Computer System

Virendra Singh
Associate Professor
Computer Architecture and Dependable Systems Lab
Department of Electrical Engineering
Indian Institute of Technology Bombay
http://www.ee.iitb.ac.in/~viren/
E-mail: viren@ee.iitb.ac.in

Computer Organization & Architecture

Lecture 3 (18 March 2013)

CADSL

Performance Growth

Unmatched by any other industry !
[John Crawford, Intel]

- Doubling every 18 months (1982-1996): 800x
  - Cars travel at 44,000 mph and get 16,000 mpg
  - Air travel: LA to NY in 22 seconds (MACH 800)
  - Wheat yield: 80,000 bushels per acre

- Doubling every 24 months (1971-1996): 9,000x
  - Cars travel at 600,000 mph, get 150,000 mpg
  - Air travel: LA to NY in 2 seconds (MACH 9,000)
  - Wheat yield: 900,000 bushels per acre

Technology Push

- Technology advances at varying rates
  - E.g. DRAM capacity increases at 60%/year
  - But DRAM speed only improves 10%/year
  - Creates gap with processor frequency!
- Inflection points
- Current issues causing an "inflection point"
  - Power consumption
  - Reliability
  - Variability

Application Pull

- Corollary to Moore's Law:
  Cost halves every two years

  In a decade you can buy a computer for less than its sales tax today. –Jim Gray

- Computers cost-effective for
  – National security – weapons design
  – Enterprise computing – banking
  – Departmental computing – computer-aided design
  – Personal computer – spreadsheets, email, web
  – Pervasive computing – prescription drug labels

Application Pull

- What about the future?

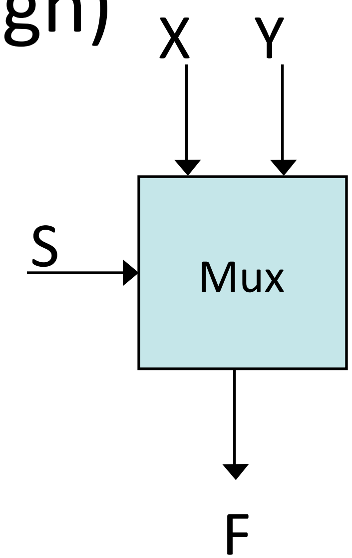
- Must dream up applications that are not cost-effective today
  - Virtual reality
  - Telepresence
  - Sensing, analyzing, actuating in real-world environments

Abstraction

- Difference between interface and implementation
  - Interface: WHAT something does
  - Implementation: HOW it does so

Abstraction, E.g.

- 2:1 Mux (Digital Design)
- Interface

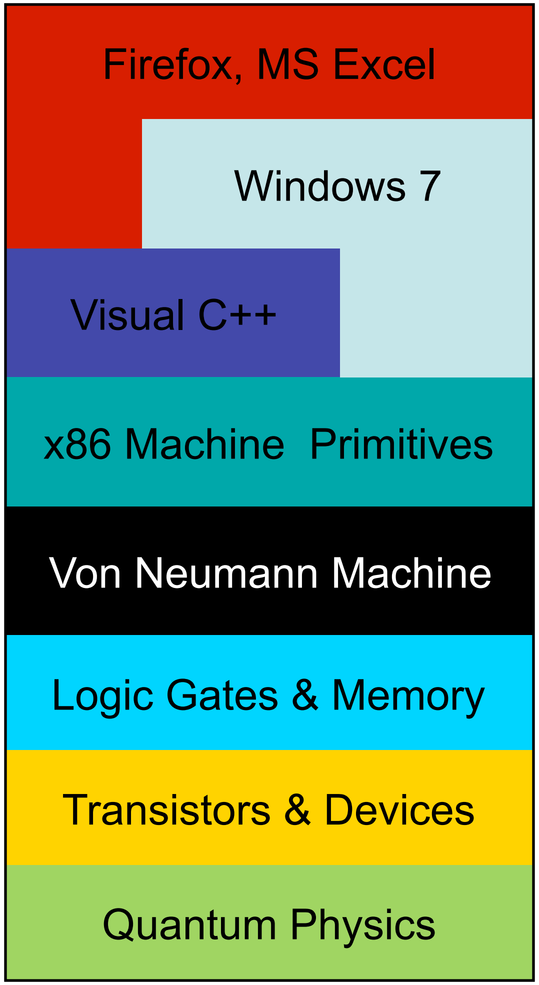


S	F
0	X
1	Y

- Implementations
  – Gates (fast or slow), pass transistors

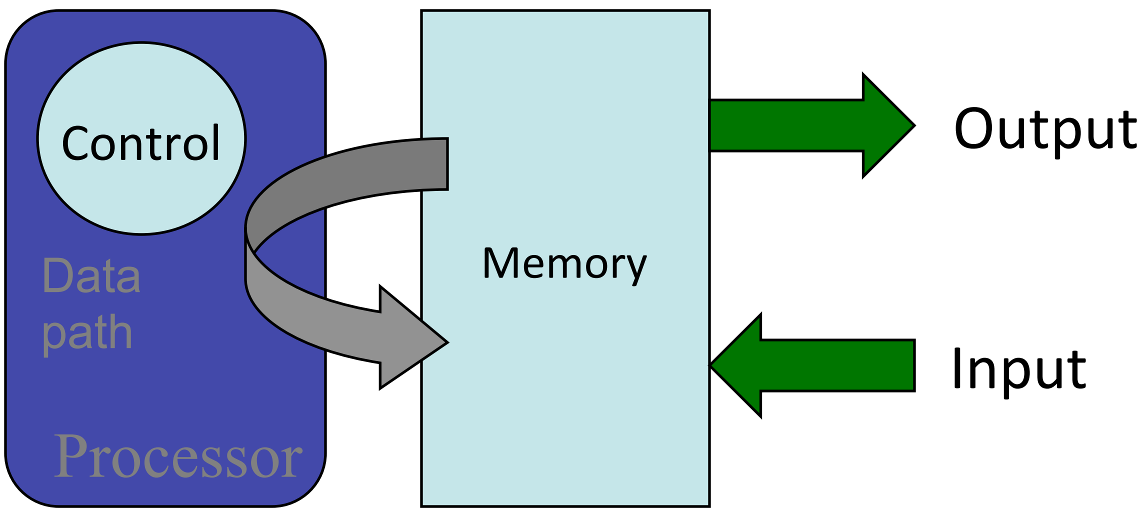
What's the Big Deal?

- Tower of abstraction
- Complex interfaces implemented by layers below
- Abstraction hides detail
- Hundreds of engineers build one product
- Complexity unmanageable otherwise



Basic Division of Hardware

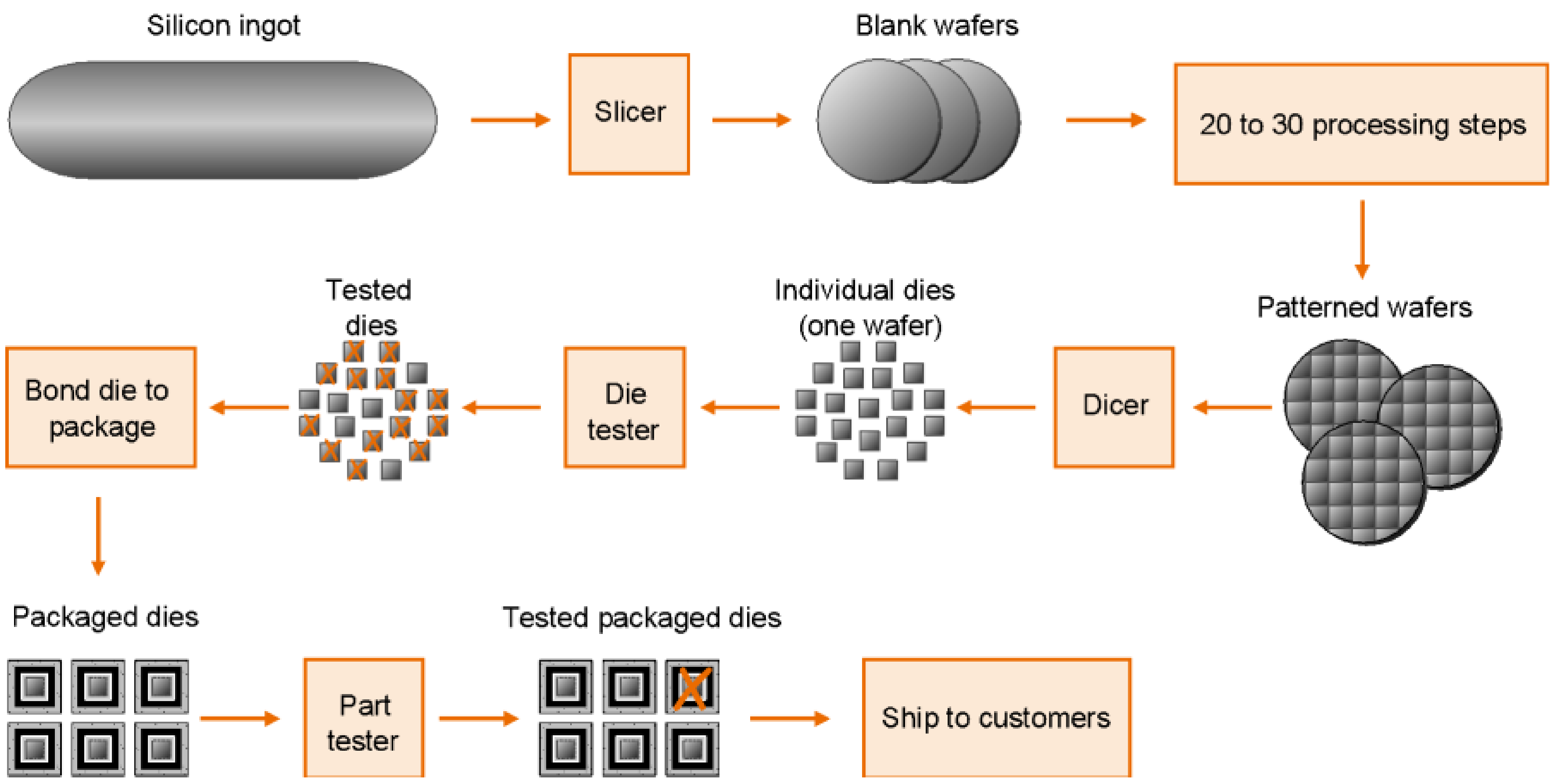
- In space (vs. time)



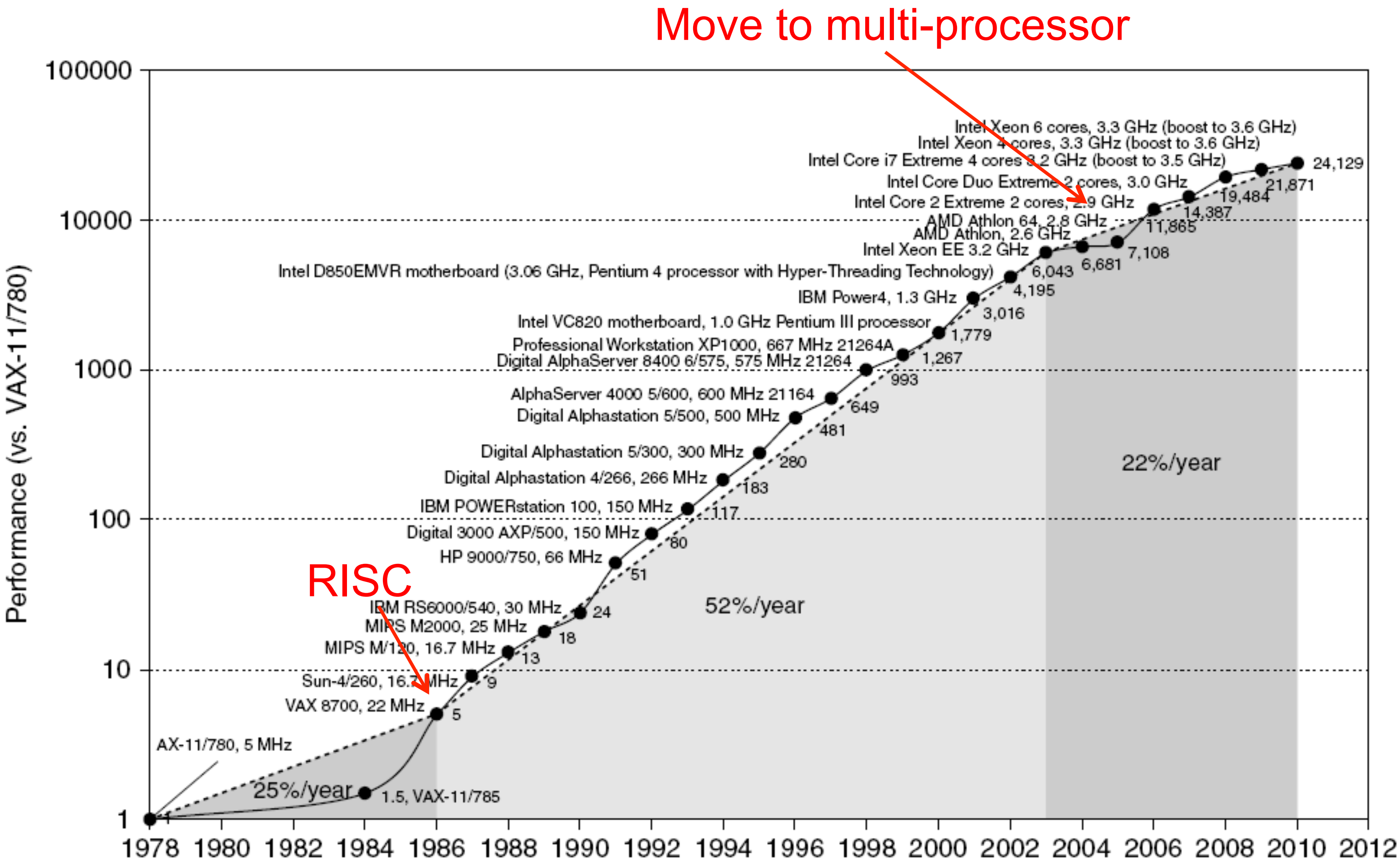
Building Computer Chips

- Complex multi-step process
  - ✓ Slice silicon ingots into wafers
  - ✓ Process wafers into patterned wafers
  - ✓ Dice patterned wafers into dies
  - ✓ Test dies, select good dies
  - ✓ Bond to package
  - ✓ Test parts
  - ✓ Ship to customers and make money

Building Computer Chips



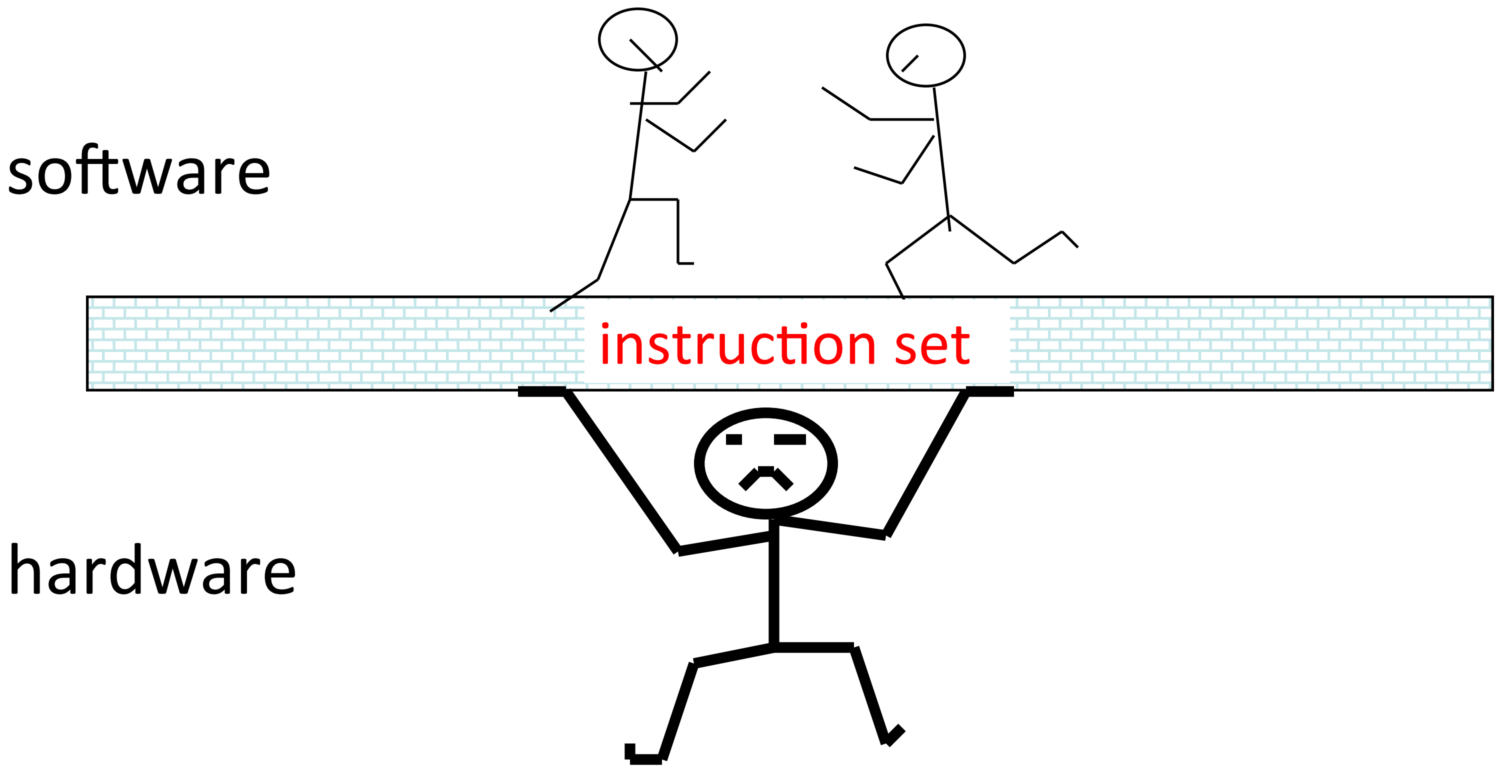
Single Processor Performance



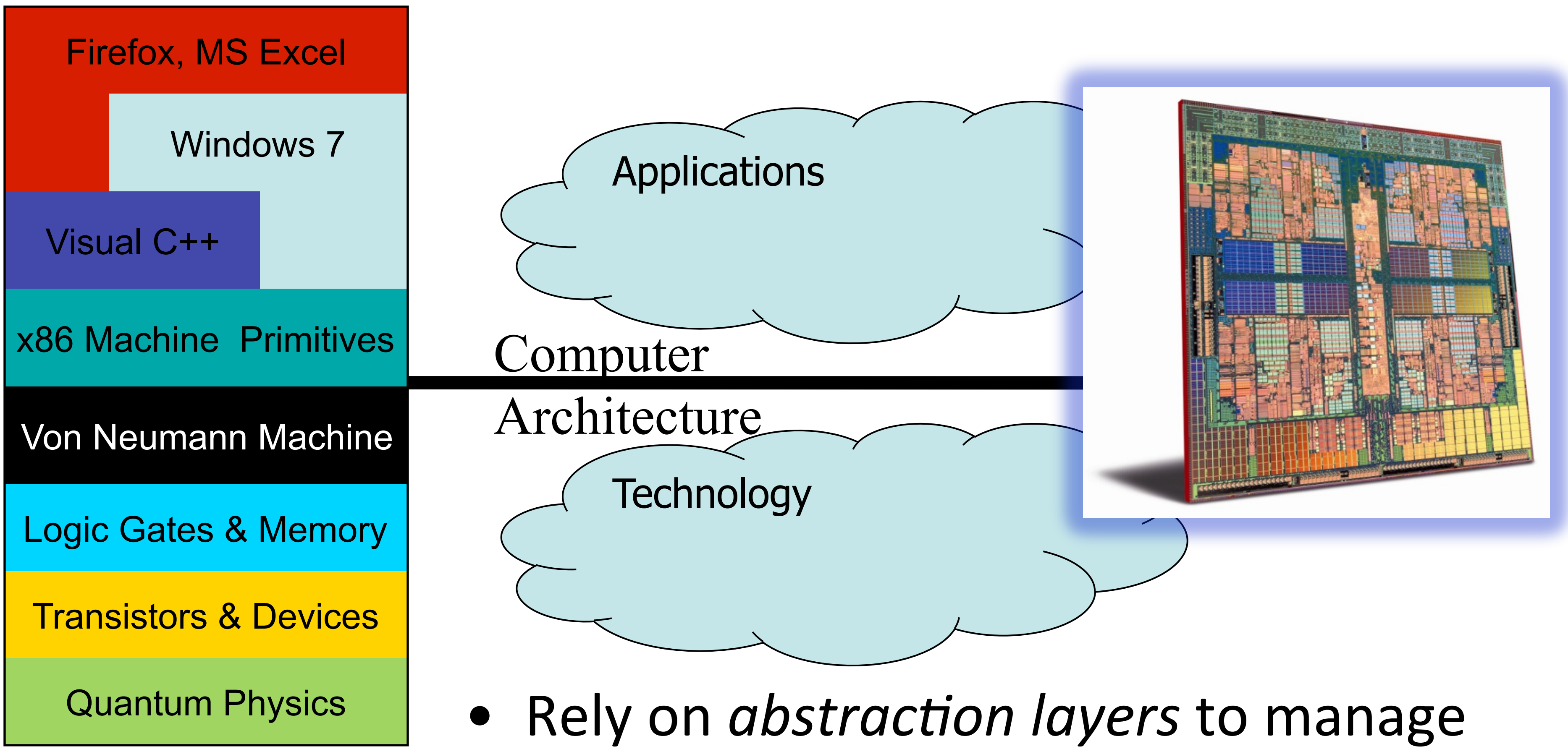
Computer Architecture's Changing Definition

- 1950s to 1960s:
  Computer Architecture Course = Computer Arithmetic
- 1970s to mid 1980s:
  Computer Architecture Course = Instruction Set Design, especially ISA appropriate for compilers
- 1990s onwards:
  Computer Architecture Course = Design of CPU (Processor Microarchitecture), memory system, I/O system, Multiprocessors

Instruction Set Architecture (ISA)



Computer Architecture



- Rely on *abstraction layers* to manage complexity

Bottom Line

- Designers must know BOTH software and hardware
- Both contribute to layers of abstraction
- IC costs and performance
- Compilers and Operating Systems

Performance and Cost

- Which of the following airplanes has the best performance?

Airplane	Passengers	Range (mi)	Speed (mph)
Boeing 737-100	101	630	598
Boeing 747	470	4150	610
BAC/Sud Concorde	132	4000	1350
Douglas DC-8-50	146	8720	544

- How much faster is the Concorde vs. the 747
- How much bigger is the 747 vs. DC-8?

Performance and Cost

- Which computer is fastest?
- Not so simple
  - Scientific simulation – FP performance
  - Program development – Integer performance
  - Database workload – Memory, I/O

Performance of Computers

- Want to buy the fastest computer for what you want to do?
  - Workload is all-important
  - Correct measurement and analysis
- Want to design the fastest computer for what the customer wants to pay?
  - Cost is an important criterion

Defining Performance

- What is important to whom?
- Computer system user
  - Minimize elapsed time for program = time_end – time_start
  - Called response time
- Computer center manager
  - Maximize completion rate = #jobs/second
  - Called throughput

Response Time vs. Throughput

- Is throughput = 1/av. response time?
  - Only if NO overlap
  - Otherwise, throughput > 1/av. response time
  - E.g. a lunch buffet – assume 5 entrees
  - Each person takes 2 minutes/entrée
  - Throughput is 1 person every 2 minutes
  - BUT time to fill up tray is 10 minutes
  - Why and what would the throughput be otherwise?
    - 5 people simultaneously filling tray (overlap)
    - Without overlap, throughput = 1/10

What is Performance for us?

- For computer architects
  - CPU time = time spent running a program
- Intuitively, bigger should be faster, so:
  - Performance = 1/X time, where X is response, CPU execution, etc.
- Elapsed time = CPU time + I/O wait
- We will concentrate on CPU time

Improve Performance

- Improve (a) response time or (b) throughput?
  - Faster CPU
    - Helps both (a) and (b)
  - Add more CPUs
    - Helps (b) and perhaps (a) due to less queueing

Performance Comparison

- Machine A is n times faster than machine B iff perf(A)/perf(B) = time(B)/time(A) = n
- Machine A is x% faster than machine B iff
  - perf(A)/perf(B) = time(B)/time(A) = 1 + x/100
- E.g. time(A) = 10s, time(B) = 15s
  - 15/10 = 1.5 => A is 1.5 times faster than B
  - 15/10 = 1.5 => A is 50% faster than B

Breaking Down Performance

- A program is broken into instructions
  - H/W is aware of instructions, not programs
- At lower level, H/W breaks instructions into cycles
  - Lower level state machines change state every cycle
- For example:
  - 1GHz Snapdragon runs 1000M cycles/sec, 1 cycle = 1ns
  - 2.5GHz Core i7 runs 2.5G cycles/sec, 1 cycle = 0.25ns

Iron Law

$$\text{Processor Performance} = \frac{\text{Time}}{\text{Program}}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Time}}{\text{Cycle}}$$

(code size) (CPI) (cycle time)

Architecture --> Implementation --> Realization

Compiler Designer **Processor Designer** **Chip Designer**

Iron Law

- Instructions/Program
  - Instructions executed, not static code size
  - Determined by algorithm, compiler, ISA
- Cycles/Instruction
  - Determined by ISA and CPU organization
  - Overlap among instructions reduces this term
- Time/cycle
  - Determined by technology, organization, clever circuit design

Our Goal

- Minimize time which is the product, NOT isolated terms
- Common error to miss terms while devising optimizations
  - e.g. ISA change to decrease instruction count
  - BUT leads to CPU organization which makes clock slower
- Bottom line: terms are inter-related

Thank You