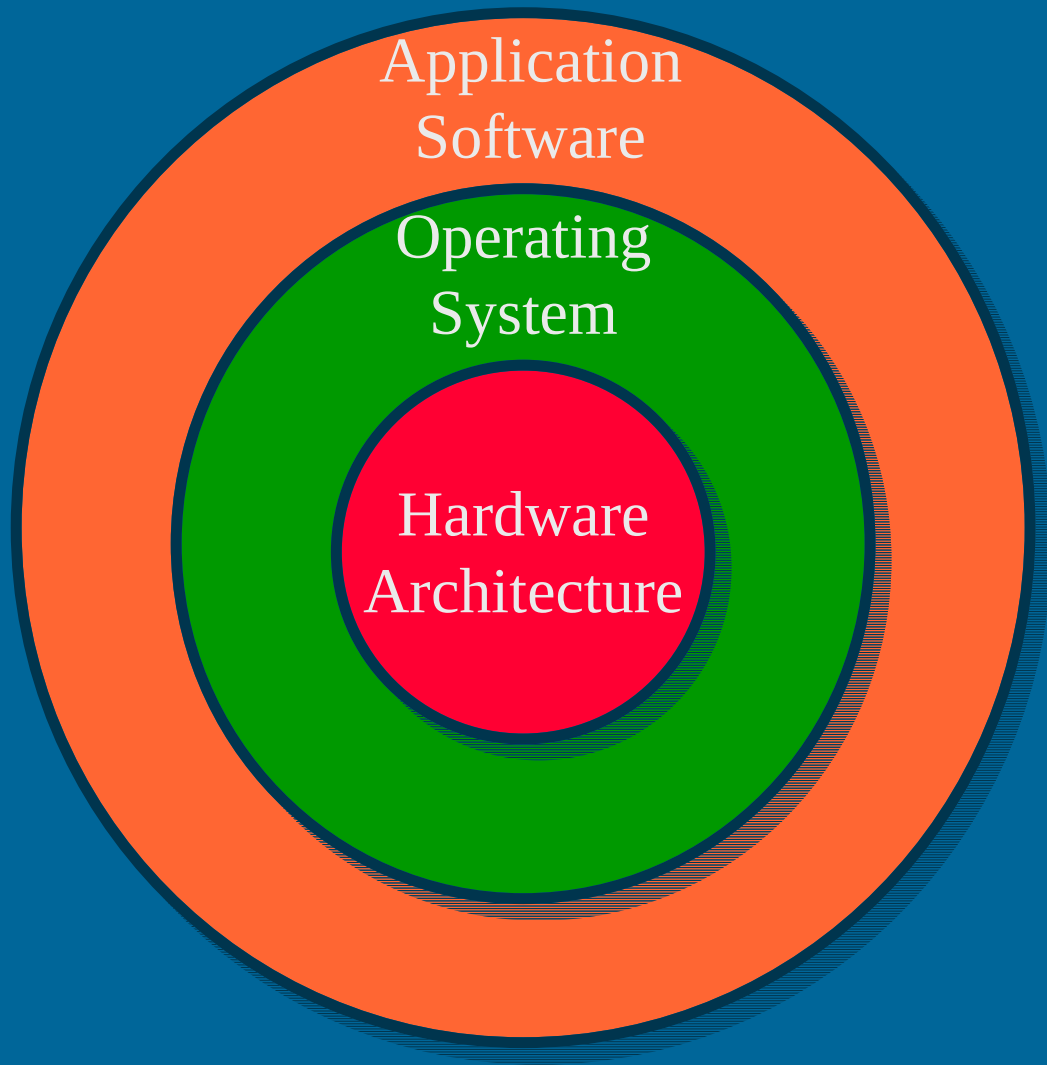
Advanced Computer Architecture

The Architecture of Parallel Computers

Computer Systems

No Component
Can be Treated
In Isolation
From the Others



Hardware Issues

- Number and Type of Processors
- Processor Control
- Memory Hierarchy
- I/O devices and Peripherals
- Operating System Support
- Applications Software Compatibility

Operating System Issues

- Allocating and Managing Resources
- Access to Hardware Features
  - Multi-Processing
  - Multi-Threading
- I/O Management
- Access to Peripherals
- Efficiency

Applications Issues

- Compiler/Linker Support
- Programmability
- OS/Hardware Feature Availability
- Compatibility
- Parallel Compilers
  - Preprocessor
  - Precompiler
  - Parallelizing Compiler

Architecture Evolution

- Scalar Architecture
- Prefetch Fetch/Execute Overlap
- Multiple Functional Units
- Pipelining
- Vector Processors
- Lock-Step Processors
- Multi-Processor

Flynn's Classification

- Consider Instruction Streams and Data Streams Separately.
- SISD - Single Instruction, Single Data Stream
- SIMD - Single Instruction, Multiple Data Streams
- MIMD - Multiple Instruction, Multiple Data Streams.
- MISD - (rare) Multiple Instruction, Single Data Stream

SISD

- Conventional Computers.
- Pipelined Systems
- Multiple-Functional Unit Systems
- Pipelined Vector Processors
- Includes most computers encountered in everyday life

SIMD

- Multiple Processors Execute a Single Program
- Each Processor operates on its own data
- Vector Processors
- Array Processors
- PRAM Theoretical Model

MIMD

- Multiple Processors cooperate on a single task
- Each Processor runs a different program
- Each Processor operates on different data
- Many Commercial Examples Exist

MISD

- A Single Data Stream passes through multiple processors
- Different operations are triggered on different processors
- Systolic Arrays
- Wave-Front Arrays

Programming Issues

- Parallel Computers are Difficult to Program
- Automatic Parallelization Techniques are only Partially Successful
- Programming languages are few, not well supported, and difficult to use.
- Parallel Algorithms are difficult to design.

Performance Issues

- Clock Rate / Cycle Time = τ
- Cycles Per Instruction (Average) = CPI
- Instruction Count = I_c
- Time, $T = I_c \times CPI \times \tau$
- p = Processor Cycles, m = Memory Cycles, k = Memory/Processor cycle ratio
- $T = I_c \times (p + m \times k) \times \tau$

Performance Issues II

- Ic & p affected by processor design and compiler technology.
- m affected mainly by compiler technology

  τ affected by processor design
- k affected by memory hierarchy structure and design

Other Measures

- MIPS rate - Millions of instructions per second
- Clock Rate for similar processors
- MFLOPS rate - Millions of floating point operations per second.
- These measures are not neccessarily directly comparable between different types of processors.

Parallelizing Code

- Implicitly
  - Write Sequential Algorithms
  - Use a Parallelizing Compiler
  - Rely on compiler to find parallelism
- Explicitly
  - Design Parallel Algorithms
  - Write in a Parallel Language
  - Rely on Human to find Parallelism

Multi-Processors

- Multi-Processors generally share memory, while multi-computers do not.
  - Uniform memory model
  - Non-Uniform Memory Model
  - Cache-Only
- MIMD Machines

Multi-Computers

- Independent Computers that Don’t Share Memory.
- Connected by High-Speed Communication Network
- More tightly coupled than a collection of independent computers
- Cooperate on a single problem

Vector Computers

- Independent Vector Hardware
- May be an attached processor
- Has both scalar and vector instructions
- Vector instructions operate in highly pipelined mode
- Can be Memory-to-Memory or Register-to-Register

SIMD Computers

- One Control Processor
- Several Processing Elements
- All Processing Elements execute the same instruction at the same time
- Interconnection network between PEs determines memory access and PE interaction

The PRAM Model

- SIMD Style Programming
- Uniform Global Memory
- Local Memory in Each PE
- Memory Conflict Resolution
  - CRCW - Common Read, Common Write
  - CREW - Common Read, Exclusive Write
  - EREW - Exclusive Read, Exclusive Write
  - ERCW - (rare) Exclusive Read, Common Write

The VLSI Model

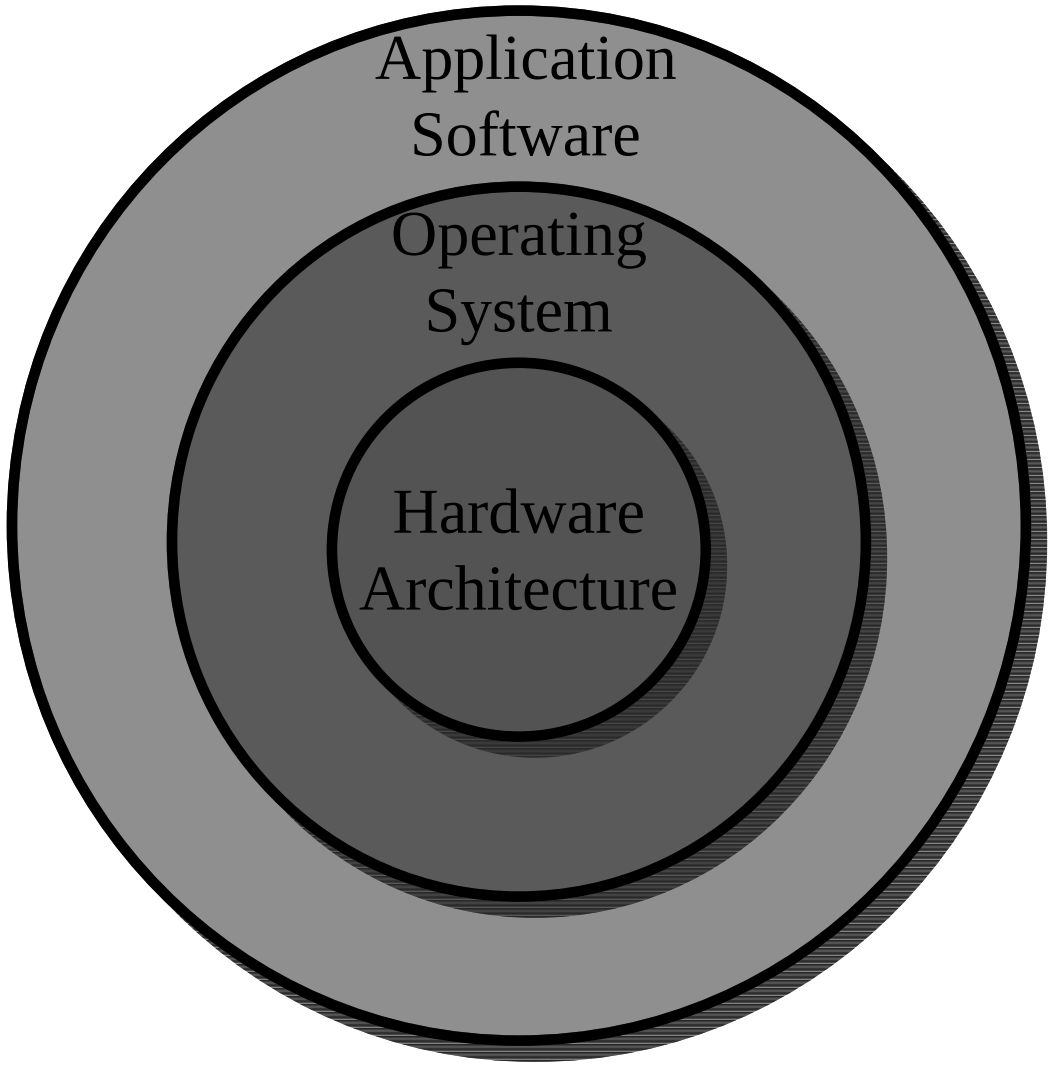
- Implement Algorithm as a mostly combinational circuit
- Determine the area required for implementation
- Determine the depth of the circuit

Advanced Computer Architecture

The Architecture of Parallel Computers

Computer Systems

No Component
Can be Treated
In Isolation
From the Others



Hardware Issues

- Number and Type of Processors
- Processor Control
- Memory Hierarchy
- I/O devices and Peripherals
- Operating System Support
- Applications Software Compatibility

Operating System Issues

- Allocating and Managing Resources
- Access to Hardware Features
  - Multi-Processing
  - Multi-Threading
- I/O Management
- Access to Peripherals
- Efficiency

Applications Issues

- Compiler/Linker Support
- Programmability
- OS/Hardware Feature Availability
- Compatibility
- Parallel Compilers
  - Preprocessor
  - Precompiler
  - Parallelizing Compiler

Architecture Evolution

- Scalar Architecture
- Prefetch Fetch/Execute Overlap
- Multiple Functional Units
- Pipelining
- Vector Processors
- Lock-Step Processors
- Multi-Processor

Flynn's Classification

- Consider Instruction Streams and Data Streams Separately.
- SISD - Single Instruction, Single Data Stream
- SIMD - Single Instruction, Multiple Data Streams
- MIMD - Multiple Instruction, Multiple Data Streams.
- MISD - (rare) Multiple Instruction, Single Data Stream

SISD

- Conventional Computers.
- Pipelined Systems
- Multiple-Functional Unit Systems
- Pipelined Vector Processors
- Includes most computers encountered in everyday life

SIMD

- Multiple Processors Execute a Single Program
- Each Processor operates on its own data
- Vector Processors
- Array Processors
- PRAM Theoretical Model

MIMD

- Multiple Processors cooperate on a single task
- Each Processor runs a different program
- Each Processor operates on different data
- Many Commercial Examples Exist

MISD

- A Single Data Stream passes through multiple processors
- Different operations are triggered on different processors
- Systolic Arrays
- Wave-Front Arrays

Programming Issues

- Parallel Computers are Difficult to Program
- Automatic Parallelization Techniques are only Partially Successful
- Programming languages are few, not well supported, and difficult to use.
- Parallel Algorithms are difficult to design.

Performance Issues

- Clock Rate / Cycle Time = τ
- Cycles Per Instruction (Average) = CPI
- Instruction Count = I_c
- Time, $T = I_c \times CPI \times \tau$
- p = Processor Cycles, m = Memory Cycles, k = Memory/Processor cycle ratio
- $T = I_c \times (p + m \times k) \times \tau$

Performance Issues II

- Ic & p affected by processor design and compiler technology.
- m affected mainly by compiler technology

  τ affected by processor design
- k affected by memory hierarchy structure and design

Other Measures

- MIPS rate - Millions of instructions per second
- Clock Rate for similar processors
- MFLOPS rate - Millions of floating point operations per second.
- These measures are not neccessarily directly comparable between different types of processors.

Parallelizing Code

- Implicitly
  - Write Sequential Algorithms
  - Use a Parallelizing Compiler
  - Rely on compiler to find parallelism
- Explicitly
  - Design Parallel Algorithms
  - Write in a Parallel Language
  - Rely on Human to find Parallelism

Multi-Processors

- Multi-Processors generally share memory, while multi-computers do not.
  - Uniform memory model
  - Non-Uniform Memory Model
  - Cache-Only
- MIMD Machines

Multi-Computers

- Independent Computers that Don’t Share Memory.
- Connected by High-Speed Communication Network
- More tightly coupled than a collection of independent computers
- Cooperate on a single problem

Vector Computers

- Independent Vector Hardware
- May be an attached processor
- Has both scalar and vector instructions
- Vector instructions operate in highly pipelined mode
- Can be Memory-to-Memory or Register-to-Register

SIMD Computers

- One Control Processor
- Several Processing Elements
- All Processing Elements execute the same instruction at the same time
- Interconnection network between PEs determines memory access and PE interaction

The PRAM Model

- SIMD Style Programming
- Uniform Global Memory
- Local Memory in Each PE
- Memory Conflict Resolution
  - CRCW - Common Read, Common Write
  - CREW - Common Read, Exclusive Write
  - EREW - Exclusive Read, Exclusive Write
  - ERCW - (rare) Exclusive Read, Common Write

The VLSI Model

- Implement Algorithm as a mostly combinational circuit
- Determine the area required for implementation
- Determine the depth of the circuit