

GSRC Soft-Systems Vision of Long-term Microarchitecture Research



Wen-mei Hwu and Kurt Keutzer

with

John W. Sias, Erik M. Nystrom, Hong-seok Kim, Chien-wei Li,
Hillary C. Hunter, Shane Ryoo, Sain-Zee Ueng, James W. Player,
Ian M. Steiner, Chris I. Rodrigues, Robert E. Kidd,
Dan R. Burke, Nacho Navarro, Steve S. Lumetta
University of Illinois at Urbana-Champaign

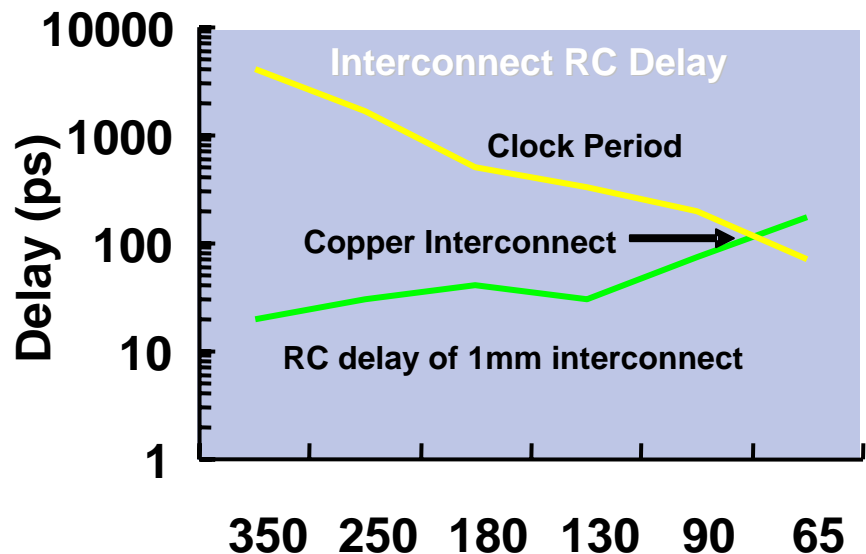
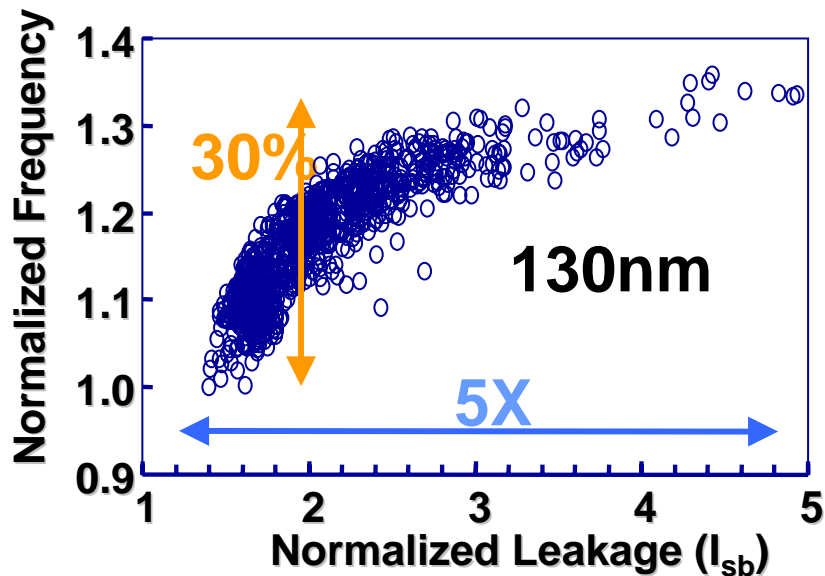
and

Yujia Jin, Andrew Mihal, Matt Moskewcz, Will Plishker,
Kaushik Ravindran, N. R. Satish, Scott Weber
University of California at Berkeley

Powerful trends

◆ Technology trends

- ◆ Increasing speed, power variability of transistors limit clock frequency increase
- ◆ Increasing interconnect delay and shrinking clock domains limit the size of individual computing engines.
- ◆ Soft errors, signal integrity, and defects require new approaches to reliability



Powerful Trends (cont.)

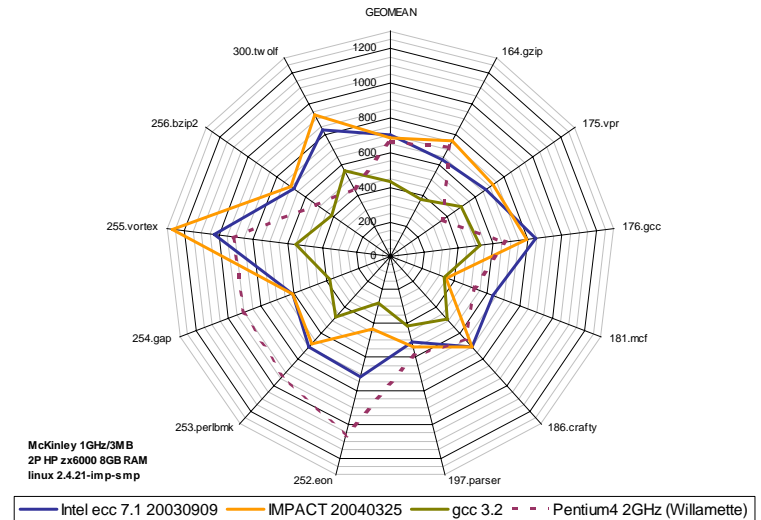


◆ Architectural trends

- ◆ No future scaling path for monolithic microarchitectures
 - ◆ Too much power, too much cache, too hard to verify, diminishing returns
- ◆ No single processing model will cover all applications – lesson from Pentium 4 vs. Itanium 2
- ◆ Traditional multi-core approach is unlikely to be fruitful without major software and microarchitecture retooling

Semiconductor Technology Roadmap (2001/02)

| Year | 2004 | 2007 | 2013 | 2016 |
|--------------------------------------|------|------|------|------|
| Technology generation (nm) | 90 | 65 | 32 | 22 |
| Wafer size (mm) | 300 | 300 | 450 | 450 |
| Defect density (per m ²) | 1356 | 1356 | 1116 | 1116 |
| μP die size (mm ²) | 310 | 310 | 310 | 310 |
| Chip Frequency (GHz) | 4 | 6.7 | 19.4 | 29 |
| MTx per Chip (Microprocessor) | 552 | 1204 | 3424 | 6200 |
| MaxPwr(W) High Performance | 160 | 190 | 251 | 290 |



Powerful Trends (cont.)



◆ Business trends

- ◆ >\$2B per fabrication facility, >\$30M per IC total design cost, >\$1M mask set, >\$1M CAD tool investment per design engineer – decreasing ASIC design starts and strong demand for programmable replacements

◆ Software trends

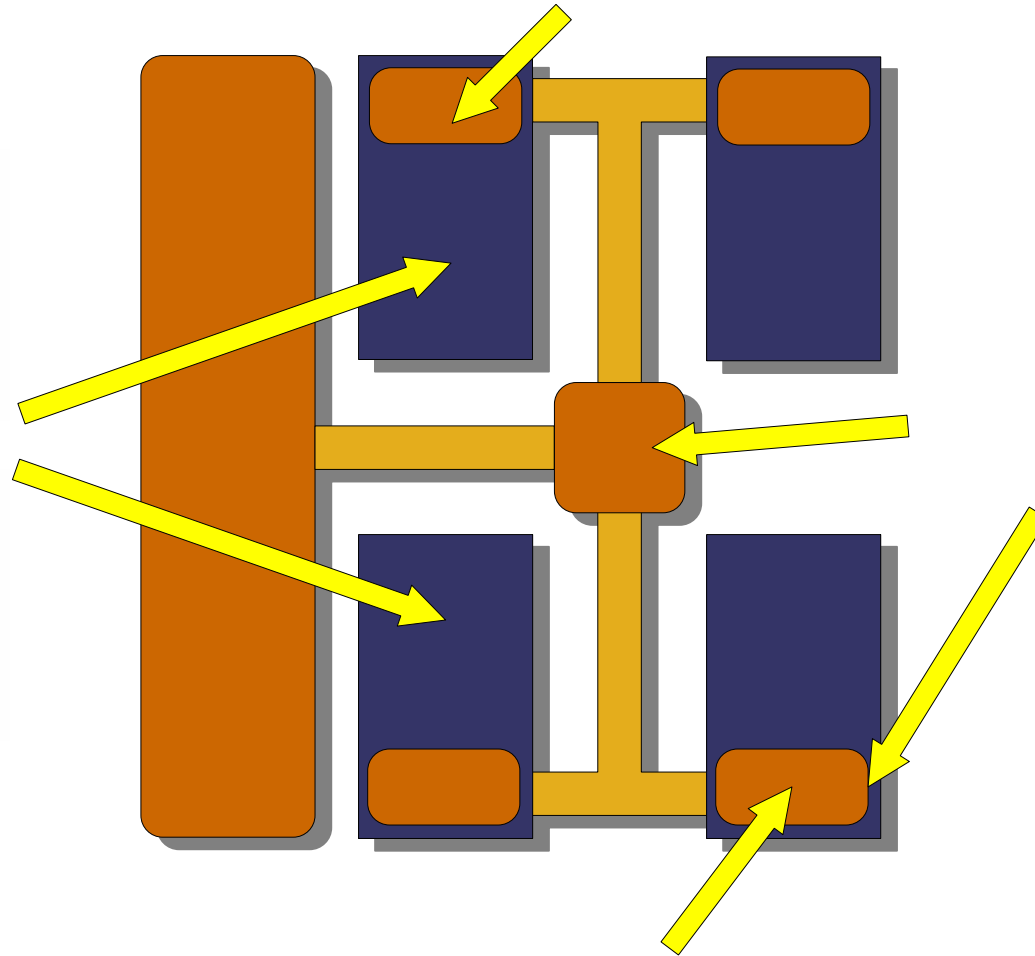
- ◆ Relatively more and more processor cycles will be devoted to applications that deal with physical world – rich in inherent parallelism
- ◆ More and more application source will be available to advanced compilers – open source movement and Microsoft Phoenix
- ◆ New programming models and environments motivated by productivity will increase compiler's ability to enhance parallelism and manage data movement
- ◆ Deep, scalable software analysis systems becoming available to application compilation at production level

Bottom line analysis

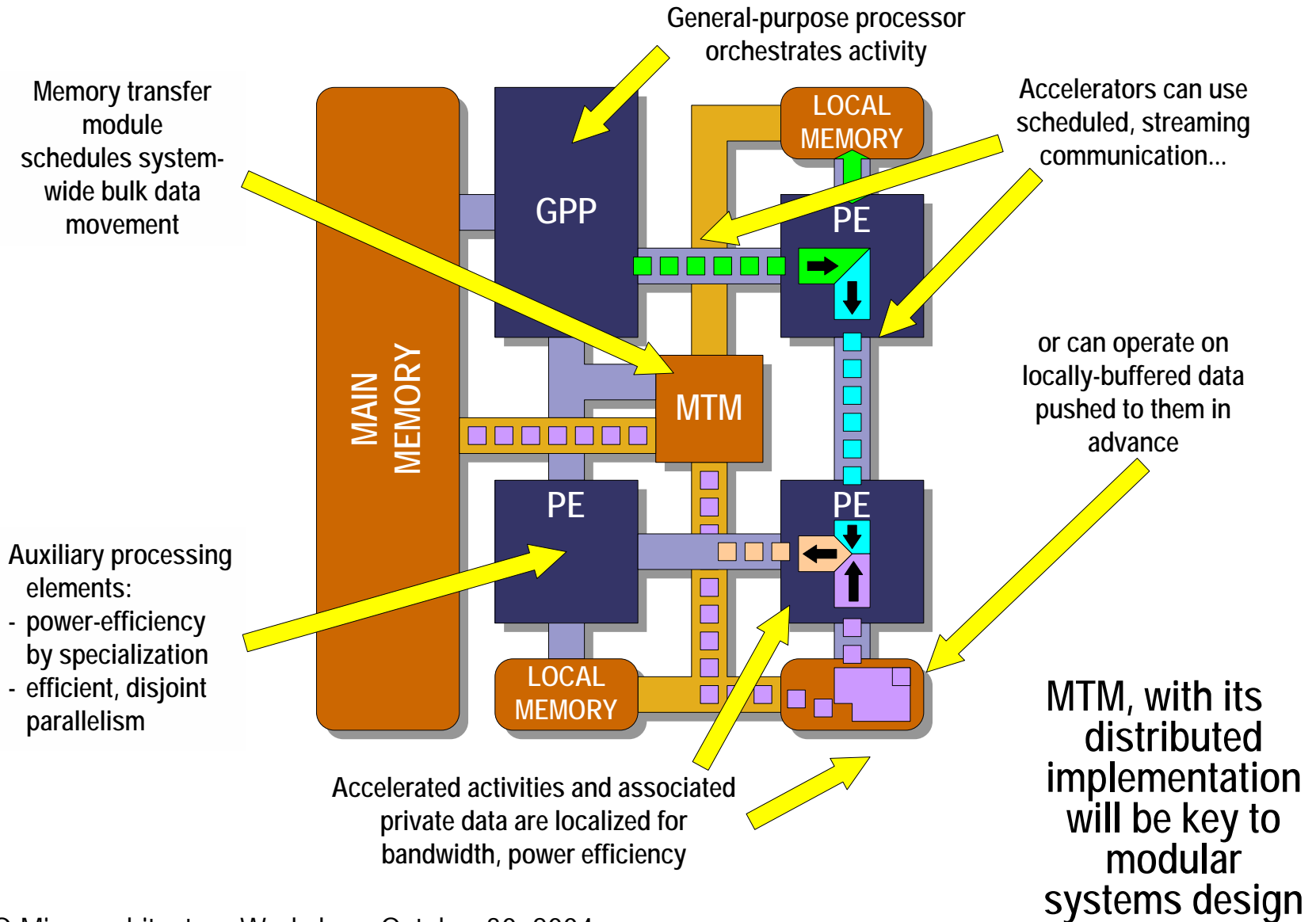


- ◆ Thread/task-level parallelism will be the main venue for continued scaling
 - ◆ Expending power while developing parallelism is decreasingly viable
 - ◆ Compilers and programmers must deliver, saving energy for computation
 - ◆ Spatial computing, decentralized models key to efficiency and reliability
 - ◆ Already prevalent in certain embedded contexts—will also dominate general purpose
- ◆ Processors will be heterogeneous compute engines tailored to their application
 - ◆ Some tailoring done at chip design time via special purpose on-chip application engines
 - ◆ Some tailoring done at user programming time via reconfigurable logic, interconnects
- ◆ Distributed memory model and proactive data management will be critical to sustaining parallel execution
 - ◆ Bulk cache structures need to be replaced or supplemented by localized storage that provide massive bandwidth at much lower power and latency
 - ◆ Pointers and dynamically allocated data structures must be systematically managed in the memory model

Near-term chip multiprocessor



Soft Systems microarchitecture model

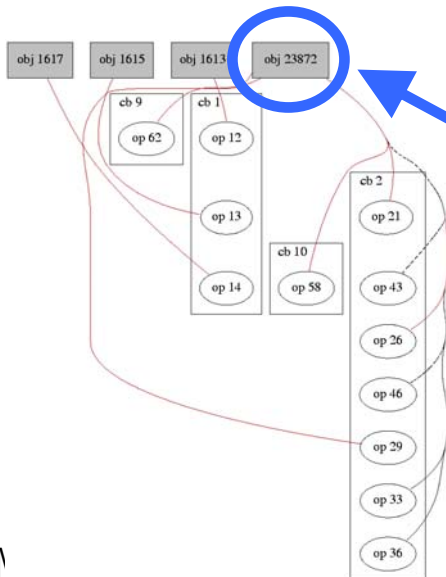


More on accuracy

Inaccurate Analysis: Steensgaard



Accurate Analysis: Andersen, Field-sensitive, Context-sensitive, Heap cloning



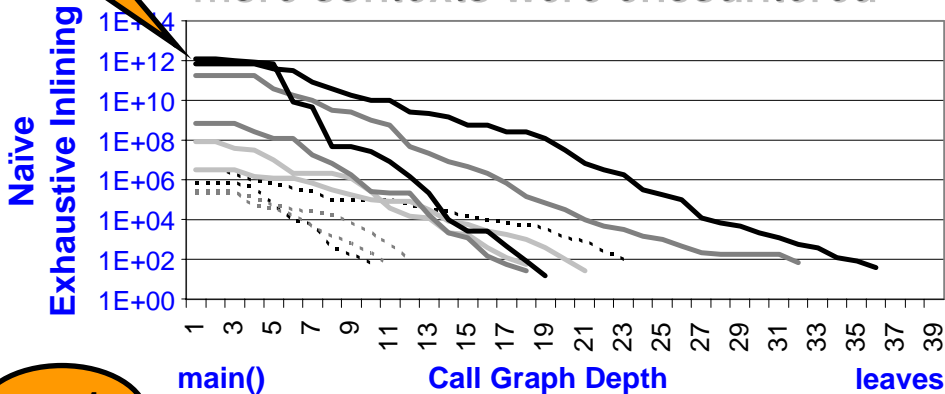
**Just One Heap Object
(calloc() allocation)**

MPEG-4 Decoder, GetContextInter()

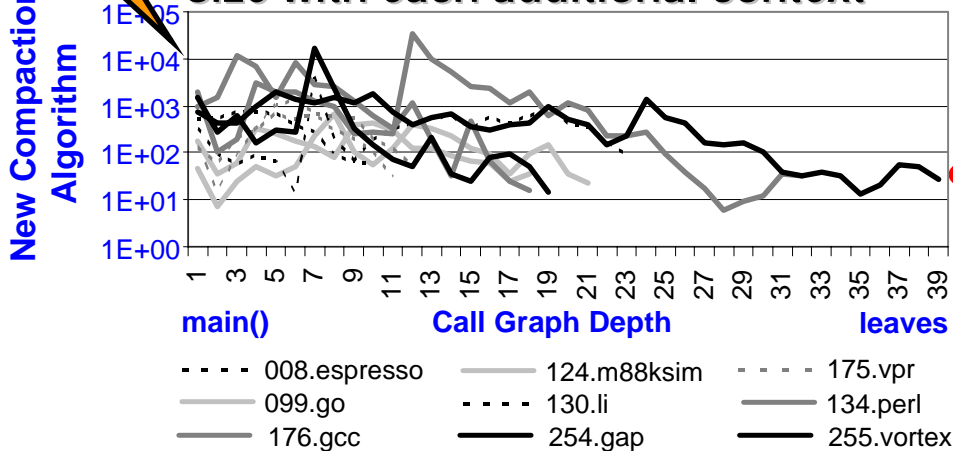
Analyzing large, complex programs [SAS2004]



10¹² Originally, problem size exploded as more contexts were encountered



10⁴ New algorithm contains problem size with each additional context

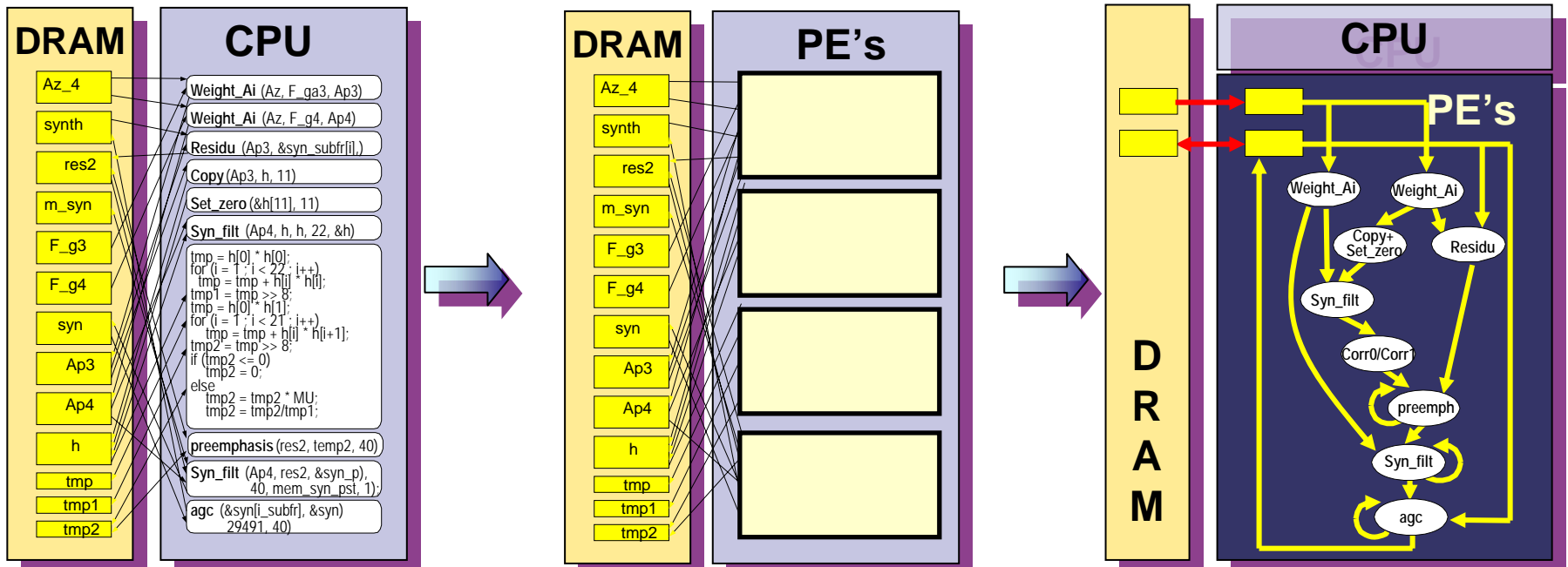


This results in an efficient analysis process without loss of accuracy

| Benchmark | INACCURATE Context Insensitive (seconds) | PREV Context Sensitive (seconds) | NEW Context Sensitive (seconds) |
|-----------|--|----------------------------------|---------------------------------|
| espresso | 2 | 9 | 1 |
| li | 1 | 1332 | 1 |
| jpeg | 2 | 85 | 1 |
| perl | 4 | 408 | 11 |
| gcc | 52 | HOURS | 124 |
| perlbmk | 155 | MONTHS | 198 |
| gap | 62 | 3350 | 117 |
| vortex | 5 | 136 | 3 |
| twolf | 1 | 2 | 1 |

Question 1: Why is this different from previous parallelizing compilers and multiprocessors?

- ◆ We are extending these previous work in three critical directions
 - Extraction of independent memory objects and conversion of implicit memory data flow into managed data movement and communication
 - Applicability to mass software base - requires pointer analysis, control flow analysis, beyond traditional array dependence analysis
 - Incorporation of high-level programming models with deep program analysis



Question 2: Haven't we tried the compiler approach in Itanium?



- ◆ We proved that one can build compilers with aggressive whole program analysis and ILP techniques that achieve excellent results on large applications
 - ◆ Whole program compilation with GCC compatibility, new Microsoft Phoenix initiative, Oracle success – many of the traditional roadblocks to pervasive optimizing compilation have been removed

- ◆ We did not (attempt to) prove that the compiler is capable of parallelizing applications at the memory and thread/task level
 - ◆ Now a major focus of the GSRC Soft Systems work in compilation of traditional and programming models
 - ◆ Need to develop (micro)architectural models for proactive management of memory and communication

Microarchitecture research focus of Soft Systems



- ◆ **Hardware: predictable PE's plugged into a heterogeneous multiprocessing system with intelligent interface units**
 - ◆ Intelligent interface units cooperatively move data and code to fully utilize processing elements under compiler and runtime control
 - ◆ Interface units help to maintain desirable software properties, contain software errors, and enable debugging
 - ◆ Processing engines that embody major forms of parallelism to cover a wide range of applications: VLIW, systolic arrays, SIMD
- ◆ **Software: software model and tools will make or break the microarchitectures**
 - ◆ Goal oriented research on robust memory and thread level analysis and transformations, integration of programming models and analysis techniques to harness computational power of highly parallel systems
 - ◆ Continue to extend compilation into major code bases: open source movement, Oracle, Microsoft Phoenix initiative, etc.
 - ◆ Aggressive migration of important application and system code into spatial accelerators