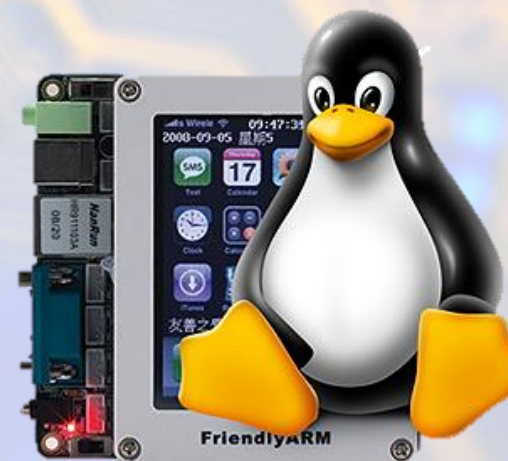
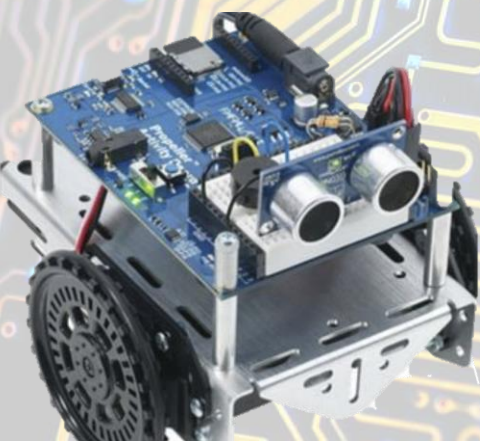


EC535 Introduction to Embedded Systems



Last Time

EC535 2025 Final Project Grading Criteria

Item	Grading	Score	Comments
Demo/ presentation	Successful project completion, performance of the demo, consideration of system as a whole, video.	20	What was the level of technical accomplishment for this project, the presentation sufficiently shows the project.
Technical Report	Written report to be submitted on 5/1.	70	Clarity, organization, and completeness of figures, diagrams, see below.
Git/commits	Completeness and adherence to weekly commit schedule	10	Commits by both team members, starting to work early (two weeks prior to presentation).

Total: 100

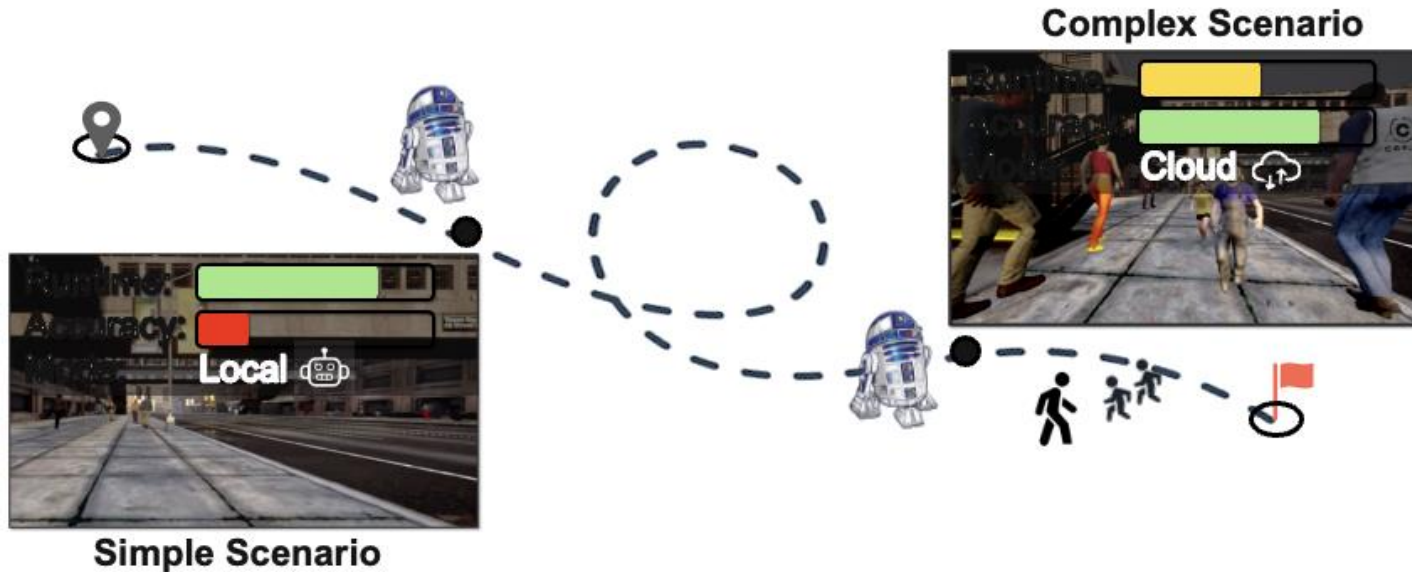
The goal of the final project is to show understanding of concepts from the class and implement into an integrated system that consider functionality – some task needs to be achieved, e.g., autonomous vehicle responses on time and arrives to a goal, user interface enables task completion, etc. The project should draw on concepts in embedded and resource-constrained systems. Project examples can be found here:

<https://docs.google.com/document/d/1oCxSCoXTv5vd8RuECxYzjg5Jnyiyxf6Fi64txKUYcOM/edit?usp=sharing>

Example report can be found here:

<https://drive.google.com/drive/folders/1D5XTPkJQXZXrxm9T4qOLDZvu1a7rzL6L?usp=sharing>

Schedule Should Depend on the Situation!



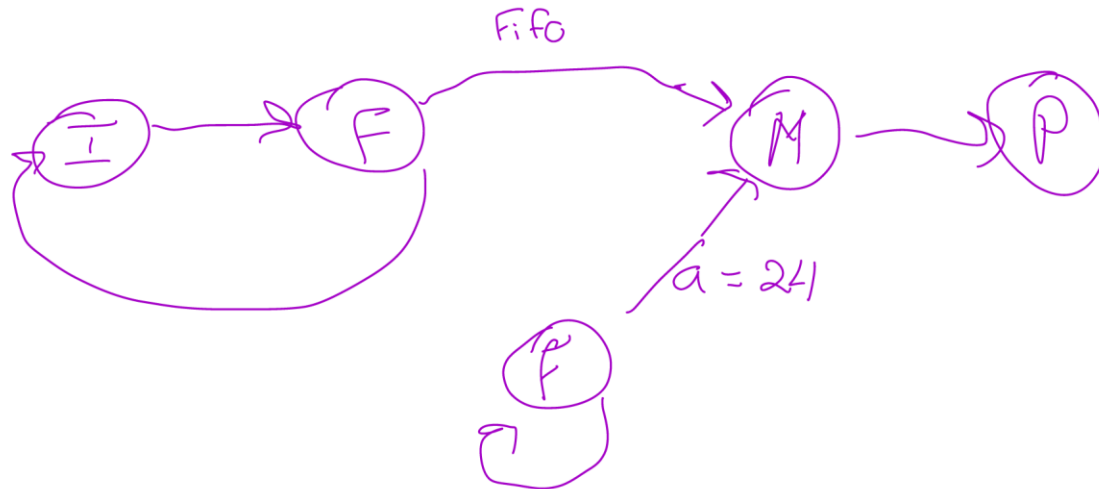
We present **UniLCD**, a novel local-cloud hybrid framework that dynamically routes computation between low-power local models and powerful cloud resources via reinforcement learning based on scenario complexity.

Dataflow design in SW

Recall synchronous dataflow (SDF) models from lectures. In this exercise, your task is to design a dataflow system that prints the multiples of 24 (24, 24*2, 24*3, 24*4, ...) using the following actors:

- The *fork* actor is a single-rate actor with one input and two outputs. Each input token is read, duplicated and copied to both outputs.
- The *increment* actor is a single-rate actor with one input and one output. An input token is read, incremented and copied to the output.
- The *multiply* actor is a single-rate actor with two inputs and one output. One token is read from each input, their token values are multiplied, and the result is copied to the output.
- The *print* actor is a single-rate actor with one input. It prints the value of the input token.

You will be developing the C code for each actor. A FIFO library to help you implement dataflow queues is provided for you. As an example, we will first implement the code for the multiply actor. Let's first consider `fifo.h`, the interface for the dataflow queue.



Embedded system functionality aspects

Embedded system functionality aspects

- **Processing**

- Transformation of data
- Implemented using processors

Embedded system functionality aspects

- **Processing**

- Transformation of data
- Implemented using processors

- **Communication**

- Transfer of data between processors and memories
- On-chip implementation
 - On-chip communication architecture
 - Network-on-chip (NOC)

Embedded system functionality aspects

- **Processing**

- Transformation of data
- Implemented using processors

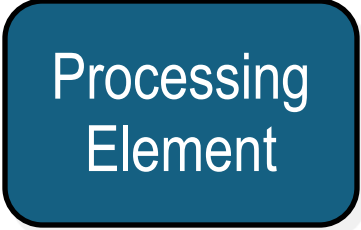
- **Communication**

- Transfer of data between processors and memories
- On-chip implementation
 - On-chip communication architecture
 - Network-on-chip (NOC)

- **Storage**

- Retention of data
- Implemented using memory

System Elements



Processing
Element

PEs may be CPUs or ASICs.

System Elements

Processing
Element

PE

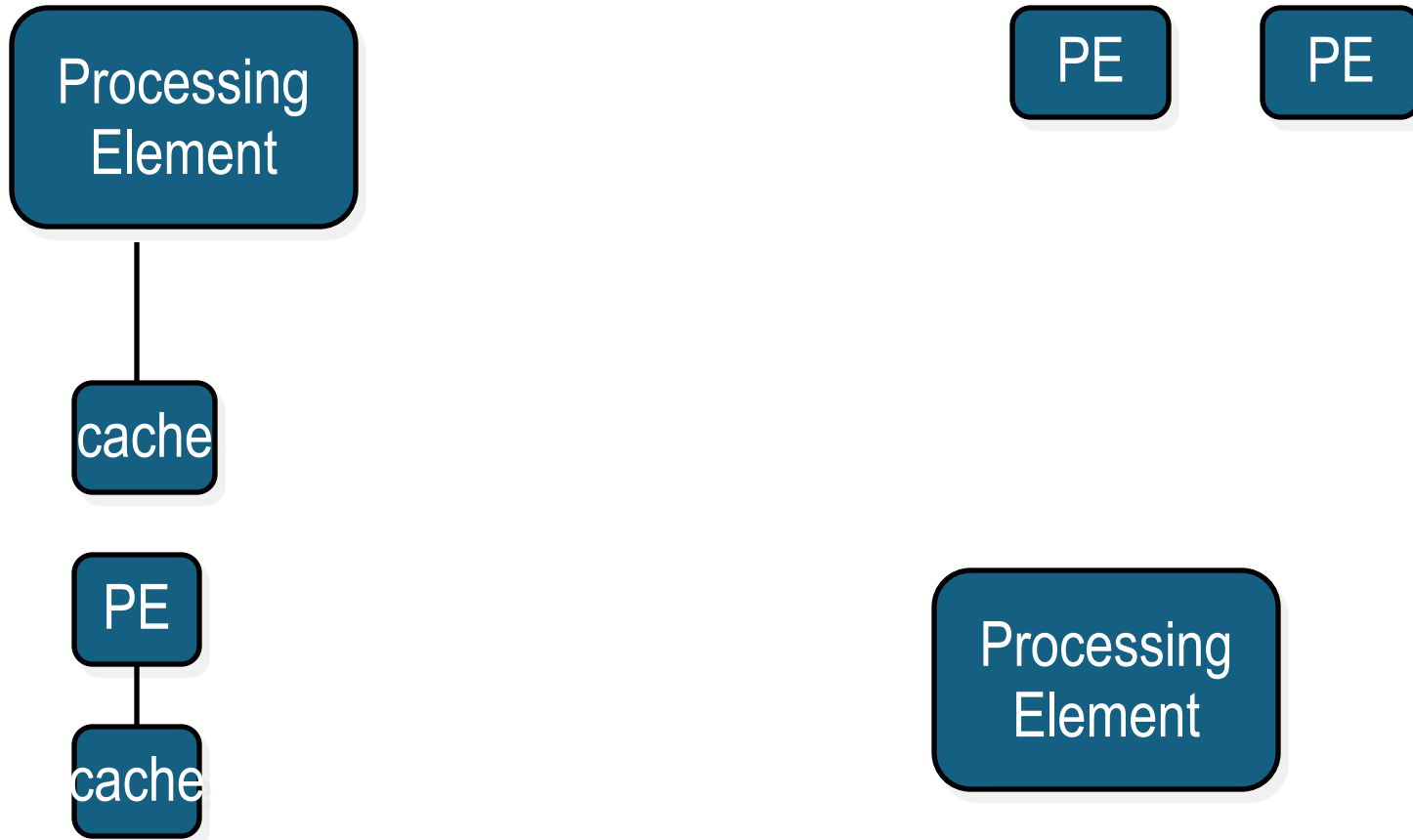
PE

PE

Processing
Element

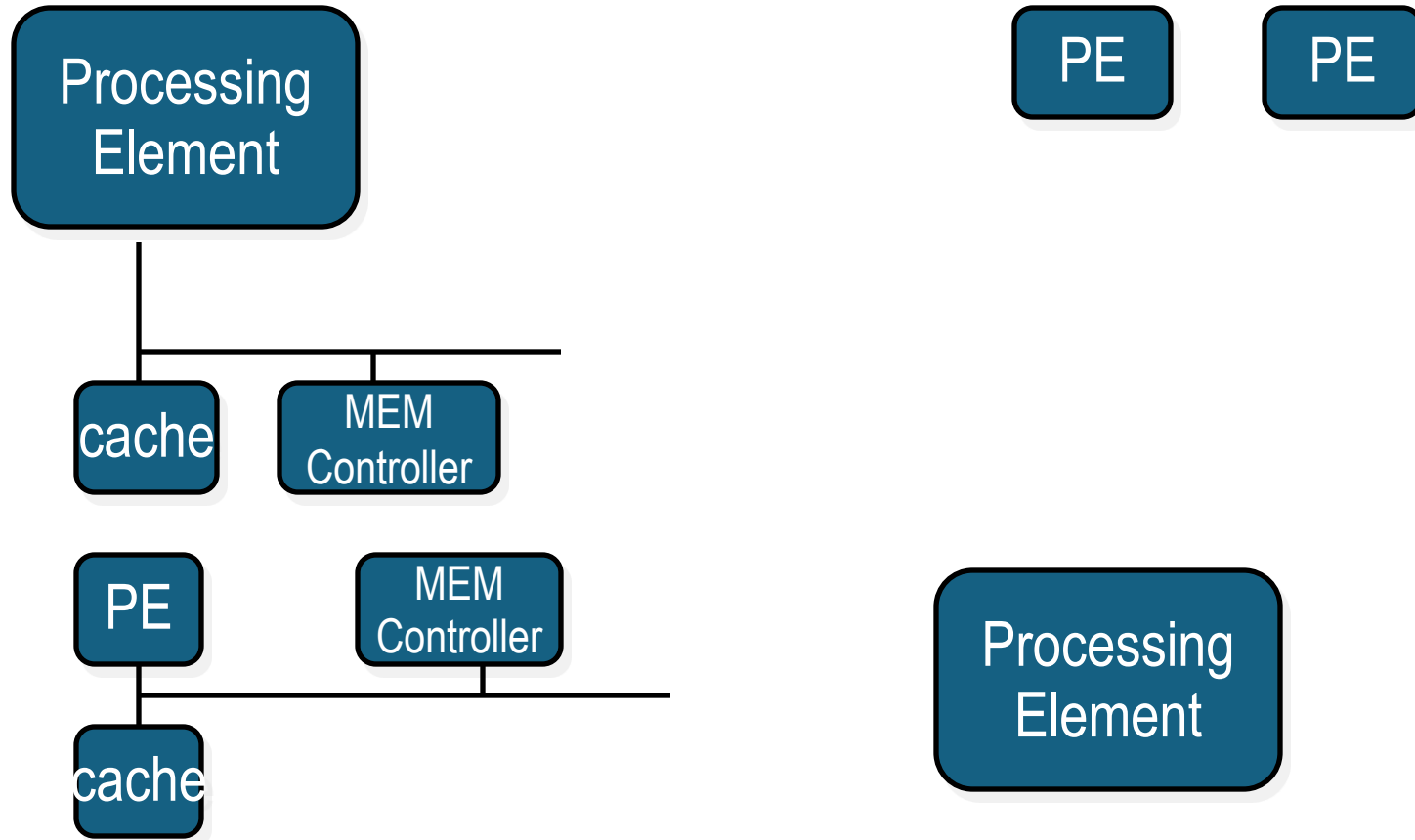
PEs may be CPUs or ASICs.

System Elements



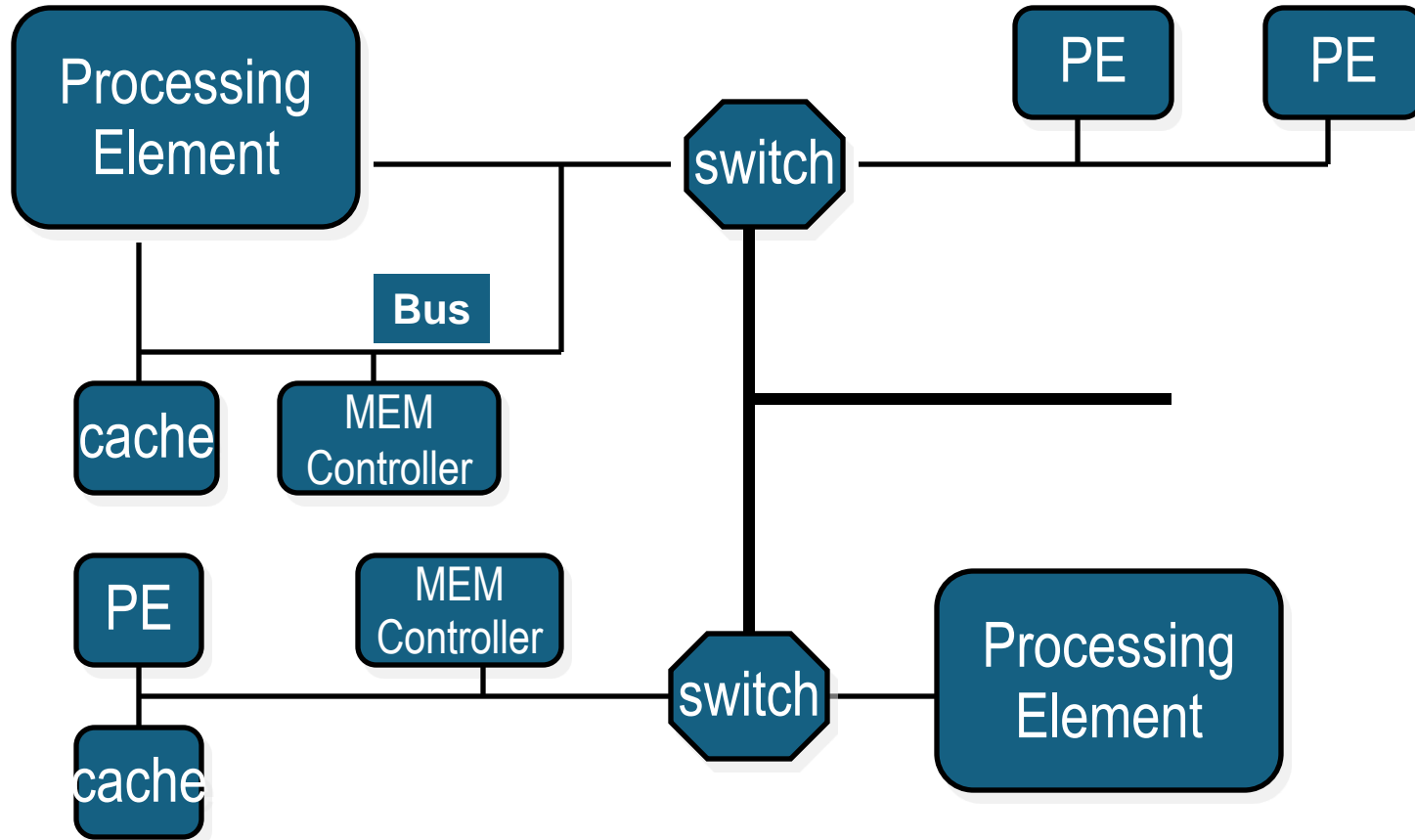
PEs may be CPUs or ASICs.

System Elements



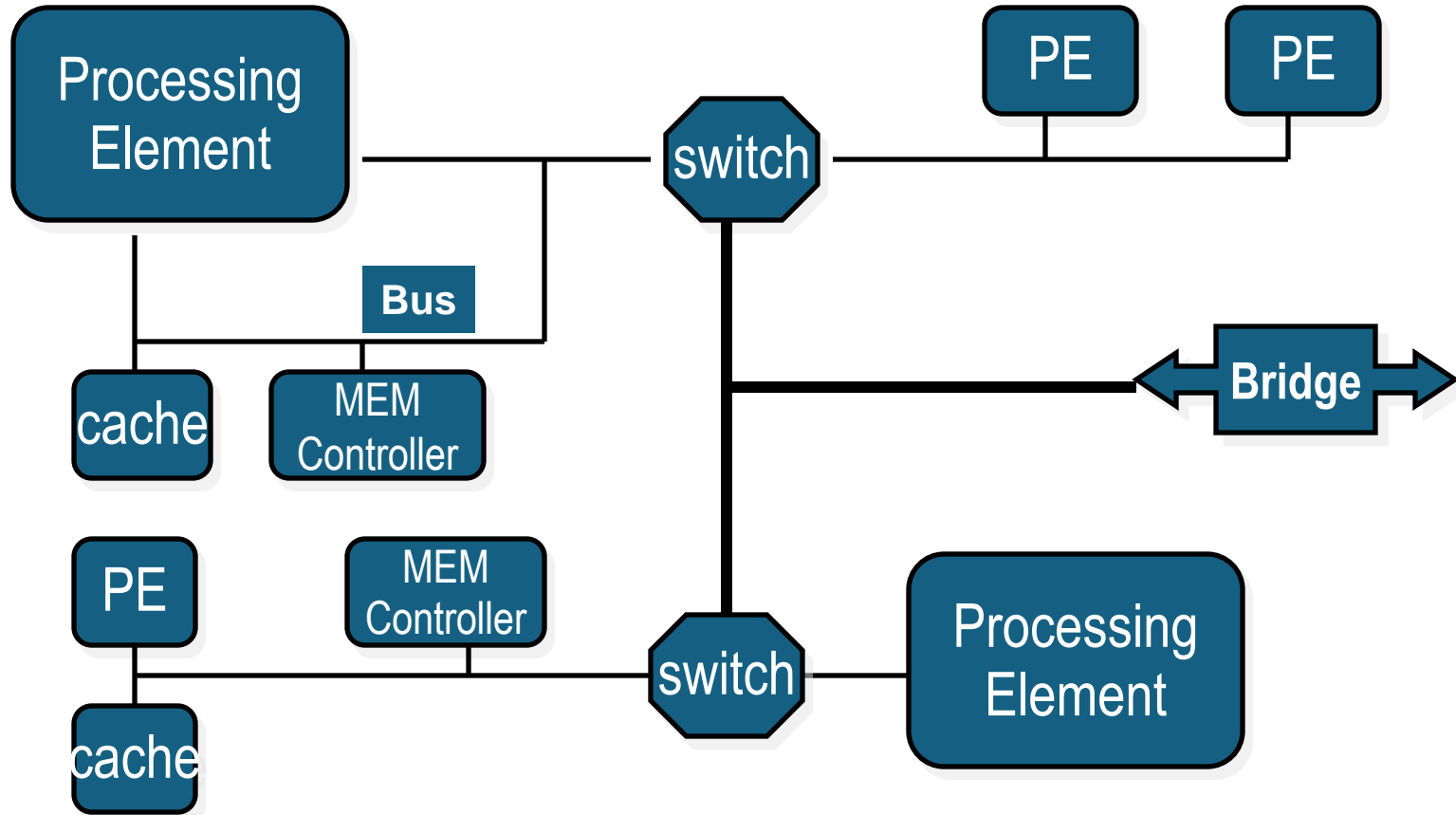
PEs may be CPUs or ASICs.

System Elements



PEs may be CPUs or ASICs.

System Elements



PEs may be CPUs or ASICs.

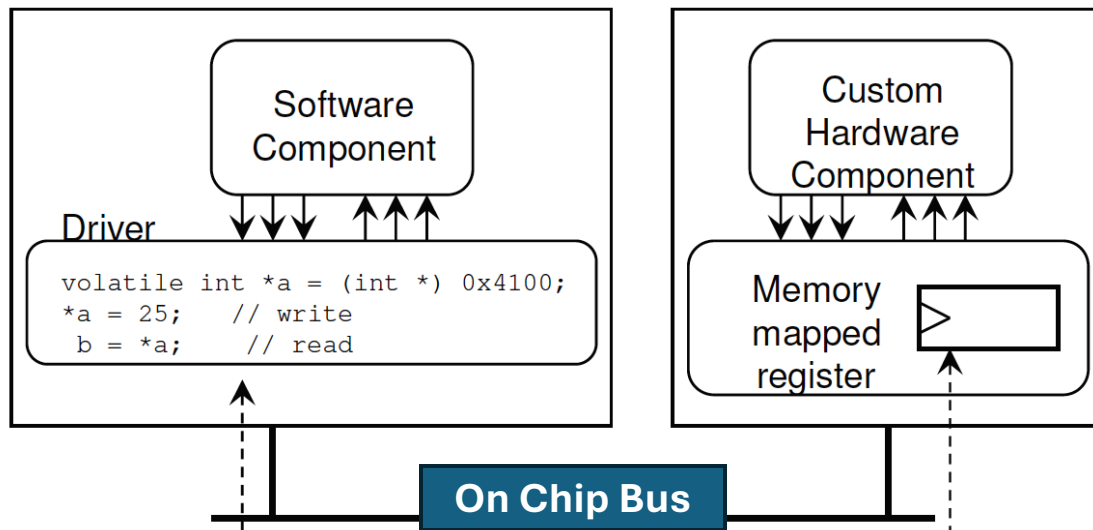
Memory Mapped Interfaces

Memory Mapped Interfaces

- Memory Mapped Register
 - Communicating hardware & software components
 - Register at the interface of the hardware component read/written from the software component
 - Attached to the on-chip bus

Memory Mapped Interfaces

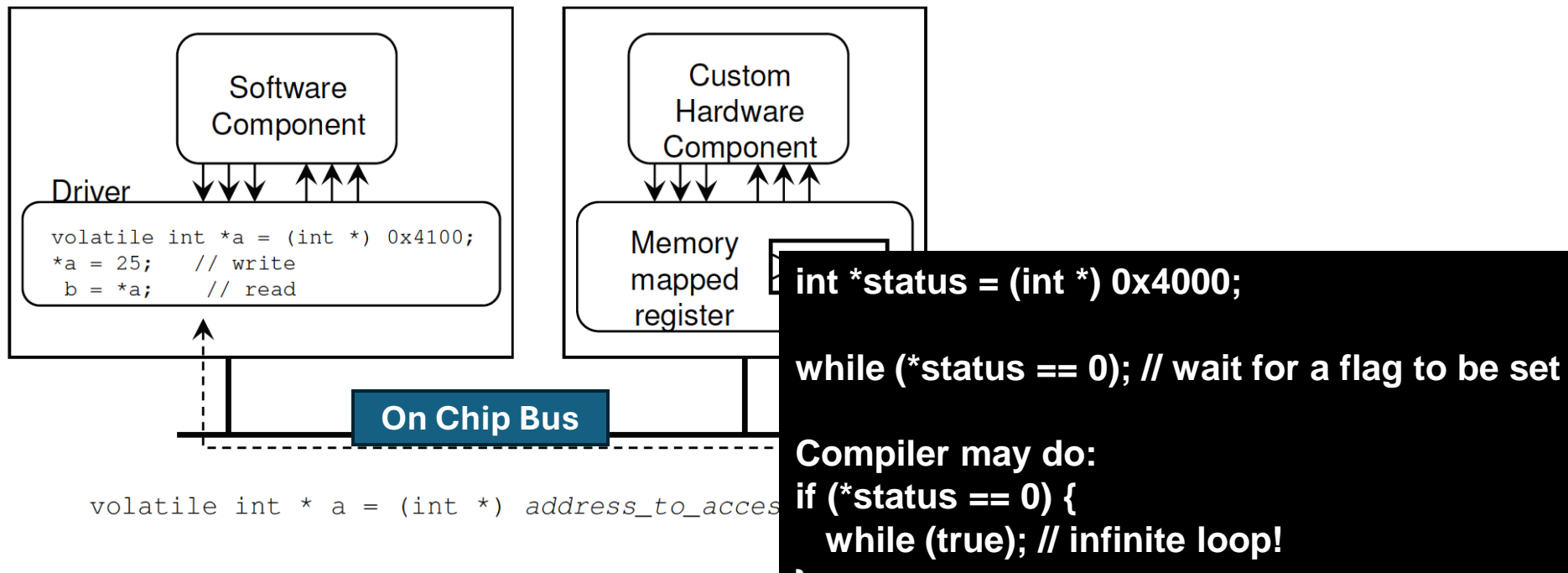
- Memory Mapped Register
 - Communicating hardware & software components
 - Register at the interface of the hardware component read/written from the software component
 - Attached to the on-chip bus



```
volatile int * a = (int *) address_to_access;
```

Memory Mapped Interfaces

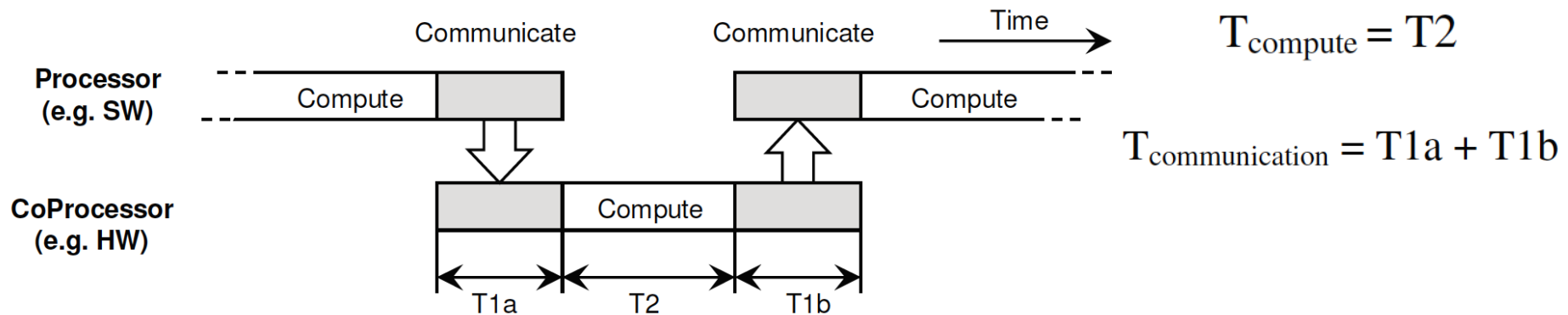
- Memory Mapped Register
 - Communicating hardware & software components
 - Register at the interface of the hardware component read/written from the software component
 - Attached to the on-chip bus



Tightly vs Loosely Coupled Design

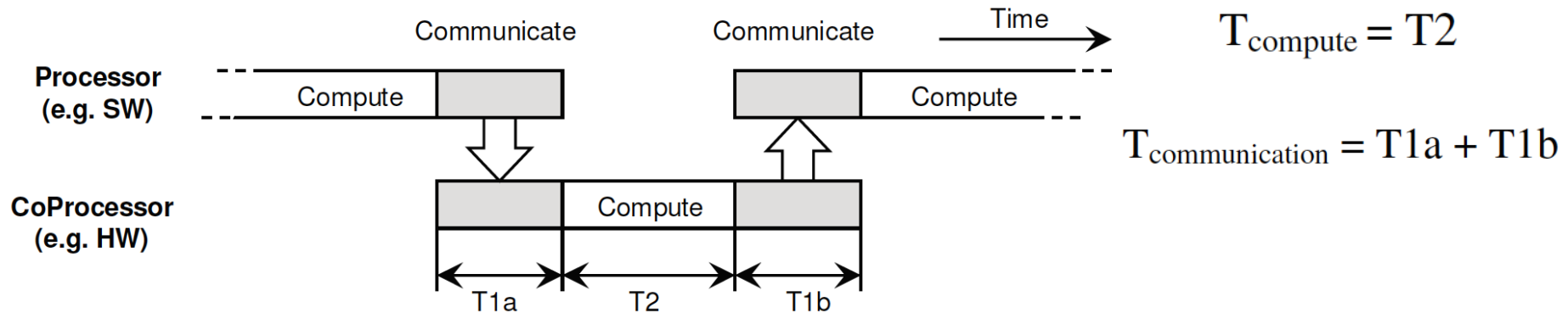
Tightly vs Loosely Coupled Design

- Computation to communication ratio affects the design choice



Tightly vs Loosely Coupled Design

- Computation to communication ratio affects the design choice



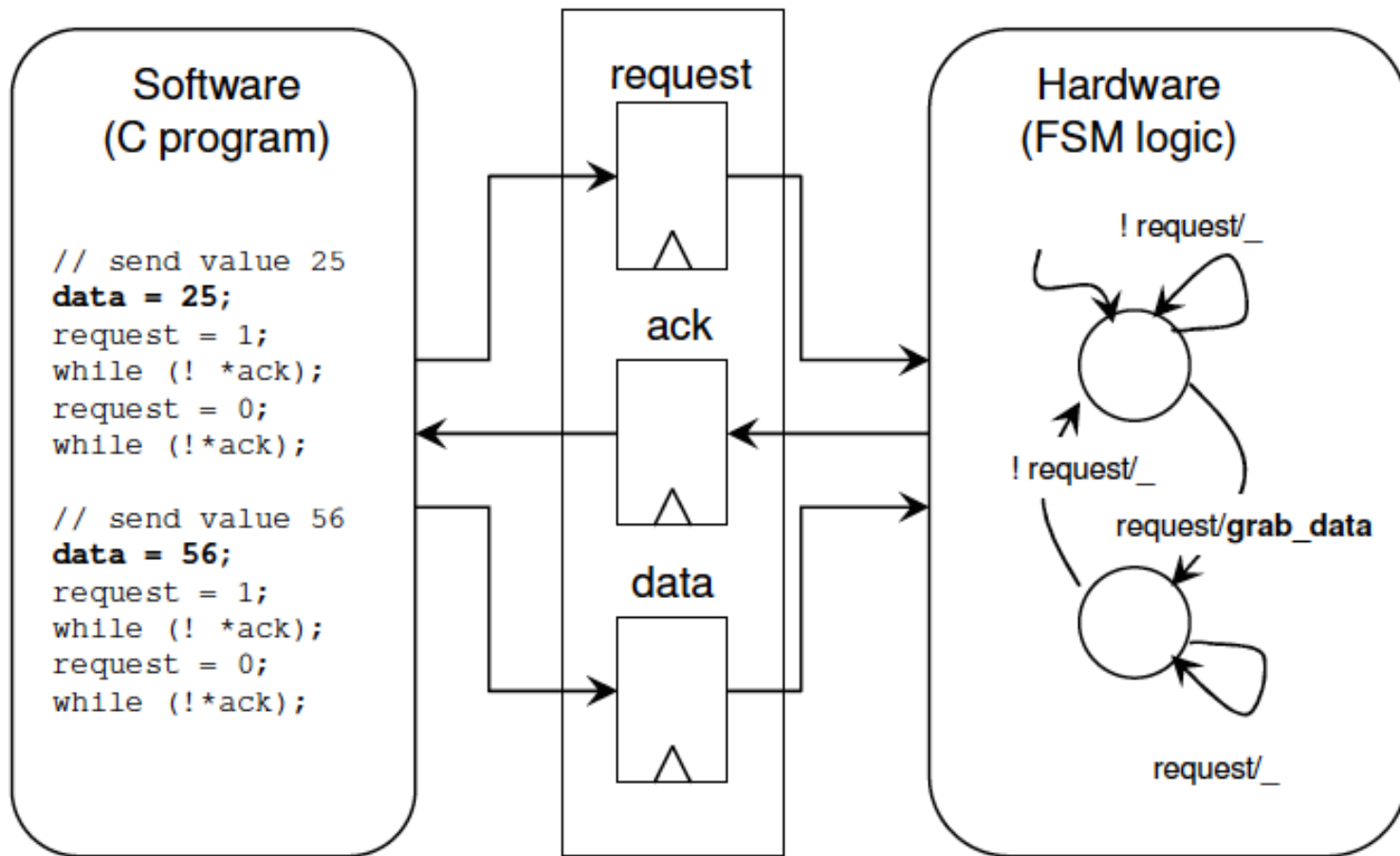
- If $T_{\text{compute}} / T_{\text{communication}} \gg 1 \rightarrow$ Loosely coupled design
- If $T_{\text{compute}} / T_{\text{communication}} \leq 1 \rightarrow$ Tightly coupled design

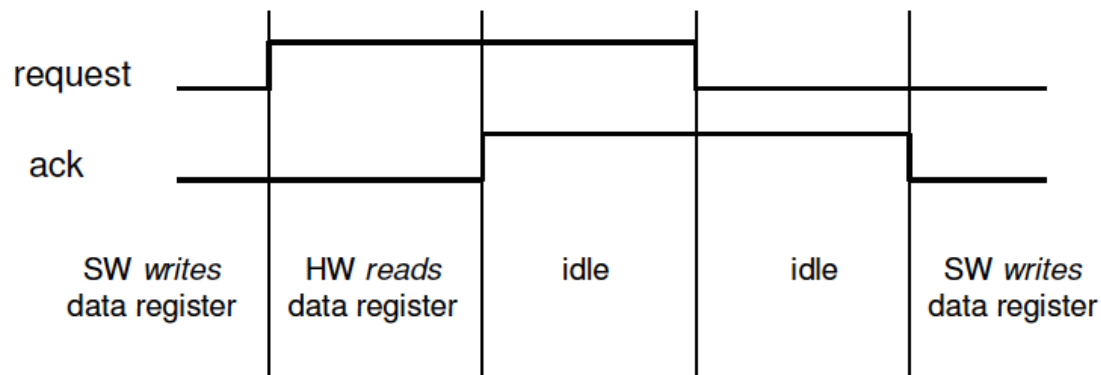
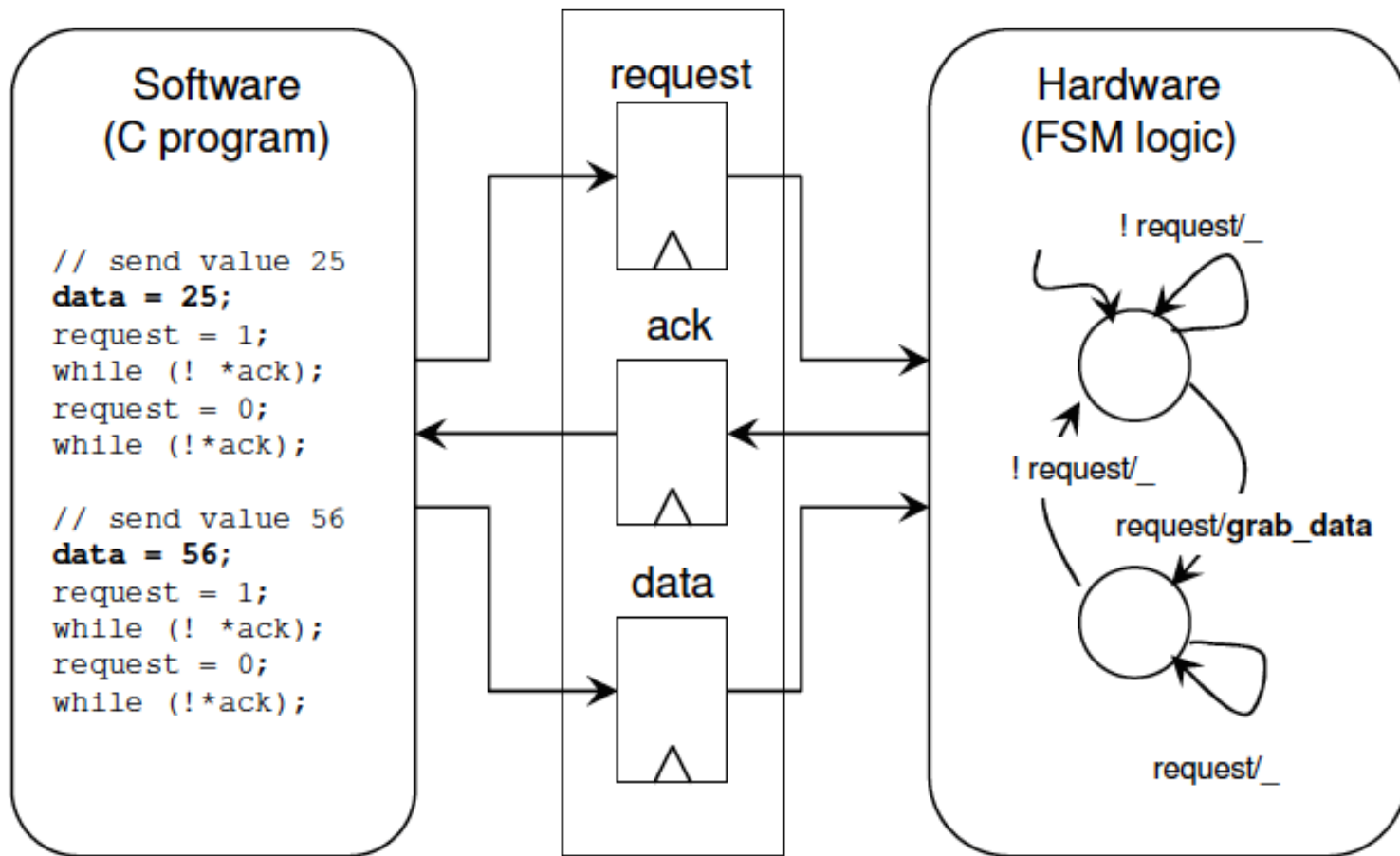
HW/SW Communication Protocol

- A problem with memory mapped registers
 - Custom HW cannot tell when SW will write into the register
 - SW cannot tell when custom HW will update the register

HW/SW Communication Protocol

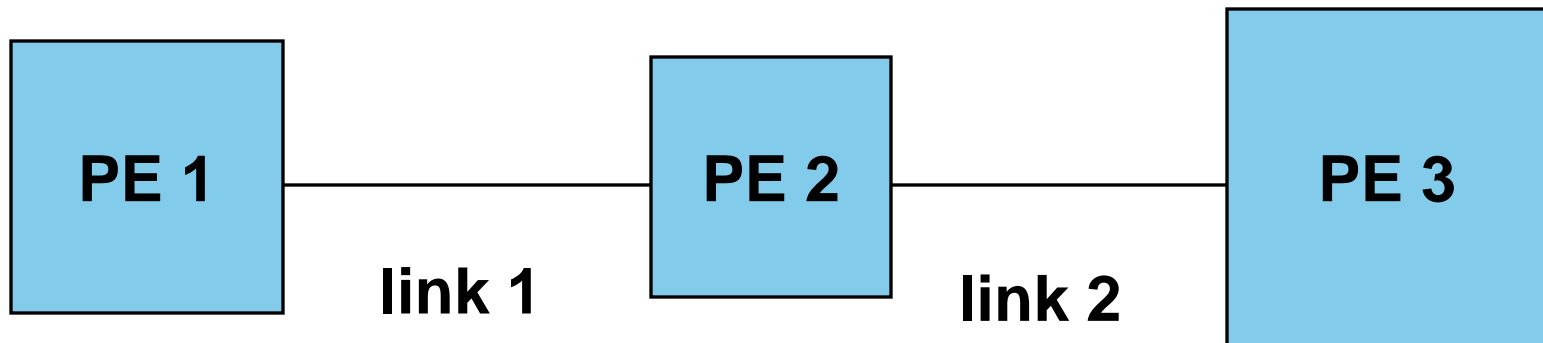
- A problem with memory mapped registers
 - Custom HW cannot tell when SW will write into the register
 - SW cannot tell when custom HW will update the register
- **Synchronize HW & SW through a protocol**
 - **Example: Mailbox register**





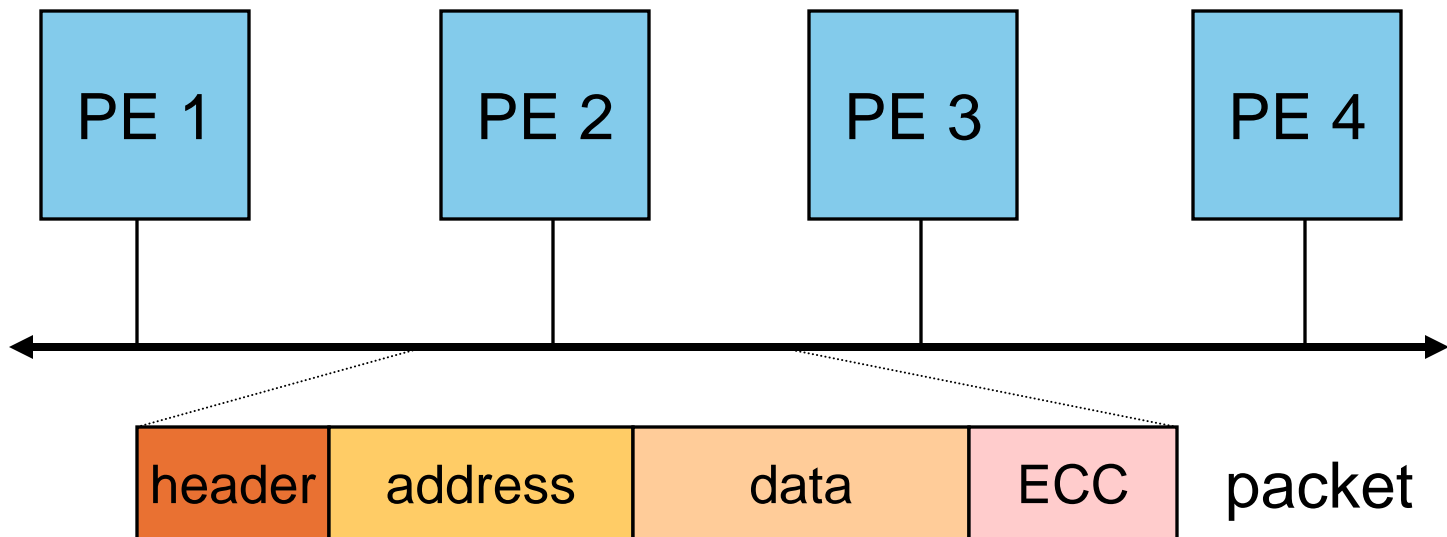
Point-to-point networks

- One source, one destination, no data switching
 - For example: RS232 serial port, IEEE 1284 parallel port



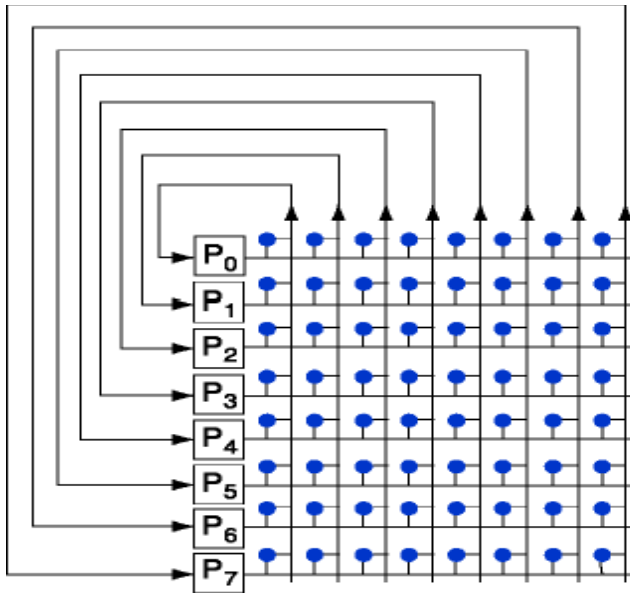
Bus networks

- Common physical connection:
 - Can be parallel or serial



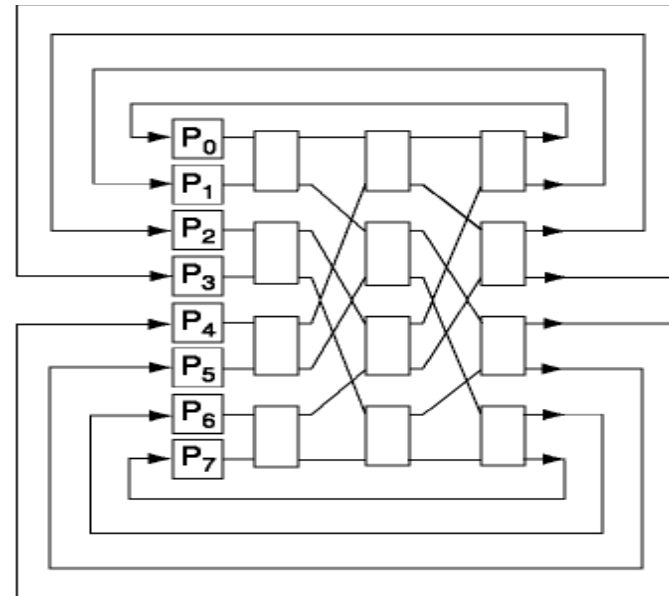
General Network

- Crossbar
 - Non-blocking.
 - Can handle arbitrary multi-cast combinations.
 - Size proportional to n^2 .



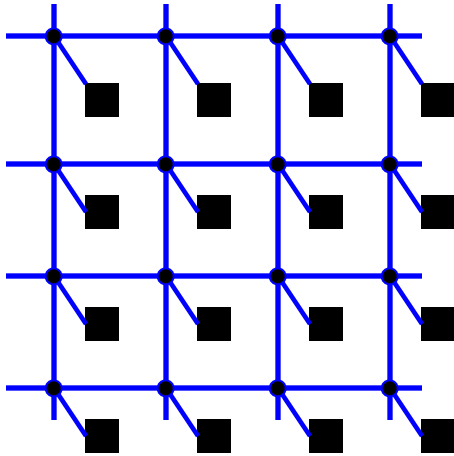
8x8 crossbar

- Multi-stage Network
 - Use several stages of switching elements.
 - Often blocking.
 - Often smaller than crossbar.

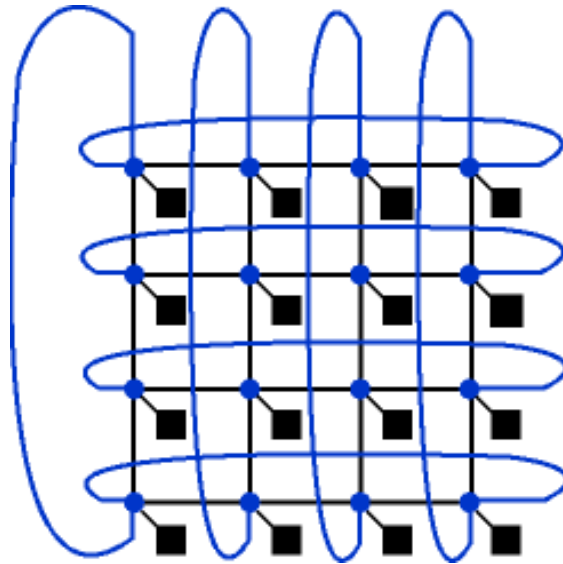


Multi-stage network

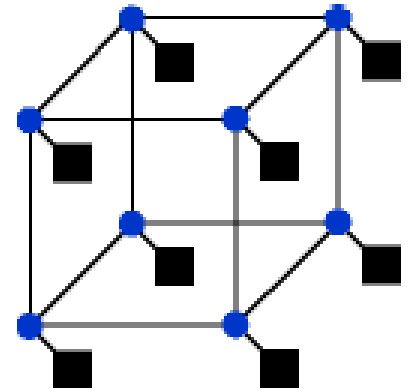
Other Network Topologies



2-D Mesh



2-D Torus

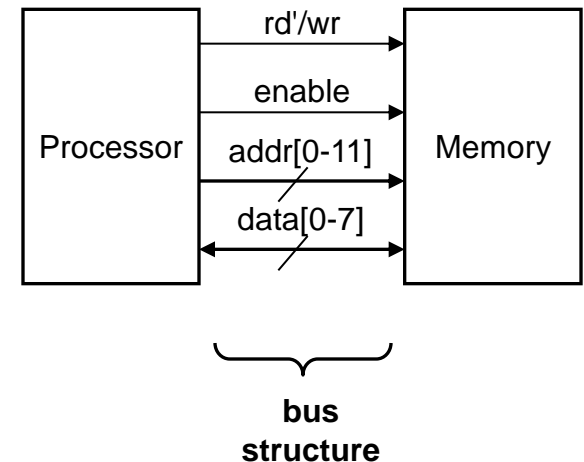


3-D Cube

Used to exist mostly in super computers, now getting into some high end embedded systems

A simple bus

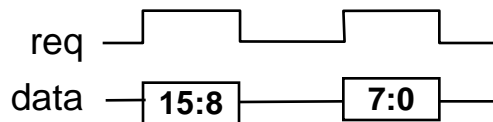
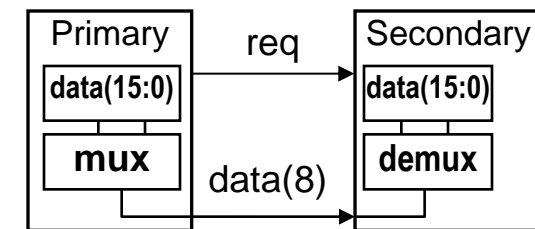
- Wires:
 - Uni-directional or bi-directional
- Bus
 - Set of wires with a single function
 - Address bus, data bus
 - Or, entire collection of wires
 - Address, data and control
 - Associated protocol: rules for communication



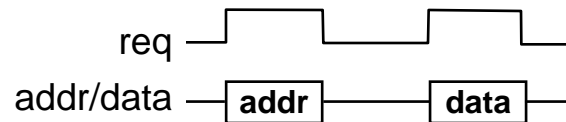
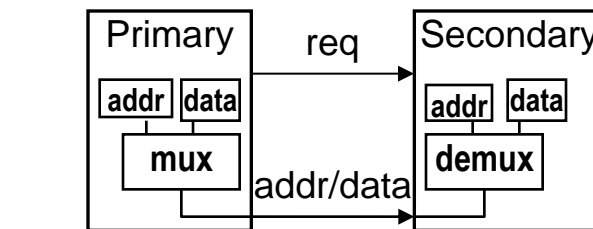
Basic protocol concepts

- Actor: *primary* initiates, *secondary* responds
- Direction: sender, receiver
- Addresses: special kind of data
 - Specifies a location in memory, a peripheral, or a register within a peripheral
- Time multiplexing
 - Share a single set of wires for multiple pieces of data
 - Saves wires at expense of time

**Time-multiplexed
data transfer**

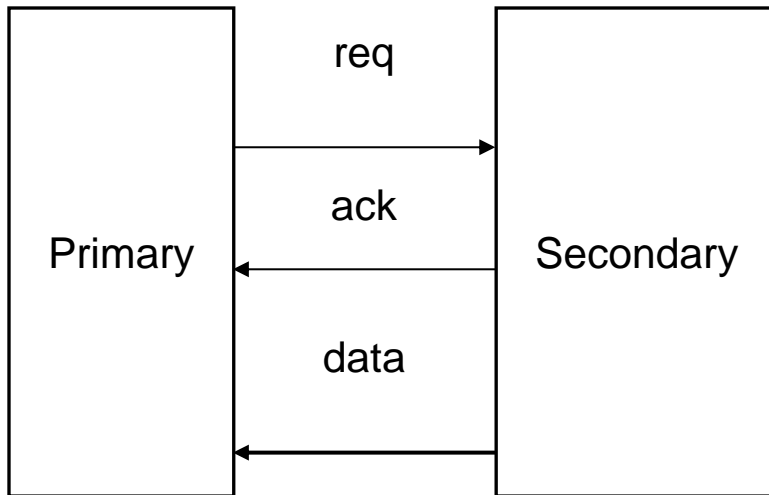


data serializing



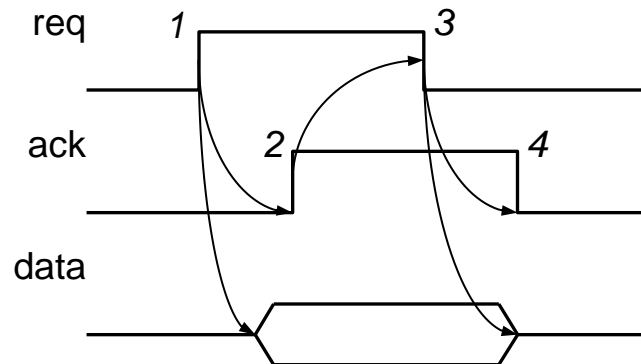
address/data muxing

Basic protocol concepts: control methods



1. Primary asserts *req* to receive data
2. Secondary puts data on bus
and asserts *ack*
3. Primary receives data and deasserts *req*
4. Secondary ready for next request

Handshake protocol

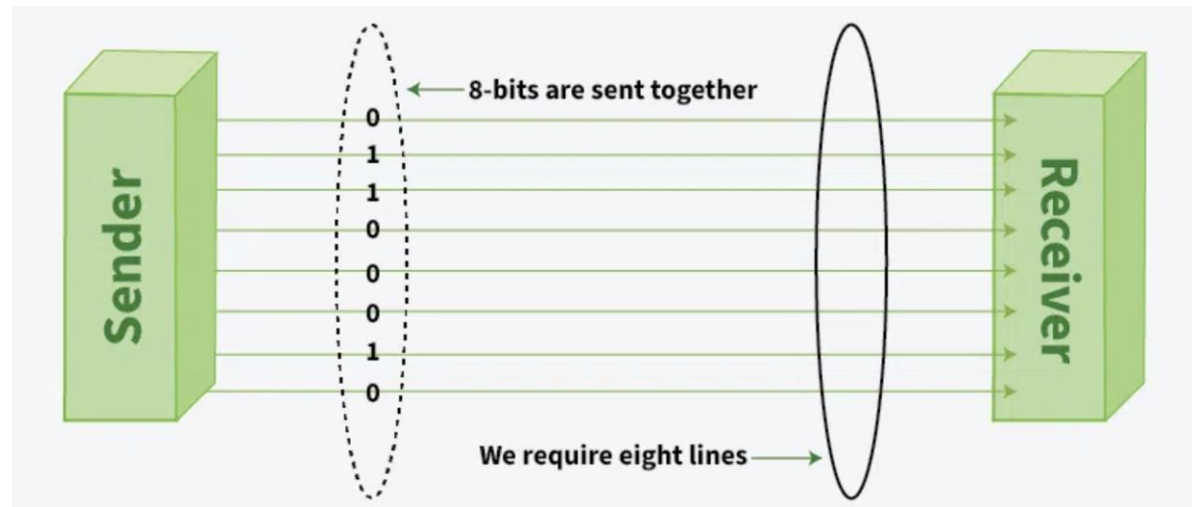


Classification based on the physical layer

Classification based on the physical layer

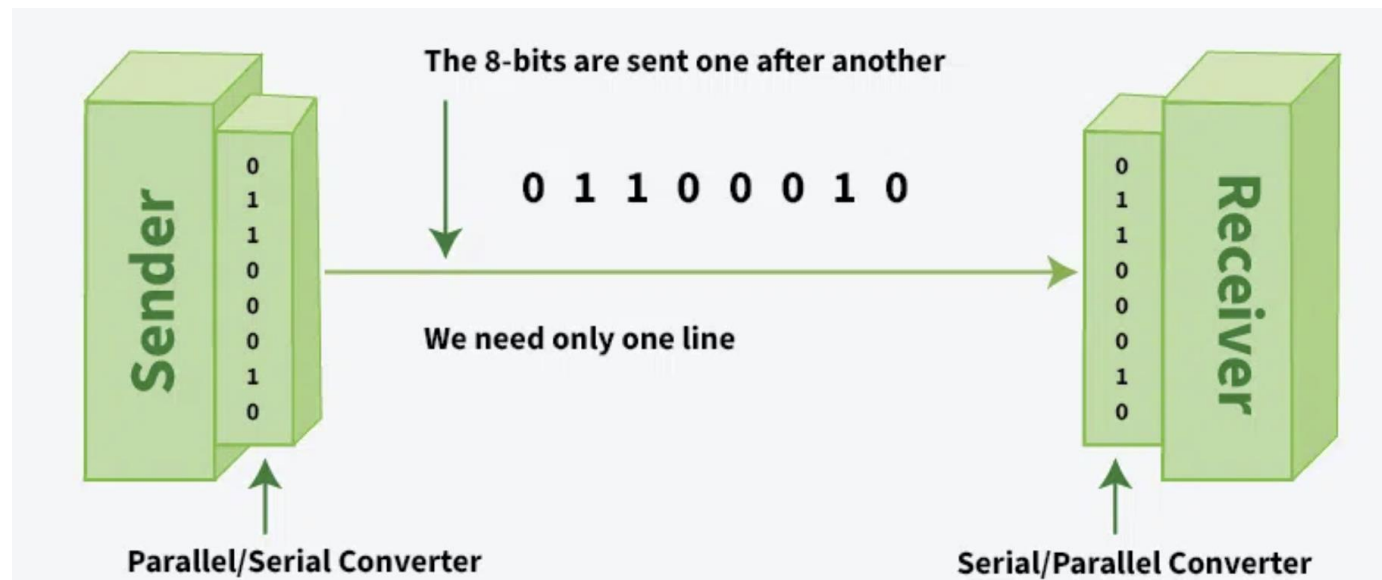
- **Parallel communication**

- Physical layer capable of transporting multiple bits of data



Classification based on the physical layer

- **Serial communication**
 - Physical layer transports one bit of data at a time



Classification based on the physical layer

- **Parallel communication**

- Physical layer capable of transporting multiple bits of data

- **Serial communication**

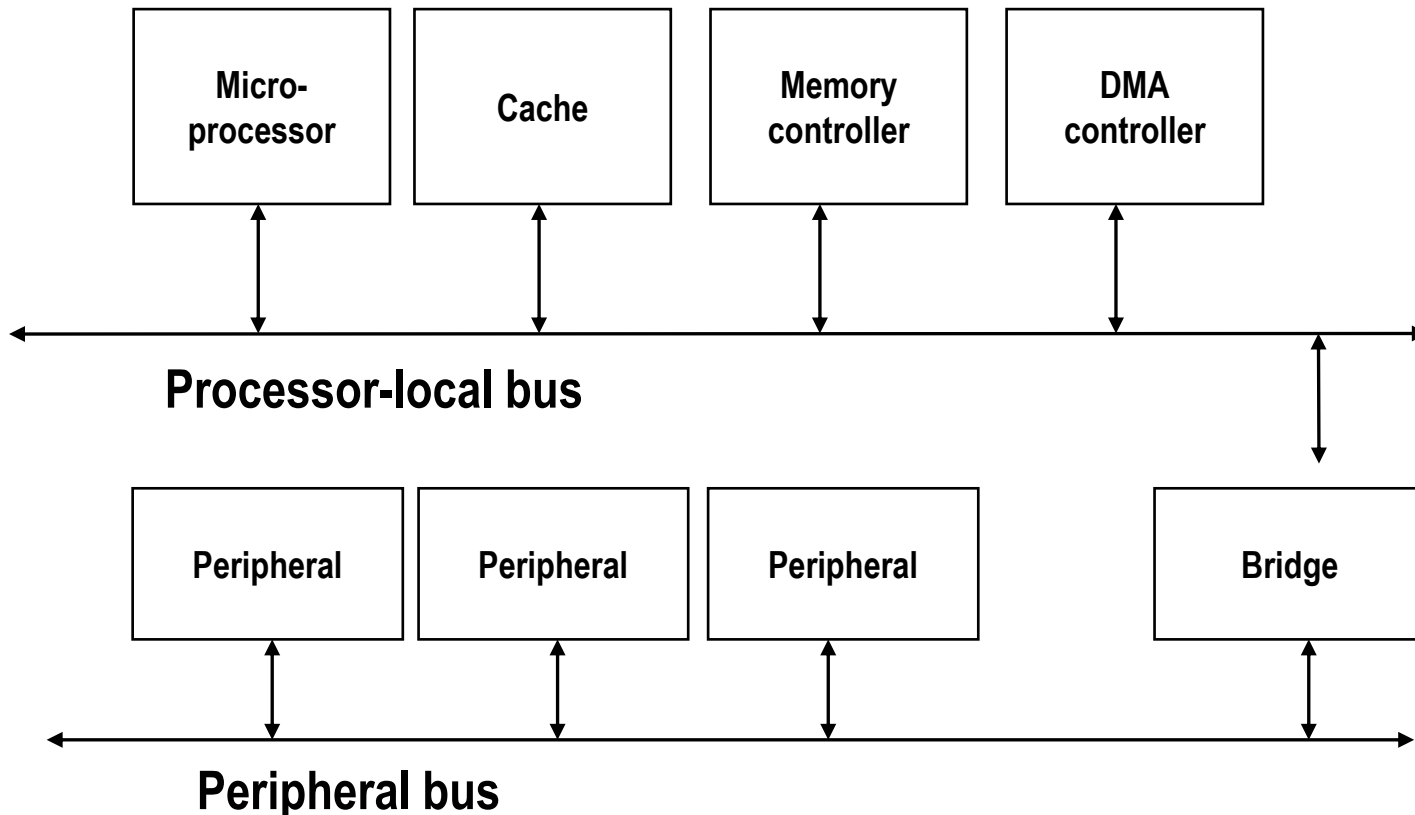
- Physical layer transports one bit of data at a time

- **Wireless communication**

- No physical connection needed for transport at physical layer

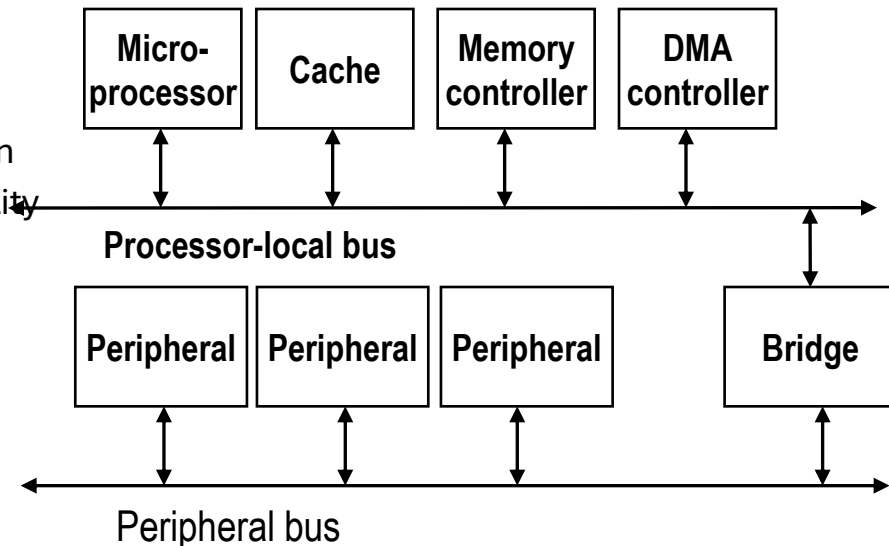
Multilevel bus architectures

- Don't want **one bus for all communication**
 - Peripherals would not need high-speed, processor-specific bus interface
 - excess gates, power consumption, and cost; less portable
 - Too many peripherals slows down bus



Multilevel bus architectures

- Don't want one bus for all communication
 - Peripherals would not need high-speed, processor-specific bus interface
 - excess gates, power consumption, and cost; less portable
 - Too many peripherals slows down bus
- Processor-local bus
 - High speed, wide, most frequent communication
- Peripheral bus
 - Lower speed, narrower, less frequent communication
 - Typically industry standard bus (ISA, PCI) for portability



Serial protocols: I²C

- I²C (Inter-IC)
 - Two-wire serial bus protocol developed by Philips 20 years ago
 - Enables low speed peripheral ICs to communicate using simple communication hardware
 - Data transfer rates up to 100 kbits/s and 7-bit addressing possible in normal mode
 - 3.4 Mb/s and 10-bit addressing in recent fast-mode
 - Common devices capable of interfacing to I²C bus:
 - EPROMS, Flash, LCD controller, some RAM memory, real-time clocks, watchdog timers, and microcontrollers