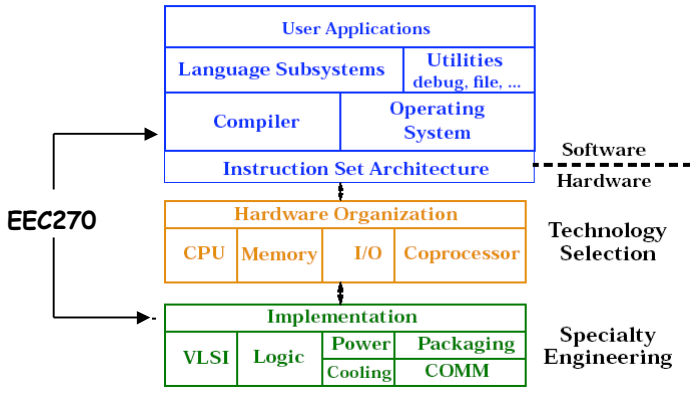
# Computer Architecture

Venkatesh Akella
EEC 270

Winter 2005



# What is it?



## Computer Architecture

- Basically a story of unprecedented improvement
- $1K buys you a machine that was 1-5 million dollars a couple of decades ago
- Why?
- Improvements in technology (process scaling)

Around 35%/year

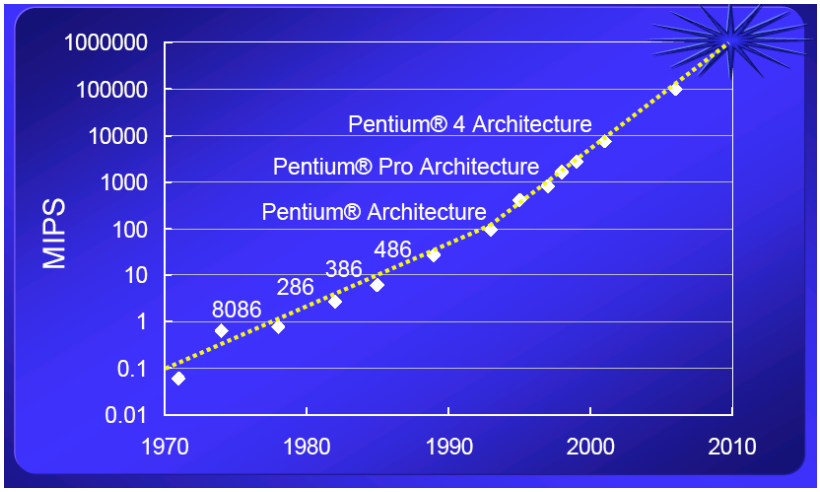
- Improvement in architecture

Around 23%

Result = Moore's law – 58% per year

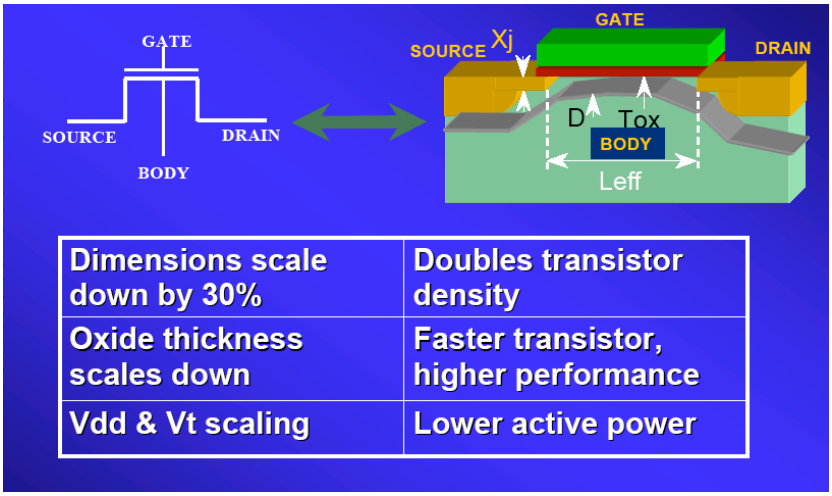


## 1 Trillions Ops/sec by 2010




S. Borkar MICRO 37, December 2004

## Technology Scaling -




| | |
|---|---|
| Dimensions scale down by 30% | Doubles transistor density |
| Oxide thickness scales down | Faster transistor, higher performance |
| Vdd & Vt scaling | Lower active power |



## Computer usage has changed

- 1950-1960 – Big Mainframes: time-share
- 1970 – Minicomputers, time share but perhaps more locally owned machines
- 1980 – microprocessors are born, personal computing has becoming a reality
- 1990 – Network is born – servers, computer farms, PDAs, cellphones, embedded computing and DSP are re-born
- Now – ubiquitous computing, where the computer is does not matter, integration of communication, computation and entertainment

## Categories of Computers

- Three Cateogries have emerged
- DESKTOP – optimized for price/performance
- SERVER – optimized for availability, scalability, and throughput
- EMBEDDED – everything around us has a computer whether it is a washing machine or the router card or the cell phone

Key issues – real time, cost, application-specific performance as opposed to general purpose performance, resource constraints like memory/ input/output etc.

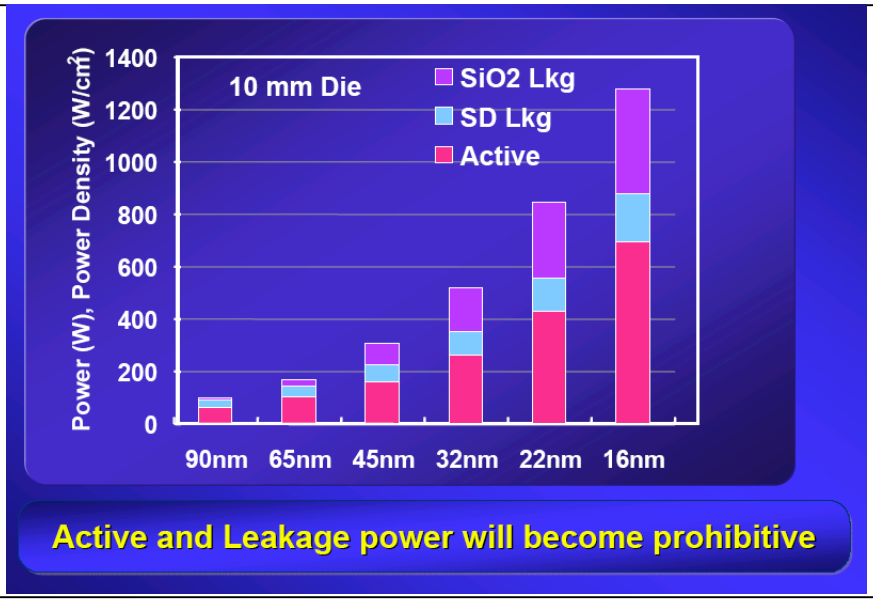


## NEW CONSTRAINTS

- Power. Power. Power
- Dynamic and now LEAKAGE power
- Clock frequencies are in the GHz – distribution of clock, skew
- Wire delay >>> Gate delay
- Real time performance – DSP, media processors

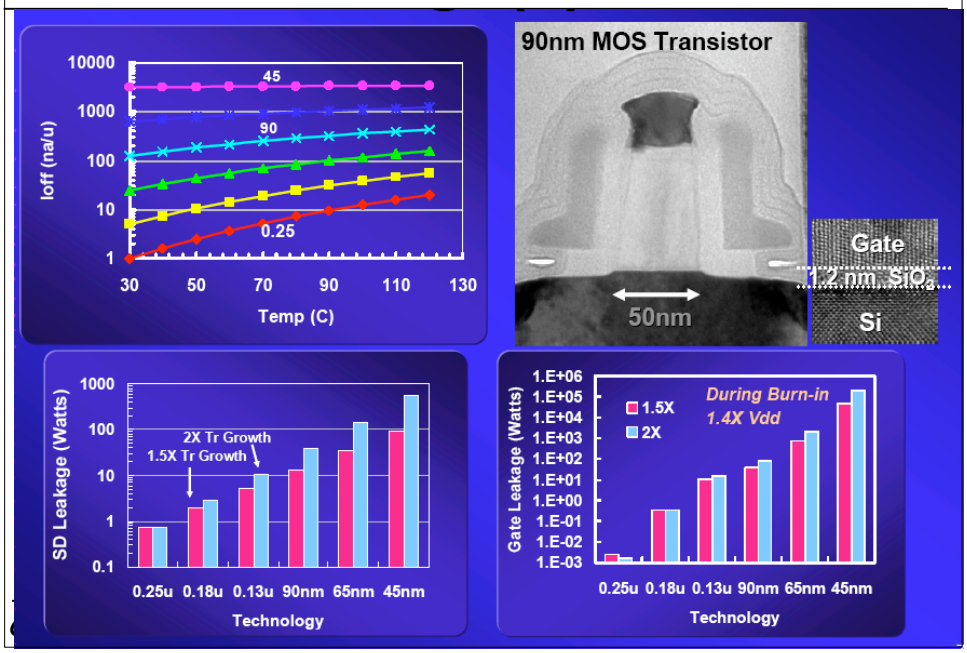
Computer Architecture has to change to handle these new constraints.

## Power is Enemy #1




Chapter 1

## Leakage Power - New Monster to Grapple with



## Server Availability Costs

| Application | down $/hr | 1%/yr (87.6 hours) | .05%/yr (43.8 hours) | 0.1%/yr (8.8 hours) |
|---|---|---|---|---|
| Stock Broker | $6450 | $565M | $283M | $56.5M |
| Credit Card Authorization | $2600 | $228M | $114M | $22.8M |
| Package Shipping Centers | $150 | $13M | $6.6M | $1.3M |
| Home Shopping Channel | $113 | $9.9M | $4.9M | $1M |
| Catalog Sales Center | $90 | $7.9M | $3.9M | $800K |
| Airline Reservation Center | $89 | $7.9M | $3.9M | $800K |
| Cellular Service Activation | $41 | $3.6M | $1.8M | $400K |
| ISP's | $25 | $2.2M | $1.1M | $200K |
| ATM's | $14 | $1.2M | $600K | $100K |

FY 2000 Data from Kembel



## Sector Demographics

| Feature | Desktop | Server | Embedded |
|---|---|---|---|
| System Price | $1K - $10K | $10K - $10M | $10 - $100K |
| Price of uP module | $100 - $1K | $200 - $2K | $0.2 - $200 |
| uP's sold per year | 150M | 4M | 300M (32 & 64-bit only) |
| Critical System Issues | price-performance & graphics performance | throughput, RAS, scal-ability | Price, power consump-tion, application specific performance |

## ASPECTS OF COMPUTER DESIGN

- Complicated game with many constraints, many objectives
- Determine the important attributes (market segment, applications)

THEN MAXIMIZE performance

WHILE staying within the COST & POWER BUDGET



## Technology Trends

- Integrated CIRCUITS
  - Density Increases at 35% per year
  - Die Size increases 10 – 20% per year
  - Combination is a chip complexity growth rate of 55% per year
  - Transistor speed increase is similar but wire delay does not track this curve, so clock rates do not go up as fast
- DRAM
  - Density Quadruples every 3-4 years (40 – 60% per year)
  - Cycle Time decreases slowly – 33% in 10 year
  - Interface changes have improved bandwidth however

What does this mean?

Product Cycle – 2 to 4 years and Market requirements something new is needed 6-12 Months

- Pipelined design efforts using multiple design teams
- Have to design for a complexity target that does not exist yet
- Infrastructure and NRE Costs – 200 million to 500 million dollars

## IC Scaling – How does it affect us?

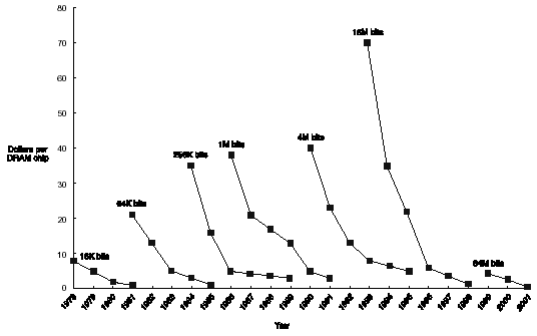
- **Wire delay is proportional to R * C**
  - As wires get smaller their cross section decreases, so R increases
  - C does not reduce linearly (cross talk, coupling)
  - So, wire delay does not track improvements in gate delay
  - %age of cycle time taken by wire delays becomes non-negligible
  - Pentium 4 uses 2 pipe stages just for signal propagation
- **Power**
  - P_TOTAL = P_ACTIVE + P_LEAKAGE
  - P_ACTIVE = $\frac{1}{2}\ \alpha CV^2f$
  - P_LEAKAGE = Tunneling + subthreshold leakage
    = As Vt is lowered and feature size shrinks increases
    = Function of number of transistors on chip, so becomes significant as number of transistors increases.



## COST OF AN IC

$$\text{IC-cost} = \frac{\text{Die-cost} + \text{Die-test-cost} + \text{Die-package-cost}}{\text{Final-Test-Yield}}$$

### DRAM Costs



## COST OF A DIE

- Compute
  - # dies / wafer
  - % yield

$$\text{Cost-of-die} = \frac{\text{Cost-of-wafer}}{\text{Dies-per-wafer} \times \text{Die-yield}}$$

$$\text{Dies-per-wafer} = \frac{\pi \times (\text{Wafer-diameter}/2)^2}{\text{Die-area}} - \frac{\pi \times \text{Wafer-diameter}}{\sqrt{2 \times \text{Die-area}}} - \text{Test-dies-per-wafer}$$

$$\text{Die-yield} = \text{Wafer-yield} \times \left(1 + \left(\frac{\text{Defects-per-unit-area} \times \text{Die-area}}{\alpha}\right)\right)^{-\alpha}$$

Where alpha depends on the process - the more complex the higher the alpha value - for today's multilevel metal CMOS $\alpha$ ~= 4 and defects per unit area are typically between .4 and .8 per $cm^2$



## Die Testing & Packages

- Testing is a significant portion of chip cost
  - Varies from 10% to over 50% for Military specification parts
- In 1990 tester costs were about $150/hr
- In 1993 tester costs are $500 per hours, runs 10 times faster but chips are 100 times more complex
- Now test costs about $10,000 per hour – you see the trend
- Packaging material depends on use and power
- Configuration determined by cavity and pins
  - » 200 pin plastic quad flat pack = $3
  - » 400 pin ceramic PGA = $50
- Testing and Packaging are significant contributors to cost especially as complexity increases

## Cost Breakdown of a 1000 dollar PC

- CABINET = 6%
- I/O Devices = 37% (includes keyboard, monitor, 20 GB hard drive, DVD drive ..)
- PROCESSOR BOARD = 37%
  - » CPU = 22%
  - » DRAM = 5% (huge difference from 33% in 1995)
  - » VIDEO CARD = 5%
  - » Motherboard and networking = 5%
- SOFTWARE = 20% (a.k.a Microsoft Tax)



## PERFORMANCE

- 2 key aspects
  - execution time
  - throughput
  - making 1 faster may slow the other
- Comparing performance
  - Performance = 1/execution time
  - if X is n times faster than Y:

$$\frac{\text{Execution Time}_{Y}}{\text{Execution Time}_{X}} = N$$

  - Similar for throughput comparisons
  - Improved performance ==> decreasing XEQ time
    - or increasing throughput

## But what time?

- Should not consider time spent waiting for I/O delays, because someone else is using the same resources as in a multitasking/timeshared system
- User CPU Time – time spent to execute the program in question
- System CPU Time – the amount of time the OS spends on behalf of your program
- Unix Time Command
- 27.2 u 11.1s 56.6 68%



## Which Programs to choose?

- Real Programs
  - Clearly the right choice but porting them maybe a problem
  - Burden on the user. Need to know exactly what your workload is
- Kernels
  - Computational intensive pieces of real programs
  - Livermore loops and Linpack are examples
  - Not Real programs – so might be misleading
- Toy Benchmarks – Not a good idea
- Synthetic Benchmarks
  - Has some merit especially during early design stage
  - Since they are not real, they do not actually represent anything that a user maybe interested in

## BENCHMARKS

- Dhrystone – tells you how well integer code works
- Loops/LINPACK – floating point, matrix algebra
- PC SPECIFIC
  - Business Winstone – office apps and browser
  - CC Winstone – content creation – Photoshop, audio/video editing
  - WinBench = collection that targets CPU, disk, video
- SPEC2000
  - 4th generation, to test CPU performance
  - CINT2000 – 11 integer benchmarks
  - CFP2000 – 14 floating point benchmarks
  - SPECWeb – web server tests
- TPC = Transaction processing council
  - TPCA – simple bank teller transaction
  - TPCC – complex database query, TPC-H – decision support
- EEMBC
  - 35 kernels in 5 classes – automotive, consumer, networking, office automation and telecommunication



## Other Problems

| | Machine A | Machine B | Machine C |
|---|---|---|---|
| Program 1 (secs) | 1 | 10 | 20 |
| Program 2 (secs) | 1000 | 100 | 20 |
| Total Time (secs) | 1001 | 110 | 40 |

Which is better?

By how much?

Are the programs equally important?

## Aggregating and Reporting Performance

- Arithmetic Mean - provides a simple average

$$\frac{1}{n}\sum_{i=1}^{n} \text{Time}i$$

  - doesn't account for weight - all programs treated equal
- Or if rate (as opposed to time) is given - use the Harmonic Mean

$$\frac{n}{\sum_{i=1}^{n} \frac{1}{\text{Rate}i}}$$

  - still independent of weight



## Weighted Aggregates

***Weight is the frequency % of use***

- Weighted arithmetic mean

$$\sum_{i=1}^{n} \text{Weight}i \times \text{Time}i$$

  - better but beware the dominant program time
- Weighted harmonic mean

$$\frac{n}{\sum_{i=1}^{n} \frac{\text{Weight}i}{\text{Rate}i}}$$

  - same problem - no surprise

## Normalized Execution Times

- Normalize with respect to a reference machine such as SPARC-10
- We get a set of ratios – r1, r2, … rN
- How do we represent aggregate performance?

1. Arithmetic mean of ratios

   - Problem – depends on the reference machine

   - Depends on running time of a specific program, so results can be manipulated

2. Geometric Mean of Ratios

   - Consistent Results independent of the choice of reference machine



## Normalized Aggregates

- Geometric Mean

$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution Time Ratio}_i}$$

- Has the nice property that:
  - ratio of the means = Mean of the ratios
  - independent of running times of individual programs
- Better than arithmetic means but
  - still do not form accurate prediction models
- Still have to remain cautious

**SPEC USES Geometric Mean**

reference machine you choose

## What's wrong with GM?

- Does not predict the execution time, which violates the fundamental basis of performance analysis
- You could still manipulate the numbers by focusing the optimizations on programs that are easy (small) than the ones that are the slowest
- Why?

If Program 1 is improved from 2 sec to 1 sec

Program 2 is improved from 10000 sec to 5000sec

The improvement in the spec numbers is still the same though it maybe much easier to optimize program 1 by just increasing the cache size or block size in cache.



## Amadahl's Law

***quantification of the diminishing return principle***

- defines speedup gained from a particular feature

$$\text{Speedup} = \frac{\text{Execution time without using the enhancement}}{\text{Execution time using the enhancement}}$$

- note XEQ-time = 1/Performance so another variant is possible
- depends on 2 factors
  - fraction of original computation time that can take advantage of the enhancement - e.g. the *commonality* of the feature
  - level of improvement gained by the feature
- Amdahl's law

$$\text{Speedup}_{\text{Overall}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

## A Simple Example of Amadahl's Law

Given Instruction Mix : FP = 50%, FPSQRT = 20% and other 30%

Designers say 40x improvement in FPSQRT, 2X improvement FP or 8X improvement of other ops for the same cost (say time frame)

What would you choose?

FPSQRT

$$\frac{1}{(1-\text{Fraction}_{\text{enhanced}})+\frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}} = \text{Speedup}_{\text{FPSQRT}} = \frac{1}{(1-0.2)+\frac{0.2}{40}} = 1.242$$

FP

$$\text{Speedup}_{\text{FP}} = \frac{1}{(1-0.5)+\frac{0.5}{2}} = 1.333$$

Other

$$\text{Speedup}_{\text{Other}} = \frac{1}{(1-0.3)+\frac{0.3}{8}} = 1.356$$

Other Wins !!! However, remember Amadahl's law does not take the Cost of implementation into account. Here we assumed everything was same



## The Performance Equation

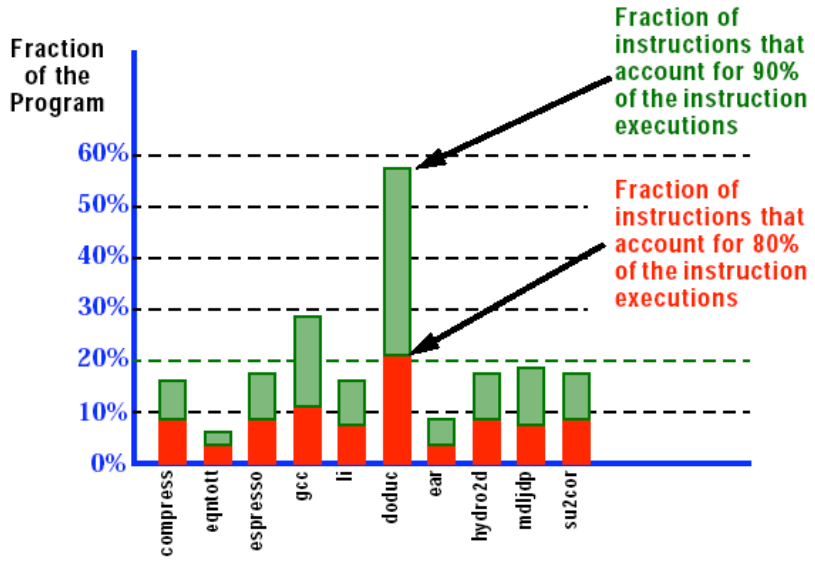
$$\text{CPU time} = \text{IC} \times \text{CPI} \times \text{Cycle Time} = \frac{\text{IC} \times \text{CPI}}{\text{Clock Rate}}$$

$$\text{CPU time} = \left( \sum_{i=1}^{n} \text{CPI}_i \times \text{IC}_i \right) \times \text{Cycle Time}$$

- IC = dynamic instruction count, depends on compiler and the instruction set of a machine
- CPI = depends on organization and ISA
- cycle time depends on HW technology, logic design, algorithms used to implement different hardware structures

SO, they are inter-related. Optimizing one without considering the impact of the optimization on the other parameters is a common pitfall.

## 90-10 Observations on SPEC92 Programs




Chapter 1