The Berkeley View: A New Framework & a New Platform for Parallel Research

David Patterson and a cast of thousands
Pardee Professor of Computer Science, U.C. Berkeley
Director, RAD Lab, U.C. Berkeley
Past President, Association for Computing Machinery
November, 2006

High Level Message

- Everything is changing
- Old conventional wisdom is out
- We DESPERATELY need a new approach to HW and SW systems based on parallelism
- Need to create a "watering hole" to bring everyone together to quickly find that solution
  - architects, language designers, application experts, numerical analysts, algorithm designers, programmers, …

Outline

- **Part I: A New Agenda for Parallel HW/SW Systems**
  - Old Conventional Wisdom vs. New Conventional Wisdom
  - 7 Questions to Frame Parallel Research
  - New Benchmarks for New Architectures
  - Hardware Building Blocks
  - Innovating at HW/SW interface without Compilers
  - Seemingly Obvious but Neglected Points
  - Reasons for Optimism towards Parallel Computing Revolution
- **Part II: A "Watering Hole" for Parallel HW/SW Systems**
  - **Research Accelerator for Multiple Processors**

Conventional Wisdom (CW) in Computer Architecture

1. *Old CW*: Power is free, but transistors expensive
- *New CW* is the "*Power wall*":
  Power is expensive, but transistors are "free"
  - Can put more transistors on a chip than have the power to turn on
2. *Old CW*: Only concern is dynamic power
- *New CW*: For desktops and servers, static power due to leakage is 40% of total power
3. *Old CW*: Monolithic uniprocessors are reliable internally, with errors occurring only at pins
- *New CW*: As chips drop below 65 nm feature sizes, they will have high soft and hard error rates

Conventional Wisdom (CW) in Computer Architecture

4. *Old CW*: By building upon prior successes, continue raising level of abstraction and size of HW designs
- *New CW*: Wire delay, noise, cross coupling, reliability, clock jitter, design validation, … stretch development time and cost of large designs at ≤65 nm

5. *Old CW*: Researchers demonstrate new architectures by building chips
- *New CW*: Cost of 65 nm masks, cost of ECAD, and design time for GHz clocks
  ⇒ Researchers no longer build believable chips

6. *Old CW*: Performance improves latency & bandwidth
- *New CW*: BW improves > (latency improvement)^2

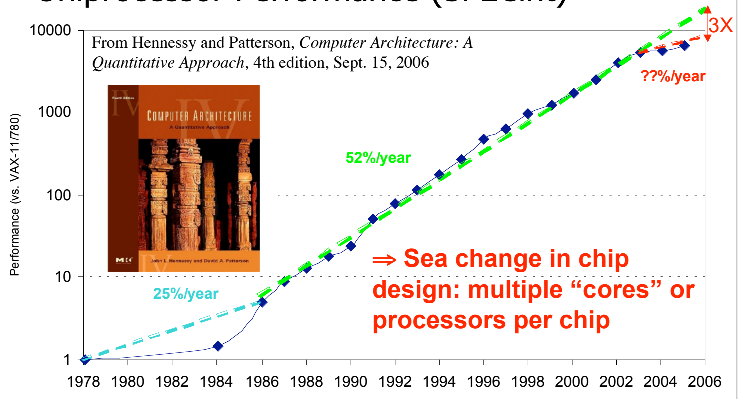
Conventional Wisdom (CW) in Computer Architecture

7. *Old CW*: Multiplies slow, but loads and stores fast
- *New CW* is the "*Memory wall*":
  Loads and stores are slow, but multiplies fast
  - 200 clocks to DRAM, but even FP multiplies only 4 clocks

8. *Old CW*: We can reveal more ILP via compilers and architecture innovation
   - Branch prediction, OOO execution, speculation, VLIW, …
- *New CW* is the "*ILP wall*":
  Diminishing returns on finding more ILP

9. *Old CW*: 2X CPU Performance every 18 months
- *New CW* is *Power Wall + Memory Wall + ILP Wall = Brick Wall*

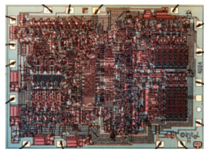
Uniprocessor Performance (SPECint)

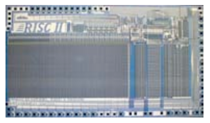


- **VAX : 25%/year 1978 to 1986**
- **RISC + x86: 52%/year 1986 to 2002**
- **RISC + x86: ??%/year 2002 to present**

Sea Change in Chip Design

- Intel 4004 (1971): 4-bit processor, 2312 transistors, 0.4 MHz, 10 micron PMOS, 11 mm^2 chip
- RISC II (1983): 32-bit, 5 stage pipeline, 40,760 transistors, 3 MHz, 3 micron NMOS, 60 mm^2 chip
- 125 mm^2 chip, 0.065 micron CMOS = 2312 RISC II+FPU+Icache+Dcache
  - RISC II shrinks to ≈ 0.02 mm^2 at 65 nm
  - Caches via DRAM or 1 transistor SRAM (www.t-ram.com) or 3D chip stacking
  - Proximity Communication via capacitive coupling at > 1 TB/s ? (Ivan Sutherland @ Sun / Berkeley)





- **Processor is the new transistor!**

Conventional Wisdom (CW) in Computer Architecture

10. *Old CW*: Increasing clock frequency is primary method of performance improvement
- *New CW*: Processors Parallelism is primary method of performance improvement

11. *Old CW*: Don't bother parallelizing app, just wait and run on much faster sequential computer
- *New CW*: Very long wait for faster sequential CPU
  - 2X uniprocessor performance takes 5 years?
  - End of La-Z-Boy Programming Era

12. *Old CW*: Less than linear scaling for a multiprocessor is failure
- *New CW*: Given the switch to parallel hardware, even sublinear speedups are beneficial

Need a New Approach

- Berkeley researchers from many backgrounds met between February 2005 and October 2006 to discuss parallelism
  - Circuit design, computer architecture, massively parallel computing, computer-aided design, embedded hardware and software, programming languages, compilers, scientific programming, and numerical analysis
- Krste Asanovíc, Rastislav Bodik, Bryan Catanzaro, Joseph Gebis, Parry Husbands, Kurt Keutzer, William Plishker, John Shalf, Samuel Williams, and Katherine Yelick + others
- Tried to learn from successes in embedded and high performance computing
- Led to 7 Questions to frame parallel research

7 Questions for Parallelism

- **Applications:**

1. What are the apps?
2. What are kernels of apps?

- **Hardware:**

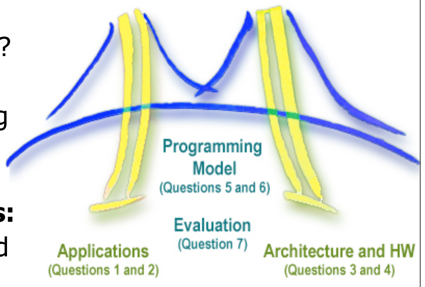
3. What are the HW building blocks?
4. How to connect them?

- **Programming Models:**

5. How to describe apps and kernels?
6. How to program the HW?

- **Evaluation:**

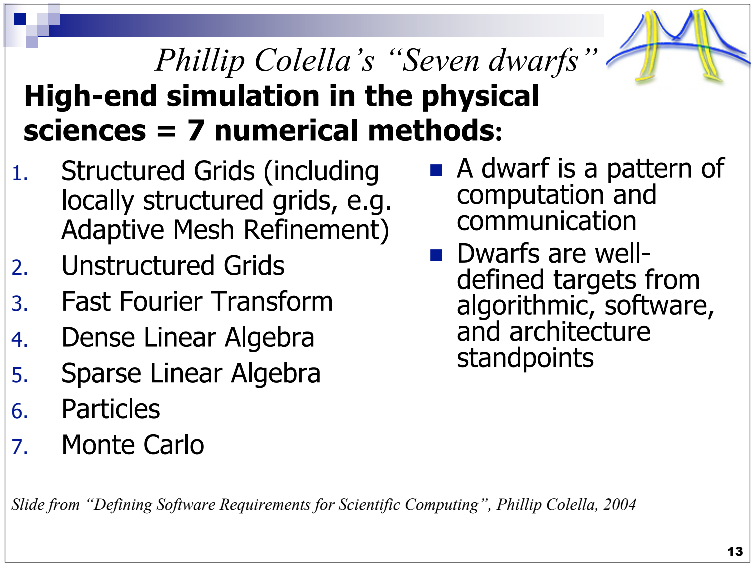
7. How to measure success?



(Inspired by a view of the Golden Gate Bridge from Berkeley)

Apps and Kernels

- Old CW: Since cannot know future programs, use old programs to evaluate computers for future, e.g., SPEC2006
- What about parallel codes?
  - Few, tied to old models, languages, architectures, …
- New approach: Design computers of future for patterns of computation and communication that will be important in the future
- Claim: 14 "dwarfs" are key for next decade, so design for them!
  - Representative codes may vary over time, but these dwarfs will be important for > 10 years



Phillip Colella's "Seven dwarfs"

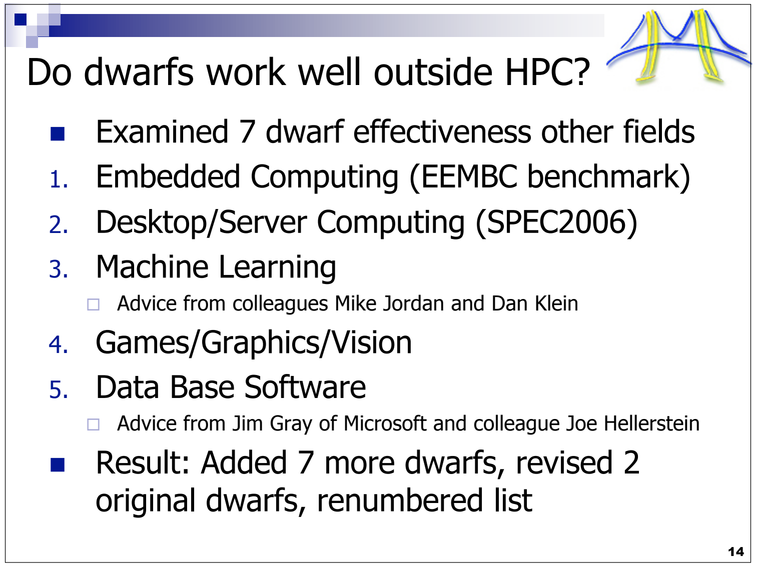
High-end simulation in the physical sciences = 7 numerical methods:

1. Structured Grids (including locally structured grids, e.g. Adaptive Mesh Refinement)
2. Unstructured Grids
3. Fast Fourier Transform
4. Dense Linear Algebra
5. Sparse Linear Algebra
6. Particles
7. Monte Carlo

- A dwarf is a pattern of computation and communication
- Dwarfs are well-defined targets from algorithmic, software, and architecture standpoints

Slide from "Defining Software Requirements for Scientific Computing", Phillip Colella, 2004

13



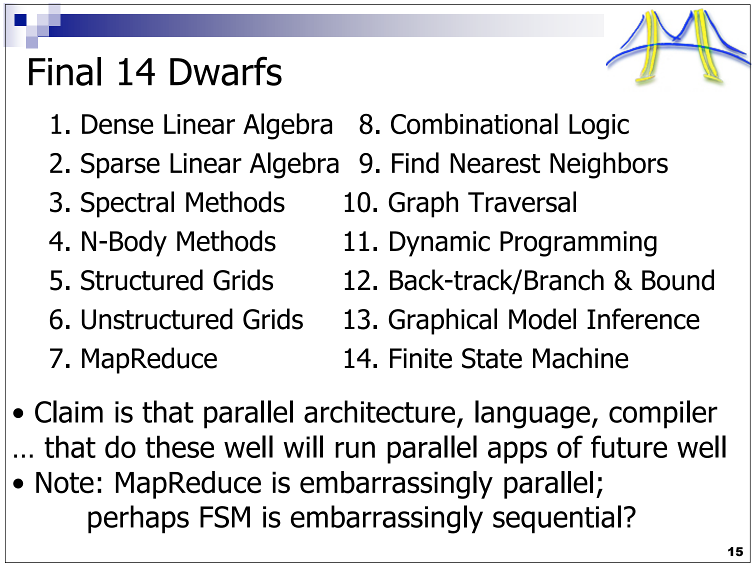
Do dwarfs work well outside HPC?

- Examined 7 dwarf effectiveness other fields

1. Embedded Computing (EEMBC benchmark)
2. Desktop/Server Computing (SPEC2006)
3. Machine Learning
   - Advice from colleagues Mike Jordan and Dan Klein
4. Games/Graphics/Vision
5. Data Base Software
   - Advice from Jim Gray of Microsoft and colleague Joe Hellerstein

- Result: Added 7 more dwarfs, revised 2 original dwarfs, renumbered list

14

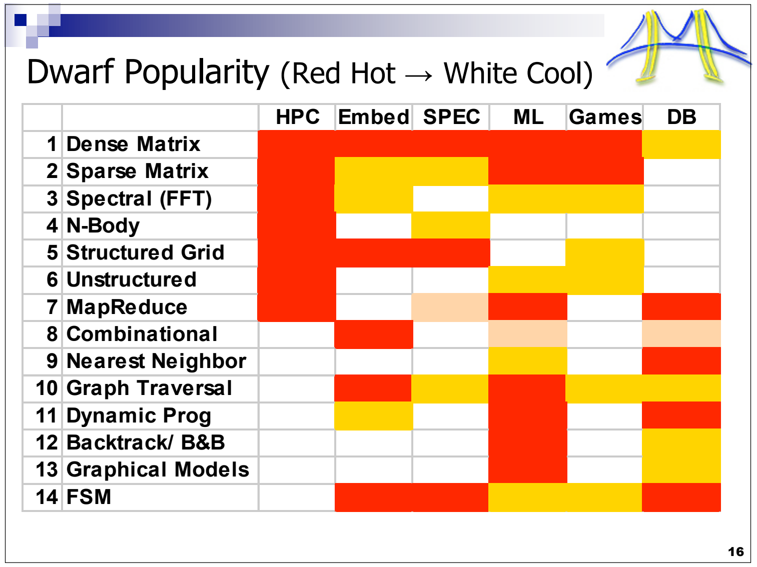


Final 14 Dwarfs

1. Dense Linear Algebra	8. Combinational Logic
2. Sparse Linear Algebra	9. Find Nearest Neighbors
3. Spectral Methods	10. Graph Traversal
4. N-Body Methods	11. Dynamic Programming
5. Structured Grids	12. Back-track/Branch & Bound
6. Unstructured Grids	13. Graphical Model Inference
7. MapReduce	14. Finite State Machine

- Claim is that parallel architecture, language, compiler … that do these well will run parallel apps of future well
- Note: MapReduce is embarrassingly parallel; perhaps FSM is embarrassingly sequential?

15



Dwarf Popularity (Red Hot → White Cool)

		HPC	Embed	SPEC	ML	Games	DB
1	Dense Matrix	🟥	🟥	🟥	🟥	🟥	🟨
2	Sparse Matrix	🟥	🟨	🟨	🟥	🟥	
3	Spectral (FFT)	🟥	🟨		🟨	🟨	
4	N-Body	🟥		🟨			
5	Structured Grid	🟥	🟥	🟥		🟨	
6	Unstructured	🟥			🟨	🟨	
7	MapReduce	🟥		🟧	🟥		🟥
8	Combinational		🟥		🟧		🟧
9	Nearest Neighbor				🟨		🟥
10	Graph Traversal		🟥	🟨	🟥	🟨	🟨
11	Dynamic Prog		🟨		🟥		🟥
12	Backtrack/ B&B				🟥		🟨
13	Graphical Models				🟥		🟨
14	FSM		🟥	🟥	🟨	🟨	🟥

16

Hardware Building Blocks: Small is Beautiful

- Given difficulty of design/validation of large designs
- Given power limits what can build, parallel is energy efficient way to achieve performance
  - Lower threshold voltage means much lower power
- Given redundant processors can improve chip yield
  - Cisco Metro 188 processors + 4 spares
  - Sun Niagara sells 6 or 8 processor version
- Expect modestly pipelined (5- to 9-stage) CPUs, FPUs, vector, SIMD PEs
- One size fits all?
  - Amdahl's Law ⇒ a few fast cores + many small cores

17

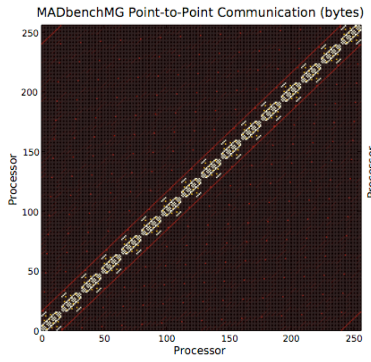
Number of Cores/Socket

- We need revolution, not evolution
- "Multicore" industry starts too small, double number of cores per generation: 2, 4, 8, 16,
- "Manycore" 100s to 1000s is highest performance per unit area, then double per generation: 128, 256, 512, 1024 …
- Multicore architectures & programming models suitable for 2 to 32 cores not likely to successfully evolve to Manycore systems of 1000's of processors
  ⇒ Desperately need HW/SW models that work for Manycore

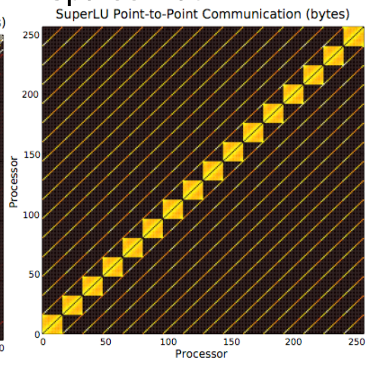
18

Dwarf Communication Patterns

- Dense Matrix



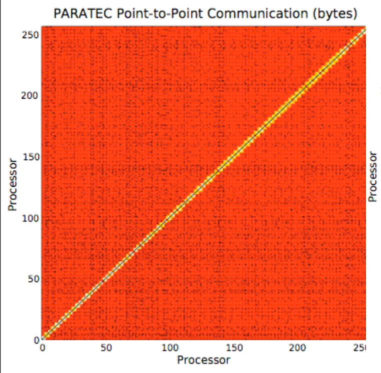
- Sparse Matrix



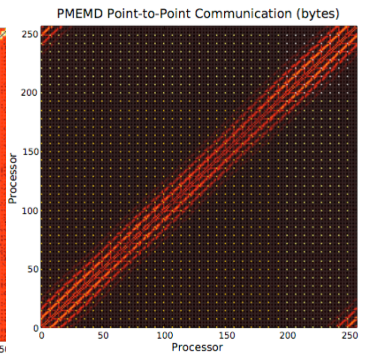
19

Dwarf Communication Patterns

- Spectral (e.g., FFT)



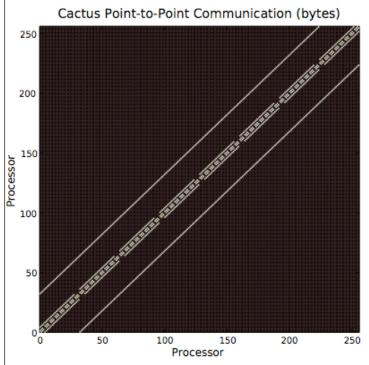
- N-Body Methods

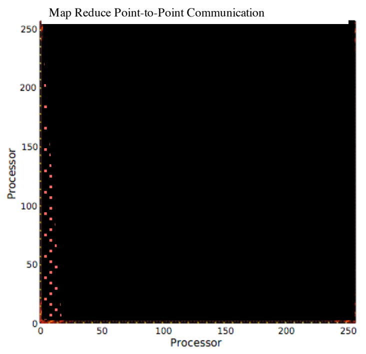


20

Dwarf Communication Patterns

- Structured Grid
- MapReduce



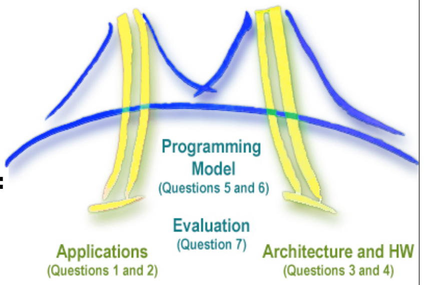


How to Connect Processors?

- Use 14 Dwarfs to gain insight into Networks On a Chip
  - Sparse connectivity for dwarfs; crossbar is overkill
  - No single best topology
- A Bandwidth-oriented network for data
  - Most point-to-point message are large and BW bound
- A separate Latency-oriented network dedicated to collectives
  - Given BW improves > (latency improvement)^2
  - E.g., Thinking Machines CM-5, Cray T3D, IBM BlueGene/L, IBM BlueGene/P

7 Questions for Parallelism

- **Applications:**
1. What are the apps?
2. What are kernels of apps?
- **Hardware:**
3. What are the HW building blocks?
4. How to connect them?
- **Programming Models:**
5. How to describe apps and kernels?
6. How to program the HW?
- **Evaluation:**
7. How to measure success?



(Inspired by a view of the Golden Gate Bridge from Berkeley)

Programming Model Considerations

- Must tradeoff Opacity vs. Visibility for *productivity* vs. *implementation efficiency*
  - Abstract underlying architecture vs. making key architecture elements of visible to programmer
- Programming model (Explicit or implicit)
  1. Identification of computational tasks
  2. Mapping computational tasks to processing elements
  3. Distribution of data to memory elements
  4. Mapping communication to inter-connection network
  5. Inter-task synchronization

Parallel Programming models: Explicit vs. Implicit

Model	***Domain***	***Task ID***	***Task Map***	***Data Distr***	***Comm Map***	***Synch***
TejaNP	Network	Exp	Exp	Exp	Exp	Exp
YAPI	DSP	Exp	Exp	Exp	Exp	Imp
MPI	HPC	Exp	Exp	Exp	Imp	Imp
Pthreads	General	Exp	Exp	Imp	Imp	Exp
Map Reduce	Data sets	Exp	Imp	Imp	Imp	Exp
OpenMP	HPC	Imp*	Imp	Imp	Imp	Imp*
HPF	HPC	Imp	Imp	Imp*	Imp	Imp

*With Directives

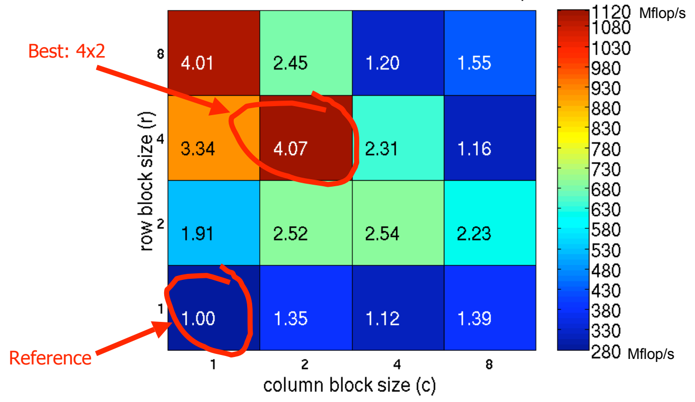
21st Century Code Generation

- Takes a decade for compiler innovations to show up in production compilers?
- New approach: "Auto-tuners" 1st run variations of program on computer to find best combinations of optimizations (blocking, padding, …) and algorithms, then produce C code to be compiled for *that* computer
  - E.g., PHiPAC (BLAS), Atlas (BLAS), Sparsity (Sparse linear algebra), Spiral (DSP), FFT-W
  - Can achieve 10X over conventional compiler
- One Auto-tuner per kernel or dwarf?
  - Exist for Dense Linear Algebra, Sparse Linear Algebra, Spectral

Sparse Matrix – Search for Blocking

For finite element problem (BCSR) [Im, Yelick, Vuduc, 2005]

900 MHz Itanium 2, Intel C v8: ref=275 Mflop/s

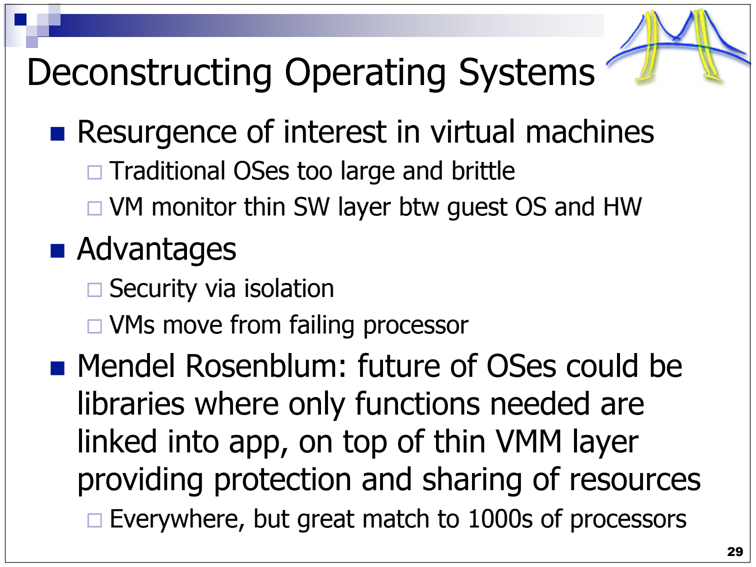


Best Sparse Blocking for 8 Computers

row block size (r)	1	2	4	8
8		Intel Pentium M		Sun Ultra 2, Sun Ultra 3, AMD Opteron
4	IBM Power 4, Intel/HP Itanium	Intel/HP Itanium 2	IBM Power 3	
2				
1				

column block size (c)

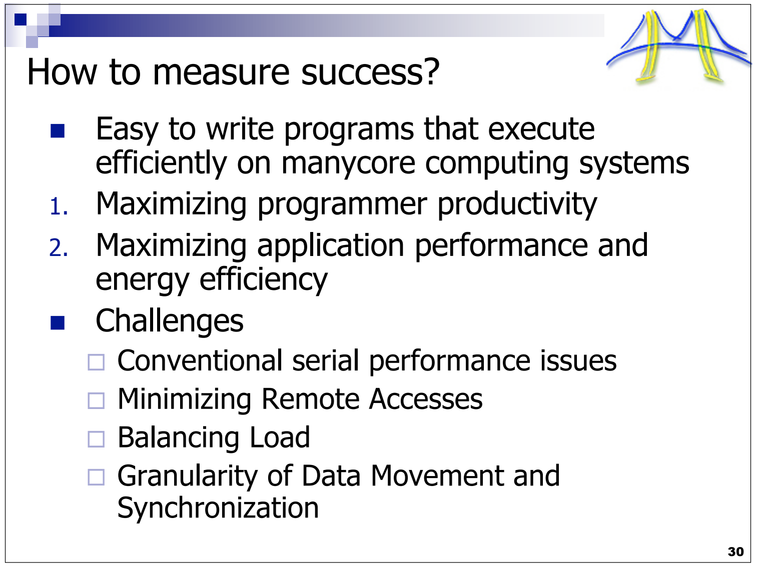
- **All possible column block sizes selected for 8 computers; How could compiler know?**



Deconstructing Operating Systems

- Resurgence of interest in virtual machines
  - Traditional OSes too large and brittle
  - VM monitor thin SW layer btw guest OS and HW
- Advantages
  - Security via isolation
  - VMs move from failing processor
- Mendel Rosenblum: future of OSes could be libraries where only functions needed are linked into app, on top of thin VMM layer providing protection and sharing of resources
  - Everywhere, but great match to 1000s of processors

29



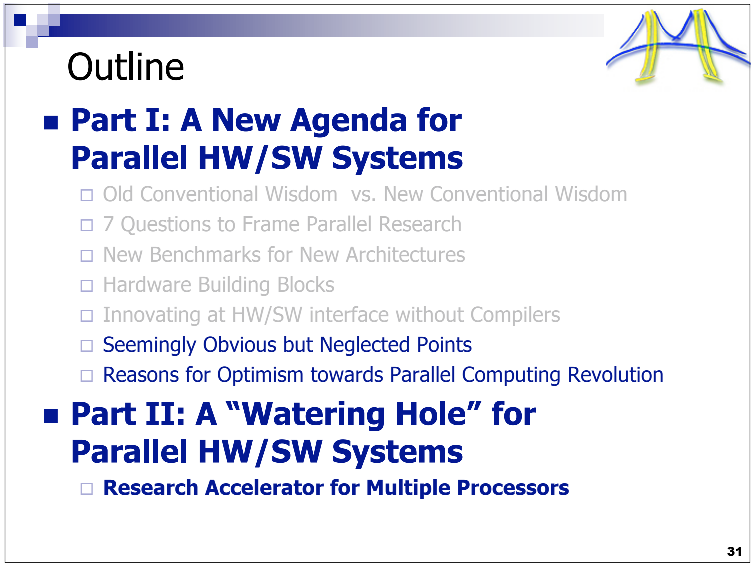
How to measure success?

- Easy to write programs that execute efficiently on manycore computing systems

1. Maximizing programmer productivity
2. Maximizing application performance and energy efficiency

- Challenges
  - Conventional serial performance issues
  - Minimizing Remote Accesses
  - Balancing Load
  - Granularity of Data Movement and Synchronization

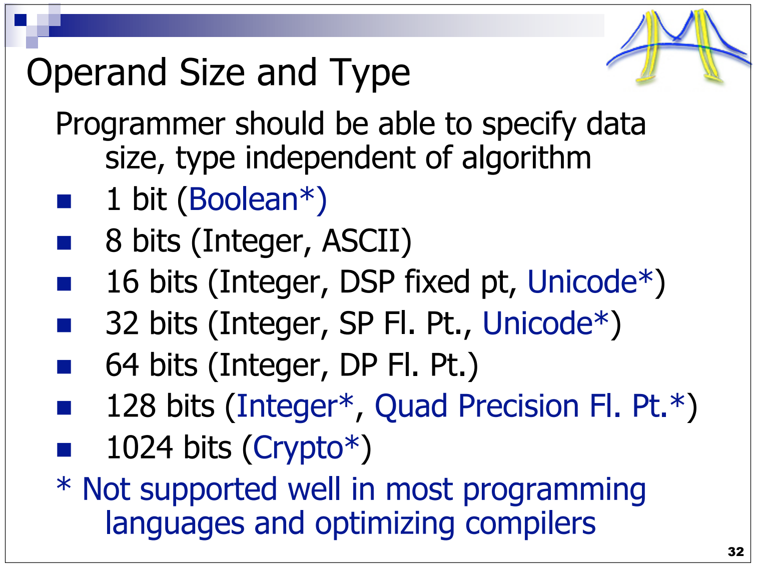
30



Outline

- **Part I: A New Agenda for Parallel HW/SW Systems**
  - Old Conventional Wisdom vs. New Conventional Wisdom
  - 7 Questions to Frame Parallel Research
  - New Benchmarks for New Architectures
  - Hardware Building Blocks
  - Innovating at HW/SW interface without Compilers
  - Seemingly Obvious but Neglected Points
  - Reasons for Optimism towards Parallel Computing Revolution
- **Part II: A "Watering Hole" for Parallel HW/SW Systems**
  - **Research Accelerator for Multiple Processors**

31



Operand Size and Type

Programmer should be able to specify data size, type independent of algorithm

- 1 bit (Boolean*)
- 8 bits (Integer, ASCII)
- 16 bits (Integer, DSP fixed pt, Unicode*)
- 32 bits (Integer, SP Fl. Pt., Unicode*)
- 64 bits (Integer, DP Fl. Pt.)
- 128 bits (Integer*, Quad Precision Fl. Pt.*)
- 1024 bits (Crypto*)

* Not supported well in most programming languages and optimizing compilers

32

Support Successful Styles of Parallelism

- Old CW: PLs, compilers, and architectures placed bets on one style of parallel programming, forcing programmers to express all parallelism in that style
- In addition to trying novel proposals (e.g., Transactional Memory, Data Flow, …), be to support those proven to work!

1. Independent task parallelism e.g., Cluster
2. Word-level parallelism e.g., Vector
3. Bit-Level parallelism e.g., "MMX SIMD"

33

Fast programs oblivious to # CPUs

- MPI forces awareness of exact mapping of computational tasks to processors
  - SPMD know N processors and which processor is which
- So far, languages oblivious to number of processors have unclear performance benefits
  - E.g., Fortress? Chapel? X10?

34

Accurate Performance-Power Counters

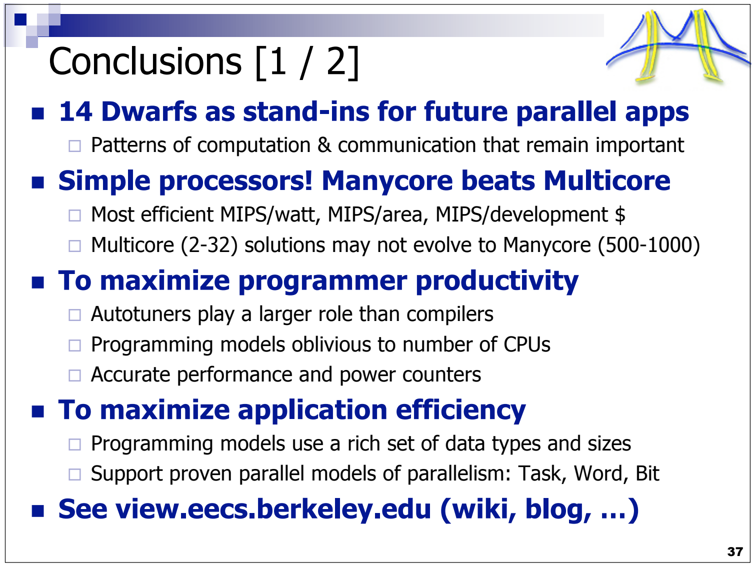
- If don't care about performance, why parallel?
- HW must have accurate, well-defined, standard, programmer-accessible counters of all the events that affect parallel performance
  - Traditionally lowest on HW designer totem pole
  - Can't measure ⇒ underutilize parallel processors
- Power/Energy is limit, so measure it
  - Need energy usage, (peak) power, and (peak) temperature since last reading
  - Per major unit: processor, I/O, …

35

Reasons for Optimism towards Parallel Revolution this time

- No looming fast sequential juggernaut to kill it
  - End of La-Z-Boy Programming Era
- Whole Industry fully committed to parallelism
- Moore's Law continues, so soon can put 1000s of simple cores on an economical chip
- Communication between cores within a chip at very low latency and very high bandwidth
  - Processor-to-Processor fast even if Processor-to-Memory slow
- Open Source Software movement means that SW stack can evolve much more quickly than in the past

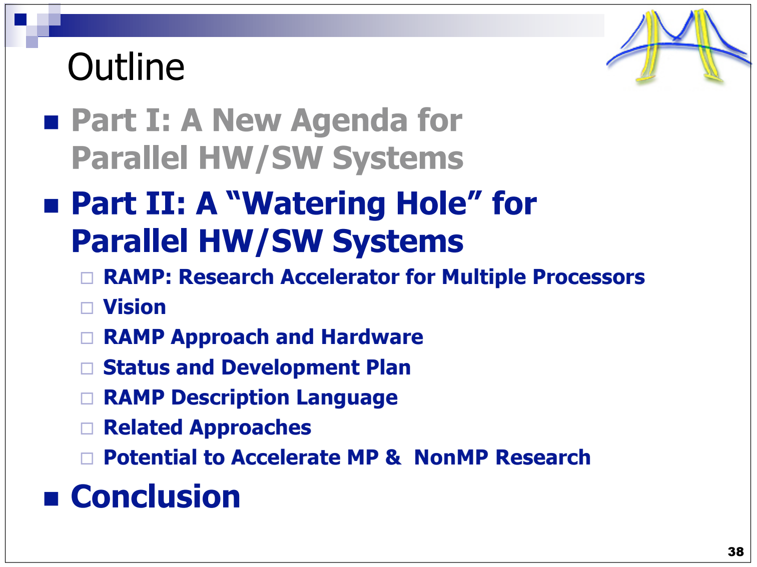
36



Conclusions [1 / 2]

- **14 Dwarfs as stand-ins for future parallel apps**
  - Patterns of computation & communication that remain important
- **Simple processors! Manycore beats Multicore**
  - Most efficient MIPS/watt, MIPS/area, MIPS/development $
  - Multicore (2-32) solutions may not evolve to Manycore (500-1000)
- **To maximize programmer productivity**
  - Autotuners play a larger role than compilers
  - Programming models oblivious to number of CPUs
  - Accurate performance and power counters
- **To maximize application efficiency**
  - Programming models use a rich set of data types and sizes
  - Support proven parallel models of parallelism: Task, Word, Bit
- **See view.eecs.berkeley.edu (wiki, blog, …)**

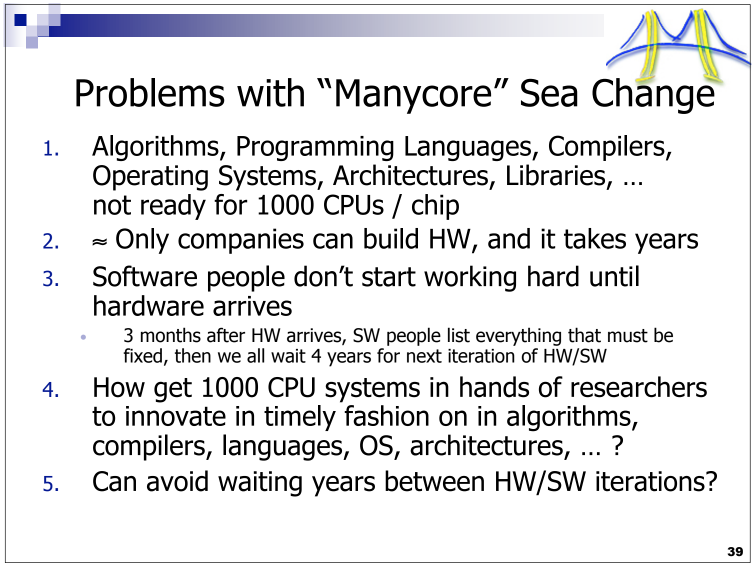
37



Outline

- **Part I: A New Agenda for Parallel HW/SW Systems**
- **Part II: A "Watering Hole" for Parallel HW/SW Systems**
  - **RAMP: Research Accelerator for Multiple Processors**
  - **Vision**
  - **RAMP Approach and Hardware**
  - **Status and Development Plan**
  - **RAMP Description Language**
  - **Related Approaches**
  - **Potential to Accelerate MP & NonMP Research**
- **Conclusion**

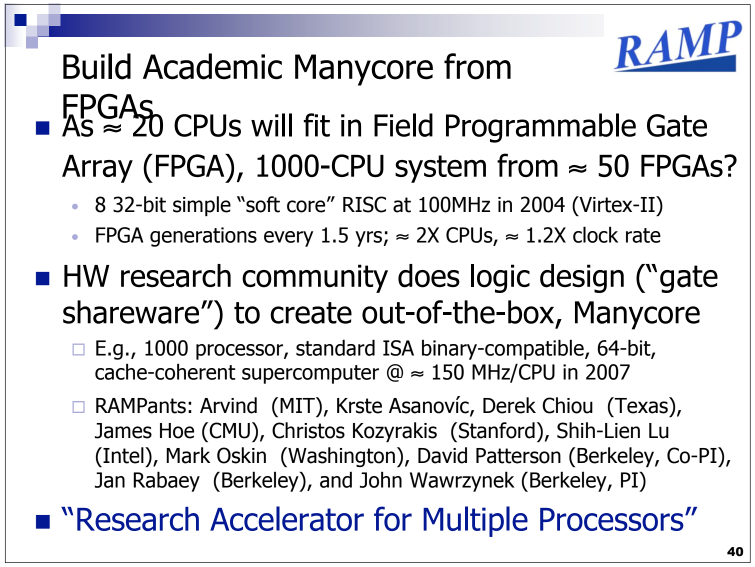
38



Problems with "Manycore" Sea Change

1. Algorithms, Programming Languages, Compilers, Operating Systems, Architectures, Libraries, … not ready for 1000 CPUs / chip
2. ≈ Only companies can build HW, and it takes years
3. Software people don't start working hard until hardware arrives
   - 3 months after HW arrives, SW people list everything that must be fixed, then we all wait 4 years for next iteration of HW/SW
4. How get 1000 CPU systems in hands of researchers to innovate in timely fashion on in algorithms, compilers, languages, OS, architectures, … ?
5. Can avoid waiting years between HW/SW iterations?

39



Build Academic Manycore from FPGAs

RAMP

- As ≈ 20 CPUs will fit in Field Programmable Gate Array (FPGA), 1000-CPU system from ≈ 50 FPGAs?
  - 8 32-bit simple "soft core" RISC at 100MHz in 2004 (Virtex-II)
  - FPGA generations every 1.5 yrs; ≈ 2X CPUs, ≈ 1.2X clock rate
- HW research community does logic design ("gate shareware") to create out-of-the-box, Manycore
  - E.g., 1000 processor, standard ISA binary-compatible, 64-bit, cache-coherent supercomputer @ ≈ 150 MHz/CPU in 2007
  - RAMPants: Arvind (MIT), Krste Asanovíc, Derek Chiou (Texas), James Hoe (CMU), Christos Kozyrakis (Stanford), Shih-Lien Lu (Intel), Mark Oskin (Washington), David Patterson (Berkeley, Co-PI), Jan Rabaey (Berkeley), and John Wawrzynek (Berkeley, PI)
- "Research Accelerator for Multiple Processors"

40

Characteristics of Ideal Academic CS Research Parallel Processor?

- Scales – Hard problems at 1000 CPUs
- Cheap to buy – Limited academic research $
- Cheap to operate, Small, Low Power – $ again
- Community – Share SW, training, ideas, …
- Simplifies debugging – High SW churn rate
- Reconfigurable – Test many parameters, different ISAs, different organizations, …
- Credible – Results translate to real computers
- Performance – Fast enough to run real OS and full apps, get results overnight

41

Why Good for Research Manycore?

	SMP	Cluster	Simulate	RAMP
Scalability (1k CPUs)	C	A	A	A
Cost (1k CPUs)	F ($40M)	C ($2-3M)	A+ ($0M)	A ($0.1-0.2M)
Cost of ownership	A	D	A	A
Power/Space (kilowatts, racks)	D (120 kw, 12 racks)	D (120 kw, 12 racks)	A+ (.1 kw, 0.1 racks)	A (1.5 kw, 0.3 racks)
Community	D	A	A	A
Observability	D	C	A+	A+
Reproducibility	B	D	A+	A+
Reconfigurability	D	C	A+	A+
Credibility	A+	A+	F	B+/A-
Perform. (clock)	A (2 GHz)	A (3 GHz)	F (0 GHz)	C (0.1 GHz)
GPA	C	B-	B	A-

Why RAMP More Credible?

- Starting point for processor is debugged design from Industry in HDL
- Fast enough that can run more software, more experiments than simulators
- Design flow, CAD similar to real hardware
  - Logic synthesis, place and route, timing analysis
- HDL units implement operation vs. a high-level description of function
  - Model queuing delays at buffers by building real buffers
- Must work well enough to run OS
  - Can't go backwards in time, which simulators can unintentionally
- Can measure anything as sanity checks

43

Can RAMP keep up?

- FGPA generations: 2X CPUs / 18 months
  - 2X CPUs / 24 months for desktop microprocessors
- 1.1X to 1.3X performance / 18 months
  - 1.2X? / year per CPU on desktop?
- However, goal for RAMP is accurate system emulation, not to be the real system
  - Goal is accurate target performance, parameterized reconfiguration, extensive monitoring, reproducibility, cheap (like a simulator) while being credible and fast enough to emulate 1000s of OS and apps in parallel (like a hardware prototype)
  - OK if ≈30X slower than real 1000 processor hardware, provided >1000X faster than simulator of 1000 CPUs

44

Example: Vary memory latency, BW

- Target system: TPC-C, Oracle, Linux on 1024 CPUs @ 2 GHz, 64 KB L1 I$ & D$/CPU, 16 CPUs share 0.5 MB L2$, shared 128 MB L3$
  - Latency: L1 1 - 2 cycles, L2 8 - 12 cycles, L3 20 - 30 cycles, DRAM 200 – 400 cycles
  - Bandwidth: L1 8 - 16 GB/s, L2 16 - 32 GB/s, L3 32 – 64 GB/s, DRAM 16 – 24 GB/s per port, 16 – 32 DDR3 128b memory ports
- Host system: TPC-C, Oracle, Linux on 1024 CPUs @ 0.1 GHz, 32 KB L1 I$, 16 KB D$
  - Latency: L1 1 cycle, DRAM 2 cycles
  - Bandwidth: L1 0.1 GB/s, DRAM 3 GB/s per port, 128 64b DDR2 ports
  - Use cache models and DRAM to emulate L1$, L2$, L3$ behavior

45

Accurate Clock Cycle Accounting

- Key to RAMP success is cycle-accurate emulation of parameterized target design
  - As vary number of CPUs, CPU clock rate, cache size and organization, memory latency & BW, interconnect latency & BW, disk latency & BW, Network Interface Card latency & BW, …
  - Least common divisor time unit to drive emulation?

1. For research results to be credible
2. To run standard, shrink-wrapped OS, DB,…
   - Otherwise fake interrupt times since devices relatively too fast

⇒ Good target clock cycle accounting is high priority for RAMP project

46

Why 1000 Processors?

- Eventually can build 1000 processors per chip
- Experience of high performance community on stress of level of parallelism on architectures and algorithms
  - 32-way: anything goes
  - 100-way: good architecture and bad algorithms or bad architecture and good algorithms
  - 1000-way: good architecture and good algorithms
- Must solve hard problems to scale to 1000
- Future is promising if can scale to 1000

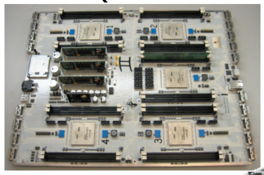
47

RAMP 1 Hardware

- Completed Dec. 2004 (14x17 inch 22-layer PCB)

Board:

5 Virtex II FPGAs, 18 banks DDR2-400 memory, 20 10GigE conn.



1.5W / computer, 5 cu. in. /computer, $100 / computer

Box:

10 compute modules in 8U rack mount chassis

1000 CPUs : ≈1.5 KW, ≈ ¼ rack, ≈ $100,000

BEE2: Berkeley Emulation Engine 2
By John Wawrzynek and Bob Brodersen with students Chen Chang and Pierre Droz

48

RAMP Storage

- RAMP can emulate disks as well as CPUs
  - Inspired by Xen, VMware Virtual Disk models
  - Have parameters to act like real disks
  - Can emulate performance, but need storage capacity
- Low cost Network Attached Storage to hold emulated disk content
  - Use file system on NAS box
  - E.g., Sun Fire X4500 Server ("Thumper")
    48 SATA disk drives,
    24TB of storage @ <$2k/TB

4 Rack Units High

the stone soup of architecture research platforms

Wawrzynek
Hardware

Chiou
Glue-support

Patterson
I/O

Hoe
Coherence

Kozyrakis
Monitoring

Asanovic
Cache

Oskin
Net Switch

Arvind
PPC

Lu
x86

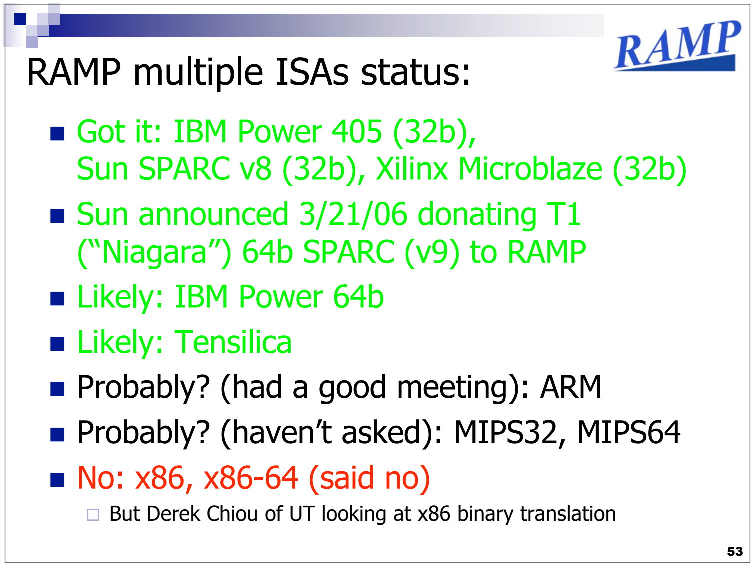
Quick Sanity Check

- BEE2 4 banks DDR2-400 per FPGA
- Memory BW/FPGA = 4 * 400 * 8B = 12,800 MB/s
- 16 32-bit Microblazes per Virtex II FPGA (last generation)
  - Assume 150 MHz, CPI is 1.5 (4-stage pipeline), 33% Load/Stores
  - BW need/CPU = 150/1.5 * (1+ 0.33) * 4B ≈ 530 MB/sec
- BW need/FPGA ≈ 16 * 530 ≈ 8500 MB/s
  - 2/3 Peak Memory BW / FPGA
- Suppose add caches (.75MB ⇒ 32KI$, 16D$/CPU)
  - SPECint2000 I$ Miss 0.5%, D$ Miss 2.8%, 33% Load/stores, 64B blocks*
  - BW/CPU = 150/1.5*(0.5% + 33%*2.8%)*64 ≈ 100 MB/s
- BW/FPGA with caches ≈ 16 * 100 MB/s ≈ 1600 MB/s
  - 1/8 Peak Memory BW/FPGA; plenty BW available for tracing, …
- Example of optimization to improve emulation

* Cantin and Hill, "Cache Performance for SPEC CPU2000 Benchmarks"

RAMP Philosophy

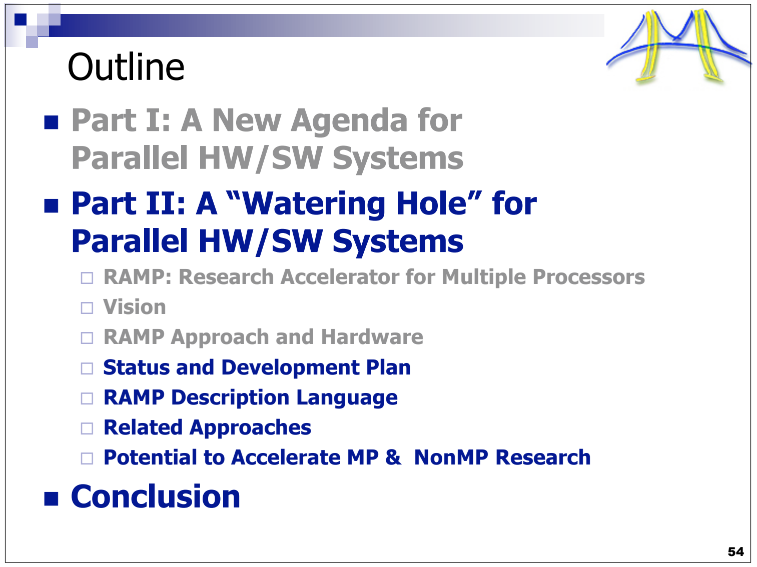
- Build vanilla out-of-the-box examples to attract software community
  - Multiple industrial ISAs, real industrial operating systems, 1000 processors, accurate clock cycle accounting, reproducible, traceable, parameterizable, cheap to buy and operate, …
- But RAMPants have grander plans (will share)
  - Data flow computer ("Wavescalar") – Oskin @ U. Washington
  - 1,000,000-way MP ("Transactors") – Asanovic @ MIT
  - Distributed Data Centers ("RAD Lab") – Patterson @ Berkeley
  - Transactional Memory ("TCC") – Kozyrakis @ Stanford
  - Reliable Multiprocessors ("PROTOFLEX") – Hoe @ CMU
  - X86 emulation ("UT FAST") – Chiou @ Texas
  - Signal Processing in FPGAs ("BEE2") – Wawrzynek @ Berkeley



RAMP multiple ISAs status:

- Got it: IBM Power 405 (32b), Sun SPARC v8 (32b), Xilinx Microblaze (32b)
- Sun announced 3/21/06 donating T1 (“Niagara”) 64b SPARC (v9) to RAMP
- Likely: IBM Power 64b
- Likely: Tensilica
- Probably? (had a good meeting): ARM
- Probably? (haven’t asked): MIPS32, MIPS64
- No: x86, x86-64 (said no)
  - But Derek Chiou of UT looking at x86 binary translation

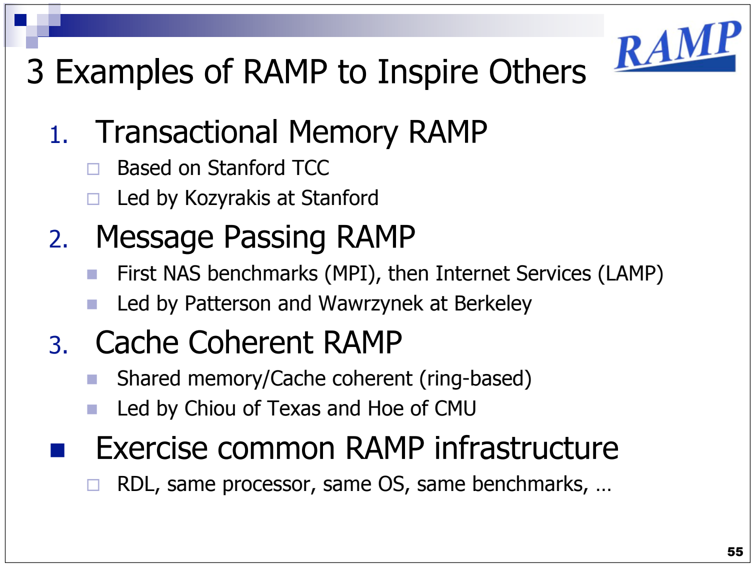
53



Outline

- **Part I: A New Agenda for Parallel HW/SW Systems**
- **Part II: A “Watering Hole” for Parallel HW/SW Systems**
  - **RAMP: Research Accelerator for Multiple Processors**
  - **Vision**
  - **RAMP Approach and Hardware**
  - **Status and Development Plan**
  - **RAMP Description Language**
  - **Related Approaches**
  - **Potential to Accelerate MP & NonMP Research**
- **Conclusion**

54

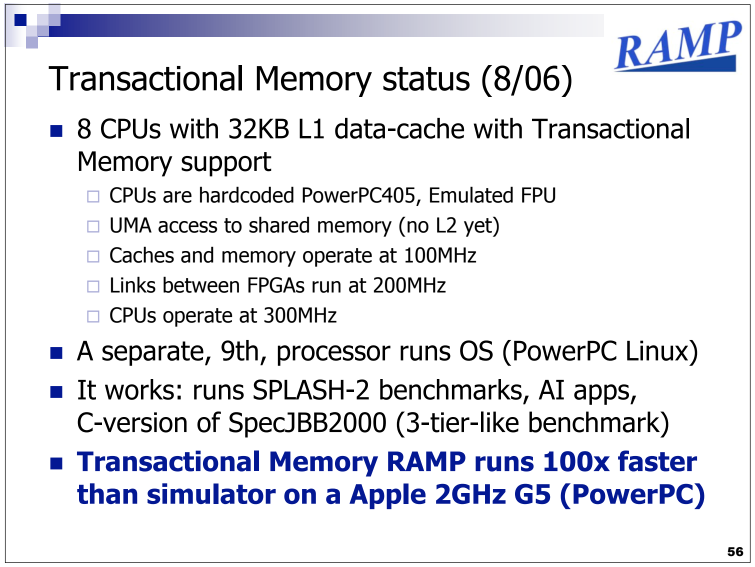


3 Examples of RAMP to Inspire Others

1. Transactional Memory RAMP
   - Based on Stanford TCC
   - Led by Kozyrakis at Stanford
2. Message Passing RAMP
   - First NAS benchmarks (MPI), then Internet Services (LAMP)
   - Led by Patterson and Wawrzynek at Berkeley
3. Cache Coherent RAMP
   - Shared memory/Cache coherent (ring-based)
   - Led by Chiou of Texas and Hoe of CMU

- Exercise common RAMP infrastructure
  - RDL, same processor, same OS, same benchmarks, …

55



Transactional Memory status (8/06)

- 8 CPUs with 32KB L1 data-cache with Transactional Memory support
  - CPUs are hardcoded PowerPC405, Emulated FPU
  - UMA access to shared memory (no L2 yet)
  - Caches and memory operate at 100MHz
  - Links between FPGAs run at 200MHz
  - CPUs operate at 300MHz
- A separate, 9th, processor runs OS (PowerPC Linux)
- It works: runs SPLASH-2 benchmarks, AI apps, C-version of SpecJBB2000 (3-tier-like benchmark)
- **Transactional Memory RAMP runs 100x faster than simulator on a Apple 2GHz G5 (PowerPC)**

56

RAMP Blue Prototype (11/06)

- 8 MicroBlaze cores / FPGA + FPU
- 8 BEE2 modules (32 “user” FPGAs) x 4 FPGAs/module = 256 cores @ 100MHz
- Full star-connection between modules
- 11/1/06 Run 1 NAS benchmark in UPC on 32 nodes (1 board)
- CPUs are softcore MicroBlazes (32-bit Xilinx RISC architecture)
- Also 32-bit SPARC (LEON3) running full Linux OS, large SW stack

RAMP Milestones

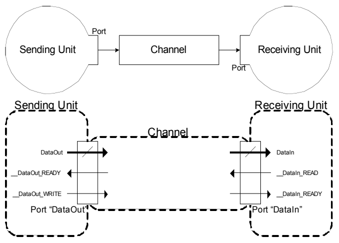
- September 2006 : Picked 32-bit SPARC (Leon) 1st ISA
  - Verification suite, Running full Linux, Size of design (LUTs / BRAMs)
  - Executes comm. app binaries, Configurability, Friendly licensing
- January 2007 milestones for all 3 RAMP examples
  - Run on Xilinx Virtex 2 XUP board
  - Run on 8 RAMP 1 (BEE2) boards
  - 64 to 128 processors
- June 2007 milestones for all 3 RAMPs
  - Accurate clock cycle accounting, I/O model
  - Run on 16 RAMP 1 (BEE2) boards and Virtex 5 XUP boards
  - 128 to 256 processors
- 2H07: RAMP 2.0 boards on Virtex 5
  - 3rd party sells board, download software and gateware from website on RAMP 2.0 or Xilinx V5 XUP boards

RAMP Project Status

- NSF infrastructure grant awarded 3/06
  - 2 staff positions (NSF sponsored), no grad students
- IBM Faculty Awards to RAMPants 6/06
  - Krste Asanovic (MIT), Derek Chiou (Texas), James Hoe (CMU), Christos Kozyrakis (Stanford), John Wawrzynek (Berkeley)
- 3-day retreats with industry visitors
  - “Berkeley-style” retreats 1/06 (Berkeley), 6/06 (ISCA/Boston), 1/07 (Berkeley), 6/07 (ISCA/San Diego)
- RAMP 1/RDL short course
  - 40 people from 6 schools 1/06

RAMP Description Language (RDL)

- RDL describes plumbing connecting units together ≈ “HW Scripting Language/Linker”
- Design composed of **units** that send messages over **channels** via ports
- Units (10,000 + gates)
  - CPU + L1 cache, DRAM controller…
- Channels (≈ FIFO)
  - Lossless, point-to-point, unidirectional, in-order delivery…
- Generates HDL to connect units



RDL at technological sweet spot

RAMP

- Matches current chip design style
  - Locally synchronous, globally asynchronous
- To plug unit (in any HDL) into RAMP infrastructure, just add RDL "wrapper"
- Units can also be in C or Java or System C or …
  ⇒ Allows debugging design at high level
- Compiles target interconnect onto RAMP paths
  - Handles housekeeping of data width, number of transfers
- FIFO communication model
  ⇒ Computer can have deterministic behavior
  - Interrupts, memory accesses, … exactly same clock cycle each run

  ⇒ Easier to debug parallel software on RAMP

RDL Developed by Krste Asanovíc and Greg Giebling

Related Approaches

RAMP

- Quickturn, Axis, IKOS, Thara:
  - FPGA- or special-processor based gate-level hardware emulators
  - HDL mapped to array for cycle and bit-accurate netlist emulation
  - No DRAM memory since modeling CPU, not system
  - Doesn't worry about speed of logic synthesis: 1 MHz clock
  - Uses small FPGAs since takes many chips/CPU, and pin-limited
  - Expensive: $5M
- RAMP's emphasis is on emulating high-level system behaviors
  - More DRAMs than FPGAs: BEE2 has 5 FPGAs, 96 DRAM chips
  - Clock rate affects emulation time: >100 MHz clock
  - Uses biggest FGPAs, since many CPUs/chip
  - Affordable: $0.1 M

RAMP's Potential Beyond Manycore

RAMP

- **Attractive Experimental Systems Platform: Standard ISA + standard OS + modifiable + fast enough + trace/measure anything**
  - Generate long traces of full stack: App, VM, OS, …
  - Test hardware security enhancements in the wild
  - Inserting faults to test availability schemes
  - Test design of switches and routers
  - SW Libraries for 128-bit floating point
  - App-specific instruction extensions (≈Tensilica)
  - Alternative Data Center designs
    - Akamai vs. Google: N centers of M computers

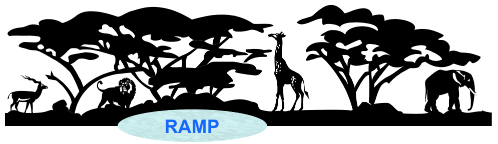
Potential to Accelerate Manycore

RAMP

- **With RAMP: Fast, wide-ranging exploration of HW/SW options + head-to-head competitions to determine winners and losers**
  - Common artifact for HW and SW researchers ⇒ innovate across HW/SW boundaries
  - Minutes vs. years between "HW generations"
  - Cheap, small, low power ⇒ Every dept owns one
  - FTP supercomputer overnight, check claims locally
  - Emulate any Manycore ⇒ aid to teaching parallelism
  - If HP, IBM, Intel, M/S, Sun, …had RAMP boxes
    ⇒ Easier to carefully evaluate research claims
    ⇒ Help technology transfer
- **Without RAMP: One Best Shot + Field of Dreams?**

Multiprocessing Watering Hole

RAMP



Parallel file system Dataflow language/computer Data center in a box
Fault insertion to check dependability Router design Compile to FPGA
Flight Data Recorder Security enhancements Transactional Memory
Internet in a box 128-bit Floating Point Libraries Parallel languages

- Killer app: ≈ All CS Research, Advanced Development
- RAMP attracts many communities to shared artifact
  ⇒ Cross-disciplinary interactions
  ⇒ Ramp up innovation in multiprocessing
- RAMP as next Standard Research/AD Platform? (e.g., VAX/BSD Unix in 1980s)

65

RAMP Supporters:

RAMP

- Gordon Bell (Microsoft)
- Ivo Bolsens (Xilinx CTO)
- Jan Gray (Microsoft)
- Norm Jouppi (HP Labs)
- Bill Kramer (NERSC/LBL)
- Konrad Lai (Intel)
- Craig Mundie (MS CTO)
- Jaime Moreno (IBM)
- G. Papadopoulos (Sun CTO)
- Jim Peek (Sun)
- Justin Rattner (Intel CTO)
- Michael Rosenfield (IBM)
- Tanaz Sowdagar (IBM)
- Ivan Sutherland (Sun Fellow)
- Chuck Thacker (Microsoft)
- Kees Vissers (Xilinx)
- Jeff Welser (IBM)
- David Yen (Sun EVP)
- *Doug Burger (Texas)*
- *Bill Dally (Stanford)*
- *Susan Eggers (Washington)*
- *Kathy Yelick (Berkeley)*

RAMP Participants: Arvind (MIT), Krste Asanovíc (MIT), Derek Chiou (Texas), James Hoe (CMU), Christos Kozyrakis (Stanford), Shih-Lien Lu (Intel), Mark Oskin (Washington), David Patterson (Berkeley, Co-PI), Jan Rabaey (Berkeley), and John Wawrzynek (Berkeley, PI)

66

Conclusions [2 / 2]

RAMP

- **Carpe Diem: need RAMP yesterday**
  - FPGAs ready now, and getting better
  - Stand on shoulders vs. toes: standardize on BEE2
  - Architects aid colleagues via gateware
- **RAMP accelerates HW/SW generations**
  - System emulation + good accounting vs. FPGA computer
  - Emulate, Trace, Reproduce anything; Tape out every day
  - RAMP + Auto-tuner ⇒ search HW ***and*** algorithm space
- "**Multiprocessor Research Watering Hole**" ramp up research in multiprocessing via common research platform ⇒ innovate across fields ⇒ hasten sea change from sequential to parallel computing
- **See ramp.eecs.berkeley.edu**

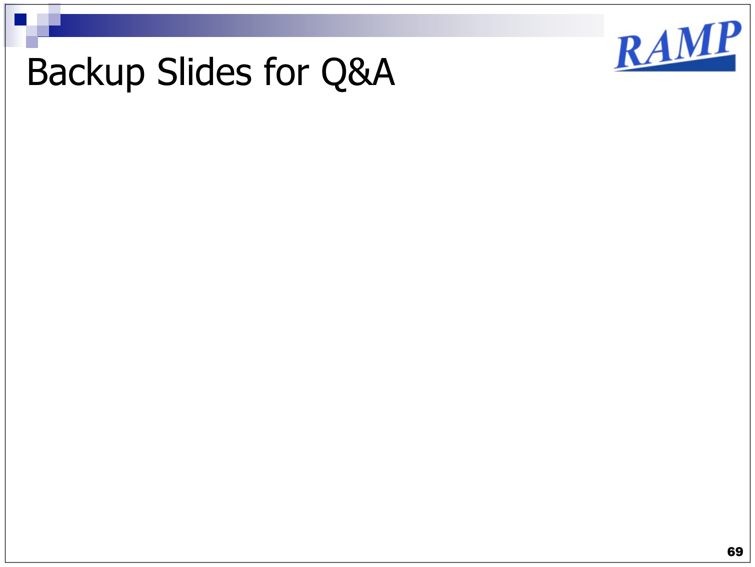
67

Acknowledgments

RAMP

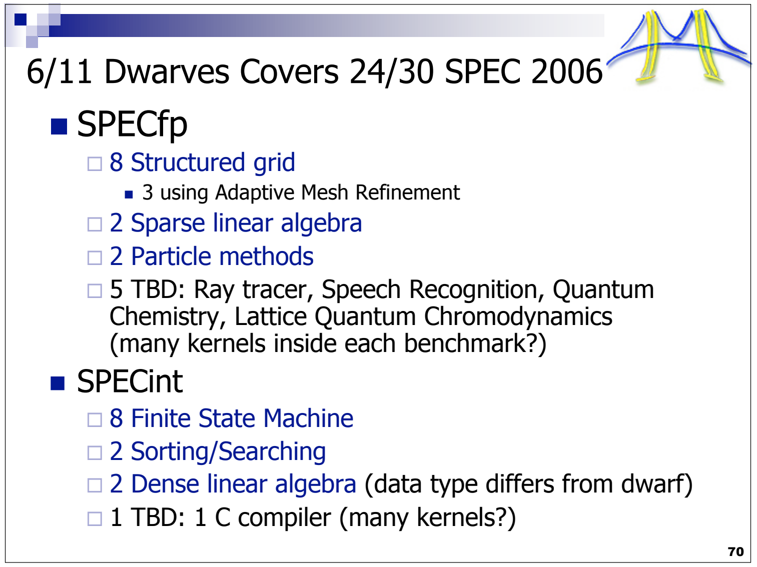
- Berkeley View material comes from discussions with:
  - Professors Krste Asanovíc, Ras Bodik, Jim Demmel, Kurt Keutzer, John Wawrzynek, and Kathy Yelick
  - UCB Grad students Bryan Catanzaro, Joe Gebis, William Plishker, and Sam Williams
  - LBNL:Parry Husbands, Bill Kramer, Lenny Oliker, John Shalf
- **See view.eecs.berkeley.edu**
- RAMP based on work of RAMP Developers:
  - Arvind (MIT), Krste Asanovíc, Derek Chiou (Texas), James Hoe (CMU), Christos Kozyrakis (Stanford), Shih-Lien Lu (Intel), Mark Oskin (Washington), David Patterson (Berkeley, Co-PI), Jan Rabaey (Berkeley), and John Wawrzynek (Berkeley, PI)
- **See ramp.eecs.berkeley.edu**

68



Backup Slides for Q&A

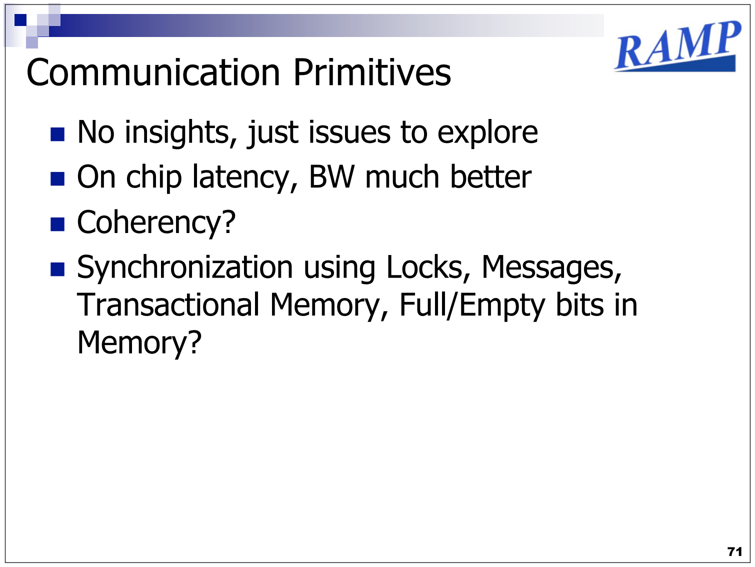
69



6/11 Dwarves Covers 24/30 SPEC 2006

- SPECfp
  - 8 Structured grid
    - 3 using Adaptive Mesh Refinement
  - 2 Sparse linear algebra
  - 2 Particle methods
  - 5 TBD: Ray tracer, Speech Recognition, Quantum Chemistry, Lattice Quantum Chromodynamics (many kernels inside each benchmark?)
- SPECint
  - 8 Finite State Machine
  - 2 Sorting/Searching
  - 2 Dense linear algebra (data type differs from dwarf)
  - 1 TBD: 1 C compiler (many kernels?)

70



Communication Primitives

- No insights, just issues to explore
- On chip latency, BW much better
- Coherency?
- Synchronization using Locks, Messages, Transactional Memory, Full/Empty bits in Memory?

71