Introduction, Focus, Overview

Outline

- ***A science of tradeoffs***
- ***The transformation hierarchy***
- ***The algorithm, the compiler, the microarchitecture***
- ***The microarchitecture view***
- ***The physical view***
- ***Speculation***
- ***Intro to Nonsense: Is hardware parallel or sequential***
- ***Design points***
- ***Design Principles***
- ***Role of the Architect***
- ***Numbers***
- ***Thinking outside the box***

Trade-offs, the overriding consideration:

What is the cost?
What is the benefit?

- ***Global view***
  - ***Global vs. Local transformations***
- ***Microarchitecture view***
  - ***The three ingredients to performance***
- ***Physical view***
  - ***Wire delay (recently relevant) – Why? (frequency)***
  - ***Bandwidth (recently relevant) – Why? (multiple cores)***
  - ***Power, energy (recently relevant) – Why? (cores, freq)***
  - ***Soft errors (recently relevant) – Why? (freq)***
  - ***Partitioning (since the beginning of time)***

Trade-offs, the overriding consideration:

What is the cost?
What is the benefit?

- ***Global view***
  - ***Global vs. Local transformations***
- ***Microarchitecture view***
  - ***The three ingredients to performance***
- ***Physical view***
  - ***Wire delay (recently relevant) – Why? (frequency)***
  - ***Bandwidth (recently relevant) – Why? (multiple cores)***
  - ***Power, energy (recently relevant) – Why? (cores, freq)***
  - ***Soft errors (recently relevant) – Why? (freq)***
  - ***Partitioning (since the beginning of time)***

Problem

Algorithm

Program

ISA (Instruction Set Arch)

Microarchitecture

Circuits

Electrons

The Triangle (originally from George Michael)

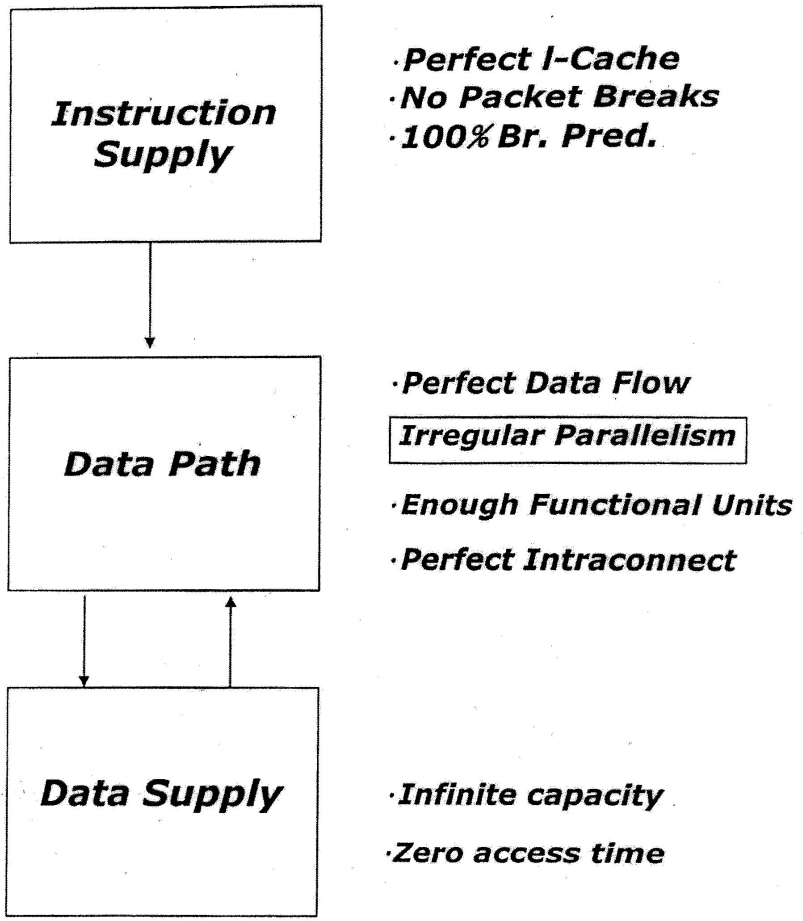
- ***Only the programmer knows the ALGORITHM***
  - ***Pragmas***
  - ***Pointer chasing***
  - ***Partition code, data***
- ***Only the COMPILER knows the future (sort of ??)***
  - ***Predication***
  - ***Prefetch/Poststore***
  - ***Block-structured ISA***
- ***Only the HARDWARE knows the past***
  - ***Branch directions***
  - ***Cache misses***
  - ***Functional unit latency***

Trade-offs, the overriding consideration:

What is the cost?
What is the benefit?

- ***Global view***
  - ***Global vs. Local transformations***
- ***Microarchitecture view***
  - ***The three ingredients to performance***
- ***Physical view***
  - ***Wire delay (recently relevant) – Why? (frequency)***
  - ***Bandwidth (recently relevant) – Why? (multiple cores)***
  - ***Power, energy (recently relevant) – Why? (cores, freq)***
  - ***Soft errors (recently relevant) – Why? (freq)***
  - ***Partitioning (since the beginning of time)***

Microarchitecture (The Requirement)



A few more words on Data Supply

- ***Memory is particularly troubling***
  - ***Off-chip latency (hundreds of cycles, and getting worse)***
  - ***What can we do about it?***
    - ***Larger caches***
    - ***Better replacement policies***

- ***Is MLP (Memory level parallelism) the answer***
  - ***Wait for two accesses at the same time***
  - ***Do parallel useful work while waiting (Runahead)***

Trade-offs, the overriding consideration:

What is the cost?
What is the benefit?

- ***Global view***
  - ***Global vs. Local transformations***
- ***Microarchitecture view***
  - ***The three ingredients to performance***
- ***Physical view (more important in the multicore era)***
  - ***Wire delay (recently relevant) – Why? (frequency)***
  - ***Bandwidth (recently relevant) – Why? (multiple cores)***
  - ***Power, energy (recently relevant) – Why? (cores, freq)***
  - ***Soft errors (recently relevant) – Why? (freq)***
  - ***Partitioning (since the beginning of time)***

Speculation

- ***Why good? – improves performance***
- ***How? – we guess***
  - ***Starting with the design of ALUs, many years ago!***
  - ***Branch prediction – enables parallelism***
  - ***Way prediction***
  - ***Data prefetching – enables parallelism***
  - ***Value prediction – enables parallelism***
  - ***Address prediction – enables parallelism***
  - ***Memory disambiguation – enables parallelism***
- ***Why bad? – consumes energyl***

Hardware – Sequential or Parallel?

- ***Hardware is inherently parallel***
  - ***It has been since time began***
  - ***Then why the sudden interest***
- ***Useful if we pay attention to it (e.g., factorial)***
- ***The key idea is Synchronization***
  - ***It can be explicit***
  - ***It can be implicit***
- ***Pipelining***
  - ***Parallelism at its most basic level***
  - ***Everyone in the world understands that (e.g., factories)***
- ***Speculation***
- ***Single thread vs. multiple threads***
- ***Single core vs. multiple cores***

Design Points

- *-- Performance*
- *-- Reliability*
- *-- Availability*
- *-- Cost*
- *-- Power*
- *-- Time to Market*

Design Principles

- ***Critical path design***
- ***Bread and Butter design***
- ***Balanced design***

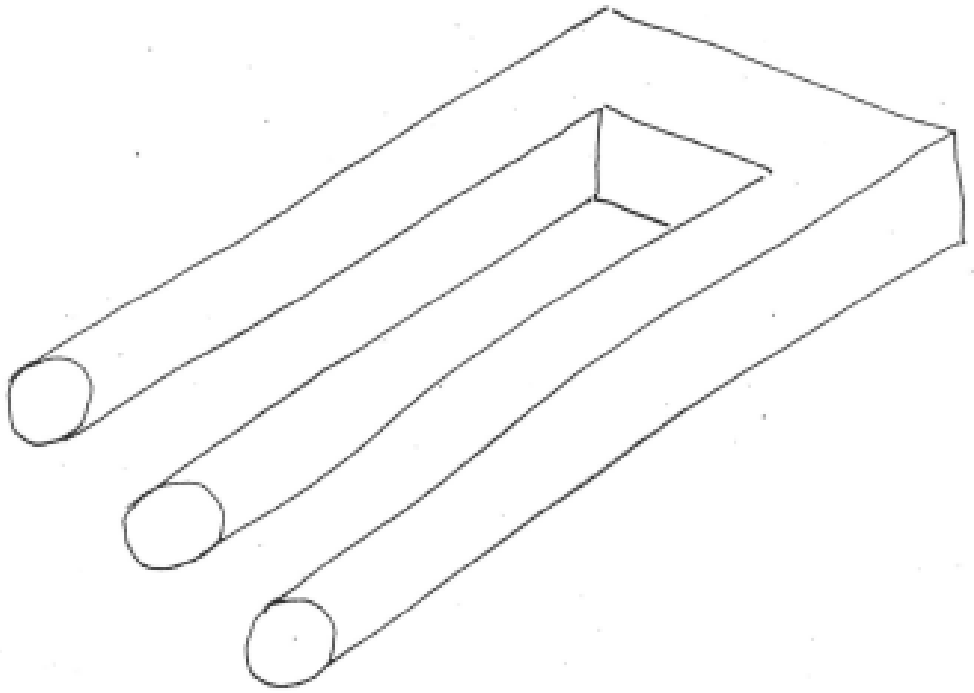
Role of the Architect

- *-- Look Backward (Examine old code)*
- *-- Look forward (Listen to the dreamers)*
- *-- Look Up (Nature of the problems)*
- *-- Look Down (Predict the future of technology)*

Numbers
(because comparch is obsessed with numbers)

- ***The Baseline – Make sure it is the best***
  - ***Superlinear speedup***
  - ***Recent example, one core vs. 4 cores with ability to fork***
- ***The Simulator you use – Is it bug-free?***
- ***Understanding vs "See, it works!"***
  - ***16/64***
- ***You get to choose your experiments***
  - ***SMT, throughput: run the idle process***
  - ***Combining cores: what should each core look like***
- ***You get to choose the data you report***
  - ***Wrong path detection: WHEN was the wrong path detected***
- ***Never gloss over anomalous data***

***Finally*, people are always telling you:**
Think outside the box



I prefer: Expand the box

A Few Specifics

- ***HPS – expanded on Tomasulo***
- ***SMT – expanded on Burton***
- ***Perceptron predictor – expanded on Widrow/Rosenblatt/etc.***

Something you are all familiar with: Look-ahead Carry Generators

- ***They speed up ADDITION***
- ***But why do they work?***

Addition