Chapter 3: Hardware System Architecture and Modeling

3.1 Introduction

System architecture defines the **high-level structure and behavior** of a hardware system.
● It involves **selecting components**, **defining their roles**, and **mapping interconnections**.
● System modeling complements architecture by offering **visualization, simulation, and validation** tools to support design decisions before physical implementation.

3.2 Importance of Architecture in Hardware Systems

- Provides a **blueprint** for engineers across disciplines
- Ensures **modular, scalable, and maintainable** designs
- Aligns system with **functional, performance, and interface** requirements
- Helps **identify bottlenecks** early in development

3.3 Architectural Design Methodologies

Methodology	Description
Top-Down Design	Start from system-level goals → break into subsystems
Bottom-Up Design	Begin with available components → integrate into systems
Meet-in-the-Middle	Parallel top-down and bottom-up approach

Model-Based Design (MBD)	Use modeling tools to define, simulate, and refine system architecture

---

3.4 System Architecture Components

Component	Role
Processing Unit	Microcontroller, microprocessor, DSP, or FPGA
Memory	Flash, SRAM, DRAM for code and data storage
Interfaces	Communication (I2C, SPI, UART, CAN, USB, Ethernet)
Sensors/Actuators	System interaction with the physical world
Power Management	Voltage regulators, power monitors, energy harvesting
Mechanical Housing	Form factor, thermal management, EMI shielding

---

3.5 Hardware Modeling Techniques

Model Type	Purpose
Functional Model	Represents system behavior (what it does)

Structural Model	Describes system components and interconnections
Behavioral Model	Captures how components respond to inputs over time
State Machine Model	Represents control logic and system modes
Timing Diagram	Illustrates timing relationships between signals

---

3.6 Tools for Architecture and Modeling

Tool	Use
Block Diagrams (Visio, Draw.io, Lucidchart)	High-level architecture
SysML (Systems Modeling Language)	UML extension for hardware systems
Simulink (MATLAB)	Model-based design for embedded systems
LTspice, Multisim	Analog/digital circuit simulation
VHDL/Verilog	Hardware description languages for FPGAs
Altium Designer, KiCad, OrCAD	Schematic and PCB design with architectural annotation

---

3.7 Example: Sensor-Based Embedded System

System Goal:

Design a temperature-monitoring IoT device

Architecture:

- **MCU**: ARM Cortex-M4
- **Sensor**: Digital temperature sensor via I2C
- **Power**: Li-ion battery + buck regulator
- **Communication**: Wi-Fi via UART
- **Modeling**:
  - **Block diagram** to define system scope
  - **State machine** for sleep/wake/transmit cycles
  - **Simulink** to model thermal response and power use

3.8 Performance Modeling and Simulation

Modeling helps simulate:

- **Power consumption** over time
- **Thermal profiles** and cooling requirements
- **Signal integrity** for high-speed buses
- **Timing margins** for real-time tasks

> Simulation improves **predictability and reliability** before committing to hardware.

3.9 Architectural Trade-offs

Consideration	Trade-off
Performance vs. Power	Faster processors consume more energy
Cost vs. Flexibility	FPGAs are reprogrammable but more expensive
Analog vs. Digital	Analog gives precision, digital gives programmability
Integration vs. Modularity	Integrated SoCs save space but reduce flexibility

---

3.10 Documentation Best Practices

Artifact	Purpose
System Block Diagram	High-level system view
Interface Control Document (ICD)	Defines signal and communication interfaces
Architecture Description Document (ADD)	Comprehensive architectural rationale
Functional Models/Simulations	Validate expected behaviors
Versioned Models and Diagrams	Maintain traceability and revisions

3.11 Summary of Key Concepts

● Hardware system architecture ensures the design is **structured, scalable, and aligned with system goals**.
● Modeling enables **early validation, simulation, and risk reduction**.
● Tools like **SysML, Simulink, and hardware simulators** support rigorous system-level thinking.
● Effective architecture balances **performance, power, cost, and modularity** across the system lifecycle.