NATIONAL INSTITUTE OF ELECTRONICS AND INFORMATION TECHNOLOGY

DEEMED TO BE UNIVERSITY UNDER DISTINCT CATEGORY

(AN AUTONOMOUS INSTITUTION UNDER MINISTRY OF ELECTRONICS & IT, GOVT. OF INDIA)

Curriculum and Syllabi for

M.Tech - Embedded System and VLSI Design

(2025-26 Scheme)

Main Campus at Ropar and Constituent Units at Aizawl, Agartala, Aurangabad, Calicut, Gorakhpur, Imphal, Itanagar, Kekri, Kohima, Patna and Srinagar

Vision

To be known globally for value based education, research, and innovation through academic excellence to serve the needs of industry and society.

Mission

1. Develop a strong foundation on fundamentals of engineering through outcome-based teaching and learning process.
2. Provide opportunities for students to work on real-world issues and develop sustainable ethical solutions.
3. Involve the students in group activities, including those of professional bodies, to develop leadership, entrepreneurship, and good communication skills.

Program Education Objectives

PEO1: To gain expertise and proficiency in the broader domains of Embedded System and VLSI Design, enabling success in contemporary industry, academia, or research.

PEO2: To understand, assess, formulate, and develop innovative problem-solving approaches within the realm of Embedded System and VLSI Design that are technically, economically, and socially feasible and acceptable.

PEO3: To demonstrate professional proficiency and leadership attributes, harmoniously integrating ethical principles for comprehensive personality development.

Program Outcomes

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems
PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions
PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions
PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

Program Specific Outcomes

PSO1: Ability to analyze, design, and verify VLSI circuits using hardware description languages (HDL) and electronic design automation (EDA) tools, ensuring optimized performance, power, and area.
PSO2: Proficiency in integrating hardware and software components for efficient system design, leveraging FPGA, ASIC, and SoC platforms while considering trade-offs in power, performance, and cost.
PSO3: Capability to apply advanced concepts such as AI/ML, RISC-V architecture, IoT, and edge computing in embedded and VLSI domains to develop innovative and efficient solutions.
PSO4: Ability to collaborate in multidisciplinary teams, conduct research, and contribute to industry and academia by developing solutions in areas like low-power VLSI, high-speed computing, and secure embedded systems.

CATEGORY WISE CREDIT DISTRIBUTION

Category	Category Code	Credits
Open Elective Courses	OE	8
Audit Courses (without grade or credit)	AU	0
Skill / Employment Enhancement Courses (Project/Internship/Seminar/Dissertation/Research)	EE	36
Program Core Courses	PC	20
Professional Elective Courses	PE	16
Total Credits (Common track)		**80**

EVALUATION AND ASSESSMENT

1. Performance of a student in a semester shall be evaluated through continuous Class assessment, Tutorial/Lab assessment, Mid-Semester Examination (MSE) and End-Semester Examination (ESE). Both the MSE and ESE shall be the University examination and will be conducted as notified by the Controller of Examinations (CoE) of the University.

2. The continuous assessment shall be based on assignments, tutorials, paper presentation / quizzes/ viva-voce / flipped classes, lab work / projects / fieldwork and attendance, etc.

3. The MSE/ESE shall be comprising of written papers.

4. The overall assessment of the students will be done as per the following scheme:

S. No.	Assessment Type	Marks Weightage
1.	Mid Semester Examination	20
2.	End Semester Examination	40
3.	Continuous Assessment - Practical /Lab/ Tutorial	25
4.	Continuous Assessment - Theory	15
Total		**100**

For more details, please refer the applicable ordinance for this programme and Assessment SOPs.

CURRICULUM

Semester 1						
Semester Code	**Course Code**	**Course Title**	**L**	**T**	**P**	**Cr**
PE-MTEVLS-101	Elective	Program Elective				4
PC-MTEVLS-102	DOEE250062	Digital IC Design	3	0	2	4
PC-MTEVLS-103	DOEE250202	VLSI Testing & Testability	3	0	2	4
PC-MTEVLS-104	DOEE250090	Embedded Programming	3	0	2	4
EE-MTEVLS-105	DASH250095	Research Methodology and IPR	3	1	0	4
AU-MTEVLS-106	Elective	Audit Elective				0

Semester 2						
Semester Code	**Course Code**	**Course Title**	**L**	**T**	**P**	**Cr**
PC-MTEVLS-201	DOEE250088	Embedded OS and RTOS	3	0	2	4
PC-MTEVLS-202	DOEE250002	Advanced Computer architecture and SoC Design	3	0	2	4
PE-MTEVLS-203	Elective	Program Elective				4
PE-MTEVLS-204	Elective	Program Elective				4
OE-MTEVLS-205	Elective	Open Elective				4

Semester 3						
Semester Code	**Course Code**	**Course Title**	**L**	**T**	**P**	**Cr**
PE-MTEVLS-301	Elective	Program Elective				4
OE-MTEVLS-302	Elective	Open Elective				4
EE-MTEVLS-303	EE	Dissertation Phase I				14

Semester 4						
Semester Code	**Course Code**	**Course Title**	**L**	**T**	**P**	**Cr**
EE-MTEVLS-401	EE	Dissertation Phase II				18

Elective Courses

Elective Courses for PE Category						
Course Code	**Course Title**	**L**	**T**	**P**	**Cr**	**For Sem./Code**
DOEE250116	High Speed Digital Design	3	0	2	4	1
DOEE250173	Python Programming for ASIC Verification	3	0	2	4	1
DOEE250087	Embedded IOT System Design	3	1	0	4	2
DOEE250115	Hardware Design Verification	3	0	2	4	2
DOEE250185	System Design Using Embedded Processors	3	0	2	4	2
DOEE250201	VLSI Signal Processing	3	0	2	4	2
DOAI250001	Advanced Data Analytics	3	0	2	4	3
DOEE250016	Analog Mixed Signal VLSI Design	3	0	2	4	3

Elective Courses for OE Category
Students may opt courses under MOOC, NPTEL, SWAYAM, NSQF/NCVET aligned courses or other relevant technical subjects not included in their core curriculum. The exercise of option shall be subject to the Course Schedule of the respective Constituent Unit, the credit requirements and availability of seats. Guidelines for choosing MOOC/Online courses are given separately and shall have to be adhered to.

Elective Courses for AU (Audit) Category					
Course Code	**Course Title**	**L**	**T**	**P**	**Cr**
DASH240027	Disaster Management	2	0	0	0
DASH240047	English for Research Paper Writing	2	0	0	0
DASH240052	Environmental Science	3	0	0	0
DASH240085	Pedagogy Studies	2	0	0	0
DASH240086	Personality Development through Life Enlightenment Skills	2	0	0	0
DASH240097	Sanskrit for Technical Knowledge	2	0	0	0
DASH240103	Stress Management by Yoga	2	0	0	0
DASH240104	Technical Presentation and Report Writing	0	0	2	0
DASH240106	Value Education	2	0	0	0
DASH240124	Constitution of India - AU	2	0	0	0
DASH250023	Constitution of India	2	1	0	0
DASH250027	Disaster Management	3	0	0	0
DASH250034	Energy and Environmental Engineering	3	0	0	0
DASH250047	English for Research Paper Writing	2	0	0	0
DASH250054	Essence of Indian Knowledge and Tradition	2	0	0	0
DASH250085	Pedagogy Studies	2	0	0	0
DASH250086	Personality Development	2	0	0	0
DASH250097	Sanskrit for Technical Knowledge	3	0	0	0
DASH250103	Stress Management by Yoga	3	0	0	0
DASH250106	Value Education	2	0	0	0

DASH250117	Innovative Thinking and Entrepreneurship Skills	1	0	0	0
DASH250118	Intellectual Property Rights	2	0	0	0
DCSA240080	Training on Computer Networking	0	0	2	0
DCSA240304	Lab Based on Statistical Package	0	0	2	0
Any other Scheduled Course as per the Course Schedule of the respective Constituent Unit can also be chosen, subject to availability of seats. However, there shall be no assessment and grading for the course chosen as Audit Course.					

The choice of all type of Electives available shall be as per the Course Schedule of the respective Constituent Unit.

Guidelines for Massive Open Online Courses (MOOC) / approved Online Courses

1. Relevant Swayam, MOOC, NPTEL, NIELIT NSQF and any other online courses approved by UGC/AICTE shall also be allowed as Open Electives(OE).

2. One credit of MOOC course should be minimum of 30 hours.

3. A student can choose any approved online course as Open Elective, subject to minimum credit requirements.

4. The students willing to opt OE under MOOC in their next semester, will submit their request to the MOOC Coordinator at their respective campus.

5. Once approved, the campus shall display the list of accepted courses.

6. The student has to submit certificate issued by the institution offering the MOOCs course, along with the number of credits and grades, to get credits transferred into his/her marks certificate issued by NDU.

7. The transfer of credits shall be as adopted by NDU.

Detailed Syllabus

Course Code	Course Name	L	T	P	Cr
DOEE250062	Digital IC Design	3	0	2	4

Course Outcomes:

CO1. Demonstrate proficiency in digital IC design concepts, methodologies, and tools.
CO2. Write synthesizable HDL code and simulate digital IC designs using industry-standard simulation tools.
CO3. Design and implement digital IC layouts that meet design requirements and constraints.
CO4. Convert digital IC designs into FPGA prototypes for verification and testing.

Module 1:

CMOS VLSI Design:
Overview of VLSI technology and its significance, Semicustom VLSI Design flow: frontend and back-end. CMOS Transistor Theory- MOSFET operation, characteristics and scaling. CMOS inverter operation and voltage transfer characteristics, Inverter sizing and optimization techniques

Module 2:

CMOS Logic Design:
Basic logic gates: AND, OR, NAND, NOR, XOR. Designing complex logic functions using CMOS logic gates. Gate Delays and Logical Effort- Gate delay modeling and estimation, logical effort analysis, Determining the best delay-power trade-off for logic gates using P/N ratio. Transistor Level Schematics and Layouts- Transistor level design techniques, Schematic design and layout considerations, Design rules, metal layers, and interconnect routing.

Module 3:

Hardware modelling and verification using Verilog HDL:
Introduction to Verilog HDL- syntax and constructs. Hardware modelling-behavioral, dataflow and structural. Register Transfer Level (RTL) design of digital circuits-modeling using Verilog HDL. Functional verification overview. Writing Verilog test benches and RTL verification

Module 4:

RTL to GDSII:
RTL to GDSII design flow. Overview of Logic synthesis. RTL coding for synthesis. Constraints and RTL synthesis. Physical design- Floor plan and power planning, Placement and clock tree synthesis, Routing, Physical Verification and DFM Checks, static timing analysis and ECO fixing, timing closure and Tape out

Module 5:

FPGA Prototyping of Digital ICs:
Overview of FPGAs-architecture and components. Advantages of FPGAs. FPGA design flow-design entry, synthesis, floor planning, placement, routing, and bitstream generation. Programming FPGA. Embedded Logic analyzers and debugging. Xilinx 7-series FPGA-a case study.

Labs / Practicals:

1. Modelling and simulation of CMOS Inverter
2. Modelling and simulation of basic gates.

3. CMOS Inverter layout design, simulation and analysis.
4. Layout design and analysis of basic gates.
5. Hardware modelling and simulation of ALU, 8 bits up down counter.
6. Hardware modeling and verification of a decade counter.
7. RTL synthesis and analysis of a decade counter.
8. RTL to GDSII generation of a decade counter
9. FPGA programming of a decade counter
10. Decade counter debugging in FPGA fabric using Embedded logic analyzers

References:

1. Digital Integrated Circuits – A Design Perspective, Jan M. Rabaey, Anantha Chandrakasan, Borivoje Nikolic, 2nd Ed., PHI.
2. Introduction to VLSI Systems: A Logic, Circuit and System Perspective – Ming-BO Lin, CRC Press, 2011.
3. Verilog HDL - A guide to Digital Design and Synthesis by Samir Palnitkar.
4. Writing Testbenches: Functional Verification of HDL Models by Janick Bergeron
5. FPGA-Based System Design by Wayne Wolf
6. Advanced FPGA Design Architecture, Implementation and optimization by Kilts

Course Code	Course Name	L	T	P	Cr
DOEE250202	VLSI Testing & Testability	3	0	2	4

Course Outcomes:

CO1: Describe different types of faults in VLSI circuits and understand fault modeling.
CO2: Analyze and apply algorithms for automatic test pattern generation (ATPG).
CO3: Implement DFT techniques such as scan design and built-in self-test (BIST).
CO4: Evaluate the effectiveness of test coverage and simulation tools.
CO5: Design a testable VLSI system incorporating testability features.

Module 1:

Fundamentals of VLSI Testing:
Importance of testing and test economics, Yield, defects, fault types (physical vs logical faults), Levels of testing: wafer, package, board, Manual vs automated testing; fault coverage, Fault modeling: Stuck-at fault model, Bridging fault model, Delay faults (transition and path delay faults). Fault equivalence and dominance

Module 2:

Fault Simulation and ATPG:
Fault simulation:Serial, parallel, and deductive fault simulation. Test generation algorithms: D-algorithm, PODEM (Path-Oriented Decision Making), FAN (Fanout-Oriented). Testability measures: SCOAP metrics, Circuit observability and controllability

Module 3:

Design for Testability (DFT) Techniques:
Need for DFT in large-scale designs, Scan Design: Full scan, partial scan, Level-sensitive scan design (LSSD). Boundary scan (IEEE 1149.1 JTAG), Compression techniques and test point insertion, Test access mechanisms (TAM)

Module 4:

Built-In Self-Test (BIST) and Memory Testing:
BIST Architecture and components: Test pattern generator (LFSR), Output response analyzer (signature analysis). Types of BIST: Logic BIST, Memory BIST (MBIST). March algorithms for memory testing, Fault coverage analysis in BIST

Module 5:

Testability in Practice and Advanced Topics:
Testing low power VLSI designs, IDDQ testing (quiescent current testing), Delay fault testing techniques, Test data compression, Introduction to analog and mixed-signal testing, Overview of commercial tools (Mentor Tessent, Synopsys TetraMAX), Case study: Scan insertion in ASIC flow

Labs / Practicals:

1. Simulation of Stuck-at Faults using a VLSI simulator (e.g., ModelSim or Cadence NC-Sim)
2. Fault coverage analysis for combinational circuits using D-algorithm
3. Implementation of PODEM algorithm in Python or Verilog
4. Observability and controllability measurement (SCOAP metrics)
5. Scan flip-flop insertion in a sequential circuit

6. Boundary Scan implementation and test signal tracing
7. Design of LFSR-based test pattern generator
8. Signature analysis and MISR implementation for BIST
9. Simulation of a March test algorithm for memory arrays
10. Mini Project: Integrating scan and BIST for a custom VLSI design block

References:

Primary Textbook:
1. "Digital Systems Testing and Testable Design" by Miron Abramovici, Melvin A. Breuer, and Arthur D. Friedman
Other References:
2. "Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits" by M.L. Bushnell and V.D. Agrawal
3. "CMOS Digital Integrated Circuits: Analysis and Design" by Sung-Mo Kang & Yusuf Leblebici
4. IEEE 1149.1 Standard Documentation (Boundary Scan)

Course Code	Course Name	L	T	P	Cr
DOEE250090	Embedded Programming	3	0	2	4

Course Outcomes:

CO1: Understand fundamental concepts of programming, algorithms, flowcharts, and the basics of computation for effective problem-solving and logical thinking.

CO2: Apply C programming constructs including data types, operators, control statements, functions, arrays, and pointers to develop modular programs.

CO3: Develop and manipulate advanced C structures such as unions, linked lists, and other data structures, with applications in embedded system contexts.

CO4: Demonstrate object-oriented programming concepts using C++, including class hierarchies, templates, and exception handling for robust code development.

CO5: Analyze and apply embedded programming standards such as MISRA C/C++, use compiler directives, and perform code optimization and profiling for reliable and efficient embedded software development.

CO6: To understand Programming & algorithms for problem solving

CO7: To understand the Programming Concepts

CO8:Build Application using High level languages C/C++ Programming

Module 1:

Introduction to Programming & algorithms for problem solving

The Basic Model of Computation, Algorithms, Flow-charts, Programming Languages, Compilation, Linking and Loading, Testing and Debugging, Algorithms for Problem Solving: Decimal Base to Binary Base conversion, Reversing digits of an integer, GCD (Greatest Common Division), LCM etc. , Use case diagrams and UML.

Module 2:

C Programming Basics

GNU Tools: gcc, gdb, gprof, Makefiles, Basic data types, operations, and flow control (decision-making statements), Flow control (loops), typecasting, and computer logic, Switch-case, arrays, and the basics of strings, pointers, functions, storage class.

Module 3:

Advanced C programming for Embedded Systems

Structure and union, Linear and nonlinear data structures, Linked List

Module 4:

Object Oriented Programming concepts with C++
Overview of C++, , Fundamentals of the object-oriented approach, Class hierarchy, Advanced class concepts, Templates, Accessing data and dealing with exceptions.

Module 5:

Embedded Programming Coding standards & Concepts
Coding Standards For Compliance , ANSI C, C89, C95, C99, C11, MISRA C & C++, Profiling & Optimisation

techniques, Pragma , floating-point exceptions rounding multi-precision libraries (GMP, MPFR, MPIR), IEEE754, Static and dynamic Code analysis.

Labs / Practicals:

Develop algorithms/ Flow diagrams for any one from the following
1.Decimal Base to Binary Base conversion
2.Reversing digits of an integer
3. GCD Greatest Common Division)
4. LCM

Familiarisation of UML diagrams With the help of open source UML tools
1. Draw UML diagrams
2. Export diagrams
3. Share diagrams using Eclipse
4. Create new, custom UML elements
5. Build sequence and activity diagrams from plain text

C Programming using Linux platforms
1. Familiarisation of Linux Commands
2. Example usage of gcc compiler with sample C programs
3. C programming covering data types, operators, loops, strings

C Programming using Linux platforms
1. Examples using Pointers using C
2. Examples using functions, pass by value, pass by reference methods.
3. Examples covering the structure and union

Advanced C Programming for Embedded Systems (Linear & Nonlinear data structure implementation using C Language)
1. Write a c program to implement queue and its operations
2. Write a C Program to implement stack and its operation
3. Write a C program to implement singly linked list and its operations
4. Write a C program to implement circular linked lists and its operations.

Class hierarchy C++ programming
1. Write program to understand the class concepts in C++
2. Write C++ program to cover inheritance concepts in C++
3. Write C++ program to cover overloading concepts in C++.

Advanced class concepts using C++ programming
1. Write C++ program to cover virtual functions in C++.
2. Write C++ program to cover Templates, Accessing data and dealing with exceptions.

References:

Text Books
1. C Programming language, Kernighan, Brian W, Ritchie, Dennis M, Prentice Hall PTR
2. “Embedded C”,Michael J. Pont, Addison Wesley Reference

Reference Books
1. The Complete Reference C++, Herbert Schildt, TMH
2. GNU C++ For Linux, Tom Swan , PrenticeHall India
3. Daniel W. Lewis, “Fundamentals of embedded software where C and assembly meet”, Pearson Education, 2002.

Course Code	Course Name	L	T	P	Cr
DASH250095	Research Methodology and IPR	3	1	0	4

Course Outcomes:

CO1: Understand the principles and process of research methodology.
CO2: Apply appropriate research design and data collection methods in computing projects.
CO3: Analyze and interpret qualitative and quantitative data using statistical tools.
CO4: Develop ethically sound, well-documented research or project reports conforming to academic and industrial standards.
CO5: Understand various forms of intellectual property and apply processes for protection and patent filing.

Module 1:

Introduction to Research and Research Ethics
Definition and objectives of research, types of research – fundamental, applied, exploratory, empirical, significance of research in computer applications, research ethics and integrity, plagiarism, citation styles (APA, IEEE), tools for plagiarism check.

Module 2:

Research Design and Data Collection
Research problem identification and formulation, hypothesis generation, literature review techniques, types of research design: experimental, descriptive, analytical, sampling techniques, primary and secondary data sources, survey design, case study design.

Module 3:

Data Analysis and Interpretation
Quantitative and qualitative data analysis, measures of central tendency and dispersion, t-test, ANOVA, correlation and regression, chi-square test, software tools: Excel, R, SPSS (demo-level), data visualization and result interpretation.

Module 4:

Technical Writing and Project Report Preparation
Structure of research papers and reports, abstract, introduction, literature review, methodology, results, and conclusions, referencing tools (Mendeley/Zotero), IEEE/ACM/SCI journal guidelines, thesis and dissertation formatting, proposal writing basics.

Module 5:

Intellectual Property Rights and Patent Process
Introduction to IPR: patents, copyrights, trademarks, industrial design, Indian and international patent systems (WIPO, TRIPS), criteria for patentability, patent filing process in India (IPINDIA portal), open-source licenses, patent drafting basics, IPR in software and AI innovations.

Labs / Practicals:

N/A

References:

1. C.R. Kothari & Gaurav Garg – Research Methodology: Methods and Techniques, New Age International.
2. Ranjit Kumar – Research Methodology – A Step-by-Step Guide, Sage Publications India.
3. P. Narayan – Intellectual Property Law, Eastern Book Company.
4. Neeraj Pandey & Khushdeep Dharni – Intellectual Property Rights, PHI Learning.
5. Yogesh Kumar Singh – Fundamentals of Research Methodology and Statistics, New Age International.
6. Bharat Bhushan – Research Methodology in Computer Science, Himalaya Publishing.
7. WIPO and IPIndia Manuals – Guidelines for Filing Patents in India.

Course Code	Course Name	L	T	P	Cr
DOEE250088	Embedded OS and RTOS	3	0	2	4

Course Outcomes:

CO 1 Attain comprehensive understanding of Embedded OS (Linux) fundamentals
CO 2 Attain comprehensive understanding of Embedded Processor and its software
CO 3 Attain comprehensive understanding of Embedded Linux Drivers
CO 4 Attain comprehensive understanding of basics of real-time concepts
CO 5 Attain comprehensive understanding of incorporating RTOS in an embedded system
CO 6 Attain comprehensive understanding of applying the knowledge for developing practical applications of modern real-time systems
CO 7 Get knowledge in Embedded OS (Linux) fundamentals
CO 8 Understand Embedded Linux Internal programming
CO 9 Comprehend Embedded Linux Drivers
CO 10 Learn about the basics of real-time OS Programming concepts

Module 1:

Introduction to Embedded Linux

Overview of Linux OS, Directory structures, basic Linux shell commands, Overview of Systems Calls, Classification of system Calls, Inter Process Communication, Multithreading and Thread Management

Module 2:

Embedded Linux Driver Development

Linux Kernel Module Programming, Character device driver development, USB device driver development, Block, Network and PCI device driver development

Module 3:

Building and Customisation of Linux for Embedded systems

Linux booting procedure, Bootloader, hypervisor, opensbi, qemu, Linux build tools - Buildroot and Yocto

Module 4:

Overview of Real Time OS

Basics of RTOS: Real-time concepts, Hard Real time and Soft Real-time, Differences between General Purpose OS & RTOS, Basic architecture of an RTOS, Scheduling Systems, RTOS Issues – Selecting a Real Time Operating System

Module 5:

RTOS for Embedded Applications

FreeRTOS, Thread creation & Management, Inter thread Communication, Mutual Exclusion, MbedOS, Other real time OS (VxWorks, Azure RTOS, SAFERTOS etc.)

Labs / Practicals:

Embedded Linux shell commands
Embedded Linux System Call programming
Embedded Linux Interprocess Communication
Development of Multithreaded application in Linux
Performing Thread Synchronization and resource protection (Mutex) based applications in Linux
Develop simple Kernel Module Programming
Develop basic character Device driver in Linux
Familiarization of FreeRTOS, Thread creation, synchronization - Port to FreeRTOS to ARM Cortex M4 development board

References:

1. GNU/LINUX Application Programming, Jones, M Tims
2. Sreekrishnan Venkateswaran Essential Linux Device Drivers, Prentice Hall 2008
3. Embedded/Real Time Systems Concepts, Design and Programming Black Book, Prasad, KVK
4. Software Design for Real-Time Systems: Cooling, J E Proceedings of 17the IEEE Real-Time Systems Symposium December 4-6, 1996 Washington, DC: IEEE Computer Society
5. FreeRTOS Reference Manual

Course Code	Course Name	L	T	P	Cr
DOEE250002	Advanced Computer architecture and SoC Design	3	0	2	4

Course Outcomes:

CO1: Understand the principles of advanced computer architecture, including instruction-level parallelism, thread-level parallelism, and memory hierarchy design.
CO2: Gain knowledge of modern processor architectures, including superscalar processors, out-of-order execution, and speculative execution techniques.
CO3: Apply the knowledge gained to undertake a practical SoC design project, from architectural exploration and modeling to implementation and validation, demonstrating the ability to solve real-world problems in advanced computer architecture and SoC design
CO4: Develop critical thinking and problem-solving skills through analysis of existing SoC architectures, identifying performance bottlenecks, and proposing innovative design solutions to improve system performance and efficiency.

Module 1:

Processor Architecture and instruction set:
Fundamentals of instruction set architecture (ISA), memory design hierarchy, advanced computer architecture-superscalar processors, accelerators, vector processors. ARM and Digital India RISC-V architecture. Instruction set architecture design: Instruction set design, implementation and performance perspectives, relative advantages of RISC and CISC instruction set, data path design.

Module 2:

Instruction level parallelism (ILP):
Pipeline data-path, data dependance. Challenges in ILP realization. Pipeline hazards and their solutions, out-of-order execution, branch prediction, and dynamic scheduling. VLIW and super scalar processors.

Module 3:

Memory Systems:
Overview of memory hierarchy. Cache design considerations, instruction vs data caches, write policy and replacement policy, analysis of cache performance, and cache design for performance enhancement. Memory technologies-SRAM, DRAM and Flash. DDR memory protocol.

Module 4:

ARM/DIR-V processor based SoC Design:
Overview of SoC, SoC design flow. Components of ARM/DIR-V processor based SoC. Basic building blocks, Peripheral and Memory. ARM/DIR-V processor based SoC a case study. AMBA/AXI Bus architecture. AMBA/AXI peripheral interfacing-GPIO, Timer, 7-segment display and UART. Interrupt mechanisms. Address, control and data path establishment in an SoC. SoC connectivity and integration. SoC simulation and synthesis. Programming SoC using Embedded C and simulation.

Module 5:

Design of Hardware accelerators and advanced SoCs:
Overview of hardware accelerators. Hardware accelerator for AI/ML application-a case study. Design and interfacing accelerator IP as a co-processor. Co-processor interfacing techniques-tightly and loosely coupled. ISA expansion for custom instructions. Simulation and synthesis of accelerator IP and integrated SoC. Multi-core SoC design and integration. ARM/DIR-V based multi-core SoC a case study. Programming using Embedded C and simulation.

Labs / Practicals:

1. RTL analysis of a RISC processor
2. Design and modelling of a hardwired control unit to execute a minimum of 4 instructions.
3. Experiment to demonstrate pipe-lined and out-off order execution.
4. Model and simulate a cache controller FSM
5. Interfacing peripherals with a processor core.
   a)GPIO
   b)UART
   c)Timer
   d)Memory
6. Experiment to demonstrate processor interrupt mechanisms.
7. Designing a hardware accelerator and interfacing as a peripheral.
8. Implementation and Simulation of Branch Prediction Mechanisms
9. DMA Controller Design and Integration with Processor
10. Design and Verification of a Custom Instruction in RISC Processor

References:

1. Computer architecture: A quantitative approach by John L Hennessy, David A Patterson.
2. Computer organization and architecture: by William Stallings. Prentice Hall, 10th edition, 2015.
3. ARM System-on-Chip Architecture by Steve B. Furber
4. https://vegaprocessors.in/
5. https://shakti.org.in/

Course Code	Course Name	L	T	P	Cr
DOEE250116	High Speed Digital Design	3	0	2	4

Course Outcomes:

CO1: Understand the fundamentals of high-speed digital design, including signal propagation, transmission lines, and digital timing analysis.
CO2: Gain proficiency in using simulation tools and techniques to analyze and validate high-speed digital designs, including signal integrity analysis
CO3: Learn about clock distribution and synchronization techniques for high-speed digital systems, including clock skew management, clock tree synthesis, and phase-locked loops (PLLs).
CO4: Develop critical thinking and problem-solving skills through analysis of high-speed digital design challenges, identifying potential issues, and proposing effective solutions.

Module 1:

Introduction to high-speed digital design:
Frequency, time and distance - Capacitance and inductance effects. High seed properties of logic gates. Wire modelling. Transmission lines.

Module 2:

Power distribution and noise:
Power supply network-power management for high-speed designs. Noise sources in digital system-power supply noise. Power supply isolation.

Module 3:

Signaling convention and circuits:
Signaling modes. Signaling over various transmission mediums. Transmitter and receiver circuits.

Module 4:

Timing convention and synchronization:
Timing fundamentals. Open loop and closed loop timing. Clocking schemes. Clock management for high-speed designs. PLL and DLL based clock aligners. Closed loop clock distribution. Clock domain crossings.

Module 5:

Power and Signal Integrity Analysis:
High speed PCB design considerations. EDA tools. Layer stack-up. Pre and post route signal integrity analysis. Power integrity analysis.

Labs / Practicals:

1. Defining stack up for an impedance-controlled PCB and signal integrity analysis for single and differential trace.
2. Create a simple schematic and simulate. Observe the effects of critical net length, rise time, topology and termination.
3. Design a bus, and to guarantee that no more than 150mV of crosstalk can occur on any of the signals. Use SI tool’s crosstalk option to meet design goals, and develop the proper routing constraints to achieve it.
4. Design a SATA1.0 circuit driving a simple, matched test load of 100 ohms (differential.) and study how to reduce crosstalk on differential pairs
5. Experiments to demonstrate the power integrity analysis.
6. Design and simulate clock domain crossing interfaces (Higher clock domain to lower and vice versa).
7. High speed board design a case study.

8. Jitter Analysis in High-Speed Serial Links
9. Pre-Layout and Post-Layout Signal Integrity Analysis
10. S-Parameter and Channel Modeling for High-Speed Interfaces

References:

1. Howard Johnson and Martin Graham, "High Speed Digital Design: A Handbook of Black Magic by",3rd Edition, (Prentice Hall Modern Semiconductor Design Series' Sub Series: PH Signal Integrity Library), 2006
2. Stephen H. Hall, Garrett W. Hall, and James A. McCall " High-Speed Digital System Design: A Handbook of Interconnect Theory and Design Practices by", Wiley, 2007.
3. Kerry Bernstein, K.M. Carrig, Christopher M. Durham, and Patrick R. Hansen "High Speed CMOS Design Styles", Springer Wiley 2006.
4. Ramesh Harjani "Design of High-Speed Communication Circuits (Selected Topics in Electronics and Systems)" World Scientific Publishing Company 2006.
5. William J Dally, John W. Poulton, "Digital Systems Engineering", Cambridge University Press 2012.

Course Code	Course Name	L	T	P	Cr
DOEE250173	Python Programming for ASIC Verification	3	0	2	4

Course Outcomes:

CO1: Gain proficiency in Python programming language, including its syntax, data structures, object-oriented programming concepts, and libraries relevant to ASIC verification.
CO2: Learn how to develop verification testbenches and test cases using Python-based verification frameworks, such as Universal Verification Methodology (UVM) or PyTest.

Module 1:

Introduction to Python Programming:
Introduction to scripting language, Parts of Python Programming Language. Control Flow Statements, Functions, Strings - Lists - Dictionaries - Tuples and Sets.

Module 2:

Modules, packages and Libraries in Python:
Python Modules and Packages - Creating Modules and Packages, Libraries for Python - Library for Mathematical functionalities and Tools. Networking Libraries, Numerical Plotting Library - GUI Libraries for Python - Imaging Libraries for Python.

Module 3:

Python OOPs Concepts:
Python Class, Python Objects, creating class and objects. Inheritance, Polymorphism, and Encapsulation in Python. Data abstraction.

Module 4:

Python test benches and test automation:
Test automation by Python scripting language. Python test benches. Cocotb-Python verification framework. Verification using Cocotb frame work.

Module 5:

Python for Methodology based Verification:
Python and UVM- pyuvm, writing UVM code on top of Cocotb. TLM, configuration database and UVM factory in pyuvm. Verification using pyuvm.

Labs / Practicals:

1. Python based script for test automation of memory and FIFO.
2. Python Object Oriented Programming Exercise: Class and Object creation, Instance variables and Methods, and Class level attributes, Model systems with class inheritance i.e., inherit From Other Classes, Parent Classes and Child Classes.
3. Writing Python test benches.
4. Memory verification using Cocotb frame work.
5. Basic pyuvm exercises
6. Memory verification using pyuvm.
7. Python Script for Parsing and Analyzing Log Files
8. Automating Simulation Runs and Regression Testing Using Python
9. Advanced pyuvm Exercises: Creating Sequences, Drivers, and Scoreboards

10. Python Script for Coverage Data Collection and Visualization

References:

1. Gowrishankar S and Veena A, "Introduction to Python Programming", CRC Press, Taylor & Francis Group, 2019
2. Fabrizio Romano, “Learn Python Programming”, Second Edition, Packt Publishing, 2018.
3. Python for RTL Verification: A complete course in Python, cocotb, and pyuvm Kindle Edition by Ray Salemi.

Course Code	Course Name	L	T	P	Cr
DOEE250087	Embedded IOT System Design	3	1	0	4

Course Outcomes:

CO1: Understand the fundamental concepts and principles of the Internet of Things (IoT), including sensors, actuators, embedded systems, and connectivity technologies.
CO2: Gain proficiency in using various IoT hardware platforms, such as microcontrollers, development boards, and sensor modules, for prototyping and experimentation.
CO3: Learn about IoT communication protocols and standards, including Wi-Fi, Bluetooth Low Energy (BLE), Zigbee, LoRaWAN, and MQTT, and understand their advantages and limitations.
CO4: Understand the principles of IoT security and privacy, including authentication, encryption, access control, and secure firmware updates, and learn techniques for securing IoT devices and networks.

Module 1:

Overview of IoT
Introduction to the Internet of Things, IoT system: architecture and standards, Networking Basics - TCP/IP, Networking Basics - IP addressing basics (IPV4 and IPV6), Optimizing IP for IoT: From 6LoWPAN to 6Lo, Routing over Low Power and Lossy Networks.

Module 2:

IoT hardware Platforms:
IoT Design Methodology – Embedded computing logic – Microcontroller-System on Chips. Hardware platforms for prototyping IoT node- Arduino, Raspberry Pi, Node MCU, ESP32, ARM Cortex Microcontrollers, IoT mote hardware platforms, Digital India RISC- V based solutions. Interfacing sensors and actuators with hardware platforms. Developing IoT applications using Raspberry Pi with Python Programming.

Module 3:

IoT connectivity & Protocols:
IoT Access Technologies: WiFi, Zigbee, Zwave, Bluetooth,UWB, sub1GHz, LoRaWAN, Sigfox and NB-IoT. Topology and Security of IEEE 802.15.4, 802.15.4g, 802.15.4e, 1901.2a, 802.11ah and LoRaWAN. IoT application-level protocols: MQTT, CoAP, XMPP, HTTP/Rest Services, Web Sockets.

Module 4:

Data analytics, Cloud and IoT Security:
No SQL Databases Vs SQL Databases. Apache web servers, JSON, Open and commercial Cloud solutions for IoT, Python Web Application Frameworks for IoT, IoT data visualization tools, IoT Security - Need for encryption, standard encryption protocol, lightweight cryptography, Trust models for IoT, ARM Cortex Microcontroller Security, Root Security Services (RSS).

Module 5:

IoT Case studies:
Smart Lighting, Smart home, Smart Agriculture, Smart farming, IoT for health care & patient monitoring, Smart and Connected Cities, Building end-to-end smart applications with TinyML.

Labs / Practicals:

1. Design and implement a basic IoT system architecture using TCP/IP networking principles. Set up communication between IoT devices and a central server using TCP/IP protocols.

2. Explore different IoT hardware platforms such as Arduino, Raspberry Pi, and Node MCU. Develop a prototype IoT node using Arduino and interface it with sensors and actuators. Write and deploy a Python script on Raspberry Pi to create a basic IoT application.
3. Compare various IoT access technologies including WiFi, Zigbee, and Bluetooth. Set up a small IoT network using Zigbee or Bluetooth modules and analyze their performance and range. Implement MQTT or CoAP protocols for communication between IoT devices and a central server.
4. Install and configure a NoSQL database (e.g., MongoDB) and compare it with a traditional SQL database (e.g., MySQL). Set up Apache web servers for hosting IoT applications and visualize IoT data using Python-based data visualization tools like Matplotlib or Plotly.
5. Explore different encryption techniques and protocols for securing IoT communication. Implement lightweight cryptography algorithms on ARM Cortex microcontrollers and analyze their performance. Set up root security services (RSS) for secure booting and firmware updates on IoT devices.
6. Design and implement a smart lighting system using IoT hardware platforms and connectivity technologies. Develop a mobile application to control the lighting system remotely and optimize energy consumption based on sensor data.
7. Explore the concept of TinyML (machine learning on microcontrollers) and its applications in IoT. Develop a simple machine learning model using TensorFlow Lite for microcontrollers and deploy it on an IoT device for real-time inference tasks.
8. Integration of IoT Devices with Cloud Platforms (AWS IoT, Azure IoT, or Google Cloud IoT)
9. Edge AI Inference Using Pre-trained Models on IoT Devices
10. Real-Time IoT Data Logging and Alert System

References:

Text Books:
1. David Hanes, Gonzalo Salgueiro, Patrick Grossetete, Rob Barton and Jerome Henry, —IoT Fundamentals: Networking Technologies, Protocols and Use Cases for Internet of Things, Cisco Press, 2017
2. Alessandro Bassi, Martin Bauer, Martin Fiedler, Thorsten Kramp, Rob van Kranenburg, Sebastian Lange, Stefan Meissner, “Enabling things to talk – Designing IoT solutions with the IoT Architecture Reference Model”, Springer Open, 2016
3.Vijay Madisetti , Arshdeep Bahga, Adrian McEwen (Author), Hakim Cassimally “Internet of Things: A Hands-on-Approach” Arshdeep Bahga & Vijay Madisetti, 2014.
4. Gian Marco Iodice, TinyML Cookbook: Combine artificial intelligence and ultra-low-power embedded devices to make the world smarter
5. Olivier Hersent, David Boswarthick, Omar Elloumi , —The Internet of Things – Key applications and Protocols, Wiley, 2012

Course Code	Course Name	L	T	P	Cr
DOEE250115	Hardware Design Verification	3	0	2	4

Course Outcomes:

CO1: Acquire knowledge of advanced verification techniques, such as constrained random testing, coverage-driven verification, and assertion-based verification.
CO2: Gain proficiency in using industry-standard verification tools and methodologies, including simulation-based verification using hardware description languages (HDLs) such as Verilog and VHDL.
CO3: Develop skills in creating comprehensive verification plans and testbenches for digital hardware designs, covering functional verification, timing verification, and design constraints.

Module 1:

Overview of hardware design verification:
Design Verification using Verilog HDL-types of test benches, writing test cases. Verification Architecture, Test automation. Random verification. Assertions and Coverage. Verification of a decade counter, memory and FIFO.

Module 2:

Hardware Design Verification Language-System Verilog:
Introduction to C/C++, Object orient programming. Overview of Verification Language-System Verilog. System Verilog test benches. Functional verification.

Module 3:

System Verilog Advanced features:
Coverage driven Verification-System Verilog functional coverage. Constraint Random Verification. Assertion based Verification. IP Verification a case study.

Module 4:

SoC Verification:
Overview of SoCs. SoC Verification architecture. Verification planning and phases of verification. Building SoC Verification environment. Writing test cases and test automation. SoC Verification a case study.

Module 5:

Methodology based Verification:
Introduction to verification methodologies. Universal Verification Methodology (UVM), UVM components. Building UVM test bench/verification environment. UVM based IP verification a case study.

Labs / Practicals:

1. Perform directed Verification of a 1024 x 8 SRAM. Use tasks to obtain good code density.
2. Develop Verification plan and Verify 16x8 FIFO. Perform test automation by scripting. Write FIFO assertions.
3. Design a ones counter with the following behavior.

Ones Counter is a Counter which counts the number of one's coming in serial stream. The Minimum value of the count is "0" and count starts by incriminating one till "15". After "15" the counter rolls back to "0". Reset is also provided to reset the counter value to "0". Reset signal is active negedge. Input is 1 bit port for which the serial stream enters. Out bit is 4 bit port from where the count values can be taken. Reset and clock pins also provided.

a) Develop RTL code for ones counter
b) Develop SV Based Verification plan, which includes coverage and assertion plan.
c) Develop interface to interface DUT with Verification Environment.

d) Develop test cases and perform Verification.
4. Develop SV Based verification plan to verify 128 x 8 fifo.
a) Execute the verification plan and complete the verification.
b) Generate Code Coverage Reports and check the percentage of code coverage.
c) Write SV Assertions to improve verification
5. Perform UVM based Verification of 8-bit ALU
6. Analysis and simulation of a SoC Verification Environment.
7. UVM-Based Verification of a UART (Universal Asynchronous Receiver/Transmitter)
8. Formal Verification using SystemVerilog Assertions (SVA)
9. Interrupt Controller Verification
10. Verification of AXI4-Lite Protocol using UVM

References:

1. Spear, C. and Tumbush, G. SystemVerilog for Verification: A Guide to Learning the Testbench Language Features,3 rd Edition, Springer, 2012.
2. Design Verification with 'e': By Samir Palnitkar.
3. Hardware Design Verification: Simulation and Formal Method-Based Approaches: By William K. Lam, Prentice Hall
4. Writing Testbenches: Functional Verification of HDL Models: By Janick Bergeron, Springer

Course Code	Course Name	L	T	P	Cr
DOEE250185	System Design Using Embedded Processors	3	0	2	4

Course Outcomes:

CO 1 Comprehend Embedded Processor and its software
CO 2 Design an Embedded system with ARM microcontrollers
CO 3 Design an Embedded system using processors, memory I/O devices and communication network within realistic constraints
CO 4 Comprehend ARM Cortex M4 microcontrollers
CO 5 Comprehend the peripheral programming of ARM cortex M4 Microcontrollers
CO 6 Comprehend advanced Embedded Controllers, Features and case studies
CO 7 Perform ARM Cortex M4 Microcontroller configuration for various applications using peripheral devices
CO 8 Demonstrate an embedded system on ARM Cortex M4 Microcontroller development board

Module 1:

Introduction to Embedded Systems
Overview of embedded system architecture, Development and debugging Tools for Embedded Systems, Overview ARM Architecture - Architecture Versions, Instruction Set Development,Thumb-2 and Instruction Set Architecture, AMBA, AXI bus overview. Overview of the ARM Cortex-Mx Processor Architectures.

Module 2:

ARM Cortex M4 Microcontroller System
ARM Cortex M4 Core, Interconnect Matrix in ARM Cortex M4 Microcontroller, System configuration Controller, NVIC, External Interrupt Controllers, DMA, Reset and Clock Control, Clock Recovery System, Power Control.

Module 3:

ARM Cortex M4 Microcontroller Peripheral Overview
Introduction to ARM Cortex-M4 Programming, overview of CMSIS, GPIO, ADC, DAC, Communication & Peripherals - USART, UART, I2C, SPI, USB, CAN
Watchdogs and Timers, PWM.

Module 4:

Memory, Safety and Security in ARM Cortex Microcontroller
Flash, Quad SPI Interface, Flexible Memory controller, CRC, Random Number Generator, memory protections, Advanced Encryption Standard HW Accelerator (AES), Safety support.

Module 5:

Advanced Embedded Controllers, Features and case studies
Programming for Power-Efficient Computing - High Level and low level Techniques, Cortex M7, M23 and M33 Controllers and Features, Overview of mbed platform, Embedded Systems case studies - Consumer, Medical, Automotive.

Labs / Practicals:

Embedded C Programming on ARM Cortex M4 Microcontroller with CMSIS/HAL libraries, Embedded C Programming - Peripheral Programming - GPIO, ADC, DAC, USART, Timers and PWM, Design of a real-time data acquisition & control system using the ARM Cortex M4 Microcontroller

References:

1. Yiu J. The Definitive Guide to ARM Cortex M3 and Cortex M4 Processors, 3rd Edition, Elsevier
2. Andrew N Sloss, Dominic Symes, Chris Wright, “ARM System Developer's Guide - Designing and Optimizing System Software”, 2006, Elsevier

3. Steve Furber, “ARM System-on-Chip Architecture”, 2nd Edition, Pearson Education
4. Raj Kamal, “Microcontroller - Architecture Programming Interfacing and System Design” 1st Edition, Pearson Education
5. P.S Manoharan, P.S. Kannan, “Microcontroller based System Design”, 1st Edition, Scitech Publications
6. Cortex-M4 Technical Reference Manual (TRM)
7. STMicroelectronics Nucleo-G4xx development board user manual

Course Code	Course Name	L	T	P	Cr
DOEE250201	VLSI Signal Processing	3	0	2	4

Course Outcomes:

CO1: Understand the fundamentals of signal processing algorithms and their mathematical representations, including discrete-time signals and systems, linear time-invariant (LTI) systems, and Fourier analysis.
CO2: Gain proficiency in using hardware description languages (HDLs) such as Verilog and VHDL for modeling, simulating, and synthesizing digital signal processing (DSP) algorithms for VLSI implementation.
CO3: Develop critical thinking and problem-solving skills through analysis of existing VLSI signal processing architectures, identifying design trade-offs, and proposing innovative solutions to improve performance and efficiency.

Module 1:

Introduction to VLSI Signal processing:
Graphical representation of DSP algorithms. Dataflow and control flow. Parallel pipelined design of DSP Algorithms. Retiming.

Module 2:

Unfolding and Folding:
Properties of unfolding, unfolding and retiming, Folding Transformation, Register Minimization Techniques.

Module 3:

VLSI Architectures for Digital Signal Processing:
Architectural Design at Register Transfer Level, Design of data path and control path units, Verilog RTL implementation.

Module 4:

VLSI implementation of Filter and Transform structures:
FIR and IIR architectures. Serial and parallel implementation of DIT-FFT algorithm.

Module 5:

VLSI array signal processing:
Overview of spatial filters. VLSI implementation of a delay and sum (DAS) spatial filter.

Labs / Practicals:

1. Fold 4-bit parallel adder using 1 bit adder. Design control circuitry for proper scheduling of input and output data. Verify the operation with proper scheduling.
2. Perform the Verilog RTL implementation for a radix-2 FFT butterfly structure.
3. Given the following DSP System y(n) = a*x(n) + b*y(n-1), with multiplier delay 9ns and adder delay 7nsec. Develop RTL description of the system using Verilog HDL Coding. Perform timing analysis.
4. Implement a 3-tap FIR filter using Verilog HDL and demonstrate the pipe-line operation.
5. Perform Verilog RTL implementation of a 6 tap FIR data broad cast structure.
6. Design and simulate an 8-channel delay and sum spatial filter.
7. Verilog RTL Implementation of a 16-point Radix-2 Decimation-in-Time FFT
8. Design and Verification of a Direct Form II IIR Filter
9. Implementation of a Systolic Array for Matrix Multiplication
10. Design and Simulation of a CORDIC Processor for Sine and Cosine Computation

References:

1. VLSI Signal Processing Systems - Keshab K Parhi, John Wiley and Son's, NY 1999.
2. Architectures for Digital Signal Processing - Peter Prissch, John Wiley and Son's NY 1998.
In addition, manufacturers Device data sheets and application notes are to be referred to get practical and application-oriented information.

Course Code	Course Name	L	T	P	Cr
DOAI250001	Advanced Data Analytics	3	0	2	4

Course Outcomes:

CO1: Understand fundamental concepts and methodologies in data analytics, including data preprocessing, exploratory data analysis (EDA), dimensionality reduction, and predictive modeling.

CO2: Apply machine learning algorithms for supervised, unsupervised, and reinforcement learning tasks in the context of data analytics.

CO3: Demonstrate proficiency in programming languages and tools commonly used in data analytics, including Python, R, and associated libraries.

CO4: Analyze large-scale datasets using distributed big data frameworks, including Apache Hadoop and Apache Spark.

CO5: Interpret results through appropriate visualization techniques and apply insights in practical domains such as social media analytics, fraud detection, and recommender systems.

Module 1:

Module 1: Introduction to Data Analysis

Introduction to Data Analysis: Concepts, Analytics Process Models, Analytical Model Requirements, Data Analytics Lifecycle Overview, Data Collection Methods, Sampling Techniques, Data Preprocessing, Handling Missing Data, Dimensionality Reduction Techniques (PCA, t-SNE, UMAP), Introduction to Feature Engineering.

Module 2:

Module 2: Predictive and Descriptive Analytics

Predictive Analytics: Linear Regression, Decision Trees, Neural Networks, Dimensionality Reduction through PCA.
Descriptive Analytics: Association Rule Mining (Apriori, FP-Growth), Market Basket Analysis, Sequential Pattern Mining, Clustering Techniques (K-Means, Hierarchical Clustering, DBSCAN).
Introduction to Reinforcement Learning Concepts for Analytics.

Module 3:

Big Data Framework and Analytics

Overview of Big Data Technologies, Challenges in Big Data Acquisition, Management, and Processing.
Big Data Architectures and Frameworks: Hadoop Ecosystem, HDFS, YARN.
Spark Ecosystem: RDDs, DataFrames, Spark SQL.
NoSQL Databases (MongoDB, Cassandra): Data Models and Applications.
Big Data Trends: Streaming, IoT Data, Real-Time Analytics.

Module 4:

Data Analysis Using R

Introduction to R: Data Import, Data Cleaning, Data Exploration, Visualization with ggplot2.
Descriptive Statistics in R, Statistical Methods for Evaluation, Exploratory Data Analysis (EDA), Handling Dirty Data, Visualization Techniques for Single and Multiple Variables, Data Presentation versus Data Exploration.

Module 5:

Big Data Techniques and Applications

MapReduce Paradigm, Hadoop System and Data Processing, Advanced Analytics with Spark Streaming, Real-Time Processing Use Cases.
Applications and Case Studies: Social Media Analytics, Recommender Systems, Fraud Detection—Introduction to Time Series Analytics: ARIMA Models.

Labs / Practicals:

1. Implement Principal Component Analysis (PCA) for dimensionality reduction and visualization.

2. Build a Linear Regression model for predictive analytics (e.g., housing price prediction).

3. Develop Decision Tree models for classification problems (Iris dataset or similar).

4. Apply Association Rule Mining (Apriori) on transactional data.

5. Implement K-Means Clustering for unsupervised grouping and visualize clusters.

6. Setup and operate Hadoop for data processing using MapReduce.

7. Data analysis and visualization in R with real datasets (EDA, plotting).

8. Perform exploratory data analysis (EDA) using R for multivariate datasets.

9. Analyze social media data using Apache Spark and perform sentiment analysis.

10. Implement Time Series Forecasting using ARIMA.

References:

1. Bart Baesens, Analytics in a Big Data World: The Essential Guide to Data Science and its Business Intelligence and Analytic Trends, John Wiley & Sons, 2014.

2. David Dietrich, Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data, John Wiley & Sons, 2015.

3. Jiawei Han, Micheline Kamber, Jian Pei, Data Mining: Concepts and Techniques, 3rd Edition, Morgan Kaufmann / Elsevier, 2011.

4. Margaret H. Dunham, Data Mining: Introductory and Advanced Topics, Pearson, 2011.

5. Wes McKinney, Python for Data Analysis, 2nd Edition, O'Reilly Media, 2018.

6. Hadley Wickham, Garrett Grolemund, R for Data Science, O'Reilly Media, 2017.

7. Official Documentation:

Apache Hadoop: https://hadoop.apache.org/

Apache Spark: https://spark.apache.org/docs/latest/

R Language: https://cran.r-project.org/

Course Code	Course Name	L	T	P	Cr
DOEE250016	Analog Mixed Signal VLSI Design	3	0	2	4

Course Outcomes:

CO1: Understand the fundamentals of analog and mixed-signal VLSI design, including basic circuit topologies, transistor-level design, and layout principles.
CO2: Develop the ability to perform transistor-level design and optimization of analog building blocks such as amplifiers, oscillators, filters, and data converters.
CO3: Acquire skills in designing and simulating mixed-signal systems, including analog-to-digital converters (ADCs), digital-to-analog converters (DACs), and interface circuits.

Module 1:

INTRODUCTION AND BASIC MOS DEVICES:
Challenges in analog and mixed mode design- MOS FET structures and characteristics-large signal model –small signal model-single stage Amplifier Source follower Common gate stage –Cascode Stage.

Module 2:

SUBMICRON CIRCUIT DESIGN:
Submicron CMOS process flow, Capacitors and resistors, Current mirrors, Digital Circuit Design, Delay Elements – Adders-OP Amp parameters and Design. Simulation and ac, dc analysis. Components available on an IC & Layout consideration, Noise in resistors and MOS transistors. Analog and mixed mode layout design considerations and lay-out design of an Op_amp

Module 3:

DATA CONVERTERS:
Characteristics of Sample and Hold -Digital to Analog Converters- 371 architecture-Differential Non linearity-Integral Non linearity-Voltage Scaling-Cyclic DAC-Pipeline DAC-Analog to Digital Converters-architecture –Flash ADC Pipeline ADC-Differential Non linearity-Integral Non linearity.

Module 4:

SNR IN DATA CONVERTERS:
Overview of SNR of Data Converters-Clock Jitters-Improving Using Averaging –Decimating Filters for ADC-Band pass and High Pass Sinc Filters-Interpolating Filters for DAC.

Module 5:

SWITCHED CAPACITOR CIRCUITS:
Resistors, first order low pass Circuit, Switched capacitor Amplifier, Switched Capacitor Integrator.

Labs / Practicals:

1. AN INVERTER : Schematic Entry and Symbol Creation Building the Inverter Test Design , Simulation with Spectre, Parametric Analysis ,Creating Layout View of Inverter ,Physical Verification ,Creating the Configuration View, Generating Stream Data
2. NAND DESIGN : Schematic Entry and Symbol Creation Building the NAND Test Design , Simulation with Spectre, Parametric Analysis ,Creating Layout View of NAND Gate ,Physical Verification ,Creating the Configuration View,Generating Stream Data
3. SRAM DESIGN: Schematic Entry and Symbol Creation Building the SRAM Test Design, Simulation with

Spectre, Creating Layout View of SRAM, Physical Verification.
4. COMMON SOURCE AMPLIFIER: Schematic Entry and Symbol Creation Building the COMMON SOURCE AMPILFIER Test Design, Analog Simulation with Spectre.
5. DIFFERENTIAL AMPLIFIER DESIGN: Schematic Entry and Symbol Creation Building the DIFFERENTIAL AMPLIFIER DESIGN Test Design, Analog Simulation with Spectre.
6. BASIC OP AMP DESIGN: Schematic Entry and Symbol Creation Building the OP AMP Test Design, Analog Simulation with Spectre.
7. ADC Design: Schematic Entry and Symbol Creation Building the ADC Test Design, Simulation with Spectre.
8. DAC (Digital-to-Analog Converter) Design
9. Bandgap Reference Circuit Design
10. Current Mirror Design (Basic and Cascode)

References:

1. VineethaP.Gejji Analog and Mixed Mode Design Prentice Hall, 1st Edition , 2011
2. JeyaGowri Analog and Mixed Mode Design Sapna publishing House 2011.
3. Gray Paul R, Meyer, Robert G, Analysis and Design of Analog Integrated Circuits, 3rd edition, John Wiley & Sons.
4. Jacob Baker, "CMOS Mixed-Signal circuit design", A John Willy & Sons, inc., publications, 2003.
5. Professor Bernhard Boser – “Analysis and Design of VLSI Analog-Digital Interface Integrated Circuits” “Addison Wisely publications” (1991).