UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Numerical Methods **Subject Code-M(CS))401**
Year: 2nd Year **Semester: Fourth**

Module Number	Topics	Number of Lectures
1	**Approximation in numerical computation:**	**4L**
	Approximation of numbers	1
	Types of errors	2
	Calculation of errors	1
2	**Interpolation:**	**6L**
	Finite differences	1
	Newton forward/backward interpolation	2
	Lagrange's method	1
	Newton's divided difference Interpolation	2
3.	**Numerical integration:**	**3L**
	Trapezoidal rule	2
	Simpson's 1/3 rule	1
4	**Numerical solution of a system of linear equations:**	**6L**
	Gauss elimination method	1
	Matrix inversion	1
	LU Factorization method	2
	Gauss-Seidel iterative method	2
5	**Numerical solution of Algebraic equation:**	**5L**
	Bisection method	2
	Regula-Falsi method	1
	Newton-Raphson method	2
6	**Numerical solution of ordinary differential equation:**	**8L**
	Euler's method	2
	Runge-Kutta methods	2
	Predictor- Corrector methods	2
	FiniteDifference method	2

Assignment:

Module-1:

1. Find the relative error if 2/3 is approximated to 0.667.
2. Find the percentage error if 625.483 is approximated to three significant figures.
3. Find the relative error in taking ® = 3.141593 as 22/7.
4. The height of an observation tower was estimated to be 47 m, whereas its actual height was 45 m. calculate the percentage relative error in the measurement.

5. Two numbers are 3.5 and 47.279 both of which are correct to the significant figures given. Find their product.

Module-2:

1. Apply Newton's backward Interpolation to the data below, to obtain a polynomial of degree4 in x

x:	1	2	3	4	5
$f(x)$:	1	-1	1	-1	1

2. Using Newton's backward Interpolation, find the value of $f(2)$ from the following table:

x:	1	3	4	5	6	7
$f(x)$:	2.68	3.04	3.38	3.68	3.96	4.21

3. Using Newton's Forward Interpolation, the area A of a circle of diameter d.

d:	80	85	90	95	100
A:	5026	5674	6362	7088	7854

Calculate the area of a circle of diameter 105.

4. Estimate the value of $f(22)$ and $f(42)$ from the following available data:

x:	20	25	30	35	40	45
$f(x)$:	354	332	291	260	231	204

Using Newton's Forward Interpolation

5. Find $f(x)$ as a polynomial in x for the following data by Newton's divided difference method:

x:	-4	-1	0	2	5
$f(x)$:	1245	33	5	9	1335

6. Using Newton's divided difference method to find $f(x)$ from the following available data:

x:	0	1	2	4	5	6
$f(x)$:	1	14	15	5	6	19.

Module-3:

1. Apply trapezoidal rule to find the integral $I = \int_0^1 \sin^{®} x\,dx.$
2. Find, from the following table the area bounded by the curve and the x-axis from $x = 7.47$ to $x = 7.52$,
   $f(7.47) = 1.93,\ f(7.48) = 1.95,\ f(7.49) = 1.98,\ f(7.50) = 2.01,$
   $f(7.51) = 2.03,\ f(7.52) = 2.06.$
3. Evaluate $I = \int_0^1 \frac{1}{1+x^2}\,dx,$ correct to three decimal places and also find the approximate value of ®.
4. A solid of revolution is formed by rotating about the x-axis the area between the x-axis, the lines $x = 0$ and $x = 1$ and a curve through the points with the following coordinates:
   $(0,1),(0.25,0.9896)(0.5,0.9589),(0.75,0.9089),(1,0.8415).$

Module-4:

1. Solve the following system of equations:

$4x + y + z = 4$

$x + 4y - 2z = 4$,

$3x + 2y - 4 = 6$

by matrix-inversion method.

2. Solve the above system by matrix-inversion method:

   $x + y - z = 2$

   $2x + 3y + 5z = -3$

   $3x + 2y - 3z = 6$

3. The following system of equations are given:

   $4x + y + z = 4$

   $x + 4y - 2z = 4$,

   $3x + 2y - 4 = 6$

   Solve the above system by LU decomposition method.

4. Solve the given system of equations by LU decomposition method:

   $x + y - z = 2$

   $2x + 3y + 5z = -3$

   $3x + 2y - 3z = 6$

Module-5:

1. Find the root of the following equations correct three decimal places by the Regulafalsi method: $x^3 + x - 1 = 0$.
2. Using Regulafalsi method, compute the real root of the following equation correct to four decimal places: $xe^x = 2$.
3. Find the root of the following equations correct three decimal places by the Regulafalsi method: $x^6 - x^4 - x^3 - 1 = 0$.
4. Find the root of the following equations correct three decimal places by the bisection method : $x - e^x = 0$
5. Find the root of the following equations, using the bisection method correct three decimal places: $x - \cos x = 0$
6. Using the bisection method to find a root of the equation to four decimal places: $x^3 - 9x + 1 = 0$

Module-6:

1. Using Runge-kutta method of order 4, find $y(0.2)$ given that $\frac{dy}{dx} = 3x + \frac{1}{2}y, y(0) = 1$ taking $h = 0.1$.
2. Using Runge-kutta method of order 4, compute $y(0.2)$ and $y(0.4)$ from $10\frac{dy}{dx} = x^2 + y^2, y(0) = 1$ taking $h = 0.1$.
3. Using Milne's predictor-corrector method to obtain the solution of the equation $\frac{dy}{dx} = x - y^2$ at $x = 0.8$ given that $y(0) = 0.0000$, $y(2) = 0.0200$, $y(4) = 0.0795$ $y(6) = 0.1762$.

4. Given $2\frac{dy}{dx} = (1+x^2)y^2$ and $y(0)=1$, $y(0.1)=1.06$, $y(0.2)=1.12$, $y(0.3)=1.21$, evaluate $y(0.4)$ by Milne's predictor-corrector method.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Basic Environmental Engineering** Subject Code: **CH-401**

Year: **2nd Year** Semester: **Fourth**

Module Number	Topics	Number of Lectures
1	**Chapter 1: General**	**6L**
	1. Basic ideas of environment, basic concepts, man, society & environment, their interrelationship.	1L
	2. Mathematics of population growth and associated problems, Importance of population study in environmental engineering, definition of resource, types of resource, renewable, non-renewable, potentially renewable, effect of excessive use vis-à-vis	2L
	3. Materials balance: Steady state conservation system, steady state system with non conservative pollutants, step function.	1L
	4. Environmental degradation: Natural environmental Hazards like Flood, earthquake, Landslide-causes, effects and control/management; Anthropogenic degradation like Acid rain-cause, effects and control. Nature and scope of Environmental Science and Engineering.	2L
	Chapter 2: Ecology	**6L**
	1. Elements of ecology: System, open system, closed system, definition of ecology, species, population, community, definition of ecosystem-components types and function.	1L
	2. Structure and function of the following ecosystem: Forest ecosystem, Grassland ecosystem, Desert ecosystem, Aquatic ecosystems, Mangrove ecosystem (special reference to Sundar ban); Food chain [definition and one example of each food chain], Food web.	2L
	3. Biogeochemical Cycle- definition, significance, flow chart of different cycles with only elementary reaction [Oxygen, carbon, Nitrogen, Phosphate, Sulphur].	1L
	4. Biodiversity- types, importance, Endemic species, Biodiversity Hot-spot, Threats to biodiversity, Conservation of biodiversity.	2L
	Chapter 3: Air pollution and control	**7L**
	1. Atmospheric Composition: Troposphere, Stratosphere, Mesosphere, Thermosphere, Tropopause and Mesopause	1L
	2. Energy balance: Conductive and Convective heat transfer, radiation heat transfer, simple global temperature model [Earth as a black body, earth as albedo], Problems.	1L
	3. Green house effects: Definition, impact of greenhouse gases on the global climate and consequently on sea water level, agriculture and marine food. Global warming and its consequence, Control of Global warming. Earth's heat budget.	1L
	4. Lapse rate: Ambient lapse rate Adiabatic lapse rate, atmospheric stability, temperature inversion (radiation inversion). Atmospheric dispersion: Maximum mixing depth, ventilation coefficient, effective stack height, smokestack plumes and Gaussian plume model.	1L

	5. Definition of pollutants and contaminants, Primary and secondary	1L
	pollutants: emission standard, criteria pollutant. Sources and effect of different air pollutants- Suspended particulate matter, oxides of carbon, oxides of nitrogen, oxides of sulphur, particulate, PAN.	
	6. Smog, Photochemical smog and London smog. Depletion Ozone layer: CFC, destruction of ozone layer by CFC, impact of other green house gases, effect of ozone modification.	1L
2	7. Standards and control measures: Industrial, commercial and residential air quality standard, control measure (ESP. Cyclone separator, bag house, catalytic converter, scrubber (ventury), Statement with brief reference).	1L
	Chapter 4: Water Pollution and Control	**8L**
	1. Hydrosphere, Hydrological cycle and Natural water.	1L
	2. Pollutants of water, their origin and effects: Oxygen demanding wastes, pathogens, nutrients, Salts, thermal application, heavy metals, pesticides, volatile organic compounds.	2L
	3. River/Lake/ground water pollution: River: DO, 5 day BOD test, Seeded BOD test, BOD reaction rate constants, Effect of oxygen demanding wastes on river[deoxygenation, reaeration], COD, Oil, Greases, pH.	1L
	4. Lake: Eutrophication [Definition, source and effect]. Ground water: Aquifers, hydraulic gradient, ground water flow (Definition only)	1L
	5. Standard and control: Waste water standard [BOD, COD, Oil, Grease], Water Treatment system [coagulation and flocculation, sedimentation and filtration, disinfection, hardness and alkalinity, softening] Waste water treatment system, primary and secondary treatments [Trickling filters, rotating biological contractor, Activated sludge, sludge treatment, oxidation ponds] tertiary treatment definition.	2L
	6. Water pollution due to the toxic elements and their biochemical effects: Lead, Mercury, Cadmium, and Arsenic	1L
3	**Chapter 5: Land Pollution**	**3L**
	1. Lithosphere; Internal structure of earth, rock and soil	1L
	2. Solid Waste: Municipal, industrial, commercial, agricultural, domestic, pathological and hazardous solid wastes; Recovery and disposal method- Open dumping, Land filling, incineration, composting, recycling. Solid waste management and control (hazardous and biomedical waste).	2L
	Chapter 5: Noise Pollution	**2L**
	1. Definition of noise, effect of noise pollution, noise classification [Transport noise, occupational noise, neighbourhood noise]	1L
	2. Definition of noise frequency, noise pressure, noise intensity, noise threshold limit value, equivalent noise level, $L10$ (18 hr Index) , $n\ Ld$, Noise pollution control.	1L
	Chapter 6: Environmental Management	**2L**
	1. Environmental impact assessment, Environmental Audit, Environmental laws and protection act of India, Different international environmental treaty/ agreement/ protocol.	2L
Total Number Of Hours = 34L		

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Basic Environmental Engineering** Subject Code: **CH-401**
Year: **2nd Year** Semester: **Fourth**

Faculty In-Charge HOD, ME Dept.

Assignment:

Module-1.
1. Write short notes for the following:
(a) Flood (b) Landslides (b) Earthquake (c) Acid Rain

2. Suppose an anemometer at a height of 40 m above ground measure wind velocity =5.5 m/s. Estimate the wind speed at an elevation of 500 m in rough terrain if atmosphere is unstable (i.e., k = 0.2).

Module-2.
1. A BOD test is run using 50 ml of wastewater mixed with 100 ml of pure water. The initial DO of the mixture is 6 mg/l and after 5 days it becomes 2 mg/l. After a long time, the DO remains fixed at 1 mg/l.
(i)What is the 5 days BOD (BOD_5)?
(ii)What is the ultimate BOD (BOD_u)?
(iii)What is the remaining BOD after 5 days?
(iv)What is the reaction rate constant measured at 20°C?
(v)What would be the reaction rate if measured at 35°C?

2. Draw the flow diagram for the following (a) Surface water treatment (b) Waste water Treatment.

3. Draw the Oxygen sag curve.

Module-3.
1. a) If two machines produces sounds of 80 dB and 120 dB simultaneously, what will be the total sound level.
b) Calculate the intensity of 100 dB sounds.

2. Write a report on the environmental problems related to an abandoned airport. Mention various measures by which it can be used again for other purposes.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Communication Engineering & Coding Theory **Subject code: CS401**
Year: 2nd Year **Semester: Fourth**

Module Number	Topics	Number of Lectures
1.	**Elements of Communication system, Analog Modulation &Demodulation, Noise, SNR Analog toDigital Conversion:**	**8L**
	1. Introduction to Base Band transmission & Modulation (basic concept)	1
	2. Elements of Communication systems (mention of transmitter, receiver and channel); origin of noise and its effect, Importance of SNR in system design.	1
	3. Basic principles of Linear Modulation (Amplitude Modulation)	1
	4. Basic principles of Non-linear modulation (Angle Modulation - FM, PM)	1
	5. Sampling theorem, Sampling rate, Impulse sampling, Reconstruction from samples, Aliasing	1
	6. Analog Pulse Modulation - PAM (Natural & flat topped sampling), PWM, PPM	1
	7. Basic concept of Pulse Code Modulation, Block diagram of PCM	1
	8. Multiplexing - TDM, FDM	1
2.	**Digital Transmission:**	**8L**
	1. Generation of AM: Concept of Gated, Square law modulators, Switching modulator Concept of Quantization & Quantization error, Uniform Quantize	1
	2. Non-uniform Quantizer, A-law & µlaw companding (mention only)	1
	3. Encoding, Coding efficiency ,Line coding & properties, NRZ & RZ, AMI, Manchester coding PCM, DPCM	2
	4. Baseband Pulse Transmission, Matched filter (mention of its importance and basic concept only), Error rate due to noise	2
	5. ISI, Raised cosine function, Nyquist criterion for distortion-less base-band binary transmission, Eye pattern, Signal power in binary digital signals	2
	Digital Carrier Modulation & Demodulation Techniques:	**8L**

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

3.	1. Bit rate, Baud rate; Information capacity, Shanon's limit	2
	2. M-ary encoding, Introduction to the different digital modulation techniques - ASK, FSK, PSK, BPSK, QPSK, mention of 8 BPSK, 16 BPSK	2
	3. Introduction to QAM, mention of 8QAM, 16 QAM without elaboration	1
	4. Delta modulation, Adaptive delta modulation (basic concept and importance only, no details.	1
	5. introduction to the concept of DPCM, Delta Modulation, Adaptive Delta modulation and their relevance	1
	6. Spread Spectrum Modulation - concept only	1
4.	**Information Theory & Coding:**	**8L**
	1. Introduction, News value & Information content (1L);, Entropy	2
	2. Mutual information; Information rate; Shanon-Fano algorithm for encoding	2
	3. Shannon's Theorem - Source Coding Theorem; Channel Coding Theorem, Information Capacity Theorem (basic understanding only)	2
	4. Error Control & Coding - basic principle only	2
Total Number Of Hours = 32		

Signature of Faculty **Signature of HOD**

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Assignment:

Module-1:

1. **Explain communication system with proper block diagram.**
2. **Explain the need for modulation in a communication system.**
3. **Explain how AM is generated?**
4. **Explain the concept of multiplexing.**
5. **Explain the working principle of TDM/FDM.**

Module-2:

1. **What is companding? Why it is needed?**
2. **Explain uniform and non-uniform quantization?**
3. **What is Nyquist criterion for zero intersymbol interference?**
4. **What is the function of raised cosine function?**
5. **What are limitations of delta modulation?**
6. **What is quantization noise?**
7. **Given an input data: 010110, Represent it using the line codes.**

Module-3:

1. **What is the difference between MSK and QPSK?**
2. **With Proper equations and signal constellation explain the concept of Quadri phase shift Keying.**
3. **Explain the concept of OFDM.**
4. **Draw an eye diagram and mention the significance of its different parts.**

Module-4:

1. **State and Prove source coding theorem**
2. **Define information.**
3. **Suppose the six symbols of the alphabet *S* and their probabilities are given by $S=\{S_0,S_1,S_2,S_3,S_4,S_5\}$ with probabilities $P_0=1/16$, $P_1=1/16$, $P_2=1/8$, $P_3=1/4$, $P_4=1/4$, $P_5=1/4$. Calculate entropy, $H(S)$ and use Huffman coding to code each symbols, average length code and efficiency.**
4. **Mention the properties of mutual information.**

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Software Engineering **Subject Code-CS402**
Year: 2nd Year **Semester: Fourth**

Module Number	Topics	Number of Lectures
	Module I	**3L**
1	1. Interactive Systems, Usability, Introduction to software engineering, Software Process Models,	1
	2. Software Life Cycle Models, Classical Waterfall Models	1
	3. Iterative Waterfall Models, Prototype Model, Spiral Model, Comparison of different Waterfall Models.	1
2	**Module II**	**3L**
	1. Requirement Gathering and Analysis	1
	2. Requirement Specification, SRS	1
	3. Formal System Specification	1
2	**Module III**	**4L**
	1. Design Process, Coupling, Cohesion, Overview of Function Oriented Design, and Object Oriented Design.	1
	2. SA/SD methodology, Structured Analysis, DFD	1
	3. System Design – Problem Partitioning, Top-Down and Bottom-Up design; Decision tree, decision table and structured English;	1
	4. Functional vs. DFD, Data Dictionary, ER diagram, Process Organization & Interactions.	1
	Module IV	**4L**
	1. UML: Use case diagram,	1
	2. Class Diagram, Sequence diagram,	1
	3. State diagram, Activity Diagram	1
	4. Collaboration diagram, Deployment Diagram, Event trace diagram.	1
3	**Module IV**	**4L**
	1. Coding & Documentation - Structured Programming, OO Programming,	1
	2. Information Hiding, Reuse, System	1
	3. Testing – Levels of Testing, Integration Testing,	1
	1. Test case Specification, Reliability Assessment. , Validation & Verification	1
	2. Metrics, Monitoring & Control.	1
4	**Module V**	**6L**
	1. Software Project Management Project Estimation Techniques	1

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

	2. Staffing Level Estimation	
	3. Scheduling(PERT, Gantt, CPM)	1
	4. Software Configuration Management,	1
	5. Quality Assurance, Project Monitoring.	1
	6. CASE TOOLS: Concepts, use and application.	6
5	**Module V**	**6L**
	1. Characteristic of software maintenance , Types of software maintenance,	1
	2. Software reverse engineering, Basic issues in any reuse of program,	2
	3. CASE TOOLS: Concepts, use and application	1
Total Number Of Hours = 28		

Faculty In-Charge HOD, CSE Dept.

Assignment:

Module-I:

1. What are the difference between Waterfall Model and Spiral Model? What do you mean by Feasibility Analysis?
2. Explain Cost- Benefit Analysis, COCOMO model

Module-II:

1. What do you mean by System Requirement Specification?
2. Write short note on the followings:
   DFD
   Data Dictionary
   ER diagram
   Process Organization & Interactions

Module-III:

1. Explain the different levels of Testing. What is Integration Testing?
2. What do you mean by Reliability Assessment?

Module-IV:

1. How Software Project Management is done? What are the steps?
2. How the quality of any product should be assured?

Module-V:

1. What are the basic issues in any reuse of program?
2. Explain the differ types of software maintenance.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Computer Architecture **Subject Code: CS403**
Year: 2nd Year **Semester: Fourth**

Module Number	Topics	Number of Lectures
1	**Review of basic computer architecture**	**12**
	1. Quantitative techniques in computer design 2. measuring and reporting performance. 3. Pipelining: Basic concepts, instruction and arithmetic pipeline 4. data hazards, control hazards and structural hazards 5. techniques for handling hazards. Exception handling, Pipeline optimization techniques 6. Compiler techniques for improving performance	1 2 2 2 2 1 2
2	**Hierarchical memory technology**	**8**
	1. Inclusion, Coherence and locality properties 2. Cache memory organizations, Techniques for reducing cache misses 3. Virtual memory organization, mapping and management techniques 4. memory replacement policies	2 2 2 2
3	**Instruction-level parallelism**	**6**
	1. Basic concepts 2. Techniques for increasing ILP 3. Superscalar, super pipelined and VLIW processor architectures 4. Array and vector processors	1 1 2 2
4	**Multiprocessor architecture**	**12**
	1. taxonomy of parallel architectures 2. Centralized shared- memory architecture 3. synchronization, memory consistency 4. interconnection networks 5. Distributed shared-memory architecture 6. Cluster computers 7. Non von Neumann architectures – i. data flow computers ii. reduction computer architectures iii. systolic architectures	1 1 2 1 2 1 4
Total Lecture Hours – 38 l.h.		

Faculty In-Charge HOD, CSE Dept.

Assignments

Unit 1 :-

1. For the same program, two different compilers are used. The table below shows the execution time of the two different compiled programs.

	Compiler A		Compiler B	
	# instructions	Execution Time	# instructions	Execution Time
Program 1	1.00 E+09	1s	1.20 E+09	1.4s
Program 2	1.00 E+09	0.8s	1.20 E+09	0.7s

a) Find the average CPI for each program give n that the processor has a clock cycle time of 1ns.
b) Assume the average CPIs found in part (a), but that the compiled programs run on two different processors. If the execution times on the two processors are the same, how much faster is the clock of the processor running compiler A's code versus the clock of the processor running compiler B's code?
c) A new compiler is developed that uses only 600 million instructions and has an average CPI of 1.1. What is the speed-up of using this new compiler versus using Compiler A or B on the original processor of part (a)?

2. a)Name two RISC and two CISC processors. What are the main characteristics of RISC processors?
   b) Define (i) superscalar and (ii) super-pipeline concepts. Derive the equation for ideal speedup for a superscalar super-pipelined processor compared to a sequential processor. Assume N instructions, k-stage scalar base pipeline, superscalar degree of m, and super pipeline degree of n.

3. Consider the following MIPS assembly code:
   LD R1, 45(R2)
   ADD R7, R1, R5
   SUB R8, R1, R6
   OR R9, R5, R1
   BNEZ R7, target
   ADD R10, R8, R5
   XOR R2, R3, R4

a) Identify each type of data dependency; list the two instructions involved; identify which instruction is dependent; and, if there is one, name the storage location involved.
b) Use MIPS five-stage pipeline (fetch, decode, register, execute, write-back) and assume a register file that writes in the first half of the clock cycle and reads in the second half cycle. Which of the dependencies that you found in part (a) become hazards and which do not? Why?

4. Design a (very) simple CPU for an instruction set that contains only the following four instructions: lw (load word), sw (store word), add, and jump (unconditional branch). Assume that the instruction formats are similar to the MIPS architecture. If you assume a different format, state the instruction formats. Show all the components, all the links, and all the control signals in the datapath. You must show only the minimal hardware required to implement these four instructions. For each instruction show the steps involved and the values of the control signals for a single cycle implementation.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Computer Architecture **Subject Code: CS403**
Year: 2nd Year **Semester: Fourth**

Unit 2 : -

1. Briefly answer the following questions:
a) What is the difference between computer architecture and computer organization?
b) What are the two locality principles observed with respect to user programs? How are these principles exploited in computer design?
c) What is virtual memory? Explain the need for virtual memory.
d) What are the main differences between a multi-processor system and a multi-computer system?
e)In a shared memory system, explain two schemes to maintain cache-coherence.

2. Your operating system uses 8 kbyte pages. The machine you are using has a 4-way set associative 32 kbyte unified L1 cache and a 64 entry fully associative TLB. Cache lines contain 32 bytes. Integer registers are 32 bits wide. Physical addresses are also 32 bits. It supports virtual addresses of 46 bits. 1 Gbyte of main memory is installed.

a) Give one advantage of a direct mapped cache.
b) What is the main disadvantage of a direct mapped cache?
c) How many sets does the cache contain?
d) How many comparators does the cache require?
e) How many bits do these comparators work on?
f) Your program is a text processor for large documents: in an initial check, it scans the document looking for illegal characters. For an 8 Mbyte document, what would you expect the L1 cache hit rate to be during the initial check? (You are expected to do a calculation and give an approximate numeric answer!)
g) Your program manipulates large arrays of data. In order to consistent good performance, you should avoid one thing. What is it? (Be precise – a numeric answer relevant to the processor described above and an explanation is required here.)
h) What is the alternative to a unified cache? What advantages does it provide?

In addition to data and tags, a cache will have additional bits associated with each entry. List these bits and add a short phrase describing the purpose of each bit (or set of bits). (In all cases, make your answers concise: simply list any differences from a preceding answer.)
(i) A set-associative write-back cache
(ii) A set-associative write-through cache
(iii) A direct mapped cache
(iv) A fully associative cache

Unit 3:-

1. Draw a diagram showing how the instruction fetch and execution units of a superscalar processor are connected. Show the widths of the datapath (in words - not bits; your diagram should be relevant to a 32-bit or 64-bit processor). Which factor primarily determines performance: the instruction issue width (number of instructions issued per cycle) or the number of functional units?
2. List the capabilities of the instruction fetch/despatch unit needed to make an effective superscalar processor.
3. Why does a VLIW machine need a good optimizing compiler?
4. Where can you find a small dataflow machine in every high performance processor?

5. How the processor performance can be improved other than the enhancement in VLSI technology?
6. Discuss symmetric multiprocessor with proper diagram.
7. What are the advantages of vector processor?
8. What if vector data is not stored in a strided fashion in memory? (irregular memory access to a vector)
9. What is meant by an array processor and how is it different to other types of parallel processors?
10. Why does a VLIW machine restrict the op-codes which may be placed in any 'slot' of its instructions?
11. What piece of software is crucial in order to achieve good performance from a VLIW machine?
12. How many instructions can a 4-way superscalar complete in one cycle?
13. How many instructions would you *expect* it to complete?
14. Why are your answers to the previous two questions different?
15. Give an example of an instruction sequence in which the performance of a processor might benefit from register renaming.
16. Why would a processor execute both branches of a conditional branch?
17. Under what circumstances will more instructions enter a processor's pipeline than are ever completed ('graduated')?
18. A 4-bit branch mask accompanies every instruction in the MIPS R10000 machine's pipeline. What is its purpose?
19. What determines whether an instruction can be issued by an instruction issue unit in a superscalar machine?

Unit 4:-

1. Some problems can be efficiently run in parallel with minimal effort ('trivially parallelisable' problems). What are the key characteristics of such problems?
2. Why does any system allowing multiple threads of execution require a 'test-and-set' instruction?
3. What is the key characteristic of this instruction?
4. Assuming that a problem has some inherent parallelism (and is not trivially parallelisable), there are some general requirements for it to run efficiently on a parallel machine. What is the most important of these?
5. Give an example of a situation where caches in a shared memory machine need to be kept coherent.
6. List the states that a cache line may be in for the most commonly implemented cache coherence protocol. Provide a short description of each state.
7. What is the advantage of having both **Exclusive** and **Shared** states for cache lines in a shared memory machine?
8. What is the name of the programming model most commonly used for distributed memory machines?
9. What are the key operations needed by this programming model?
10. What is the channel bisection width for the star network.
11. Draw Omega network with 64 inputs that uses 4x4 switches.
12. What is destination-tag routing, where is it used and how does it work?

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Formal Language & Automata Theory **Subject Code-CS404**
Year: 2nd Year **Semester: Fourth**

Module Number	Topics	Number of Lectures
1	**Introduction:**	**5L**
	1. Basic definition of sequential circuit, block diagram, mathematical representation, concept of transition table and transition diagram (Relating of Automata concept to sequential circuit concept).	2
	2. Design of sequence detector.	2
	3. Introduction to finite state model.	1
2	**Finite State Machine:**	**5L**
	1. Definitions, capability & state equivalent, kth-equivalent concept, Merger graph, Merger table, Compatibility graph	2
	2. Minimization of FSM, Equivalence between two FSM's	1
	3. Finite memory definiteness, testing table & testing graph., Limitations of FSM	2
3.	**Finite Automata:**	**10L**
	1. Definition, Application of finite automata	1
	Finite Automata with output	
	1. Concept and design of Moore &Melay machine, Mealy to Moore conversion and vice versa, minimization.	3
	Finite Automata without output	
	1. Deterministic finite automaton and non-deterministic finite automaton.Transition diagrams and Language recognizers.	2
	2. NFA with Î transitions - Significance, acceptance of languages.	1
	3. Conversions and Equivalence: Equivalence between NFA with and without Î transitions. NFA to DFA conversion.	3
4	**Regular Language and Grammar:**	**6L**
	1. Regular sets.Regular expressions, identity rules. Arden's theorem state and prove	1
	2. Constructing finite Automata for a given regular expressions, Regular string accepted by NFA/DFA	1
	3. Pumping lemma of regular sets. Closure properties of regular sets (proofs not required).	1
	4. Grammar Formalism: Regular grammars-right linear and left linear grammars.	1
	5. Equivalence between regular linear grammar and FA	1
	6. Inter conversion	1
5	**Context Free Language and Grammar:**	**7L**
	1. Definition of Context Free Grammars, Derivation trees, sentential forms. Right most and leftmost derivation of strings.	1
	2. Ambiguity in context free grammars.	1
	3. Minimization of Context Free Grammars. Chomsky normal form and Greibach normal form.	3
	4. Pumping Lemma for Context Free Languages.	1

	5. Enumeration of properties of CFL (proofs omitted). Closure property of CFL, Ogden's lemma & its applications.	1
6	**Push Down Automata:**	**5L**
	1. Push down automata of definition.Acceptance of CFL, Acceptance by final state and acceptance by empty state and its equivalence.	2
	2. Equivalence of CFL and PDA, inter-conversion. (Proofs not required).	2
	3. Introduction to DCFL and DPDA.	1
7	**Turing Machine:**	**4L**
	1. Turing Machine of definition, model, design of TM, Computable functions	1
	2. Church's hypothesis, counter machine	1
	3. Types of Turing machines (proofs not required)	1
	4. Universal Turing Machine, Halting problem	1
Total Number Of Hours = 42		

Faculty In-Charge HOD, CSE Dept.

Assignment:

Module-1(Introduction):

1. A long sequenceofpulsesentersa twooutput,twooutput synchronous sequential circuit,which isrequiredto produceanoutputpulse z=1 whenever the sequence1111 occurs.Overlapping sequencesareaccepted;forexample,iftheinput is01011111……..,therequiredoutputis00000011…….

(a)Drawastatediagram.

(b)Selectastateassignmentandshowtheexcitationandoutputtable.

(c)Writedown theexcitationfunctionsforTflip-flopsand drawthe correspondinglogic diagram.

2. Designa2-input,2-outputsynchronoussequentialcircuit,whichisrequiredtoproduce anoutputpulsez=1,wheneverasequence000101occurs.Overlappingsequencesare notaccepted. Drawthe state diagram, transitiontable andexcitationfunctiononlyusing SRflip-flop.

Module-2 (Finite sate machine):

1. MinimizethefollowingmachineM3bydeterminethesetof equivalentstates.

PS	NS,O/	
	I/P=0	I/P=1
A	E,1	C,
B	C,	A,
C	B,0	G,
D	G,	A,
E	F,1	B,0
F	E,1	D,0

G	D,0	G,
H	F,1	B,0

2. For theincompletely specified machine shown in Table .find the merger graph, mergertable&compatibilitygraph.

PS	NS,			
	X=00	X=01	X=10	X=11
A	_	_	E,1	_
B	C,	A,	B,0	_
C	C,	D,1	_	A,
D	_	E,1	B,	_
E	B,0	_	C,	B,0

3. Convertthe machinefrommealytomooreormooreto mealyas pertable.
4. i)

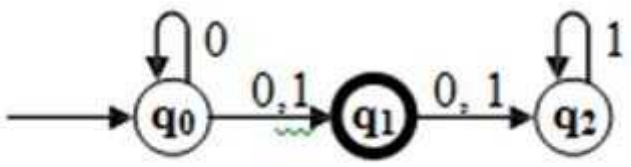
PS	NS,			
	I/P=0		I/P=1	
Q1	Q2	1	Q1	0
Q2	Q3	0	Q4	1
Q3	Q1	0	Q4	0
Q4	Q3	1	Q2	1

ii)

PS	NS		Z
	X=0	X=1	
Q0	Q1	Q2	1
Q1	Q3	Q2	0
Q2	Q2	Q1	1
Q3	Q0	Q3	1

Module-3 (Finite Automata):

1. i)
   WriteDFAtoacceptstringsof0's,1's&2'sbeginningwitha0followedbyoddnumberof1'sandendingwitha2.
   ii)Findclosureofeachstate andgivethesetof all stringsoflength3orlessacceptedbyautomaton.
2. ConvertfollowingNFAtoDFAusingsubsetconstructionmethod and minimize the DFA if possible**.**



Module-4 (Regular Language and grammar):

1. i) WriteRegular expressionforthe followingL={ $a^n b^m$: m,n areeven}L ={ a^n,b^m: m>=2, n>=2}.

ii) Obtain an NFA to accept the following language L= { w |w = $abab^n oraba^n$ where n>0}.

2. Prove that (i)$a^m b^m$, (ii) a^p(where p is n^2) is not a regular language using pumping lemma.

Module-5 (Context free language and grammar):

1. Is thefollowinggrammarambiguousS ->aB|bA; A->aS|bAA|a; B->bS |aBB |b
2. Obtainagrammartogeneratethefollowinglanguage:

   i)L={ w|$n_a(w)>n_b(w)$}

   ii)L={ $a^n b^m c^k$|n+2m=kforn>=0,m>=0}

Module-6 (Push down automata):

1. DefinePDA.ObtainPDAto acceptthelanguageL={$a^n b^n$|n>=1}byafinalstate.
2. i) ConvertPDAtoCFG.PDAisgivenbyP=({p,q},{0,1},{X,Z},δ,q,Z)),
   Transitionfunctionδisdefinedby
   δ(q,1,Z)={(q,XZ)}
   δ(q,1,X)={(q,XX)}
   δ(q,Ĥ,X)={(q,Ĥ)}
   δ(q,0,X)={(p,X)}
   δ(p,1,X)={(p, Ĥ)}

   ii)ConverttoPDA,CFGwithproductions
   AoaAA,A->aS|bS |a
   S ->SS |(S)|H
   S ->aAS |bAB|aB,
   A->bBB| aS|a,
   B->bA|a

Module-7 (Turing machine):

1. DesignaTuringmachineto acceptaPalindrome.
2. Post'sCorrespondenceproblemDesignaTMtorecognizeastringof0sand1ssuchthatthenumberof0sisnottwiceasthatof1s

Note:

For module 1 and 2 follow the book :

1. ZVI Kohavi, "Switching & Finite Automata", 2nd edition, Tata McGraw Hill.

For module 3 and 7 follow the books:

1. Mishra and Chandrashekaran,"Theory of Computer Science , Automata Languages and computation", 2nd edition, PHI.
2. Peter Linz,"Ïntroduction to Formal Language and Automata", 5th edition, Jones and Bartlett's Publications.
3. C.K.Nagpal, "Formal Languages and Automata Theory", Oxford.
4. Hopcroft H.E. and Ullman J. D,"Introduction to Automata Theory Language and Computation", Pearson Education.
5. John C Martin, "Introduction to languages and the Theory of Computation", TMH

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Technical Report Writing & Language Lab
Course Code: HU481
L-T-P scheme: 0-0-3 **Course Credit: 1**

Objectives:
1. To inculcate a sense of confidence in the students.
2. To help them become good communicators both socially and professionally.
3. To assist them to enhance their power of Technical Communication.

Learning Outcomes:

Course Contents:
Exercises that must be done in this course are listed below:
Exercise No.1: Report Types (Organizational/Commercial/Business/Project)

Exercise No. 2: Report Format &Organization of Writing Materials

Exercise No. 3: Report Writing (Practice Sessions &Workshops)

Exercise No. 4: Introductory Lecture to help the students get a clearidea of Technical Communication& the need of Language Laboratory Practice Sessions

Exercise No. 5: Conversation Practice Sessions: (To be done as real life interactions)
a) Training the students by using Language Lab Device/Recommended Texts/cassettes/cd to get their Listening Skill & Speaking killhoned
b) Introducing Role Play & honing overall Communicative Competence

Exercise No. 6: Group Discussion Sessions:
a) Teaching Strategies of Group Discussion
b) Introducing Different Models & Topics of Group Discussion
c) Exploring Live/Recorded GD Sessions for mending students' attitude/approach & for taking remedial measure Interview Sessions;
d) Training students to face Job Interviews confidently and successfully
e) Arranging Mock Interviews and Practice Sessions for integrating Listening Skill with Speaking skill in a formal situation for effective communication

Exercise No. 7: Presentation:
a) Teaching Presentation as a skill
b) Strategies and Standard Practices of Individual/Group Presentation
c) Media & Means of Presentation: OHP/POWERPOINT/Other Audio-Visual Aids

Exercise No. 8: Competitive Examination:
a) Making the studentsawareofProvincial/National/InternationalCompetitiveExaminations
b) Strategies/Tactics for success in Competitive Examinations
c) SWOT Analyse is and its Application in fixing Target

Text Book:
1. Nira Konar: English Language Laboratory: A Comprehensive Manual

D. Sudharani: Advanced Manual for Communication Laboratories& Technical Report Writing Pearson Education (W.B.edition), 2011PHI Learning,2011

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Numerical Methods Lab
Course Code: M(CS)491
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:

1. To give an overview of *what* can be done
2. To give insight into *how* it can be done
3. To give the confidence to tackle numerical solutions

An understanding of how a method works aids in choosing a method. It can also provide an indication of what can and will go wrong, and of the accuracy which may be obtained.To gain insight into the underlying physics.
"The aim of this course is to introduce numerical techniques that can be used on computers, rather than to provide a detailed treatment of accuracy or stability"

Learning Outcomes:
On completion of this course, the student will be able to:

1. Demonstrate skills in using computer programming tools for engineering calculations;
2. Demonstrate ability to construct simple computer algorithms using a programming tool;
3. Apply simple numerical methods to solve mathematical problems with relevance to civil engineering;
4. Appreciate the limitations and the applicability of the numerical methods;
5. Apply computer-based numerical methods for the solution of engineering problems.

Course Contents:
Exercises that must be done in this course are listed below:

1. Assignments on Newton forward /backward, Lagrange' s interpolation.
2. Assignments on numerical integration using Trapezoidal rule, Simpson' s 1/3 rule.
3. Assignments on numerical solution of a system of linear equations using Gauss elimination and Gauss-Seidel iterations.
4. Assignments on numerical solution of Algebraic Equation by Regular-falsi and Newton Raphson methods.
5. Assignments on ordinary differential equation: Euler' s and Runga-Kutta methods.

Text Book:

1. Introductory method of numerical analysis, Sastry S.S
2. Computer Programming in fortran 77, Rajaraman V
3. Numerical methods: for scientific and engineering computation,Mahinder Kumar Jain

.
Recommended Systems/Software Requirements:

1. Intel based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
2. Turbo C or TC3 complier in Windows XP or Linux Operating System.

Experiment No: 1(a) Newton forward interpolation
Aim: Write a C program to implement the Newton forward interpolation.
Description:
Interpolation is the process of finding the values of y corresponding to the any value of x between x0 and xn for the given values of y=f(x) for a set of values of x. Out of the many techniques of interpolation, Newton's Forward and Backward Interpolation are two very widely used formulas. In this tutorial, we're going to discuss a C program for Newton Forward Interpolation along with its sample output.
Both of Newton's formulas are based on finite difference calculus. These formulas are very often used in engineering and related science fields. Before going through the source code for Newton Forward Interpolation, let's go through the forward interpolation formula and the variables used in the C program.

Newton's forward interpolation formula contains y0 and the forward differences of y0. This formula is used for interpolating the values of y near the beginning of a set of tabulated values and extrapolation the values of y a little backward (i.e. to the left) of y0. The formula is given below:

$$P(x) = y_0 + q\,\Delta y_0 + \frac{q(q-1)}{2!}\Delta^2 y_{-1} + \frac{(q+1)\,q\,(q-1)}{3!}\Delta^3 y_{-1} +$$
$$+ \frac{(q-1)\,q\,(q-1)\,(q-2)}{4!}\Delta^4 y_{-2} + \frac{(q+2)\,(q+1)\,q\,(q-1)\,(q-2)}{5!}\Delta^5 y_{-2} + \ldots$$
$$\ldots + \frac{(q+n-1)\ldots(q-n+1)}{(2n-1)!}\Delta^{2n-1} y_{-(n-1)} + \frac{(q+n-1)\ldots(q-n)}{(2n)!}\Delta^{2n} y_{-n},$$

Compared to forward interpolation, the backward interpolation formula contains yn and the backward differences of yn. This formula is used for interpolating the values of y near the end of a set of tabulated values and also for extrapolating the values of y a little ahead (i.e. to the right) of yn.

Algorithm:

1. Function NFI ()
2. Read n, x
3. For I = 1 to n by 1 do
4. Read x[i], y[i]
5. End for
6. If ((x < x[i] or (x > x[n]))
7. Print "Value lies out of boundary"
8. Exit
9. End if
10. //Calculating p
11. p = (x – x [1]) / (x [2]-x [1])
12. // Forward diff table
13. For j = 1 to (n-1) by 1 do
14. For i =1 to (n - j) by 1 do
15. If (j=1) Then
16. d[i][j] = y [i+1] – y[i]
17. Else
18. d[i][j] = d[i+1][j-1] – d[i][j-1]
19. End if
20. End For
21. End For
22. // Applying Formula
23. Sum =y [1]
24. For I = 1 to (n-1) by 1 do
25. Prod = 1
26. For j =0 to (i-1) by 1 do
27. Prod = prod * (p-j)
28. End for
29. m = fact(i)
30. Sum = sum + (d[1][i] * prod) / m
31. End For
32. Print "Ans is", Sum
33. End Function

/* Program to implement Newton's forward interpolation*/

```
1       #include<stdio.h>
2       #include<conio.h>
3       #include<math.h>
4       #include<stdlib.h>
5       main()
6       {
7         float x[20],y[20],f,s,h,d,p;
8         int i,n;
```

```
10      scanf("%d",&n);
11      printf("enter the elements of x:");
12      for(i=1;i<=n;i++)
13       {
14           scanf("%f",&x[i]);
15            }
16                   printf("enter the elements of y:");
17                for(i=1;i<=n;i++)
18                {
19               scanf("%f",&y[i]);
20                              }
21      h=x[2]-x[1];
22      printf("Enter the value of f:");
23      scanf("%f",&f);
24     s=(f-x[1])/h;
25     p=1;
26     d=y[1];
27     for(i=1;i<=(n-1);i++)
28      {
29                    for(j=1;j<=(n-i);j++)
30                     {
31                          y[j]=y[j+1]-y[j];
32
33                     }
34                     p=p*(s-i+1)/i;
35                     d=d+p*y[1];
36      }
37     printf("For the value of x=%6.5f THe value is %6.5f",f,d);
38      getch();
39     }
```

OUTPUT:

```
how many record you will be enter: 5
enter the value of x0: 2.5
enter the value of f(x0): 9.75
enter the value of x1: 3
enter the value of f(x1): 12.45
enter the value of x2: 3.5
enter the value of f(x2): 15.70
enter the value of x3: 4
enter the value of f(x3): 19.52
enter the value of x4: 4.5
enter the value of f(x4): 23.75
Enter X for finding f(x): 4.25
u = -0.500
f(4.25) = 21.583750
```

Experiment No: 1(b) Newton backward interpolation

Aim: Write a C program to implement Newton backward interpolation.

Algorithm:

1. Function NBI ()
2. Read n, x
3. For I = 1 to n by 1 do
4. Read x[i], y[i]
5. End for

7. Print “Value lies out of boundary”
8. Exit
9. End if
10. //Calculating p
11. p = (x – x [1]) / (x [2]-x [1])
12. // Forward diff table
13. For j = 1 to (n-1) by 1 do
14. For i =1 to (n - j) by 1 do
15. If (j=1) Then
16. d[i][j] = y [i+1] – y[i]
17. Else
18. d[i][j] = d[i+1][j-1] – d[i][j-1]
19. End if
20. End For
21. End For
22. // Applying Formula
23. Sum =y [n]
24. For I = 1 to (n-1) by 1 do
25. Prod = 1
26. For j =0 to (i-1) by 1 do
27. Prod = prod * (p+j)
28. End for
29. m = fact(i)
30. Sum = sum + (d[n-1][i] * prod) / m
31. End For
32. Print “Ans is”, Sum
33. End Function

/* Program to implement Newton’s forward interpolation */

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
main()
{
float x[20],y[20],f,s,d,h,p;
intj,i,k,n;
printf(“enter the value of the elements :”);
scanf(“%d”,&n);
printf(“enter the value of x: \n\n”);
for(i=1;i<=n;i++)
    {
scanf(“%f”,&x[i]);
                }
printf(“enter the value of y: \n\n”);
for(i=1;i<=n;i++)
    {
scanf(“%f”,&y[i]);
                }
                h=x[2]-x[1];
printf(“enter the searching point f:”);
scanf(“%f”,&f);
s=(f-x[n])/h;
d=y[n];
p=1;
for(i=n,k=1;i>=1,k<n;i–,k++)
{
```

```
            {
y[j]=y[j]-y[j-1];
                        }
                        p=p*(s+k-1)/k;
                        d=d+p*y[n];
}
printf("for f=%f ,ans is=%f",f,d);
getch();
}
```

OUT PUT:

```
how many record you will be enter: 5
enter the value of x0: 2.5
enter the value of f(x0): 9.75
enter the value of x1: 3
enter the value of f(x1): 12.45
enter the value of x2: 3.5
enter the value of f(x2): 15.70
enter the value of x3: 4
enter the value of f(x3): 19.52
enter the value of x4: 4.5
enter the value of f(x4): 23.75
Enter X for finding f(x): 4.25

_____________________________________________________

x(i)    y(i)    y1(i)   y2(i)   y3(i)   y4(i)
_____________________________________________________

 2.500   9.750

 3.000   12.450  2.700

 3.500   15.700  3.250   0.550

 4.000   19.520  3.820   0.570   0.020

 4.500   23.750  4.230   0.410   -0.160  -0.180
 u = -0.500
f(4.25) = 21.583750 -
```

Experiment No: 1(c)Lagrange' s interpolation
Aim: Write a C program to implement Lagrange' s interpolation.

Algorithm:

1. Input number of Observation n
2. For i = 1 to n
3. Input Xi
4. Input Yi
5. Next i
6. Input xp at which yp to be computed
7. Initialize yp = 0
8. For i = 1 to n
9. t = 1
10. For j = 1 to n
11. If j ≠ i

13. End If
14. Next j
15. yp = yp + t * Yi
16. Next i
17. Print yp as output
18. Stop

/* Program to implement Lagrange' s interpolation*/

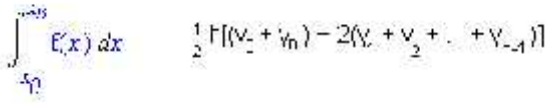
```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main()
{
  float x[10],y[10],temp=1,f[10],sum,p;
  inti,n,j,k=0,c;

  printf("\nhow many record you will be enter: ");
  scanf("%d",&n);
  for(i=0; i<n; i++)
  {
   printf("\n\nenter the value of x%d: ",i);
   scanf("%f",&x[i]);
   printf("\n\nenter the value of f(x%d): ",i);
   scanf("%f",&y[i]);
  }
  printf("\n\nEnter X for finding f(x): ");
  scanf("%f",&p);

  for(i=0;i<n;i++)
  {
    temp = 1;
    k = i;
    for(j=0;j<n;j++)
    {
      if(k==j)
      {
        continue;
      }
      else
      {
        temp = temp * ((p-x[j])/(x[k]-x[j]));
      }
    }
    f[i]=y[i]*temp;
  }

  for(i=0;i<n;i++)
  {
     sum = sum + f[i];
  }
  printf("\n\n f(%.1f) = %f ",p,sum);
  getch();
}
```

OUTPUT:

```
enter the value of n 4
enter the value to be found 2.5
```

1 1
2 8
3 27
4 64
X = 2.500000
sum = 15.625000

Experiment No:2.Trapezoidal rule
Aim: Write a C program to implement Trapezoidal rule.
Description:
A number of definite integrals need to be solved in applied mathematics, physics and engineering. The manual analytical solution of definite integrals is quite cumbersome and time consuming. So, in this post I have presented source code in C program for Trapezoidal method as one of the computer-programming-based solutions of definite integrals. The techniques of numerical methods are used to solve this equation; it involves a number of calculations and efforts have been made to minimize error in the program.
The trapezium or trapezoidal rule can be used as a way of estimating the area under a curve because the area under a curve is given by integration. So, the trapezoidal rule gives a method of estimating integrals. This is useful when you come across integrals that you don't know how to evaluate. So, the program for trapezoidal method in C given here is applicable to calculate finite integral or area under a curve.
$h=(x_n - x_0)/n$
After that, the C source code for trapezoidal method uses the following formula to calculate the value of definite integral:



Algorithm:

1. Read x1, x2, e {x1 and x2 are the two end points of the internal the allowed error in integral is e}
2. h = x2 - x1
3. SI = (f(x1) + f(x2))/2;
4. I = h - si
5. i = 1 Repeat
6. x = x1 + h/2
7. for J= 1 to I do
8. SI = SI + f(x)
9. x = x + h
10. End for
11. i = 21
12. h = h/2 {Note that the internal has been halved above and the number of points where the function has to be computed is doubled}
13. i0 = i1
14. i1 = h.si. until / I1 - i0 / <= c./i1/
15. Write I1, h, i
16. Stop.

/* Program to to implement Trapezoidal rule */

```
#include<stdio.h>
#include<math.h>
main()
{
float h, a, b, n, x[20], y[20], sum = 0, integral;
int i;
clrscr();
```

```
scanf("%f %f %f", &a, &b, &n);
printf("enter the values of x:");
for(i = 0; i <= (n-1); i++)
 {
scanf("%f", &x[i]);
 }
printf("\n enter the values of y:");
for(i = 0; i <= (n-1); i++)
 {
scanf("%f", &y[i]);
 }
 h = (b-a)/n;
x[0] = a;
for(i = 1; i <= n-1; i++)
 {
x[i] = x[i-1] + h;
sum = sum + 2 * y[i];
 }
sum = sum + y[b];
integral = sum * (h/2);
printf("approximate integral value is: %f", integral);
getch();
}
```

OUTPUT :

```
enter the values of a, b, n
123
enter the values of x:
123
enter the values of y:
123
approximate integral value is 2.166667
```

Experiment No:2(a)Simpson' s 1/3 rule

AIM: Write a C Program to implement Simpson' s 1/3 rule.

Description:

n the source code below, a function *f(x) = 1/(1+x)* has been defined. The calculation using **Simpson 1/3 rule in C** is based on the fact that the small portion between any two points is a parabola. The program follows the following steps for calculation of the integral.

- As the program gets executed, first of all it asks for the value of lower boundary value of x i.e. x_0, upper boundary value of x i.e. x_n and width of the strip, h.
- Then the program finds the value of number of strip as $n=(x_n - x_0)/h$ and checks whether it is even or odd. If the value of 'n' is odd, the program refines the value of 'h' so that the value of 'n' comes to be even.
- After that, this C program calculates value of f(x) i.e 'y' at different intermediate values of 'x' and displays values of all intermediate values of 'y'.
- After the calculation of values of 'c', the program uses the following formula to calculate the value of integral in loop.
  Integral = $*((y_0 + y_n) + 4(y_1 + y_3 + \ldots\ldots\ldots + y_{n-1}) + 2(y_2 + y_4 + \ldots\ldots\ldots + y_{n-2}))$
- Finally, it prints the values of integral which is stored as 'ans' in the program.

If f(x) represents the length, the value of integral will be area, and if f(x) is area, the output of Simpson 1/3 rule C program will be volume. Hence, numerical integration can be carried out using the program below; it is very easy to use, simple to understand, and gives reliable and accurate results.

f(x) = 1/(1+x)

.

Algorithm:

2. h = (x2 - x1)/2
3. i = 2
4. si = f(x1) + f(x2)
5. s2 = 0
6. s4 = f(x1 + h)
7. I0 = 0
8. In = (s + 4s4).(h/3)
9. Repeat
10. s2 = s2 + s4 { s2 stores already computed functional value and s4 the value computed in the new nitration }
11. s4 = 0
12. x = x1 + h/2
13. for j = 1 to I do
14. s4 = s4 + f(x)
15. x = x + h
16. h = h/2
17. i = 2i
18. io = in
19. in = (s1 + 2s2 + 4s4) . (h/3)
20. until |In-Io|≤e. /in
21. Write In, h, i
22. STOP

/* Program to to implement Simpson' s 1/3 rule. */

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
main()
{
float h, a, b, n, x[20], y[20], sum = 0, itgl;
int i;
clrscr();
printf("enter the values of a, b, n");
scanf("%f%f%f", &a, &b, &n);
printf("enter the values of x");
for(i = 0; i <= n; i++)
 {
scanf("%f", &x[i]);
 }
printf("\n enter the values of y");
for(i = 0; i <= n; i++)
 {
scanf("%f", &y[i]);
 }
 h = (b - a)/n;
 a = x[0];
 b = x[n];
for(i = 0; i <= (n-2); i++)
 {
x[i] = x[i] + h;
if(i % 2 == 0)
  {
sum = sum + 4 * y[i];
  }
else
  {
```

```
  }
 }
itgl = sum * (h/3);
printf("integral value%f", itgl);
getch();
}
```

OUTPUT :

```
enter the values of a, b, n
123
enter the value of x
4567
enter the values of y
8912
integral value is 5.555556
```

Experiment No: 3(a)Gauss elimination.

AIM: Write a C Program to implement Gauss elimination method.

Description:

let us first consider the following three equations:

a1x + b1y + c1z = d1
a2x + b2y + c2z = d2
a3x + b3y + c3z = d3

Assuming a1 ≠ 0, x is eliminated from the second equation by subtracting (a2/ a1) times the first equation from the second equation. In the same way, the C code presented here eliminates x from third equation by subtracting (a3/a1) times the first equation from the third equation.

Then we get the new equations as:

a1x + b1y + c1z = d1
b'2y + c'2z = d'2
c''3z = d''3

The elimination procedure is continued until only one unknown remains in the last equation. After its value is determined, the procedure is stopped. Now, Gauss Elimination in C uses back substitution to get the values of x, y and z as:

z= d''3 / c''3
y=(d'2 – c'2z) / b'2
x=(d1- c1z- b1y)/ a1.

Algorithm:

1. Start
2. Declare the variables and read the order of the matrix n.
3. Take the coefficients of the linear equation as:

   Do for k=1 to n

   Do for j=1 to n+1

   Read a[k][j]

   End for j

   End for k

4. Do for k=1 to n-1
   Do for i=k+1 to n
   Do for j=k+1 to n+1
   a[i][j] = a[i][j] – a[i][k] /a[k][k] * a[k][j]
   End for j
   End for i
   End for k
5. Compute x[n] = a[n][n+1]/a[n][n]
6. Do for k=n-1 to 1
   sum = 0
   Do for j=k+1 to n
   sum = sum + a[k][j] * x[j]
   End for j
   x[k] = 1/a[k][k] * (a[k][n+1] – sum)
   End for k
7. Display the result x[k]
8. Stop

/* Program to to implement Gauss elimination method */

```
#include<stdio.h>
int main()
{
   inti,j,k,n;
   float A[20][20],c,x[10],sum=0.0;
   printf("\nEnter the order of matrix: ");
   scanf("%d",&n);
   printf("\nEnter the elements of augmented matrix row-wise:\n\n");
   for(i=1; i<=n; i++)
   {
      for(j=1; j<=(n+1); j++)
      {
         printf("A[%d][%d] : ", i,j);
         scanf("%f",&A[i][j]);
      }
   }
   for(j=1; j<=n; j++) /* loop for the generation of upper triangular matrix*/
   {
      for(i=1; i<=n; i++)
      {
         if(i>j)
         {
            c=A[i][j]/A[j][j];
            for(k=1; k<=n+1; k++)
            {
               A[i][k]=A[i][k]-c*A[j][k];
            }
         }
      }
   }
```

```
/* this loop is for backward substitution*/
for(i=n-1; i>=1; i--)
{
   sum=0;
   for(j=i+1; j<=n; j++)
   {
      sum=sum+A[i][j]*x[j];
   }
   x[i]=(A[i][n+1]-sum)/A[i][i];
}
printf("\nThe solution is: \n");
for(i=1; i<=n; i++)
{
   printf("\nx%d=%f\t",i,x[i]); /* x1, x2, x3 are the required solutions*/
}
return(0);
```

OUTPUT :

```
No of Equtions : 3
Enter Coefficients of Eqution
4 3 -2
1 1 1
3 -2 1
Enter Constant value
5 3 2
Eliminated matrix as :-
4.00 3.00 -2.00 5.00
0.00 0.25 1.50 1.75
0.00 0.00 28.00 28.00
Solution :
X3 = 1.00
X2 = 1.00
X1 = 1.00
```

Experiment No:3(b) Gauss-Seidel iterations.
AIM: Write a C Program to implement Gauss-Seidel iterations method.

Description:
The program for Gauss-Seidel method in C works by following the steps listed below:

When the program is executed, first of all it asks for the value of elements of the augmented matrix row wise.
Then, the program asks for allowed error and maximum number of iteration to which the calculations are to be done. The number of iterations required depends upon the degree of accuracy.
The program assumes initial or approximate solution as y=0 and z=0 and new value of x which is used to calculate new values of y and z using the following expressions:
x= 1/a1 (d1-b1y-c1z)
y=1/b2 (d2-a2x-c2z)
z=1/c3 (d3-a3x-b3y)

Algorithm:

1. Start
2. Declare the variables and read the order of the matrix n
3. Read the stopping criteria er

4. Read the coefficients aim as
   Do for i=1 to n
   Do for j=1 to n
   Read a[i][j]
   Repeat for j
   Repeat for i
5. Read the coefficients b[i] for i=1 to n
6. Initialize x0[i] = 0 for i=1 to n
7. Set key=0
8. For i=1 to n
   Set sum = b[i]
   For j=1 to n
   If (j not equal to i)
   Set sum = sum – a[i][j] * x0[j]
   Repeat j
   x[i] = sum/a[i][i]
   If absolute value of ((x[i] – x0[i]) / x[i]) >er, then
   Set key = 1
   Set x0[i] = x[i]
   Repeat i
9. If key = 1, then
   Goto step 6
   Otherwise print results

.

/* Program to implement Gauss-Seidel iterations method. */

```
#include<stdio.h>
#include<math.h>
#define X 2
main()
{
   float x[X][X+1],a[X], ae, max,t,s,e;
   inti,j,r,mxit;
   for(i=0;i<X;i++) a[i]=0;
   puts(" Eneter the elemrnts of augmented matrix rowwise\n");
   for(i=0;i<X;i++)
   {
   for(j=0;j<X+1;j++)
   {
   scanf("%f",&x[i][j]);
   }
   }
   printf(" Eneter the allowed error and maximum number of iteration: ");
   scanf("%f%d",&ae,&mxit);
   printf("Iteration\tx[1]\tx[2]\n");
   for(r=1;r<=mxit;r++)
```

```
        max=0;
        for(i=0;i<X;i++)
        {
           s=0;
           for(j=0;j<X;j++)
           if(j!=i) s+=x[i][j]*a[j];
           t=(x[i][X]-s)/x[i][i];
           e=fabs(a[i]-t);
           a[i]=t;
        }
        printf(" %5d\t",r);
        for(i=0;i<X;i++)
        printf(" %9.4f\t",a[i]);
        printf("\n");
        if(max<ae)
        {
           printf(" Converses in %3d iteration\n", r);
           for(i=0;i<X;i++)
           printf("a[%3d]=%7.4f\n", i+1,a[i]);
           return 0;
        }

        }
    }
```

OUTPUT :

```
Enter the number of equations: 3
Enter the co-efficients of the equations:
a[1][1]= 2
a[1][2]= 1
a[1][3]= 1
a[1][4]= 5
a[2][1]= 3
a[2][2]= 5
a[2][3]= 2
a[2][4]= 15
a[3][1]= 2
a[3][2]= 1
a[3][3]= 4
a[3][4]= 8

x[1] =2.500000
x[2] =1.500000
x[3] =0.375000

x[1] =1.562500
x[2] =1.912500
x[3] =0.740625

x[1] =1.173437
x[2] =1.999688
x[3] =0.913359

x[1] =1.043477
x[2] =2.008570
x[3] =0.976119
```

x[2] =2.004959
x[3] =0.994933

x[1] =1.000054
x[2] =2.001995
x[3] =0.999474

converges to solution
x[1]=1.000054
x[2]=2.001995
x[3]=0.999474

Experiment No: 4(a)Regular-falsi

AIM: Write a program to implement Regular-falsi method.

Description:
The C Program for regulafalsi method requires two initial guesses of opposite nature. Like the secant method, interpolation is done to find the new values for successive iterations, but in this method one interval always remains constant.

The programming effort for RegulaFalsi or False Position Method in C language is simple and easy. The convergence is of first order and it is guaranteed. In manual approach, the method of false position may be slow, but it is found superior to the bisection method.

- tr – a counter which keeps track of the no. of iterations performed
- maxmitr – maximum number of iterations to be performed
- x0, x1 – the limits within which the root lies
- x2 – the value of root at nth iteration
- x3 – the value of root at (n+1)th iteration
- allerr – allowed error
- x – value of root at nth iteration in the regula function
- f(x0), f(x1) – the values of f(x) at x0 and x1 respectively

f(x) = cos(x) – x*e^x

Algorithm:

1. Start
2. Read values of x0, x1 and e

   *Here x0 and x1 are the two initial guesses

   e is the degree of accuracy or the absolute error i.e. the stopping criteria*
3. Computer function values f(x0) and f(x1)
4. Check whether the product of f(x0) and f(x1) is negative or not.

   If it is positive take another initial guesses.

   If it is negative then goto step 5.
5. Determine:

   x = [x0*f(x1) – x1*f(x0)] / (f(x1) – f(x0))
6. Check whether the product of f(x1) and f(x) is negative or not.

   If it is negative, then assign x0 = x;

   If it is positive, assign x1 = x;

7. Check whether the value of f(x) is greater than 0.00001 or not.
   If yes, goto step 5.
   If no, goto step 8.
   Here the value 0.00001 is the desired degree of accuracy, and hence the stopping criteria.
8. Display the root as x.
9. Stop
10.

/* Program to implement Regular-falsi method */

```
#include<stdio.h>
#include<math.h>
float f(float x)
{
   returncos(x) - x*exp(x);
}
voidregula (float *x, float x0, float x1, float fx0, float fx1, int *itr)
{
   *x = x0 - ((x1 - x0) / (fx1 - fx0))*fx0;
   ++(*itr);
   printf("Iteration no. %3d X = %7.5f \n", *itr, *x);
}
void main ()
{
   intitr = 0, maxmitr;
   float x0,x1,x2,x3,allerr;
   printf("\nEnter the values of x0, x1, allowed error and maximum iterations:\n");
   scanf("%f %f %f %d", &x0, &x1, &allerr, &maxmitr);
   regula (&x2, x0, x1, f(x0), f(x1), &itr);
   do
   {
      if (f(x0)*f(x2) < 0)
         x1=x2;
      else
         x0=x2;
      regula (&x3, x0, x1, f(x0), f(x1), &itr);
      if (fabs(x3-x2) <allerr)
      {
         printf("After %d iterations, root = %6.4f\n", itr, x3);
         return 0;
      }
      x2=x3;
   }
   while (itr<maxmitr);
   printf("Solution does not converge or iterations not sufficient:\n");
   return 1;
}
```

OUTPUT :

```
Enter the value of x0: -1

Enter the value of x1: 1
```

-1.000000 1.000000 0.513434 -4.540302 1.459698 -0.330761
0.513434 1.000000 0.603320 -0.330761 1.459698 -0.013497
0.603320 1.000000 0.606954 -0.013497 1.459698 -0.000527
0.606954 1.000000 0.607096 -0.000527 1.459698 -0.000021

App.root = 0.607096

Experiment No: 4(b) Newton Raphson methods
AIM: Write a program to implement Newton Raphson methods.

Algorithm:

1. Start
2. Read x, e, n, d
   *x is the initial guess
   e is the absolute error i.e the desired degree of accuracy
   n is for operating loop
   d is for checking slope*
3. Do for i =1 to n in step of 2
4. f = f(x)
5. f1 = f'(x)
6. If ([f1] < d), then display too small slope and goto 11.
   [ ] is used as modulus sign
7. x1 = x – f/f1
8. If ([(x1 – x)/x1] < e), the display the root as x1 and goto 11.
   [ ] is used as modulus sign
9. x = x1 and end loop
10. Display method does not converge due to oscillation.
11. Stop

/* Program to implement Newton Raphson methods */

```
#include<stdio.h>
#include<math.h>
float f(float x)
{
   return x*log10(x) - 1.2;
}
floatdf (float x)
{
   return log10(x) + 0.43429;
}
```

```
{
   intitr, maxmitr;
   float h, x0, x1, allerr;
   printf("\nEnter x0, allowed error and maximum iterations\n");
   scanf("%f %f %d", &x0, &allerr, &maxmitr);
   for (itr=1; itr<=maxmitr; itr++)
   {
      h=f(x0)/df(x0);
      x1=x0-h;
      printf(" At Iteration no. %3d, x = %9.6f\n", itr, x1);
      if (fabs(h) <allerr)
      {
         printf("After %3d iterations, root = %8.6f\n", itr, x1);
         return 0;
      }
      x0=x1;
   }
   printf(" The required solution does not converge or iterations are insufficient\n");
   return 1;
}
```

OUTPUT :

```
ENTER THE TOTAL NO. OF POWER:::: 3

x^0::-3

x^1::-1

x^2::0

x^3::1

    THE POLYNOMIAL IS :::  1x^3 0x^2 -1x^1 -3x^0
    INTIAL X1---->3
 ITERATION    X1    FX1    F'X1
  1        2.192  21.000 26.000
 2        1.794  5.344  13.419
 3        1.681  0.980  8.656
 4        1.672  0.068  7.475
 5        1.672  0.000  7.384
THE ROOT OF EQUATION IS 1.671700
```

Experiment No:5(a)Euler' s methods
AIM: Write a program to simulate Euler' s method.

Description:
Solving an ordinary differential equation or initial value problem means finding a clear expression for y in terms of a finite number of elementary functions of x. Euler's method is one of the simplest method for the numerical solution of such equation or problem. This **C program for Euler's method** considers an ordinary differential equations, and the initial values of x and y are known. Mathematically, here, the curve of solution is approximated by a sequence of short lines i.e. by the tangent line in each interval. Using these information, the value of the value of 'y_n' corresponding to the value of 'x_n' is to determined by dividing the length $(x_n - x)$ into n strips.
Therefore, strip width= $(x_n - x)/n$ and $x_n = x_0 + nh$.
Again, if m be the slope of the curve at point, $y_1 = y_0 + m(x_0 , y_o)h$.
Similarly, values of all the intermediate y can be found out.
Below is a source code for **Euler's method in C** to solve the ordinary differential equation **dy/dx = x+y**. It asks for the value of of x_0 , y_0 , x_n and h. The value of slope at different points is calculated

The values of y are calculated in while loop which runs till the initial value of x is not equal to the final value. All the values of 'y' at corresponding 'x' are shown in the output screen.
dy/dx = x+y

Algorithm:

1. Start
2. Define function
3. Get the values of x0, y0, h and xn
   *Here x0 and y0 are the initial conditions
   h is the interval
   xn is the required value
4. n = (xn – x0)/h + 1
5. Start loop from i=1 to n
6. y = y0 + h*f(x0,y0)
   x = x + h
7. Print values of y0 and x0
8. Check if x <xn
   If yes, assign x0 = x and y0 = y
   If no, goto 9.
9. End loop i
10. Stop

/* Program to simulate Euler' s method */

```
#include<stdio.h>
float fun(float x,float y)
{
   float f;
   f=x+y;
   return f;
}
main()
{
   floata,b,x,y,h,t,k;
   printf("\nEnter x0,y0,h,xn: ");
   scanf("%f%f%f%f",&a,&b,&h,&t);
   x=a;
   y=b;
   printf("\n  x\t  y\n");
   while(x<=t)
   {
      k=h*fun(x,y);
      y=y+k;
      x=x+h;
      printf("%0.3f\t%0.3f\n",x,y);
   }
}
```

OUTPUT :

Enter the value of range: 1 1.5

Enter the h: 0.1
y1 = 5.000

x = 1.000 => y2 = 5.500

x = 1.100 => y3 = 6.105

x = 1.200 => y4 = 6.838

x = 1.300 => y5 = 7.726

x = 1.400 => y6 = 8.808

x = 1.500 => y7 = 10.129

Experiment No:5(b)Runga-Kutta methods
AIM: Write a program to simulate Runga-Kutta methods.

/* Program to simulate Runga-Kutta methods*/

```
#include<stdio.h>
#include<math.h>
float f(float x,float y);
int main()
{
   float x0,y0,m1,m2,m3,m4,m,y,x,h,xn;
   printf("Enter x0,y0,xn,h:");
   scanf("%f %f %f %f",&x0,&y0,&xn,&h);
   x=x0;
   y=y0;
   printf("\n\nX\t\tY\n");
   while(x<xn)
   {
      m1=f(x0,y0);
      m2=f((x0+h/2.0),(y0+m1*h/2.0));
      m3=f((x0+h/2.0),(y0+m2*h/2.0));
      m4=f((x0+h),(y0+m3*h));
      m=((m1+2*m2+2*m3+m4)/6);
      y=y+m*h;
      x=x+h;
      printf("%f\t%f\n",x,y);
   }
}
float f(float x,float y)
{
   float m;
   m=(x-y)/(x+y);
   return m;
}
```

OUTPUT:
Enter the value of x0: 0

Enter the value of y0: 2

Enter the value of h: 0.05

Enter the value of last point: 0.1

k2 = 0.1025

y(0.0500) = 2.101

k1 = 0.1026

k2 = 0.1052

y(0.1000) = 2.205

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Title of Course: Communication Engineering & Coding Theory Lab
Course Code: CS491
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:

This course provides the basic theory of communication and principle ofvarious components in a communication system. Digital communication uses electrical signaling methods to transmit information over a physical channel separating atransmitter and receiver with the channel properties often time varying. This course presents the theory and practice of communication engineering including signal design, modulation methods, demodulation methods, wireless channel basics and the application of this to the design of modern OFDM systems.Students shouldbe able to design the components with specifications for a given fiber Analog communication system.

Learning Outcomes:

Upon successful completion of this course, the students will be able to:

1. Understand the basic concepts of advanced digital communication systems
2. Apply different modulation schemes to baseband signals
3. Analyze the BER characteristics of Baseband Modulated signals
4. The learner must be able to appreciate the need for modulation and calculate the antenna size for different carrier frequencies. From the functional representation of the modulated carrier wave, the learner must be able to identify the type of modulation, calculate the side-band frequencies, identify the modulating and carrier frequencies, decide the type of generation method to be adopted.
5. After understanding the basic concepts, the learner must be able to compare between the different demodulation methods, design an envelope detector, calculate the IF and image frequencies for the super heterodyne receivers given the carrier and modulating frequencies, calculate the oscillator frequency.
6. From the functional representation of the modulated carrier wave, the learner must be able to identify the type of modulation, calculate the side-band frequencies, identify the modulating and carrier frequencies; decide the type of generation method to be adopted.

Course Contents:
Exercises that must be done in this course are listed below:

1. To generate Amplitude Modulated signal and calculate its modulation index for varying amplitude of modulating signal.
2. To generate Amplitude Modulated signal and calculate sideband power, carrier power and efficiency.
3. To generate Amplitude Modulated DSB-SC Wave.
4. To generate Frequency Modulated wave and determine modulation index and bandwidth for different values of amplitude and frequency of modulating signal.
5. To observe the effect of pre-emphasis and de-emphasis on a given input signal.
6. To study different types of signal sampling and its reconstruction.
7. To study Pulse Position Modulation
8. To study the operation of Amplitude Shift Keying modulation and demodulation with the help of circuit connections.
9. To generate Pulse shift key (PSK) With Wave forms.
10. To study Quadri Phase Shift Keying (QPSK).

1. L. W. CouchIi, “Digital and Analog Communication Systems”, 2/e, Macmillan Publishing Blake, Electronic Communication Systems- Cengage Learning
2. S Sharma, Analog Communication Systems- Katson Books
3. J. D. Proakis and M. Salehi (2008), Digital Communication,
4. McGraw Hill, David Silage (2009), Digital Communication

Recommended Systems/Software Requirements:

1. Recommended Kits and Equipment Requirements:
2. AM, FM, TDM,Transmitter and Receiver of am kits
3. DSO, FG, Probes

EXPERIMENT-1

AIM: To generate Amplitude Modulated signal and calculate its modulation index for varying amplitude of modulating signal.

APPARATUS REQUIRED: Amplitude Modulation Kit, Function generator, DSO, Probes.

THEORY: Amplitude Modulation is defined as a process in which the amplitude of the carrier wave c(t) isvaried linearly with the instantaneous amplitude of the message signal m(t).The standard form of anamplitude modulated (AM) wave is defined by

$$S(t)=A_c[1+K_a m(t)]\cos(2\pi f_c t)$$

Where, K_a is a constant called the amplitude sensitivity of the modulator.
The modulation index of an Amplitude Modulated wave is defined as,

$$\mu = \frac{A_{c_m} - A_{c_m}}{A_{c_m} + A_{c_m}}$$

Where, A_{c_m} a A_{c_m} are the maximum and minimum amplitudes of the modulated wave.

WAVEFORMS:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

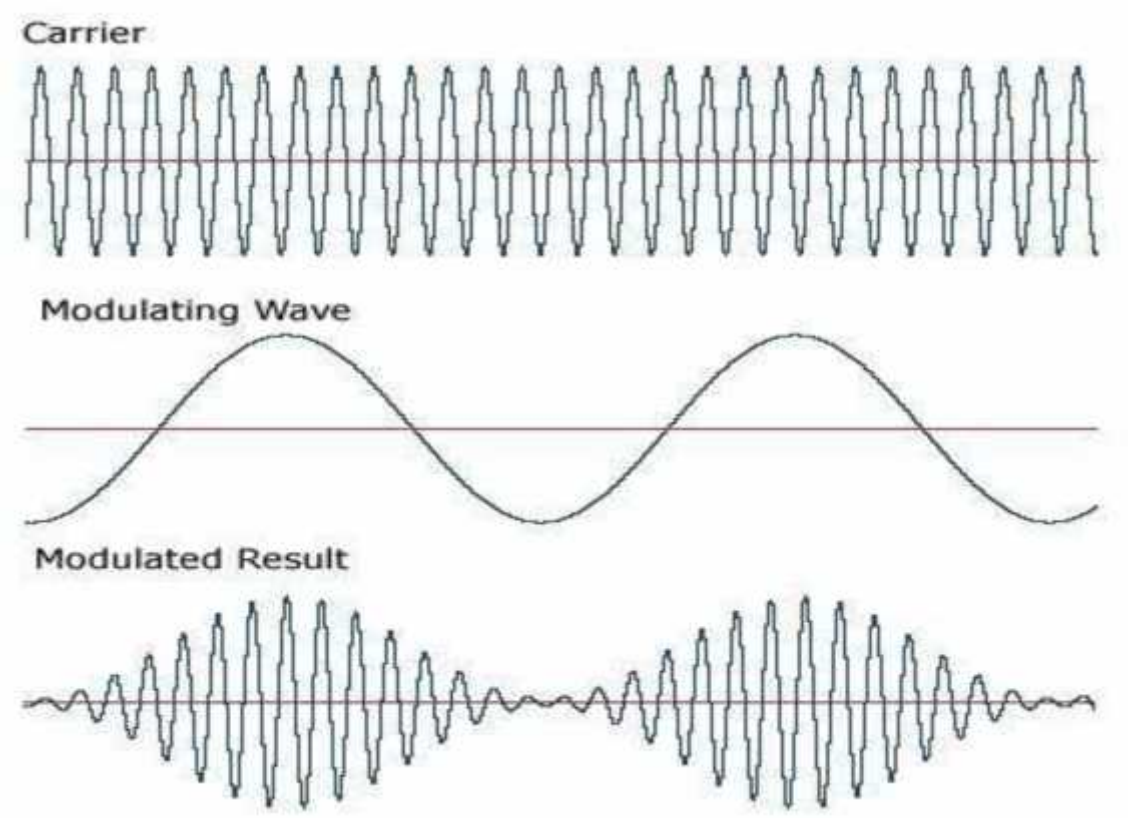


Fig. 1: Showing carrier wave, modulating wave and Amplitude modulated wave

OBSERVATION TABLE:

A_c= , f_c= , f_m=

SL. NO.	A_m	A_{c_m}	A_{c_m}	$\mu = \frac{A_{c_m} - A_{c_m}}{A_{c_m} + A_{c_m}}$

PRECAUTION:

1. Check the connections before giving the power supply.
2. Observations should be done carefully.

CONCLUSION:

EXPERIMENT-2

AIM: To generate Amplitude Modulated signal and calculate sideband power, carrier power and efficiency.

APPARATUS REQUIRED: Amplitude Modulation Kit, Function generator, DSO, Probes.

THEORY: Amplitude Modulation is defined as a process in which the amplitude of the carrier wave c(t) isvaried linearly with the instantaneous amplitude of the message signal m(t).The standard form of anamplitude modulated (AM) wave is defined by

$$S(t)=A_c[1+K_a m(t)]\cos(2\pi f_c t)$$

Where, $m(t)=A_m\cos(2\pi f_m t)$

$c(t)= A_c\cos(2\pi f_c t)$

hence,

$$S(t)=A_c\cos(2\pi f_c t) + K_a A_m A_c \cos(2\pi f_m t)\cos(2\pi f_c t)$$

$$Or,\ S(t)=A_c\cos(2\pi f_c t) + \mu A_c \cos(2\pi f_m t)\cos(2\pi f_c t)$$

Carrier Sidebands

The modulation index (μ) of an Amplitude Modulated wave is defined as,

$$\mu = \frac{A_{c_m} - A_{c_m}}{A_{c_m} + A_{c_m}}$$

The carrier power can be calculated as,

$$P_c = \frac{A_c^2}{2}$$

The sideband power can be calculated as,

$$P_s = \frac{\mu^2 A_c^2}{4}$$

Total power, $P_T = P_c + P_s$

$$= \frac{A_c^2}{2}\left(1 + \frac{\mu^2}{2}\right)$$

Mathematically, Transmission efficiency can be calculated as,

$$\eta = \frac{P_s}{P_T}$$

$$= \frac{\mu^2}{2+\mu^2}$$

WAVEFORMS:

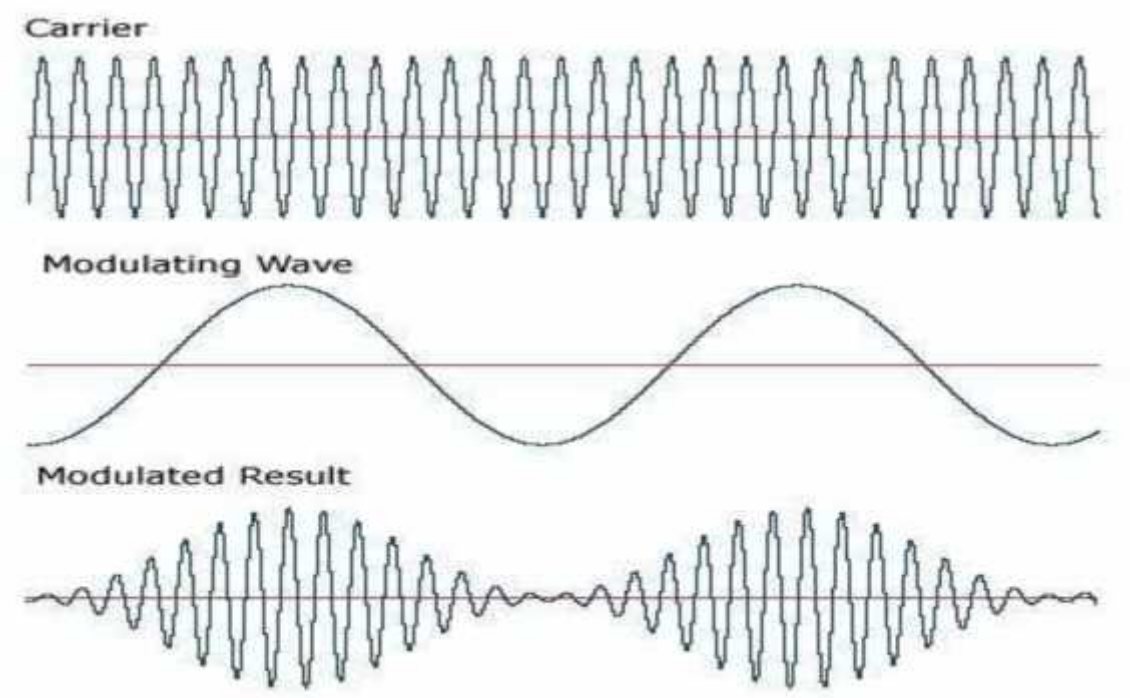


Fig. 1: Showing carrier wave, modulating wave and Amplitude modulated wave

OBSERVATION TABLE

A_m	A_C	A_{max}	A_{min}	$\mu = \frac{A_{c_m} - A_{c_m}}{A_{c_m} + A_{c_m}}$	P_s	P_c	η

PRECAUTION:

1. Check the connections before giving the power supply.
2. Observations should be done carefully.

CONCLUSION:

EXPERIMENT-3

AIM: To generate Amplitude Modulated DSB-SC Wave.

APPARATUS REQUIRED: DSB-SC transmitter kit, DSO, Probes.

THEORY: Balanced modulator is used for generating DSB-SC signal. A balanced modulator consists oftwo standard amplitude modulators arranged in a balanced configuration so as to suppress the carrierwave. The two modulators are identical except the reversal of sign of the modulating signal applied tothem.

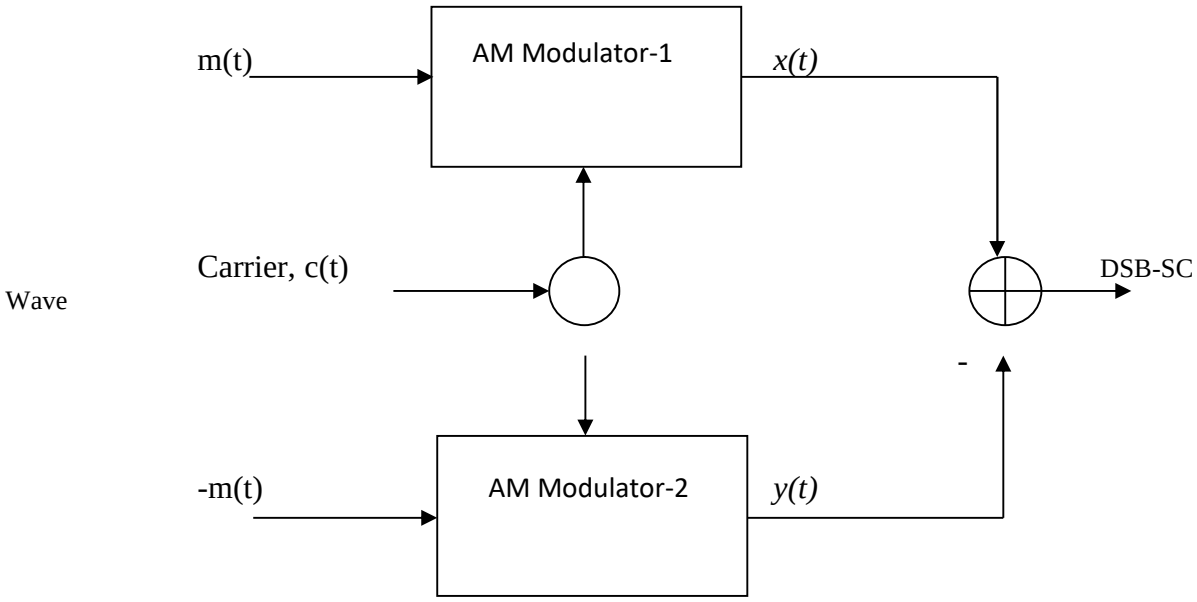


Fig.1: Block Diagram of Balanced Modulator to generate DSB-SC wave

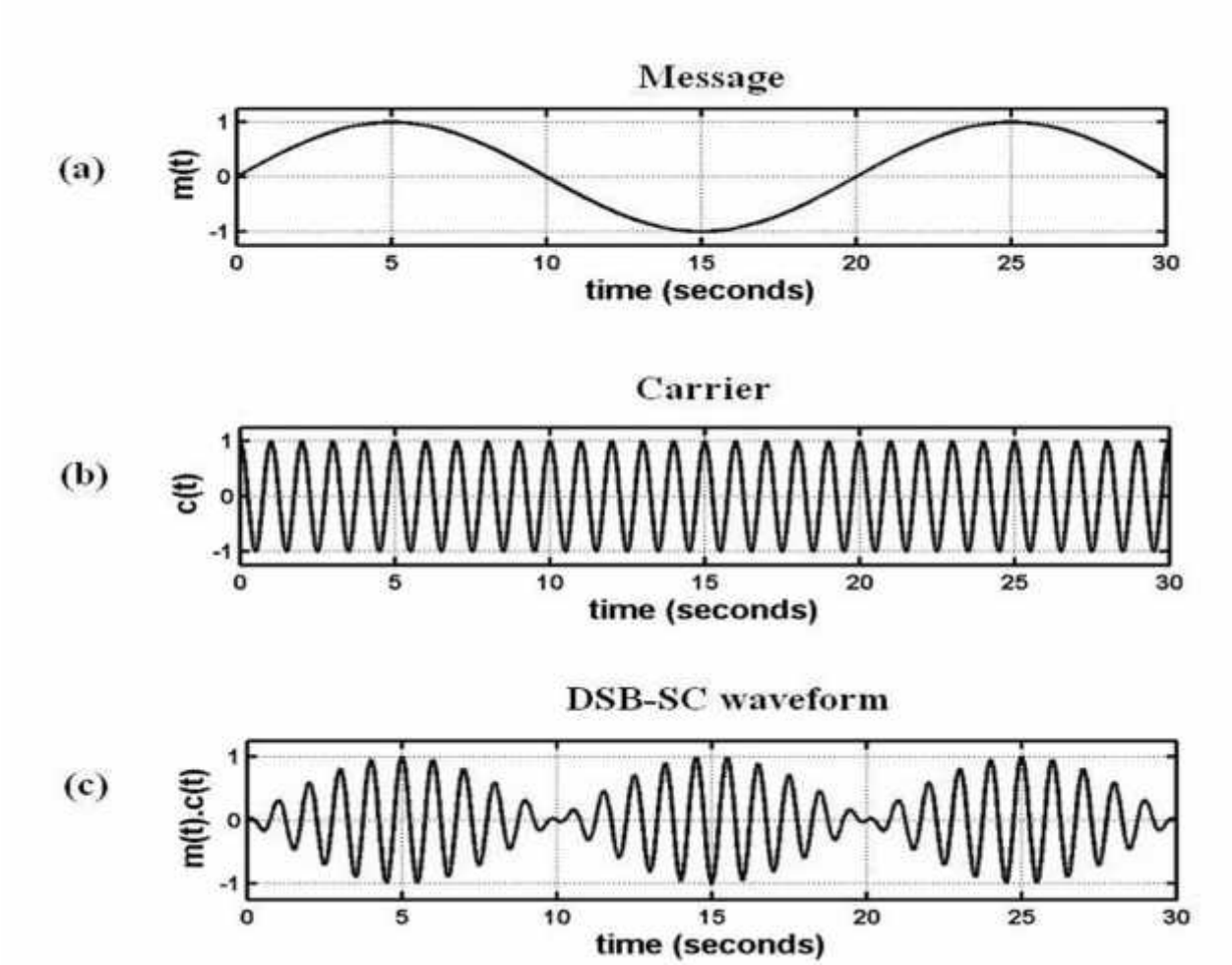
From the above block diagram we have the following expressions,

$$x(t)=A_c[1+K_a m(t)]\cos(2\pi f_c t)$$

$$y(t)=A_c[1-K_a m(t)]\cos(2\pi f_c t)$$

hence, $S(t)=2A_cK_a m(t)\cos(2\pi f_c t)$, which is clearly a DSB-SC Wave with a scaling factor of $2K_a$.

WAVEFORMS:



PRECAUTION:

1. Check the connections before giving the power supply.
2. Observations should be done carefully.

CONCLUSION:

EXPERIMENT-4

AIM: To generate Frequency Modulated wave and determine modulation index and bandwidth for different values of amplitude and frequency of modulating signal.

APPARATUS REQUIRED: IC566, 1.8KΩ, 10KΩ, 15KΩ, 0.01µF, 470pF, Function generator, DSO, Probes.

THEORY: The process, in which the frequency of the carrier is varied in accordance with the instantaneous amplitude of the modulating signal, is called "Frequency Modulation". The FM signal is expressed as

$$S(t)=A_c\cos[2\pi f_c t+\beta\sin(2\pi f_m t)]$$

Where, A_c is the carrier amplitude

f_c is the carrier frequency

and β is the modulation index of the FM wave.

Where,

$$\beta=\frac{\Delta f}{f_m}$$

where, $\Delta f=|f_c-f_{min}|$

The bandwidth of the FM wave can be calculated using the carson's formula,

$$BW=2(\beta+1)f_m$$

CIRCUIT DIAGRAM:

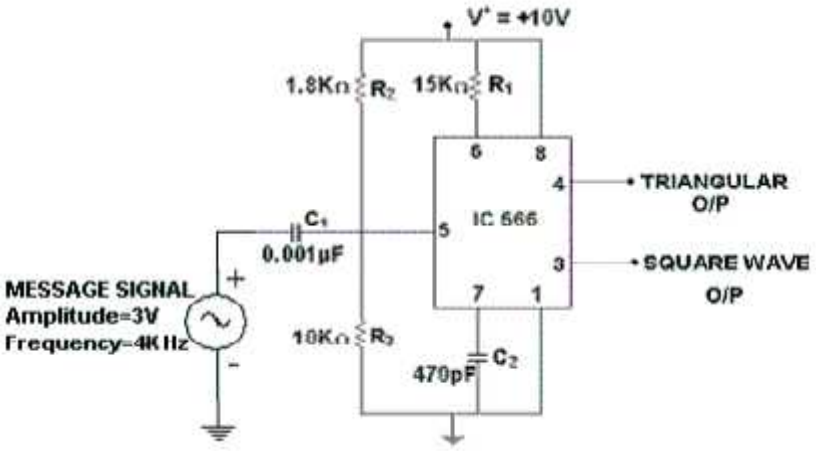


Fig. 1: FM Modulator using IC566

WAVEFORMS:

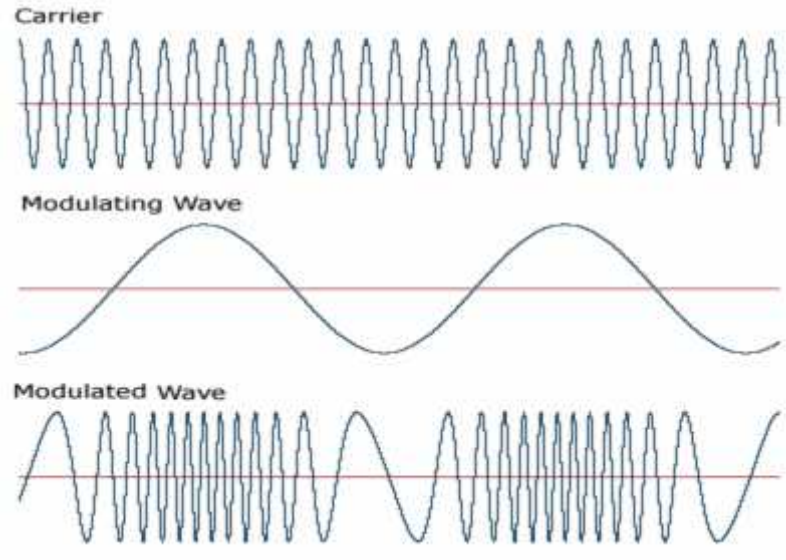


Fig. 2: Showing the frequency modulated wave

OBSERVATIONS TABLE:

Table-I

f_c=

Sl. No.	f_m	T_{max}	f_{min}	Δf	β	BW

Table-II

f_c= f_m=

Sl. No.	A_m	T_{max}	f_{min}	Δf	β	BW

PRECAUTION:

1. Check the connections before giving the power supply.
2. Observations should be done carefully.

CONCLUSION:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

EXPERIMENT-5

AIM: To observe the effect of pre-emphasis and de-emphasis on a given input signal.

APPARATUS REQUIRED: Transistor (BC107), Resistors (10KΩ,7.5KΩ,6.8KΩ), Capacitors (10nF, 0.1μF), Function generator, DSO, Regulated power supply.

THEORY: The noise has a effect on the higher modulating frequencies than on the lower ones. Thus, if the higher frequencies were artificially boosted at the transmitter and correspondingly cut at the receiver, an improvement in noise immunity could be expected, thereby increasing the SNR ratio. This boosting of the higher modulating frequencies at the transmitter is known as pre-emphasis and the compensation at the receiver is called de-emphasis.

CIRCUIT DIAGRAMS:

For Pre-Emphasis:

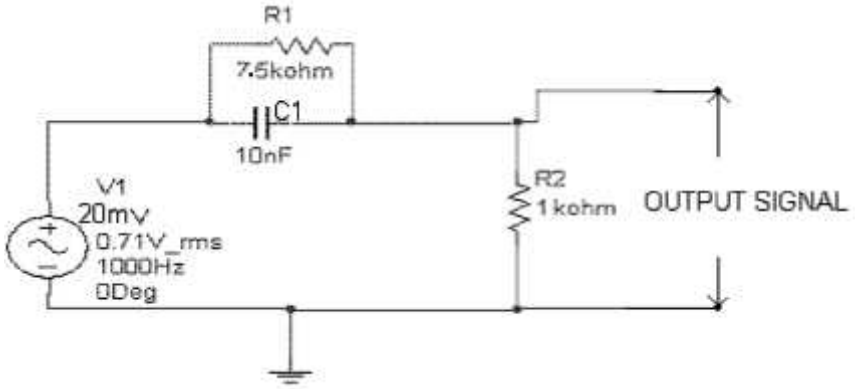


Fig.1Pre-Emphasis Circuit

For De-Emphasis:

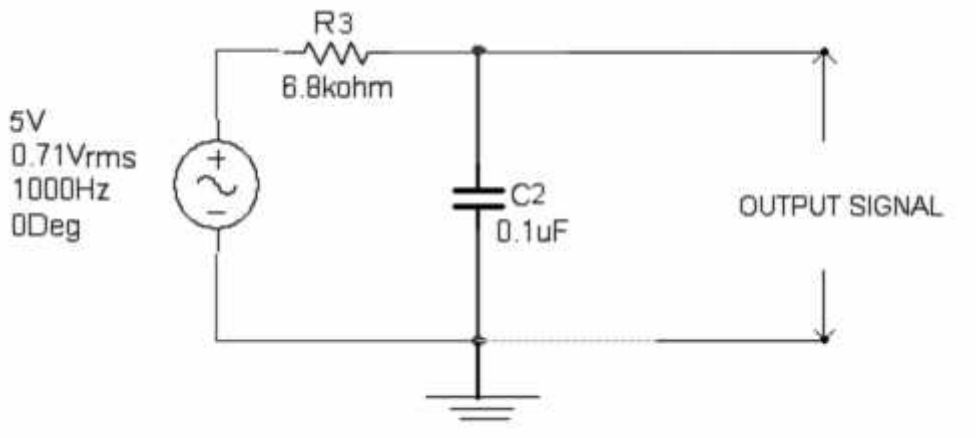


Fig.2 De-Emphasis Circuit

OBSERVATION TABLES:

TABLE-I: Pre-emphasis V_i=20mV

Frequency(KHz)	V_o(mV)	Gain in dB(20 logV_o /V_i)

TABLE-II: De-emphasis V_i=5V

Frequency(KHz)	V_o(V)	Gain in dB(20 logV_o /V_i)

GRAPHS:

Should plot in a graph paper

PRECAUTION:

1. Check the connections before giving the power supply.
2. Observations should be done carefully.

CONCLUSION:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

EXPERIMENT-6

Aim: To study different types of signal sampling and its reconstruction.

Apparatus Required:

1. Sampling and its reconstruction Kit-DCL01
2. Digital Storage Oscilloscope(DSO)
3. Power supply
4. Patch cords

Theory: Regardless of the sampling method used, by definition it captures only pieces of the message. So, how can the sampled signal be used to recover the whole message? This question can be answered by considering the mathematical model that defines the sampled signal:

Sampled message = the sampling signal × the message

As you can see, sampling is actually the multiplication of the message with the sampling signal. And, as the sampling signal is a digital signal which is actually made up of a DC voltage and many sinewaves (the fundamental and its harmonics) the equation can be rewritten as:

Sampled message = (DC + fundamental + harmonics) × message

Block Diagram:

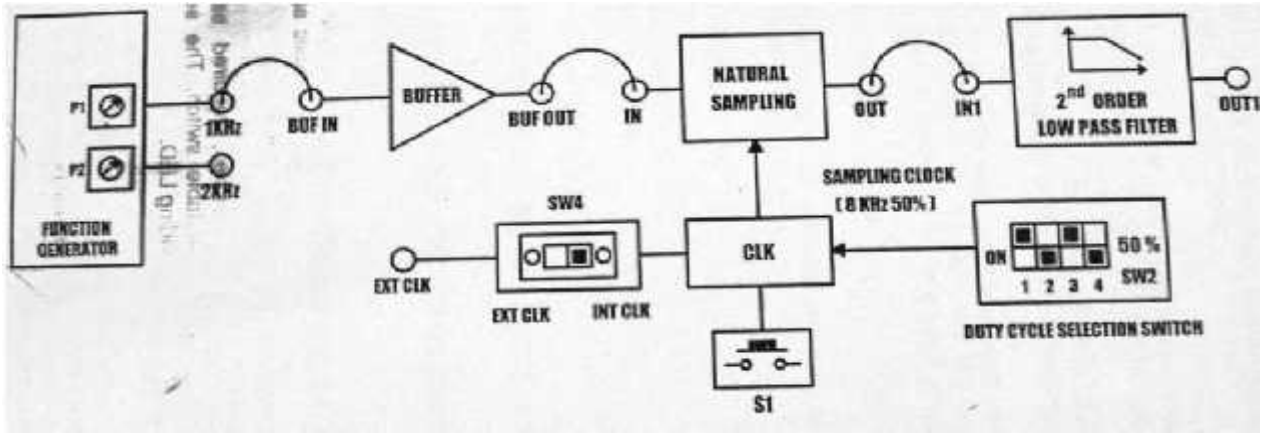


Fig. 1.1BlockDiagram for Natural Sampling

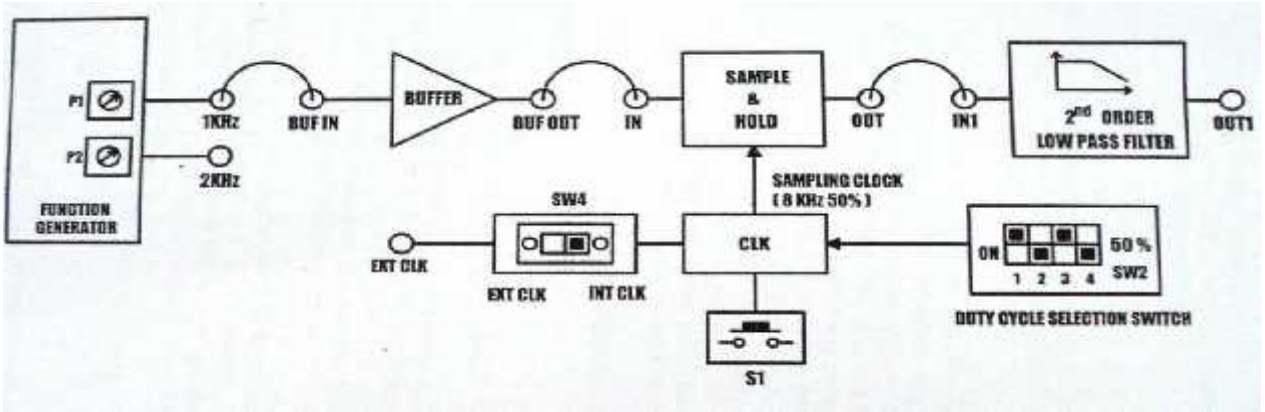


Fig. 1.2BlockDiagram for Sample and Hold

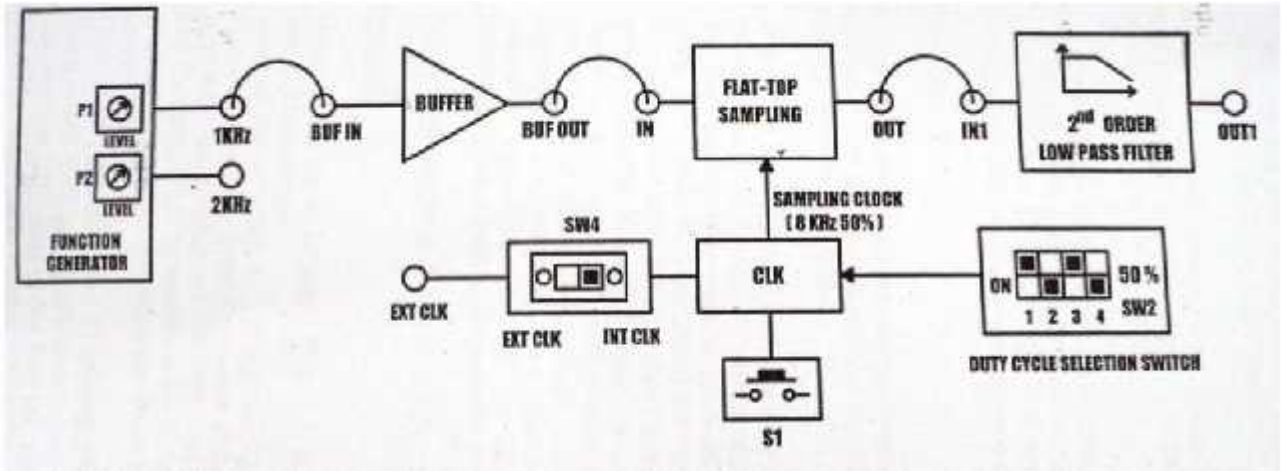
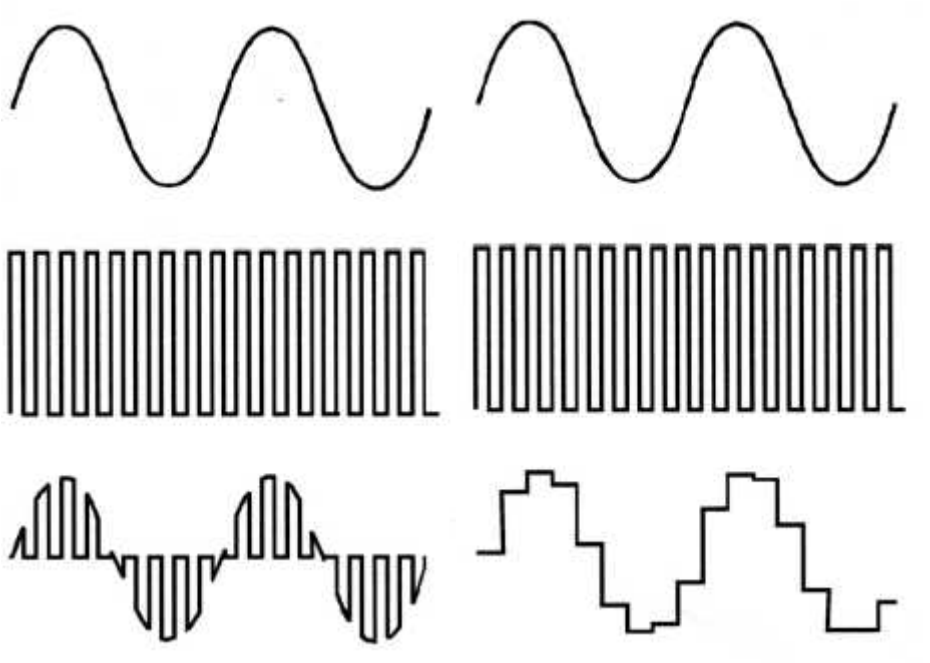


Fig. 1.3BlockDiagram for Flat Top Sampling

Waveforms



a. Sampling **b. Reconstruction**

Fig. 1. Showing sampling and reconstruction waves

Conclusion:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

EXPERIMENT-7

Aim: To study Pulse Position Modulation.

Apparatus Required: Dual trace CRO, PPM kit, connecting leads.

THEORY:

In Pulse Position Modulation, both the pulse amplitude and pulse duration are held constant but the position of the pulse is varied in proportional to the sampled values of the message signal. Pulse time modulation is a class of signalling techniques that encodes the sample values of analog signal onto the time axis of a digital signal and it is analogous to angle modulation techniques. The two main types of PTM are PWM and PPM. In PPM the analog sample value determines the position of a narrow pulse relative to the clocking time. In PPM rise time of pulse decides the channel bandwidth. It has low noise interference.

CIRCUIT DIAGRAM and WAVEFORM:

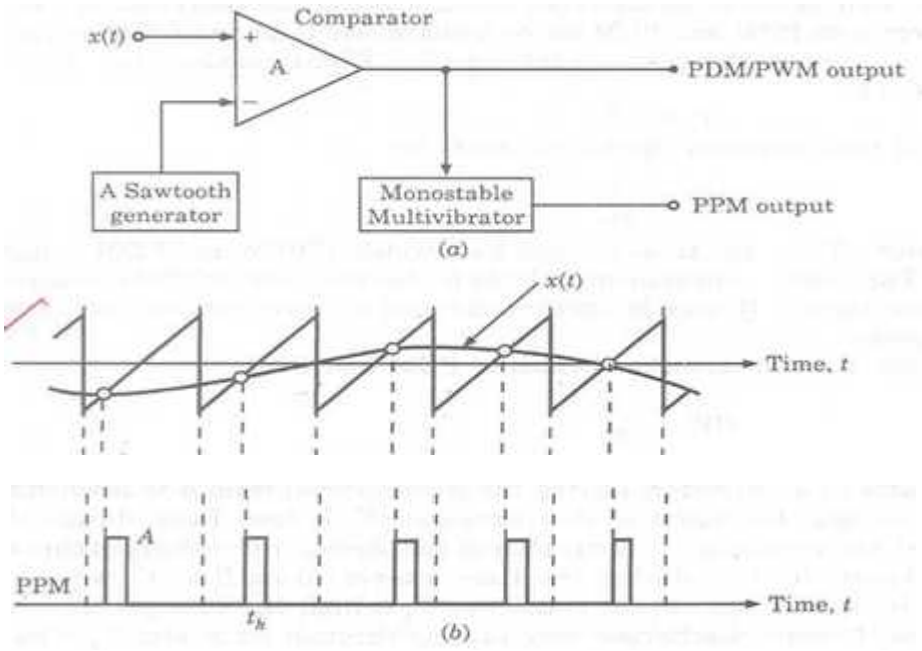


Fig. 1. PPM generation circuit and PPM waveform

Conclusion:

EXPERIMENT-8

Aim: To study the operation of Amplitude Shift Keying modulation and demodulation with the help of circuit connections.

Apparatusrequired:

1. Resistors	1.2KΩ	3
2. Transistor	BC 107	2
3. DSO		1
4. Functiongenerator	0-1MHz	1
5. Regulated PowerSupply	0-30V,1A	1
6. Probes	---	1

Theory:

The binary ASK system was one of the earliest form of digital modulation used in wireless telegraphy. In an binary ASK system binary symbol 1 is represented by transmitting a sinusoidal carrier wave of fixed amplitude A_C and fixed frequency f_C for the bit duration T_b whereas binary symbol 0 is represented by switching of the carrier for T_b seconds. This signal can be generated simply by turning the carrier of a sinusoidal oscillator ON and OFF for the prescribed periods indicated by the modulating pulse train. For this reason, the scheme is also known as on-off shift testing. Let the sinusoidal carrier can be represented by Ec (t) $=A_C \cos(2\Pi f_C t)$ then the binary ASK signal can be represented by a wave s(t) given by $S(t) = A_C \cos(2\Pi f_C t)$, symbol 1 ASK signal can be generated by applying the incoming binary data and the sinusoidal carrier to the two inputs of a product modulator. The resulting output is the ASK wave. The ASK signal which is basically product of the binary sequence and carrier signal has a same as that of base band signal but shifted in the frequency domain by $\pm f_C$. The band width of ASK signal is infinite but practically it is $3/T_b$.

Circuit diagram:

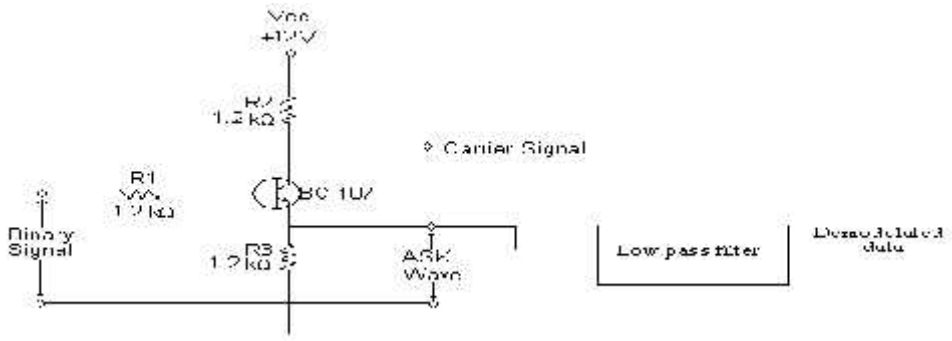


Fig. 1. Amplitude Shift Keying and demodulation Circuit

Waveforms:

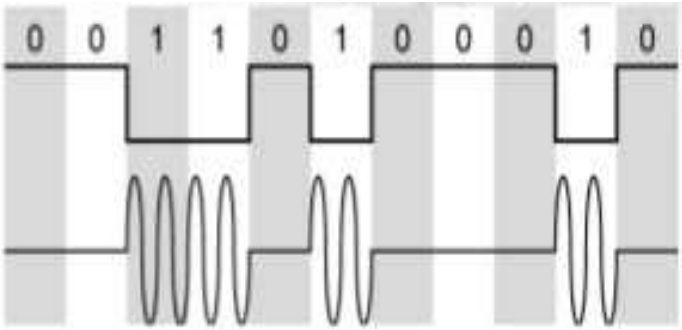


Fig. 2. Showing Amplitude Shift Keying signal

Conclusion:

EXPERIMENT No. :9

Aim: To study Phase Shift Keying (PSK).

Apparatus Used: Trainer kit, Connecting wires, DSO.

Theory: Phase shift keying (PSK) involves the phase shift change of the carrier sine wave between 0º and 180º in accordance with the data stream to be transmitted. PSK is also known as phase reversal keying (PSK).
Functionally, the PSK modulator is very similar to the ASK modulator . both uses balanced modulator to multiply the carrier with the modulating signal. Bit in contrast to ASK technique, the digital signal applied to the modulation input for PSK generation is bipolar i.e. have equal positive and negative voltage levels. When the modulating input is positive the output of modulator is a sine wave in phase with the carrier input. Whereas for the negative voltage levels, the output of modulator is a sine wave which is shifted out of phase 180º from the carrier input.

Block Diagram:

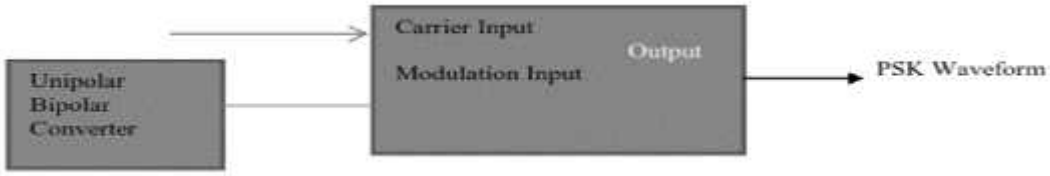


Fig. 1: Block diagram of PSK modulator

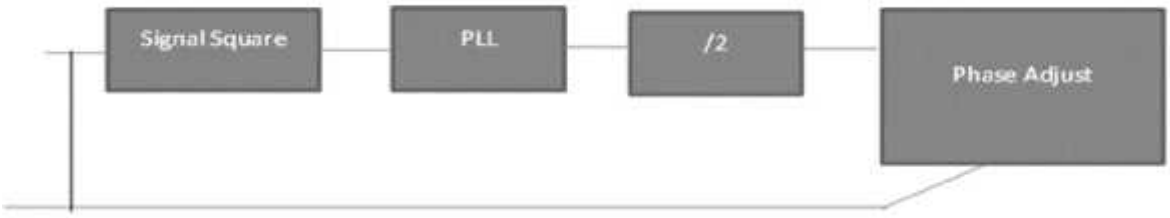


Fig. 2. Block diagram of PSK demodulator

Waveforms:

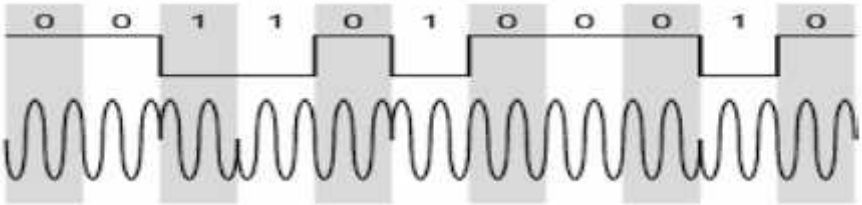


Fig. 3. PSK waveform

Conclusion:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

EXPERIMENT No. :10

Aim: To study Quadri Phase Shift Keying (QPSK).

Apparatus Used: Trainer kit, Connecting wires, DSO.

Theory: With quadrature phase shift keying modulation (also called quaternary PSK,Quadri phase PSK or 4-PSK), a sinusoidal waveform is varied in phase while keeping the amplitude and frequency constant. The term quadrature indicates that there are four possible phases. Equation (1) shows the general expression for a QPSK waveform.

$$S_i(t) = A \quad [\omega_c t + \varphi_i(t)] \qquad (1)$$

Block Diagram:

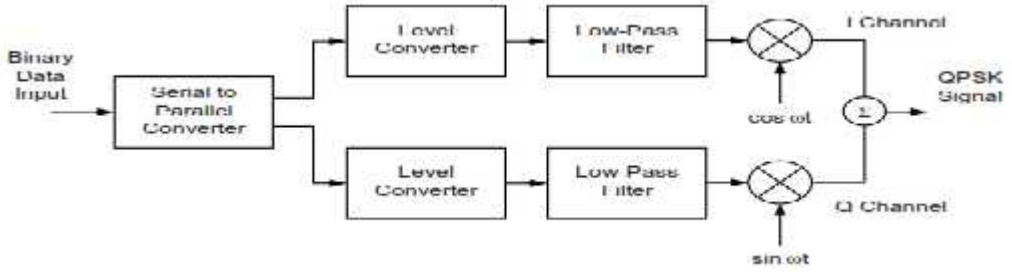


Fig. 1. Block diagram of QPSK Modulator.

Waveforms and Constellation diagram:

(a) (b)

Fig. 2. Constellation diagram (a) $\varphi = \frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}$ (b) $\varphi = 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}$

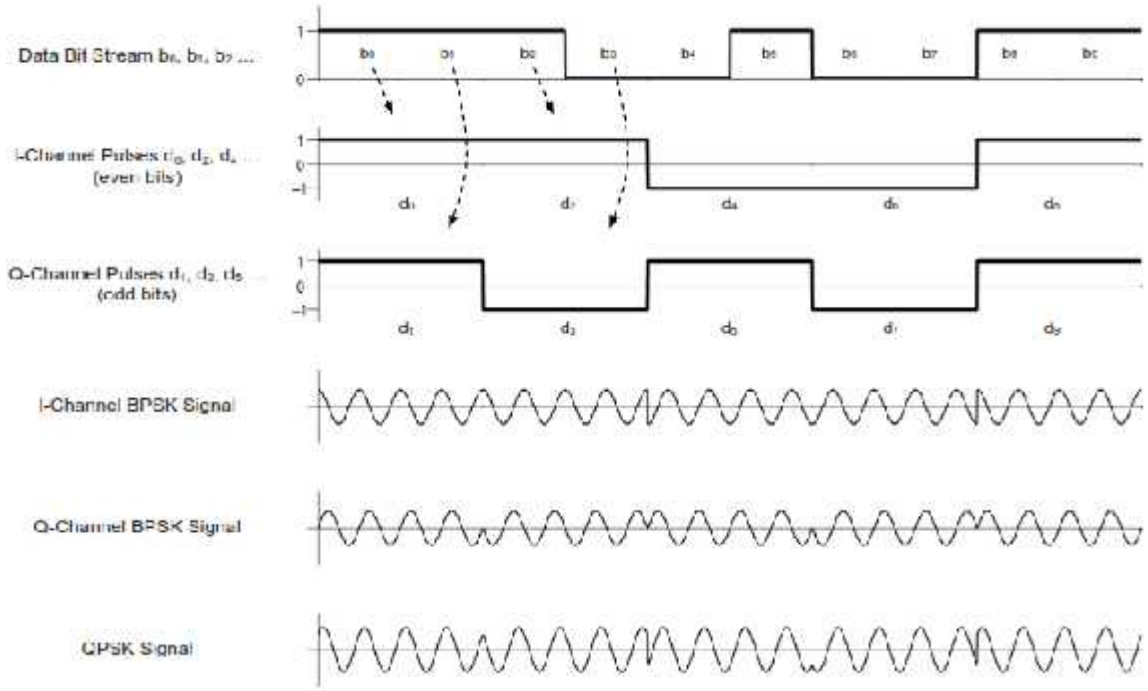


Fig. 3. Waveform of QPSK signal

Conclusion:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Title of Course: Software Engineering Lab
Course Code: CS492
L-T-P scheme: 0-0-3 **Course Credit: 2**

Prerequisite: Students must have already registered for the course, "Software Engineering".

Objectives: Students will be capable to acquire the generic software development skill through various stages of software life cycle. He will also be able to ensure the quality of software through software development with various protocol based environment.

Learning Outcomes: After completion of course student will be able to generate test cases for software testing. Students will also be able to handle software development models through rational method.

Course Contents:
Unit I- Introduction to software engineering: Code comprehension.
Unit II- Requirement engineering: Requirement Elicitation, specification, IEEE standard template for SRS, Requirement Engineering tools.
Unit III- UML Modeling: Use case diagram , State diagram, Activity Diagram, Class Diagram, Sequence diagram, Collaboration diagram, Deployment Diagram, Component Diagram, Event trace diagram , c++ code generation, Introduction to Sec UML.
Unit IV- Software Metrics: Product, process and project metrics.
Unit V- Software Testing: Structural testing, functional Testing, Testing Strategies and Tactics, Test Case design.

List of Experiments

1. Identifying the Requirements from Problem Statements Requirements, Characteristics of Requirements, Categorization of Requirements, Functional Requirements, Identifying Functional Requirements
2. E-R Modeling from the Problem Statements, Entity Relationship Model, Entity Set and Relationship Set, Attributes of Entity, Keys, Weak Entity, Entity Generalization and Specialization, Mapping Cardinalities, ER Diagram, Graphical Notations for ER Diagram Importance of ER modeling
3. Identifying Domain Classes from the Problem Statements, Domain Class, Traditional Techniques for Identification of Classes, Grammatical Approach Using Nouns, Advantages, Disadvantages, Using Generalization, Using Subclasses, Steps to Identify Domain Classes from Problem Statement, Advanced Concepts
4. Modeling UML Use Case Diagrams and Capturing Use Case Scenarios, Use case diagrams, Actor, Use Case, Subject, Graphical Representation, Association between Actors and Use Cases, Use Case Relationships, Include
5. Modeling UML Class Diagrams and Sequence diagrams, Structural and Behavioral aspects, Class diagram, Elements in class diagram, Class Relationships, Sequence diagram, Elements in sequence diagram, Object, Life-line bar, Messages.
6. Modeling Data Flow Diagrams, Data Flow Diagram, Graphical notations for Data Flow Diagram, Explanation of Symbols used in DFD, Context diagram and leveling DFD
7. Statechart and Activity Modeling Statechart Diagrams, Building Blocks of a Statechart Diagram State, Transition, Action, Guidelines for drawing Statechart Diagrams, Activity Diagrams, Components of an Activity Diagram, Activity, Flow Decision, Merge, Fork, Join, Note, Partition, A Simple Example, Guidelines for drawing an Activity Diagram
8. Estimation of Project Metrics Project Estimation Techniques, COCOMO, Basic COCOMO Model, Intermediate COCOMO Model, Complete COCOMO Model, Advantages of COCOMO, Drawbacks of COCOMO, Halstead's Complexity Metrics.
9. Estimation of Test Coverage Metrics and Structural Complexity, Control Flow Graph, Terminologies, McCabe's Cyclomatic Complexity, Computing Cyclomatic Complexity,

References

1. R.S. Pressman, “Software Engineering: A Practitioner’s Approach”, 7Edition, McGraw Hill, 2010
2. 2. Fundamentals of Software Engineering: Mall, Rajib, Prentice Hall of India, New Delhi (2nd Edition).
3. 2. Sommerville, “Introduction to Software Engineering”, 8Edition, Addison-Wesley, 2007
4. Ghezzi, Jazayeri and Mandrioli, “Fundamentals of Software Engineering”, 2Edition, Prentice-Hall, 2003
5. Peters and Pedrycz, “Software Engineering: An Engineering Approach, John Wiley, 2004
6. Len Bass, “Software Architecture in Practice”, 2Edn. Addison Wesley, 2003
7. Allamaraju, “Professional Java Server Programming”, Apress, 2004

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Title of Course: Computer Architecture Lab
Course Code: CS493

L-T-P Scheme: 0-0-3 **Course Credits: 2**

Objective: 1. 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)

2.In particular defined data flow, behavioral and 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.

3.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.

Recommended Systems/ Software Requrements:

1. Intel based Desktop PC/Laptop with minimum of 166 MHZ or faster processor with at least 1 GB RAM and 2 GB free disk space.

2. Xilinx 9.1 i

Experiment – 1

Aim:-

To implement **And**&**Xor** Gate in Dataflow & Behavioral 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-

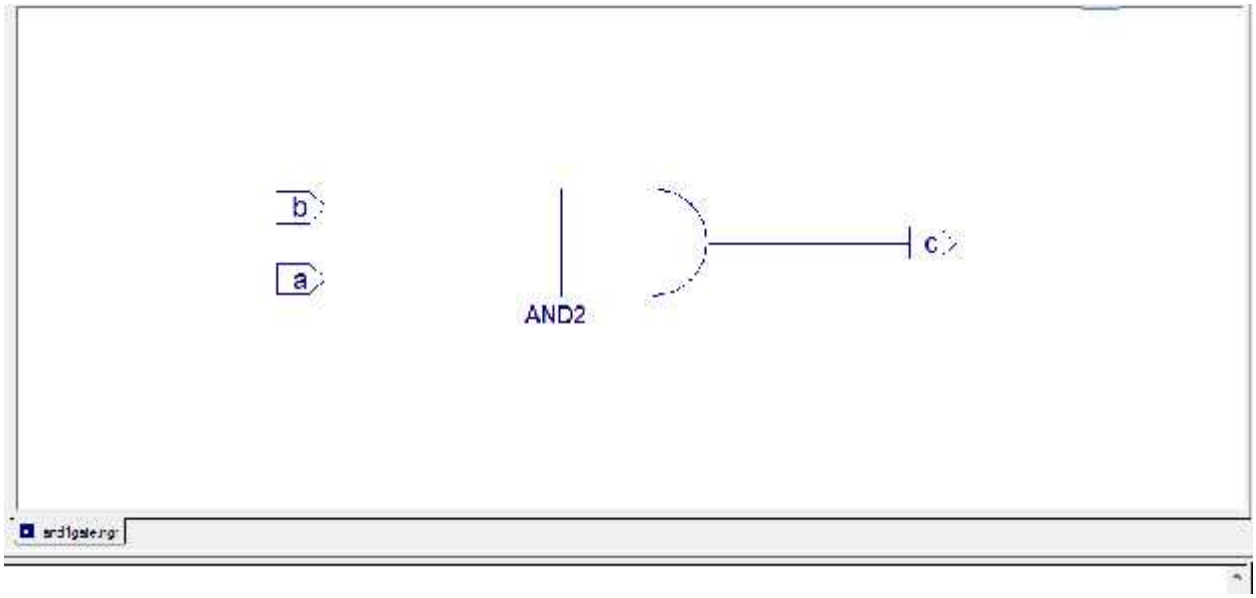
```
begin
p: process (a, b)
                begin
                if (a='1' and b='1')then
                 c<='1';
                else
                 c<='0';
                end if;
                end process;
end Behavioral;
```

XOR GATE-

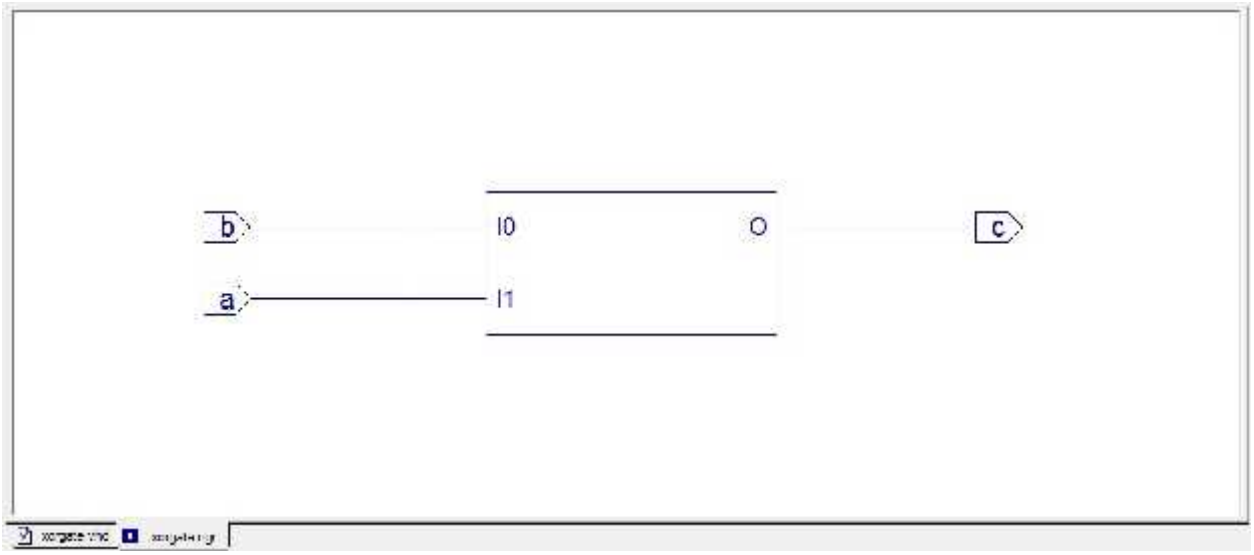
```
begin
p:process(a,b)
        begin
        If (a=b) then
                c<='0';
               else
                c<='1';
               end if;
               end process;
end Behavioral;
```

RTL Schematic:-

For AND-Gate:-

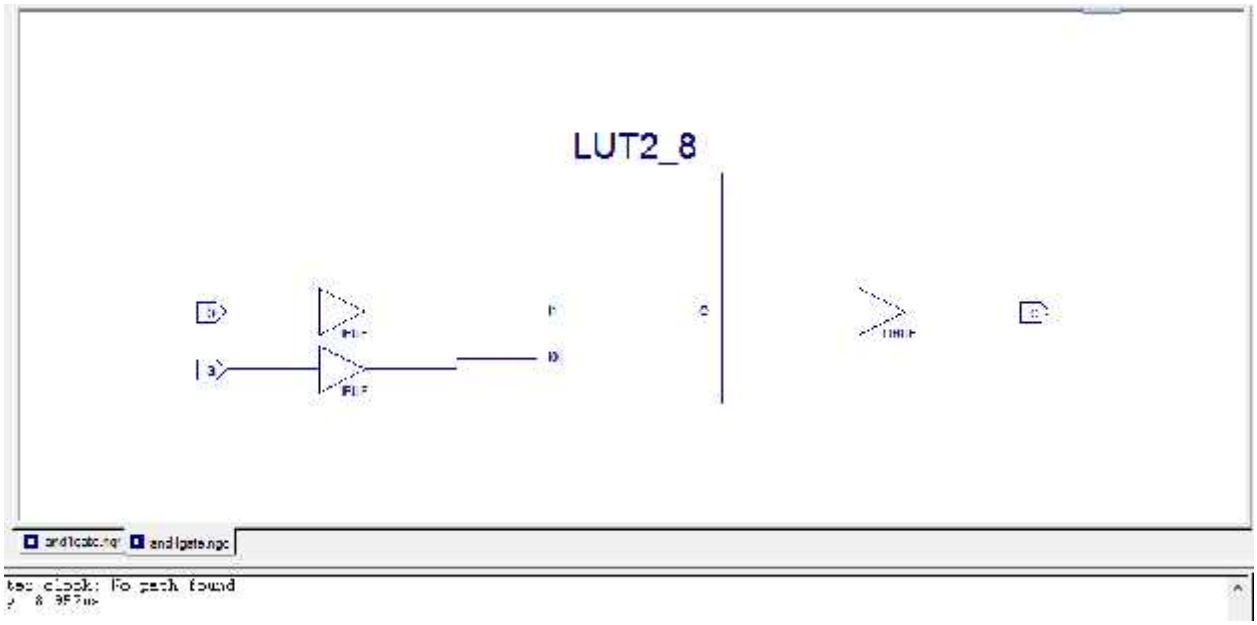


For XOR-Gate:-

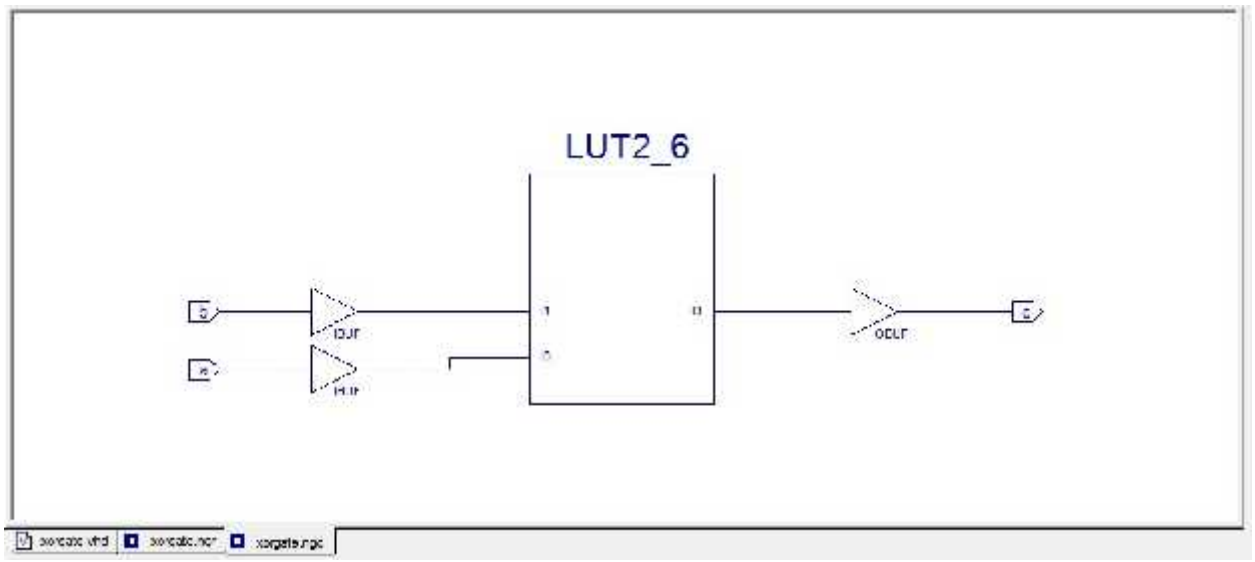

b
a
I0
I1
O
c

Technology Schematic:-

For AND-Gate:-

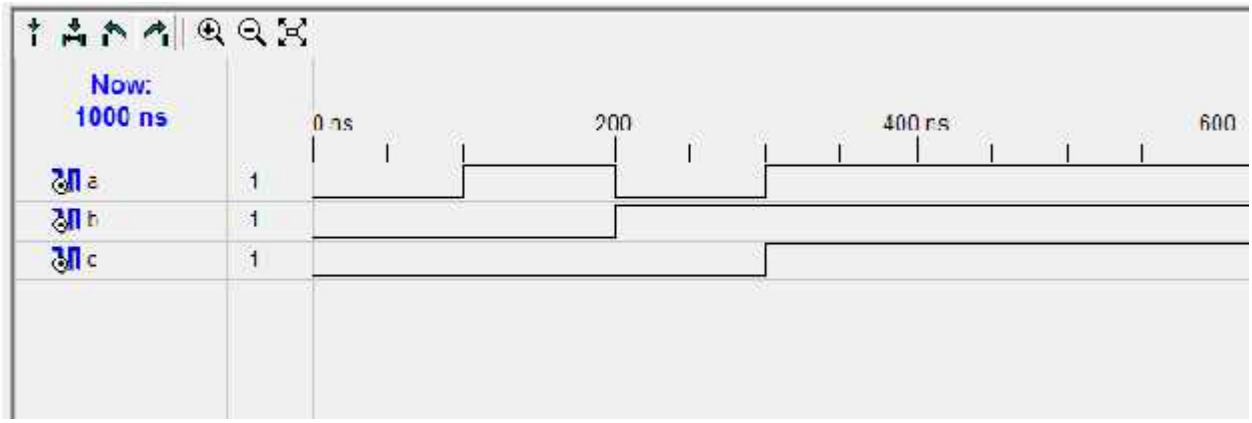


For XOR-Gate:-

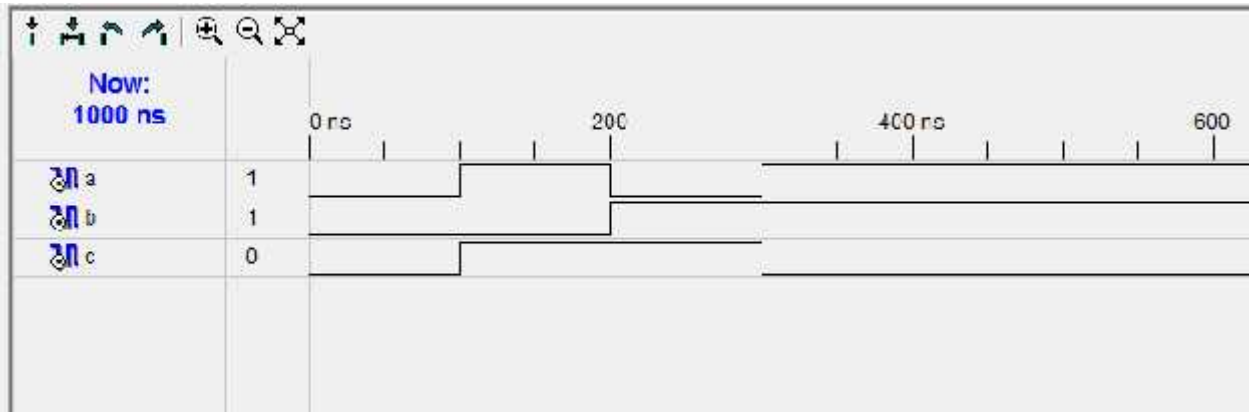


Test Bench Waveform:-

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 Dataflow & Behavioral 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:-

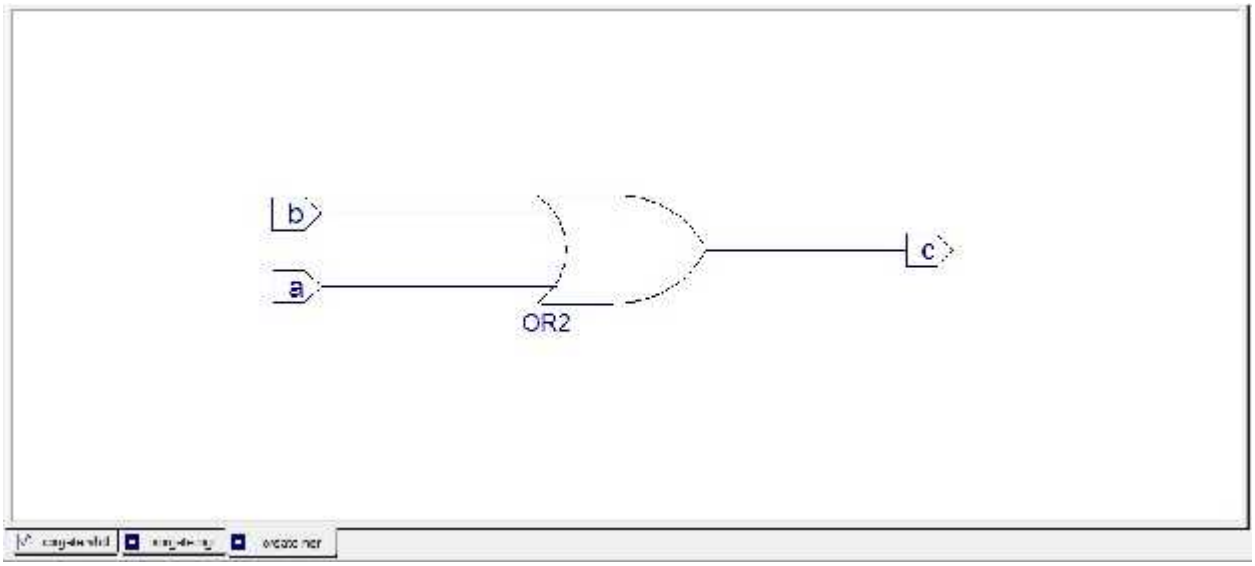
Data Flow Model:-
OR GATE-

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

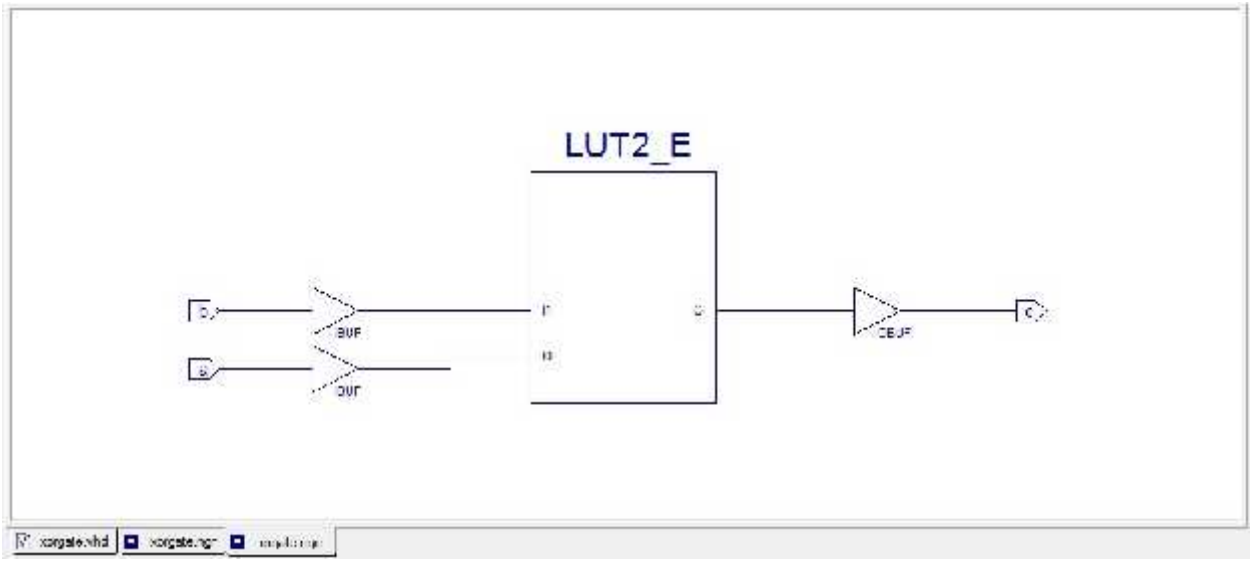
Behavioral Module:-
OR GATE-

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

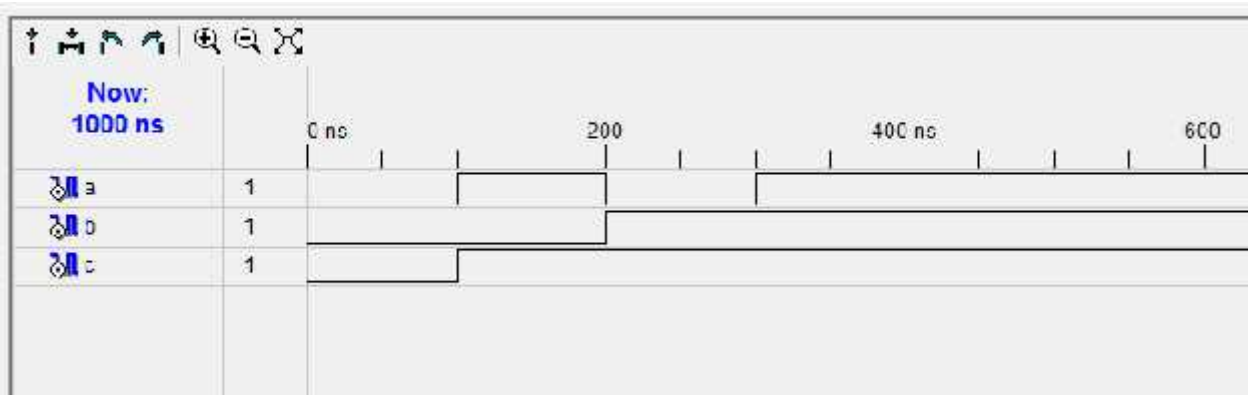
RTL Schematic:-



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 Dataflow & Behavioral 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:-

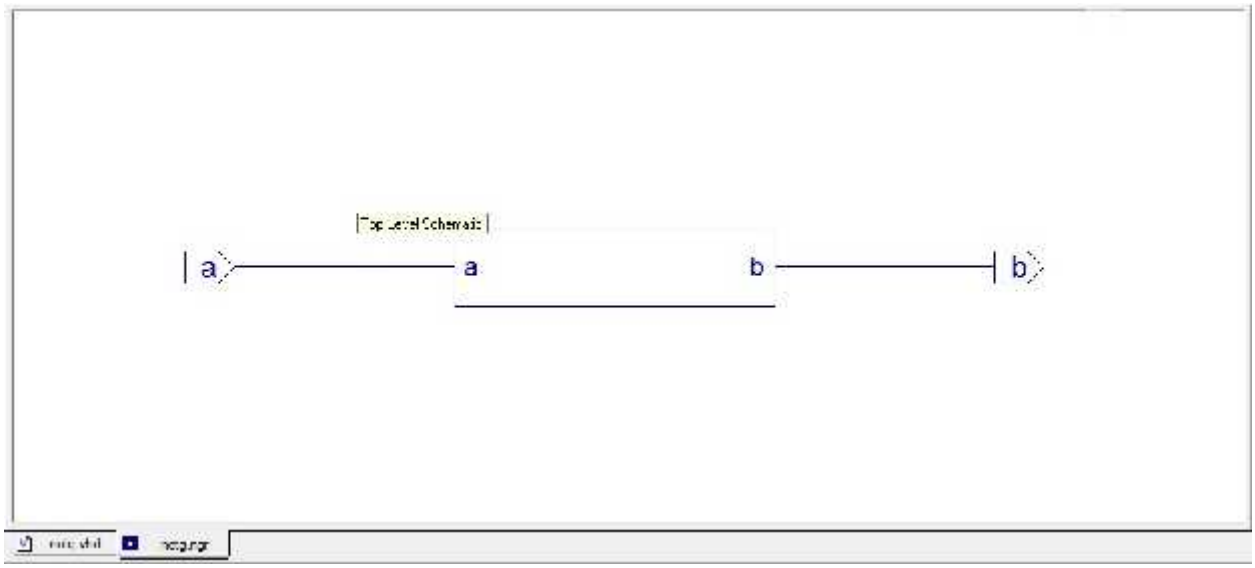
Data Flow Model:-
NOT GATE

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

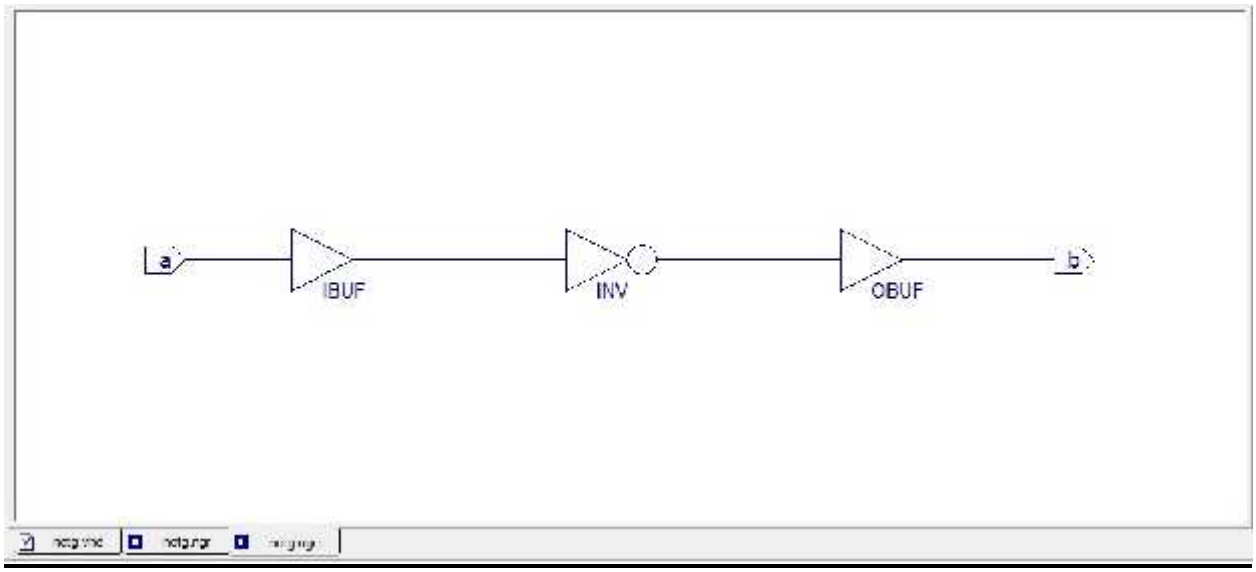
Behavioral Module:-
NOT GATE

```
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 Dataflow & Behavioral method using Xilinix 7.1i

Tools/Apparatus Used:-

(a) Xilinx 7.1i
(b) Windows 8.1
(c) 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:-

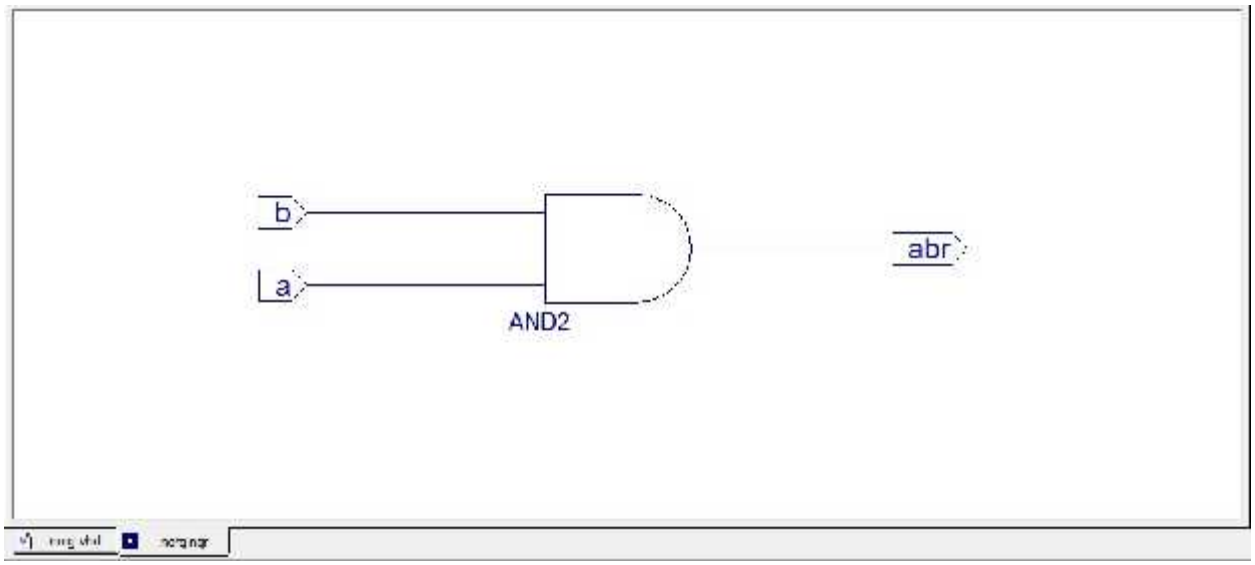
Data Flow Model:-
NOR GATE

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

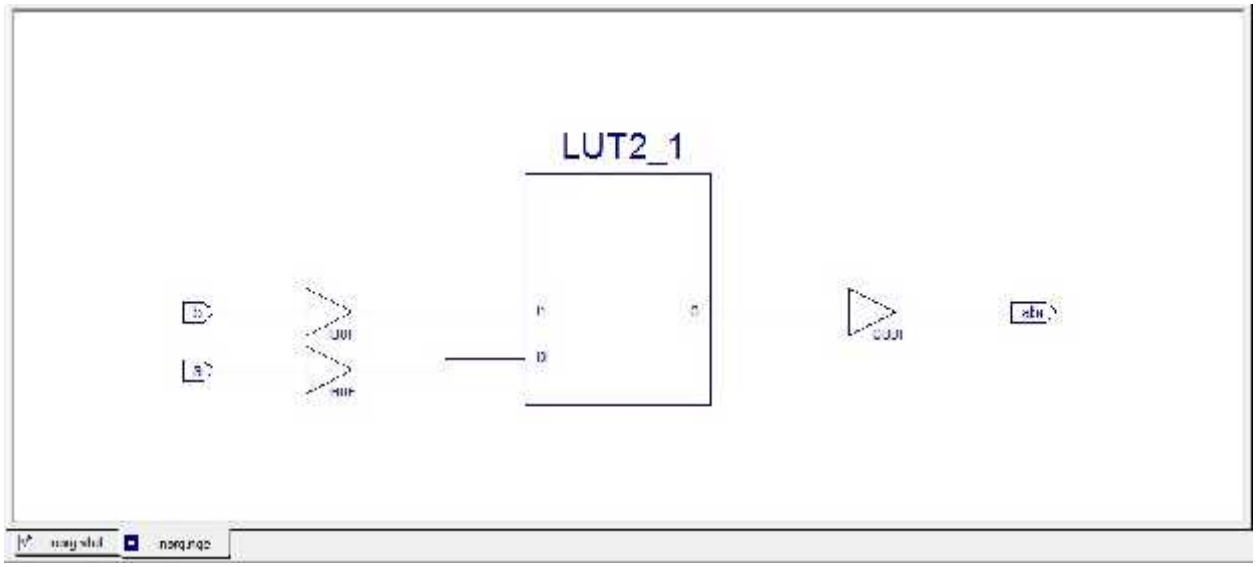
Behavioral Module:-
NOR GATE

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

end Behavioral;
```

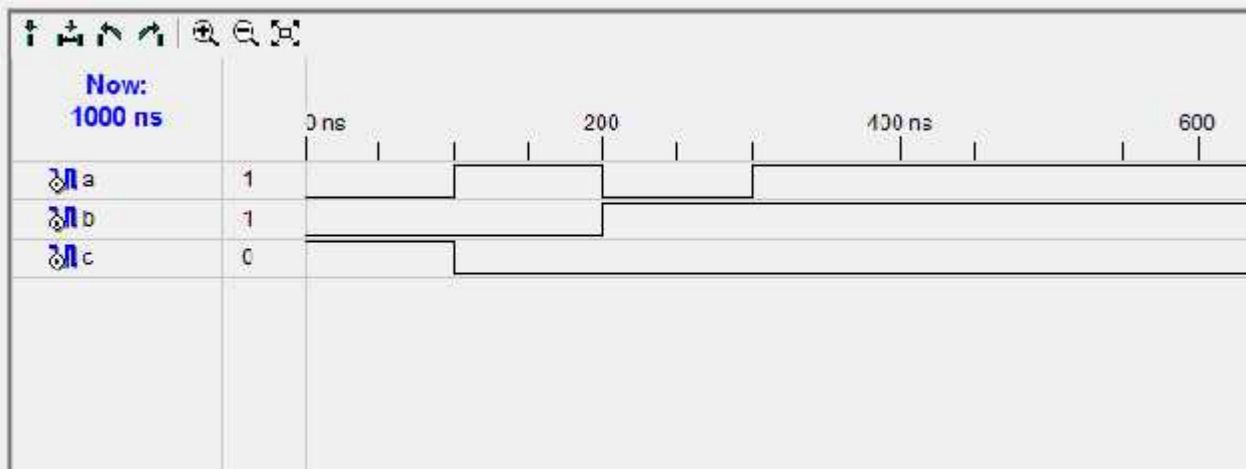
RTL Schematic:-



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 Dataflow & Behavioral method using Xilinx 7.1i

Tools/Apparatus Used:-

(d) Xilinx 7.1i
(e) Windows 8.1
(f) 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:-

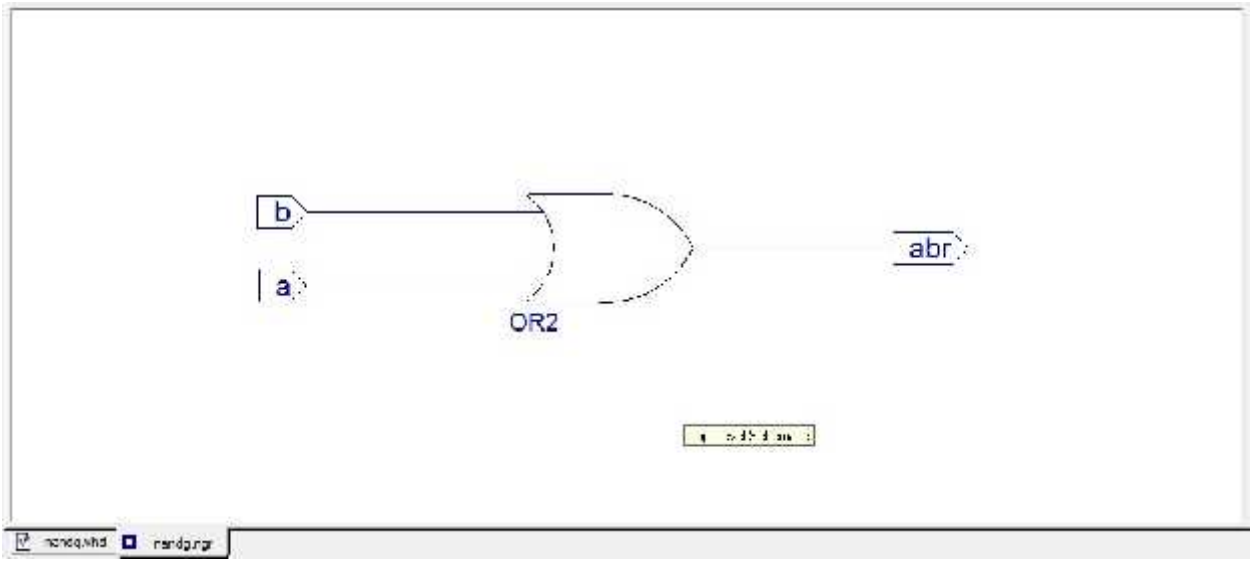
Data Flow Model:-
NAND GATE-

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

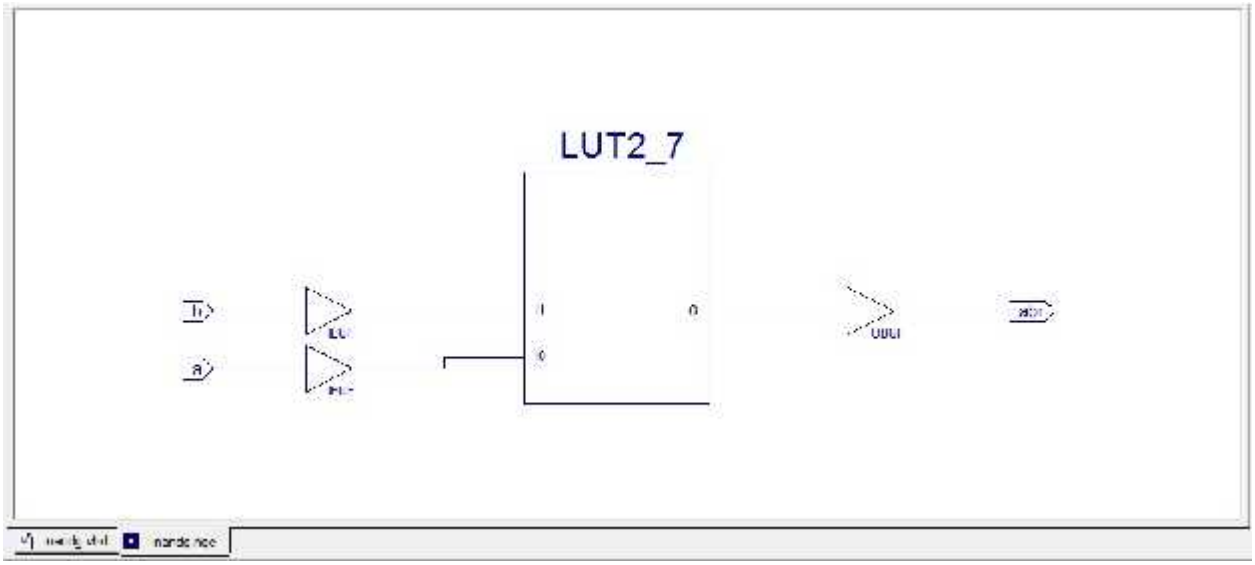
Behavioral Module:-
NAND GATE-

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

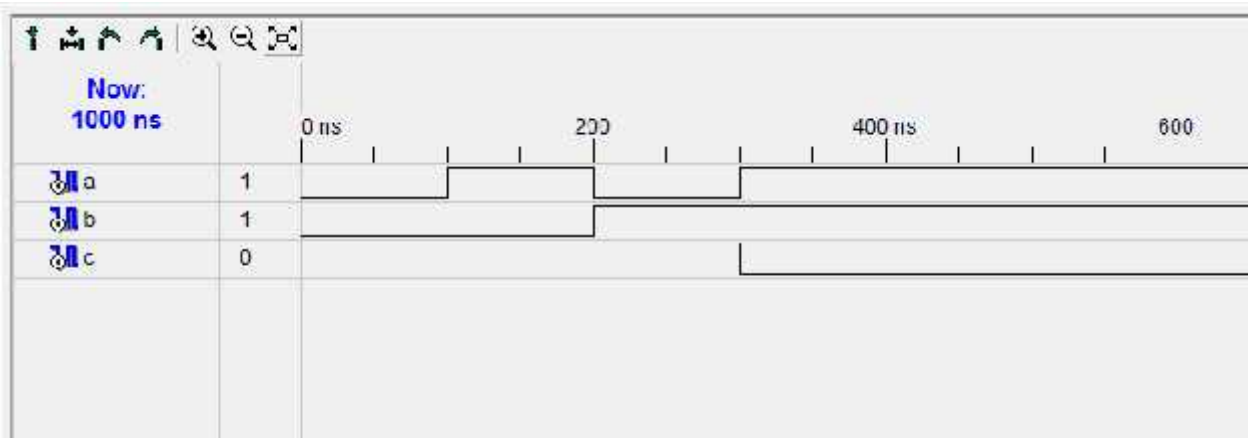
RTL Schematic:-



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 Half Adder in Dataflow & Behavioral method using Xilinx 7.1i

Tools/Apparatus Used:-

(g) Xilinx 7.1i
(h) Windows 8.1
(i) 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:-

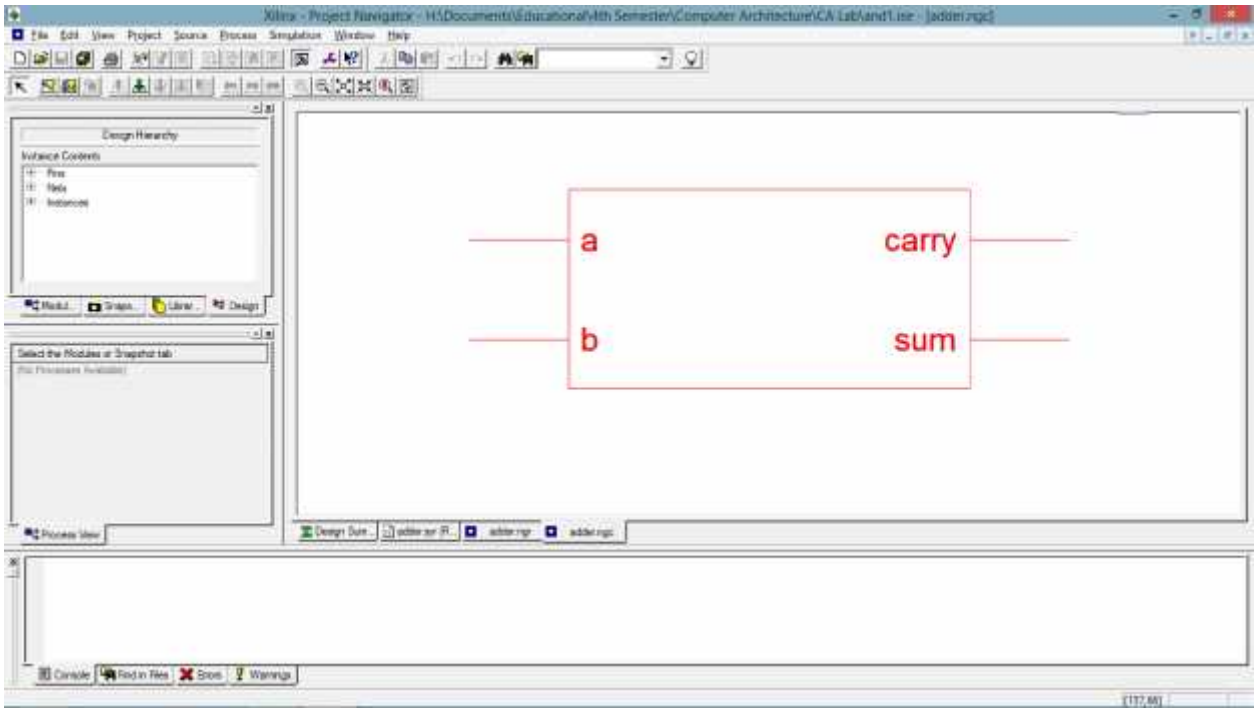
Data Flow Model-
HALF ADDER-

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

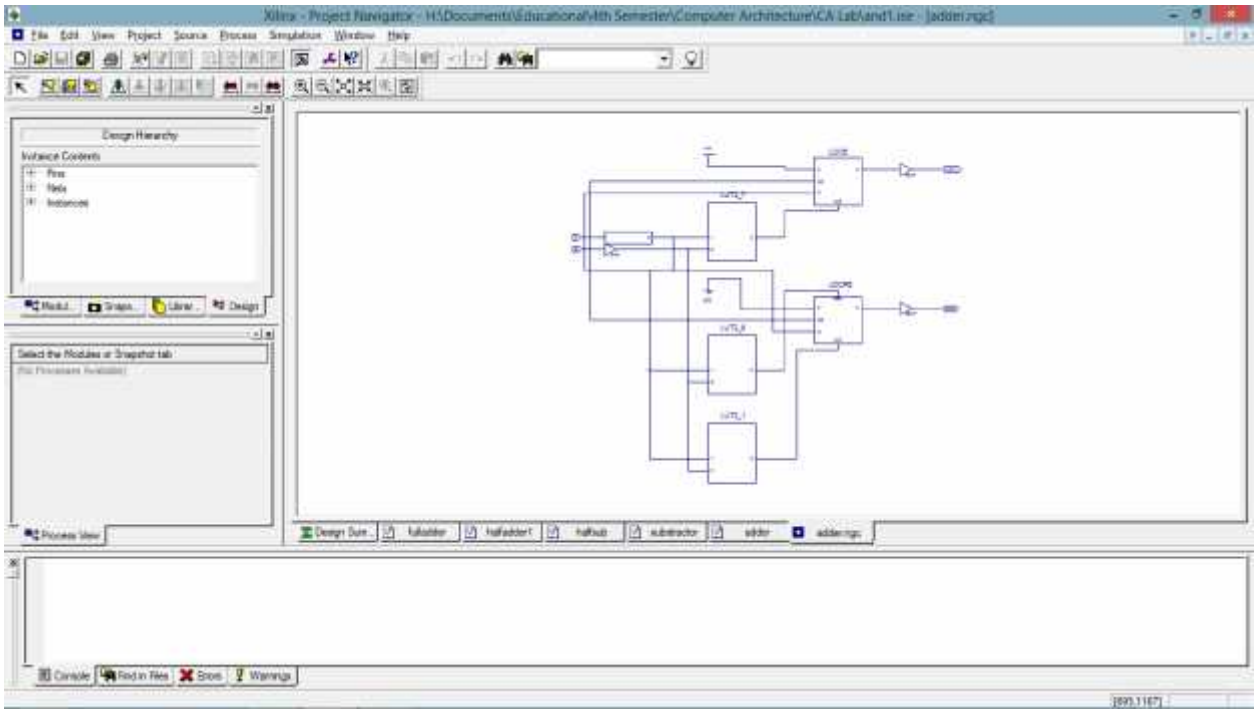
Behavioral Module:-

```
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;
```

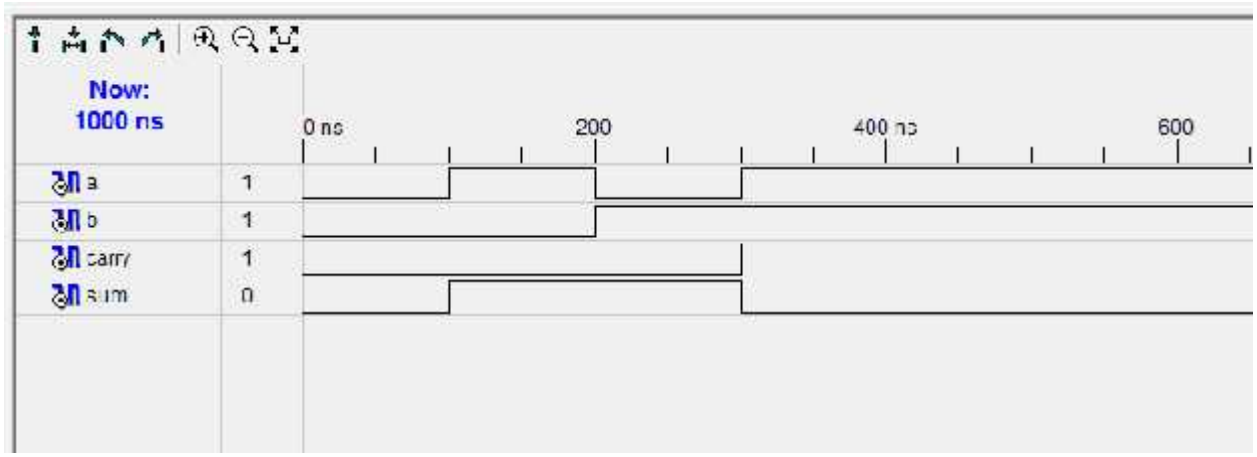
RTL Schematic:-



Technology Schematic:-



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 Dataflow & Behavioral method using Xilinx 7.1i

Tools/Apparatus Used:-

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

Procedure:-

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

Source Code:-

Data Flow Model:-

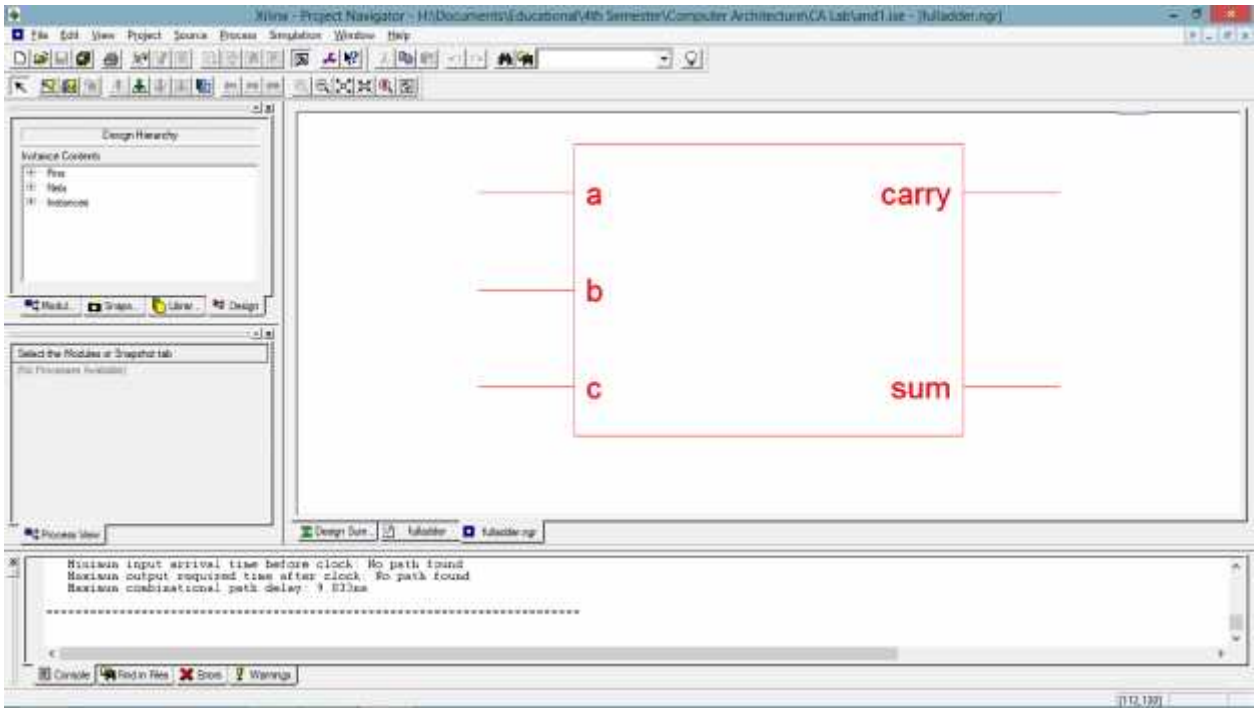
FULL ADDER:-

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

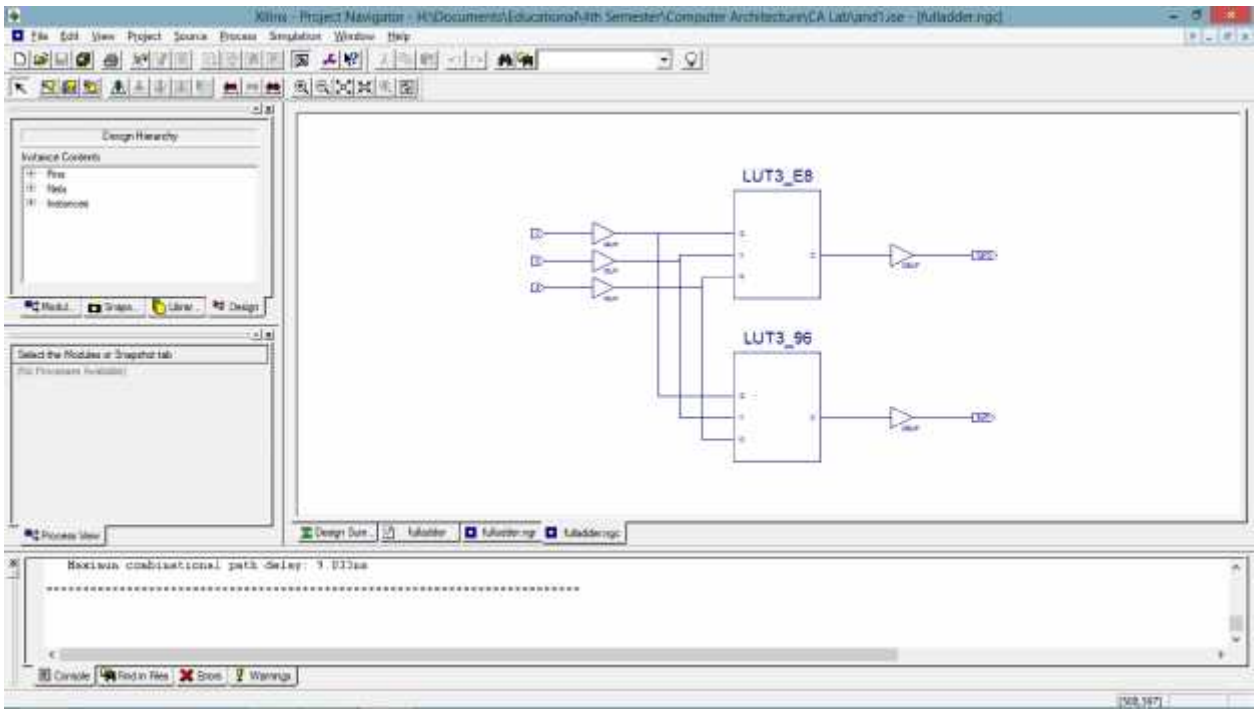
Behavioral Module:-

```
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;
```

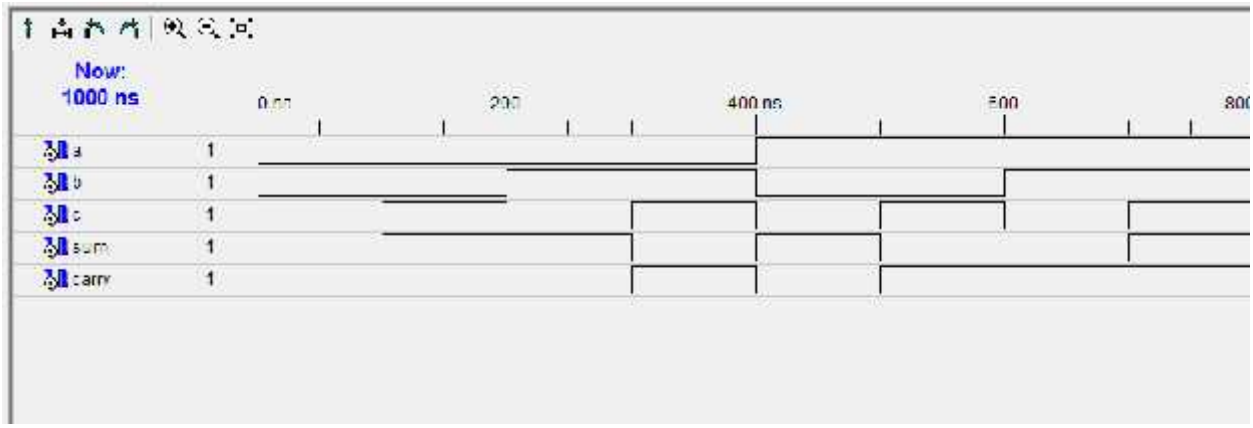
RTL Schematic:-



Technology Schematic:-



Test Bench Waveform:-



Conclusion:-

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

Experiment – 8

Aim:- To implement Full Subtractor in Dataflow & Behavioral method using Xilinx 7.1i

Tools/Apparatus Used:-

(a) Xilinx 7.1i.

(b) Windows 8.1.

(c) Other necessary requirements.

Procedure:-

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

Source Code:-

Data Flow Model:-

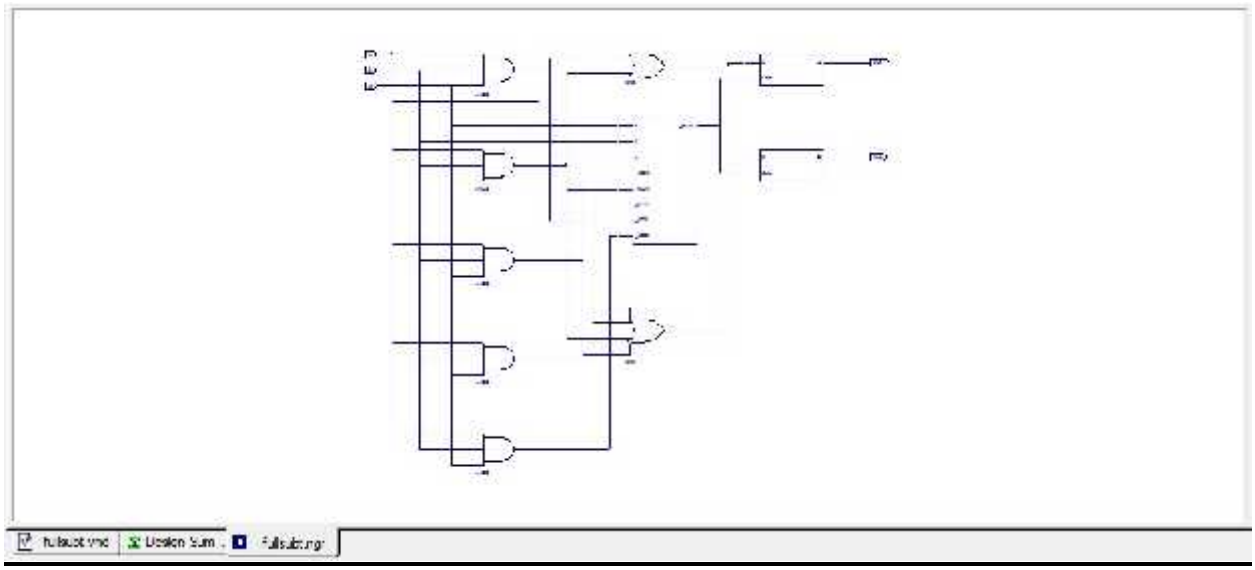
Begin

```
sub<= (a and (b xor c)) or (a not (b xor c));
borrow<= ((not a) and c);
end Dataflow;
```

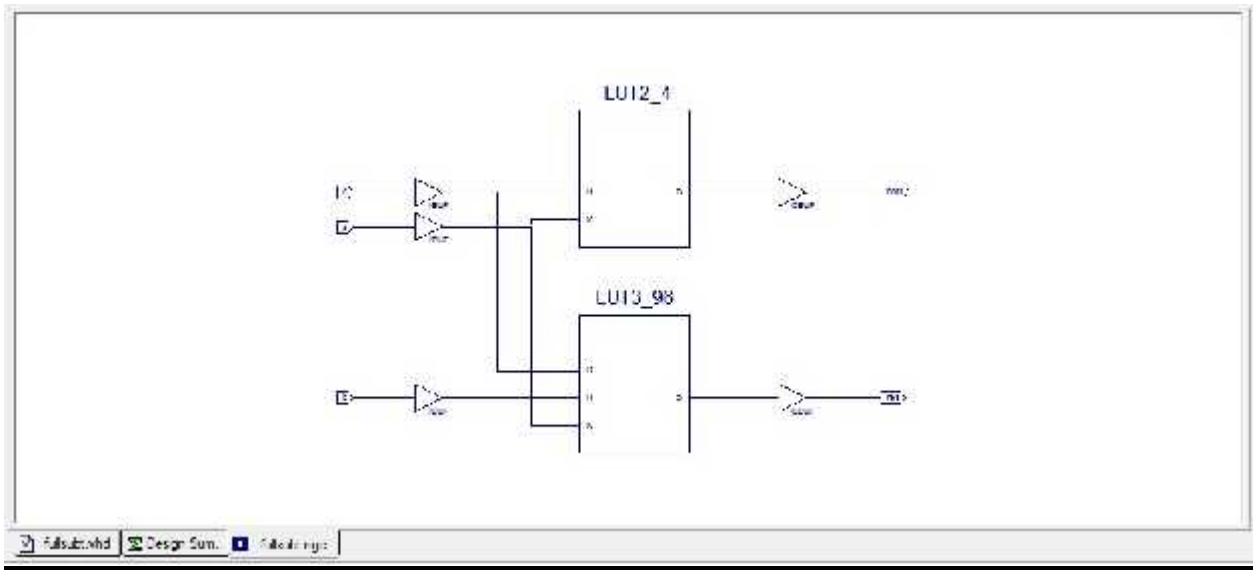
Behavioral Module:-

```
begin
p:process(x,y,bin)
begin
if ( ( x = '0' and ( y = not bin ) ) or ( x = '1' and ( y = bin ) ) ) then
d <= '1' ;
else
d <= '0';
end if;
if ( ( x = '0' and ( y = '1' or bin = '1' ) ) or ( y = '1' and bin = '1' ) ) then
bout <= '1' ;
else
bout <= '0' ;
end if;
end process;
end Behavioral;
```

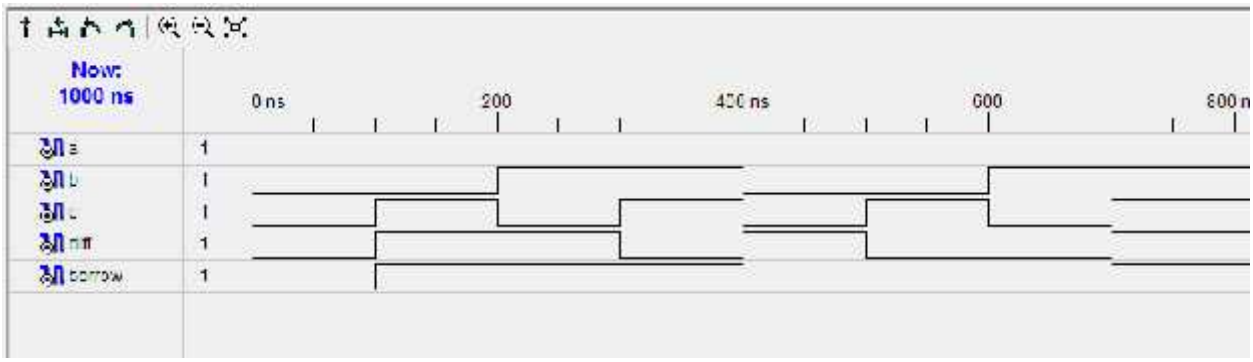
RTL Schematic:-



Technology Schematic:-



Test Bench Waveform:-



Conclusion:-

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