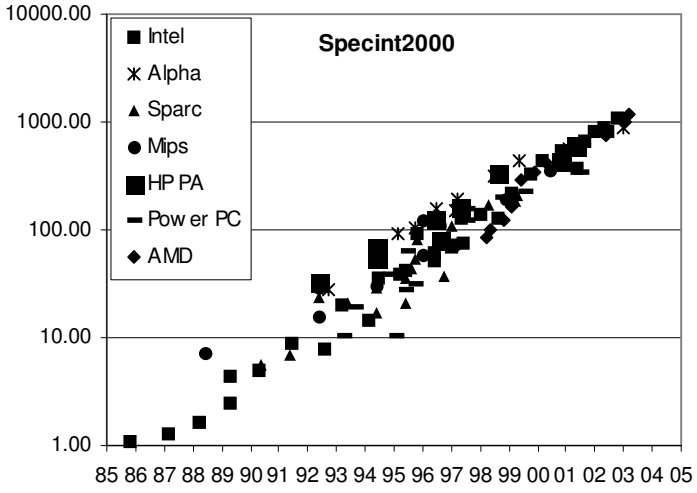
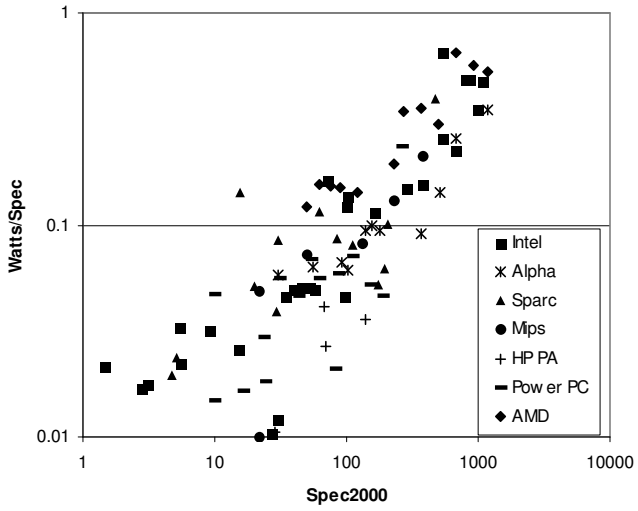
Multi-core Architectures

Rakesh Kumar
rakumar@cs.ucsd.edu

Progress of processor technology/architecture



Price being paid



Lessons learned

- Marginal utility of transistors decreasing
  - If n be the number of transistors
    - Power and Area are O(n)
    - Performance is O(sqrt(n))
      - Wrong side of square law
- Increasingly difficult to squeeze performance
  - Not enough exploitable ILP in programs
  - Easy ILP already extracted
- More transistors available than we know to how make use of when applied to a single processor

Clearly, we have a problem!

One way of handling a problem is….

- ..instead of confronting the problem try skipping to a simpler one
  - Change the focus from single-thread performance to throughput
  - Don't have increasingly complex uniprocessors
  - Have multiple simple processors on the same die instead [Olukotun *et al*, ASPLOS96]
  - Each on-chip processor (called core) can execute a program now

We can now jump to the right side of the square law

- If n be the number of transistors on a die:
  - Area = O(n)
  - Performance = $O(n^{1-x})$
    - Roughly O(sqrt(n))
- More aggregate performance (throughput) can be had using large number of small cores than small number of large cores
  - At the expense of single-thread performance
- For example,
  - In terms of area:
    - 1 EV6⇔5 EV5 cores
  - In terms of throughput:
    - 1 EV6⇔2.0-2.2 EV5 cores
    - 5EV5 cores >=2 EV6 cores
      - Performance doubled just by having multiple cores!

The main motivation for having multi-core architecures

Multi-core Architecture: Definition

A multi-core architecture (or a chip multiprocessor) is a general-purpose processor that consists of multiple cores on the same die and can execute programs simultaneously

Multi-core architecture: Advantages

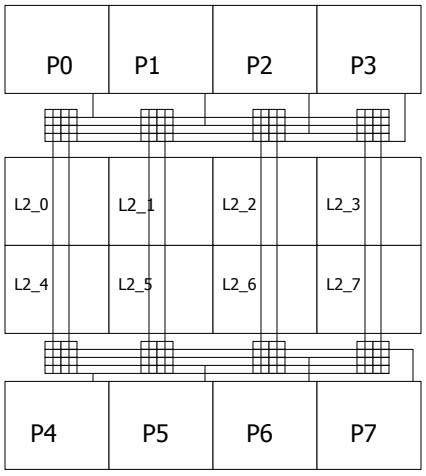
- (Relatively) High performance/watt
- (Relatively) High performance/area
- Simpler core
  - Possibility of lower cycle time, better optimisation etc.
  - Ease of design, verification etc.

So, the next question to ask obviously is…

How should one design a multi-core architecture?

This is the question I address in my thesis research

A Naive methodology for Multi-core Design



Clean, easy way to design Multi-core oblivious multi-core design!

Goals of my thesis research

- Demonstrate that the prior methodology is highly inefficient in terms of area and power
- Demonstrate the need to do holistic design of multi-core architectures
  - Subsystem design should be aware of the multi-core architecture it is going to be a part of
- Propose and evaluate novel and efficient multi-core architecture design methodologies that follow a holistic approach

Assumptions inherent to the naïve approach

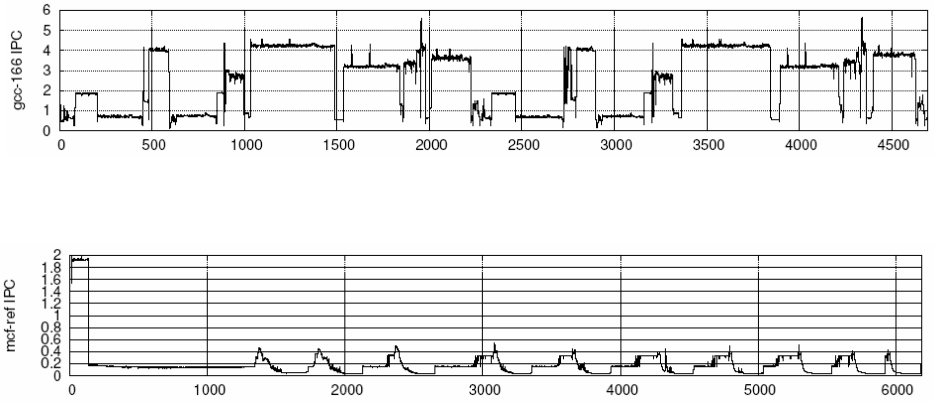
- All cores have to be the same
- Each core is distinct
- Core/memory and interconnect can be designed in isolation

I will talk about the first assumption today

Before scrutinizing the “identical cores” assumption...

…let’s consider characteristics of typical workloads

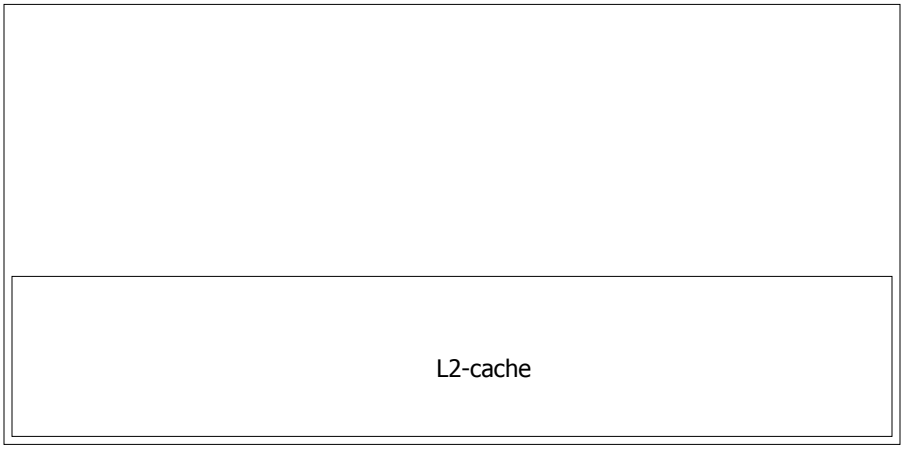
There is enormous diversity among applications



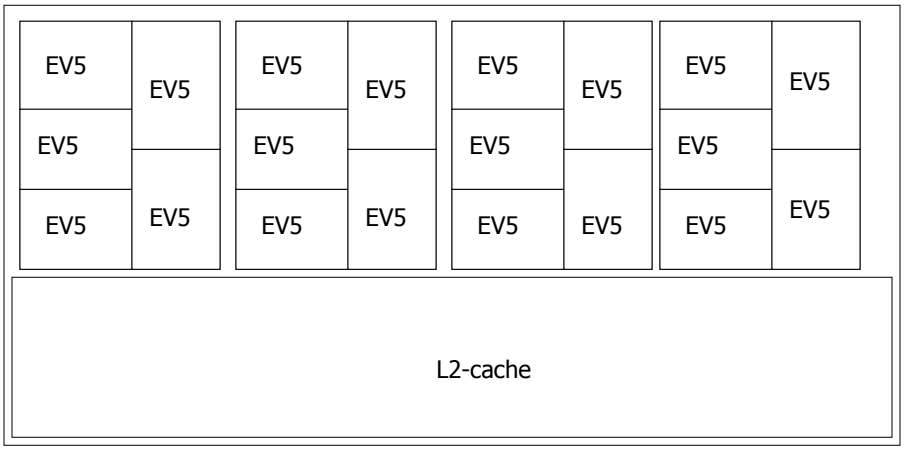
Implication of diversity on multi-core design

- If all cores are to be identical, then can't address diverse workload demands
  - E.g. need to decide beforehand if the core targets gcc or mcf
- Either way one application loses
  - Underutilization or low performance

An example multi-core architecture



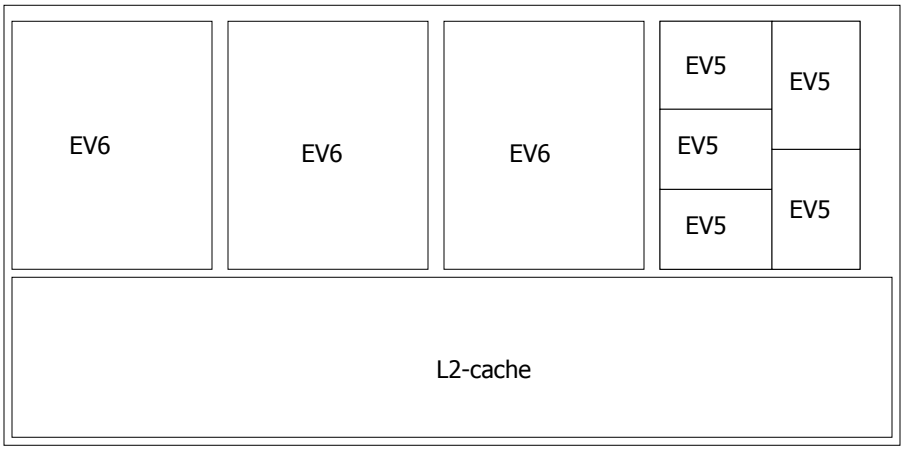
An example multi-core architecture



Processors and Program diversity

- Some applications will run much faster on an EV6 than on an EV5
- Others will take little advantage of the larger processor and run at the same speed on either
- With a homogeneous architecture,
  - you either have the former running very slowly on small processors,
  - or the latter unnecessarily wasting the capabilities of the large processor.

An alternate multi-core architecture



An alternate multi-core architecture

We can potentially have eight jobs running at or close to EV6 speeds, in the space of 4 EV6s !

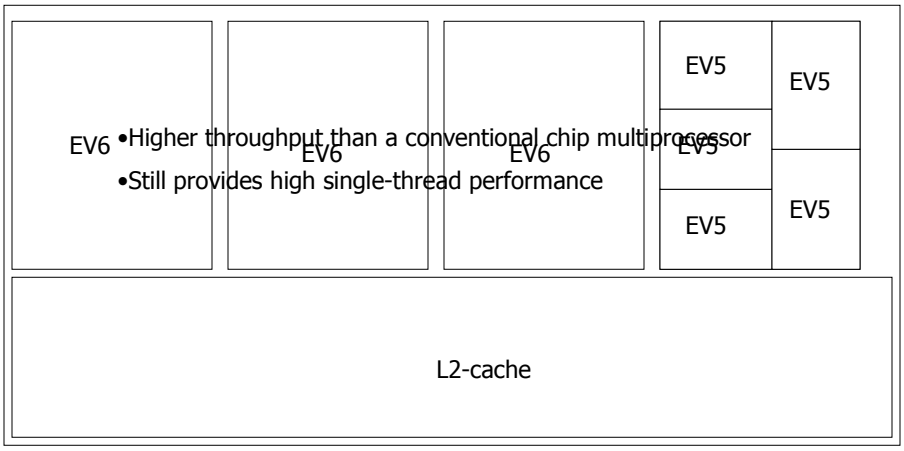
Single-ISA Heterogeneous Multi-core Architectures

- Have multiple heterogeneous cores on the same die
  - Each core-type represents a different point in the power performance space
    - i.e. while one core-type might be small low-performance, low-power, some other core-type might be big high performance, high power
  - Each core capable of executing the same ISA
    - Unlike SoCs/embedded heterogeneous multi-core architectures

Such an architecture will be highly efficient on workloads with diverse applications

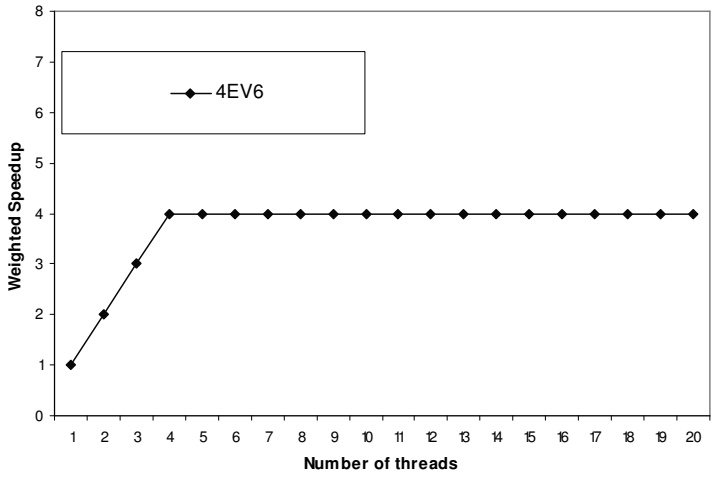
Another Performance Advantage: Adjusts to varying TLP

Another Performance Advantage: Adjusts to varying TLP

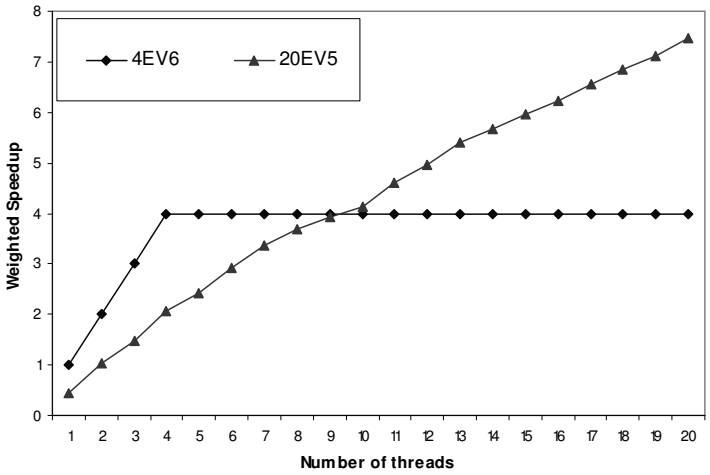


- Higher throughput than a conventional chip multiprocessor
- Still provides high single-thread performance

Comparing Single-ISA Heterogeneous Architectures against Conventional CMPs

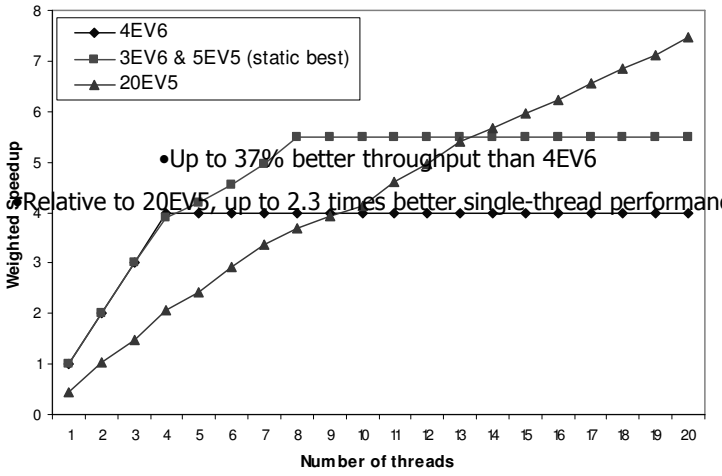


Comparing Single-ISA Heterogeneous Architectures against Conventional CMPs



A choice has to be made between throughput and ST performance

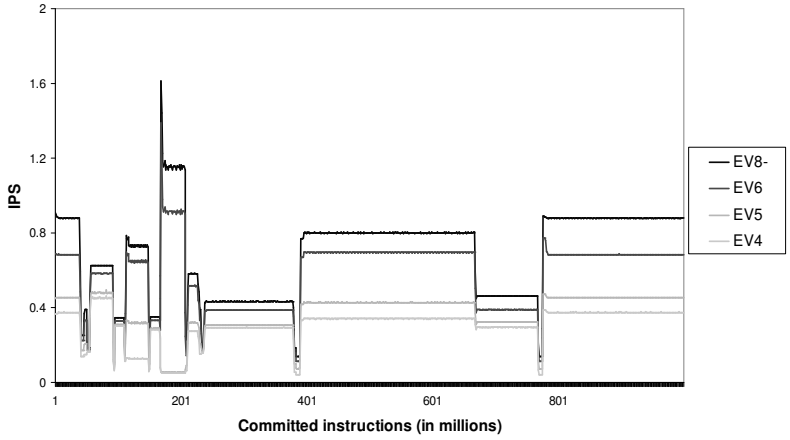
Comparing Single-ISA Heterogeneous Architectures against Conventional CMPs



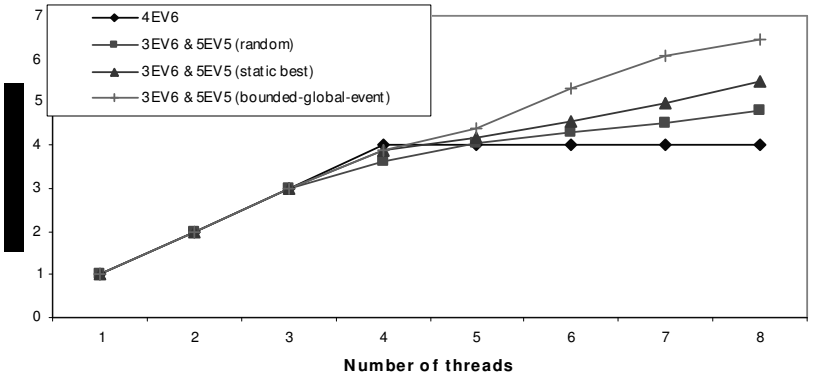
- Up to 37% better throughput than 4EV6
- Relative to 20EV5, up to 2.3 times better single-thread performance

Best of both the worlds!

Then there is intra-program diversity as well!



Dynamic scheduling results



To sum up….

- Single-ISA Heterogeneous architectures a good design point for throughput as well as performance:
  - Efficient use of die-area for a given thread-level parallelism
    - Provides low-latency for few application on powerful cores
    - A large number of applications can be hosted at once on simple cores
  - Efficient adaptation to application diversity
    - Enables it approach the performance of an architecture with a large number of complex cores
    - Provides higher performance in the same area than a conventional chip multiprocessor

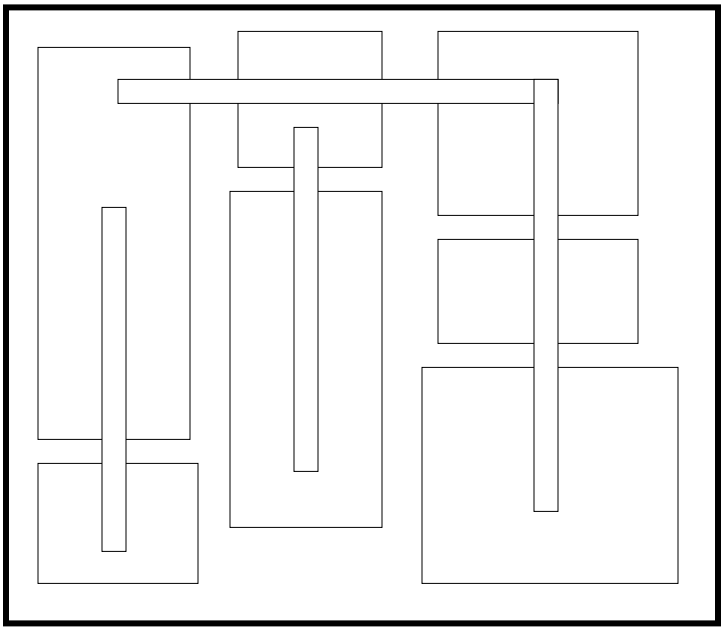
Talk Outline

- ~~All cores have to be the sa~~me
  - Single-ISA heterogeneous multi-core architectures
    - Performance Benefits
    - Power Benefits

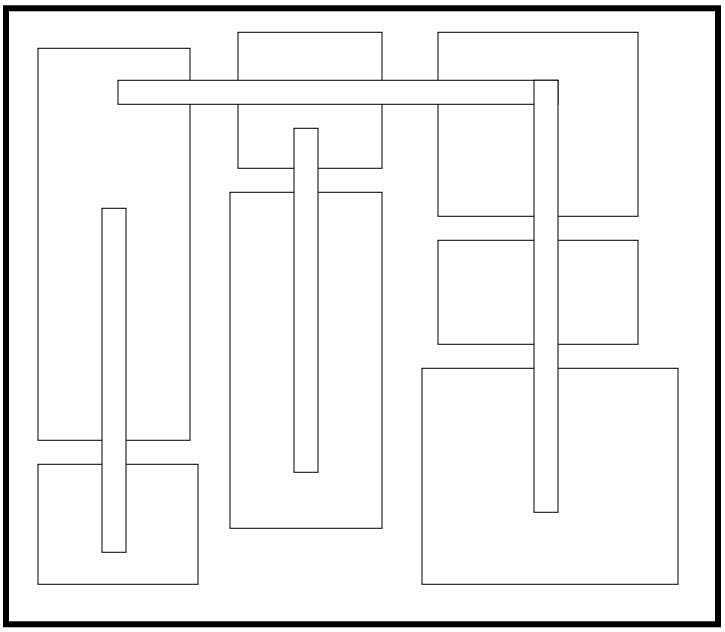
Reducing power for a conventional multi-core architecture

- Done at the core-level
  - Each core optimised for power and then replicated multiple times
  - Multi-core oblivious
- Processor power reduction typically involves V/f scaling, gating etc for the core
- Power reduction techniques applied at single-core level have limited effectiveness

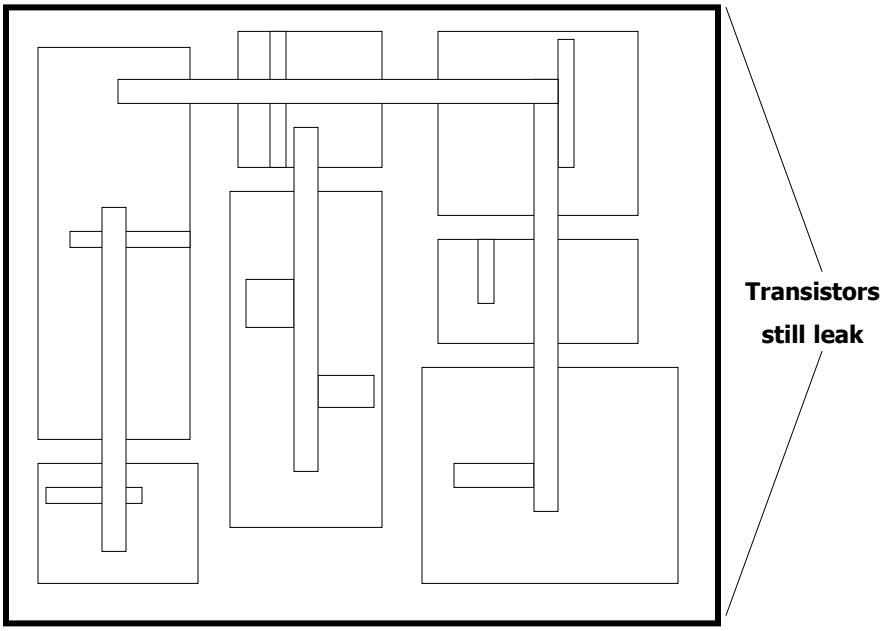
Clock Gating

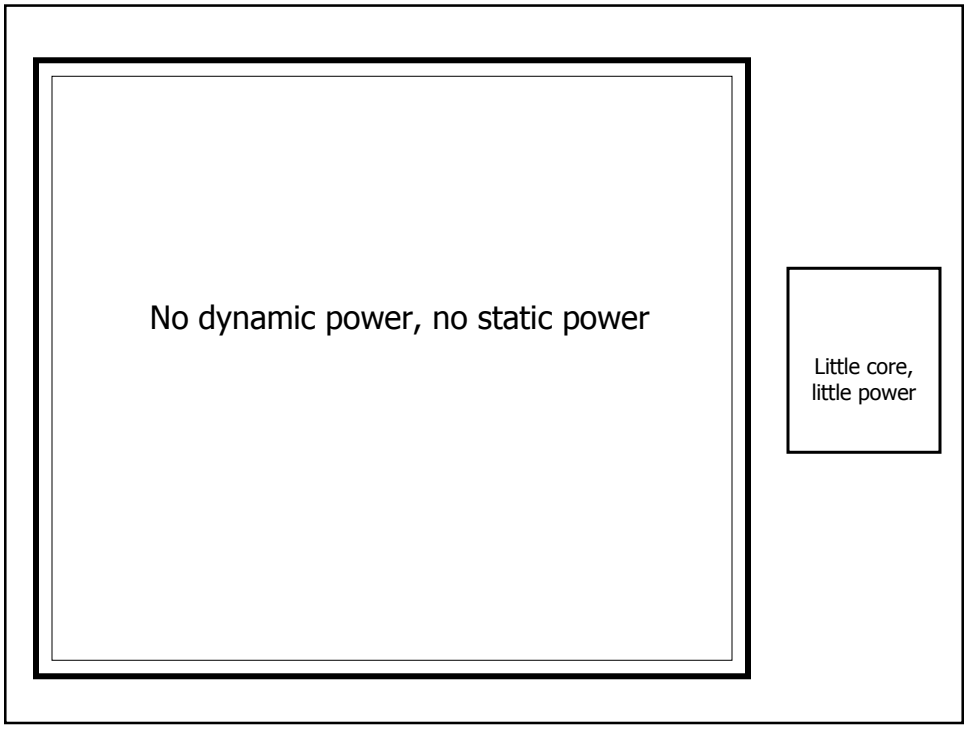


Clock Gating



Clock Gating





Single-ISA Heterogeneous Multi-Core Architectures for Processor Power Reduction

- Have multiple heterogeneous cores on the same die
- Match workload (or workload phase) to core that achieves best efficiency according to some objective function
- Power down the unused cores completely

An example Single-ISA heterogeneous multi-core architecture

- Consider a processor with four generation of Alpha cores on the same die
- All cores assumed to be implemented in 0.1micron and clocked at 2.1GHz, input voltage 1.2V

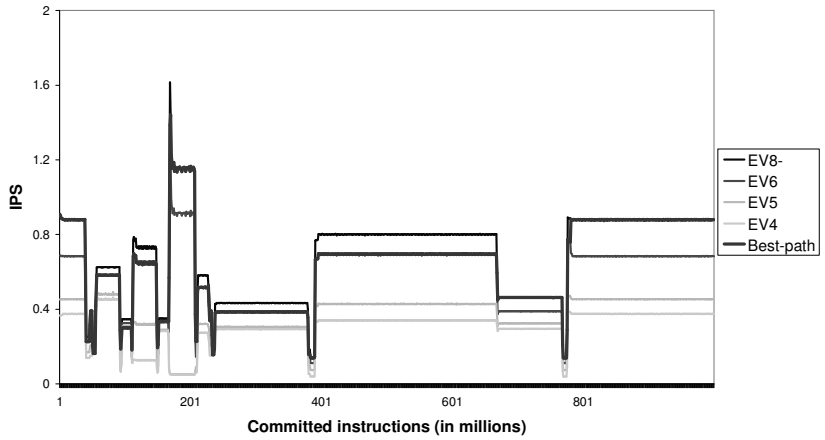
Processor	Peak-power (in W)	Core-area (in mm^2)
EV4	4.97	3
EV5	9.83	5
EV6	17.80	24
EV8-	92.88	260

The processor only marginally bigger than EV8- !

The objective is to switch to a simpler core and still be within performance bounds

Choosing Dynamically the Core with Least Energy (perf. loss<10%)

Choosing Dynamically the Core with Least Energy (perf. loss<10%)



All cores get used

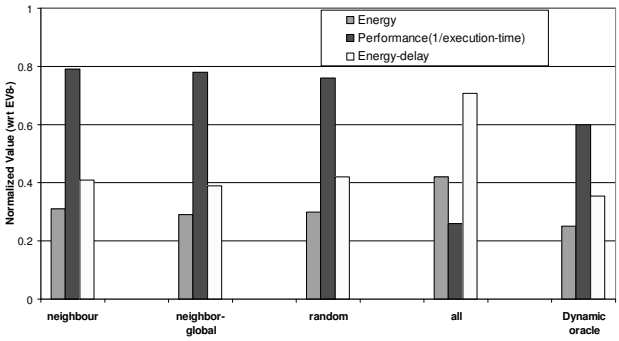
Choosing Dynamically the Core with Least Energy (perf. loss<10%) [Summary of results]

	Energy Savings(%)	Performance Degradation(%)
Maximum	**77.3**	**8.5**
Minimum	**0.1**	**0.1**
Mean	**38.5**	**3.4**

Results "verified" by other researchers using real prototypes

[Grochowski ICCD2004, Ghiasi CF2005]

Realistic heuristics



Achieves up to 93% of the oracle

To sum up…

- A single-ISA heterogeneous multi-core architecture offers enormous potential for even power-savings
- Realistic heuristics can achieve much of the savings potential
- Beats chip-wide voltage scaling handsomely (50.6% ED^2 improvement)
  - Subsequent research has shown this technique to better than dynamic V/f scaling, gating, adaptive optimizations etc. [Grochowski et al ICCD2004]

Bottomline

All cores do not have to be the same

In fact, should not be same

Summary of talk

- Decreasing marginal utility of transistors is leading us to multi-core architectures
- Conventional multi-core architectures have identical cores
- Having heterogeneous architectures lead to higher performance and lower power