ESD IA1

M1Q1) Difference between embedded and general computing systems (5 marks)

Feature	Embedded System	General Computing System
Definition	A system designed for a specific task or function.	A system designed for multiple applications.
Hardware	Specialized hardware optimized for the task.	General-purpose hardware with high flexibility.
Operating System	May or may not have an OS; often uses RTOS.	Runs a full-fledged OS like Windows or Linux.
User Programmability	Firmware is usually fixed or limited in modification.	Users can install and modify software easily.
Performance Goal	Optimized for efficiency, reliability, and real-time constraints.	Optimized for versatility and high processing power.
Power Consumption	Designed to be power-efficient.	Generally consumes more power.
Examples	Washing machines, ATMs, digital cameras, pacemakers.	Laptops, desktops, tablets, and smartphones.

M1Q2) Purpose of Embedded Systems? (5 marks)

Embedded systems are designed to perform specific tasks efficiently. The main purposes of embedded systems include:

1. **Data Collection, Storage, and Representation**
    - Embedded systems collect data from sensors and process/store it.
    - Example: Digital cameras store images, ECG machines record heart activity.
2. **Data Communication**
    - Used for transferring data between devices through wired or wireless communication.
    - Example: Routers, modems, and IoT devices.

3. **Data (Signal) Processing**
    - Processes analog or digital signals for applications like image, video, or audio processing.
    - Example: Digital hearing aids, MP3 players, speech recognition systems.
4. **Monitoring Systems**
    - Continuously track and analyze environmental or system parameters.
    - Example: Medical monitoring devices like heart rate monitors.
5. **Control Systems**
    - Embedded systems control external devices based on input conditions.
    - Example: Automatic washing machines, industrial robots, air conditioning systems.

M1Q3) Components of typical embedded system in detail (5 marks)

An embedded system consists of both **hardware** and **software (firmware)** components designed to perform a specific function. The main components are:

1. **Processor (Microcontroller or Microprocessor)**
    - The core of an embedded system, responsible for executing instructions.
    - **Microprocessor**: Requires external memory and peripherals (e.g., Intel processors).
    - **Microcontroller**: Contains CPU, memory, and peripherals on a single chip (e.g., 8051, AVR, ARM).
2. **Memory (ROM & RAM)**
    - **ROM (Read-Only Memory)**: Stores firmware/software permanently (e.g., Flash memory, EEPROM).
    - **RAM (Random Access Memory)**: Temporary storage for data during execution.
3. **Sensors and Actuators**

- **Sensors**: Detect physical parameters (e.g., temperature, pressure, motion).
- **Actuators**: Convert electrical signals into physical action (e.g., motors, relays).

4. **Communication Interfaces**
   - Allows the system to communicate with external devices.
   - Examples: UART, I2C, SPI, USB, Ethernet, CAN bus.
5. **Power Supply Unit**
   - Provides the required voltage and current to run the system.
   - Can be battery-powered (e.g., in wearables) or externally powered.
6. **Timers and Counters**
   - Used for scheduling tasks, generating delays, and measuring time intervals.
   - Example: Timer in a washing machine to control washing cycles.
7. **Embedded Firmware (Software)**
   - The program running on the embedded system, written in C, C++, or Assembly.
   - Controls the system's functionality and real-time operations.
8. **PCB (Printed Circuit Board) & Passive Components**
   - The circuit board connects all components.
   - Includes passive elements like resistors, capacitors, and inductors.

M1Q4) Operations of Zigbee and Wifi networks (5 marks)

Zigbee and Wi-Fi are both **wireless communication technologies** but are used for different applications.

1. Zigbee Network Operations

Zigbee is a **low-power, short-range wireless communication protocol** based on the **IEEE 802.15.4** standard.

- **Network Topology:** Supports **Star, Mesh, and Tree** topologies.
- **Data Transmission:** Uses **low data rate** (up to 250 kbps), ideal for sensor networks.
- **Power Consumption:** Extremely low, designed for battery-operated devices.
- **Range:** Up to 100 meters (can be extended in mesh networks).
- **Frequency Band:** Operates in **2.4 GHz, 915 MHz, and 868 MHz** ISM bands.
- **Applications:** Smart home automation, industrial monitoring, IoT devices, medical sensors.

2. Wi-Fi Network Operations

Wi-Fi is a **high-speed wireless communication technology** based on the **IEEE 802.11** standard.

- **Network Topology:** Uses **Infrastructure Mode** (via routers) or **Ad-Hoc Mode** (device-to-device).
- **Data Transmission:** High data rate (up to **9.6 Gbps** in Wi-Fi 6).
- **Power Consumption:** High, requires a continuous power source.
- **Range:** Typically **30–100 meters** indoors, **up to 300 meters** outdoors.
- **Frequency Band:** Operates on **2.4 GHz and 5 GHz** bands.
- **Applications:** Internet access, video streaming, online gaming, smart devices, IoT.

M1Q5) Difference between CISC and RISC processors (5 marks)

CISC (Complex Instruction Set Computing) and RISC (Reduced Instruction Set Computing) are two types of processor architectures designed for different computational approaches.

Feature	CISC (Complex Instruction Set Computing)	RISC (Reduced Instruction Set Computing)
Instruction Set	Large and complex instructions (100+).	Small and simple instructions (fewer than 100).

Execution Speed	Slower due to complex instructions.	Faster due to simple, uniform instructions.
Clock Cycles per Instruction	Multiple cycles (complex operations in one instruction).	Mostly single-cycle execution.
Registers Used	Fewer general-purpose registers.	More general-purpose registers.
Memory Usage	More memory required for instructions.	Less memory usage due to smaller instructions.
Code Efficiency	Higher code density, fewer lines of code.	Requires more instructions for complex tasks.
Pipelining	Difficult due to variable-length instructions.	Easier due to fixed-length instructions.
Examples	Intel x86, Motorola 68000.	ARM, MIPS, PowerPC.

M1Q6) Static RAM (SRAM) cell (5 marks)

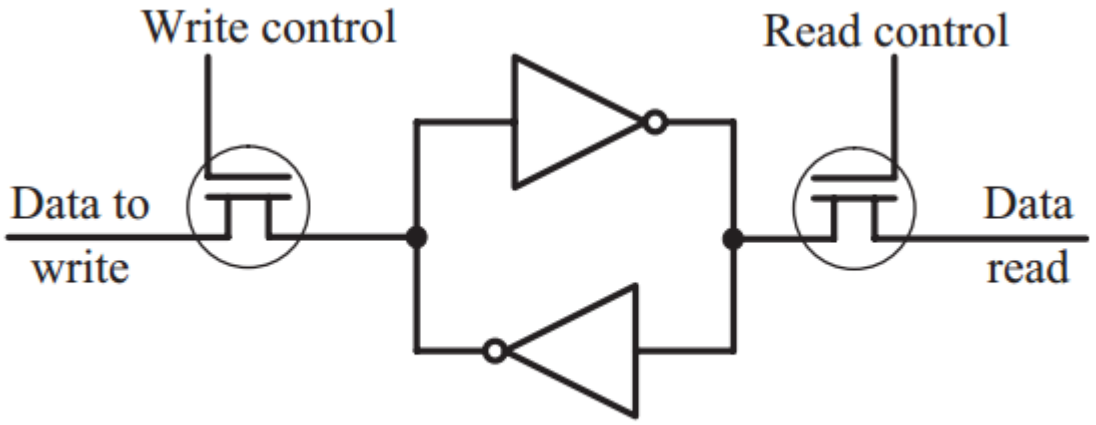

Fig. 2.11 Visualisation of SRAM cell

Static RAM (SRAM) is a type of volatile memory that **retains data as long as power is supplied**. Unlike Dynamic RAM (DRAM), it **does not require periodic refreshing**, making it faster and more power-efficient for cache memory and high-speed applications.

Structure of an SRAM Cell:

An SRAM cell is typically built using **6 transistors (6T SRAM cell)** in a cross-coupled arrangement.

Components of a 6T SRAM Cell:

1. **Two Cross-Coupled Inverters:** Forms a **bistable flip-flop** to store a single bit (either 0 or 1).
2. **Two Access Transistors (MOSFETs):** Control access to the memory cell during read and write operations.

Operation of an SRAM Cell:

1. **Write Operation:**
   - The word line (WL) is activated, enabling the access transistors.
   - Data is written into the cell by forcing **Q = 1, Q' = 0** (or vice versa).
2. **Read Operation:**
   - The word line is activated, and the stored value is read through the bit lines (BL & BL').
3. **Hold State:**
   - When WL is **inactive**, the inverters hold the data without consuming additional power.

M1Q7) Opto Coupler (5 marks)

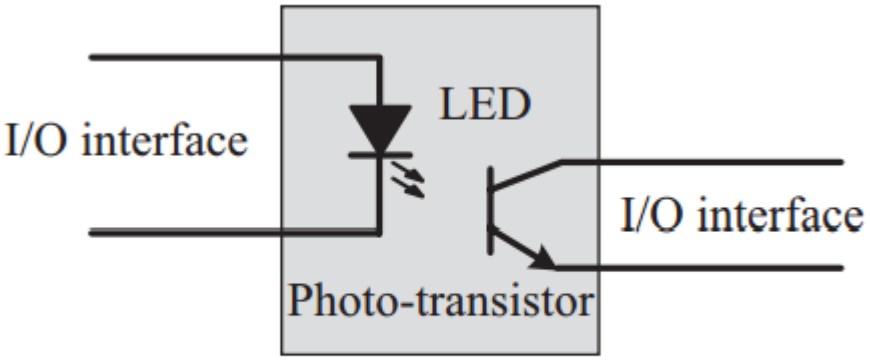

Fig. 2.16 An optocoupler device

An **Opto Coupler** (or **Opto Isolator**) is an electronic component that **transfers electrical signals between two isolated circuits using light**. It ensures electrical isolation while allowing signal transmission, protecting low-voltage circuits from high-voltage spikes.

Construction of an Opto Coupler:

An Opto Coupler consists of:

1. **Light Emitting Diode (LED):** Converts electrical signal into light.
2. **Photodetector (Phototransistor, Photodiode, or Photo SCR):** Detects the light and converts it back to an electrical signal.
3. **Isolation Barrier:** Air gap or transparent encapsulation ensures no direct electrical connection.

Working of an Opto Coupler:

1. **Input Stage:** When current flows through the LED, it emits infrared light.
2. **Transmission:** The light crosses the isolation barrier.
3. **Output Stage:** The photodetector receives the light, generating an electrical output.
4. **Electrical Isolation:** There is no direct electrical connection, ensuring safety.

M2Q1) Characteristics of embedded system in detail (5 marks)

Embedded systems are **specialized computing systems** designed for specific tasks. They differ from general-purpose computers in terms of **efficiency, reliability, and real-time performance**.

1. **Application-Specific Functionality**
    - Embedded systems are designed to perform **a dedicated task** rather than multiple functions.
    - Example: A washing machine controller is programmed only to manage wash cycles.
2. **Real-Time Operation**
    - Many embedded systems function in **real-time**, meaning they must process data and respond within a strict time limit.
    - **Types of Real-Time Systems:**

    - **Hard Real-Time:** Failure to meet the deadline causes system failure (e.g., Airbag system in cars).
    - **Soft Real-Time:** Some delay is acceptable but affects performance (e.g., Video streaming).

3. **Reliability and Stability**
    - Embedded systems operate **continuously for long durations** without failure.
    - They must handle **harsh environmental conditions**, such as **temperature changes, power fluctuations, and vibrations**.
    - Example: Embedded systems in medical devices (e.g., pacemakers) must function without failure.
4. **Power Efficiency**
    - Embedded systems are **optimized for low power consumption** to enhance battery life, especially in portable devices.
    - Example: Wearable fitness trackers like **Fitbit** have ultra-low power consumption.
5. **Memory Constraints**
    - Most embedded systems have **limited memory (ROM & RAM)** and storage capacity since they execute **specific tasks**.
    - Example: An **ATM machine** has a predefined firmware stored in ROM.
6. **Size and Cost Efficiency**
    - Designed to be **compact and cost-effective** for mass production.
    - Example: A **microcontroller-based embedded system** fits into small devices like digital cameras.
7. **Connectivity and Communication**
    - Many modern embedded systems communicate with other devices via **UART, I2C, SPI, Wi-Fi, Bluetooth, and Zigbee**.
    - Example: **Smart home devices** (IoT) use Wi-Fi/Zigbee for connectivity.
8. **Embedded Software (Firmware)**
    - Runs on **low-level programming** (C, C++, Assembly).

- Once programmed, the firmware is **not frequently changed**.
- Example: A **traffic light controller** runs a fixed algorithm.

M2Q2) Operation quality attributes (5 marks)

Operational quality attributes define the **non-functional requirements** of an embedded system during its working condition. These attributes ensure **efficiency, reliability, and performance** under various conditions.

1. **Response Time (Real-Time Performance)**
   - The **time taken by the system to respond** to an input or event.
   - Critical in **real-time systems** where a delay can lead to failure.
2. **Throughput**
   - The **amount of work** the system can perform in a given time.
   - Measured as the number of **tasks completed per second**.
3. **Reliability**
   - The system must operate **without failure for a long duration**.
   - Includes **error handling, fault tolerance, and self-recovery** mechanisms.
4. **Maintainability**
   - The ease of **diagnosing, fixing, and upgrading** the system.
   - Firmware updates and debugging tools enhance maintainability.
5. **Security**
   - Protection against **unauthorized access, cyberattacks, and data breaches**.
   - Important for **IoT devices, banking systems, and automotive systems**.
6. **Safety**
   - Ensures the system **does not cause harm** to users or the environment.
   - Important in **industrial, medical, and automotive applications**.

M2Q3) Non-Operation quality attributes (5 marks)

Non-operational quality attributes define the **design, development, and long-term usability aspects** of an embedded system. These attributes influence the system's **upgradability, cost-effectiveness, and adaptability** over time.

1. **Testability & Debug-ability**
    - The ease with which an embedded system can be **tested and debugged** during development and maintenance.
    - Systems should support **diagnostic tools, simulators, and in-circuit debuggers**.
2. **Evolvability (Scalability & Upgradability)**
    - The ability to **adapt and expand** in the future without major redesign.
    - Ensures **software updates, additional features, or hardware modifications** can be integrated easily.
3. **Portability**
    - The ability to **run on different hardware platforms** with minimal modifications.
    - Embedded software should be **hardware-independent** as much as possible.
4. **Time-to-Market & Time-to-Prototyping**
    - **Time-to-Market (TTM):** The speed at which an embedded product can be developed and sold.
    - **Time-to-Prototyping:** The time taken to build the first functional version of the system.
5. **Cost Efficiency**
    - The system should be designed to **minimize production and operational costs** while maintaining performance.
    - Includes **hardware selection, power consumption, and production scalability.**

M2Q4) Working of a washing machine (5 marks)

A washing machine is an **embedded system** designed for **automated washing of clothes**. It operates using **sensors, actuators, and a microcontroller** to

control different washing cycles.

1. **Input Selection & Initialization**
    - The user selects the **wash mode, water level, and temperature** using a control panel.
    - A **microcontroller** reads user inputs and configures the washing process accordingly.
2. **Water Inlet Control**
    - **Sensors** detect the required water level based on the selected wash mode.
    - **Solenoid valves** control the flow of water into the drum.
3. **Washing Process**
    - The **motor** rotates the drum in both directions to ensure proper cleaning.
    - **Detergent dispenser** releases detergent at the right time.
4. **Rinsing Process**
    - Dirty water is drained using a **pump and drain valve**.
    - Fresh water enters the drum to rinse the clothes and remove detergent residues.
5. **Spinning & Drying**
    - The motor runs at **high speed (RPM)** to remove excess water from clothes.
    - **Centrifugal force** pushes water out through small holes in the drum.
6. **Completion & Door Unlocking**
    - The washing cycle ends, and the **buzzer or LED indicator** notifies the user.
    - The **safety lock mechanism** ensures the door opens only after the cycle ends.

********** EOF **********