I M.Tech. I Semester

# Embedded System Design

## Course Overview

This course introduces the fundamental concepts and design methodologies of embedded systems. It begins by distinguishing embedded systems from general-purpose computing systems, highlighting their specialized nature and application-specific design. Students will explore the essential building blocks of embedded systems, including hardware components, microcontrollers, and interfacing techniques. Emphasis is placed on the development and role of embedded firmware, including how firmware interacts with hardware through APIs. By the end of the course, students will gain a foundational understanding required to design, develop, and analyze embedded systems used in modern electronic devices and real-time applications.

## Course Outcomes (COs)

By the end of this course, students will be able to:

1. Expected to differentiate the design requirements between General Purpose and Embedded Systems.
2. Expected to acquire the knowledge of firm ware design principles.
3. ExpectedtounderstandtheroleofRealTimeOperatingSysteminEmbeddedDesign.
4. ToacquiretheknowledgeandexperienceoftasklevelCommunicationinanyEmbedded System.

## Syllabus

**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 Interface: Onboard and External Communication Interfaces.

**Unit III:**

**Embedded Firmware:** Reset Circuit, Brown-out Protection Circuit, Oscillator Unit, Real Time Clock, Watchdog Timer, Embedded Firmware Design Approaches and Development Languages.

**Unit IV:**

**RTOS Based Embedded System Design:** Operating System Basics, Types of Operating Systems, Tasks, Process and Threads, Multiprocessing and Multitasking, Task Scheduling.

**Unit V:**

**Task Communication:** Shared Memory, Message Passing, Remote Procedure Call and Sockets,
Task Synchronization: Task Communication/Synchronization Issues, Task Synchronization Techniques, Device Drivers, How to Choose an RTOS.

TEXTBOOK:

1. Introduction to Embedded Systems-ShibuK. V,McGrawHill.

REFERENCEBOOKS

1. Embedded Systems-RajKamal, TMH.
2. Embedded System Design-FrankVahid, Tony Givargis, JohnWiley.
3. Embedded Systems–Lyla, Pearson,2013
4. An Embedded Software Primer-David E.Simon,PearsonEducation.

I M.Tech. I Semester

## CMOS VLSI Design

**Course Overview**

This course provides an in-depth study of CMOS (Complementary Metal-Oxide-Semiconductor) technology and its application in Very Large Scale Integration (VLSI) circuit design. It begins with the fundamentals of MOS device operation and transient response, laying the groundwork for designing efficient digital circuits. Students will explore the design methodologies of both combinational and sequential MOS logic circuits. The course also covers advanced topics such as dynamic logic techniques and memory cell design, enabling students to analyze and implement complex integrated circuits. By the end of the course, learners will be equipped with the theoretical knowledge and practical skills needed to design and optimize CMOS-based VLSI systems.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Design of combinational MOS logic and sequential MOS logic circuits
2. Design of different Memories using MOS transistors
3. Design a circuits based on dynamic logic
4. Use CMOS transmission gates in various applications

### Syllabus

Unit I:

**MOS Design**

Pseudo NMOS logic- Inverter, Inverter threshold voltage, output high voltage, output low voltage, gain at gate threshold voltage, transient response, risetime, fall time, pseudo NMOS logic gates, transistor equivalency, CMOS inverter logic.

**Unit II:**

**Combinational MOS logic circuits**

MOS logic circuits with NMOS loads, Primitive CMOS logic gates- NOR and NAND gates,
Complex logic circuits design realizing Boolean expressions using NMOS gates and CMOS gates, AOI and OIA gates, CMOS full adder, CMOS transmission gates, designing with transmission gates.

**Unit III:**

**Sequential MOS logic circuits:** Behavior of bistable elements, SR Latch, Clocked Latch and Flip-flop circuits, CMOS D Latch and edge triggered flip-flop.

**Unit IV:**

**Dynamic Logic Circuits:** Basic principle, Voltage Boots trapping, Synchronous
Dynamic pass transistor circuits, Dynamic CMOS transmission gate logic, high performance dynamic CMOS circuits.

**Unit V:**

**Semiconductor Memories:** Types, RAM array Organization, DRAM types, operation, leakage currents in DRAM cell and refresh operation, SRAM operation, leakage currents in SRAM cells, Flash memory- NOR flash and NAND flash.

**TEXTBOOKS:**

1. Digital Integrated Circuit Design-Ken Martin, OxfordUniversityPress,2011.
2. CMOS Digital Integrated Circuit Analysis and Design–Sung MoKang,YusufLeblebici, TMH, 3rdEd., 2011.

**REFERENCEBOOKS:**

1. Introduction to VLSI Systems: A Logic, Circuit and System Perspective-MingBoLin, CRC Press, 2011.
2. Digital Integrated Circuits: A Designs Perspective -JanM.Rabaey,Anantha Chandrakasan, Borivoje Nikolic, 2ndEd., PHI.

Advanced RISC Architectures

## Course Overview

This course introduces students to the fundamentals of processor design with a focus on instruction pipelining, memory organization, and performance optimization techniques. Emphasis is placed on the RISC-V Instruction Set Architecture (ISA), enabling students to understand and implement efficient IC designs. The course also explores pipeline hazards, cache optimization strategies, and real-world design challenges in modern processors. Students will gain hands-on experience by developing applications using the indigenous VEGA THEJAS32 microcontroller, bridging theoretical knowledge with practical system-level design and embedded programming.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Apply the Instruction pipeline concept in IC design
2. Analyze the Hazards and Performance issues in Pipelining
3. Evaluate the Optimization techniques in Cache memory
4. Create IC designs using RISC-V Instruction set Architecture
5. Create an application using VEGA THEJAS32 Microcontroller

**Syllabus**

**Unit I:**
Fundamental techniques of computer design: RISC and CISC architectures – Computer arithmetic – Comparison of RISC and CISC architectures. Verilog: Introduction and review of basic designs using Verilog. MIPS processor: Introduction to MIPS features – MIPS instruction set – Logical design of MIPS data path – Control unit and instruction decoder.

**Unit II:**
**Processor Pipelining:** Basics of Pipelining, Classic five stage pipelining in RISC processor, Performance issues in pipelining, Pipeline Hazards (Structural hazards, Data Hazards, Control Hazards), Data forwarding and bypassing techniques, Branch prediction technique: Static and Dynamic branch prediction.

**Unit III:**
**Memory hierarchy:** Memory hierarchy, Locality of References, Cache memory principles, Types of caches (Virtual and Physical cache), Cache architecture, Direct mapped, set associative and fully associative caches, Block Replacement Techniques and Write Strategy, Design Concepts in Cache Memory. Basic and Advanced Optimization Techniques in Cache Memory.

**Unit IV:**
**RISC-V Architecture:** RISC-V Instruction Set Architecture, Registers – General Purpose Registers, Control and Status Registers, Operating Modes, Programmers' Model for Base Integer ISA, Base Instruction Formats, Exceptions, Traps, and Interrupts, Machine-Level CSRs misa, mhartid, mstatus, mtvec medeleg and mideleg, mip and mie, mepc, mcause, mtval.

**Unit V:**
**VEGA THEJAS32 Microcontroller:** Functional Bock diagram, CPU, Memory Mapped input output and Interrupts Project using ARIES Development board.

TEXTBOOKS:

1. Computer Architecture, A quantitative approach by John L Hennessy and David APatterson Fifth Edition.
2. Computer organization and architecture, designing for performance, William Stallings Eight Edition.

REFERENCEBOOKS:

1. Georg Hager, Gerhard Wellein, Introduction to High Performance Computing for Scientists and Engineers, Chapman & Hall / CRC Computational Science series,2011.
2. The RISC-V Reader by David A Patterson and Andrew Waterman First Edition.
3. The RISC-V Instruction Set Manual Volume I: User-Level ISA Version 2.1

## Advanced Computer Architectures

### Course Overview

This course delves into the advanced concepts of computer system design and architecture. It begins with the fundamentals of computer design principles and progresses to explore instruction-level and thread-level parallelism. Key topics include pipeline processing, superscalar architectures, and performance optimization techniques. The course also addresses the design challenges and solutions related to interconnection networks in parallel computing environments. By the end of the course, students will gain a strong understanding of high-performance architecture design and the ability to analyze and evaluate complex computing systems.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Familiarize the instruction set, memory addressing of Computer
2. Handle the issues in pipelining and parallelism
3. Familiarize the practical issues in inter network

Syllabus

**Unit I:**

**Fundamentals of Computer Design:** Fundamentals of Computer design, Changing faces of computing and task of computer designer, Technology trends, Cost price and their trends, measuring and reporting performance, quantitative principles of computer design, Amdahl's law. Instruction set principles and examples- Introduction, classifying instruction set- memory addressing- type and size of operands, operations in the instruction set.

**Unit II:**

**Pipelines:** Introduction, basic RISC instruction set, Simple implementation of RISC instruction set, Classic five stage pipe line for RISC processor, Basic performance issues in pipelining, Pipeline hazards, Reducing pipeline branch penalties.

**Memory Hierarchy Design:** Introduction, review of ABC of cache, Cache performance, Reducing cache miss penalty, Virtual memory.

**Unit III:**

**Instruction Level Parallelism the Hardware Approach:** Instruction-Level parallelism, Dynamic scheduling, Dynamic scheduling using Tomasulo's approach, Branch prediction, high performance instruction delivery-hardware based speculation.

**ILP Software Approach:** Basic compiler level techniques, static branch prediction, VLIW approach, Exploiting ILP, Parallelism at compile time, Cross cutting issues -Hardware verses Software.

**Unit IV:**

**Multi Processors and Thread Level Parallelism:** Multi Processors and Thread level Parallelism- Introduction, Characteristics of application domain, Systematic shared memory architecture, Distributed shared – memory architecture, Synchronization.

**Unit V:**

**Inter Connection and Networks:** Introduction, Interconnection network media, Practical issues in interconnecting networks, Examples of inter connection, Cluster, Designing of clusters.

**Intel Architecture:** Intel IA- 64 ILP in embedded and mobile markets Fallacies and pit falls.

**TEXTBOOKS:**

1. John L. Hennessy, David A. Patterson, "Computer Architecture: A Quantitative Approach", 3rd Edition, Elsevier.

**REFERENCEBOOKS:**

1. JohnP.Shenand MiikkoH.Lipasti,"Modern Processor Design: Fundamentals of Super Scalar Processors", 2002, Beta Edition, McGraw-Hill
2. KaiHwang, FayeA.Brigs.,"Computer Architecture and Parallel Processing", McGrawHill.
3. DezsoSima, Terence Fountain, Peter Kacsuk, "Advanced Computer Architecture - A Design Space Approach", Pearson Education.

# GPU Architectures

**Course Overview**

This course introduces students to the principles and practices of GPU computing, with a focus on CUDA programming and parallel algorithm design. It covers the architectural foundations of modern GPUs, emphasizing how they differ from traditional CPUs in terms of memory hierarchy, thread scheduling, and computational throughput. Students will gain hands-on experience in writing and optimizing GPU code, debugging parallel programs, and applying GPU-specific optimization strategies such as memory coalescing and shared memory utilization. The course also includes practical exposure to applying GPU acceleration in areas such as scientific computing and deep learning, particularly on embedded platforms.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Understand working proficiency with CUDA, algorithmic GPU programming and parallel computing
2. Comprehend with classic scientific computing algorithms and problems
3. Optimize GPU code and debug GPU code
4. Analyze architecture specific details like memory access coalescing, shared memory usage, GPU thread scheduling
5. Apply deep learning algorithms on embedded GPUs

## Syllabus

**Unit I:**

**GPU architectures:** Introduction to the ideas of parallelism and the GPU programming model CPU vs GPU Parallelizing algorithms on paper, First CUDA program.

**Unit II:**

**CUDA programming:** Hardware of Graphics Processing Units and parallel communication patterns, Brief on GPU architecture, Basics of CUDA C, Floating point precision and support on GPUs.

**Unit III:**

**Parallel primitives and algorithms on GPU:** The CUDA programming language will be mastered while learning how to implement these algorithms., Matrix Operations, Stencil – Image Blurring, Filters, Gauss Jacobi-Finite difference updates for PDEs, Histogram, binning 1, Reduce – Maximum and Minimum – Summation, Prefix-sum (Scan) Algorithm – Radix Sort, Generating Cummulative Distributions, Complex algorithms – N-body solutions.

**Unit IV:**

**Optimizing GPU Applications:** Coalesced Memory Transactions, Grid Blocks, Thread Blocks, domain decomposition, Asynchronous Kernels and Multi streaming Possible Items: Libraries on GPU, cuBLAS Thrust, cuFFT, cuRAND, Multi-node GPU processing, Multi-GPU per node processing, CUDA in other languages (Python/Fortran), Scaling.

**Unit V:**

**Deep learning on GPUs:** Deep learning on GPUs, Combining graphics and compute, Display the results of computations– Interactive systems, Collision detection with voxelized solid (Gargoyle), Ray tracing in CUDA kernels, or ray tracing cores, Microsoft DXR (DX12 API), Vulkan, NVIDIA
OptiX / RTX, NVIDIA Turing: “World‘s First Ray Tracing GPU”- Quadro RTX, Geforce RTX

**Reference Books**

1. GPUs for Graphics: OpenGL 4.0 Shading Language Cookbook, 2nd Edition
2. Jason Sanders, Edward Kandrot, CUDA by Example: An Introduction to General-Purpose GPU Programming, Publisher: Addison-Wesley Professional,2013, 3rd Edition.

## Machine Learning and Deep Learning

**Course Overview**

This course introduces the principles and practices of Machine Learning and Deep Learning, focusing on both theoretical understanding and practical implementation. It begins with the fundamentals of machine learning, including learning paradigms, model evaluation, and the challenges that have led to deep learning advances. The course then explores deep neural architectures, including feedforward networks, convolutional and recurrent networks, alongside critical concepts such as regularization and optimization. Finally, students are equipped with practical methodologies for building robust models and exposed to real-world applications across computer vision, speech recognition, and natural language processing.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Analyse machine learning algorithms and evaluate model performance using concepts like overfitting, bias-variance, and likelihood-based estimation.
2. Design deep feedforward neural networks using appropriate architectures, activation functions, and backpropagation techniques.
3. Apply regularization and optimization techniques to improve generalization and training efficiency of deep neural networks.
4. Implement convolutional and recurrent neural network models for processing spatial and sequential data.
5. Evaluate model performance and deployment strategies in real-world applications such as computer vision, NLP, and speech recognition.

**Syllabus**

**Unit-I:**

**Machine Learning Basics:** Learning algorithms, Capacity, overfitting and underfitting, estimators, bias and variance, maximum likelihood estimation, Bayesian statistics, Supervised and unsupervised learning algorithms, building a machine learning algorithm, challenges motivating deep learning.

**Unit-II:**

**Deep Feedforward Networks:** Gradient-based learning, hidden units, architecture design, backpropagation and other differentiation algorithms.

**Unit-III:**

**Regularization for Deep Learning:** Norm penalties, Dataset augmentation, multi-task learning, early stopping, sparse representations, ensemble methods, dropout. Optimization: Optimization for Training Deep Models, challenges in neural network optimization, basic algorithms, parameter initialization strategies, algorithms with adaptive learning rates.

**Unit-IV:**

**Convolutional Neural Networks:** The convolution operation, motivation, pooling, convolution and pooling as an infinitely strong prior, variants of the basic convolution function, structures outputs, data types, efficient convolution algorithms, random or unsupervised features.

**Sequence Modeling:**

Recurrent and Recursive Nets, Recurrent Neural Networks, Recursive Neural Networks, Long ShortTerm Memory, optimization for long-term dependencies.

**Unit V:**

**Practical Methodology:** Performance metrics, selecting hyperparameters, debugging strategies. Applications: Large-scale deep learning, computer vision, speech recognition, natural language processing, other applications.

**Text Book**

1. Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016.

**Reference Books**

1. Géron, Aurélien. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 2nd ed., O'Reilly Media, 2019.
2. Bishop, Christopher M. Pattern Recognition and Machine Learning. Springer, 2006.
3. Chollet, François. Deep Learning with Python. 2nd ed., Manning Publications, 2021.

# Edge Computing

**Course Overview**

This course introduces the fundamentals of Edge Computing and its role in real-time, low-latency applications. It explores the shift from cloud to edge, emphasizing scenarios like autonomous vehicles and predictive maintenance. Students will learn key technologies such as Docker, Kubernetes, MQTT, and distributed analytics on edge data centers. The course includes hands-on exposure to frameworks for edge-cloud collaboration and time-critical data processing. It also covers AI, machine learning, and deep learning applications tailored for edge environments.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Understand the fundamentals of edge computing, its architecture, and its role in low-latency, real-time applications
2. Analyze and implement distributed data analytics techniques and edge-cloud frameworks for intelligent data processing
3. Gain hands-on experience with edge computing technologies such as Docker, Kubernetes, MQTT, Kafka, and time synchronization
4. Apply machine learning and deep learning models in edge scenarios such as autonomous systems, predictive maintenance, and reinforcement learning

**Syllabus**

**Unit I**

Introduction to Cloud and its limitations to support low latency and RTT. From Cloud to Edge computing: Waves of innovation, Introduction to Edge Computing Architectures

**Unit II:**

Edge Computing to support User Applications (5G-Slicing, self-driving cars and more). Concepts of distributed systems in edge computing such as time ordering and clock synchronization, distributed snapshot, etc.

**Unit III:**

Introduction to Edge Data Center, Lightweight Edge Clouds and its services provided by different service providers. Introduction to Docker containers and Kubernetes in edge computing. Design of edge storage systems like key-value stores

**Unit IV:**

Introduction to MQTT and Kafka for an end-to-end edge pipeline. Edge analytics topologies for M2M and WSN network (MQTT).

**Unit V:**

Use cases of machine learning for edge sensor data in predictive maintenance, image classifiers and self-driving cars. Deep Learning On-Device inference at the edge to support latency-based applications.

**Textbook& References**

1. Cloud Computing: Principles and Paradigms", Editors: Rajkumar Buyya, James Broberg, Andrzej M. Goscinski, Wiley, 2011
2. "Fog and Edge Computing: Principles and Paradigms", Rajkumar Buyya (Editor), Satish Narayana Srirama (Editor), Wiley, 2019.
3. "Cloud and Distributed Computing: Algorithms and Systems", Rajiv Misra, Yashwant Patel, Wiley 2020.

## Physical Design Automation with AI

**Course Overview**

This course introduces students to the fundamentals of VLSI physical design and explores how Artificial Intelligence (AI) is transforming Electronic Design Automation (EDA). Students will gain a comprehensive understanding of the VLSI design flow, physical design stages, partitioning algorithms, placement, routing, clock tree synthesis, and timing closure. The course also introduces the role of machine learning in optimizing physical design processes, covering key applications and challenges. Emphasis is placed on design rules, optimization techniques, and algorithmic approaches essential for real-world chip design.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Explain the physical design flow in VLSI and apply basic graph algorithms used in EDA tools.
2. Demonstrate understanding of chip planning techniques including floor planning, placement, and routing.
3. Analyze and implement routing strategies and clock tree synthesis for efficient signal distribution.
4. Evaluate timing closure techniques to ensure performance and reliability in VLSI designs.
5. Identify and discuss the applications and limitations of machine learning in physical design automation.

**Syllabus**

**Unit I:**

**Introduction to Physical Design and AI in EDA:** Introduction - Electronic Design Automation (EDA), VLSI Design Flow, VLSI Design Styles, Layout Layers and Design Rules, Physical Design optimizations, Algorithms and Complexity, Graph Theory Terminology, Common EDA Terminology.
Netlist and System Partitioning– Introduction, KL and FM Algorithms, Multilevel Partitioning.

**Unit-II:**

**Chip Planning and Placement:** Introduction, optimization goals, floorplan representations, floor planning algorithms, pin assignment, power and ground routing. Global placement, legalization and detailed placement.

**Unit-III:**

**Routing and Clock Tree Synthesis:** Introduction, optimization goals, representations of routing regions, the global routing flow, single-net and full-net routing, detailed routing – horizontal and vertical constraint graphs, channel and switchbox routing, area routing, clock routing, modern clock tree synthesis.

**Unit – IV:**

**Timing Closure:** Timing analysis, and performance constraints, timing-driven placement, timing driven routing, physical synthesis, performance-driven design flow.

**Unit-V:**

**Machine Learning in Physical Design:** Introduction ML promises and challenges in physical design, Canonical ML applications, the state of ML for physical design.

**Text Books:**

1. Kahng, Andrew B., et al. VLSI Physical Design: From Graph Partitioning to Timing Closure. 2nd ed., Springer International Publishing, 2022.
2. Sherwani, Naveed A. Algorithms for VLSI Physical Design Automation. 3rd ed., Springer, 2012.

**Reference Books:**

1. Hsiao, Michael. Machine Learning Applications in Electronic Design Automation. Morgan & Claypool, 2020.
2. Cong, Jason, and Bei Yu. Machine Learning and AI for EDA: From Fundamentals to Advanced Applications. Springer, 2022.

I M.Tech. I Semester

## Embedded System Design Laboratory

**Course Overview**

The Embedded System Design Laboratory focuses on the practical aspects of designing, developing, and testing embedded systems using real-time hardware platforms. Through a series of hands-on experiments, students will learn to configure embedded devices, interface sensors and actuators, perform GPIO programming, and implement embedded applications for automation and control. The course also explores advanced topics such as wireless communication, web hosting, media streaming, and custom firmware porting. Emphasis is placed on understanding system-level integration, debugging, and real-world deployment of embedded solutions.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Set up embedded development environments by flashing operating systems and configuring device interfaces.
2. Interface and program embedded peripherals such as sensors, actuators, and displays using GPIO.
3. Design and implement embedded applications for monitoring, automation, and communication.
4. Deploy embedded systems with network capabilities for web hosting, media transmission, and real-time data access.

**List of Experiments:**

1. **Functional Testing of Devices:**
   Flashing the OS onto the device in to a stable functional state by porting desktop environment with necessary packages.
2. **Exporting Display On To Other Systems:** Making use of available laptop/desktop displays as a display for the device using SSHclient &X11 display server.
3. **GPIO Programming**: Programming of available GPIO pins of the corresponding device using native programming language. Interfacing of I/O devices like LED/Switch etc., and testing the functionality.
4. **InterfacingChronoseZ430:** Chronos device is a programmable texas instruments watch which can be used for multiple purposes like PPT control, Mouse operations etc., Exploit the features of the device by interfacing with devices.
5. **ON/OFF Control Based on Light Intensity:** Using the light sensors, monitor the surrounding light intensity & automatically turn ON/OFF the high intensity LED's by taking some pre-defined threshold light intensity value.
6. **Battery Voltage Range Indicator:**
   Monitor the voltage level of the battery and indicating the same using multiple LED's (forex: for3 V battery and 3 led's, turn on 3 led's for 2-3V, 2 led's for 1-2V, 1 led for 0.1-1V & turn off all for 0V).
7. **Dice Game Simulation:** Instead of using the conventional dice, generate a random value similar to dice value and display the same using a 16X2 LCD. A possible extension could be to provide the user with option of selecting single or double dice game.
8. **Displaying RSS News Feed on Display Interface:** Displaying the RSS news feed headlines on a LCD display connected to device. This can be
   Adapted to other websites like twitter or other information websites. Python can be used to acquire data from the internet.
9. **Porting Open wrt To the Device:** Attempt to use the device while connecting to a wifi network using a USB dongle and at the same time providing a wireless access point to the dongle.
10. **Hosting a website on Board**:
    Building and hosting a simple website (static/dynamic) on the device and make it accessible online. There is a need to install server (eg: Apache) and thereby host the website.

11. **Web cam Server:**
    Interfacingtheregularusbwebcamwiththedeviceandturnitintofullyfunctional IPwebcam&tes t the functionality.
12. **FM Transmission**: Transforming the device into a regular fm transmitter capable of transmitting audio at desired frequency (generally 88-108 Mhz).

# CMOS VLSI Design Laboratory

**Course Overview**

This laboratory course offers hands-on experience in the design and analysis of CMOS-based digital and analog circuits using industry-standard simulation tools. Students will simulate and evaluate various combinational and sequential logic circuits, as well as model arithmetic and logic units. The course also includes the simulation of analog components like amplifiers and memory elements, reinforcing theoretical concepts through practical experimentation. Emphasis is placed on using EDA tools to analyze circuit behavior and performance, and students will learn to document their findings effectively through detailed lab reports. This lab builds essential skills for VLSI design and verification in modern semiconductor design workflows.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Analyze various combinational and sequential logic circuits using simulation tools.
2. Model arithmetic logic circuits using simulation tools.
3. Analyze various amplifiers using simulation tools.
4. Simulate memories using simulation tools.
5. Make an effective report based on experiments

**List of Experiments:**

1. Design and simulate a CMOS inverter2
2. Design and simulate CMOS Basic Gates
3. Design and simulate CMOS Adders
4. Design and simulate pass transistor logic
5. Design and simulate multiplexers
6. Design and simulate Latch
7. Design and simulate amplifiers
8. Design and simulate SRAM
9. Design and simulate Sampling switch

I M.Tech. I Semester

## Research Methodology and IPR

**Course Overview**

This course provides foundational knowledge and practical tools for conducting systematic research, analyzing data, and understanding intellectual property rights. Students will explore research design, data collection techniques, statistical analysis, and ethical considerations in scientific inquiry. In parallel, the course introduces IPR concepts, patent filing procedures, and their role in fostering innovation and protecting intellectual contributions. Emphasis is placed on developing the skills required for rigorous academic research and responsible innovation in science and engineering.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Formulate structured research problems using appropriate methods and review strategies.
2. Design and implement effective research methodologies for data collection and analysis.
3. Apply statistical tools to analyze research data and interpret results in context.
4. Evaluate types of intellectual property and their significance in protecting innovations.
5. Demonstrate ethical practices in research, publication, and intellectual property management.

### Syllabus

**Unit I:**

**Introduction to Research Methodology:** Meaning and objectives of research, Types of research: basic, applied, qualitative, quantitative, and mixed methods, Research process and design, Formulation of research problem, Literature survey and review techniques

**Unit II:**

**Research Design and Data Collection Methods:** Research design types: descriptive, experimental, exploratory, Sampling methods and techniques, Primary and secondary data sources, Data collection tools: surveys, interviews, observations, Validity and reliability of data

**Unit III:**

**Data Analysis and Interpretation:** Statistical tools and software, Descriptive and inferential statistics, Hypothesis testing and ANOVA, Regression and correlation analysis, Interpretation and presentation of data, Report writing and referencing styles

**Unit IV:**

**Intellectual Property Rights (IPR):** Overview and importance of IPR, Types of IPR: patents, trademarks, copyrights, industrial designs, Trade secrets and geographical indications, IPR and technology transfer, Role of IPR in research and innovation

**Unit V:**

**Patent Filing and Research Ethics:** Patent filing process: national and international (PCT), Patent search and analysis tools, Research ethics and scientific misconduct, Plagiarism and tools for plagiarism detection, Ethical issues in publication and authorship

**Text Books**

1. Kothari, C. R., and Gaurav Garg. Research Methodology: Methods and Techniques. 4th ed., New Age International, 2019.
2. Kumar, Ranjit. Research Methodology: A Step-by-Step Guide for Beginners. 5th ed., Sage Publications, 2021.

**Reference Books**

1. Subbaram, N.R.Hand book of Intellectual Property Law. Commercial Law Publishers, 2022.
2. Das,Resnik. The Ethics of Science: AnIntroduction.Routledge,1998.
3. WIPO.WIPO Intellectual Property Hand book: Policy, Lawand Use. World Intellectual Property Organization, 2020.

## Embedded Real Time Operating Systems

**Course Overview**

This course provides a comprehensive introduction to Real-Time Operating Systems (RTOS) and their applications in embedded systems. Students will explore the fundamental concepts of real-time embedded systems, understand task management, synchronization mechanisms like semaphores and message queues, and delve into exception handling and timer services. The course also covers memory management strategies and practical techniques for modularizing applications to meet real-time constraints. Additionally, it addresses common design challenges such as deadlocks and priority inversion, preparing students to design, analyse, and implement robust real-time embedded applications.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Analyse key concepts and roles of real-time operating systems in embedded system design.
2. Evaluate task scheduling and synchronization methods in real-time systems.
3. Design interrupt and timer handling mechanisms for embedded real-time environments.
4. Apply real-time memory management techniques in embedded applications.
5. Create modular real-time applications to resolve deadlocks and priority inversion.

**Syllabus**

**Unit-I:**

**Introduction to RTOS:** Real life examples of embedded systems, real-time embedded systems, the future of embedded systems, a brief history of operating systems, defining an RTOS, the scheduler, objects, services, key characteristics of an RTOS.

**Unit-II:**

**Tasks:** Introduction, defining a task, task states and scheduling, typical task operations, typical task structure, synchronization, communication, and concurrency.

Semaphores: Introduction, defining semaphores, typical semaphore operations, typical semaphore use.

**Message Queues:** Introduction, defining message queues, message queue states, message queue content, message queue storage, typical message queue operations, and typical message queue use.

**Unit-III:**

**Exceptions and Interrupts:** Introduction, what are exceptions and interrupts, a closer look at exceptions and interrupts, processing general exceptions, the nature of spurious interrupts.

**Timer and Timer Services:** Introduction, real-time clocks and system clocks, programmable interval timers, timer interrupt service routines, a model for implementing the soft-timer handling facility, timing wheels, soft timers and timer related operations.

**Unit-IV:**

**Memory Management:** Introduction, dynamic memory allocation in embedded systems, fixed-size memory management in embedded systems, blocking vs. non-blocking memory functions, hardware memory management units.

**Unit-V:**

**Modularizing Applications:** Introduction, an outside-in approach to decomposing applications, guidelines and recommendations for identifying concurrency, schedulability analysis – rate monotonic analysis.

Common Design Problems: Introduction, resource classification, deadlocks, priority inversion.

**Text Book**

1. Qing Li, and Caroline Yao. Real-Time Concepts for Embedded Systems. CMP Books, 2003.

**Reference Books**

1. Prasad, K. V. K. K. Embedded Real-Time Systems: Concepts, Design & Programming. Dream Tech Press, 2005.
2. Simon, David E. An Embedded Software Primer. 1st ed., 5th impression, Addison-Wesley Professional, 2007.
3. Singh, Rajib Mall. Real-Time Systems: Theory and Practice. Pearson Education India, 2006.

# IoT System Design

## Course Overview

This course introduces the foundational principles and system-level design of the Internet of Things (IoT). It covers the basic concepts, definitions, and significance of IoT in modern connected systems. Students will learn about the interaction between IoT and Machine-to-Machine (M2M) communication, and how they integrate to form intelligent networks. The course provides a detailed understanding of IoT architecture, including device-level design, network protocols, and data management. By the end of the course, students will have a comprehensive view of how to design and develop scalable, interoperable, and secure IoT systems.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Integrate the sensors and actuator depending on the applications
2. Interface the IoT and M2M with value chains
3. Write Python programming for Arduino, Raspberry Pi devices
4. Design IoT based systems such as Agricultural IoT, Vehicular IoT etc.,

Syllabus

**Unit I:**

**IoT introduction:** Introduction and definition of IoT, Evolution of IoT, IoT growth, Application areas of IoT, Characteristics of IoT, IoT stack, Enabling technologies, IoT levels, IoT sensing and actuation, Sensing types, Actuator types.

**Unit II:**

**IoT and M2M:** M2M to IoT – A Basic Perspective– Introduction, Differences and similarities between M2M and IoT, SDN and NFV for IoT.M2M Value Chains, IoT Value Chains, An emerging industrial structure for IoT, The international driven global value chain and global information monopolies.

**Unit III:**

**IoT Hands-on:** Introduction to Arduino Programming, Integration of Sensors and Actuators with Arduino. Introduction to Python programming, Introduction to Raspberry Pi, Interfacing Raspberry Pi with basic peripherals, Implementation of IoT with Raspberry Pi.

**Unit IV:**

**IoT Architecture:** IoT Architecture components, Comparing IoT architectures, A simplified IoT architecture, The core IoT functional stack, IoT data management and compute stack.

**Unit V:**

**IoT System design:** Challenges associated with IoT, Emerging pillors of IoT, Agricultural IoT, Vehicular IoT, Healthcare IoT, Smart cities, Transportation and logistics.

**Text Book**

1. Sudip Misra, Anandarup Mukherjee, Arijit Roy “Introduction to IOT”, Cambridge University Press.
2. David Hanes, Gonzalo salgueiro, Patrick Grossetete, Rob barton, Jerome henry “IoT Fundamentals Networking technologies, protocols, and use cases for IoT”, Cisco Press

**Reference Books**

3. Cuno pfister, “Getting started with the internet of things”, O Reilly Media, 2011
4. Francis daCosta, “Rethinking the Internet of Things: A Scalable Approach to Connecting Everything”, 1 st Edition, Apress Publications.
5. “Internet of Things concepts and applications”, Wiley
6. Arshdeep Bahga,Vijay Madisetti “Internet of Things A Hands on approach”, Universities Press
7. Shriram K Vasudevan, RMD Sundaram, Abhishek S Nagarajan, “Internet of things” John Wiley and Sons.
8. Massimo Banzi, Michael Shiloh Make: Getting Started with the Arduino, Shroff Publisher/Maker Media Publishers.

I M.Tech. II Semester

## Scripting Languages for Design Automation

**Course Overview**

This course introduces the use of scripting languages—PERL, TCL, and Python—in the context of Electronic Design Automation (EDA) workflows. Students will learn to write efficient automation scripts for parsing tool outputs, managing design files, and controlling EDA tools such as synthesis and simulation environments. The course emphasizes practical application, covering syntax, control structures, file I/O, regular expressions, and data processing. By the end of the course, learners will be able to automate common VLSI design tasks, improve productivity, and select the appropriate scripting language based on the use case.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Describe the role of scripting languages in EDA workflows and differentiate between scripting and compiled languages.
2. Write PERL scripts to perform text parsing, data extraction, and report generation.
3. Develop TCL scripts to automate tool commands and control design processes in EDA tools.
4. Construct Python scripts for file handling, data processing, and automation of design-related tasks.
5. Compare the features of PERL, TCL, and Python, and select suitable scripting languages for specific EDA applications.

**Syllabus**

**Unit I:**

**Introduction to Scripting Languages in EDA:** Introduction to scripting and automation, Scripting vs compiled languages, Using interpreters and writing first scripts in Perl, Tcl, and Python, Command line execution, Variable types and assignments (overview), Control flow basics (if, loops – overview), Basic file I/O (overview), Importance of scripting in EDA tools and flows.

**Unit II:**

**PERL Scripting:** Scalar data, Arrays and list data, Hashes, Input and output, Control structures, Regular expressions, Pattern matching with regex, Substitution and translation, Using files and file handles, String manipulation, Subroutines, Using Perl modules, Command-line arguments and environment variables, Text parsing examples, Report generation

**Unit III:**

**TCL Scripting:** Tcl syntax and structure, Variables and data types, Lists and arrays, Expressions and operators, Control flow (if, switch, while, for, foreach), Procedures and variable scope, File input and output, String and list manipulation, Error handling, Working with commands and arguments, Tool specific scripting conventions, Example tool scripts for synthesis and simulation

**Unit IV:**

**Python for Design Automation and Data Processing:** Python basics and data types, Expressions and operators, Flow control (if, for, while), Functions, Lists and dictionaries, String manipulation, File reading and writing, Pattern matching with regular expressions, Working with CSV and JSON files, Automating the keyboard and mouse, Using os, sys, and subprocess modules, Writing utility scripts for automation

**Unit V:**

**Advanced Integration and Comparison:** Comparison of PERL, TCL, and Python features, Strengths and weaknesses in EDA use-cases, Best practices for automation and maintainability, Calling external commands and shell integration in all three languages, Script interoperability using intermediate files (CSV, JSON), Efficiency, readability, and debugging considerations, Choosing the right scripting language for given EDA tasks

**Text Books**

1. Schwartz, Randal L., brian d foy, and Tom Phoenix. Learning Perl. 7th ed., O'Reilly Media, 2016.
2. Welch, Brent B., Ken Jones, and Jeffrey Hobbs. Practical Programming in Tcl and Tk. 4th ed., Prentice Hall, 2003.

3. Sweigart, Al. Automate the Boring Stuff with Python: Practical Programming for Total Beginners. 2nd ed., No Starch Press, 2019.

**Reference Books**

1. Robbins, Arnold. Scripting the UNIX System: Using Bash, Perl, and More. Addison-Wesley, 2003.
2. Ousterhout, John K. Tcl and the Tk Toolkit. 2nd ed., Addison-Wesley Professional, 2009.
3. Python Software Foundation. Python Language Reference Manual. https://docs.python.org.

I M.Tech. II Semester

## VLSI Architectures for Digital Signal Processing

### Course Overview

This course provides a comprehensive study of Very Large-Scale Integration (VLSI) architectures tailored for Digital Signal Processing (DSP) systems. It explores the principles, techniques, and hardware implementations that support high-performance DSP applications in modern CMOS technologies. Topics include algorithmic representations, iteration bounds, pipelining, parallelism, unfolding, folding, and convolution techniques, as well as bit-level and redundant arithmetic architectures. The course also introduces the design and use of programmable DSP processors, with a focus on real-time, low-power, and high-throughput design requirements.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Identify typical DSP algorithms and analyze their computational requirements for efficient VLSI implementation.
2. Compute iteration bounds and apply retiming techniques to optimize DSP data flow graphs for performance.
3. Design pipelined and parallel FIR filters and utilize unfolding and folding methods for architectural transformations.
4. Implement fast convolution methods and develop efficient bit-level and redundant arithmetic units for DSP systems.
5. Evaluate and compare programmable DSP processor architectures for various application domains such as mobile and multimedia systems

**Syllabus**

**Unit I:**

**Introduction to Digital Signal Processing Systems:** Introduction, typical DSP algorithms, DSP application demands and scaled CMOS technologies.

**Iteration Bound:** data-flow graph representations, loop bound, and iteration bound, algorithms for computing iteration bound.

**Pipelining and Parallel Processing:** Introduction, pipelining of FIR digital filters, parallel processing.

**Retiming:** Definitions and properties, solving systems of inequalities, retiming techniques.

**Unit II:**

**Unfolding:** An algorithm for unfolding, properties and applications of unfolding. Folding: Folding transformation, register minimization techniques, folding of multirate systems

**Unit III:**

**Fast Convolution:** Introduction, Cook-Toom algorithm, Winograd algorithm, iterated convolution, cyclic convolution, design of fast convolution algorithm by inspection.

**Unit IV:**

**Bit-Level Arithmetic Architectures:** Introduction, parallel multipliers, bit-serial multipliers, bit-serial filter design and implementation. **Redundant Arithmetic:** Introduction, redundant number representations, carry-free radix-2 addition and subtraction, radix-2 hybrid redundant multiplication architectures, data format conversion, redundance to nonredundant converter.

**Unit V:**

**Programmable Digital Signal Processors:** Introduction, evolution of programmable digital signal processors, important features of DSP processors, DSP processors for mobile and wireless communications, processors for multimedia signal processing.

**Text Book**

1. Parhi, Keshab K. VLSI Digital Signal Processing Systems: Design and Implementation. Wiley, 1999.

**Reference Books**

1. V. K. Madisetti, VLSI Digital Signal Processors: An Introduction to Rapid Prototyping and Design Synthesis, IEEE Press, New York.
2. Meyer-Baese, Uwe. Digital Signal Processing with Field Programmable Gate Arrays. 4th ed., Springer, 2014.
3. S. Y. Kung, H. J. Whitehouse, VLSI and Modern Signal Processing, Prentice Hall.

I M.Tech. II Semester

## SoC Architectures

**Course Overview**

This course provides a comprehensive understanding of System-on-Chip (SoC) architectures, focusing on the design methodologies and architectural components essential for modern integrated systems. It begins with an in-depth study of processor architectures and various design approaches adopted in SoC development. The course then explores memory design challenges and solutions within SoC environments, emphasizing performance, power, and area trade-offs. Additionally, it covers interconnect and bus architectures, which are critical for efficient data communication between components within the SoC. By the end of the course, students will be equipped with the knowledge required to analyze and design scalable and efficient SoC systems for a wide range of embedded applications.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Design System on Chip different processors.
2. Acquire knowledge of different internal bus architectures.
3. Understand AES algorithm and JPEG compression implementation in SoC.
4. know how the system forms with the lot of component and has majority about system level interconnections.
5. Introduce hardware and software programmability verses performance.

**Syllabus**

**Unit I:**

**Introduction to The System Approach**

System Architecture, Components of the System, Hardware & Software, Processor Architectures, Memory and Addressing. System Level Interconnection, An Approach for SoC Design, System Architecture and Complexity.

**Unit II:**

**Processors**

Introduction, Processor Selection for SoC, Basic Concepts in Processor Architecture, Basic Concepts in Processor Micro Architecture, Basic Elements in Instruction Handling. Buffers: Minimizing Pipeline Delays, Branches, More Robust Processors, Vector Processors and Vector Instructions Extensions, VLIW Processors, Superscalar Processors.

**Unit III:**

**Memory Design for SoC**

Overview of SoC External Memory, Internal Memory, Size, Scratchpads and Cache Memory, Cache Organization, Cache Data, Write Policies, Strategies for Line Replacement at Miss Time, Types of Cache, Split – I, And D – Caches, Multilevel Caches, Virtual to Real Translation, SoC Memory System, Models of Simple Processor – Memory Interaction.

**Unit IV:**

**Interconnect Customization and Configuration**

Interconnect Architectures, Bus: Basic Architectures, SoC Standard Buses, Analytic Bus Models, Using the Bus model, Effects of Bus Transactions and Contention Time. SoC Customization: An Overview, Customizing Instruction Processor, Reconfiguration Technologies, Mapping Design onto Reconfigurable Devices, Instance-Specific Design, Customizable Soft Processor, Reconfiguration - Overhead Analysis and Trade-Off Analysis on Reconfigurable Parallelism.

**Unit V:**

**Application Studies / Case Studies:**

SoC Design approach, AES algorithms, Design and evaluation, Image compression – JPEG Compression.

**Text Books**

1. Computer System Design System-on-Chip, Michael J. Flynn and Wayne Luk, Wiely India Pvt. Ltd, 2011.
2. ARM System on Chip Architecture, Steve Furber, 2nd Edition, Addison Wesley Professional, 2000.

**Reference Books**

1. Design of System on a Chip: Devices and Components, Ricardo Reis, 1st Edition, Springer, 2004.
2. Co-Verification of Hardware and Software for ARM System on Chip Design (Embedded Technology), Jason Andrews, Newnes, 2005.
3. System on Chip Verification: Methodologies and Techniques, Prakash Rashinkar, Peter Paterson and Leena Singh L, Kluwer Academic Publishers, 2001.

I M.Tech. II Semester

## Wireless Sensor Networks

**Course Overview**

This course offers a thorough understanding of Wireless Sensor Networks (WSNs), focusing on their architecture, applications, and design challenges. Students will explore the fundamental concepts, including network structures, node architectures, and communication protocols. The course covers key topics such as routing and MAC protocols, data gathering, and data dissemination techniques essential for efficient sensor network operation. Emphasis is also placed on emerging technologies, hardware platforms, software tools, and design principles required for developing real-world WSN applications. By the end of the course, students will be equipped with the knowledge and skills to design, deploy, and manage wireless sensor networks across various domains.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Analyze and compare various architectures of Wireless Sensor Networks
2. Understand Design issues and challenges in wireless sensor networks
3. Analyze and compare various data gathering and data dissemination methods.
4. Design, Simulate and Compare the performance of various routing and MAC protocol

**Syllabus**

**Unit I:**

Introduction to Sensor Networks, unique constraints and challenges, Advantage of Sensor Networks, Applications of Sensor Networks, Types of wireless sensor networks.

**Unit II:**

Mobile Ad-hoc Networks (MANETs) and Wireless Sensor Networks, Enabling technologies for Wireless Sensor Networks. Issues and challenges in wireless sensor networks.

**Unit III:**

Routing protocols, MAC protocols: Classification of MAC Protocols, S-MAC Protocol, BMAC protocol, IEEE 802.15.4 standard and ZigBee.

**Unit IV:**

Dissemination protocol for large sensor network. Data dissemination, data gathering, and data fusion; Quality of a sensor network; Real-time traffic support and security protocols.

**Unit V:**

Design Principles for WSNs, Gateway Concepts Need for gateway, WSN to Internet Communication, and Internet to WSN Communication. Single-node architecture, Hardware components & design constraints, Operating systems and execution environments, introduction to TinyOS and nesC.

**TEXTBOOKS:**

1. Ad-Hoc Wireless Sensor Networks- C. Siva Ram Murthy,B. S. Manoj, Pearson
2. Principles of Wireless Networks – Kaveh Pah Laven and P. Krishna Murthy, 2002, PE

**REFERENCEBOOKS:**

1. Wireless Digital Communications – Kamilo Feher, 1999, PHI.
2. Wireless Communications-Andrea Goldsmith, 2005 Cambridge University Press.
3. Mobile Cellular Communication – Gottapu Sasibhushana Rao, Pearson Education, 2012.
4. Wireless Communication and Networking – William Stallings, 2003, PHI.

I M.Tech. II Semester

## Automotive Embedded Systems

**Course Overview**

This course offers an in-depth exploration of embedded systems in modern vehicles, covering functional domains, standardized architectures like AUTOSAR, and in-vehicle communication protocols such as CAN and FlexRay. Students will learn about intelligent vehicle technologies, model-based development, and software reuse in the automotive domain. Emphasis is placed on system reliability, safety, and the integration of emerging technologies. The course bridges theoretical foundations with practical applications, preparing students for advanced roles in automotive software and embedded systems engineering.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Understand and analyze the structure, requirements, and standardized architecture of in-vehicle embedded systems including AUTOSAR.
2. Explain the role of intelligent technologies and communication protocols in modern vehicles.
3. Explain and Analyze FlexRay and CAN protocols.
4. Apply principles of software product-line engineering and variability modeling.
5. Demonstrate an understanding of automotive architecture description languages and modelbased development techniques.

**Syllabus**

**Unit I**

**Vehicle Functional Domains and Their Requirements**: General Context, Functional Domains: Power Train Domain, Chassis, Domain, Body Domain, Multimedia, Telematic, and HMI, Active/Passive Safety, Diagnostic. Standardized Components, Models, and Processes: In-Vehicle Networks and Protocols, Operating Systems, Middleware, Architecture Description Languages for Automotive Applications. Certification Issue of Safety-Critical In-Vehicle Embedded Systems.

**Application of the AUTOSAR Standard: Motivation:** Shortcomings in Former Software Structures, Setting up AUTOSAR, Main Objectives of AUTOSAR, Working Methods in AUTOSAR. Mainstay of AUTOSAR-AUTOSAR Architecture: AUTOSAR Concept, Layered Software Architecture. Main Areas of AUTOSAR Standardization: BSW and RTE: BSW, BSW Conformance Classes, RTE. Main Areas of AUTOSAR Standardization-Methodology and Templates: Objectives of the Methodology, Description of the Methodology, AUTOSAR Models, Templates, and Exchange Formats, System Configuration, ECU Configuration, Implementation to Existing Development Processes and Tooling. AUTOSAR in Practice-Conformance Testing. AUTOSAR in Practice-Migration to AUTOSAR ECU. AUTOSAR in Practice-Application of OEM– Supplier Collaboration. AUTOSAR in Practice: Demonstration of AUTOSAR-Compliant ECUs:

Description of the Demonstrator, Concepts Shown by the Demonstrator.

**Unit II:**

**Intelligent Vehicle Technologies: Introduction**: Road Transport and Its Evolution: Such a Wonderful Product, Safety Problems, Congestion Problem, Energy and Emissions. New Technologies: Sensor Technologies, Sensor Fusion, Wireless Network Technologies, Intelligent Control Applications, Latest Driving Assistance. Dependability Issues: Introduction, Fail-Safe Automotive Transportation Systems, Intelligent Auto diagnostic. Fully Autonomous Car-Dream or Reality?: Automated Road Vehicles, Automated Road Network, Automated Road Management, Deployment Paths.

**Embedded Automotive Protocols:** Automotive Communication Systems-Characteristics and Constraints: From Point-to-Point to Multiplexed Communications, Car Domains and Their Evolution, Different Networks for Different Requirements, Event-Triggered versus Time-Triggered. In-Car Embedded Networks: Priority Buses, TT Networks, Low-Cost Automotive Networks, Multimedia Networks. Middleware Layer: Rationale for a Middleware, Automotive MWs Prior to AUTOSAR, AUTOSAR. Open Issues for Automotive Communication Systems: Optimized Networking Architectures, System Engineering.

**Unit III:**

**FlexRay Protocol: Introduction:** Event-Driven versus Time-Driven Communication, Objectives of FlexRay, History of FlexRay. FlexRay Communication: Frame Format, Communication Cycle, Static Segment, Dynamic

Segment. FlexRay Protocol: Protocol Architecture, Protocol Wakeup and Startup, Wakeup, Clock Synchronization, Fault-Tolerance Mechanisms. FlexRay Application: FlexRay Implementation, FlexRay Tool Support. Impact on Development, Verification of FlexRay.

**Dependable Automotive CAN Networks:** Introduction: Main Requirements of Automotive Networking, Networking Technologies, CAN Features and Limitations. Data Consistency Issues: Management of Transient Channel Faults in CAN, Impairments to Data Consistency, On the Probability of the Data Inconsistency Scenarios, Solutions to Really Achieve Data, Consistency over CAN. CANcentrate and ReCANcentrate-Star Topologies for CAN: Rationale, CANcentrate and

ReCANcentrate Basics, Other Considerations. CANEL: Clock Synchronization, Data Consistency, Error Containment, Support for Fault Tolerance, CANELy Limitations. FTT-CAN-Flexible TimeTriggered Communication on CAN: FTT System Architecture, Dual-Phase Elementary Cycle, SRDB, Main Temporal Parameters within the EC, Fault-Tolerance Features, Accessing the Communication Services. FlexCAN-A Deterministic, Flexible, and Dependable Architecture for Automotive Networks: Control System Transactions, FlexCAN Architecture, How FlexCAN Addresses CAN Limitations, FlexCAN Applications and Summary. Other Approaches to Dependability in CAN: TTCAN, Fault-Tolerant Time-Triggered Communication Using CAN, TCAN, ServerCAN, Fault-Tolerant Clock Synchronization Over CAN.

**Unit IV:**

**Product Lines in Automotive Electronics:** Characteristics of Automotive Product Lines: Basic Concepts of Software Product Lines, Characteristics and Needs of Automotive Electronics with Respect to Product-Line Engineering. Basic Terminolog: Software Product Lines, Variability, Feature Modeling as a Form of Variability Modeling, Discussion-Feature Modeling for the Automotive Domain. Global Coordination of Automotive Product-Line Variability: Coordination of Small- to Medium-Sized Product Lines, Coordination of Highly Complex Product Lines. ArtifactLevel Variability: Basic Approach, Difficulties Related to Artifact-Local Variability, Representing Variability in ECU, Requirements Specifications, Evaluation of Representations, Mapping Representations to a Common Basis.

**Reuse of Software in Automotive Electronics:** Reuse of Software-A Challenge for Automotive OEMs. Requirements for the Reuse of Software in the Automotive Domain. Supporting the Reuse of

Application Software Components in Cars: Processes, Development of Modularized Automotive Software Components, Function Repository, Development of an In-Vehicle Embedded System. Application Example.

**Unit V:**

**Automotive Architecture Description Languages:** Introduction. Engineering Information Challenges: Reducing Cost and Lead Time, Development Organization and Information Exchange, Product Complexity, Quality and Safety, Concurrent Engineering, Reuse and Product Line Architectures, Analysis and Synthesis, Prototyping. State of Practice: Model-Based Design, Tools, Problems beyond Model-Based Design. ADL as a Solution: General Aspects on an Automotive ADL, What Needs to Be Modeled. Existing ADL Approaches: Forsoft Automotive, SysML, Architecture and Analysis Description Language, Modeling and Analysis of Real-Time and Embedded Systems, AUTOSAR Modeling, EAST-ADL.

**Model-Based Development of Automotive Embedded Systems:**

Introduction: What Is MBD?Motivating MBD for Automotive Embedded Systems: Role of MBD in Automotive Embedded Systems Development, MBD Means, Driving Factors for MBD, Potential Benefits of MBD Approaches. Context, Concerns, and Requirements: Contextual Requirements on MBD, Product Concerns Addressed by MBD Efforts. MBD Technology: Modeling Languages: Abstractions, Relations, and Behavior, Analysis Techniques, Synthesis Techniques, Tools. State of the Art and Practice: Automotive State of Practices, Research and Related Standardization Efforts. Guidelines for Adopting MBD in Industry: Strategic Issues, Adopting MBD: Process and Organizational Considerations, Desired Properties of MBD Technologies, Common Arguments against MBD and Pitfalls.

**Text Book**

1. N. Navet and F. Simonot-Lion, Eds., Automotive Embedded Systems Handbook. Boca Raton, FL, USA: CRC Press, 2008.
2. William B Ribbens, Understanding Automotive Electronics: An Engineering Perspective, Seventh Edition, Elsevier, 2012.
3. Feiler, P. H., Gluch, D. P., & Hudak, J. J. (2012). Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language. Addison-Wesley/SAE International.

**Reference Books**

1. Ola Larses, Architecting and Modeling Automotive Embedded Systems, KTH, 2005
2. AUTOSAR. (2025). Standards of AUTOSAR. Retrieved from https://www.autosar.org/standards
3. Vlacic, L., Parent, M., & Harashima, F. (2001). Intelligent Vehicle Technologies: Theory and Applications. Elsevier Science.

I M.Tech. II Semester

## Hardware Software Co-Design

**Course Overview**

This course focuses on the integrated development of hardware and software components in embedded system design. It introduces students to key co-design issues, including system partitioning, prototyping, and emulation techniques. The course explores architecture-specific optimization strategies that balance performance, power, and flexibility. Additionally, students will gain practical knowledge of industry-standard tools used in hardware/software co-design. By the end of the course, learners will be able to effectively design, implement, and evaluate embedded systems through a unified hardware-software development approach.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Acquire the knowledge on various models of Co-design.
2. Explore the interrelationship between Hardware and software in a embedded system
3. Acquire the knowledge of firmware development process and tools during Co-design
4. Implement validation methods and adaptability.

**Syllabus**

**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 – Springer, 2009.

**Reference Books**

1. Hardware / Software Co- Design - Giovanni De Micheli, Mariagiovanna Sami, Kluwer Academic Publishers, 2002.
2. A Practical Introduction to Hardware/Software Co-design -Patrick R. Schaumont, Springer, 2010

I M.Tech. II Semester

## Embedded Real Time Operating Systems Laboratory

**Course Overview**

The Embedded RTOS Laboratory course is designed to provide hands-on experience and practical understanding of Real-Time Operating System (RTOS) concepts as applied in embedded systems. The course focuses on the simulation-based design and analysis of core operating system functionalities, enabling students to explore process management, inter-process communication (IPC), and I/O handling within an RTOS environment. Students will gain proficiency in using RTOS commands and system call interfaces essential for embedded application development. Through a series of structured laboratory experiments and mini-projects, the course also introduces key concepts in real-time embedded software design, task scheduling, synchronization, and resource management. By the end of the course, students will be equipped to develop and test real-time applications using industry-standard RTOS platforms.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Apply various real time concepts in building embedded systems
2. Implement the RTOS development tools in building real time embedded Systems
3. Appreciate the necessity of Inter Process Communication and Synchronization mechanisms
4. Apply the concept of RTOS in the designing of real time systems

**List of Experiments**

Develop program in Linux for the following:

1. Develop a program utilizing command line arguments (argc and argv)
2. Create new process using fork ( ).
3. Communicate between parent and child process using pipes.
4. Communicate between parent and child process using FIFOs
5. Develop a program to communicate between processes using message queue.
6. Develop a program using system calls that is similar to ‘cp’ command.
7. Create a new thread using POSIX Thread library
8. Develop a program to demonstrate the use of synchronizing access to shared resource using semaphores. (POSIX Thread based)
9. Develop a program to demonstrate the use of synchronizing access to shared resource using mutex. (POSIX Thread based)
10. Develop a program to demonstrate the use of signaling semaphore for sending event from one thread to another. (POSIX Thread based)

I M.Tech. II Semester

## IoT System Design Laboratory

**Course Overview**

This Internet of Things: Systems Design' course provides a structured series of lab sessions, each with defined objectives, aimed at reinforcing key IoT concepts through hands-on experience. While some foundational knowledge in networking and basic familiarity with programming (especially C/C++) is beneficial, the lab book accommodates learners of all levels by offering clear instructions, explanations, and external resources. The labs are intended to be completed within two hours and cover essential IoT practices without delving deeply into programming languages, encouraging students to independently explore and bridge any knowledge gaps.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Program microcontrollers and single-board computers for sensor and actuator interfacing.
2. Build and deploy complete IoT systems with cloud connectivity.
3. Use tools and protocols for IoT communication, including BLE, RFID, and NFC.
4. Simulate, debug, and enhance IoT applications using Raspberry Pi and related tools.

**List of Experiments**

1. Micro-Controller Programming
2. Single-board Computer Programming
3. Posting Data to an IoT Cloud Platform
4. Connecting an IoT Gateway to an IoT Cloud
5. Connecting a Sensor Node to IoT Gateway
6. End to End Full Stack IoT Development
7. Introduction to Wireshark on Raspberry Pi
8. Programming Arduino with Blockly
9. Programming Raspberry Pi with Python
10. Bluetooth Low Energy (BLE) Based Systems
11. RFID and NFC Based Tracking
12. Multimedia Communication
13. Microcontroller Programming Simulator
14. Advance Sensors, Actuators, Components
15. Getting Started with Raspberry Pi Camera
16. Debugging the Raspberry Pi

II M.Tech. I Semester

## Hardware Security in VLSI Design

**Course Overview**

This course provides an in-depth understanding of security challenges and vulnerabilities in modern VLSI systems, particularly within SoC and PCB designs. Students will learn about hardware threats such as Trojans, reverse engineering, IP piracy, side-channel attacks, test-oriented and physical attacks, and how to design secure hardware using primitives like PUFs and TRNGs. Through a mix of theoretical foundations and hands-on experimentation, students will be equipped to identify, analyse, and mitigate hardware security threats in real-world VLSI systems.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Identify and explain the key layers and components of VLSI systems vulnerable to hardware attacks.
2. Analyse different types of hardware Trojans and IP piracy techniques and suggest appropriate countermeasures.
3. Evaluate side-channel and test-oriented attacks on SoC and FPGA-based systems using suitable models.
4. Demonstrate an understanding of physical attacks and PCB vulnerabilities through attack modeling and analysis.
5. Design and implement hardware security primitives such as PUFs and TRNGs to protect VLSI designs from cloning and counterfeiting.

**Syllabus**

**Unit I**

**Introduction to Hardware Security:** Overview of a computing system, layers of a computing system, hardware security vs. hardware trust, attacks, vulnerabilities and countermeasures, conflict between security and test/debug, evolution of hardware security, overview of electronic hardware – nanoscale technologies, ASICs and FPGAs, printed circuit board, embedded systems, hardware-firmware software interaction.

**Unit II:**

**Hardware Trojans:** Introduction, SoC design flow, hardware trojans, hardware trojans in FPGA designs, hardware trojans taxonomy, trust benchmarks, countermeasures against hardware trojans, hardware trojan attacks. Hardware IP Piracy and Reverse Engineering: Introduction, hardware intellectual property (IP), security issues in IP-based SoC design, security issues in FPGA, reverse engineering and tampering.

**Unit III:**

**Side-Channel Attacks:** Introduction, taxonomy of side-channel attacks, uncommon side-channel attacks, power analysis attacks, electromagnetic (EM) side-channel attacks, fault-injection attacks, timing attacks.Test-Oriented Attacks: Introduction, scan-based attacks, JTAG-based attacks.

**Unit IV:**

**Physical Attacks and Countermeasures:** Introduction, reverse engineering, probing attack.Attacks on PCB: Security challenges, attack models, bus snooping attack.

**Unit-V:**

**Hardware Security Primitives:** Introduction, physical unclonable functions (PUFs), true random number generator (TRNG), design of anti-counterfeit, existing challenges and attacks. Security and Trust Assessment: Security assets and attack models, pre-silicon and post-silicon security and trust assessment.

**Text Book**

1. Bhunia, Swarup, and Mark M. Tehranipoor. Hardware Security: A Hands-on Learning Approach. 1st ed., Academic Press, 2019.

**Reference Books**

1. Tehranipoor, Mark, and Cliff Wang, editors. Introduction to Hardware Security and Trust. Springer, 2012.
2. Bhunia, Swarup, and Sandip Ray. Fundamentals of IP and SoC Security: Design, Verification, and Debug. Springer, 2017.
3. Wolf, Marilyn. Embedded System Interfacing: Design for the Internet-of-Things (IoT). Morgan Kaufmann, 2019.
4. Skorobogatov, Sergei. Semi-Invasive Attacks: A New Approach to Hardware Security Analysis. Springer, 2007.

**II M.Tech. I Semester**

## Software Defined Radio

**Course Overview**

This course offers a foundational understanding of Software Defined Radio (SDR), a transformative technology in modern wireless communication systems. It introduces the essential radio frequency (RF) components and demonstrates how they can be implemented and controlled through software. Students will learn about the software architectures that enable flexibility and reconfigurability in SDR platforms. The course also emphasizes hands-on experience in building and analyzing SDR systems, highlighting their adaptability and relevance in evolving communication standards and protocols. By the end of the course, students will be prepared to design, simulate, and evaluate SDR systems for current and next-generation wireless applications.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Learn the software radio and its importance.
2. Understand techniques to improve data conversion performance.
3. Understand the difference between ideal and practical converters.
4. Acquire knowledge of digital hardware architectures and its development methods.
5. Understand the object-oriented representation of radios and the analysis of various case studies.

**Syllabus**

**Unit I**

**Introduction:** The Need for Software Radios, What is Software Radio, Characteristics and Benefits of Software Radio-Design Principles of Software Radio, RF Implementation Issues, The Purpose of RF Front, End, Dynamic Range- The Principal Challenge of Receiver Design, RF Receiver Front,

End Topologies, Enhanced Flexibility of the RF Chain with Software Radios- Importance of the Components to Overall Performance-Transmitter Architectures and Their Issues, Noise and Distortion in the RF Chain, ADC and DAC Distortion.

**Unit II:**

**Multi Rate Signal Processing:** Introduction, Sample Rate Conversion Principles, Polyphase Filters, Digital Filter Banks, Timing Recovery in Digital Receivers using Multirate Digital Filters.

Digital Generation of Signals: Introduction, Comparison of Direct Digital Synthesis with Analog Signal Synthesis, Approaches to Direct Digital Synthesis, Analysis of Spurious Signals, Spurious Components due to Periodic Jitter- Band Pass Signal Generation, Performance of Direct Digital Synthesis Systems, Hybrid DDS-PLL Systems, Applications of Direct Digital Synthesis- Generation of Random Sequences, ROM Compression Techniques.

**Unit III:**

**Analog to Digital and Digital to Analog Conversion:** Parameters of Ideal Data Converters, Parameters of Practical Data Converters- Analog to Digital and Digital to Analog Conversion, Techniques to Improve Data Converter Performance- Common ADC and DAC Architectures

**Unit IV:**

**Digital Hardware Choices:** Introduction, Key Hardware Elements, DSP Processors, Field Programmable Gate Arrays, Trade-Offs in Using DSPs, FPGAs, and ASICs, Power Management Issues using a Combination of DSPs, FPGAs, and ASICs

**Unit V:**

**Object–Oriented Representation of Radios and Network Resources:** Networks- Object Oriented Programming- Object Brokers- Mobile Application Environments- Joint Tactical Radio System. Case Studies in Software Radio Design: Introduction and Historical Perspective, SPEAK Easy, JTRS, Wireless Information Transfer System, SDR-3000 Digital Transceiver Subsystem, Spectrum Ware, CHARIOT.

**Text Book**

1. Software Radio: A Modern Approach to Radio Engineering, Jeffrey H. Reed, PEA Publication, 2002.
2. Software Defined Radio: Enabling Technologies, Walter Tuttle Bee, Wiley Publications, 2002.

**Reference Books**

1. Software Defined Radio for 3G, Paul Burns, Artech House, 2002.
2. Software Defined Radio: Architectures, Systems and Functions, Markus Dillinger, Kambiz Madani and Nancy Alonistioti, Wiley, 2003.
3. Software Radio Architecture: Object Oriented Approaches to Wireless System Engineering, Joseph Mitola III, John Wiley & Sons, 2000.

II M.Tech. I Semester

## Embedded Biomedical Applications

**Course Overview**

This course provides an interdisciplinary introduction to the field of biomedical engineering with a focus on embedded system applications. It begins by exploring the fundamentals of biomedical engineering, followed by an in-depth study of wearable health monitoring devices and their embedded architectures. Students will gain insights into the hardware requirements for medical image processing and the role of embedded systems in diagnostic and therapeutic applications. The course also covers various assistive and support devices commonly used in clinical settings. By the end of the course, learners will be equipped with the foundational knowledge and practical skills to develop and analyze embedded solutions for modern biomedical challenges.

**Course Outcomes (COs)**

By the end of this course, students will be able to:

1. Demonstrate the fundamental art of biomedical engineering.
2. Illustrate about wearable health devices and its importance.
3. Implement image processing applications using software and hardware.
4. Compare various embedded diagnostic applications.
5. Build and analyze of some biomedical equipment.

**Syllabus**

**Unit I**

**INTRODUCTION TO BIOMEDICAL ENGINEERING**: Origin of bio potential and its propagation-Resting and Action Potential – Bio signals characteristics Types of electrodes - Types of transducers and applications-Bio-amplifiers- Types of recorders,components of a biomedical system.

**Unit II:**

**WEARABLE HEALTH DEVICES:** Concepts of wearable technology in health care-Components of wearable devices- Biosensors- Blood glucose sensors - Head worn- Hand worn- Body wornpulse oxymeter-Cardiac pacemakers – Hearing aids and its recent advancements-wearable artificial kidney.

**Unit III:**

**EMBEDDED SYSTEM FOR MEDICAL IMAGE PROCESSING:** Introduction to embedded image processing. ASIC vs FPGA - memory requirement-, power consumption- parallelism - Design issues in VLSI implementation of Image processing algorithms - interfacing. Hardware implementation of image processing algorithms: Segmentation and compression

**Unit IV:**

**EMBEDDED SYSTEM FOR DIAGNOSTIC APPLICATIONS:** ICCU patient monitoring system – ECG-EEG-EMG acquisition system-MRI scanner - CT scanner Sonography

**Unit V:**

**CASE STUDY:** Respiratory measurement using spirometer- IPPB unit for monitoring respiratory parameters - ventilators- -Defibrillator- Glucometer-Heart- Lung machine.

**Reference Books**

1. Leslie Cromwell, “Biomedical Instrumentation and Measurement”, Prentice Hall of India New Delhi, 2007.
2. John G.Webster, “Medical Instrumentation Application and Design”, 3rd Edition, Wiley India Edition, 2007.
3. Khandpur R.S, Handbook of Biomedical Instrumentation, Tata McGraw Hill, New Delhi, 3rd Edition, 2014.
4. L.A Geddes and L.E.Baker, Principles of Applied Biomedical Instrumentation, 3rd Edition John Wiley and Sons, Reprint 2008.
5. Richard S.Cobbold, Transducers for Biomedical Measurements; Principle and applications John Wiley and sons, 1992.