Obtaining High Performance in the Multi-core Era: What is the role of the Transformation Hierarchy

Yale Patt

The University of Texas at Austin

Barcelona Multicore Workshop 2011

November 2,3, 2011

Outline

- ***What is the transformation hierarchy?***
- ***How do we deal with it?***
  - ***First the nonsense***
  - ***Second, what we need to do differently?***
  - ***What can we expect if we do?***
- ***How do we make that happen?***

Outline

- ***What is the transformation hierarchy?***
- ***How do we deal with it?***
  - ***First the nonsense***
  - ***Second, what we need to do differently?***
  - ***What can we expect if we do?***
- ***How do we make that happen?***

Problem

Algorithm

Program

ISA (Instruction Set Arch)

Microarchitecture

Circuits

Electrons

Outline

- ***What is the transformation hierarchy?***
- ***How do we deal with it?***
  - ***First the nonsense***
  - ***Second, what we need to do differently?***
  - ***What can we expect if we do?***
- ***How do we make that happen?***

Outline

- ***What is the transformation hierarchy?***
- ***How do we deal with it?***
  - ***First the Mega-nonsense***
  - ***Second, what we need to do differently?***
  - ***What can we expect if we do?***
- ***How do we make that happen?***

Important to disabuse ourselves of a lot of Mega-nonsense

- ***Multi-core was an architectural solution***
- ***Hardware works sequentially***
- ***Make the hardware simple – thousands of cores***
- ***ILP is dead***
- ***Abstraction is a pure good***
- ***All programmers need to be protected***
- ***Thinking in parallel is hard***

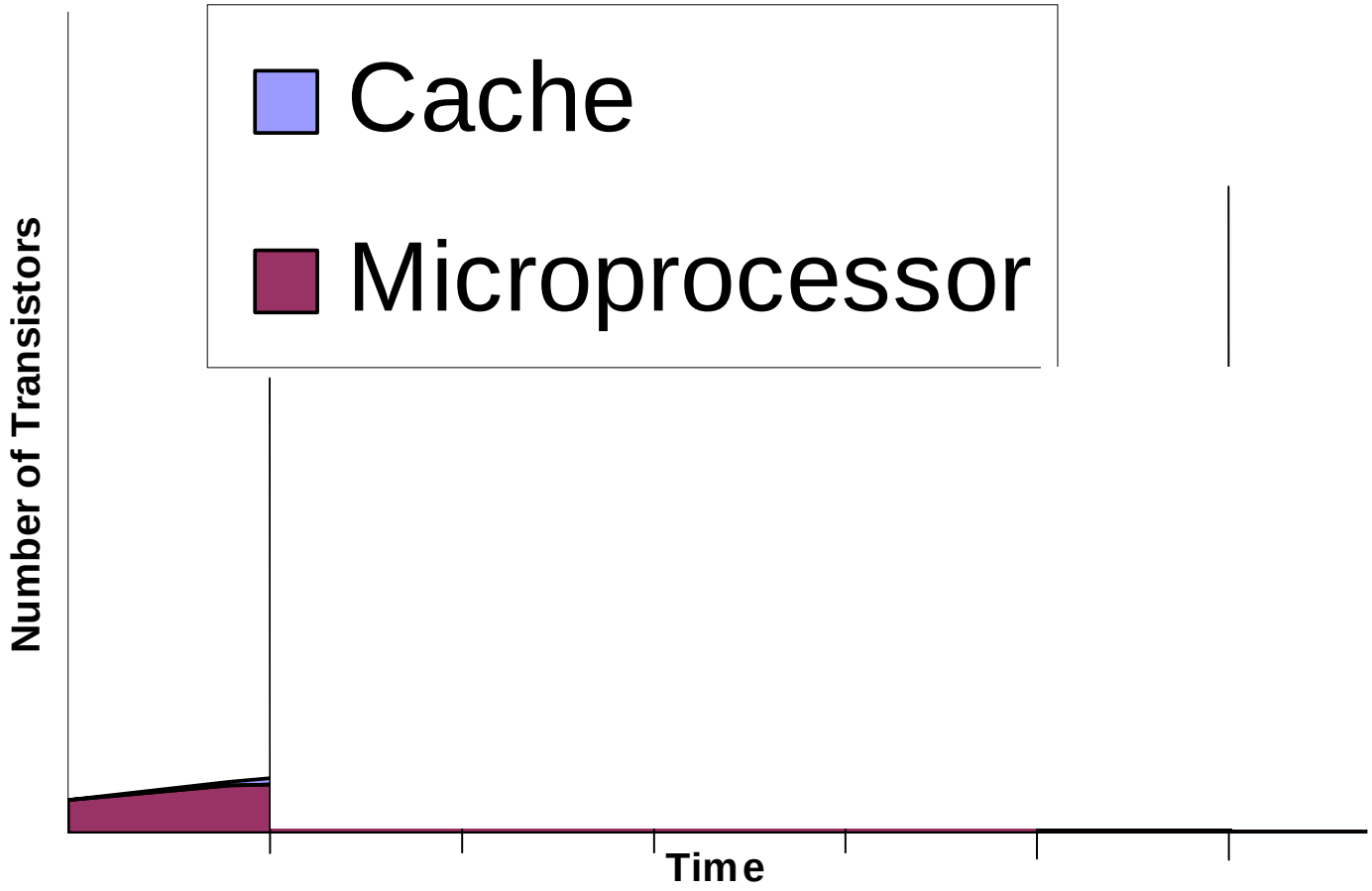
Some of the nonsense

- ***Multi-core was an architectural solution***
- ***Hardware works sequentially***
- ***Make the hardware simple – thousands of cores***
- ***ILP is dead***
- ***Abstraction is a pure good***
- ***All programmers need to be protected***
- ***Thinking in parallel is hard***

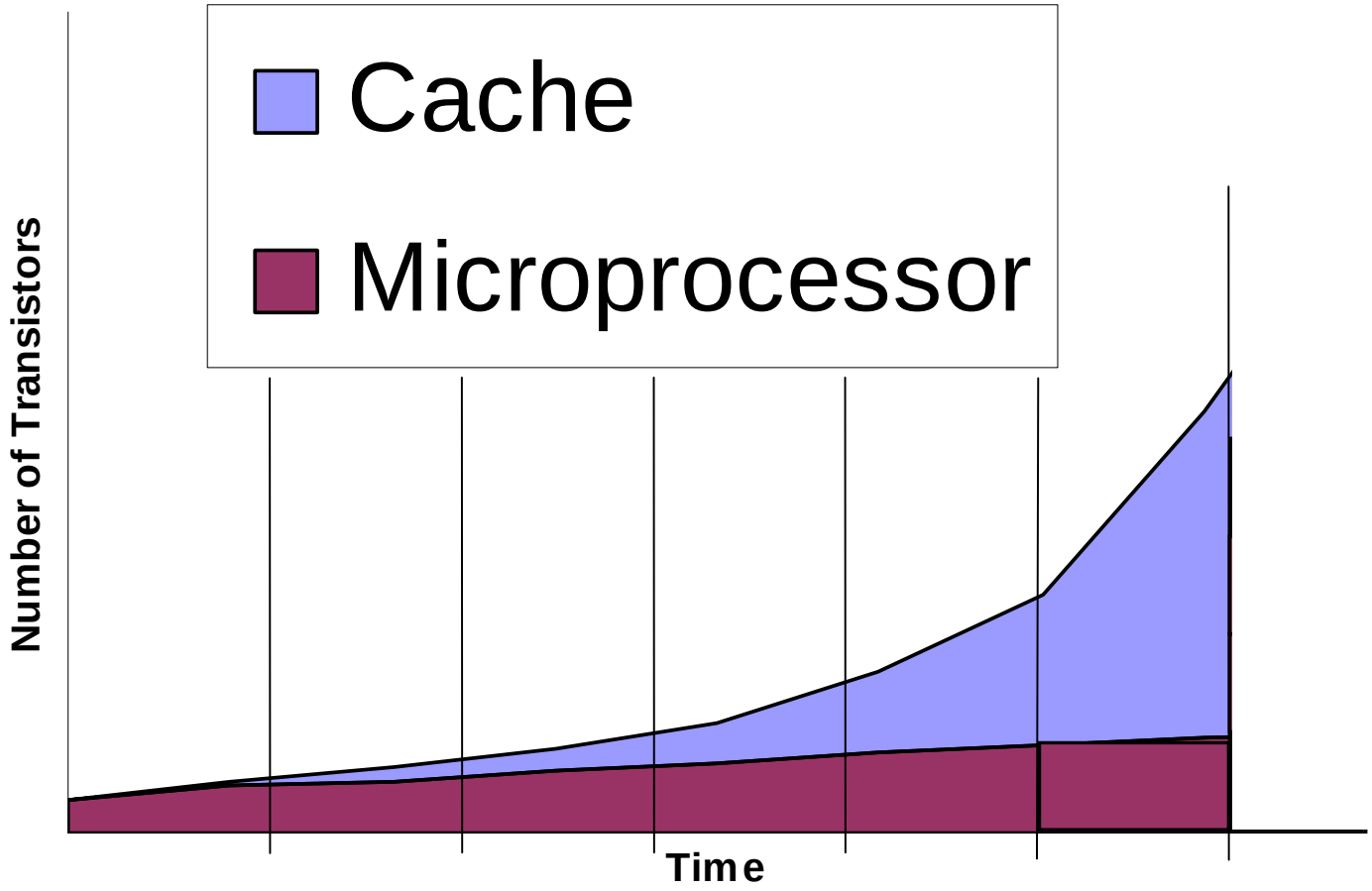
Multi-core: How we got to where we are (i.e., Moore's Law)

- ***The first microprocessor (Intel 4004), 1971***
  - ***2300 transistors***
  - ***106 KHz***
- ***The Pentium chip, 1992***
  - ***3.1 million transistors***
  - ***66 MHz***
- ***Today***
  - ***more than one billion transistors***
  - ***Frequencies in excess of 5 GHz***
- ***Tomorrow ?***

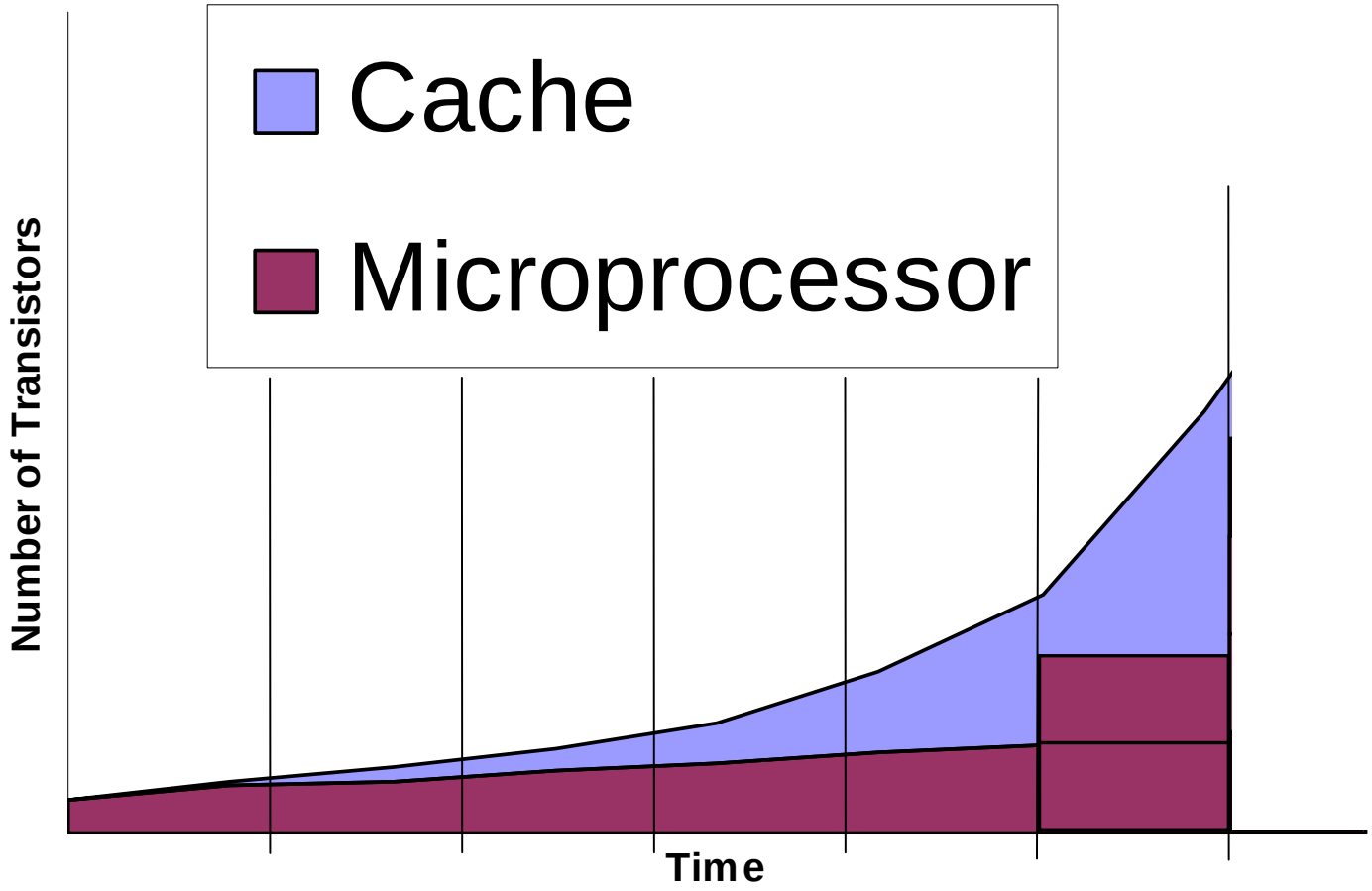
What have we done with these transistors?



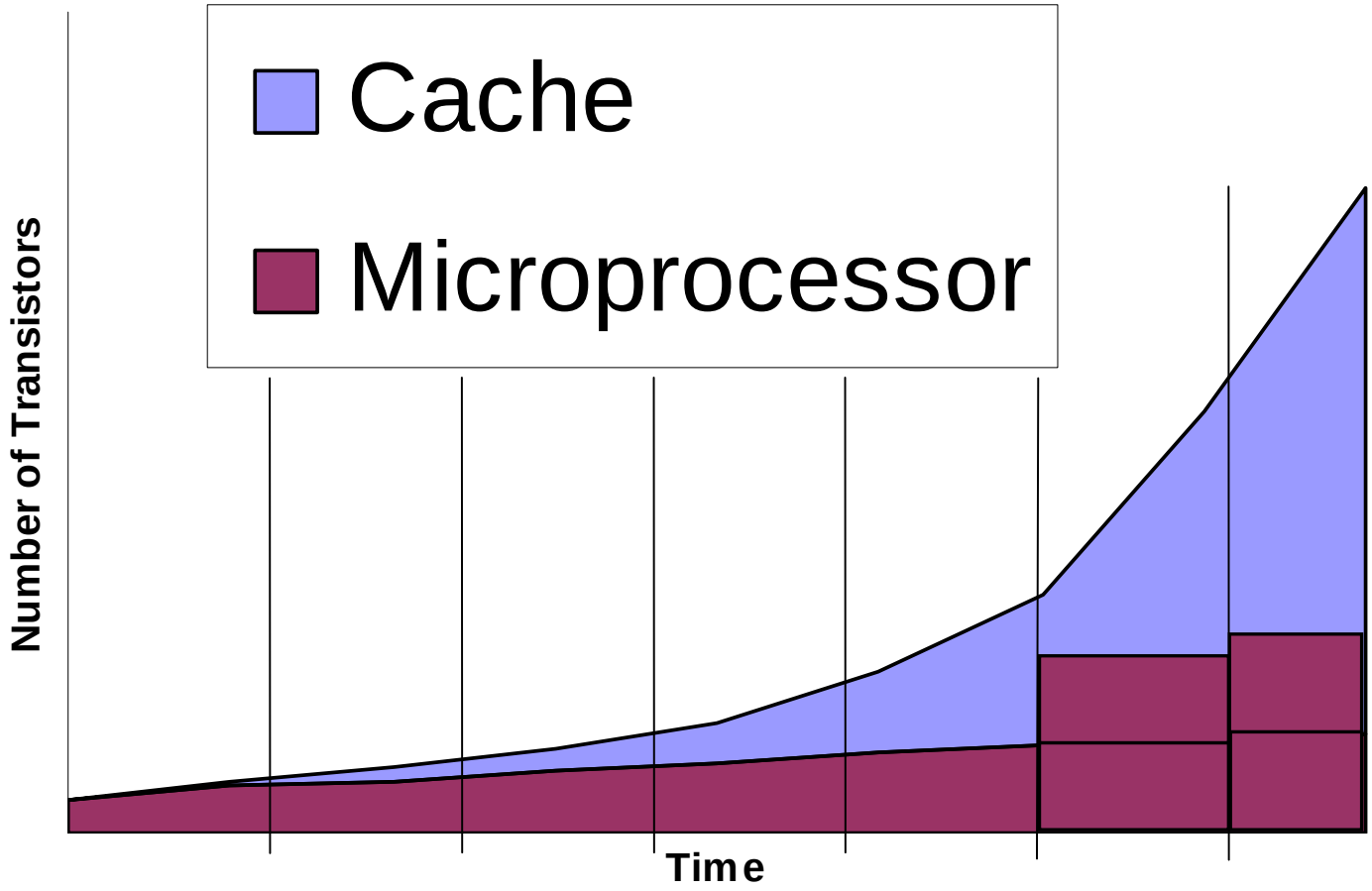
What have we done with these transistors?



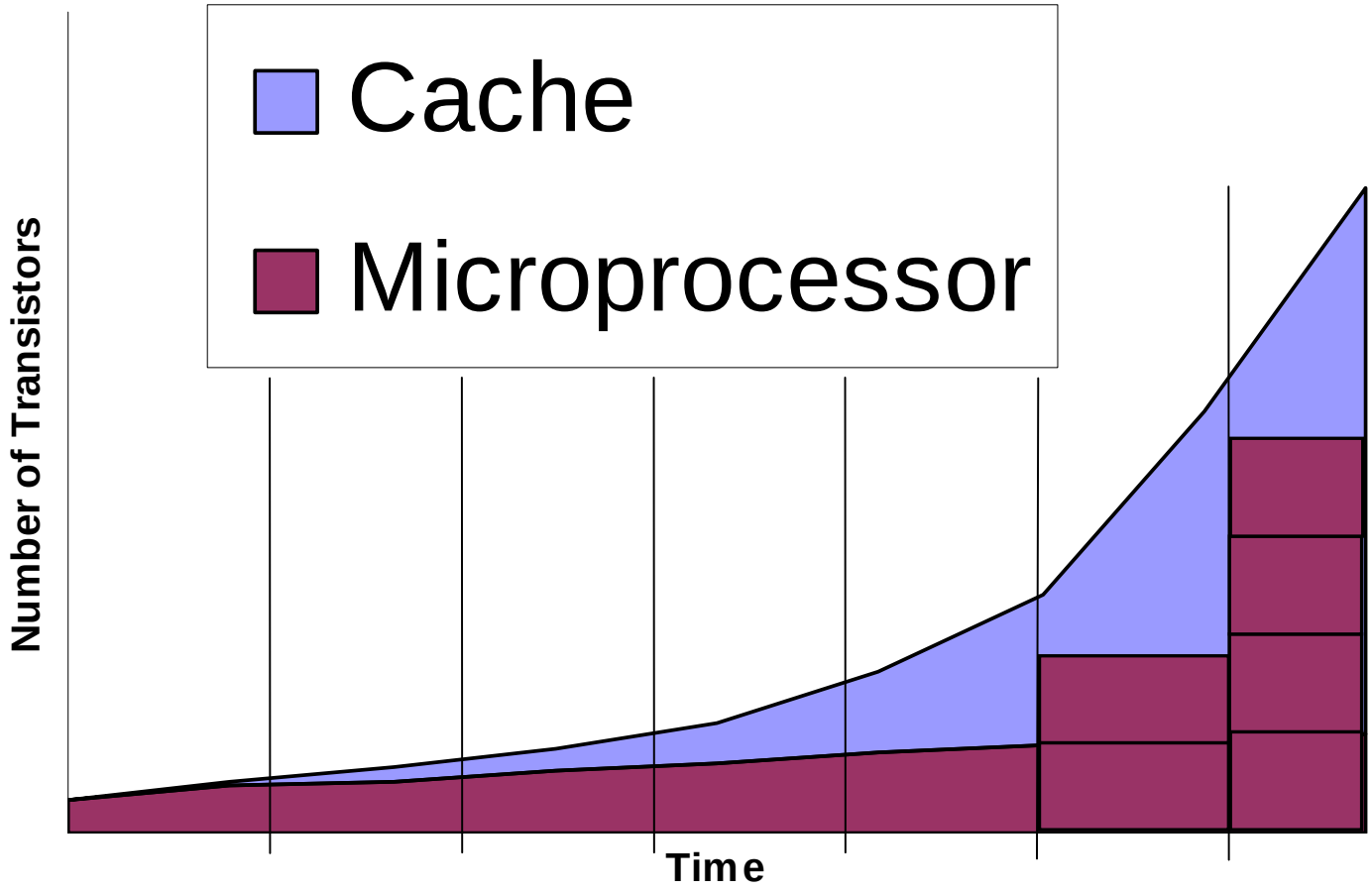
What have we done with these transistors?



What have we done with these transistors?



What have we done with these transistors?



Multi-core: How we got to where we are

- ***In the beginning: a better and better uniprocessor***
  - ***improving performance on the hard problems***
  - ***…until it just got too hard***
- ***Followed by: a uniprocessor with a bigger L2 cache***
  - ***forsaking further improvement on the "hard" problems***
  - ***utilizing chip area sub-optimally***
- ***Today: adding more and more cores***
  - ***dual core, quad core, octo core, 32 cores and counting…***
  - ***Why: It was the obvious most cost-effective next step***
- ***Tomorrow: ???***

So, What's the Point

- ***Multi-core exists; we will have 1000 core chips (…if we are not careful!)***
- ***But it wasn't because of computer architects***
- ***Ergo, we do not have to accept multi-core as it is***
- ***i.e., we can get it right in the future, and that means:***

  What goes on the chip

  What are the interfaces

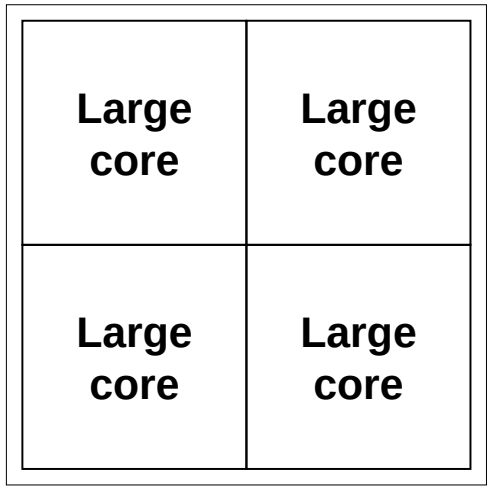
Some of the nonsense

- ***Multi-core was an architectural solution***
- ***Hardware works sequentially***
- ***Make the hardware simple – thousands of cores***
- ***ILP is dead***
- ***Abstraction is a pure good***
- ***All programmers need to be protected***
- ***Thinking in parallel is hard***

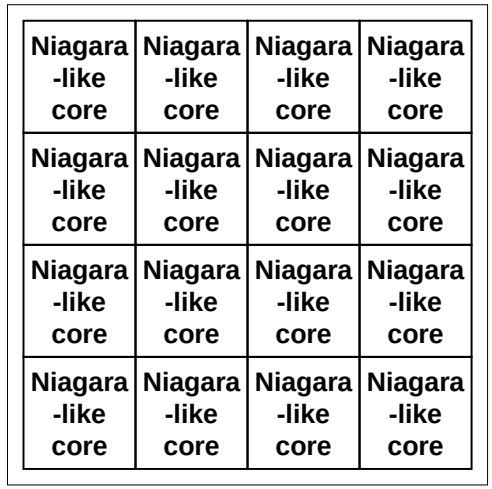
Some of the nonsense

- ***Multi-core was an architectural solution***
- ***Hardware works sequentially***
- ***Make the hardware simple – thousands of cores***
- ***ILP is dead***
- ***Abstraction is a pure good***
- ***All programmers need to be protected***
- ***Thinking in parallel is hard***

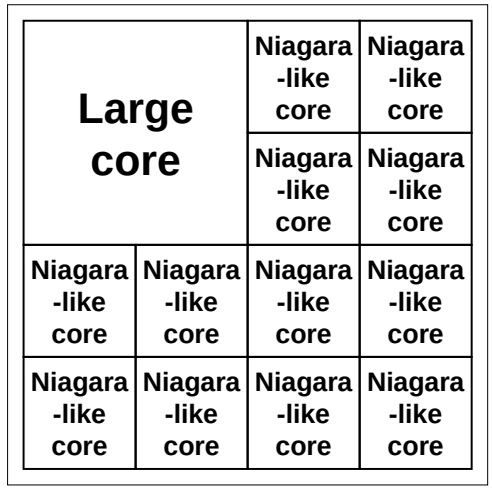
The Asymmetric Chip Multiprocessor (ACMP)



"Tile-Large" Approach



"Niagara" Approach

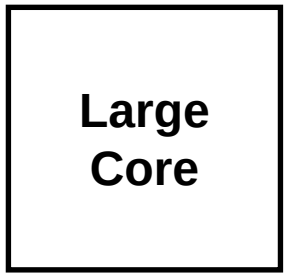


ACMP Approach

Heterogeneous, not Homogeneous

- ***My mantra since 2002***
  - ***(The Robert Chien lecture at Illinois)***
  - ***Pentium X / Niagara Y***
- ***More recently, ACMP***
  - ***First published as a UT Tech Report in February, 2007***
  - ***Later: PhD dissertation (2010) of M. Aater Suleman***
  - ***In between:***
    - ***ISCA 2010 (Data Marshalling)***
    - ***ASPLOS (ACS – Handling critical sections)***
- ***Today, we are working on MorphCore***

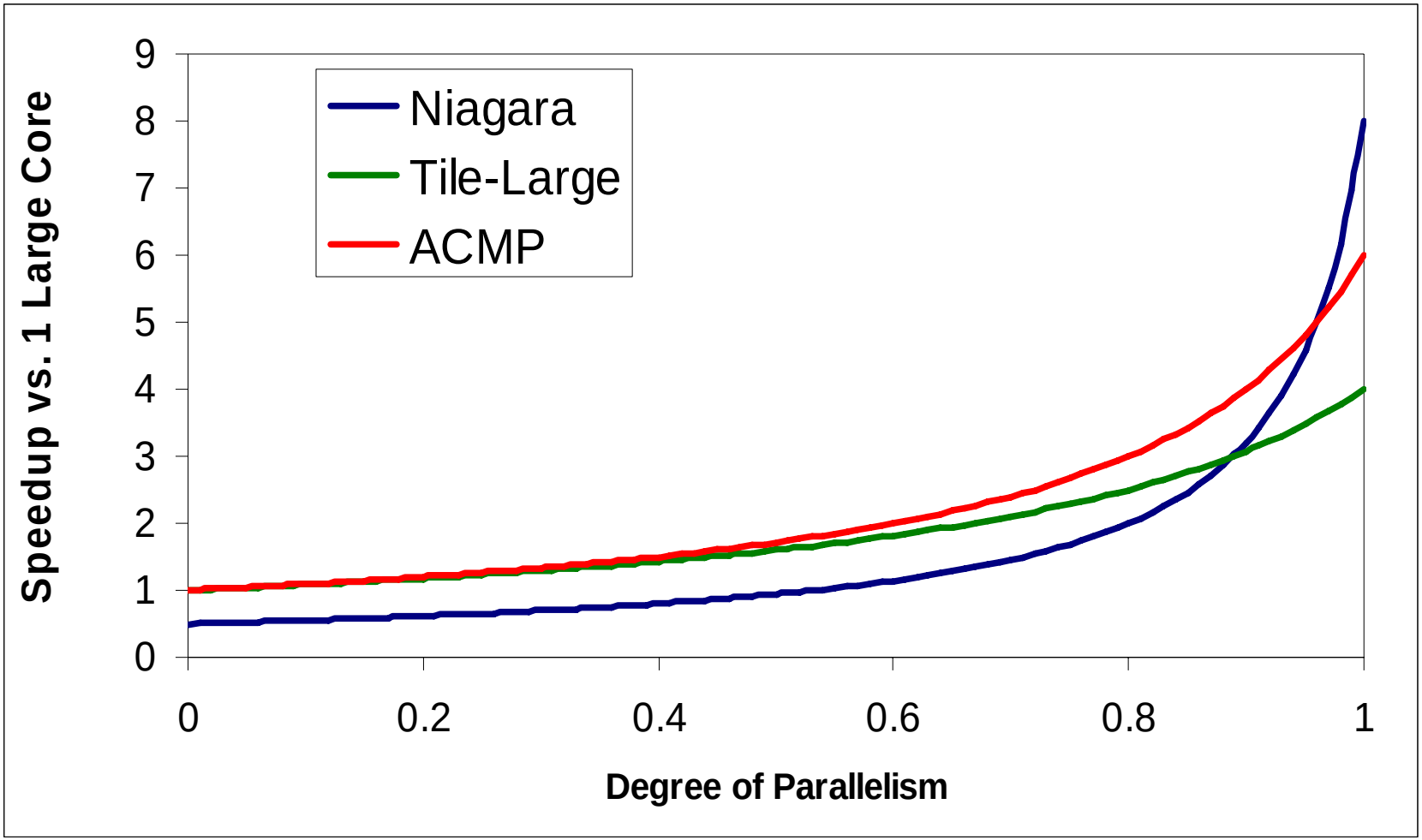
Large core vs. Small Core



- ***Out-of-order***
- ***Wide fetch e.g. 4-wide***
- ***Deeper pipeline***
- ***Aggressive branch predictor (e.g. hybrid)***
- ***Many functional units***
- ***Trace cache***
- ***Memory dependence speculation***

- ***In-order***
- ***Narrow Fetch e.g. 2-wide***
- ***Shallow pipeline***
- ***Simple branch predictor (e.g. Gshare)***
- ***Few functional units***

Throughput vs. Serial Performance



Some of the nonsense

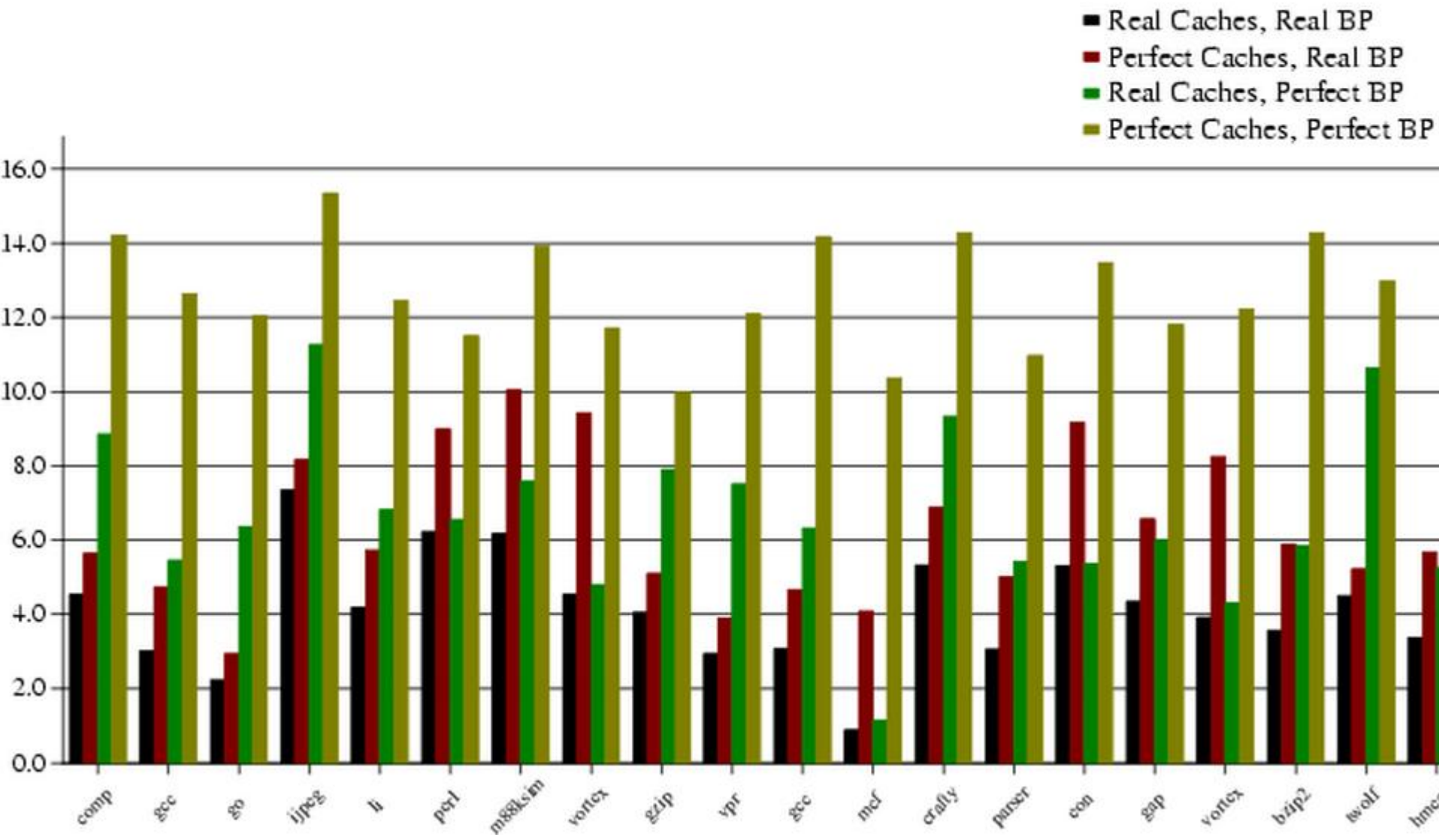
- ***Multi-core was an architectural solution***
- ***Hardware works sequentially***
- ***Make the hardware simple – thousands of cores***
- ***ILP is dead***
- ***Abstraction is a pure good***
- ***All programmers need to be protected***
- ***Thinking in parallel is hard***

ILP is dead

- ***We double the number of transistors on the chip***
  - ***Pentium M: 77 Million transistors (50M for the L2 cache)***
  - ***2nd Generation: 140 Million (110M for the L2 cache)***
- ***We see 5% improvement in IPC***
- ***Ergo: ILP is dead!***
- ***Perhaps we have blamed the wrong culprit.***

- ***The EV4,5,6,7,8 data: from EV4 to EV8:***
  - ***Performance improvement: 55X***
  - ***Performance from frequency: 7X***
  - ***Ergo: 55/7 > 7 -- more than half due to microarchitecture***

Potential of Improving Caches and BP



Moore's Law

- ***A law of physics***
- ***A law of process technology***
- ***A law of microarchitecture***
- ***A law of psychology***

Some of the nonsense

- ***Multi-core was an architectural solution***
- ***Hardware works sequentially***
- ***Make the hardware simple – thousands of cores***
- ***ILP is dead***
- ***Abstraction is a pure good***
- ***All programmers need to be protected***
- ***Thinking in parallel is hard***

Can Abstractions lead to trouble?

- ***Taxi to the airport***
- ***The Scheme chip***
- ***Sorting***
- ***Microsoft developers (Deeper understanding)***

Some of the nonsense

- ***Multi-core was an architectural solution***
- ***Hardware works sequentially***
- ***Make the hardware simple – thousands of cores***
- ***ILP is dead***
- ***Abstraction is a pure good***
- ***All programmers need to be protected***
- ***Thinking in parallel is hard***

Two kinds of people with problems to solve

- ***Those who need the utmost in performance***
  - ***and are willing to trade ease of use for extra performance***
  - ***For them, we need to provide a more powerful engine (Heavyweight cores for ILP, Accelerators for specific tasks)***
- ***Those who just need to get their job done***
  - ***and are content with trading performance for ease of use (They really do not want to know how computers work)***
  - ***For them, we need a software layer to bridge the gap***

Some of the nonsense

- ***Multi-core was an architectural solution***
- ***Hardware works sequentially***
- ***Make the hardware simple – thousands of cores***
- ***ILP is dead***
- ***Abstraction is a pure good***
- ***All programmers need to be protected***
- ***Thinking in parallel is hard***

Thinking in Parallel is Hard?

Thinking in Parallel is Hard?

- ***Perhaps: Thinking is Hard***

Thinking in Parallel is Hard?

- ***Perhaps: Thinking is Hard***

- ***How do we get people to believe:***
  Thinking in parallel is natural

Outline

- ***What is the transformation hierarchy?***
- ***How do we deal with it?***
  - ***First the nonsense***
  - ***Second, what we need to do differently?***
  - ***What can we expect if we do?***
- ***How do we make that happen?***

How do we harness tens of billions of transistors on each chip?

- ***In the old days, just add "stuff"***
  - ***Everything was transparent to the software***
  - ***Performance came because the engine was more powerful***
    - ***Branch Prediction, out-of-order Execution***
    - ***Larger caches***
  - ***But lately, the "stuff" has been more cores, which is no longer transparent to the software***
- ***Today, Bandwidth and Energy can kill you***
- ***Tomorrow, both will be worse…***
- ***Unless we find a better paradigm***

In my view, that means:

exploiting the transformation hierarchy

I propose four steps to help make that happen

Problem

Algorithm

Program

ISA (Instruction Set Arch)

Microarchitecture

Circuits

Electrons

Step 1: We Must Break the Layers

- ***(We already have in limited cases)***
- ***Pragmas in the Language***
- ***The Refrigerator***
- ***X + Superscalar***
- ***The algorithm, the language, the compiler, & the microarchitecture all working together***

IF we break the layers:

- ***Compiler, Microarchitecture***
  - ***Multiple levels of cache***
  - ***Block-structured ISA***
  - ***Part by compiler, part by uarch***
  - ***Fast track, slow track***

- ***Algorithm, Compiler, Microarchitecture***
  - ***X + superscalar – the Refrigerator***
  - ***Niagara X / Pentium Y***

- ***Microarchitecture, Circuits***
  - ***Verification Hooks***
  - ***Internal fault tolerance***

Step 2: We must recognize we need ILP cores

- ***Probably MorphCores***
- ***High performance out-of-order for serial code***
- ***SMT in-order when throughput dominates***

Step 3: We need more than one interface

- ***Those who need the utmost in performance***
  - ***and are willing to trade ease of use for extra performance***
  - ***For them, we need to provide a more powerful engine (Heavyweight cores for ILP, Accelerators for specific tasks)***
- ***Those who just need to get their job done***
  - ***and are content with trading performance for ease of use***
  - ***For them, we need a software layer to bridge the multi-core***
- ***…which means:***
  - ***Chip designers understanding what algorithms need***
  - ***Algorithms folks understanding what hardware provides***
  - ***Knowledgeable Software folks bridging the two interfaces***

Step 4: The future Run-time System

- ***Much closer to the microarchitecture***
- ***Procedures called by compiled code***
- ***Inputs from on-chip monitoring***
- ***Decides how to utilize on-chip resources***

Outline

- ***What is the transformation hierarchy?***
- ***How do we deal with it?***
  - ***First the nonsense***
  - ***Second, what we need to do differently?***
  - ***What can we expect if we do?***
- ***How do we make that happen?***

If we are willing:

- ***IF we are willing to continue to pursue ILP***
  - ***Because high performance tasks can not do without it***
- ***IF we are willing to break the layers***
  - ***Because we don't have a choice anymore***
- ***IF we are willing to embrace parallel programming***
- ***IF we are willing to accept more than one interface***
  - ***One for those who can exploit details of the hardware***
  - ***Another for those who can not***

Then we can provide the chips of tomorrow

- ***Performance that keeps pace with demand***
- ***Realistic power requirements***
- ***Fault tolerance***
- ***Verifiable***
- ***Secure***
- ***…***

And we can support both sets of users:

- ***For those who wish to "cure cancer"***
  - ***Chip designers must provide the low-level interface***
  - ***Knowledgeable algorithm designers will exploit it***
- ***For those who wish to use a high-level interface***
  - ***Chip designers must provide the hooks***
  - ***Knowledgeable software designers must bridge the gap***

Outline

- ***What is the transformation hierarchy?***
- ***How do we deal with it?***
  - ***First the nonsense***
  - ***Second, what we need to do differently?***
  - ***What can we expect if we do?***
- ***How do we make that happen?***

To start with: Education

i.e., Do not accept the premise:

It is okay to know only one layer.

Students can understand more than one layer

- ***What if we get rid of "Objects" FIRST***
  - ***Students do not get it – they have no underpinnings***
  - ***Objects are too high a level of abstraction***
  - ***So, students end up memorizing***
  - ***Memorizing isn't learning (and certainly not understanding)***

- ***What if we START with "motivated" bottom up***
  - ***Students build on what they already know***
  - ***Memorizing is replaced with real learning***
  - ***Continually raising the level of abstraction***

- ***The student sees the layers from the start***
  - ***The student makes the connections***
  - ***The student understands what is going on***
  - ***The layers get broken naturally***

Why do I waste your time talking about education?

- ***Microsoft developers who started with my motivated bottom-up approach say It makes better programmers out of them***
- ***Bill Gates has complained more than once about the capability of new graduates they interview***
- ***Microsoft can influence what gets taught***

Again:

- ***IF we understand more than our own layer***
  of the transformation hierarchy
  so we really can talk to each other,

- ***THEN maybe we can make all of the above happen.***

Thank you!