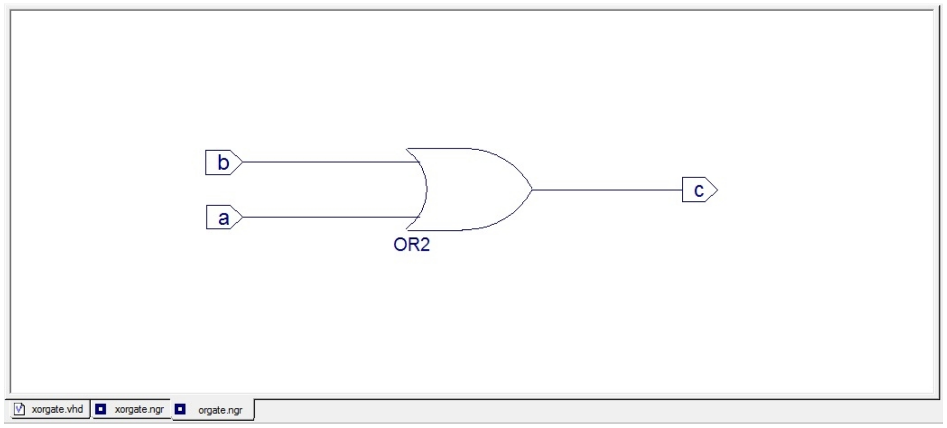
(a) Make the VHDL Module with required port specification.
(b) Calculate the value of "c" (c <= a & b) for data flow and similar calculation in behavioral model using if-else statements.
(c)Make the TBW (Teat Bench Waveforms) and check the output for a given input.

Source Code:-

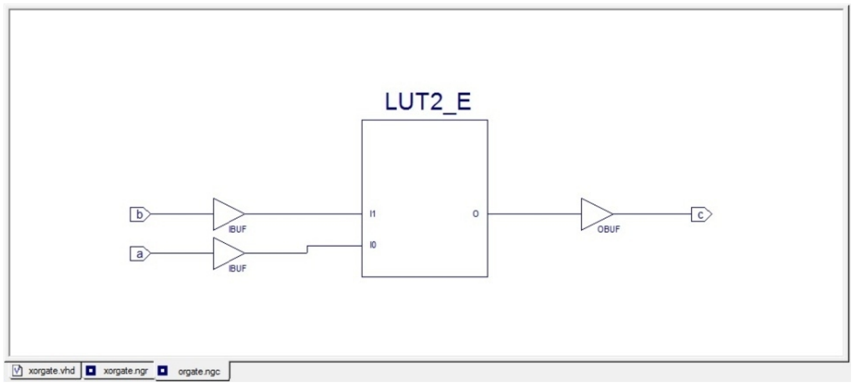
Data Flow Model:-
OR GATE-
```
begin
c<=a or b;
```

Behavioral Module:-
OR GATE-
```
begin
p:process(a,b)
                begin
                if (a='0' and b='0')then
                 c<='0';
                else
                 c<='1';
                end if;
                end process;
end Behavioral;
```

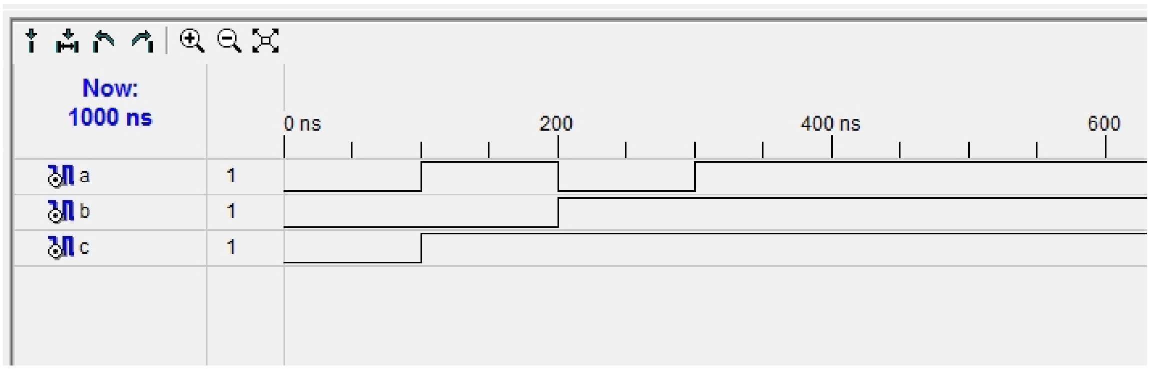
RTL Schematic:-



Technology Schematic:-



Test Bench Waveform:-



Conclusion:-

The Desired output is received along with the desired Circuitry as per the TBW & RTL Schematic.

Experiment – 3

Aim:-

To implement **NOT** Gate in Behavioral & Structural method using Xilinx 7.1i

Tools/Apparatus Used:-

(a) Xilinx 7.1i
(b) Windows 8.1
(c) Other necessary requirements

Procedure:-

(a) Make the VHDL Module with required port specification.
(b) Calculate the value of "c" (c <= a & b) for data flow and similar calculation in behavioral model using if-else statements.
(c) Make the TBW (Teat Bench Waveforms) and check the output for a given input.

Source Code:-

Data Flow Model:-

NOT GATE

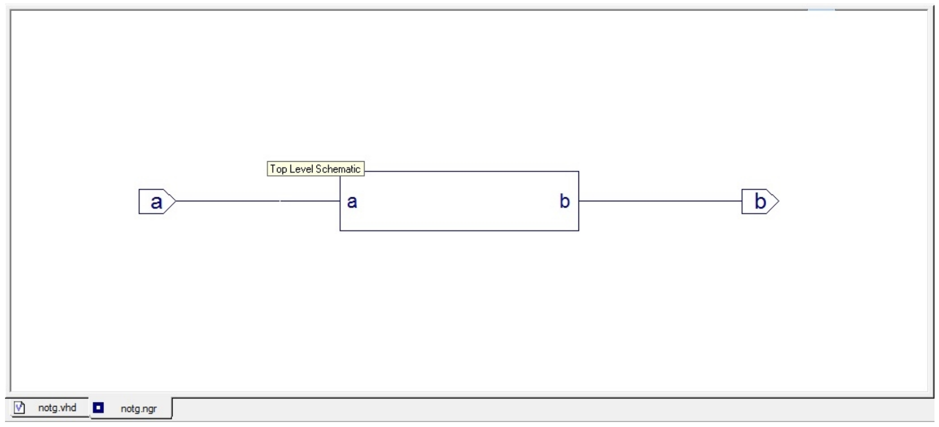
```
begin
b<=not a;
```

Behavioral Module:-

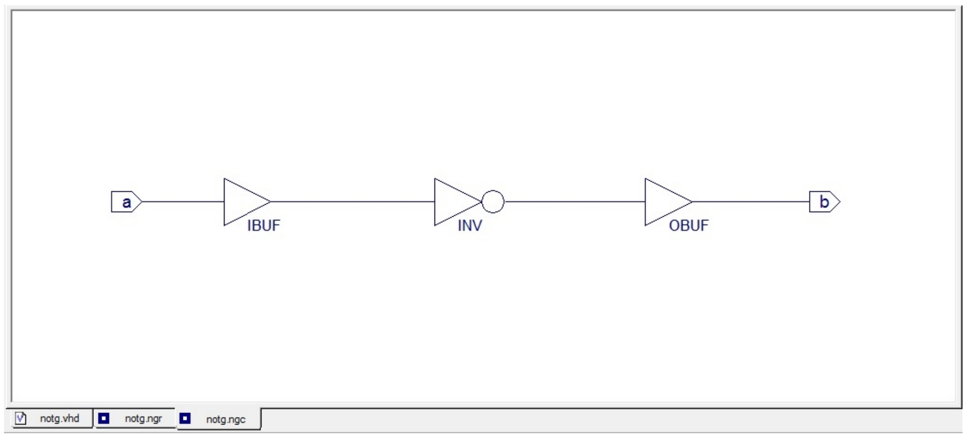
NOT GATE

```
begin
p:process(a)
begin
if (a='0')then
A`<='1';
else
a`<='0';
end if;
end process;
end Behavioral;
```

RTL Schematic:-



Technology Schematic:-



Test Bench Waveform:-

Conclusion:-

The Desired output is received along with the desired Circuitry as per the TBW & RTL Schematic.

Experiment – 4

Aim:-
To implement **NOR** Gate in Behavioral & Structural method using Xilinix 7.1i

Tools/Apparatus Used:-

(a) Xilinx 7.1i
(b) Windows 8.1
(c) Other necessary requirements

Procedure:-

(a) Make the VHDL Module with required port specification.
(b) Calculate the value of "c" (c <= a & b) for data flow and similar calculation in behavioral model using if-else statements.
(c) Make the TBW (Teat Bench Waveforms) and check the output for a given input.

Source Code:-

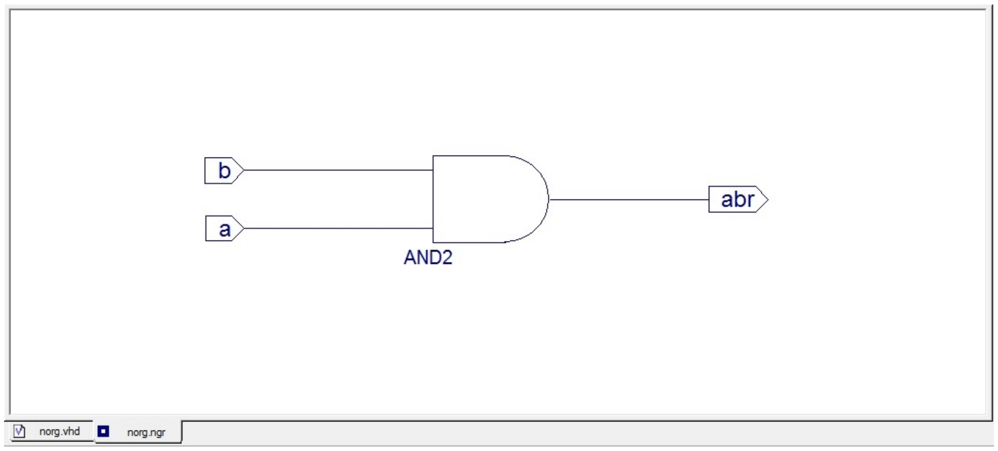
Data Flow Model:-
NOR GATE

```
begin
c<=a nor b;
```

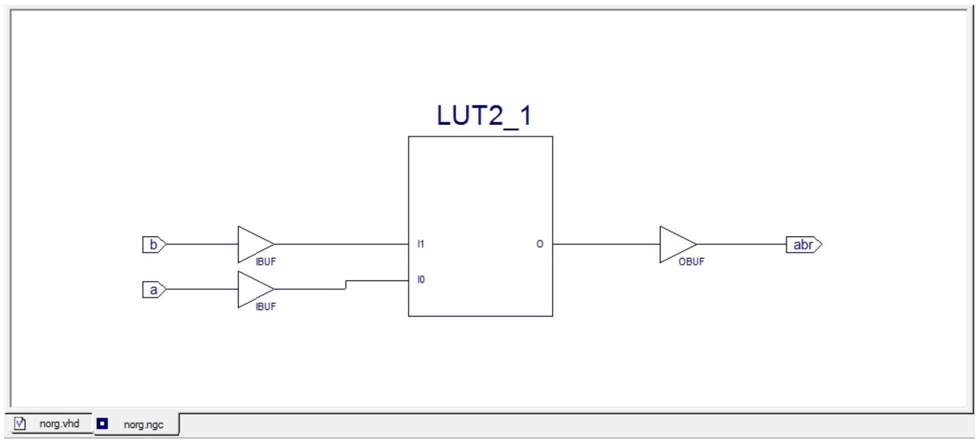
Behavioral Module:-
NOR GATE

```
begin
        p:process(a,b)
                begin
                if (a='0' and b='0')then
                abr<='1';
                else
                 abr<='0';
                 end if;
                 end process;

end Behavioral;
```

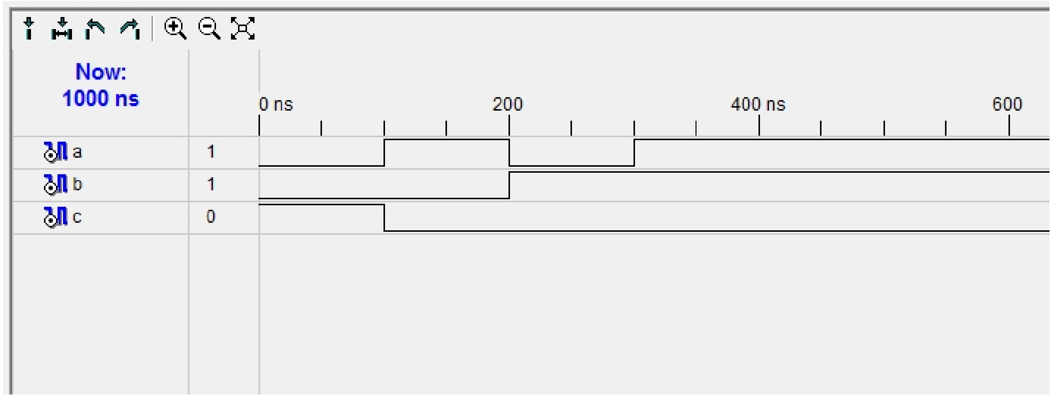
RTL Schematic:-



Technology Schematic:-



Test Bench Waveform:-



Conclusion:- The Desired output is received along with the desired Circuitry as per the TBW & RTL Schematic.

Experiment – 5

Aim:-

To implement **NAND** Gate in Behavioral & Structural method using Xilinx 7.1i

Tools/Apparatus Used:-

(d) Xilinx 7.1i
(e) Windows 8.1
(f) Other necessary requirements

Procedure:-

(d) Make the VHDL Module with required port specification.
(e) Calculate the value of “c” (c <= a & b) for data flow and similar calculation in behavioral model using if-else statements.
(f) Make the TBW (Teat Bench Waveforms) and check the output for a given input.

Source Code:-

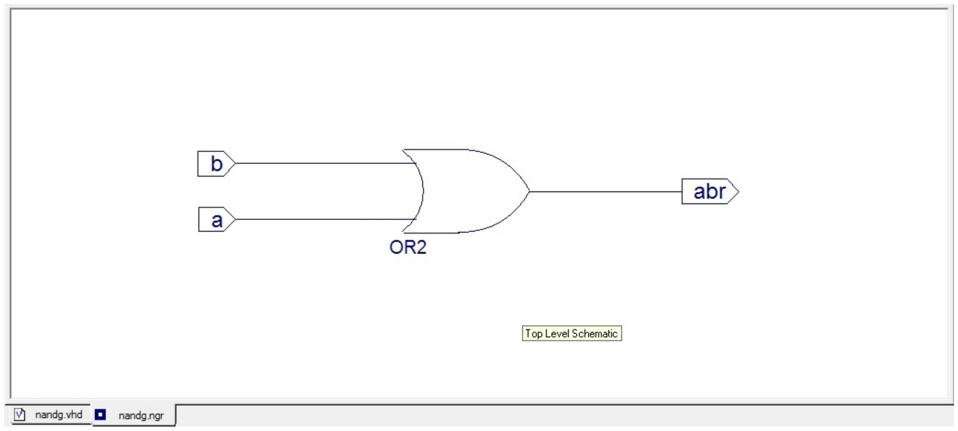
Data Flow Model:-
NAND GATE-

```
c<= a nand b;
```

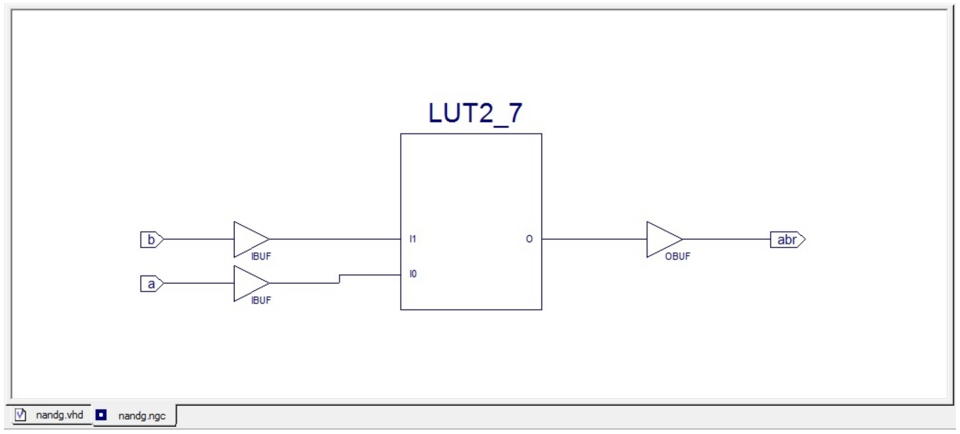
Behavioral Module:-
NAND GATE-

```
begin
p:process(a,b)
                begin
                if(a='1' and b='1')then
                abr<='0';
                else
                abr<='1';
                end if;
                end process;
end Behavioral;
```

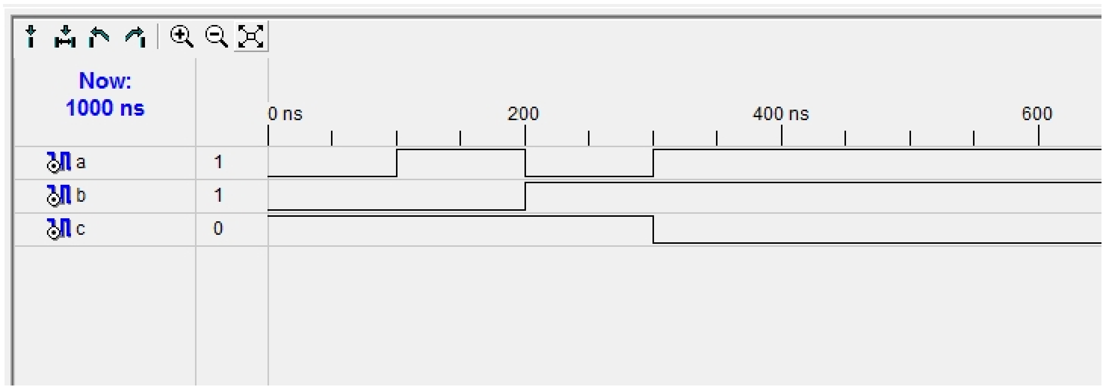
RTL Schematic:-



Technology Schematic:-



Test Bench Waveform:-



Conclusion:-

The Desired output is received along with the desired Circuitry as per the TBW & RTL Schematic.

Experiment – 6

Aim:-

To implement Adder in Behavioral& Structural method using Xilinx 7.1i

Tools/Apparatus Used:-

(g) Xilinx 7.1i
(h) Windows 8.1
(i) Other necessary requirements

Procedure:-

(g) Make the VHDL Module with required port specification.
(h) Calculate the value of “c” (c <= a& b) for data flow and similar calculation in behavioral model using if-else statements.
(i) Make the TBW (Teat Bench Waveforms) and check the output for a given input.

Source Code:-

Data Flow Model:-

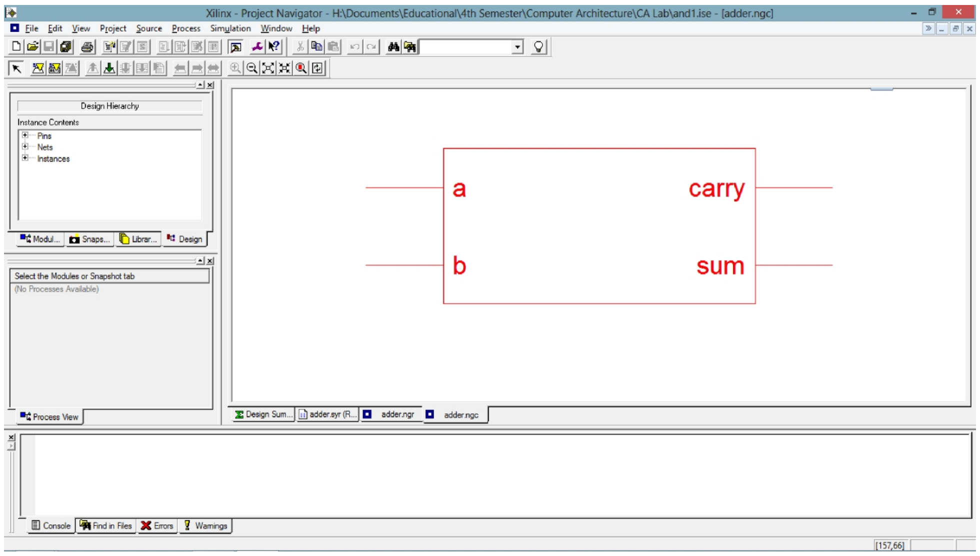
Data Flow Model-
HALF ADDER-

```
begin
sum<=a xor b;
carry<= a and b ;
```

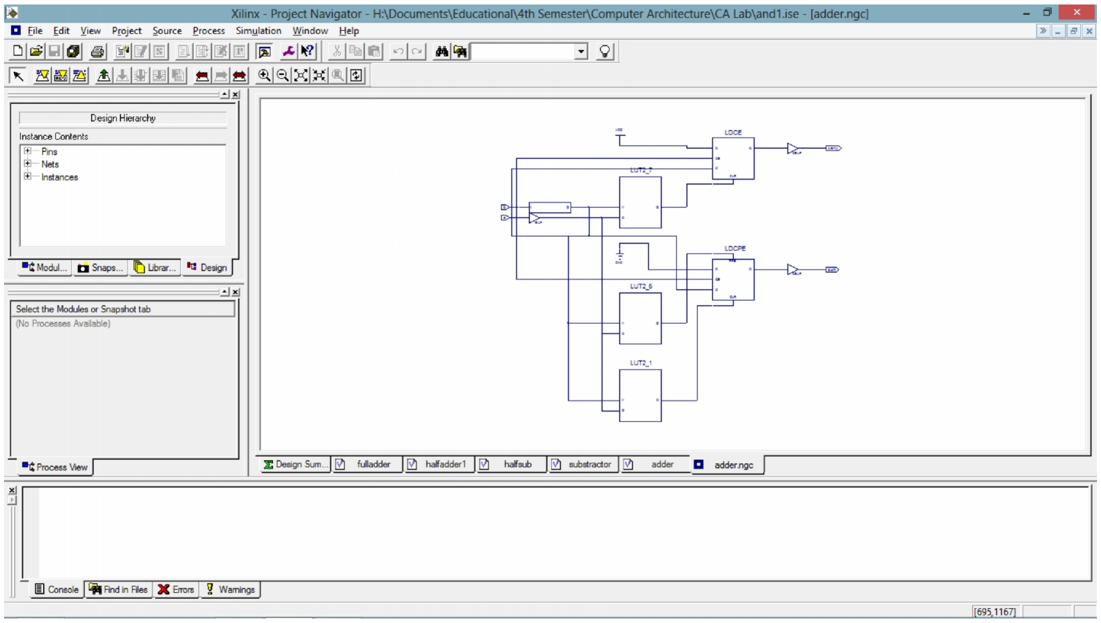
Behavioral Module:-

```
begin
p:process(a,b)
begin
if(a='0' and b='0')then
sum<='0';
carry<='0';
elsif(a='0' and b='1' )then
sum<='1';
carry<='0';
elsif(a='1' and b='0' )then
sum<='1';
carry<='0';
elsif(a='1' and b='1' )then
sum<='0';
carry<='1';
end if;
end process;
```

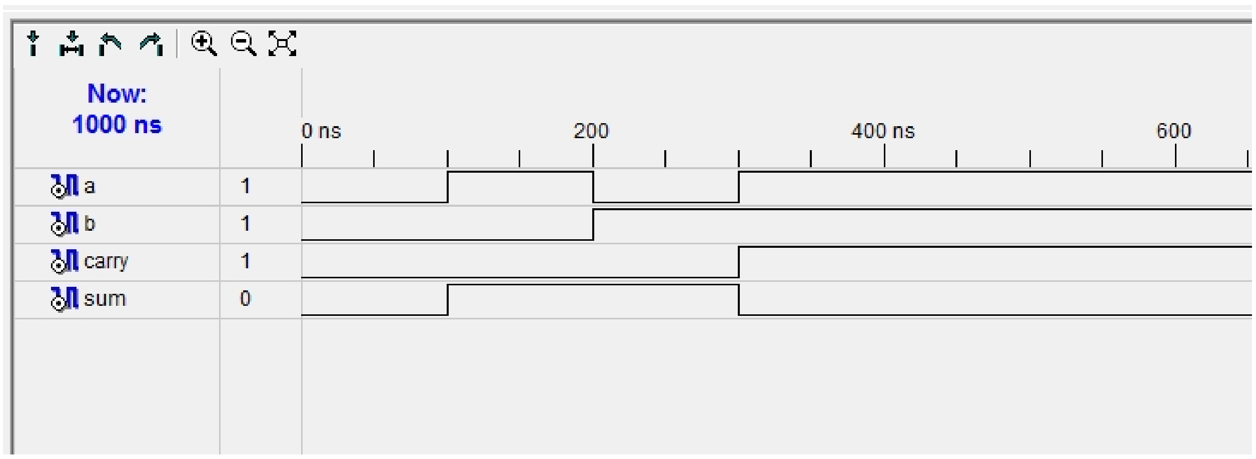
RTL Schematic:-


Xilinx - Project Navigator - H:\Documents\Educational\4th Semester\Computer Architecture\CA Lab\and1.ise - [adder.ngc]
File Edit View Project Source Process Simulation Window Help
Design Hierarchy
Instance Contents
Pins
Nets
Instances
Modul... Snaps... Librar... Design
Select the Modules or Snapshot tab
(No Processes Available)
Process View
a
b
carry
sum
Design Sum... adder.syr (R... adder.ngr adder.ngc
Console Find in Files Errors Warnings
[157,66]

Technology Schematic:-


Xilinx - Project Navigator - H:\Documents\Educational\4th Semester\Computer Architecture\CA Lab\and1.ise - [adder.ngc]
File Edit View Project Source Process Simulation Window Help
Design Hierarchy
Instance Contents
Pins
Nets
Instances
Modul... Snaps... Librar... Design
Select the Modules or Snapshot tab
(No Processes Available)
Process View
LDCE
LUT2_7
LDCPE
LUT2_6
LUT2_1
Design Sum... fulladder halfadder1 halfsub substractor adder adder.ngc
Console Find in Files Errors Warnings
[695,1167]

Test Bench Waveform:-



Conclusion:-

The Desired output is received along with the desired Circuitry as per the TBW & RTL Schematic.

Experiment – 7

Aim:-

To implement Full Adder in Behavioral & Structural method using Xilinx 7.1i

Tools/Apparatus Used:-

(j) Xilinx 7.1i
(k) Windows 8.1
(l) Other necessary requirements

Procedure:-

(j) Make the VHDL Module with required port specification.
(k) Calculate the value of "c" (c <= a & b) for data flow and similar calculation in behavioral model using if-else statements.
(l) Make the TBW (Teat Bench Waveforms) and check the output for a given input.

Source Code:-

Data Flow Model:-

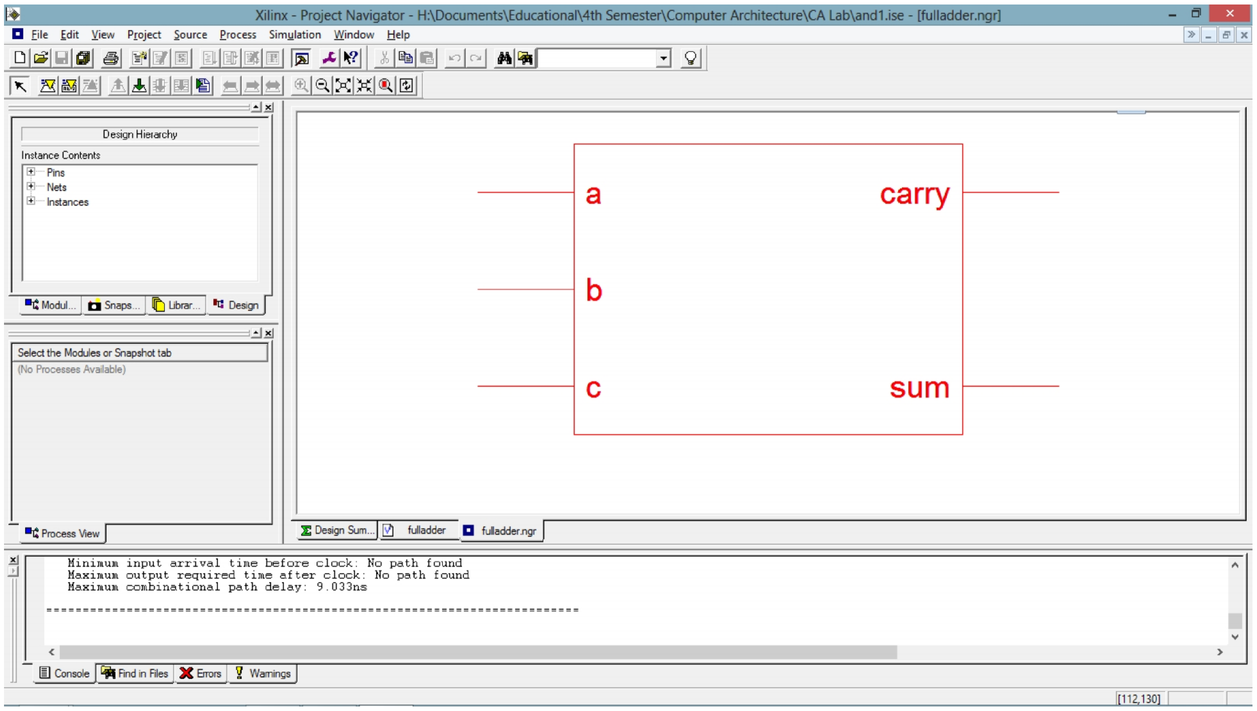
FULL ADDER:-

```
begin
sum<=a xor b xor c;
carry<= (c and(a xor b)) or (a and b);
```

Behavioral Module:-

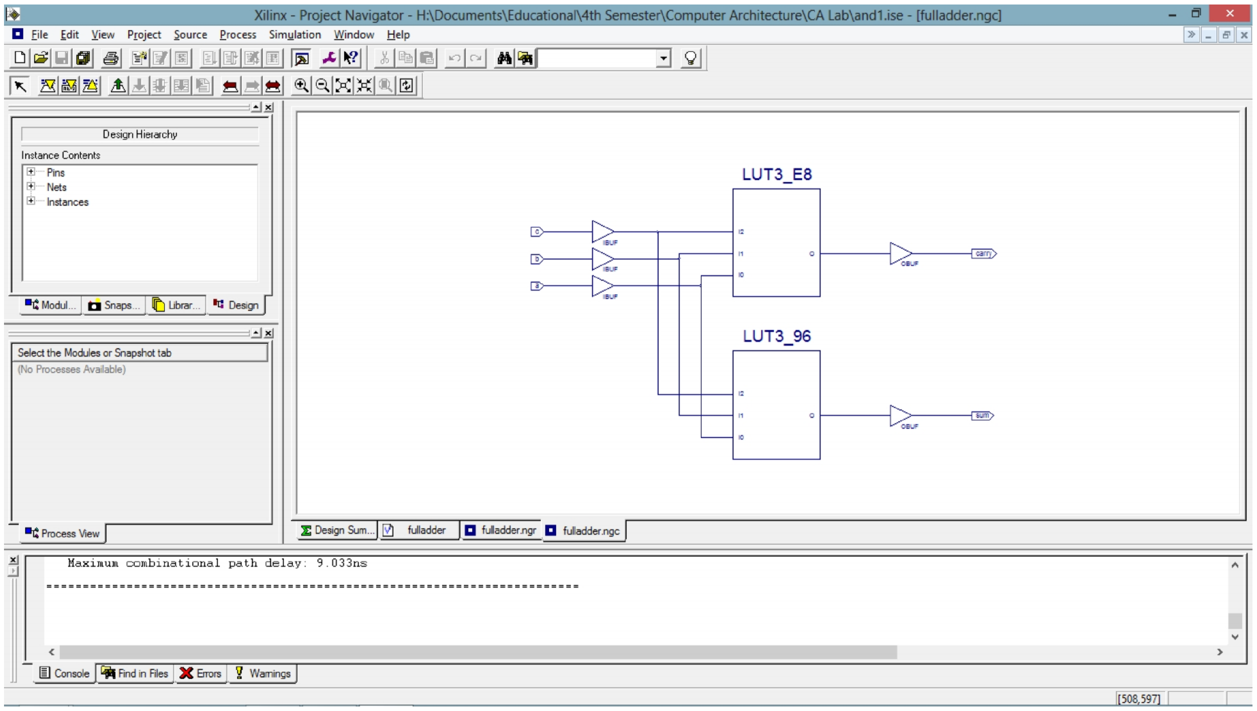
```
begin
p:process(a,b,c)
begin
if(a='0' and b='0' and c='0')then
sum<='0';
carry<='0';
elsif(a='0' and b='0' and c='1')then
sum<='1';
carry<='0';
elsif(a='0' and b='1' and c='0')then
sum<='1';
carry<='0';
elsif(a='0' and b='1' and c='1')then
sum<='0';
carry<='1';
elsif(a='1' and b='0' and c='0')then
sum<='1';
carry<='0';
elsif(a='1' and b='0' and c='1')then
sum<='0';
carry<='1';
elsif(a='1' and b='1' and c='0')then
sum<='0';
carry<='1';
else
sum<='1';
carry<='1';
end if;
end process;
```

RTL Schematic:-


Xilinx - Project Navigator - H:\Documents\Educational\4th Semester\Computer Architecture\CA Lab\and1.ise - [fulladder.ngr]
File Edit View Project Source Process Simulation Window Help
Design Hierarchy
Instance Contents
Pins
Nets
Instances
Modul... Snaps... Librar... Design
Select the Modules or Snapshot tab
(No Processes Available)
Process View
a
b
c
carry
sum
Design Sum... fulladder fulladder.ngr
Console Find in Files Errors Warnings
[112,130]

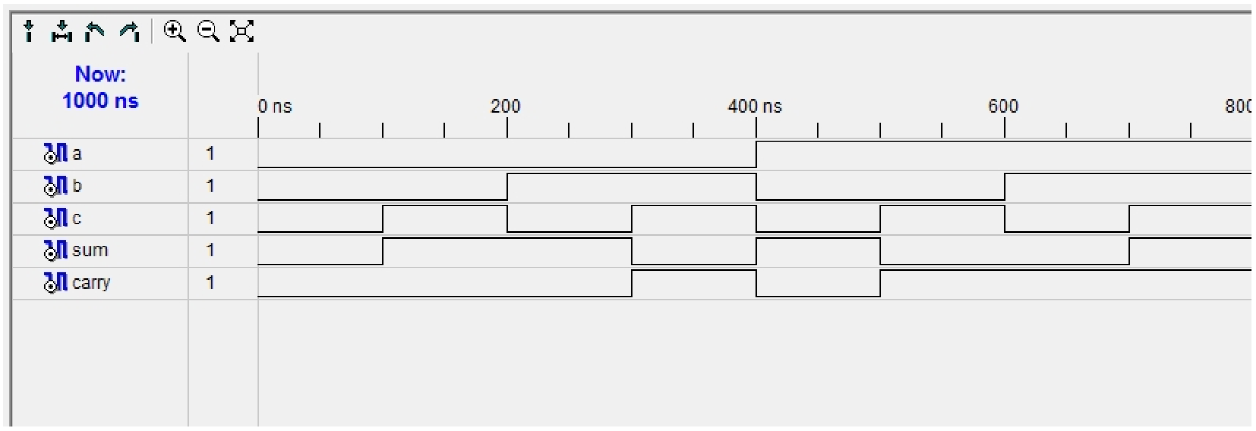
```
    Minimum input arrival time before clock: No path found
    Maximum output required time after clock: No path found
    Maximum combinational path delay: 9.033ns

=========================================================================
```

Technology Schematic:-


Xilinx - Project Navigator - H:\Documents\Educational\4th Semester\Computer Architecture\CA Lab\and1.ise - [fulladder.ngc]
File Edit View Project Source Process Simulation Window Help
Design Hierarchy
Instance Contents
Pins
Nets
Instances
Modul... Snaps... Librar... Design
Select the Modules or Snapshot tab
(No Processes Available)
Process View
LUT3_E8
LUT3_96
c
b
a
IBUF
IBUF
IBUF
I2
I1
I0
O
OBUF
OBUF
carry
sum
Design Sum... fulladder fulladder.ngr fulladder.ngc
Console Find in Files Errors Warnings
[508,597]

```
    Maximum combinational path delay: 9.033ns

=========================================================================
```

Test Bench Waveform:-



Conclusion:-

The Desired output is received along with the desired Circuitry as per the TBW & RTL Schematic.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Experiment – 8

Aim:-

To implement Full Subtractor in Behavioral & Structural method using Xilinx 7.1i

Tools/Apparatus Used:-

(a) Xilinx 7.1i.

(b) Windows 8.1.

(c) Other necessary requirements.

Procedure:-

(a) Make the VHDL Module with required port specification.
(b) Calculate the value of "c" (c <= a & b) for data flow and similar calculation in behavioral model using if-else statements.
(c) Make the TBW (Teat Bench Waveforms) and check the output for a given input.

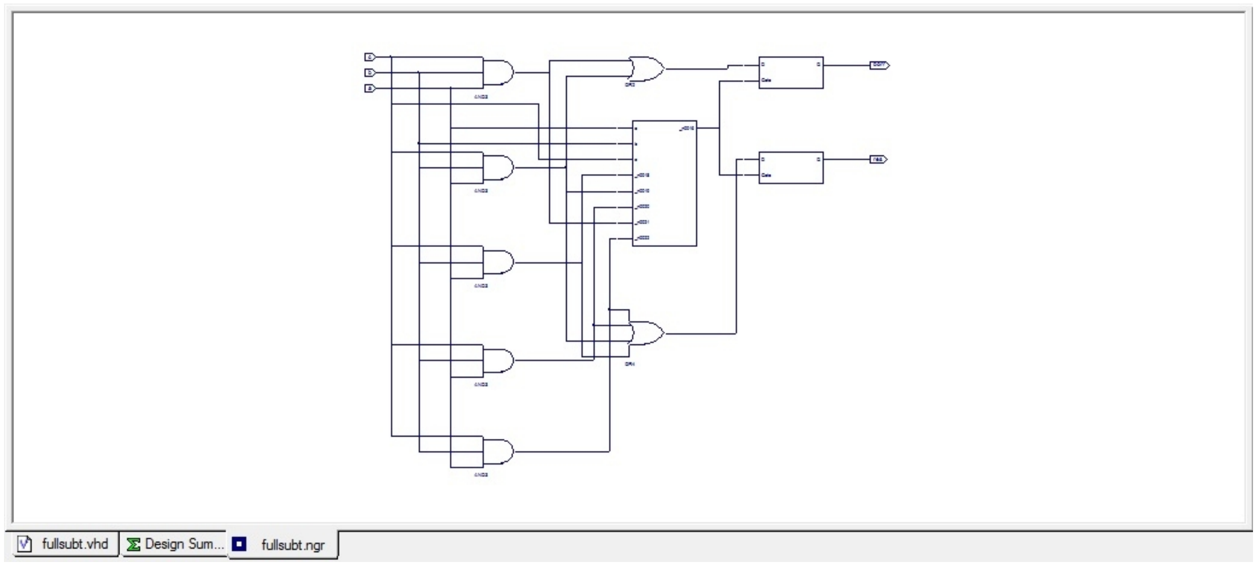
Source Code:-

Data Flow Model:-

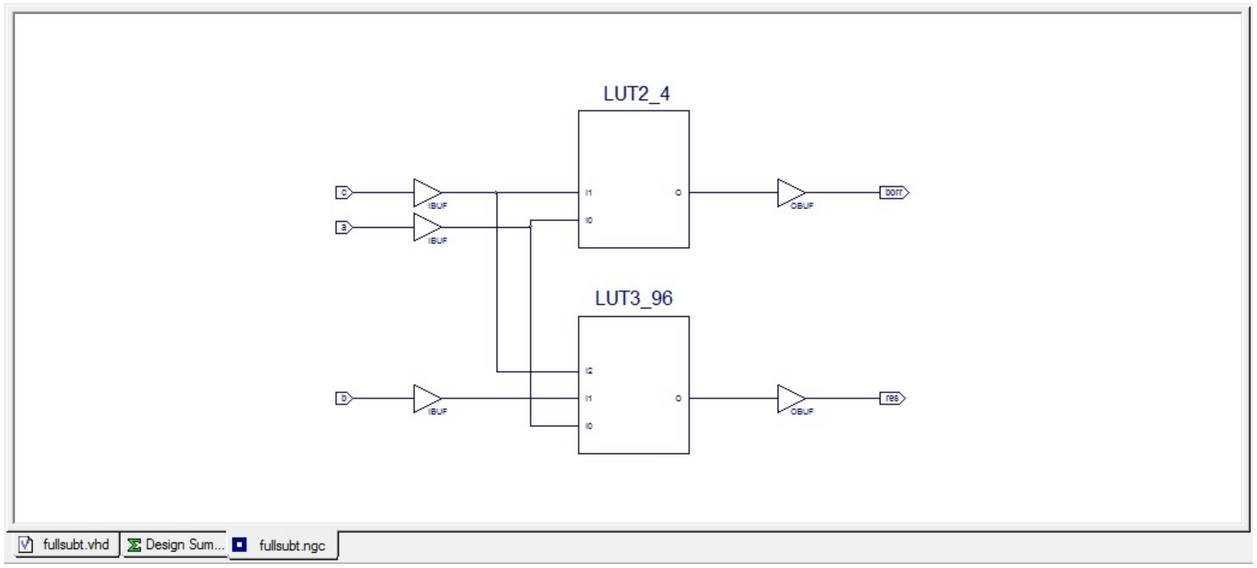
FULL SUBTRACTOR-

```
Begin

sub<= (a and (b xor c)) or (a not (b xor c));

borrow<= ((not a) and c);
```

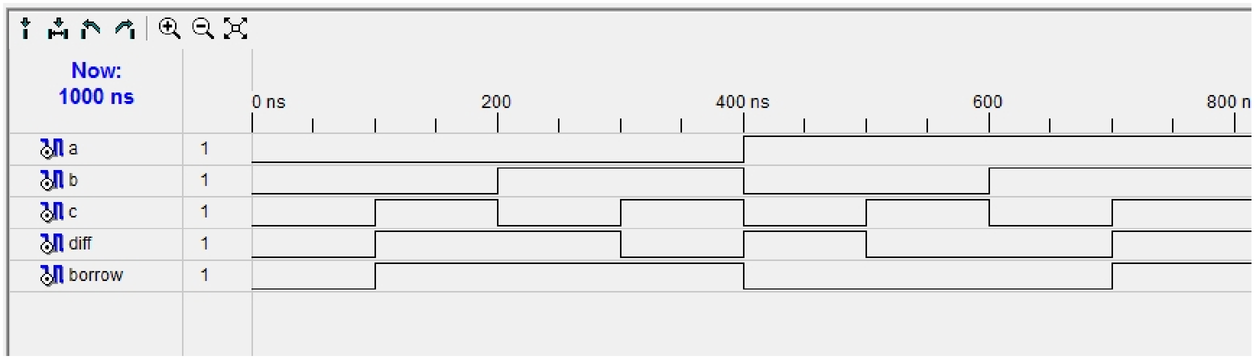
RTL Schematic:-



Technology Schematic:-



Test Bench Waveform:-



Conclusion:-

The Desired output is received along with the desired Circuitry as per the TBW & RTL Schematic.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Title of Course: Programming Lab (C)
Course Code: MCA192
L-T-P scheme: 0-0-3 **Course Credit: 2**

Introduction:
This course is designed to familiarize students with the basic components of a computer, so as to be able to operate it and be able to interact with it, and carry out simple tasks. In addition, it will initiate the students into the discipline of Programming. It aims to start off the development of problem solving ability using computer programming. This course teaches not only the mechanics of programming, but also how to create programs that are easy to read, maintain, and debug. Students are introduced to the design principles for writing good programs regardless of the hardware and the software platforms.

Objective:
Students will develop their ability to design, develop, test and document structured programs in C language.

Learning Outcomes: Students should be able to
1. Understand the basic terminology used in computer programming
2. Write, compile and debug programs in C language.
3. Use different data types in a computer program.
4. Design programs involving decision structures, loops and functions.
5. Explain the difference between call by value and call by reference
6. Understand the dynamics of memory by the use of pointers.
7. Enhance programming skills through problem solving and code development of small-size software applications.
8. Improve self-learning, teamwork and communication skills through project development practices.
9. Engage in continuing professional development under minimal guidance.

Course Contents:
Exercises that must be done in this course are listed below:
1 Introduction to C programming
2 Structured Program Development in C
3 Flowchart and Algorithm
4 C Program Control
5 C Functions
6 C Arrays
7 C Pointers
8 C Characters and Strings
9 C Structures, Unions, Bit Manipulations and Enumerations
10 C File Processing

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

CYCLE – I

1. Write a program to evaluate area of triangle using the formula sqrt(s(s-a)(s-b)(s-c))
2. Write a program to swap two numbers.
3. Write a program to find the greatest of three numbers and print the numbers in ascending order.
4. Write a program to perform the arithmetic expression using switch statement.
5. Write a program to find a factorial of given number using do while statement.
6. Write a program to print all prime numbers upto ‘N’ numbers.
7. Write a program to print sum of ‘N’ natural numbers.
8. Write a program to find the total number of even integers and odd integers of ‘N’ numbers.
9. Write a program to find the sum of odd numbers and even numbers upto ‘N’ numbers.
10. Write a program to print the product of two matrices of any order.
11. Write a program to read ‘N’ number of students with 5 subject marks.
12. Write a program to find greatest of ‘n’ numbers using functions.
13. Write a program to print Fibonacci series using recursion.
14. Write a program to convert all lower case to uppercase characters.
15. Write a program to sort 5 city names in alphabetical order.

CYCLE II

16. Write a program to extract a string.
17. Write a program to implement the concept of call by value.
18. Write a program to implement the concept of call by reference.
19. Write a program to implement the concept of structure and union.
20. Write a program to access a variable using pointer.
21. Write a program to print the element of array using pointers.
22. Write a program to print the elements of a structure using pointers.
23. Write a program to display student information by initializing structures.
24. Write a program to pass structure as arguments to function and calculate total marks of 5 subjects.
25. Write a program to write integer data into file and read it from file.

Ex. No. 1 **AREA OF TRIANGLE**

AIM:

To write a program for evaluating the area of triangle using the formula sqrt(s(s-a)(s-b)(s-c)).

ALGORITHM:

```
Step1: Start the program.
Step2: Get the inputs a, r, t and s.
Step3: Calculate s = (a+b+c) / 2.
Step4: Calculate area=sqrt(s*(s-a)*(s-b)*(s-c)).
Step 5: Print the result 'area'.
Step 6: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
#include<math.h>
void main()
{
   int a,b,c;
   float s,area;
   clrscr();
   printf("Enter the values of a,b,c: ");
   scanf("%d%d%d",&a,&b,&c);
   s=(a+b+c)/2;
   area=sqrt(s*(s-a)*(s-b)*(s-c));
   printf("The area of a triangle is =%f",area);
   getch();
}
```

OUTPUT:

```
Enter the values of a,b,c: 10 20 30

The area of a triangle is = 0.000000
```

RESULT:

Thus the C program to find the area of triangle using the formula sqrt(s(s-a)(s-b)(s-c)) has been successfully executed and verified.

Ex. No. 2 **SWAP TWO NUMBERS**

AIM:

To write a program for swapping of two numbers.

ALGORITHM:

```
Step1: Start the program.
Step2: Get the inputs a and b.
Step3: Find a=a+b.
Step4: Find b=a-b.
Step 5: Find a=a-b.
Step6: Print the result 'a' and 'b'.
Step7: Stop the program.
```

PROGRAM:

```c
#include<stdio.h>
#include<conio.h>
void main()
{
  int a,b;
  clrscr();
  printf("Enter the values of a and b: ");
  scanf("%d%d",&a,&b);
  a=a+b;
  b=a-b;
  a=a-b;
  printf("The values of a and b are: %d %d", a, b);
  getch();
}
```

OUTPUT:

```
Enter the values of a and b: 10 20

The values of a and b are: 20 10
```

RESULT:

Thus the C program to swap two numbers has been successfully executed and verified.

Ex. No. 3 GREATEST OF THREE NUMBERS AND PRINT ASCENDING ORDER AIM:

To write a program for finding the greatest of three numbers and printing the numbers in ascending order.

ALGORITHM:

```
Step1: Start the program.
Step2: Get the inputs a, b and c.
Step3: Check if((a>b) &&(a>c))
Step4: Again check if(b>c)
Step5: Then print the greatest number and display a, b, c.
Step6: Else print the greatest number and display a, c, b.
Step7: Check if((b<c) &&(b<a))
Step8: Again check if(c<a)
Step9: Then print the greatest number and display b, c, a.
Step10: Else print the greatest number and display b, a, c.
Step11: Check if((c<a) && (c<b))
Step12: Again check if(a<b)
Step13: Then print the greatest number and display c, a, b.
Step14: Else print the greatest number and display c, b, a.
Step15: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
clrscr();
printf("Enter the values of a, b and c: ");
scanf("%d%d%d", &a, &b, &c);
if(a<b && a<c)
{
        if(b<c)
```

```
                {
                        printf("The greatest number is: %d", a);
                        printf("The ascending order: %d%d%d", a, b, c);
                }
        else
                if(b>c)
                {
                        printf("The greatest number is: %d", a);
                        printf("The ascending order: %d%d%d", a, c, b);
                }
}
else if(b<c && b<a)
        {
                if(c<a)
                {
                        printf("The greatest number is: %d", b);
                        printf("The ascending order: %d%d%d", b, c, a);
                }
        else
                {
                        printf("The greatest number is: %d", b);
                        printf("The ascending order: %d%d%d", b, a, c);
                }
        }
else
        if(b<a)
        {
                printf("The greatest number is: %d", c);
                printf("The ascending order: %d%d%d", c, b, a);
        }
        else
                {
                        printf("The greatest number is: %d", c);
                        printf(The ascending order: %d%d%d", c, a, b);
                }
}
```

OUTPUT:

```
Enter the values of a, b and c: 6 4 5

The greatest number is: 6
The ascending order: 4 5 6
```

RESULT:

Thus the C program to find greatest of three and to print the numbers in ascending order has been successfully executed and verified.

Ex. No. 4 ARITHMETIC EXPRESSION USING SWITCH STATEMENT

AIM:

To write a program for performing the arithmetic expression using switch statement.

ALGORITHM:

```
Step1: Start the program.
Step2: Display 1. Addition 2. Subtraction 3. Multiplication and 4. Division
Step3: Get the input a and b.
Step4: Get the choice.
Step5: Switch(result)
Step6: case '+': print the sum of a & b.
Step7: case '-': print the difference of a & b.
Step8: case '*': print the multiplication of a & b.
Step9: case '/': print the division of a & b.
Step10: default: invalid option.
Step11: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b;
int op;
clrscr();
printf("Enter the values of a & b: ");
scanf("%d%d", &a, &b);
printf(" 1.Addition\n 2.Subtraction\n 3.Multiplication\n 4.Division\n");
printf("Enter your choice: ");
scanf("%d", &op);
switch(op)
  {
        case 1 :printf("Sum of %d and %d=%d", a, b, a+b);
        break;
        case 2 :printf("Subtraction of %d and %d=%d", a, b, a-b);
        break;
        case 3 :printf("Multiplication of %d and %d=%d", a, b, a*b);
        break;
        case 4 :printf("Division of %d and %d=%d", a, b, a/b);
        break;
        default : printf(" Enter Your Correct Choice.");
        break;
  }
getch();
}
```

OUTPUT:

```
Enter the values of a & b: 10 20

1. Addition
2. Subtraction
3. Multiplication
4. Division

Enter your choice: 1

Sum of 10 and 20 = 30
```

RESULT:

Thus the C program for arithmetic expression using switch statement has been successfully executed and verified.

Ex. No. 5 FACTORIAL OF A NUMBER USING DO WHILE STATEMENT

AIM:

To write a program for finding the factorial of a given number using do while statement.

ALGORITHM:

```
Step1: Start the program.
Step2: Assign f=i=1.
Step3: Get the input n.
Step4: do .. the following.
Step5: Find f=f*i
Step6: Increment i=i+1
Step7: Repeat from step5 to step6 till while(i<=no).
Step8: Then print f.
Step9: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n,i,f;
        f=i=1;
        clrscr();
        printf("Enter a number: ");
        scanf("%d",&n);
        do
         {
                fact*=i;
                i++;
         }while(i<=no);
        printf("Factorial of %d=%d\n", no, fact;
}
```

OUTPUT:

```
Enter a number: 5

Factorial of 5 = 120
```

RESULT:

Thus the C program for finding the factorial of a given number using do while statement has been successfully executed and verified.

Ex. No. 6 **GENERATE PRIME NUMBERS UPTO N NUMBERS**

AIM: To write a program for printing all prime numbers upto N numbers.

ALGORITHM:

```
Step1: Start the program.
Step2: Get the n value.
Step3: for(i=1;i<=n;i++)
Step4: Repeat a, b, c, d & e
    a) Assign fact=0
    b) for(j=1;j<=n;j++) repeat c & d
    c) if i percentage j equal to zero
    d) fact equal to fact added with one
    e) if fact equal to 2 print i as prime number
Step5: Display the prime number till nth number.
Step6: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n,i,fact,j;
        printf("Enter the range: ");
        scanf("%d",&n);
        printf("Prime numbers are: \n");
        for(i=1;i<=n;i++)
        {
                fact=0;
                for(j=1;j<=n;j++)
                {
                if(i%j==0)
                fact++;
                if(f==2)
                printf("%d ",i);
                }
          getch();
        }
}
```

OUTPUT:

```
Enter the range: 10
Prime numbers are: 3 5 7
```

RESULT:

Thus the C program for printing all prime numbers upto N numbers has been successfully executed and verified.

Ex. No. 7 **SUM OF N NATURAL NUMBERS**

AIM:

To write a program for printing the sum of N natural numbers.

ALGORITHM:

```
Step1: Start the program.
Step2: Get the n value.
Step3: Initialize i=0 and sum=0.
Step4: Perform from step 5 to step 6 until i<=n
Step5: i++
Step6: sum+=i
Step7: Print the sum.
Step9: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n,i=0,sum=0;
  clrscr( );
  printf("Enter the Limit : ");
  scanf("%d",&n);
  while(i<=n)
           {
            i++;
            sum+=i;
           }
  printf("Sum of %d natural numbers = %d",n,sum);
  getch();
}
```

OUTPUT:

```
Enter the Limit : 10

Sum of 10 natural numbers = 55
```

RESULT:

Thus the C program for printing the sum of N natural numbers has been successfully executed and verified.

Ex. No. 8 TOTAL NUMBER OF EVEN INTEGERS AND ODD INTEGERS OF 'N' NUMBERS

AIM: To write a program for finding the total number of even integers and odd integers of 'N' numbers.

ALGORITHM:

```
Step1: Start the program.
Step2: Declare int i, n, odd=0 and even=0;
Step3: Get the n value
Step4: for( i=0;i<=n;i++) do the following step.
        a) Check if(i%2==0)
        b) even=even+1;
        c) Else odd=odd+1;
Step5: Print the odd and even value.
Step6: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n,i,odd=0,even=0;
  clrscr();
  printf("Enter the n value: ");
  scanf("%d",&n);
  for(i=1;i<n;i++)
   {
    if(i%2==0)
    even=even+1;
    else
    odd=odd+1;
   }
  prinf("The total number of odd integers =%d",odd);
  prinf("The total number of even integers =%d",even);
  getch();
 }
```

OUTPUT:

```
Enter the n value: 10
The total number of odd integers =5
The total number of even integers = 5
```

RESULT:

Thus the above C program for finding the total number of even integers and odd integers of 'N' numbers has been successfully executed and verified.

Ex. No. 9 SUM OF EVEN INTEGERS AND ODD INTEGERS OF 'N' NUMBERS AIM:

To write a program for finding the sum of even integers and odd integers of 'N' numbers.

ALGORITHM:

```
Step1: Start the program.
Step2: Declare int i, n, odd=0 and even=0;
Step3: Get the n value
Step4: for( i=0;i<=n;i++) do the following step.
    a) Check if(i%2==0)
    b) even=even+i;
    c) Else odd=odd+i;
Step5: Print the odd and even value.
Step6: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,n,sum,even=0,odd=0;
clrscr();
printf("Enter any number: ");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
if(i%2==0)
even=even+i;
else
odd=odd+i;
}
printf("Sum of even integer is: %d",even);
printf("Sum of odd integer is: %d",odd);
getch();
}
```

OUTPUT:

```
Enter any value: 5
Sum of even integer is: 6
Sum of odd integer is: 9
```

RESULT:

Thus the C program for finding the sum of even integers and odd integers of 'N' numbers has been successfully executed and verified.

Ex. No. 10 **PRODUCT OF TWO MATRICES OF ANY ORDER**

AIM:

To write a program for finding the product of two matrices of any order.

ALGORITHM:

```
Step1: Start the program.
Step2: Declare int Matrix A[9][9] , MatrixB[9][9] , Matrixsproduct [9][9].
Step3: Declare int n , i , j , k, Row1 , Row2 , Column1 , Column2.
Step4: Enter the order of Matrix A Row1, Column1.
Step4: Enter the order of Matrix B Row2, Column2.
Step5: Check if(Column1 == Row2)
Step6: Enter the elements of Matrix A and B using for loops.
Step7: Find Matrixproduct[i][j] = Matrixproduct[i][j] +(Matrix A[i][k] * Matrix
B[k][j] using for loops.
Step7: Print the resultant matrix Matrixproduct[i][j] using for loop.
Step8: Else print invalid order so multiplication not possible.
Step9: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int Matrix A[9][9] , MatrixB[9][9] , Matrixsproduct [9][9] ;
  int n , i , j , k; /* 'i' used for rows and 'j' used for columns */ int Row1 , Row2 ,
  Column1 , Column2; clrscr();

  printf(" Enter the order of Matrix A\n");
  scanf("%d * %d " , &Row1 , &Column1);
  printf(" Enter the order of Matrix B\n");
  scanf("%d * %d " , &Row2 , &Column2);
  if(Column1 == Row2)
   {
    printf(" Enter the elements of Matrix A\n");
    for(i=0 ; i<Row1 ; i++)
    {
      for(j=0 ; j<Column1 ; j++)
      {
       scanf("%d" , &Matrix A[i][j] );
      }
    }
    printf(" Enter the elements of Matrix B\n");
    for(i=0 ; i<Row2 ; i++)
     {
       for(j=0 ; j<Column2 ; j++)
       {
         scanf("%d" , &Matrix B[i][j] );
       }
     }
    for(i=0 ; i<Row1 ; i++)
     {
       for(j=0 ; j<Column2 ; j++)
       {
         Matrixproduct[i][j] = 0 ;
         for(k=0 ; k<Row2 ; k++)
         {
          Matrixproduct[i][j] = Matrixproduct[i][j] +(Matrix A[i][k] * Matrix B[k][j] );
         }
       }
     }
    printf(" Product Matrix\n");
     for(i=0 ; i< Row1 ; i++)
     {
       for(j=0 ;j< Column2;j++)
       {
         printf("%d" , Matrixproduct[i][j] );
        }
       printf("\n");
     }
  }
  else
    printf(" Invalid order so Multiplication not possible\n");
}
```

OUTPUT:

```
Enter the order of Matrix A
 2 * 2
 Enter the order of MatrixB
 2 * 2
 Enter the elements of Matrix A
 1
 2
 3
 4
 Enter the elements of Matrix B
 5
 6
 7
 8
 Product Matrix
 19      22
 43      50
```

RESULT:

Thus the C program for finding the product of two matrices of any order has been successfully executed and verified.

Ex. No. 11 READ 'N' NUMBER OF STUDENTS WITH 5 SUBJECT MARKS

AIM:

To write a program for reading 'N' number of students with 5 subject marks.

ALGORITHM:

```
Step1: Start the program.
Step2: Initialize a character array n and integer array r and s.
Step3: Initialize integer i, j and n.
Step3: Read the value of n.
Step4: for(i=0;i<n;i++)
    a) Enter rollno,name,,,,,,
    b) Read these and enter 5 subject marks using for loop and array.
Step5: Display n[i],r[i],s[i][j]
Step6: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char n[20][10];
        int i,j,r[20],s[20][6];
        printf("Enter n value: ");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("Enter name,rollno,....");
                scanf("%s%d",&n[i],&r[i]);
                printf("Enter 5 subject marks:");
                s[i][5]=0;
                for(j=0;j<5;j++)
                {
                        scanf("%d",s[i][j]);
                        s[i][5]=s[i][5]+s[i][j];
                }
        }
        printf("The data entered is: \n");
        for(i=0;i<n;i++)
        {
                printf("%s\t%d\t",n[i],r[i]);
                for(j=0;j<5;j++)
                        printf("%d\t",s[i][j]);
        }
getch();
}
```

OUTPUT:

```
Enter n value: 1
Enter name,rollno,….Eswar 20
Enter 5 subject marks:
10 50 34 06 42
The data entered is:
Eswar 20            10      50      34      06      42
```

RESULT:

Thus the C program for reading ‘N’ number of students with 5 subject marks has been successfully executed and verified.

Ex. No. 12 **GREATEST OF 'N' NUMBERS USING FUNCTION**

AIM: To write a program for finding greatest of 'n' numbers using function.

ALGORITHM:

```
Step1: Start the program.
Step2: Initialize integer a, b and c.
Step3: Read the value of a,b and c.
Step4: Call the function large().
        a) Check if((a>b) && (a>c)) then print a is greater.
        b) Check elseif (b>c) then print b is greater.
        c) Check else print c is greater.
Step5: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a,b,c;
        printf(" Enter the value of a,b and c: ");
        scanf("%d, %d, %d", &a, &b, &c);
        large(a,b,c);
        getch();
}

large(int a, int b, int c)
{
  if((a>b) && (a>c))
        print("%d is greater than %d, %d", a, b, c);
  elseif (b>c)
        print("%d is greater than %d, %d", b, a, c);
else
        print("%d is greater than %d, %d", c, a, b);
}
```

OUTPUT:

```
Enter the value of a,b and c: 10 30 20
30 is greater than 10, 20
```

RESULT:

Thus the C program for finding greatest of 'n' numbers using function has been successfully executed and verified.

Ex. No. 13 **FIBONACCI SERIES USING RECURSION**

AIM:

To write a program for finding Fibonacci series using recursion.

ALGORITHM:

```
Step1: Start the program.
Step2: Initialize a function as int Fibonacci(int).
Step3: Initialize integer i=0, c and n in main function.
Step3: Read the value of n.
Step4: Within for loop call the Fibonacci(int) recursively.
Step5: In Fibonacci(int) function calculate ( Fibonacci(n-1) + Fibonacci(n-2) )
recursively and return the value.
Step6: Print the result.
Step7: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
 int Fibonacci(int);
 int main()
        {
                int n, i = 0, c;
                printf("Enter the n value: ");
                scanf("%d",&n);
                printf("Fibonacci series\n");
                for ( c = 1 ; c <= n ; c++ )
                {
                        printf("%d\n", Fibonacci(i));
                        i++;
                }
                return 0;
        }
 int Fibonacci(int n)
{
  if ( n == 0 )
    return 0;
  else if ( n == 1 )
    return 1;
  else
    return ( Fibonacci(n-1) + Fibonacci(n-2) );
}
```

OUTPUT:

```
Enter the n value: 9
Fibonacci series: 0 1 1 2 3 5 8 13 21
```

RESULT:

Thus the C program for finding Fibonacci series using recursion has been successfully executed and verified.

Ex. No. 14 LOWER CASE TO UPPERCASE CHARACTERS

AIM: To write a program for converting all lower case to uppercase characters.

ALGORITHM:

```
Step1: Start the program.
Step2: Take a string a function of return value data type is void str upper.
Step3: Read a string.
Step4: While (s[i]! ='\0') the do the following
        a)if((s[i]>='a') &&(s[i]<='z'))
        b)s[i]=s[i]-32;
        c)i++;
Step5: Display changed string.
Step6: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char str;
        printf("Enter a string: ");
        scanf("%s",str);
        to_str_upper(char[]);
        printf("Changed to: %s",str);
}
void to_str_upper(char[])
{
        int i=0;
        while(s[i]!='\0')
        {
                if((s[i]>='a') && (s[i]>='z'))
                s[i]=s[i]-32;
                i++;
        }
}
```

OUTPUT:

```
Enter a string : gnec
changed to: GNEC
```

RESULT:

Thus the C program for converting all lower case to uppercase characters has been successfully executed and verified.

Ex. No. 15 **SORT 5 CITY NAMES IN ALPHABETICAL ORDER**

AIM:

To write a program for sorting 5 city names in alphabetical order.

ALGORITHM:

```
Step1: Start the program.
Step2: Using for loop and array get the city name.
Step3: Using loop for(i=65;i<122;i++) and for(j=0;j<5;j++)
    a)  Check if(city[j][0]==i)
    b)  Display the sorted list of cities.
Step4: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        ch city[5][20];
        int i,j;
        clrscr();
        printf("Enter the names of cities...\n\n");
        for(i=0;i<5;i++)
                scanf("%s",&city[i]);
        printf("Sorted list of cities...\n\n");
        for(i=65;i<122;i++)
        {
                for(j=0;j<5;j++)
                {
                        if(city[j][0]==i)
                        printf("\n%s",city[j]);
                }
        }}
```

OUTPUT:

```
Enter the names of cities: Hyderabad Chennai Bombay Goa Vizag
Sorted list of cities:
Bombay
Chennai
Goa
Hyderabad
Vizag
```

RESULT: Thus the C program for sorting 5 city names in alphabetical order has been successfully executed and verified.

Ex. No. 16 **EXTRACTS THE PART OF A STRING**

AIM:

To write a program for extracting the part of a string.

ALGORITHM:

```
Step1: Start the program.
Step2: Declare the character array s[30] and r[30].
Step3: Declare the integer variables i, j, m & n.
Step4: Get the input string using gets().
Step5: Get the value of m and n for extracing from the input string.
Step6: Initialize j=0.
Step7: Using a loop for(i=n-1;i<m+n-1;i++)
    a)  Assign r[j]=s[i];
    b)  Increment J by 1.
Step8: Print the extracted part of the string.
Step9: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
#include<string.h>
void main()
{
  char s[30],r[30];
  int i,j,m,n;
  clrscr();
  printf("Enter a string: ");
  gets(s);
  printf("Enter the values of m & n: ");
  scanf("%d%d",&m,&n);
  j=0;
  for(i=n-1;i<m+n-1;i++)
   {
    r[j]=s[i];
    j++;
   }
  printf("The extracted part of string %s: ",r);
  getch();
}
```

OUTPUT:

```
Enter a string: Gurunanak
Enter the values of m & n: 3 5
The extracted part of string: run
```

RESULT:

Thus the C program for extracting a part from the given string was executed and verified.

Ex. No. 17 **CALL BY VALUE**

AIM:

To write a program to increment the value of an argument using call by value.

ALGORITHM:

```
Step1: Start the program.
Step2: Declare the integer variable x and a integer function incr()
Step3: Initialize x=7.
Step4: Pass the x value to the function incr(x).
    a)  Within the function increment the x value by 1.
    b)  Return the value.
Step5: Print the original value and incremented value of x.
Step6: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
#include<string.h>
main()
  {    int x;
       int incr(int n);
       printf("***Call by Value***\n");
       x = 7;
       printf("Original value of x is: %d/n: ", x);
       printf("Value of incr(x) is: %d/n ", incr(x));
       printf("The value of x is: %d/n: ", x);
  }

  /* Function increments n */
  int incr(int n)
  {
       n = n + 1;
       return n;
  }
```

OUTPUT:

```
Original value of x is: 7
Value of incr(x) is : 8
The value of x is: 7
```

RESULT:

Thus the C program to increment the value of an argument using call by value was executed and verified.

Ex. No. 18 **CALL BY REFERENCE**

AIM:

To write a program for swapping two values using call by reference method.

ALGORITHM:

Step1: Start the program.
Step2: Assign the integer variable a=10 and b=20.
Step3: Call the swap() function.
Step4: Swap the values using pointer.
Step5: Print the original value and swapped value of a & b.
Step6: Stop the program.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void swap( int *x, int *y )
{
   int t ;
   t = *x ;
   *x = *y ;
   *y = t ;
   printf( "\nx = %d y = %d", *x,*y);
}

int main( )
{
   int a = 10, b = 20 ;
   swap ( &a, &b ) ;
   printf ( "\na = %d b = %d", a, b ) ;
   getch();
}
```

OUTPUT:

```
a=10 b=20
x=20 y=10
```

RESULT:

Thus the C program to swap two values using call by reference method was executed and verified.

Ex. No. 19(a) **STRUCTURE**

AIM:

To write a program for displaying student information by initializing structures.

ALGORITHM:

Step1: Start the program.
Step2: Initialize a structure student with name as character array and roll number and age as integer.
Step3: In the main program crate a object s1 for the structure student.
Step4: Using the object s1 print the student name, roll number and age.
Step6: Stop the program.

PROGRAM:

```
#include<stdio.h>
struct student
{
        char name[10];
        int rollno;
        int age;
};
main()
{
        static struct student s1;
        clrscr();
        printf("Enter the name, rollno & age");
        scanf("%s%d%d\n",&s1.name,&s1.rollno,&s1.age);
        printf("%s %d %d",s1.name,s1.rollno,s1.age); getch();
}
```

OUTPUT:

```
Enter name, rollno & age
Ravi 11 25
Ravi 11 25
```

RESULT:

Thus the C program to display student information by initializing structures was executed and verified.

Ex. No. 19(b) **UNION**

AIM:

To write a program for implementing the concept of union data type.

ALGORITHM:

```
Step1: Start the program.
Step2: Initialize a union Data with Str as character array, i as integer and f as float.
Step3: In the main program crate a variable name data for the union Data.
Step4: Using the variable and member access operator print all the members of the union
Data.
Step5: Stop the program.
```

PROGRAM:

```
#include <stdio.h>
#include <string.h>
 union Data
{
   int i;
   float f;
   char str[20];
};

int main( )
{
   union Data data;
   data.i = 10;
   printf( "data.i : %d\n", data.i);
   data.f = 220.5;
   printf( "data.f : %f\n", data.f);
   strcpy( data.str, "C Programming");
   printf( "data.str : %s\n", data.str);
   return 0;
}
```

OUTPUT:

```
data.i : 10
data.f : 220.500000
data.str : C Programming
```

RESULT:

Thus the C program to implement the concept of union data type was executed and verified.

Ex. No. 20 ACCESS THE VALUE OF VARIABLES USING POINTER

AIM:

To write a program for accessing the value of variables using pointer.

ALGORITHM:

```
Step1: Start the program.
Step2: Declare integer as a, b, c and two pointer variables *p1 & *p2.
Step3: Intialize a=12 and b=4.
Step4: Assign the a & b values to the pointer variables p1 & p2.
Step5: Perform arithematic operations.
Step6: Print the adderss of a & b and print the a, b, c, x & y values.
Step7: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
main()
{
  int a,b,*p1,*p2,x,y,z;
  clrscr();
  a=12,b=4;
  p1=&a; p2=&b;
  x=*p1**p2-6;
  y=(4-*p2)**p1+10;
  printf("Address of a=%d\n",p1);
  printf("Address of b=%d\n",p2);
  printf("a=%d,b=%d\n",a,b);
  printf("x=%d,y=%d\n",x,y);
  *p2=*p2+3; *p1=*p2-5;
  z=*p1**p2-6;
  printf("a=%d,b=%d\n",a,b);
  printf("z=%d\n",z);
  getch();
}
```

OUTPUT:

```
Address of a = 65543

Address of b = 64455
a = 12 b = 4
x =       y =
z=42
```

RESULT:

Thus the C program to access the value of variables using pointer was executed and verified.

Ex. No. 21 **PRINT THE ELEMENT OF ARRAY USING POINTERS**

AIM:

To write a program for printing the element of array using pointers.

ALGORITHM:

```
Step1: Start the program.
Step2: Declare integer array a[5] and a pointer variable *p=&[0]
Step3: Intialize i as integer.
Step4: Using the for loop for(i=0;i<5;i++)
Step5: prtint the value of *(p+i).
Step6: Then using the for loop for(i=0;i<5;i++)
Stop7: Print the value of (p+1).
Step7: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
main()
{
  int a[5]={5,4,6,8,9};
  int *p=&a[0];
  int i;
  clrscr();
  for(i=0;i<5;i++)
     printf("%d",*(p+i));
  for(i=0;i<5;i++)
     printf(" %u\n",(p+i));
  getch();
}
```

OUTPUT:

```
1 2 3 4 5

1 2 3 4 5
```

RESULT:

Thus the C program to print the element of array using pointers was executed and verified.

Ex. No. 22 **PRINT THE ELEMENTS OF A STRUCTURE USING POINTERS**

AIM:

To write a program printing the elements of a structure using pointers.

ALGORITHM:

```
Step1: Start the program.
step2:   Take a character array name, a number and price in structure
step3:   In main take a struct variable product and a pointer
Step4: Using a loop       for(*ptr=product;ptr<product+3;ptr++)
Step5: Read the value by using array operator
           ptr->name,ptr->no,ptr->price
step6: Display name,no,price.
Step7: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
struct invest
{
   char name[20];
   int number;
   float price;
};
main()
{
  struct invest product[3],*ptr;
  clrscr();
  printf("input\n\n");
  for(*ptr=product[3];ptr<product+3;ptr++)
    scanf("%s%d%f",&ptr->name,&ptr->number,&ptr->price);
  printf("\nResult: \n\n");
  ptr=product;
  while(ptr<product+3)
   {
     printf("%20s%5d%10.2f\n",ptr->name,ptr->number,ptr->price);
     ptr++;
   }
  getch();
}
```

OUTPUT:

```
Raja
11
120
Result:
Raja
11
120
```

RESULT:

Thus the C program to print the elements of a structure using pointers was executed and verified.

Ex. No. 23 DISPLAY COLLEGE ADDRESS USING STRUCTURES AND POINTERS AIM:

To write a program for displaying college address using structures and pointers.

ALGORITHM:

Step1: Start the program.
Step2: Take name, location and city inside the collegeaddress structure.
Step3: Enter the required data.
Step4: Print the result.
Step5: Stop the program.

PROGRAM:

```
#include<stdio.h>
struct collegeaddress
{
        char name[20],location[20],city[20];
};
main()
{
        struct collegeaddress add,*ptr;
        p=&add;
        p->name={"Annamalai University"};
        p->location={"Annamalainagar"};
        p->city={"Chidambaram"};
        printf("%s%s%s",p->name,p->location,p->city);
}
```

OUTPUT:

```
Annamalai University Annamalainagar Chidambaram
```

RESULT:

Thus the C program to display college address using structures and pointers was executed and verified.

Ex. No. 24 PASS STRUCTURE AS ARGUMENT TO FUNCTION

AIM:

To write a program for passing structure as argument to function and calculate total marks of 5 subjects.

ALGORITHM:

```
Step1: Start the program.
Step2: Inside the structure ex2 declare 6 integers.
Step3: Declare structure ex2 as s1.
Step4: Declare structure ex2 as s2,ex2 as fun().
Step5: Display the message as enter the marks.
Step6: Take value of the subjects from the user.
Step7: Store the return value in s2.total.
Step8: Print the value of s2.total.
Step9: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
struct ex2
{
  int m1,m2,m3,m4,m5,total;
};
main()
{
  struct ex2 s1;
  struct ex2 s2;
  struct ex2 fun();
  printf("enter the marks");
  scanf("%d%d%d%d%d",&s1.m1,&s1.m2,&s1.m3,&s1.m4,&s1.m5);
  s2=fun(s3);
  printf("%d",s1.total);
}
struct ex2 fun(s3)
struct ex2 s3;
{
  s3.total=s3.m1+s3.m2+s3.m3+s3.m4+s3.m5;
  return(s3);
}
```

OUTPUT:

```
Enter the marks
10 20 30 40 50
150
```

RESULT:

Thus the C program to pass structure as argument to function and calculate total marks of 5 subjects was executed and verified.

Ex. No. 25 **WRITE INTEGER DATA INTO FILE AND READ IT FROM FILE**

AIM:

To write a program for writing integer data into file and read it from file.

ALGORITHM:

```
Step1: Start the program.
Step2: Initialize integer num.
Step3: Declare FILE *f2.
Step4: Open the file f2 using fopen() in write mode.
Step5: Get the integer from user and write it into the file using putw().
Step6: Close the file.
Step7: Open the file f2 using fopen() in read mode.
Step8: Read the integer using getw().
Step9: Print the integer.
Stop10: Close the file.
Step11: Stop the program.
```

PROGRAM:

```
#include<stdio.h>
main()
{
  int num;
  FILE *f2;
  f2=fopen("data.int","w");
  scanf("%d",&num);
  putw(num,f2);
  fclose(f2);
  f2=fopen("data.int","r");
  num=getw(f2);
  printf("%d",num);
  fclose(f2);
}
```

OUTPUT:

```
12
12
```

RESULT:

Thus the C program to write integer data into file and read it from file was executed and verified.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Course Description

Title of Course: Business Presentation & Languages Lab
Course Code: HU191
L-T Scheme: 0-0-4 **Course Credits: 2**

Introduction:

This course can enhance the drafting and understanding skills of engineering students.

Objectives:

1. This Course has been designed to impart advanced skills of Technical Communication in English through Language Lab. Practice Sessions to 1STSemester UG students of Engineering &Technology.
2. To enable them to communicate confidently and competently in English Language in all spheres.

Learning Outcomes:

Knowledge:

1. This course will help the students to learn English very easily. Even the Hindi medium students can translates easily.
2. The technical communication will help the students to improve their speaking skills and drafting skill for engineering students.

Course Contents:

Unit 1: ENGLISH LANGUAGE GRAMMAR-Correction of Errors in Sentences Building Vocabulary Word formation Single Word for a group of Words Fill in the blanks using correct Words Sentence Structures and Transformation Active &Passive Voice Direct &Indirect Narration (MCQ Practice during classes).

Unit 2: READING COMPREHENSION-Strategies for Reading Comprehension Practicing Technical & Non Technical Texts for Global/Local/Inferential/Referential comprehension; Précis Writing

Unit 3: TECHNICAL COMMUNICATION-the Theory of Communication–Definition & Scope Barriers of Communication Different Communication Models Effective Communication (Verbal/Nonverbal) Presentation / Public Speaking Skills (MCQ Practice during classes)

Unit 4: MASTERING TECHNICAL COMMUNICATION- Technical Report (formal drafting) Business Letter (formal drafting) Job Application (formal drafting) Organizational

Unit 5: GROUP DISCUSSION–Principle & Practice

Text Books

1. Board of Editors: Contemporary Communicative English for Technical Communication Pearson Longman, 2010
2. Technical Communication Principle sand Practice by Meenakshi Raman, Sangeeta Sharma (Oxford Higher Education)
3. Effective Technical Communication by Barun K. Mitra (Oxford Higher Education).
4. P C WREN & H.MARTIN (English language & grammar)

References

1. D.Thakur: Syntax Bharati Bhawan, 1998
2. Longman Dictionary of Contemporary English (New Edition) for Advanced Learners
3. Internet