COMPUTER SYSTEM DESIGN

GENERAL DESIGN ISSUES

Objectives

- Meaning of computer system design
- Description of computer design trends
- Analysis of qualitative and quantitative design aspects

Contents

- What is computer architecture
- Technological trends
- Power and Energy in ICs
- Cost
- Quantitative Principles of Computer Design
- Qualitative Aspects of Computer Design

- We want to design our computer at MUST!!

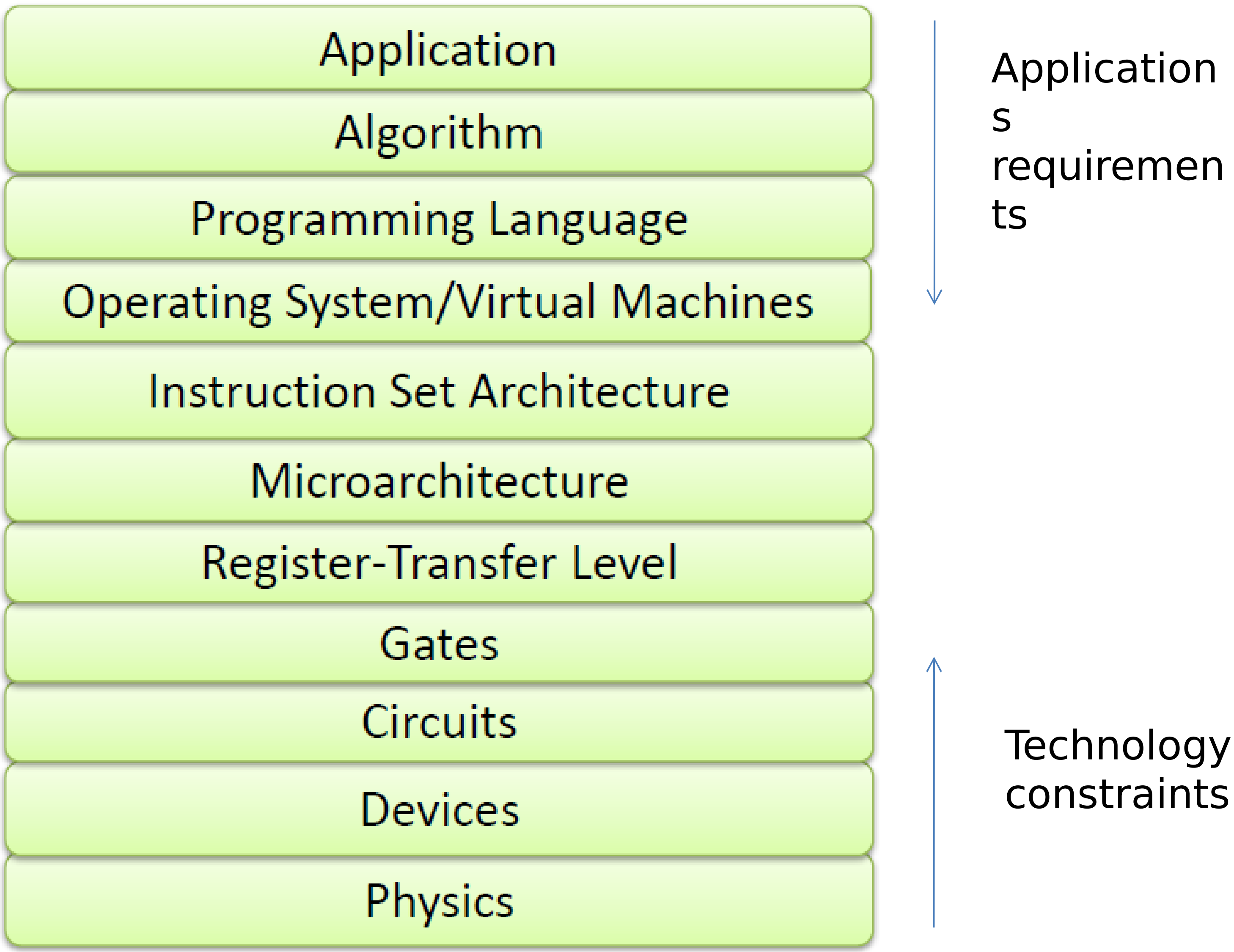
- What are we going to do?


PROJECT
Conception & Initiation
1
PROJECT
Definition & Planning
2
PROJECT
Launch or Execution
3
PROJECT
Performance & Control
4
PROJECT
Project Close
5
Project Charter
Project Initiation
Scope & Budget
Work Breakdown Schedule
Gantt Chart
Communication Plan
Risk Management
Status & Tracking
KPIs
Quality
Forecasts
Objectives
Quality Deliverables
Effort & Cost Tracking
Performance
Post mortem
Project Punchlist
Reporting

What is computer architecture

- A computer designer faces a complex task
- Has to determine what attributes are important for a new computer
- Then design a computer to maximize
  - Performance
  - Energy efficiency
- While staying within three constraints
  - Cost
  - Power
  - Availability

A computer system- Abstract



- This task has many aspects
  - Instruction set design
  - Functional organization
  - Logic design
  - implementation.
    - Integrated circuit design
    - Packaging
    - Power
    - Cooling

- Optimizing the design requires familiarity with a very wide range of technologies
  - From compilers and operating systems to logic design and packaging.

- Several years ago, the term computer architecture often referred only to instruction set design

But …

- In its broadest definition, computer architecture is the design of the abstraction/implementation layers that allow us to execute information processing applications efficiently using manufacturing technologies

- **Thus,** a computer architect has to specify the performance requirements of various parts of a computer system, to define the interconnections between them, and to keep it harmoniously balanced.

- The computer architect's job is more than designing the Instruction Set, as it has been understood for many years.
- The more an architect is exposed to all aspects of computer design, the more efficient she will be.

- The implementation of a computer has two components: **organization** and **hardware.**
- The term organization (Microarchitecture) includes aspects such as:
  - The memory system
  - The memory interconnect
  - The design of the internal processor or CPU

- AMD Opteron and the Intel Core i7
  - Both processors implement the x86 instruction set
  - But they have very different pipeline and cache organizations.

- Hardware refers to the specifics of a computer, including the detailed **logic design** and the **packaging technology** of the computer

- Often a line of computers contains computers with identical ISA and nearly identical organizations, but they differ in the **detailed hardware implementation**.

- Computer architects must design a computer to meet **functional requirements** as well as price, power, performance, and availability goals.

- The requirements may be specific features inspired by the market.
- Application software often drives the choice of certain functional requirements by determining how the computer will be used

- Architects must also be aware of important trends in both the technology and the use of computers
- These trends affect the future cost and the longevity of an architecture.

Trends in Technology

- To plan for the evolution of a computer, the designer must be aware of rapid changes in implementation technology.
- Five critical implementation technologies, which change at a dramatic pace:
    1. Integrated circuit logic technology
    2. Semiconductor DRAM
    3. Semiconductor Flash (EEPROM)
    4. Magnetic disk technology
    5. Network technology

- **Integrated circuit logic technology**
  - Transistor density increases by about 35% per year
  - Increases in die size are less predictable and slower, ranging from 10% to 20% per year
    - The die or processor die is a rectangular pattern on a wafer that contains circuitry to perform a specific function
  - The combine effect is a growth rate in transistor count on a chip of about 40% to 55% per year, or doubling every 18 to 24 months.
  - Moore’s law is dead (Various)

NB

- Moore's law is a term used to refer to the observation made by Gordon Moore in 1965 that the number of transistors in a dense integrated circuit (IC) doubles about every two years.

- **Semiconductor DRAM**
- Capacity per DRAM chip has been reported to increase by about 25% to 40% per year in recent years.
  - Doubling roughly every two to three years.
- There is even concern as whether the growth rate will stop sometimes soon.

- Semiconductor Flash (EEPROM)
- This is the standard storage device in PMDs
- Capacity per Flash chip has increased by about 50% to 60% per year recently
- Flash memory is many times cheaper per bit than DRAM.
  - A serious consideration

- Magnetic disk technology
  - Disks are significantly cheaper per bit than Flash
  - Disks are now more than 300 times cheaper per bit than DRAM
  - This technology is central to server and warehouse scale storage

- Network technology
  - Network performance depends both on the performance of switches and on the performance of the transmission system

- These rapidly changing technologies shape the design of a computer that, with speed and technology enhancements, may have a lifetime of three to five years.

END OF PART 1

Performance Trends

- **Bandwidth over Latency**
- Bandwidth or throughput is the total amount of work done in a given time
  - Such as megabytes per second for a disk transfer
- Latency or response time is the time between the start and the completion of an event
  - Such as milliseconds for a disk access

- A simple rule of thumb is that bandwidth grows by **at least the square of the improvement in latency.**
- Computer designers should plan accordingly. How???

Scaling of Transistor Performance

- Integrated circuit processes are characterized by the feature size:
- The minimum size of a transistor or a wire in either the x or y dimension.
- Feature sizes have decreased from 10 microns in 1971 to less than 22nm today
    - **1microns = 0.001nm**

- **Wires** in an IC do not improve in performance with decreased feature size.
- The signal delay for a wire increases in proportion to the product of its resistance and capacitance.
  - As feature size shrinks, wires get shorter, but the resistance and capacitance per unit length get worse

- **Wire delay** scales poorly compared to transistor performance, creating additional challenges for the designer.
- In addition to the power dissipation limit, wire delay has become a major design limitation for large ICs
  - Often more critical than transistor switching delay.

- Larger and larger fractions of the clock cycle have been consumed by the propagation delay of signals on wires

Trends in Power and Energy in Integrated Circuits

- Power is the biggest challenge facing the computer designer for nearly every class of computer
- Power must be brought in and distributed around the chip
- Modern microprocessors use hundreds of pins and multiple interconnect layers just for power and ground

- Power is dissipated as heat and must be removed.

Power and Energy

- From the viewpoint of a system designer there are three primary concerns

1. What is the maximum power a processor ever requires

2. What is the sustained power consumption?

- **thermal design power (TDP) != peak power != actual average power**
- A typical power supply for a system is usually sized to exceed the TDP, and a cooling system is usually designed to match or exceed TDP
- Failure to provide adequate cooling will allow the junction temperature in the processor to exceed its maximum value, resulting in device failure and possibly permanent damage.

- Modern solutions:
- Circuitry reduces the **clock rate**, thereby reducing power
- A second thermal overload trip is activated to power down the chip.

3. Energy and Energy efficiency.

- Power is simply energy per unit time
  - 1 watt = 1 joule per second.
- Which metric is the right one for comparing processors?
- Energy is always a better metric because it is tied to a specific task and the time required for that task.

- **For example**,
- Processor A may have a 20% higher average power consumption than processor B
- But if A executes the task in only 70% of the time needed by B
- Its energy consumption will be $1.2 \times 0.7 = 0.84$, which is clearly better.
  - If our cloud were populated with processor As rather than Bs, then the cloud would do less work for the same amount of energy expended.

- Whenever we have a fixed workload:
- ✓ Comparing energy will be the right way to compare processor alternatives
- ✓ Electricity bill or battery lifetime are both determined by the energy consumed.

- The first microprocessors consumed less than a watt and the first 32-bit microprocessors (like the Intel 80386) used about 2 watts, while a 3.3 GHz Intel Core i7 consumes 130 watts.
- Given that this heat must be dissipated from a chip that is about 1.5 cm on a side,
- **We have reached the limit of what can be cooled by air**.

- Power is now the major constraint to using transistors.

- Modern microprocessors offer many techniques to try to improve energy efficiency despite flat clock rates and constant supply voltages:

- Do nothing well - turn off the clock of inactive modules to save energy
- Dynamic Voltage-Frequency Scaling (DVFS) - offer a few clock frequencies and voltages during periods of low activity

- Design for typical case

✓ DRAMs have a series of increasingly lower power modes to extend battery life in PMDs and laptops (when idle)
  - The problem is you must return to fully active mode to read or write, no matter how low the access rate.

✓ Relying on on-chip temperature sensors to detect when activity should be reduced automatically to avoid overheating.

- Overclocking
- The chip decides that it is safe to run at a higher clock rate for a short time possibly on just a few cores until temperature starts to rise
  - The 3.3 GHz Core i7 can run in short bursts for 3.6 GHz.
  - For single threaded code, these microprocessors can turn off all cores **but one** and run it at an even higher clock rate.

Static power

- It is proportional to number of devices
- Dissipated by leakage currents that flow even when the device is inactive.
- Proportional to $\text{Current}_{\text{static}} \times \text{Voltage}$
- Thus, increasing the number of transistors increases power even if they are idle.
- The only hope to stop leakage is to turn off power to subsets of the chips.

- **Race-to-halt :**
- The processor is just a portion of the whole energy cost of a system
- Use a faster, less energy-efficient processor to allow the rest of the system to go into a sleep mode.

WE STOP HERE – NOV 19-2021

Trends in Cost

- Cost-sensitive designs are of growing significance
- The cost of a manufactured computer component decreases over time even without major improvements in the basic implementation technology
  - **The learning curve**

- Increasing volumes affect cost in several ways.
  - They decrease the time needed to get down the learning curve
  - Volume decreases cost, since it increases purchasing and manufacturing efficiency
- The competition among the suppliers also lowers the cost

- Cost of an Integrated Circuit
- The manufacturing process dictates the cost of an IC

- Cost of Operation:
- The cost to operate the computers may become significant in addition to the cost of purchase
- E.g: Warehouse Scale Computers(WSC)
  - To lower operational costs in a WSC architects need to use energy efficiently

Dependability

- One difficult question is deciding when a system is operating properly
  – Service level agreements (SLAs) or
  – Service level objectives (SLOs)
- An SLA could be used to decide whether the system was up or down

- Systems alternate between two states of service with respect to an SLA:
  1. Service accomplishment, where the service is delivered as specified
  2. Service interruption, where the delivered service is different from the SLA
  - From state 1 to state 2 = **failure**
  - From state 2 to state 1 = **restoration**

- Quantifying these transitions leads to the two main measures of dependability:
- Module reliability and
- Module availability
- Its now possible to quantify these two values and hence answer the question whether the system is operating properly (Dependable)

Measuring, Reporting, and Summarizing Performance

- In comparing design alternatives, we often want to relate the performance of two different computers, say, X and Y.
- The phrase “X is faster than Y” is used here to mean that the response time is lower on X than on Y for the given task.
  - Response time: the time between the start and the completion of an event—also referred to as execution time

- "X is n times faster than Y"

$$n = {}^{execution\ time\ of\ Y}/_{execution\ time\ of\ X}$$

- Execution time is the reciprocal of performance

$$n = {}^{perfomance\ of\ X}/_{performance\ of\ Y}$$

- But what is time?
  - wall-clock time
  - response time/elapsed time
  - **CPU time**
- The response time seen by the user is the elapsed time of the program, not the CPU time.
  - Service time + Wait time + Transmission time

- CPU time = IC x CPI x Tc = IC / (IPC x F)
  - IC is the instruction count
  - CPI is the clock cycles per instruction and IPC = 1/CPI is the
  - Instruction count per clock cycle
  - Tc is the clock cycle time and F=1/Tc is the clock frequency

- Reducing the execution time of a program can be obtained by reducing either IC or CPI or Tc or both of them.
- Reducing CPI is increasing IPC and reducing Tc means increasing F.

- Computer users who routinely run the same programs would be the perfect candidates to evaluate a new computer.
- To evaluate a new system the users would simply compare the execution time of their workloads – Programs and OS

Benchamarking

- Collections of benchmark applications, called benchmark suites, are a popular measure of performance of processors with a variety of applications.

- Electronic Design News Embedded Microprocessor Benchmark Consortium (or EEMBC, pronounced “embassy”) benchmarks.

- SPEC (Standard Performance Evaluation Corporation)
  - Had its roots in efforts in the late 1980s to deliver better benchmarks for workstations
  - HAS several benchmark versions https://www.spec.org/

Quantitative Principles of Computer Design

- Take Advantage of Parallelism
  - One of the most important methods for improving performance
- Principle of Locality i.e locality of reference
  - Programs tend to reuse data and instructions they have used recently
  - It is the locality of reference that allows us to build memory hierarchies

Types of localities

- Temporal locality: refers to the fact that recently accessed items from memory are likely to be accessed again in the near future
- **loops** in a program are a good illustration for temporal locality;

- Spatial locality: items that are near to each other in memory tend to be referenced near one to another in time;
- Data structures and arrays are good illustrations for spatial locality.

- Focus on the Common Case
- In making a design trade-off, favor the frequent case over the infrequent case

- A common example is related to multiply/divide in a CPU: in most programs the multiplications (integer or float), by far exceed the number of divisions.
- It is therefore no wonder that many CPUs have hardware support for multiplication (at least for integers) while division is emulated in software.

- A common concern is, how to detect the common case and to compute the performance gain when this case is optimized.
  - Easy for systems for dedicated application difficulty for general purpose systems.

- A fundamental law, called Amdahl’s law, can be used to quantify this principle.

Qualitative Aspects of Design

Functional requirements	Required feature
1. **Application area**	
General purpose	Balanced performance
Scientific	Efficient floating point arithmetic
Commercial	Support for Cobol, data bases and transaction processing
Special purpose	High performance for specific tasks

Functional requirements	Required features
2. **Software compatibility**	
Object code	Frozen architecture; programs move easily from one machine to another without any investment.
High level language	Designer has maximum freedom; substantial effort in software (compilers) is needed

Operating system requirements	
Size of address space	Too low an address space may limit application
Memory management	Flat, paged, segmented etc.
Protection	Page protection v. segment Protection
Context switch	required to interrupt and restart Programs
Interrupts	Hardware support, software support

Standards	
Buses	VME, SCSI, IPI etc.
Floating point	IEEE 754, IBM, DEC
Operating system	UNIX, DOS, Windows NT, OS/ 2, proprietary
Networks	Ethernet, FDDI, etc
Programming languages	The choice will influence the instruction set

Summary