# Computer Architecture


## by: Tassadaq Hussain

**Director Centre for AI and BigData**

**Professor Department of Electrical Engineering**

**Namal University Mianwali**


**Collaborations:**

**Barcelona Supercomputing Center, Spain**

**European Network on High Performance and Embedded Architecture and Compilation**

**Pakistan Supercomputing Center**

# Computer Architecture

- Design, organization, and functionality of a computer system, focusing on how its components interact to perform tasks efficiently. It encompasses the hardware, software, and interfaces that define a computer's capabilities and performance.

# Key Aspects of Computer Architecture (Including Basic Hardware Components)

- Instruction Set Architecture (ISA)
- Microarchitecture
- Memory Systems
- Buses
- Input/Output (I/O) Systems
- Control Mechanisms
- Performance Optimization
- Interconnects

# Applications of Computer Systems

## 01
**Manufacturing Equipment**

## 02
**Domestic Appliances**

## 03
**Audio/Video Equipment**

## 04
**Gamming**

## 05
**Telecommunication**

## 06
**Medical Devices**

## 07
**Cars And Vehicles**

## 08
**Sensor Integration**

# Types of Computer Architecture

## Based on Application

- Real Time
- Stand alone
- Networked
- Mobile

1

## Based on Performance of Microcontroller

- Small Scale
- Large Scale
- Sophisticated

2

## Based on Complexity

- Hard-Real Time
- Soft Real Time

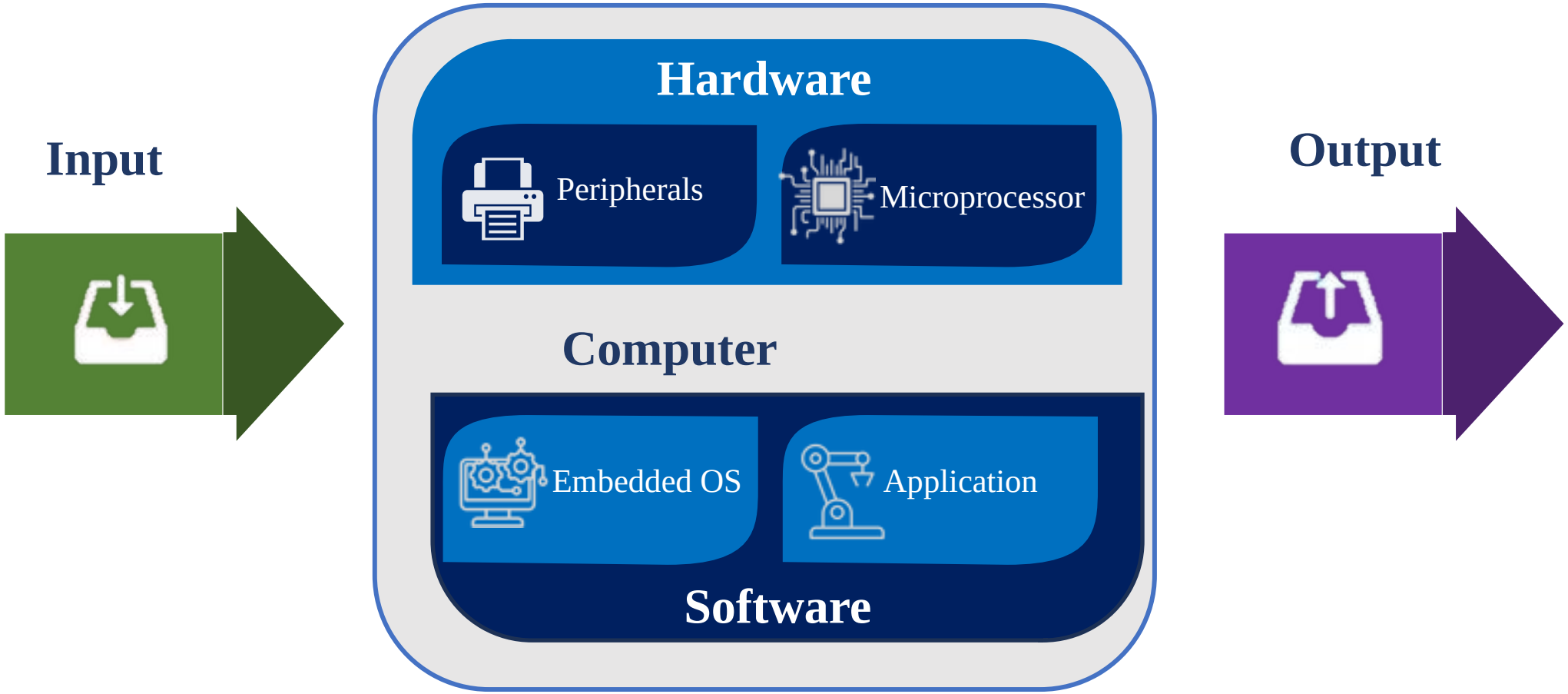
3

## Based on Functional Requirements

- Control Systems
- Monitoring Systems
- Data Acquisition Systems

4

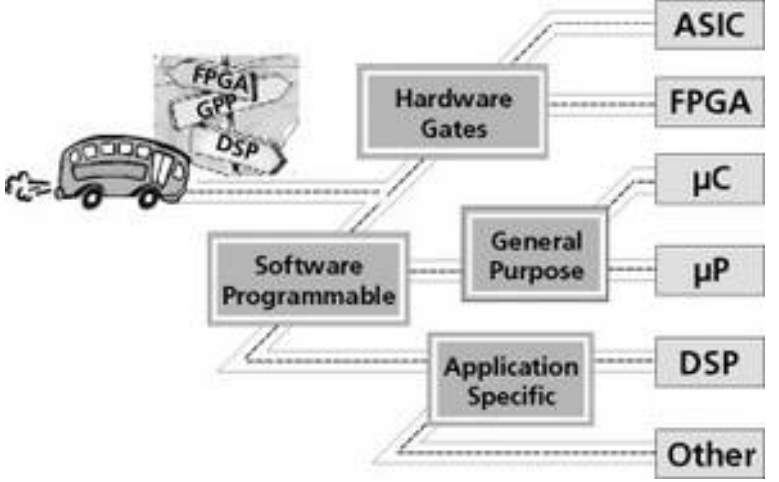
# Basic Computer Architecture



# Contents



# Microprocessors and Microcontrollers




## Microprocessor:

A central processing unit (CPU) on a single integrated circuit (IC) designed to perform general-purpose computation.

**Key Characteristics:** High processing power, requires external components for I/O, memory, and storage.

Examples: Intel Core, AMD Ryzen

**Microcontrollers:**

An integrated circuit designed to perform specific control functions, containing a CPU, memory, and I/O peripherals on a single chip.

**Key Characteristics:** Compact, low power, designed for specific tasks.

**Examples:** Arduino (ATmega328), PIC (Microchip PIC16F877A)

## Key Differences

- **Architecture**:
  - Microprocessors: CPU only, needs external components.
  - Microcontrollers: CPU, memory, I/O peripherals integrated.
- **Usage**:
  - Microprocessors: General-purpose computing, PCs, servers.
  - Microcontrollers: Embedded systems, appliances, automotive.
- **Power Consumption**:
  - Microprocessors: Higher power consumption.
  - Microcontrollers: Lower power consumption.
- **Complexity and Cost**:
  - Microprocessors: More complex, higher cost.
  - Microcontrollers: Simpler, cost-effective for specific tasks.

# Major Units in Computer Architecture

**Memory Management Unit (MMU)**

Purpose: Translates virtual addresses to physical addresses, handles memory protection, and manages virtual memory.

Integration: Essential for supporting sophisticated memory management required by modern operating systems.

**Bus System**

Purpose: Facilitates communication between the CPU, memory, and peripherals.

Components:

- Address Bus: Carries memory addresses.
- Data Bus: Transfers data.
- Control Bus: Sends control signals.

Integration: Crucial for ensuring all parts of the computer system can communicate effectively.

**Input/Output (I/O) System**

Purpose: Manages data flow between the CPU and external devices.

Components:

- I/O Controllers: Interfaces that manage the interaction between the system and peripheral devices.
- I/O Ports: Connection points for external devices.

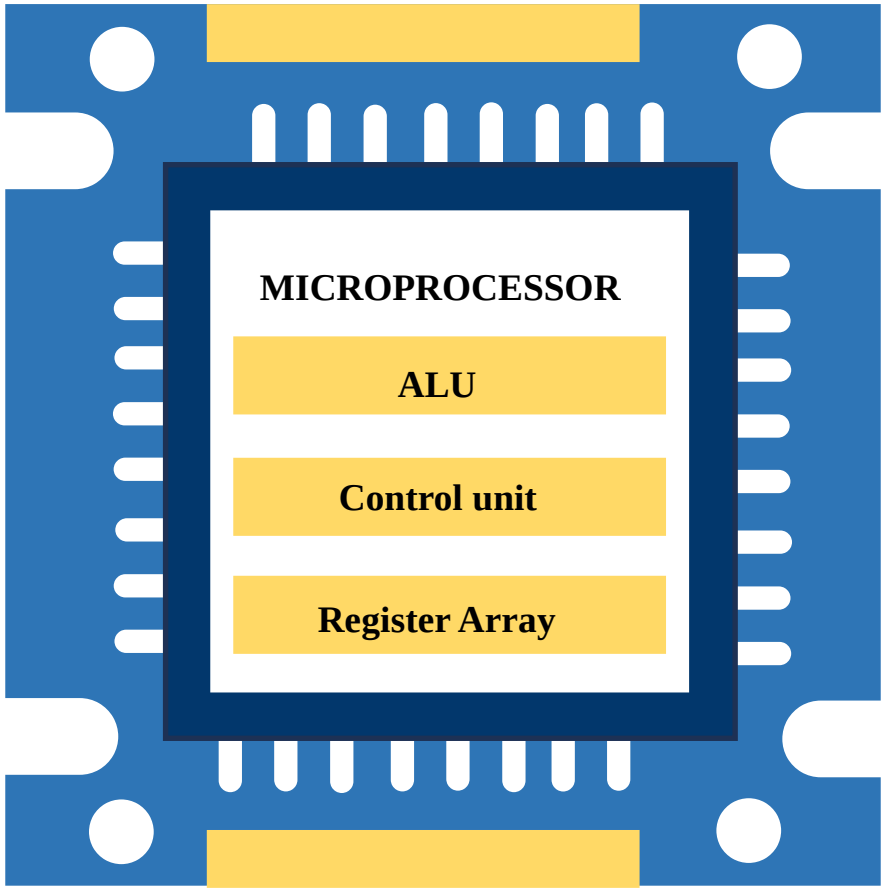
Integration: Often considered part of the bus system, as it involves communication pathways.

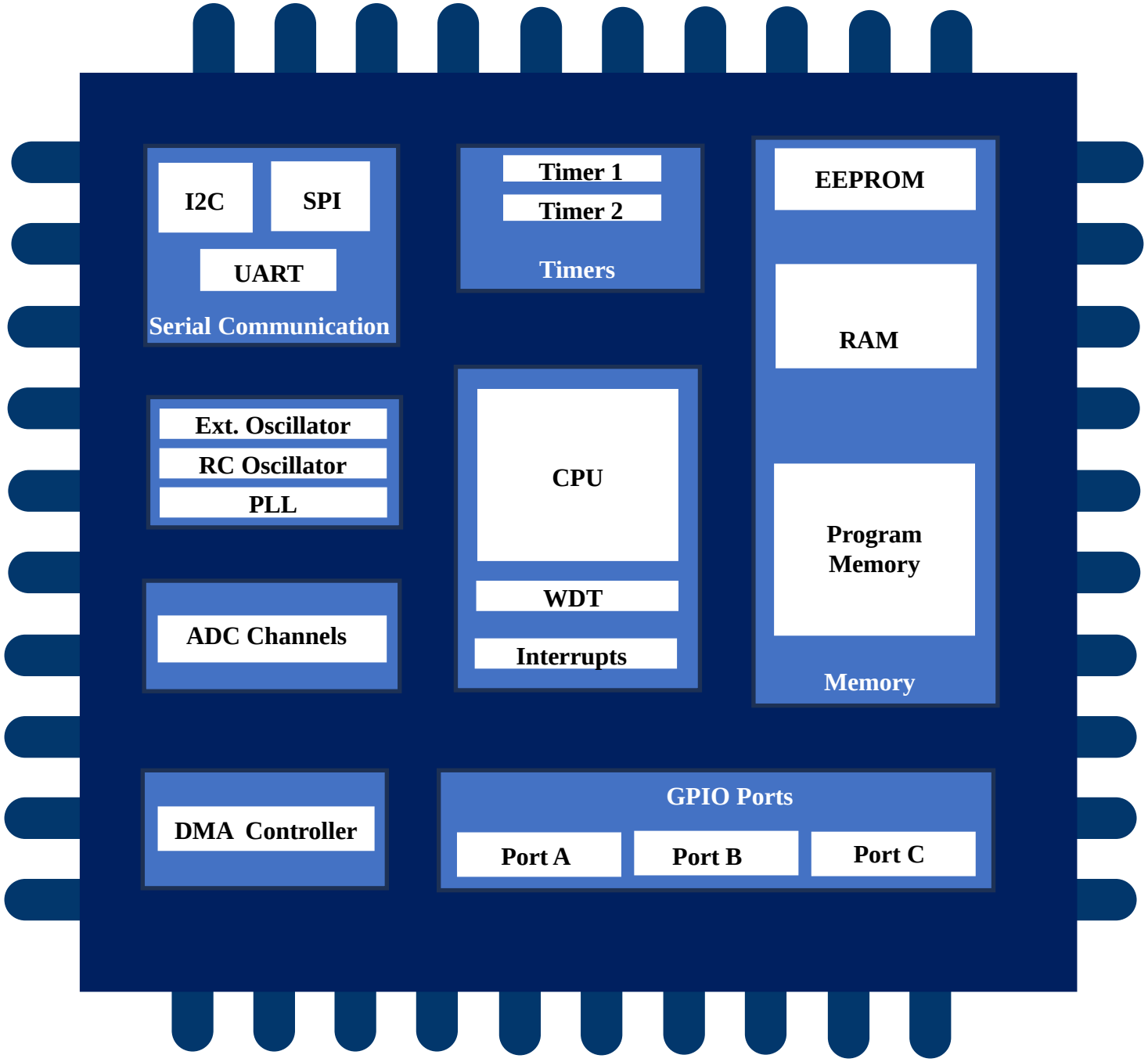
**System Software and Firmware**

BIOS/UEFI: Initializes hardware and provides a runtime environment for the operating system.

Operating System: Manages hardware resources and provides services for application software.

Firmware: Low-level software embedded in hardware to control device-specific functions.


MICROPROCESSOR
ALU
Control unit
Register Array


I2C
SPI
UART
Serial Communication
Timer 1
Timer 2
Timers
EEPROM
RAM
Program Memory
Memory
Ext. Oscillator
RC Oscillator
PLL
CPU
WDT
Interrupts
ADC Channels
DMA Controller
GPIO Ports
Port A
Port B
Port C

# Microprocessors and Microcontrollers

### Applications of Microprocessors

- Personal Computers
- Servers and Workstations
- Gaming Consoles
- Smartphones and Tablets
- High-performance computing systems

### Applications of Microcontrollers

- Home Appliances (Microwaves, Washing Machines)
- Automotive Systems (Engine Control Units, Airbags)
- Consumer Electronics (Remote Controls, Toys)
- Industrial Automation (Robotic Controls, Sensors)
- IoT Devices (Smart Home Devices, Wearables)

### Architecture Comparison

Key Components: ALU, Control Unit, Registers, Memory (RAM/ROM), I/O Ports, Timers

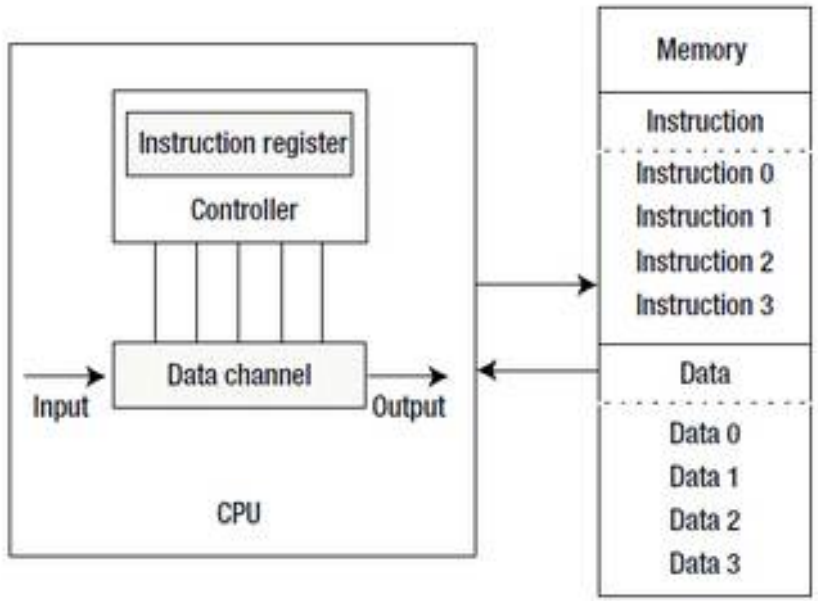
### Development Tools

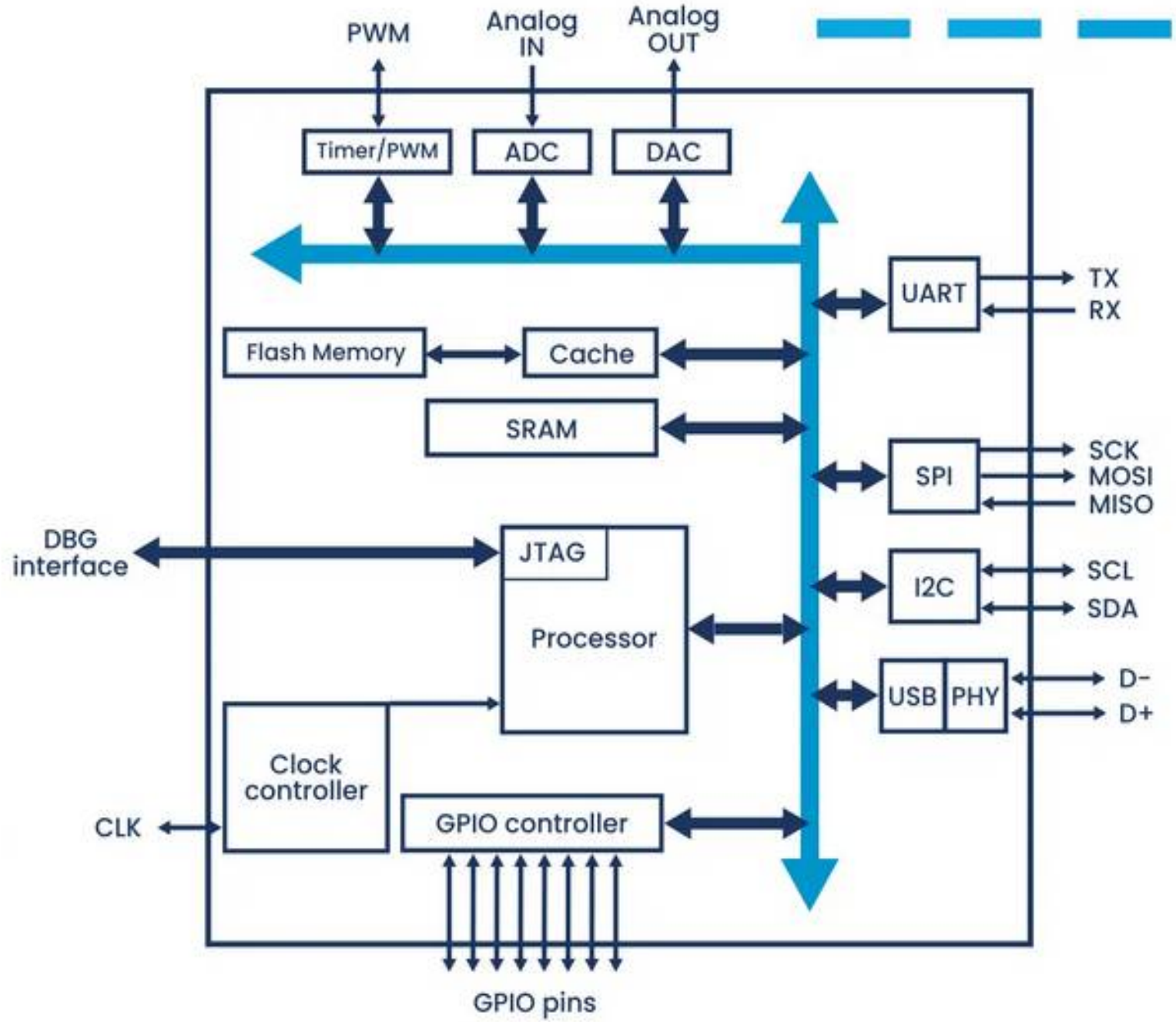
- Microprocessors: Compilers, Debuggers, IDEs (e.g., GCC, Visual Studio)
- Microcontrollers: Integrated Development Environments (IDEs), Simulators, Debuggers (e.g., MPLAB, Keil uVision, Arduino IDE)

### Trends and Future Directions

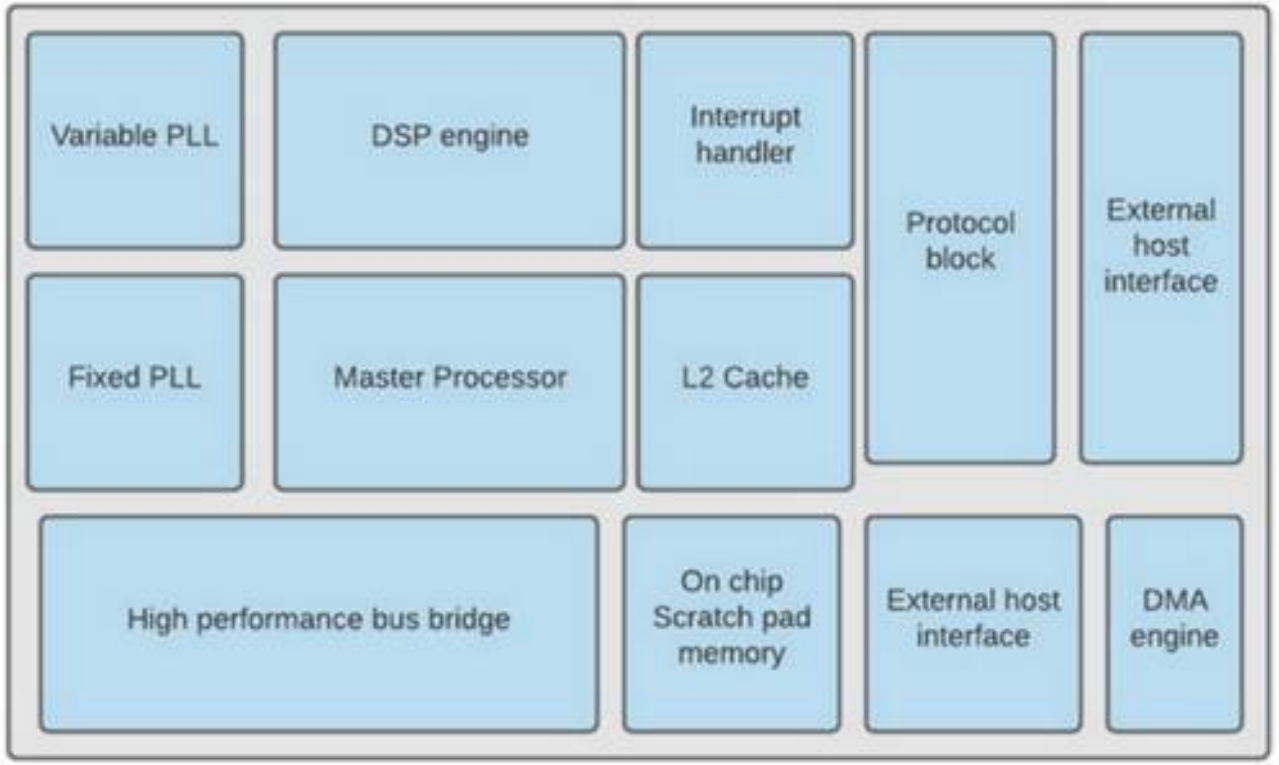
Increasing Integration and Miniaturization AI and Machine Learning Integration IoT and Edge Computing Low Power and Energy-Efficient Designs

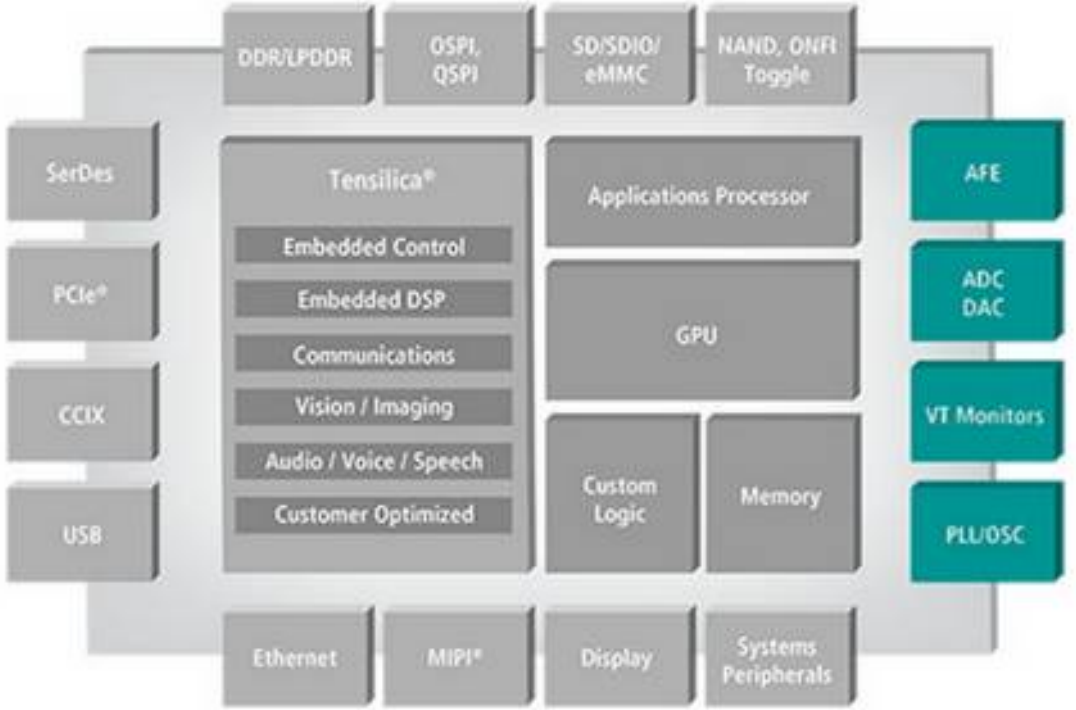
# Contents


Memory
Instruction
Instruction 0
Instruction 1
Instruction 2
Instruction 3
Data
Data 0
Data 1
Data 2
Data 3
Instruction register
Controller
Data channel
Input
Output
CPU


PWM
Analog IN
Analog OUT
Timer/PWM
ADC
DAC
UART
TX
RX
Flash Memory
Cache
SRAM
SPI
SCK
MOSI
MISO
DBG interface
JTAG
Processor
I2C
SCL
SDA
USB
PHY
D-
D+
Clock controller
CLK
GPIO controller
GPIO pins

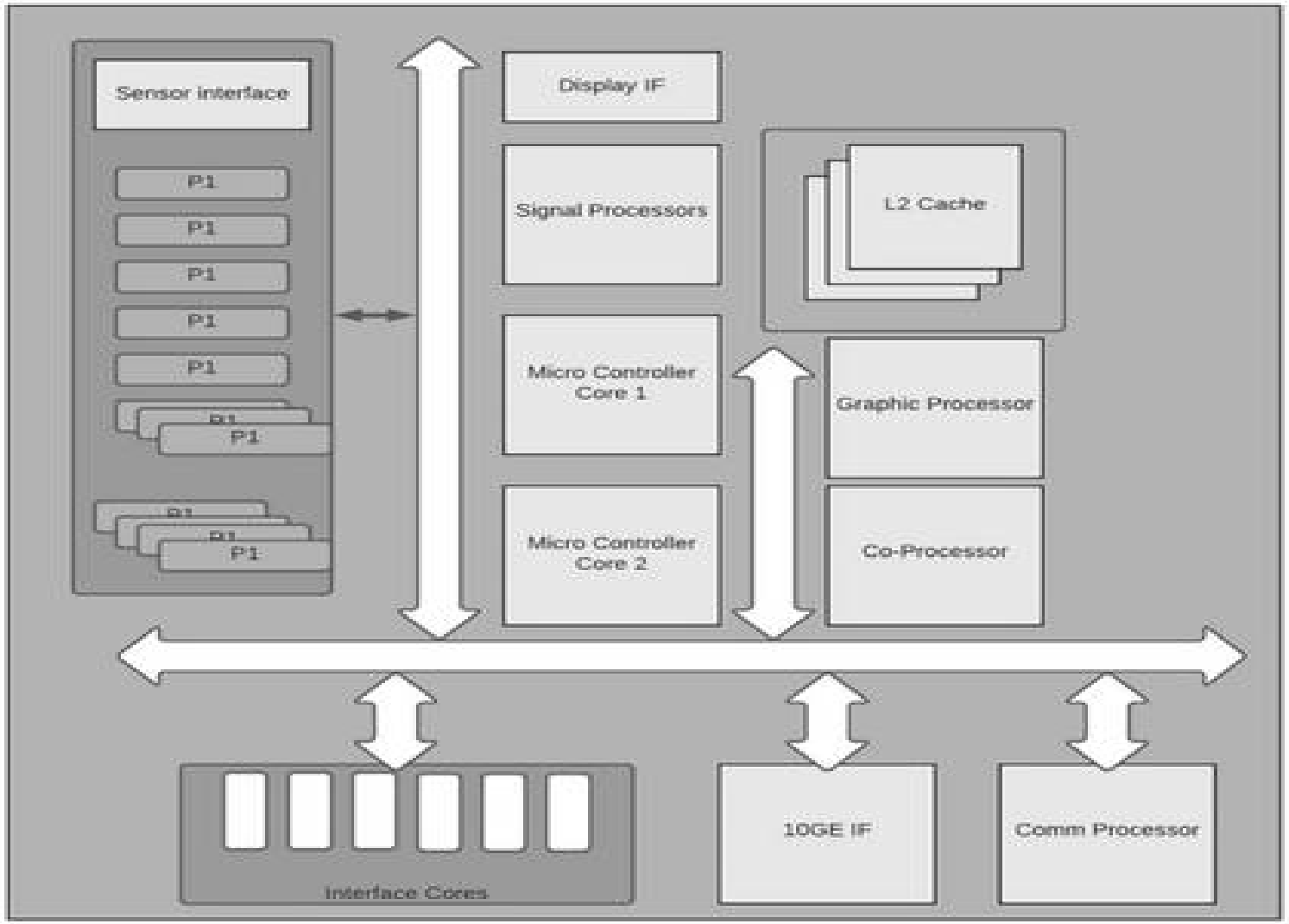
# HPC SoC Systems



# Computer Arch: SoC Key Components

- **Processor**
- **ISA**
- Internal Bus
- Memory Unit
- Power
- Scheduling
- Input / Output
- DMA



# Processor

- RISC (Reduced Instruction Set Computing):
- Common ISAs include ARM and RISC-V.
- Simplified instructions for efficient execution and low power consumption.

# Internal Bus: System on Chip (SoC)

- **Integrated Components:**
- Combines CPU, memory, I/O ports, and other peripherals on a single chip.
- Reduces physical space and power consumption.
- **Peripheral Integration:**
- Includes components such as ADCs, DACs, timers, PWM controllers, and communication interfaces (UART, SPI, I2C, etc.).

# Memory Unit

- **On-Chip Memory:**
- Typically includes SRAM and ROM/Flash memory.
- Fast access times for real-time performance.
- **External Memory Interfaces:**
- Support for connecting to external memory modules like DRAM or NOR/NAND Flash.

**SRAM (Static RAM)**

Purpose: Temporary storage for variables and data during execution.
Characteristics: Volatile, fast access.
Usage: Stores variables, stack, and temporary data.

**EEPROM**

Purpose: Stores data that must persist across power cycles.
Characteristics: Non-volatile, byte-addressable.
Usage: Saves user settings and calibration data.

**Flash Memory (Program Memory)**

Purpose: Stores the firmware or program code.
Characteristics: Non-volatile, reprogrammable.
Usage: Holds the application code and bootloader.

# Scheduling

- **Real-Time Operating System (RTOS) Support:**
- Features for deterministic execution and low-latency interrupts.
- **Dedicated Timers and Counters:**
- Hardware support for precise timing operations.
- **Interrupt Handling:**
- Fast and efficient interrupt processing capabilities.

# Power Consumption

- **Power Management Features:**
- Multiple power modes (active, idle, sleep, deep sleep).
- Dynamic voltage and frequency scaling (DVFS).
- **Efficient Instruction Execution:**

Instructions optimized for minimal power use per operation.

# I/O and Communication Interfaces

- **Integrated Communication Peripherals:**
- Support for serial communication protocols like UART, SPI, I2C, CAN, and USB.
- **GPIO (General-Purpose Input/Output) Pins:**
- Configurable pins for direct hardware interfacing and control.

# External Buses Low Performance

Increasing demand for high-speed data transfer rates

- Growing need for low latency and high throughput
- Scalability and flexibility requirements
- Advancements in technology and cost reductions

- **UART (Universal Asynchronous Receiver-Transmitter)**
  - - Theory: UART is a serial communication protocol that uses asynchronous transmission, meaning that data is transmitted one bit at a time, without a clock signal.
  - Pros:
    - Simple and easy to implement
    - Low power consumption
    - Widely used in serial communication applications
  - Cons:
    - Limited data transfer rate (typically up to 115.2 kbps)
    - No built-in error detection or correction
    - Not suitable for high-speed or real-time applications

## SPI (Serial Peripheral Interface)

SPI is a serial communication protocol that uses synchronous transmission, meaning that data is transmitted with a clock signal.

- Pros:

 - Full-duplex communication (simultaneous read and write)

 - High data transfer rates (up to 100 Mbps or more)

 - Simple and easy to implement

- Cons:

 - Requires four wires (MOSI, MISO, SCK, SS)

 - No built-in error detection or correction

 - Can be prone to noise and interference issues

## I2C (Inter-Integrated Circuit)

- Theory: I2C is a serial communication protocol that uses synchronous transmission, meaning that data is transmitted with a clock signal.

- Pros:

 - Multi-master, multi-slave communication

 - Built-in error detection and correction (ACK/NAK protocol)

 - Widely used in microcontroller and sensor applications

- Cons:

 - Limited data transfer rate (up to 400 kbps in standard mode, 3.4 Mbps in fast mode)

 - More complex than UART, requiring more pins and logic

 - Can be prone to noise and interference issues

# Direct Memory Access (DMA) in Microcontrollers

**Offload CPU:**

DMA allows the CPU to delegate data transfer tasks to the DMA controller, freeing up the CPU to perform other processing tasks.

**Efficient Data Transfer:**

DMA enables high-speed data transfers directly between memory and peripherals without CPU intervention for each data byte or word, improving overall system efficiency.

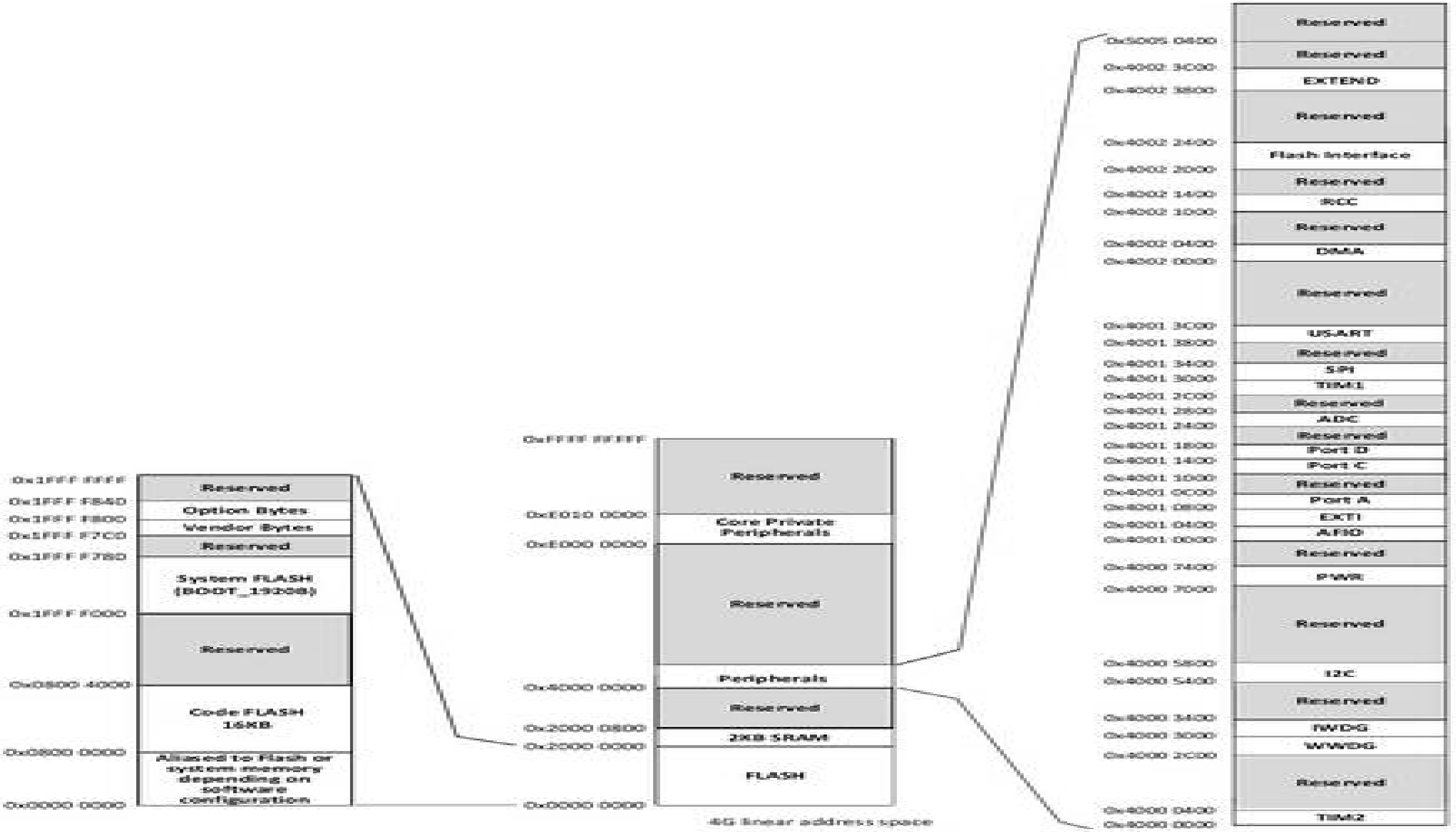
**Extended Address Space:**

DMA can handle block transfers using a single address setup, efficiently moving large amounts of data and effectively increasing the addressable data space.

# Contents

# Memory Mapping

- Memory mapping is a crucial aspect of System on Chip (SoC) architecture. It refers to the way different components of the SoC are allocated addresses in the memory space. This mapping allows the CPU and other components to access and interact with various parts of the system's memory and peripherals.
- **Key Aspects of Memory Mapping in SoCs.**
  - Bus System
  - Address Space Allocation
  - Memory Regions
  - Memory Mapping Techniques:
  - Memory Map Tables
  - Access Mechanisms
  - Virtual Memory Mapping
  - Address Decoding:
- **Example of Memory Mapping in an SoC**
  - 0x0000_0000 - 0x1FFF_FFFF: RAM (1 GB of addressable RAM)
  - 0x2000_0000 - 0x3FFF_FFFF: ROM or Flash memory
  - 0x4000_0000 - 0x5FFF_FFFF: Peripheral registers (e.g., GPIO, UART)
  - 0x6000_0000 - 0x7FFF_FFFF: External memory or memory-mapped I/O space


0x1FFF FFFF
Reserved
0x1FFF F80F
Option Bytes
0x1FFF F800
Vendor Bytes
0x1FFF F7CD
Reserved
0x1FFF F7BD
System FLASH (BOOT_19208)
0x1FFF F000
Reserved
0x0800 4000
Code FLASH 16KB
0x0800 0000
Aliased to Flash or system memory depending on software configuration
0x0000 0000
0xFFFF FFFF
Reserved
0xE010 0000
Core Private Peripherals
0xE000 0000
Reserved
0x4000 0000
Peripherals
Reserved
0x2000 0800
2KB SRAM
0x2000 0000
FLASH
0x0000 0000
4G linear address space
0x5005 0400
Reserved
0x4002 3C00
Reserved
0x4002 3800
EXTEND
0x4002 3400
Reserved
0x4002 2000
Flash interface
0x4002 1400
Reserved
0x4002 1000
RCC
0x4002 0400
Reserved
0x4002 0000
DMA
Reserved
0x4001 3C00
USART
0x4001 3800
Reserved
0x4001 3400
SPI
0x4001 3000
TIM1
0x4001 2C00
Reserved
0x4001 2800
ADC
0x4001 2400
Reserved
0x4001 1800
Port D
0x4001 1400
Port C
0x4001 1000
Reserved
0x4001 0C00
Port A
0x4001 0800
EXTI
0x4001 0400
AFIO
0x4001 0000
Reserved
0x4000 7400
PWR
0x4000 7000
Reserved
0x4000 5800
I2C
0x4000 5400
Reserved
0x4000 3400
IWDG
0x4000 3000
WWDG
0x4000 2C00
Reserved
0x4000 0400
TIM2
0x4000 0000

# Bus System Components

- Address Bus: Carries address information from the CPU to memory and peripherals. The address bus width determines the range of addresses that can be used in memory mapping.
- Data Bus: Transfers data between components based on the address specified on the address bus.
- Control Bus: Carries control signals that manage the read and write operations and other control functions.
- Functions:
  - Memory Map Configuration
  - Interconnects and Buses
  - Address Decoding
  - Memory-Mapped I/O

Example:

On-Chip Memory and Peripheral Mapping: Within the bus system, the memory map determines the layout of on-chip memory, peripheral registers, and I/O devices. The bus system ensures that the CPU and other components access the correct addresses based on this map.

# Address Map/Space Allocation

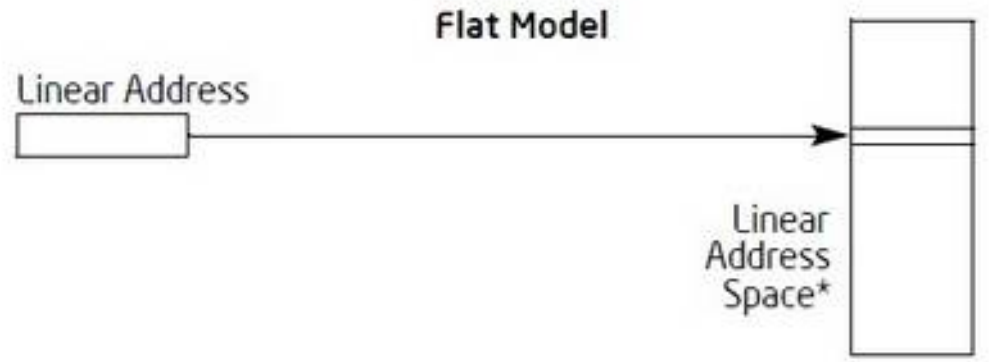
- Memory Address Space: Defines the range of addresses used to access different types of memory, including RAM, ROM, and external memory.
- Peripheral Address Space: Allocates addresses for various peripherals and I/O devices.

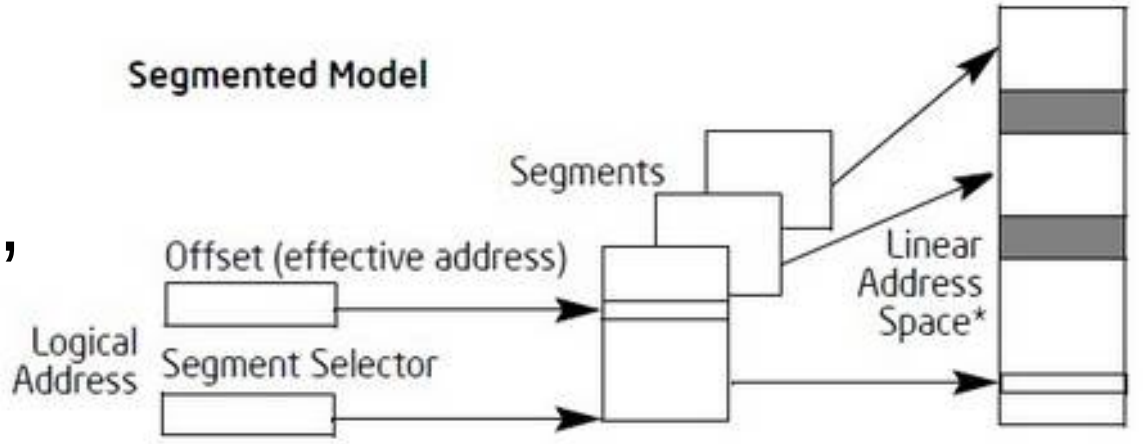
# Memory Regions

- Boot Memory: Often used to store the bootloader or initial firmware.
- Code Memory: Stores executable code and program instructions.
- Data Memory: Used for storing variables, stack, and heap data.
- Peripheral Registers: Memory-mapped addresses used to control and interact with peripheral devices (e.g., timers, UARTs, GPIOs).

# Memory Mapping Techniques:

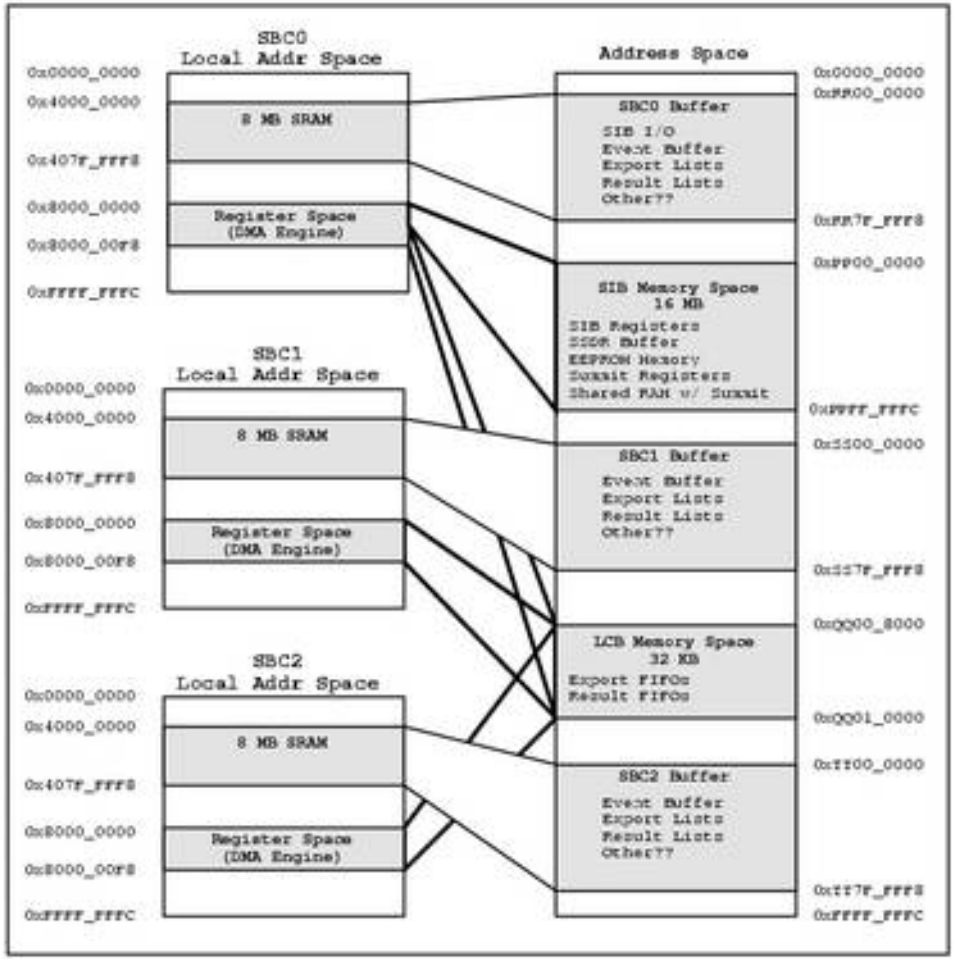
- Flat Memory Model: All memory and peripherals are mapped into a single, linear address space.
- Segmented Memory Model: Memory and peripherals are divided into segments or blocks, each with a specific address range.



# Memory Map Tables and Access Mechanisms

- Memory Map Table: A detailed table that outlines the starting address, size, and type of each memory region and peripheral.
- Memory-Mapped I/O: Peripherals are accessed by reading from or writing to specific memory addresses.
- Access Mechanism:
  - Linker Script: Define how different code and data sections are placed in memory.
  - Direct Memory Access (DMA): Allows peripherals to directly access memory without CPU intervention, reducing latency and improving performance.



# Virtual Memory Mapping

- Virtual Address Space: Some SoCs use virtual memory systems to abstract physical memory addresses, providing flexibility in memory management.
- Virtual Address: It is an address of a program's memory space.
- Page Table: The table contains mappings from virtual addresses to physical addresses. Each entry in the page table corresponds to a "page" of memory.
- Page Size: Memory is divided into fixed-size pages, typically ranging from 2 KB to 16 KB (though sizes like 4 KB or 8 KB are common). The virtual address is split into two parts:
  - Page Number: Identifies the page within the virtual address space.
  - Offset: Identifies the specific location within the page.
- Translation: When a virtual address is used, the page number is looked up in the page table to find the corresponding physical page. The offset is then added to this physical page to get the final physical address.
- Physical Address: The final physical address points to the exact location in the system's memory (RAM) where the data is stored.

# Contents

# System Clock Tree

- It is responsible for distributing clock signals throughout the system. It ensures that all components receive accurate and synchronized timing signals necessary for proper operation. Here's a detailed look at the clock tree and its role in embedded systems:
- Purpose of the Clock Tree:
- Timing Distribution: The clock tree distributes clock signals from a central oscillator or clock source to various components and subsystems within the embedded system.
- Synchronization: Ensures that different parts of the system operate in sync, which is crucial for reliable and predictable system performance.

# Components of a Clock Tree:

- Clock Source: The primary oscillator or clock generator that provides the initial clock signal.
- Clock Distributors: Distributes the clock to various parts of the system. This may include clock buffers, drivers, and multiplexers.
- Clock Dividers: Reduce the frequency of the clock signal to provide lower frequency clocks for different subsystems.
- Clock Multipliers: Increase the frequency of the clock signal if higher frequencies are required for certain components.
- Phase-Locked Loops (PLLs) and Delay-Locked Loops (DLLs): Used to generate stable, high-frequency clock signals from a lower-frequency reference clock, or to align the phase of clocks.
- Clock Gating: Mechanism to enable or disable the clock signal to specific parts of the system to save power when those parts are not in use.