JNTU-GV

COURSE STRUCTURE AND SYLLABUS

M.Tech

in

VLSI & EMBEDDED SYSTEMS (Course Code:68)

And

EMBEDDED SYSTEMS (Course Code:55)

I Year-I Semester

S.NO	Course Code	**Course Title**	**L**	**T**	**P**	**Credits**
1	M255501	CMOS Digital Integrated Circuit Design	3	1	0	4
2	M255502	CMOS Analog Integrated Circuit Design	3	1	0	4
3	M255503	Embedded System Design	3	1	0	4
4	M255504	Scripting Languages	3	1	0	4
	M255505	Soc design				
	M255506	VLSI architectures				
	M255507	Processor Verification				
5	M255508	1. Embedded C	3	1	0	4
	M255509	2. Hardware Software Co-Design				
	M255510	Advanced Computer architecture				
	M255511	IOT				
6	M255512	Analog CMOS Circuit Design Lab	0	1	2	2
7	M255513	Embedded System Design Lab	0	1	2	2
8	M255514	Seminar-I	0	0	2	1
		Total	**15**	**7**	**6**	**25**

I Year II – Semester

S.No	**Course Code**	**Course Title**	**L**	**T**	**P**	**Credits**
1	N255501	CMOS Mixed signal design	3	1	0	4
2	N255502	RTOS	3	1	0	4
3	N255503	Advanced Processors And Controllers	3	1	0	4
4	N255504	Design for Testability	3	1	0	4
	N255505	Physical design and verification				
	N255505	System Verilog & UVM				
	N255507	Low power VLSI Design				
5	N255508	Network security and cryptography	3	1	0	4
	N255509	Embedded Networking				
	N255510	Data Acquisition Systems				
	N255511	Linux Systems with OOPS				
6	N255512	Digital CMOS Circuit Design Lab	0	1	2	2
7	N255513	RTOS Lab	0	1	2	2
8	N255514	Seminar-II	0	0	2	1
		Total	**15**	**7**	**6**	**25**

II YEAR I – SEMESTER

Sl.No.	Course Code	Course Title	L	T	P	C
1	0255501	Research Methodology and IPR / Swayam 12-week MOOC course – RM&IPR	3	0	0	2
2	0255501	Summer Internship/ Industrial Training		-	-	2
3	0255501	Dissertation Part – A	-	-	20	10
		TOTAL	**3**	**-**	**20**	**14**

II YEAR II – SEMESTER

Sl.No.	Course Code	Course Title	L	T	P	C
1	P255501	Dissertation Part – B	0	0	32	16
		Total	**0**	**0**	32	**16**

CMOS DIGITAL INTEGRATED CIRCUIT DESIGN

Course overview:

This course provides a thorough understanding of CMOS VLSI design, covering MOS transistors, logic and sequential circuits, arithmetic units, and memory architectures. It emphasizes both static and dynamic behavior, layout strategies, and performance trade-offs in digital integrated circuit design.

Course Objective:

- To understand MOSFET operation under static and dynamic conditions for digital design.
- To analyze CMOS inverter characteristics including power, delay, and energy efficiency.
- To design combinational & sequential logic using various CMOS logic styles.
- To explore arithmetic building blocks focusing on speed and area trade-offs.
- To learn memory architectures including SRAM, ROM, and dynamic memory design techniques.

Course Outcomes:

- Analyze and design MOSFET-based circuits, focusing on static, dynamic, and power characteristics.
- Design combinational logic circuits using static CMOS and dynamic logic with optimization techniques.
- Implement and analyze sequential circuits, including latches, registers, pipelines, and timing considerations.
- Design efficient arithmetic building blocks and memory architectures, focusing on speed and area tradeoffs.

UNIT- I

MOS Transistor Principles & CMOS Inverter: MOSFET characteristics under Static and Dynamic Conditions, MOS Transistor Secondary Effects, CMOS Inverter, Static and Dynamic Characteristics, Power, Energy, Energy Delay Parameters, Stick Diagram & Layout Diagrams.

UNIT-II

Combinational Logic Circuits: Static CMOS Design, Different Styles of Logic Circuits, Logical Effort of Complex Gates, Static and Dynamic Properties of Complex Gates, Interconnect Delay, Dynamic Logic Gates.

UNIT-III

Sequential Logic Circuits: Static Latches and Registers, Dynamic Latches and Registers, Timing Issues, Pipelines, Non-Bistable Sequential Circuits.

UNIT-IV

Arithmetic Building Blocks: Data Path Circuits, Architectures for Adders, Accumulators, Multipliers, Barrel Shifters, Speed and Area Tradeoffs.

UNIT-V

Memory Architectures: Memory Architectures and Memory Control Circuits: Read-Only Memories, ROM Cells, Read-Write Memories (RAM), Dynamic Memory Design, 6- Transistor SRAM Cell, Sense Amplifiers.

Text Books:

1. Jan Rabaey, Anantha Chandrakasan, BNikolic, "Digital Integrated Circuits: A Design Perspective", Prentice Hall of India, 2nd Edition, Feb 2003
2. .N. Weste, K. Eshraghian, "Principles of CMOS VLSI Design",Addision Wesley,2nd Edition, 1993.

Reference Books:

1. MJSmith ,"Application Specific Integrated Circuits",Addisson Wesley,1997
2. **Sung-Mo Kang, Yusuf Leblebici, Chulwoo Kim**, CMOS Digital Integrated Circuits: Analysis and Design, McGraw-Hill Education, **Fourth Edition, 2014 (International) / 2019.**

e-Resources:

1.https://onlinecourses.nptel.ac.in/noc20_ee05/preview

CMOS ANALOG IC DESIGN

Course Overview: This course provides a detailed understanding of MOS devices, CMOS analog sub-circuits, and amplifier design. Topics include MOS transistor modeling, current mirrors, and amplifiers, with a focus on noise analysis and CMOS operational amplifiers. Students will learn about design techniques, compensation methods, and measurement techniques for building efficient and reliable analog circuits.

Course Objective:

- To understand the operation and modeling of MOS transistors in CMOS circuits.
- To design and analyze analog CMOS sub-circuits, such as current mirrors and voltage references.
- To explore single-stage and multi-stage amplifier designs, including gain boosting and cascode techniques.
- To study noise behavior in CMOS amplifiers and mitigate its effects.

Course Outcomes:

- Design basic building blocks of CMOS Analog ICs.
- Carry out the design of single and two stage operational amplifiers and voltage references.
- Determine the device dimensions of each MOSFETs involved.
- Design various amplifiers like differential, current and operational amplifiers.

UNIT- I

MOS Devices and Modeling: The MOS Transistor, Passive Components - Capacitor & Resistor, Integrated Circuit Layout, CMOS Device Modeling - Simple MOS Large-Signal Model, Other Model Parameters, Small-Signal Model for the MOS Transistor, Computer Simulation Models, Sub-threshold MOS Model.

UNIT-II

Analog CMOS Sub-Circuits: MOS Switch, MOS Diode, MOS Active Resistor, Current Sinks and Sources, Current Mirrors - Current Mirror with Beta Helper, Degeneration, Cascode Current Mirror and Wilson Current Mirror, Current and Voltage References, Bandgap Reference.

UNIT- III

Single Stage Amplifier: Common Source Stage with Resistive Load, Diode Connected Load, Triode Load, CS Stage with Source Degeneration, Source Follower, CG Stage, Gain Boosting Techniques, Cascode, Folded Cascode, Choice of Device Models.

UNIT- IV

CMOS Amplifiers and Noise: Inverters, Differential Amplifiers, Cascode Amplifiers. Noise - Statistical Characteristics, Types, Noise in Single-Stage Amplifiers, Noise in Differential Pairs, Noise Bandwidth.

UNIT- V

CMOS Operational Amplifiers: Design of CMOS Op Amps, Compensation of Op Amps, Design of Two-Stage Op Amps, Power Supply Rejection Ratio of Two-Stage Op Amps, Cascode Op Amps, Measurement Techniques of Op Amps.

Text Books:

1. Philip E. Allen and Douglas R. Holberg, "*CMOS Analog Circuit Design*", Oxford University Press, Third Edition, 2013.
2. Behzad Razavi, "*Design of Analog CMOS Integrated Circuits*", McGraw-Hill Education, Second Edition, 2016.

Reference Books:

1. David A. Johns and Ken Martin, "*Analog Integrated Circuit Design*", Wiley Student Edition, 2013.
2. Paul R. Gray, Paul J. Hurst, Stephen H. Lewis, and Robert G. Meyer, "*Analysis and Design of Analog Integrated Circuits*", Wiley, Sixth Edition, 2024.

e-Reference:

NPTEL: https://www.youtube.com/playlist?list=PLdlPA9pGVVtZ7abJ1MAJc71S2ta OnGux

EMBEDDED SYSTEM DESIGN

Course Overview:
This course provides an in-depth understanding of embedded systems with emphasis on hardware and software co-design. It introduces the ARM architecture, embedded firmware, and programming with Raspberry Pi for real-world interfacing applications. Students will gain knowledge of processor architectures, instruction sets, interfacing techniques, and design methodologies required for building efficient embedded solutions.

Course Objective:
- To introduce the principles and characteristics of embedded systems.
- To understand the architecture of ARM processors and their programming model.
- To provide insights into embedded firmware and its design methodologies.
- To enable students to design embedded applications using Raspberry Pi and Python.
- To develop skills in interfacing sensors and actuators with embedded platforms.

Course Outcomes:
- Understand fundamentals and characteristics of embedded system.
- Analyze ARM architecture, instruction sets, and operating modes.
- Design and develop embedded firmware and interfacing circuits.
- Apply Raspberry Pi for embedded programming and sensor/actuator interfacing.

UNIT- I
Introduction to Embedded Systems
Definition of Embedded System, Embedded Systems Vs General Computing Systems, History of Embedded Systems, Classification, Major Application Areas, Purpose of Embedded Systems, Characteristics and Quality Attributes of Embedded Systems.
UNIT-II
Typical Embedded System
Core of the Embedded System: General Purpose and Domain Specific Processors, ASICs, PLDs, Commercial Off-The-Shelf Components (COTS).Memory: ROM, RAM, Memory according to the type of Interface, Memory Shadowing, Memory selection for Embedded Systems. Sensors and Actuators, Communication Interfaces: Onboard and External
Interfaces.
UNIT-III
EmbeddedFirmware
Reset Circuit, Brown-out Protection Circuit, Oscillator Unit, Real Time Clock, Watchdog Timer, Embedded Firmware Design Approaches and Development Languages.
UNIT-IV
ARM Architecture and Instruction Sets
ARM design philosophy, data flow model and core architecture, registers, program status

register, instruction pipeline, interrupts and vector table, operating modes and ARM processor families. Instruction Sets: Data processing instructions, addressing modes, branch,load, store instructions, PSR instructions, conditional instructions.

UNIT-V

Raspberry Pi and Interfacing

Raspberry Pi board and its processor, Programming the Raspberry Pi using Python, Communication facilities on Raspberry Pi (I^2C, SPI, UART), Interfacing of sensors and actuators.

Text Books:

1. Jan Rabaey, Anantha Chandrakasan, BNikolic, "Digital Integrated Circuits: A Design Perspective", Prentice Hall of India, 2nd Edition, Feb 2003
2. N. Weste, K. Eshraghian, "Principles of CMOS VLSI Design",Addision Wesley,2nd Edition, 1993.

Reference Books:

1. MJSmith ,"Application Specific Integrated Circuits",Addisson Wesley,1997.
2. Sung-Mo Kang, Yusuf Leblebici, "*CMOS Digital Integrated Circuits: Analysis and Design"*, 4th Edition, McGraw-Hill, 2013.

e Resources:

https://onlinecourses.nptel.ac.in/noc25_cs30/preview

SCRIPTING LANGUAGES

Course Overview:

This course introduces scripting languages with applications in VLSI design and CAD automation. It covers PERL, TCL, and PYTHON for data handling, design automation, and tool integration. Students will gain hands-on experience in automating workflows and developing efficient EDA solutions. The course also prepares learners to apply scripting in system development and web applications.

Course Objective:

- To introduce the fundamentals of scripting languages and their role in VLSI design automation.
- To describe PERL concepts for handling large data sets in VLSI/CAD applications.
- To utilize TCL for CAD tool interfacing and automation.
- To interpret the PYTHON language and its role in design, testing, and application development.

Course Outcomes:

- Gain fluency in programming with scripting languages (PERL, TCL, PYTHON).
- Create and execute scripts in PERL/TCL/PYTHON for CAD tool automation and data handling.
- Demonstrate scripting in PERL/TCL/PYTHON for system-level and web applications.
- Analyze security issues, advanced data structures, and tool interfacing using scripting languages.

UNIT- I

Introduction to Scripts and Scripting

Basics of Linux ,Origin of scripting languages , Characteristics, Uses of scripting languages, Applications in VLSI and CAD tools.

Unit-II:

PERL Basics

Introduction to PERL, Names and values, Variables and assignments, Scalar expressions ,Control structures ,Built-in functions, Arrays, Lists, and Hashes ,Simple Input/Output ,Strings ,Regular Expressions , Subroutines , Command-line arguments.

Unit-III:

Advanced PERL

Looping structures ,Advanced Subroutines, Pack and Unpack ,File handling ,Type globs, Eval and References ,Data structures ,Packages, Libraries, and Modules ,Object-Oriented PERL , Tied Variables ,OS Interfacing ,Security Issues.

Unit-IV:

TCL Programming

TCL phenomena ,Philosophy and structure , Syntax and Parser, Variables and data types , Control flow , Data structures , Input/Output ,Procedures , String handling , Patterns , File and Pipe handling , Example applications in EDA.

Unit-V:

PYTHON Programming

Introduction to PYTHON ,Syntax and Statements , Functions ,Built-in Functions and Methods, Modules ,Exception Handling ,Applications of PYTHON in VLSI, system development, and web applications.

Text Books:

1. David Barron, *The World of Scripting Languages*, Wiley Student Edition, 2010.
2. Steve Holden, David Beazley, *PYTHON Web Programming*, New Riders Publications, 2012.

Reference Books:

1. Clif Flynt, *TCL/TK: A Developer's Guide*, Morgan Kaufmann, 2003.
2. Chun, *Core PYTHON Programming*, Pearson Education, 2006.
3. Randal L. Schwartz, *Learning PERL*, O'Reilly Publications, 6th Edition, 2011.
4. Richard Peterson, *Linux: The Complete Reference*, McGraw Hill, 6th Edition, 2008.

e Resources:

https://www.classcentral.com/course/youtube-electronics-linux-programming- scripting-47539

SYSTEM ON CHIP

Course Overview:

This course focuses on System-on-Chip (SoC) design methodologies with hardware-software co-design. It covers processor and memory architecture, bus interconnections, and customization techniques. Students will study design trade-offs involving performance, power, and area in SoC architectures. Real-world case studies like AES encryption and JPEG compression are analyzed for practical insights.

Course Objective:

- To introduce system-level approaches and architectures for SoC design.
- To provide knowledge of processor selection, pipeline optimization, and advanced processor architectures.
- To discuss SoC memory hierarchy, cache design, and processor–memory interactions.
- To study interconnect architectures, SoC standard buses, and reconfiguration techniques, analyze case studies and applications of SoC in cryptography and image processing

Course Outcomes:

- Understand the fundamentals of system architecture, processor architectures, and SoC design approach.
- Analyze processor architectures, pipeline delays, superscalar and VLIW processors.
- Design memory systems including cache hierarchies, scratchpads, and processor- memory interaction models.
- Evaluate interconnect architectures, customization techniques, and reconfiguration trade-offs.

UNIT- I

Introduction to the System Approach

System Architecture – Components of a System, Hardware and Software, Processor Architectures, Memory and Addressing, System-Level Interconnection, SoC Design Approaches, System Complexity.

UNIT-II:

Processor Selection for SoC – Basic Concepts of Processor Architecture & Micro-Architecture, Instruction Handling, Buffers & Pipeline Delay Minimization, Branch Prediction, Vector Processors and Extensions, VLIW Processors, Superscalar Processors.

UNIT-III:

Memory Design in SoC

SoC External and Internal Memory ,Cache Organization & Policies ,Scratchpads ,Write Policies – Line Replacement Strategies ,Split I/D Caches ,Multilevel Caches ,Virtual to Real Translation, SoC Memory Models ,Processor, Memory Interaction Models.

UNIT-IV:

Interconnect Customization and Configuration

Interconnect Architectures – Bus Architectures ,SoC Standard Buses ,Analytic Bus Models, Bus Transaction Analysis ,Customizing Instruction Processors, Reconfiguration Technologies ,Mapping Designs to Reconfigurable Devices ,Instance-Specific Design, Customizable Soft Processors ,Trade-off Analysis of Reconfiguration Overheads.

UNIT-V:

Applications and Case Studies

SoC Design Approach in Applications – AES Algorithm Design and Evaluation – JPEG Compression for Image Processing – Emerging SoC Applications.

Text Books:

1. Michael J. Flynn, Wayne Luk, *Computer System Design: System-on-Chip*, Wiley India Pvt. Ltd.
2. Steve Furber, *ARM System on Chip Architecture*, 2nd Edition, Addison Wesley Professional, 2000.

Reference Books:

1. Ricardo Reis, *Design of System on a Chip: Devices and Components*, Springer, 2004.
2. Jason Andrews, *Co-Verification of Hardware and Software for ARM System on Chip Design*, Newnes, 2004.

e Resources:

1. https://youtu.be/PRQXzjTrCJY?si=Yb60zL9i7AQ7J24S
2. https://youtu.be/FUhCrWoNA2c?si=KLaTDyQy9ME6Kd5t

VLSI ARCHITECTURES

Course Overview:

This course explores programmable logic devices with a focus on FPGA/CPLD architectures and FSM design. It emphasizes architecture-centered design, system-level partitioning, and application-specific implementations. Students will gain knowledge of advanced methodologies like one-hot encoding and ASM-based design. The course also covers controller–datapath design for efficient digital system implementation.

Course Objective:

- To understand programmable logic devices (CPLD/FPGA) and their internal architectures.
- To analyze and compare CPLD/FPGA device families and performance.
- To study finite state machine (FSM) design methodologies and implementation strategies.
- To apply FSM architectures such as one-hot and ASM methods for digital design and develop system-level designs with controller, datapath, and functional partitioning.

Course Outcomes:

- Explain programmable logic device architectures (CPLD, FPGA) and their applications.
- Analyze FPGA/CPLD architectures and evaluate their speed and performance.
- Design finite state machines using PLDs and apply advanced techniques such as one-hot encoding and ASM charts.
- Develop system-level digital designs using controller–datapath partitioning and functional modules.

UNIT–I:

Programmable Logic Devices: Complex Programmable Logic Devices (CPLD): ROM, PLA, PAL, PLD, PGA – Features, programming and applications, Altera MAX 5000/7000 series, Altera FLEX 10000 series CPLD,AMD's CPLD (Mach 1–5), Cypress FLASH 370 device technology, Lattice LSI's 3000 series – Speed, performance, in-system programmability. Introduction to FPGAs: Logic blocks, routing architecture, design flow, technology mapping.

UNIT–II:

FPGA/CPLD Architectures

FPGA/CPLD device architectures: Xilinx XC4000 series, Altera FLEX 8000/10000 series, AT&T ORCA (Optimized Reconfigurable Cell Array),Actel ACT-1, ACT-2, ACT-3 families Comparison of architectures with respect to speed, performance, and area trade-offs.

UNIT–III:

Finite State Machines (FSM) Top-down FSM design, state transition tables, state assignment for FPGAs, Initial state assignment for one-hot encoding, Realization of state machine charts with PAL, Alternative realization using microprogramming, Linked state machines One-hot state machines, Petri nets for FSM design – concepts, properties, and parallel controllers, Meta-stability and synchronization issues, **Case Study:** FSM-based digital system design.

UNIT–IV:

FSM Architecture

Architectures centered around non-registered PLDs, FSM design using shift registers,One-hot design method and its applications, use of Algorithmic State Machines (ASM) in one- hot design Advantages and trade-offs of FSM-based architecture.

UNIT–V:

System-Level Design

System-level partitioning: Controller, datapath, and functional partitions, design of parallel adder cells and sequential circuits, counters, multiplexers, and system controllers, parallel controllers for high-performance designs, application-oriented system design using FPGAs/CPLDs.

eResources:

1.https://www.youtube.com/playlist?list=PLfMCiCIRnpUnFgNSy0QuOuqIIrG0fe5eD

EMBEDDED C

Course Overview:

This course provides an in-depth understanding of embedded systems, focusing on both hardware and software aspects of design. Students will learn about real-time operating systems (RTOS), hardware-software co-design, interfacing techniques, and design methodologies used in developing reliable and efficient embedded solutions for real-world applications.

Course Objective:

- To provide an overview of Design Principles of Embedded System.
- To provide clear understanding about the role of firmware.
- To understand the necessity of operating systems in correlation with hardware systems.
- To learn the methods of interfacing and synchronization for tasking.

Course Outcomes:

- Apply and analyze the applications in various processors and domains of embedded system.
- Analyze and develop embedded hardware and software development cycles and tools.
- Analyze to understand what a microcomputer, core of the embedded system.
- Analyze to understand different concepts of a RTOS, sensors, memory interface, communication interface.

UNIT- I

Programming Embedded Systems in C:

Introduction, What is an embedded system, Which processor should you use, Which programming language should you use, Which operating system should you use, How do you develop embedded software, Conclusions.

Introducing the 8051 Microcontroller Family:

Introduction, What's in a name, The external interface of the Standard 8051, Reset requirements, Clock
frequency and performance, Memory issues, I/O pins, Timers, Interrupts, Serial interface, Power consumption, Conclusions.

Unit-II:

Reading Switches: Introduction, Basic techniques for reading from port pins, Example: Reading and writing bytes, Example: Reading and writing bits (simple version), Example: Reading and writing bits (generic version), The need for pull-up resistors, Dealing with switch bounce, Example: Reading switch inputs (basic code), Example: Counting goats, Conclusions.

Unit-III:

Code Adding Structure to the Code: Introduction, Object-oriented programming with C, The Project Header (MAIN.H), The Port Header (PORT.H), Example: Restructuring the 'Hello Embedded World' example, Example: Restructuring thegoat-counting example, Further examples, Conclusions.

Unit-IV:

Meeting Real-Time Constraints: Introduction, Creating 'hardware delays' using Timer 0 and Timer 1, Example: Generating a precise 50 ms delay, Example: Creating a portable hardware delay, Why not use Timer 2?, The need for 'timeout' mechanisms, Creating loop timeouts, Example: Testing loop timeouts, Example: A more reliable switch interface, Creating hardware timeouts, Example: Testing a hardware timeout, Conclusions.

Unit-V:

Case Study-Intruder Alarm System: Introduction, The software architecture, Key software components used in this example, running the program, the software, Conclusions.

Text Books:

1. Embedded C - Michael J. Pont, 2nd Ed., Pearson Education, 2008.

Reference Books:

1. PIC MCU C-An introduction to programming, The Microchip PIC in CCS C – Nigel Gardner.
2. Embedded System Design - Frank Vahid, Tony Givargis, John Wiley.
3. Embedded Systems – Lyla, Pearson, 2013.
4. An Embedded Software Primer - David E. Simon, Pearson Education.

e Resources:

1. NPTEL : http://nptel.ac.in/courses/108102045/
2. Embedded Systems 1. http://nptel.ac.in/courses/108102045/

HARDWARE SOFTWARE CO DESIGN

Course Overview:

This course introduces the fundamentals of embedded software co-design, focusing on hardware– software partitioning, co-simulation, and verification techniques. Students will learn methodologies, tools, and languages for building efficient and reliable embedded systems through integrated hardware– software development

Course Objective:

- To introduce the fundamentals of embedded hardware-software co-design principles.
- To understand co-design methodologies, partitioning, and synthesis techniques.
- To explore prototyping, emulation, and system-level design languages.
- To study embedded compilation tools and verification methodologies.
- To gain practical experience with embedded co-design case studies and applications.

Course Outcomes:

- Apply co-design methodologies for embedded software-hardware systems.
- Analyze and partition embedded applications into hardware and software components.
- Design and evaluate co-simulation and co-synthesis techniques for embedded systems.
- Develop and verify embedded applications using co-design tools and languages.

UNIT- I

Co- Design Issues :Co- Design Models, Architectures, Languages, A Generic Co-design Methodology.

Co- Synthesis Algorithms: Hardware software synthesis algorithms: hardware – software partitioning distributed system co synthesis.

UNIT-II

Prototyping and Emulation

Prototyping and emulation techniques, prototyping and emulation environments, future developments in emulation and prototyping architecture specialization techniques, system communication infrastructure **Target Architectures:**Architecture Specialization techniques, System Communication infrastructure, Target Architecture and Application System classes, Architecture for control dominated systems (8051-Architectures for High performance control), Architecture for Data dominated systems (ADSP21060, TMS320C60), Mixed Systems.

UNIT-III:

Compilation Techniques and Tools for Embedded Processor Architectures

Modern embedded architectures, embedded software development needs, compilation technologies, practical consideration in a compiler development environment.

UNIT-IV:

Design Specification and Verification

Design, co-design, the co-design computational model, concurrency coordinating concurrent computations, interfacing components, design verification, implementation verification, verification tools, Interface verification.

UNIT-V:

Languages for System-Level Specification and Design-I

System-level specification, design representation for system level synthesis, system level specification languages.

Languages for System-Level Specification and Design-II

Heterogeneous specifications and multi-language co-simulation, the cosyma system and lycos system..

Text Books:

1.Hardware / Software Co- Design Principles and Practice – Jorgen Staunstrup, Wayne Wolf – 2009, Springer.
2. Hardware / Software Co- Design - Giovanni De Micheli, Mariagiovanna Sami, 2002, Kluwer Academic Publishers.

Reference Books:

1. A Practical Introduction to Hardware/Software Co-design -Patrick R. Schaumont - 2010 – Springer Publications.

e-Resources:

1.NPTEL : https://nptel.ac.in/courses/106103182

ADVANCED COMPUTER ARCHITECTURE

Course Overview:

This course introduces the fundamentals of parallel computer architecture and models. It covers processor technologies, memory hierarchies, pipelining, and superscalar techniques. Students learn about parallelism, scheduling, interconnection networks, and performance metrics.

Course Objective:

- To Understand the Foundations of Parallel Computing.
- To Analyze Parallel Program Design and System Interconnects.
- To Explore Processor and Memory Architectures.
- To Examine Advanced Parallel Processing Techniques.

Course Outcomes:

- **Understand and compare** various parallel computer models, including multiprocessors and multicomputers.
- **Analyze and apply** program partitioning, scheduling techniques, and interconnect architectures for achieving effective parallelism in computing systems.
- **Evaluate** advanced processor technologies, memory hierarchy, cache and shared memory organizations to optimize processor and memory interactions in high-performance systems.
- **Design and assess** pipelining techniques, superscalar architectures, and multithreading models for efficient execution in multiprocessor and vector processing systems.

UNIT- I

Parallel Computer Models – System attributes to performance, Multiprocessors and Multicomputers,

Classifications of Architectures, Multi vector and SIMD Computers, Architecture development tracks.

UNIT-II:

Program and Network Properties- Conditions for parallelism, Program partitioning and Scheduling, Program flow mechanisms, System interconnect architectures, Performance metrics and measures, Parallel Processing Applications.

UNIT-III:

Processors and Memory Hierarchy- Advanced Processor Technology, Superscalar and Vector processors, Memory hierarchy technology, Virtual Memory, Backplane bus systems, Cache memory organizations, Shared memory organizations.

UNIT-IV:

Pipelining and Superscalar Techniques Linear Pipeline processors, Nonlinear pipeline processors, Instruction pipeline design, Arithmetic pipeline design, Superscalar and Super Pipeline Design.

UNIT-V:

Multiprocessors and Multicomputers Multiprocessor System Interconnects, Cache Coherence and Synchronization mechanisms, Three generations of Multicomputers, Message passing mechanisms, Vector Processing principles, Principles of Multithreading.

Text Books:

1. Hwang kai, “Advanced Computer Architecture”, McGraw-Hill, 2001.

Patterson, David and Hennessy John, Morgn Kaufmann, “Computer Architecture”,2001.

Reference Books:

1. William Stallings, Computer Organization and Architecture, 8th Edition, Prentice-Hall India, 2010.
2. David A Patterson and John L. Hennesey, Computer Organization and Design, 4th Edition, Elsevier India, 2011.
3. Andrew S Tanenbaum and James R Goodman, Structured Computer Organization, 5th Edition Prentice Hall India, 2009.

e-Resources:

1. NPTEL : https://onlinecourses.nptel.ac.in/noc25_cs01/preview

INTERNET OF THINGS

Course Overview:
This course introduces Embedded Systems and IoT architectures, covering ARM microcontroller programming, Arduino, Raspberry Pi, and Python-based sensor/actuator interfacing. It also explores IoT platforms, cloud integration, and real-world application deployment using tools like IBM Watson and Node-RED.

Course Objective:

- To identify problems that are amenable to solution by AI methods, and which AI methods may be suited to solving a given problem.
- To formalize a given problem in the language/framework of different AI methods (e.g., as a search problem, as a constraint satisfaction problem, as a planning problem, as a Markov decision process, etc).
- To implement basic AI algorithms (e.g., standard search algorithms or dynamic programming).
- To design and carry out an empirical evaluation of different algorithms on problem formalization, and state the conclusions that the evaluation supports.

Course Outcomes:

- Demonstrate knowledge and understanding of the security and ethical issues of the Internet of Things.
- Conceptually identify vulnerabilities, including recent attacks, involving the Internet of Thing.
- Develop critical thinking skills.
- Compare and contrast the threat environment based on industry and/or device type.

UNIT– I

Introduction to Embedded Systems and Internet of Things (IOT) :
Architecture of Embedded Systems, Embedded Systems Development process, Architecture of Internet of Things, applications of Embedded Systems and IoT, Design Methodology for IOT Products.

UNIT–II

ARM Microcontrollers Architecture and Programming Architecture, Instruction set, Programming ports, Timer/Counter, Serial communication, Interrupts in C, Introduction ARM mBed platform.

UNIT- III

Overview of Open Source Hardware and Its relevance to IOT Introduction and Programming Arduino Development Board, Working with Sensor Integration, Interfacing Input/Output devices (LCD,Key Pad, LED Matrix etc)

UNIT IV

Fundamentals of Python Programming & Raspeberry PI Introduction to python programming, Working with functions, classes, REST full Web Services, Client Libraries, Introduction & programming Raspberry Pi3,Integrating Input Output devices with Raspberry Pi3.

UNIT V

IOT: Technologies, Standards and Tools

Fundamental characteristics and high level requirements of IoT, IoT Reference models, Introduction to Communication Technologies & Protocols of IoT: BLE, Wi-Fi, LoRA, 3G/4G Technologies and HTTP, MQTT, CoAP protocols, Relevant Practical's on above technologies

IOT Platform: Cloud Computing Platforms for IOT Development (IBM CLOUD – Register in IBM Blue mix website) IOT Platform Architecture (IBM Internet of Things & Watson Platforms), API Endpoints for Platform Services, Devices Creation and Data Transmission, Introduction to NODE-RED and Application deployment.

Text Books :

1. Internet of Things: A Hands-On Approach by by Arsheep Bahga, Vijay Madisetti
2. Embedded Real Time Systems: Concepts, Design and Programming‖ by Dr.K.V.K.K.Prasad, Dream Tech Publication, 2003.

Reference Books:

1.Embedded Systems: Real-Time Interfacing to Arm(r) Cortex -M Microcontrollers: volume-1 & 2 by Jonathan W Valvano.

2. Designing the Internet of Things‖ by Adrian McEwen, Hakim Cassimally, Wiley Publications, 2012

e Resources:

1. http://www.slideshare.net/RealTimeInnovations/io-34485340
2. http://internetofthings.electronicsforu.com

ANALOG CMOS CIRCUIT DESIGN LAB

Course Overview: This lab course focuses on designing and simulating analog CMOS circuits using industry-standard EDA tools. Students gain hands-on experience in schematic design, layout, simulation, and verification, enhancing their understanding of VLSI design methodologies and analog integrated circuit performance optimization.

Course Objective:

- To understand CMOS analog design using EDA tools and techniques.
- To develop schematics and layouts for optimized analog circuit performance.
- To perform simulations verifying functionality of pre-layout and post-layout designs.
- To apply physical verification ensuring correctness and manufacturability of layouts.

Course Outcomes:

- Apply VLSI design methodologies using Mentor Graphics tools to design and verify IC circuits.
- Analyze CMOS analog circuits' role in full-custom IC design flows for performance optimization.
- Apply physical verification techniques in layout design to ensure design correctness and manufacturability.
- Create pre-layout and post-layout simulations to validate and optimize VLSI designs for functionality.

List of Experiments:

Exp No.	**Experiment Name**
1	MOS Device Characterization and parametric analysis
2	Common Source Amplifier.
3	Common Source Amplifier with source degeneration.
4	Cascode amplifier.
5	Simple current mirror.
6	Cascode current mirror.
7	Wilson current mirror.
8	Differential Amplifier.
9	Operational Amplifier.
10	Sample and Hold Circuit.
11	Direct-conversion ADC.
12	R-2R Ladder Type DAC.

Lab Requirements:
Software:
Mentor Graphics – Pyxis Schematic, IC Station, Calibre, ELDO Simulator/
Industry Equivalent Standard Software
Hardware:
Personal Computer with necessary peripherals, configuration and operating System.

EMBEDDED SYSTEM DESIGN LAB

Course Objectives :
IDE for Embedded System Design using MSP430; Interfacing Switch & LED; Timers-WDT, Configuring, Programming; ADC-usage; Power down modes; DAC; PWM Generator; Networking - SPI, Wi-Fi.

Course Outcomes :
- Demonstrate knowledge in designing complex energy efficient embedded systems.
- Analyze usage of various on-chip resources like GPIO, Timers, Interrupts, ADC, DAC, Comparator, SPI.
- Apply appropriate techniques, resources, and CCSV6 based IDE for modeling embedded systems with understanding of limitations.
- Work individually and in a group to develop embedded systems.

List of Experiments:

Exp No.	**Experiment Name**
1	Introduction to MSP430 launch pad and Programming Environment.
2	Read input from switch and Automatic control/flash LED (soft-ware delay).
3	Interrupts programming example using GPIO.
4	Configure watchdog timer in watchdog & interval mode.
5	Configure timer block for signal generation (with given frequency).
6	Read Temperature of MSP430 with the help of ADC.
7	Test various Power Down modes in MSP430.
8	PWM Generator.
9	Use Comparator to compare the signal threshold level.
10	Speed Control of DC Motor.
11	Master slave communication between MSPs using SPI.
12	Networking MSPs using Wi-Fi.

TOOL REQUIREMENT:

Code Composer Studio Version 6, MSP430 based launch pads, Wi-Fi booster pack.

CMOS MIXED SIGNAL DESIGN

Course Overview: This course covers advanced topics in analog CMOS design, including two-stage op-amp design, switched capacitor circuits, sample-and-hold circuits, comparators, data converters, and phase-locked loops (PLLs). Emphasis is placed on circuit analysis, noise reduction, parasitic effects, and performance limitations. Students will learn practical design strategies for high-performance analog circuits.

Course Objective:

- To design and analyze two-stage operational amplifiers, focusing on parasitic effects and biasing.
- To explore switched capacitor circuits and their applications in filter design and integrators.
- To understand the principles and design of comparators, including issues like propagation delay and slew rate.
- To study data converters and PLLs, emphasizing performance limitations, noise, and circuit optimization.

Course Outcomes:

- Understand the necessity of mixed signal syatem.
- Analyze Op-Amp to meet the mixed signal specifications.
- Design CMOS comparators to meet the high- speed requirements of digital circuitry.
- Develop efficient data converter circuits for mixed signal systems.

UNIT- I

Two-Stage OP-AMP Design: Parasitic Effects on Design of Two-Stage OP-AMP, Wide-Swing Cascode Current Mirrors, Design of Rugged Biasing Circuit with Temperature-Independent Compensation, Challenges in Mixed-Signal Circuit Design.

Switched Capacitor Circuits : Constituents: Op-Amp, Capacitors, Switches, Non- overlapping Clocks; Basic Operation and Analysis, Resistor Equivalence of a Switched Capacitor, Parasitic-Sensitive Integrator, Parasitic-Insensitive Integrators, Signal-Flow- Graph Analysis, Design of Filters Based on Switched Capacitor Circuits.

UNIT- II

Sample-and-Hold Circuit: Testing Sample and Holds, MOS Sample-and-Hold Basics, Examples of CMOS S/H Circuits, Charge-Injection Errors, Making Charge-Injection Signal Independent, Minimizing Errors Due to Charge-Injection, Effect of Offset and Application of Switched Capacitor Circuits to Minimize Offset Errors, Parasitic Effects.

UNIT- III

Comparators: Ideal Comparator, Practical Model of Comparator, Resolving Capability, Propagation Delay, Small Signal Analysis, Conditions for Slewing, Evaluation of Propagation Delay for Single Pole and Two Pole Comparators, Design of Linear Response Comparators,

Slew-Rate Limited Comparators, Comparators with Positive Feedback, Analysis of Latched Comparators, Architecture of High-Speed Comparators, Self-Biased
Comparators, Push Pull Comparators.

UNIT- IV

Data Converters: Classification, Ideal D/A Converter, Ideal A/D Converter, Quantization Noise: Deterministic and Stochastic Approach, Signed Codes, Performance Limitations: Resolution, Offset and Gain Error, Accuracy and Linearity, Integrating Converters, Design of Successive-Approximation Converters, DAC-Based and Charge-Redistribution SAR, Interleaved, Pipelined, Flash, Principles of Sigma-Delta ADC, Testing of Data Converters.

UNIT- V

PLL and Oscillators: Basic PLL Architecture, VCO, Divider, Phase Detector, Loop Filter, PLL in Lock, Linearized Small-Signal Analysis, Second-Order PLL Model, Limitations, PLL Characterization and Design Example, Jitter and Phase Noise: Period Jitter, Cycle Jitter, Adjacent Period Jitter, Spectral Representations and PDF of Jitter, Ring and LC Oscillators, Phase Noise in Oscillators and PLLs.

Text Book:

1. David Johns, Tony Chan Carusone and Kenneth Martin, Analog Integrated Circuit Design, Wiley, 2012, 2ndEdition.
2. Behzad Razavi, Design of Analog CMOS Integrated Circuits"McGraw Hill Education, 2017, 2ndEdition.

Referenc Books:

1. Roubik Gregorian and Gabor C. Temes, Analog MOS integrated circuits for signal processing, Wiley, 1986.
2. Roubik Gregorian, Introduction to CMOS Op-Amps and Comparators, Wiley, 2008.

e-Reference:

NPTELCourses(https://onlinecourses.nptel.ac.in/noc22_ee34/preview)

RTOS

Course Overview:

This course explores Real-Time Operating Systems (RTOS) with a focus on embedded systems. Key topics include task scheduling, interrupt handling, and memory management. Students gain hands-on experience with RTOS like µC/OS-II, VxWorks, and RT Linux. Practical skills in Unix/Linux programming and real-time application development are emphasized.

Course Objective:

- To introduce the fundamental services and features of operating systems, with emphasis on real-time constraints.
- To familiarize students with programming concepts of different RTOS used in embedded systems.
- To provide knowledge of program modeling through real-world case studies in embedded systems.
- To develop skills for RTOS-based application development and target image creation using Linux and to understand and implement real-time programming concepts using RT Linux and its APIs.

Course Outcomes:

- Explain the architecture, services, and functions of operating systems in a real-time environment.
- Demonstrate the ability to write and analyze programs using µC/OS-II, VxWorks, Windows CE, and RT Linux.
- Model and evaluate embedded system applications through case studies like smart cards, mobile phones, etc.
- Develop and debug RTOS-based applications and perform target image creation using Linux tools.

UNIT- I

Introduction: OS Services, Process Management, Timer Functions, Event Functions, Memory Management, Device, File and IO Systems Management, Interrupt Routines in RTOS Environment and Handling of Interrupt Source Calls, Real-Time Operating Systems, Basic Design Using an RTOS, RTOS Task Scheduling Models, Interrupt Latency and Response of the Tasks as Performance Metrics, OS Security Issues.

UNIT-II:

RTOS Programming: Basic Functions and Types of RTOS for Embedded Systems, RTOS mCOS- II, RTOS Vx Works, Programming concepts of above RTOS with relevant Examples, Programming concepts of RTOS Windows CE, RTOS Linux 2.6.x and RTOS RT Linux.

UNIT-III:

Program Modeling – Case Studies: Case study of digital camera hardware and software architecture, Case Study of Embedded System for an Adaptive Cruise Control (ACC) System in Car, Case Study of Embedded System for a Smart Card, Case Study of Embedded System of Mobile Phone Software for Key Inputs..

UNIT-IV:

Target Image Creation & Programming in Linux: Operating System Software, Target Image Creation for Window XP Embedded, Porting RTOS on a Micro Controller based Development Board. Overview and programming concepts of Unix/Linux Programming, Shell Programming, System Programming.

UNIT-V:

Programming in RT Linux: Overview of RT Linux, Core RT Linux API, Program to display a message periodically, semaphore management, Mutex, Management, Case Study of Appliance Control by RT Linux System.

Text Books:

1 Rajkamal: "Embedded Systems-Architecture, Programming and Design", Tata McGraw Hill Publications, Second Edition, 2008.

2. Dr. K.V.K.K. Prasad: "Embedded/Real-Time Systems" Dream Tech Publications, 2005 Edition, Black pad book.

Reference Books:

1. Labrosse, "Embedding system building blocks ", CMP publishers.
2. Rob Williams," Real time Systems Development", Butterworth Heinemann Publications.

e-Resources:

1. https://nptel.ac.in/courses/106105172

ADVANCED PROCESSORS AND CONTROLLERS

Course Overview:

This course introduces the architecture and programming of modern processors including the Pentium, ARM, Motorola 68HC11, and PIC microcontrollers. It covers instruction sets, operating modes, interrupts, and memory management. Emphasis is placed on application development using ARM processors and embedded systems concepts..

Course Objective:

- To understand the architecture, instruction set, and operation of general-purpose and embedded processors.
- To explore the ARM architecture and its application in high-performance embedded systems.
- To develop practical skills in programming microcontrollers (ARM, 68HC11, PIC) in Assembly and C.
- To understand peripheral interfacing, interrupt handling, and memory management in embedded systems.

Course Outcomes:

- Describe CPU architectures, instruction sets, and basic operations of Pentium and RISC processors.
- Analyze ARM architecture, instruction cycle timing, and implement optimized embedded programs.
- Develop ARM-based applications including DSP functions, bootloaders, and peripheral interface routines.
- Program and interface with microcontrollers (68HC11, PIC) using C and assembly language.

UNIT- I

High Performance Cisc Architecture –Pentium CPU Architecture - Bus Operations – Pipelining – Brach predication – floating point unit- Operating Modes –Paging –Multitasking – Exception and Interrupts – Instruction set – addressing modes – Programming the Pentium processor.

UNIT-II:

High Performance Risc Architecture

ARM Arcon RISC Machine – Architectural Inheritance – Core & Architectures - Registers – Pipeline- Interrupts – ARM organization - ARM processor family – Co-processors - ARM instruction set- Thumb Instruction set - Instruction cycle timings - The ARM Programmer‟s model – ARM Development tools – ARM Assembly Language Programming – C programming – Optimizing ARM Assembly Code – Optimized Primitives.

UNIT-III:

Arm Application Development

Introduction to DSP on ARM –FIR filter – IIR filter – Discrete fourier transform – Exception handling-Interrupts –Interrupt handling schemes- Firmware and boot loader – Embedded Operating systems-Integrated Development Environment- STDIO Libraries – Peripheral Interface – Application of ARM Processor - Caches – Memory protection Units – Memory Management units – Future ARM Technologies.

UNIT-IV:

Motorola 68hc11 Microcontrollers

Instruction set addressing modes – operating modes- Interrupt system- RTC-Serial Communication Interface – A/D Converter PWM and UART.

UNIT-V:

Pic Microcontroller

CPU Architecture – Instruction set – interrupts- Timers- I2C Interfacing –UART- A/D Converter – PWM and introduction to C-Compilers.

Text Books:

1. Andrew N.Sloss, Dominic Symes and Chris Wright " ARM System Developer"s Guide : Designing and Optimizing System Software" , First edition, Morgan Kaufmann Publishers, 2004.
2. Steve Furber , "ARM System –On –Chip architecture", Addision Wesley, 2000.

Reference Books:

1. Daniel Tabak , "Advanced Microprocessors", Mc Graw Hill. Inc., 1995
2. James L. Antonakos , " The Pentium Microprocessor", Pearson Education, 1997.

e-Resources:

1. https://onlinecourses.nptel.ac.in/noc22_ee12/preview

DESIGN FOR TESTABILITY

Course Overview:
This course covers the principles and techniques for designing testable digital systems in VLSI. It includes fault modeling, logic and fault simulation, and testability analysis. Students learn scan-based design, BIST, and boundary scan techniques. Emphasis is placed on creating reliable, manufacturable systems through effective test design..

Course Objective:
- To understand the fundamental concepts and philosophies of digital system testing.
- To explore various fault modeling and simulation techniques for test evaluation.
- To introduce and apply scan-based design and testability enhancement techniques.
- To analyze and implement BIST and boundary scan architectures in digital systems.

Course Outcomes:
- Explain various fault models, simulation techniques, and their role in VLSI testing.
- Apply testability analysis methods and scan-based design techniques to digital circuits.
- Analyze and implement built-in self-test (BIST) architectures for logic and memory circuits.
- Design and configure boundary scan systems using IEEE standards and interpret BSDL specifications.

UNIT- I
Introduction to Testing: Testing Philosophy, Role of Testing, Digital and Analog VLSI Testing, VLSI Technology Trends affecting Testing, Types of Testing, Fault Modeling: Defects, Errors and Faults, Functional Versus Structural Testing, Levels of Fault Models, Single Stuck-at Fault.
UNIT-II:
Logic and Fault Simulation: Simulation for Design Verification and Test Evaluation, Modeling Circuits for Simulation, Algorithms for True-value Simulation, Algorithms for Fault Simulation, ATPG.
UNIT-III:
Testability Measures: SCOAP Controllability and Observability, High Level Testability Measures, Digital DFT and Scan Design: Ad-Hoc DFT Methods, Scan Design, Partial-Scan Design, Variations of Scan.
UNIT-IV:
Built-In Self-Test: The Economic Case for BIST, Random Logic BIST: Definitions, BIST Process, Pattern Generation, Response Compaction, Built-In Logic Block Observers, Test-Per-Clock, Test Per-Scan BIST Systems, Circular Self Test Path System, Memory BIST, Delay Fault BIST.
UNIT-V:
Boundary Scan Standard: Motivation, System Configuration with Boundary Scan: TAP Controller and Port, Boundary Scan Test Instructions, Pin Constraints of the Standard, Boundary

Scan Description Language: BDSL Description Components, Pin Descriptions.

Text Books:

1. M.L. Bushnell, V. D. Agrawal, "Essential of Electronic Testing for Digital, Memory and Mixed Signal VLSI Circuits", Kluwer Academic Publishers.

Reference Books:

1. M. Abramovici, M. A. Breuer and A.D Friedman, Digital Systems and Testable Design", Jaico Publishing House.
2. P. K. Lala, "Digital Circuits Testing and Testability", Academic Press.

e Resources:

https://nptel.ac.in/courses/117105137

PHYSICAL DESIGN VERIFICATION

Course Overview: This course covers VLSI physical design, focusing on design cycles, fabrication processes, and layout techniques. Topics include design styles, interconnect delay, complexity issues, graph algorithms, and data structures for layout editors. It also explores partitioning, floor planning, pin assignment, and routing algorithms, equipping students with tools for efficient VLSI design.

Course Objective:

- To understand the VLSI design and physical design cycles, including fabrication and layout processes.
- To explore design styles and interconnect issues, focusing on noise, crosstalk, and yield.
- To study graph algorithms and data structures for solving physical design problems.
- To learn partitioning, floor planning, and routing algorithms for efficient VLSI design.

Course Outcomes:

- Understand the relationship between design automation algorithms and Various constraints posed by VLSI fabrication and design technology.
- Adapt the design algorithms to meet the critical design parameters.
- Identify layout optimization techniques and map them to the algorithms.
- Develop proto-type EDA tool and test its efficacy.

UNIT- I

VLSI Design Cycle, Physical Design Cycle, Design Rules, Layout of Basic Devices, and Additional Fabrication. Design styles: Full Custom, Standard Cell, Gate Arrays, Field Programmable Gate Arrays, Sea of Gates and Comparison, System Packaging Styles, Multi- Chip Modules. Design Rules, Layout of Basic Devices, Fabrication Process and Its Impact on Physical Design, Interconnect Delay, Noise and Crosstalk, Yield and Fabrication Cost.

UNIT- II

Factors, Complexity Issues and NP-hard Problems. Basic Algorithms (Graph and Computational Geometry): Graph Search Algorithms, Spanning Tree Algorithms, Shortest Path Algorithms, Matching Algorithms, Min-Cut and Max-Cut Algorithms, Steiner Tree Algorithms.

UNIT- III

Basic Data Structures: Atomic Operations for Layout Editors, Linked List of Blocks, Bin Based Methods, Neighbour Pointers, Corner Stitching, Multi-Layer Operations.

UNIT- IV

Graph Algorithms for Physical Design: Classes of Graphs, Graphs Related to a Set of Lines, Graphs Related to a Set of Rectangles, Graph Problems in Physical Design, Maximum Clique and Minimum Coloring, Maximum k-Independent Set Algorithm, Algorithms for Circle Graphs.

UNIT- V

Partitioning Algorithms: Design Style Specific Partitioning Problems, Group Migrated Algorithms, Simulated Annealing and Evolution. Floor Planning and Pin Assignment, Routing and Placement Algorithms.

Text Books:

1. Naveed Shervani, Algorithms for VLSI Physical Design Automation, 3rd Edition, Kluwer Academic, 1999.
2. Charles J Alpert, Dinesh P Mehta, Sachin S Sapatnekar, Handbook of Algorithms for Physical Design Automation, CRC Press, 2008.

Reference Books:

1. "CMOS VLSI Design: A Circuits and Systems Perspective" by Neil H. E. Weste and David Harris.
2. "VLSI Physical Design: From Graph Partitioning to Timing Closure" by K. D. Meade and L. F. D. DeMicheli.

e-Reference:

NPTEL Courses (**https://nptel.ac.in/courses/106103016**)

SYSTEM VERILOG & UVM

Course Overview:

This course introduces System Verilog and UVM for advanced hardware verification. It covers testbench design, constrained random verification, and functional coverage techniques. Students learn to create assertions and apply formal verification methods to debug designs. The course also addresses hardware security threats and countermeasures using modern verification tools.

Course Objective:

- To introduce students to digital design verification using System Verilog and object- oriented programming.
- To develop skills in building layered, coverage-driven testbenches using System Verilog and UVM.
- To understand formal verification concepts, assertions, and concurrency mechanisms for robust bug detection.
- To explore the dimensions of hardware security including threats like IP piracy and Trojans, and mitigation strategies.

Course Outcomes:

- Understand the principles of System Verilog verification, including testbench design and OOP features.
- Apply constrained random verification and functional coverage to evaluate DUT behavior using System Verilog & UVM.
- Analyze and construct assertions and formal verification strategies to detect and debug design flaws.
- Identify and assess hardware security threats and apply suitable countermeasures using modern verification tools.

UNIT- I

System Verilog Design Verification-I: The concept of Device under test (DUT), Complexity of verification, lexical elements of system Verilog (SV), Layered testbench and SV verification environment.

Unit-II:

System Verilog Design Verification-II: - Basics of SV testbench, concurrency in system Verilog, object oriented programming, encapsulation and randomization.

Unit-III:

System Verilog Design Verification –III: - Inter-thread communications, mailbox, code coverage, functional coverage and building efficient SV testbenches. System Verilog UVM.

Unit-IV:

Formal Verification (FV): -Importance of FV, Challenges in implementing FV, Boolean Algebra, Boolean satisfiability, Basic assertion concepts, immediate assertions, concurrent assertions, Sequences, clocks, resets and coverage.

Unit-V:

Hardware Security: - Reasons for raise of hardware security issues, IC Counterfeiting, IP piracy, Hardware Trojans, Debug security and applications of Physical Unclonable Functions(PUF).

Text Books:

1. System Verilog for Verification: A Guide to Learning the Testbench Language Features by Chris Spear , Greg Tumbush, Springer; 3rd ed. 2012 edition.
2. System Verilog Assertions and Functional Coverage: Guide to Language, Methodology and Applications, by Ashok B. Mehta, Springer; 2014 edition.

Reference Books:

1. Formal Verification: An Essential Toolkit for Modern VLSI Design 1st Edition by Erik Seligman, Tom Schubert , M V Achutha Kiran Kumar, Morgan Kaufmann; 2015.
2. Clifford E. Cummings, “System verilog Assertions - Bind files & Best Known Practices for Simple SVA Usage”

e-Resources:

1. https://onlinecourses.nptel.ac.in/noc21

LOW POWER VLSI DESIGN

Course Overview:
This course focuses on power-aware design methodologies essential for modern VLSI systems. It examines dynamic and static power dissipation in digital circuits and strategies to minimize them. Topics include voltage scaling, leakage reduction, and efficient clocking. Students will also learn power estimation and optimization through analysis and simulation.

Course Objectives:
- To understand the different sources and types of power dissipation in digital CMOS circuits.
- To Analyze power dissipation characteristics at gate-level and circuit-level.
- To Explore voltage scaling and its challenges as a means for reducing power.
- To Apply architectural techniques such as parallelism and pipelining for low-power design.

Course Outcomes:
- Identify the causes of dynamic and static power dissipation and quantify them using analytical models.
- Apply supply voltage scaling and architectural techniques like pipelining and parallelism to reduce power consumption.
- Analyze and implement switching activity reduction techniques such as clock gating, bus encoding, and FSM optimization.
- Use simulation-based tools and modeling approaches (e.g., SPICE, Monte Carlo) to estimate and minimize power in VLSI circuits.

UNIT- I

Sources of Power Dissipation: Introduction, Short-Circuit Power Dissipation, Switching Power Dissipation, Dynamic Power for Complex Gate, Reduced Voltage Swing, Switching Activity, Leakage Power Dissipation, p–n Junction Reverse-Biased Current, Band-Band Tunneling Current, Subthreshold Leakage Current, Short-Channel Effects.

UNIT-II:

Supply Voltage Scaling for Low Power: Device Feature Size Scaling, Constant-Field Scaling, Constant-Voltage Scaling, Architectural- Level Approaches: Parallelism for Low Power, Pipelining for Low Power, Combining Parallelism with Pipelining, Voltage Scaling Using High-Level Transformations: Multilevel Voltage Scaling Challenges in MVS Voltage Scaling Interfaces, Static Timing Analysis
Dynamic Voltage and Frequency Scaling.

UNIT-III:

Switched Capacitance Minimization: Probabilistic Power Analysis: Random logic signals, probability and frequency, probabilistic power analysis techniques, signal entropy, Bus Encoding: Gray Coding, One-Hot Coding, Bus- Inversion, T0 Coding, Clock Gating, Gated-Clock FSMs FSM State Encoding, FSM Partitioning, Precomputation, Glitching Power Minimization.

Unit-IV:

Leakage Power Minimization: Fabrication of Multiple Threshold Voltages, Multiple Channel Doping, Multiple Oxide CMOS, Multiple Channel Length, Multiple Body Bias, VTCMOS Approach, MTCMOS Approach, Power Gating, Clock Gating Versus Power Gating, Power-Gating Issues, Isolation Strategy, State Retention Strategy, Power-Gating Controller, Power Management, Combining DVFS and Power Management

Unit-V:

Low power clock distribution & Simulation Power Analysis: Low power clock distribution: Power dissipation in clock distribution, single driver versus distributed buffers, Zero skew versus tolerable skew, chip and package co design for clock network. Simulation Power Analysis: SPICE circuit simulators, gate level logic simulation, capacitive power estimation, architecture level analysis, data correlation analysis of DSP systems, Monte Carlo Simulation.

Text Books:

1. Low-Power VLSI Circuits and Systems, Ajit Pal, SPRINGER PUBLISHERS
2. Practical Low Power Digital VLSI Design ,Gary Yeap Motorola, Springer Science Business Media, Llc.

Reference Books:

1. Low Power CMOS Design – Anantha Chandrakasan, IEEE Press/Wiley International, 1998. 2
2. Massoud Pedram, Jan M. Rabaey,"Low power design methodologies ", Kluwer Academic Publishers.
3. Low Power CMOS VLSI Circuit Design – A. Bellamour, M. I. Elamasri, Kluwer Academic Press, 1995.

e Resources:

https://nptel.ac.in/courses/106105034

NETWORK SECURITY AND CRYPTOGRAPHY

Course Overview: This course covers the principles and techniques used to secure information systems and networks. It includes classical and modern cryptography, authentication, and secure communication protocols. Students learn about threats like intrusions and malware, and countermeasures like firewalls and trusted systems. Emphasis is placed on both theoretical foundations and practical applications of cybersecurity.

Course Objective:

- To understand the fundamental concepts of cryptography and network security mechanisms.
- To explore classical and modern encryption algorithms and their security properties.
- To learn public key cryptographic systems, key exchange protocols, and associated number theory.
- To understand message authentication, digital signatures, and secure communication protocols.
- To study real-world network security applications such as IP security, web security, email security, and threat mitigation.

Course Outcomes:

- Explain the fundamental principles of cryptography and classical encryption techniques.
- Apply symmetric and asymmetric encryption algorithms in securing data communication.
- Analyze and evaluate cryptographic techniques such as RSA, Diffie-Hellman, and digital signatures for secure systems.
- Design and implement secure communication mechanisms using protocols like SSL, IPsec, and authentication systems.

UNIT- I

Introduction: Attacks, Services and Mechanisms, Security attacks, Security services, A Model for Internetwork security. Classical Techniques: Conventional Encryption model, Steganography, Classical Encryption Techniques. Modern Techniques: Simplified DES, Block Cipher Principles, Data Encryption standard, Strength of DES, Differential and Linear Cryptanalysis, Block Cipher Design Principles and Modes of operations.

UNIT-II:

Encryption Algorithms: Triple DES, International Data Encryption algorithm, Blowfish, RC5, CAST-128, RC2, Characteristics of Advanced Symmetric block cifers. Conventional Encryption :Placement of Encryption function, Traffic confidentiality, Key distribution, Random Number Generation. 12 3 Public Key Cryptography: Principles, RSA Algorithm, Key Management.

UNIT-III:

Public Key Cryptography: Principles, RSA Algorithm, Key Management, Diffie-Hellman Key exchange, Elliptic Curve Cryptography. Number Theory: Prime and Relatively prime numbers, Modular arithmetic, Fermat's and Euler's theorems, Testing for primality, Euclid's Algorithm, the Chinese remainder theorem, Discrete logarithms.

UNIT-IV:

Message Authentication and Hash Functions: Authentication requirements and functions, Message Authentication, Hash functions, Security of Hash functions and MACs. Hash and Mac Algorithms. MD File, Message digest Algorithm, Secure Hash Algorithm, RIPEMD-160, HMAC. Digital signatures and Authentication protocols: Digital signatures, Authentication Protocols, Digital signature standards. Authentication Applications: Kerberos, X.509 directory Authentication service. Electronic Mail Security: Pretty Good Privacy, S/MIME..

UNIT-V:

IP Security: Overview, Architecture, Authentication, Encapsulating Security Payload, Combining security Associations, Key Management. Web Security: Web Security requirements, Secure sockets layer and Transport layer security, Secure Electronic Transaction. Intruders, Viruses and Worms Intruders, Viruses and Related threats. Fire Walls: Fire wall Design Principles, Trusted systems.

Text Books:

1. Cryptography and Network Security: Principles and Practice - William Stallings, Pearson Education.
2. Network Security Essentials (Applications and Standards) by William Stallings Pearson Education.

Reference Books:

1. Fundamentals of Network Security by Eric Maiwald (Dreamtech press)
2. Network Security - Private Communication in a Public World by Charlie Kaufman, Radia Perlman and Mike Speciner, Pearson/PHI.
3. Principles of Information Security, Whitman, Thomson.
4. Network Security: The complete reference, Robert Bragg, Mark Rhodes, TMH
5. Introduction to Cryptography, Buchmann, Springer.

e Resources:

https://nptel.ac.in/courses/106105031

EMBEDDED NETWORKING

Course Overview:

This course covers the fundamentals and advanced concepts of embedded networking in both wired and wireless systems. Topics include serial, parallel, USB, CAN, and Ethernet communication protocols. Students explore wireless sensor networks and secure communication in embedded systems. Practical learning involves programming microcontrollers like the PIC18 for network interfacing and design.

Course Objective:

- To understand the principles and applications of serial, parallel, and bus-based communication in embedded systems.
- To explore USB and CAN bus communication protocols and implement simple applications using micro controllers.
- To analyze and implement Ethernet-based communication for embedded systems.
- To design and secure embedded systems capable of TCP/IP networking and internet-based communication.

Course Outcomes:

- Explain various embedded communication protocols including RS232, SPI, I2C, and PCI.
- Develop and implement USB and CAN- based communication using PIC microcontrollers.
- Design Ethernet-enabled embedded systems using TCP/IP and serve dynamic web content.
- Analyze and compare wireless embedded networking protocols with focus on MAC and routing efficiency..

UNIT- I

Embedded Communication Protocols:

Embedded Networking: Introduction – Serial/Parallel Communication – Serial communication protocols -RS232 standard – RS485 – Synchronous Serial Protocols -Serial Peripheral Interface (SPI) – Inter Integrated Circuits (I2C) – PC Parallel port programming - ISA/PCI Bus protocols – Firewire.

UNIT-II:

USB and CAN Bus: USB bus-Introduction – Speed Identification on the bus – USB States – USB bus communication: Packets –Data flow types –Enumeration –Descriptors –PIC 18 Microcontroller USB Interface – C Programs –CAN Bus – Introduction - Frames –Bit stuffing – Types of errors – Nominal Bit Timing – PIC microcontroller CAN Interface –A simple application with CAN.

UNIT-III:

Ethernet Basics: Elements of a network – Inside Ethernet – Building a Network: Hardware options – Cables, Connections and network speed – Design choices: Selecting components – Ethernet Controllers – Using the internet in local and internet communications – Inside the

Internet Protocol.

UNIT-IV:

Embedded Ethernet: Exchanging messages using UDP and TCP – Serving web pages with Dynamic Data – Serving web pages that respond to user Input – Email for Embedded Systems – Using FTP – Keeping Devices and Network secure.

UNIT-V:

Wireless Embedded Networking: Wireless sensor networks – Introduction – Applications – Network Topology – Localization –Time Synchronization - Energy efficient MAC protocols – SMAC – Energy efficient and robust routing – Data Centric routing.

Text Books:

1. Embedded Systems Design: A Unified Hardware/Software Introduction - Frank Vahid, Tony Givargis, John & Wiley Publications, 2002
2. Parallel Port Complete: Programming, interfacing and using the PCs parallel printer port - Jan Axelson, Penram Publications, 1996.

Reference Books:

1. Advanced PIC microcontroller projects in C: from USB to RTOS with the PIC18F series - Dogan Ibrahim, Elsevier 2008.
2. Embedded Ethernet and Internet Complete - Jan Axelson, Penram publications, 2003.

e Resources:

https://onlinecourses.nptel.ac.in/noc22_cs93/preview

DATA ACQUISITION SYSTEMS

Course Overview:

This course offers an in-depth study of data acquisition systems (DAS), focusing on ADCs and DACs. It explores converter classifications, design principles, and advanced configurations like high-speed and hybrid systems. Practical applications in communication, DSP, and instrumentation are covered. Key topics include error budgeting, noise reduction, and microprocessor interfacing.

Course Objective:

- To understand the principles and classification of various ADC architectures and their operating mechanisms.
- To analyze non-linear data converter configurations and evaluate
- their applications in real-time systems.
- To design different types of DACs and apply them in programmable and signal- processing systems.
- To valuate performance parameters, error sources, and noise reduction techniques in data acquisition systems.

Course Outcomes:

- Classify different ADC types and explain their working principles.
- Analyze and develop embedded hardware and software development cycles and tools
- Analyze to understand what a microcomputer, core of the embedded system
- Analyze to understand different concepts of a RTOS, sensors, memory interface, communication interface.

UNIT- I

INTRODUCTION: Objective of a DAS, single channel DAS, Multi-channel DAS, Components used in DAS- Converter Characteristics-Resolution-Non-linearity, settling time, Monotonícity.

Unit-II:

ANALOG TO DIGITAL CONVERTERS (ADCS):**Classification of A/D converters**:Parallel feedback, successive approximation, Ramp comparison, Dual slope Integration,voltage to frequency, Voltage to Time, logarithmic types of ADCS.

Non-Linear Data Converters NDC: Basic NDC configurations - Some common NDACS and NADCS, Programmable non-linear ADCS, NADC using optimal sized ROM High speed hybrid NADC, PLS based NADC, Switched capacitor NDCS.

ADC Voice Applications: Data Acquisition systems - Digital signal processing systems - PCM systems, Test and measurement instruments, electronic weighing machines.

Unit-III:

DIGITAL TO ANALOG CONVERTERS (DACS): Principles and design of- Parallel R 2R, Weighted resistor, inverted ladder, D/A decoding-Codes other than ordinary binary. Data

Converter Applications: DAC applications - Digitally programmable V/I sources - Arbitrary waveform generators- Digitally programmable gain amplifiers, Analog multipliers/ dividers, Analog delay lines.

Unit-IV:

Monolithic data converters: Typical study of monolithic DACS and ADCS. Interfacing of DACS and ADCS to a uP.

Unit-V:

Error budget of DACS and ADCS: Error sources, error reduction and noise reduction techniques in DAS. Error budget analysis of DAS, case study of a DAC and an ADC.

Text Books:

1. Electronic data converters fundamentals and applications, Dinesh K. Anvekar, B.S. Sonde - Tata McGraw Hill.
2. Electronic Analog/ Digital conversions - Hermann Schmid- Tata McGraw Hill.

Reference Books:

1.E.R. Hanateck, User's Handbook of D/A and A/D converters – Wiley.
2. Electronic instrumentation by HS Kalsi- TMH 2 nd Edition, 2004.
3. Data converters by G.B. Clayton.

e-Resources:

https://www.youtube.com/watch?v=WwQSfk6SSSo&t=15s

LINUX PROGRAMMING AND OOPS

Course Overview:

This course provides students to Linux operating system fundamentals including file systems, commands, utilities, shell programming, and memory management. It also covers Object- Oriented Programming with C++, focusing on classes, objects, functions, arrays, and strings, enabling students to develop structured and modular software applications. Practical exposure to shell scripting and C++ programming strengthens problem-solving and application development skills.

Course Objective:

- To understand the Linux operating system, its file system, commands, utilities, and memory management policies.
- To develop skills in shell scripting for process automation, text processing, and system administration tasks.
- To learn the principles of Object-Oriented Programming (OOP) using C++, including classes, objects, constructors, functions, and data encapsulation.
- To gain practical experience in implementing OOP concepts for software development using C++ arrays, strings, and class structures.

Course Outcomes:

- Apply Linux commands and shell scripting for file handling, automation, and system utilities.
- Develop shell and AWK scripts for text processing, data manipulation, and computations.
- Implement UNIX system calls in C for file, directory, and process management.
- Demonstrate process creation, client- server communication, and inter-process interaction in C.

UNIT- I

Linux Basics: Introduction to Linux, File System of the Linux, General usage of Linux kernel 7 basic commands, Linux users and group, Permissions for file, directory and users, Searching a file & directory, Zipping and unzipping concepts, Editors and Utilities. Memory Management Policies: Swapping – Demand paging.

Unit-II:

Linux Utilities-File handling utilities, Security by file permissions, Process utilities, Disk utilities, Networking commands, Filters, Text processing utilities and Backup utilities. Sed-Scripts, Operation, Addresses, Commands, awk-Execution, Fields and Records, Scripts, Operation, Patterns, Actions, Associative Arrays, String and Mathematical functions, System commands in awk, Applications.

Unit-III:

Shell programming with Bourne again shell(bash)- Introduction, shell responsibilities, pipes and Redirection, here documents, running a shell script, the shell as a programming language, shell meta characters, file name substitution, shell variables, command substitution, shell commands, the environment, quoting, test command, control structures, arithmetic in shell, shell script examples, interrupt processing, functions, debugging shell scripts.

Unit-IV:

Introduction to Object Oriented Programming: Need for Object Oriented Programming - Characteristics of Object-Oriented Languages – Comparison of C and C++ - Structures: Structures - Enumerations – Functions: Simple Functions – Passing Arguments to Functions – Returning Values from Functions – Reference Arguments - Overloaded Functions – Recursion – Inline Functions – Default Arguments –Scope and Storage Class – Returning by Reference – const Function Arguments.

Unit-V:

Objects and Classes: A Simple Class – C++ Objects as Physical Objects – C++ Objects as Data Types - Constructors – Objects as Function Arguments - Copy Constructor – Structures and Classes – Classes, Objects and Memory - Static class data – Constant Member functions and constant objects - Arrays and Strings: Array Fundamentals – Arrays as Class Member Data – Array of Objects – C-Strings – The Standard C++ String Class.

Text Books:

1. Unix System Programming using C++, T. Chan, PHI, 2015
2. Robert Lafore, Object Oriented Programming in C++, Fourth Edition, Tech Media, 2002. ISBN 0- 672-32308-7

Reference Books:

1. Beginning Linux Programming, 4th Edition, N. Mathew, R. Stones, Wrox, Wiley India Edition, 2007.
2. Unix Concepts and Applications, 4th Edition, Sumitabha Das, TMH, 2017.

e Resources:

NOC: Linux Programming and OOPS https://nptel.ac.in/courses/117106113

DIGITAL CMOS CIRCUIT DESIGN LAB

Course Overview: The Digital CMOS Circuit Design Lab provides hands-on experience with CMOS digital circuit design using industry-standard software tools like Mentor Graphics, Cadence, or Synopsys. Students will design, analyze, and verify various logic circuits, including multiplexers, flip-flops, counters, and memory cells, while learning physical verification and layout extraction techniques.

Course Objective:

- To apply VLSI design methodologies using standard industry EDA tools.
- To design and analyze CMOS logic circuits for digital applications.
- To perform layout extraction and physical verification for CMOS designs.
- To analyze digital CMOS circuits including sequential and combinational components.

Course Outcomes:

- Apply VLSI Design Methodologies using industry-standard tools like Mentor Graphics.
- Analyze and design basic CMOS logic circuits for full-custom IC design.
- Perform physical verification and layout extraction for CMOS circuits.
- Analyze CMOS digital circuits, including sequential and combinational logic components

List of Experiments:

Exp No.	**Experiment Name**
1	Inverter Characteristics.
2	NAND and NOR Gate.
3	XOR and XNOR Gate.
4	2:1 Multiplexer.
5	Full Adder.
6	RS-Latch.
7	Clock Divider.
8	JK-Flip Flop.
9	Synchronous Counter.
10	Asynchronous Counter.
11	Static RAM Cell.
12	Dynamic Logic Circuits.
13	Linear Feedback Shift Register.

Lab Requirements:

Software:

Industry Standard Software (Mentor Graphics Tool/Cadence/ Synopsys/Equivalent)

Hardware:

Personal Computer with necessary peripherals, configuration and operating System.

REAL TIME OPERATING SYSTEMS LAB

Course Objective:

The Students are required to write the programs using C-Language according to the Experiment requirements using RTOS Library Functions and macros ARM-926 developer kits and ARM- Cortex.

Course Outcomes:

Demonstrate the ability to create and manage tasks, handle interrupts, and implement synchronization techniques using PERFECT RTOS on ARM-926

Implement resource allocation and synchronization mechanisms.

Interface peripheral devices (e.g., display, ADC/DAC) with the ARM-Cortex processor.

Develop embedded applications involving serial communication for data logging and modem functionality using ARM-based development boards.

List of Experiments:

Part-I: Experiments using ARM-926 with PERFECT RTOS

1. Register a new command in CLI.
2. Create a new Task.
3. Interrupt handling.
4. Allocate resource using semaphores.
5. Share resource using MUTEX.
6. Avoid deadlock using BANKER'S algorithm.
7. Synchronize two identical threads using MONITOR.
8. Reader's Writer's Problem for concurrent Tasks.

Part-II: Experiments on ARM-CORTEX processor using any open source RTOS. (Coo-Cox-Software-Platform)

1. Implement the interfacing of display with the ARM- CORTEX processor.
2. Interface ADC and DAC ports with the Input and Output sensitive devices.
3. Simulate the temperature DATA Logger with the SERIAL communication with PC.
4. Implement the developer board as a modem for data communication using serial port communication between two PC's.

Lab Requirements:

Software:

- Eclipse IDE for C and C++ (YAGARTO Eclipse IDE), Perfect RTOS Library, COO- COX Software Platform, YAGARTO TOOLS, and TFTP SERVER.

LINUX Environment for the compilation using Eclipse IDE & Java with latest version.

Hardware:

- The development kits of ARM-926 Developer Kits and ARM-Cortex Boards.
- Serial Cables, Network Cables and recommended power supply for the board.