

IEEE/ACM International Symposium on Microarchitecture



Lake Como, Italy
Nov. 8-12, 2008



Microarchitecture in the System-level Integration Era

Chuck Moore
AMD Senior Fellow
November 10, 2008

Useful Anecdotes – *Chuck's Top Ten List*

- 1) Never forget the role of Uniprocessor performance (John Cocke, ~1988)
- 2) Power will be the number 1 design constraint in the future (Mark Horowitz, ~2000)
- 3) It's the synchronization, stupid!
- 4) The value of a 5% optimization goes down exponentially in the final throes of design convergence (in favor of schedule)
- 5) If something seems really hard at the start of a project, it will turn out to be impossible by the end of the project
- 6) If something seems simple at the start of a project, it will turn out to be really hard by the end.
- 7) No clever architecture idea ever goes unpunished (Marty Hopkins)
- 8) Designing an x86 processor won't make your head explode (Fred Weber)
- 9) Scarcity and abundance are the key parameters of technology (George Gilder)
- 10) Moore's Law relates to the ***economics*** of the semiconductor industry just as much as it does to technology progression



What is the System-level Integration Era?

- Single-chip CPU Era: 1986 – 2004
 - Extreme focus on single-threaded performance optimizations
 - Multi-issue, out-of-order execution core plus moderate cache hierarchy
- Chip Multiprocessor (CMP) Era: 2004 – 2010
 - Early: Hasty integration of multiple cores into same chip/package
 - Mid-life: Address some of the HW scalability and interference issues
 - Current: Homogeneous CPUs plus moderate system-level functionality
- System-level Integration Era: ~2010 onward
 - Integration of substantial system-level functionality
 - Heterogeneous processors and accelerators
 - Introspective control systems for managing on-chip resources & events



Enablers of the System-level Integration Era

- Moore's Law is projected to continue well beyond 22nm
 - Traditional components get really small in 22nm
 - Opteron-class core: $\sim 5 \text{ mm}^2$; 1MB fast cache memory: $\sim 4.5 \text{ mm}^2$
 - 3D chip stacking on the near horizon
 - Initially, this will mostly involve stacking various types of RAM
 - Over time, multiple chips with logical functionality
- Large customer value potential in Platform-level optimizations
 - Cost and power reductions from integration (vs. discrete chips)
 - Performance/watt/\$\$ is the new value proposition
 - Balance on-chip processing with available system-level BW
 - Balance on-chip processing with ***actual usage scenarios***



Challenges in the System-level Integration Era

- Complexity management
 - Principles for managing exponential growth
 - Development expense and TTM
- Exploitation of available parallelism
 - Single thread performance outlook
 - Parallel threads, Throughput and Distributed computing
 - Optimized SW for System-level Solutions
- Memory system balance
- The Power Wall issues keep getting *worse*



Challenges in the System-level Integration Era

- Complexity management
 - Principles for managing exponential growth
 - Development expense and TTM
- Exploitation of available parallelism
 - Single thread performance outlook
 - Parallel threads, Throughput and Distributed computing
 - Optimized SW for System-level Solutions
- Memory system balance
- The Power Wall issues keep getting *worse*



Principles for Managing Exponential Growth

- Scarcity and Abundance: *the yin and yang of technology*
 - Leverage abundance to solve problems w/ scarcity
 - **Abundant**: *transistors, raw FLOPs, bandwidth, cores?*
 - **Scarce**: *power, latency, HW and SW productivity, TTM*
- Take advantage of Abstractions
 - Powerful and proven principle for scientific advancement
 - Thomas Kuhn (1962): *"The Structure of Scientific Revolutions"*
 - Already broadly used in the computer field
 - Hierarchical CAD; Embedded Design and Re-use; SW encapsulation
- Understand the role of "good enough" and Disruptive Technologies
 - My version: ***The low-end tends to eat the high-end***
 - Disk drives, "attack of the killer micros", ubiquitous networking
 - Clayton Christensen (1997): *"The Innovators Dilemma"*



Development Expense and Time-to-Market

- Leading edge chip design today is **very** expensive
 - Multi-year design teams of 300-500 people, and growing fast
 - Multi-million dollar mask sets for each full revision of the chip
 - Platform Qualification (HW *and* SW): Hundreds of engineers
- Complexity & Scope challenge Time-to-Market (TTM)
 - Integration adds value, but it also adds complexity
 - Design verification methods are stretched to their limits
 - Microcode-based workarounds to bugs becoming increasingly inadequate at the system level
- Average Sales Price (ASP) in many markets continues to drop
 - Very unforgiving time-to-market expectations
 - Good for the consumer, but bad for the chip developers



Implications of Expense and TTM Pressures

- Need dramatic improvements in **Design Productivity**
 - SOC-style Partitioning, Design and Integration
 - Develop modular and re-usable “IP blocks” (cores, controllers, PHYs)
 - Develop common “chip infrastructure” for families of chip variants
 - Increased use of Design Automation techniques
- Need increased focus on **Design for Verification (DFV)**
 - Invest real hardware mechanisms to expose corner cases
 - Expand bug mitigation techniques
- Need more robust **System-level Modeling & Analysis** framework
 - Discover on-chip system-level bottlenecks earlier in the design cycle
 - RAMP, COTSon (HP/AMD)
- **Flexible chip-level architecture** to cover multiple platform solutions
 - Expand reconfigurability options (#cores, cache size, I/O footprint, etc)
 - Harvest power from unused sections to augment active sections



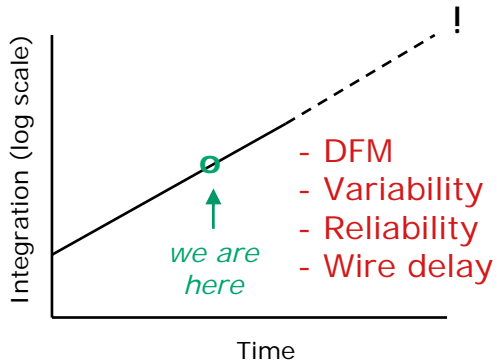
Challenges in the System-level Integration Era

- Complexity management
 - Principles for managing exponential growth
 - Development expense and TTM
- Exploitation of available parallelism
 - Single thread performance outlook
 - Parallel threads, Throughput and Distributed computing
 - Optimized SW for System-level Solutions
- Memory system balance
- The Power Wall issues keep getting *worse*

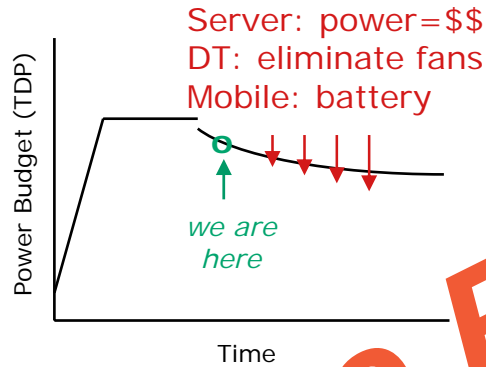


Single-thread Performance

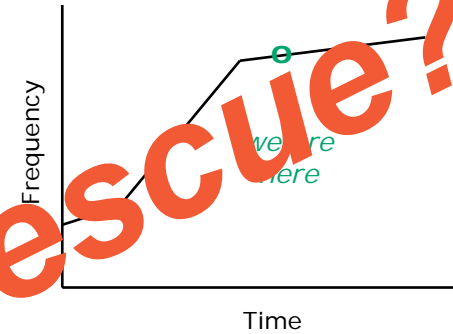
Moore's Law ☺



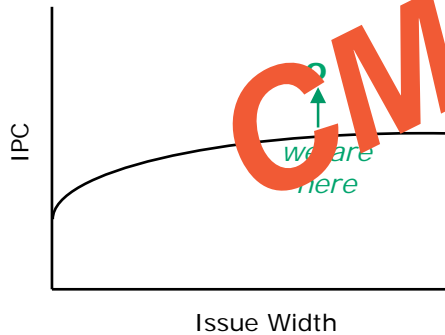
The Power Wall ☹



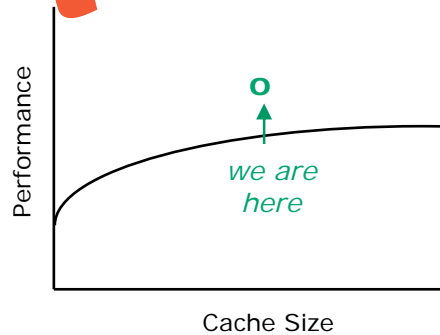
The Frequency Wall ☹



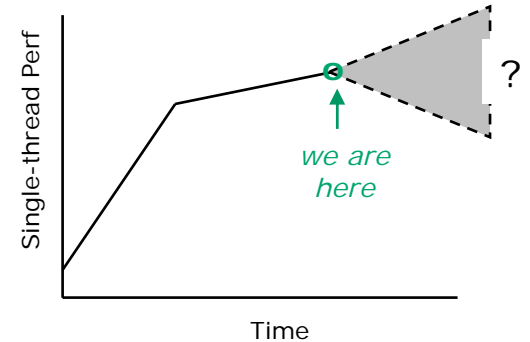
The IPC Complexity Wall ☹



Locality ☹



Single thread Perf (!)



Cooperating Subsystems must ...

- **Communicate** with one another & higher-level frameworks
 - Fork/Join semantics; Producer/Consumer protocols
 - OS scheduler, resource managers and exception handlers
- **Synchronize** the use of shared resources
 - Implicit sync through storage ordering constraints
 - Explicit sync through semaphores and locks
- Enable **Data Movement** for optimized performance
 - Coherent caches do a pretty good job with system memory
 - Still, for devices we often see “page pinning” & explicit moves

The resulting overhead adds to the serial component
of parallel programs

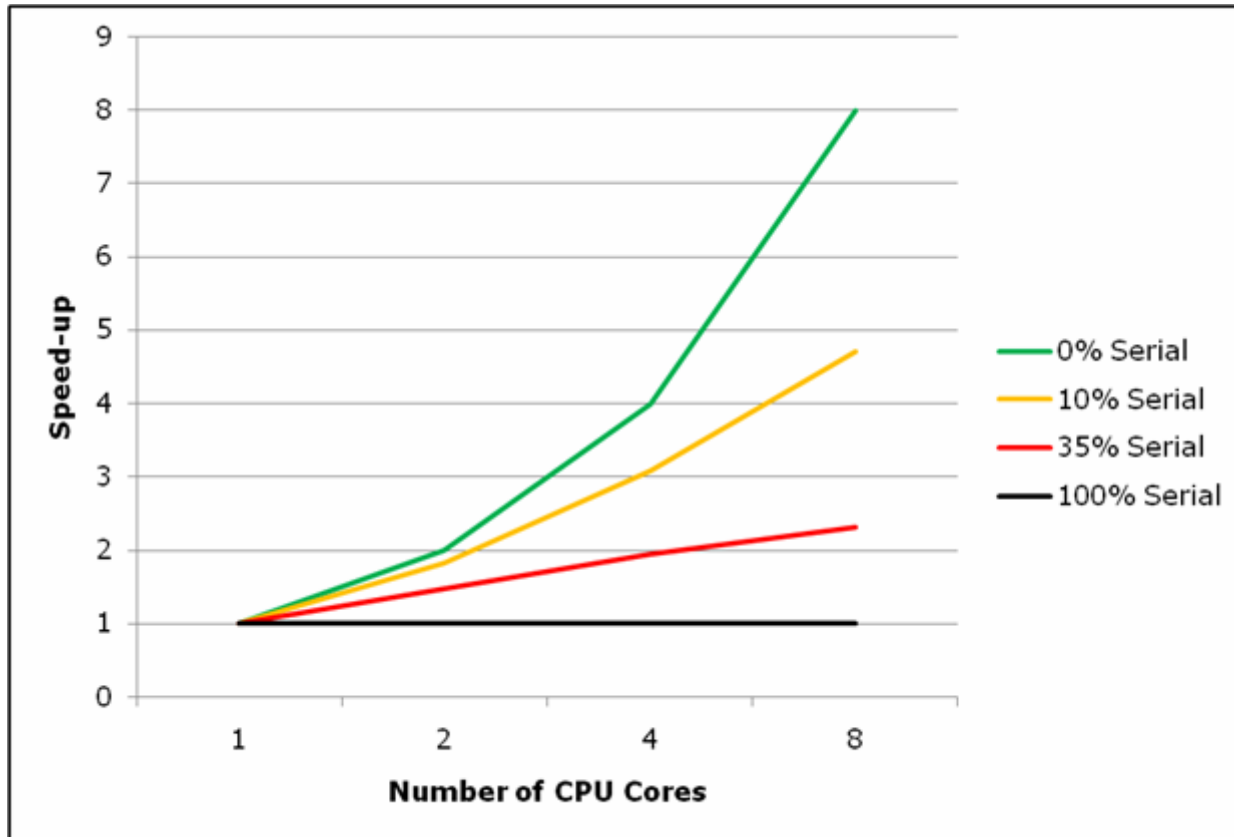


Parallel Programs and Amdahl's Law

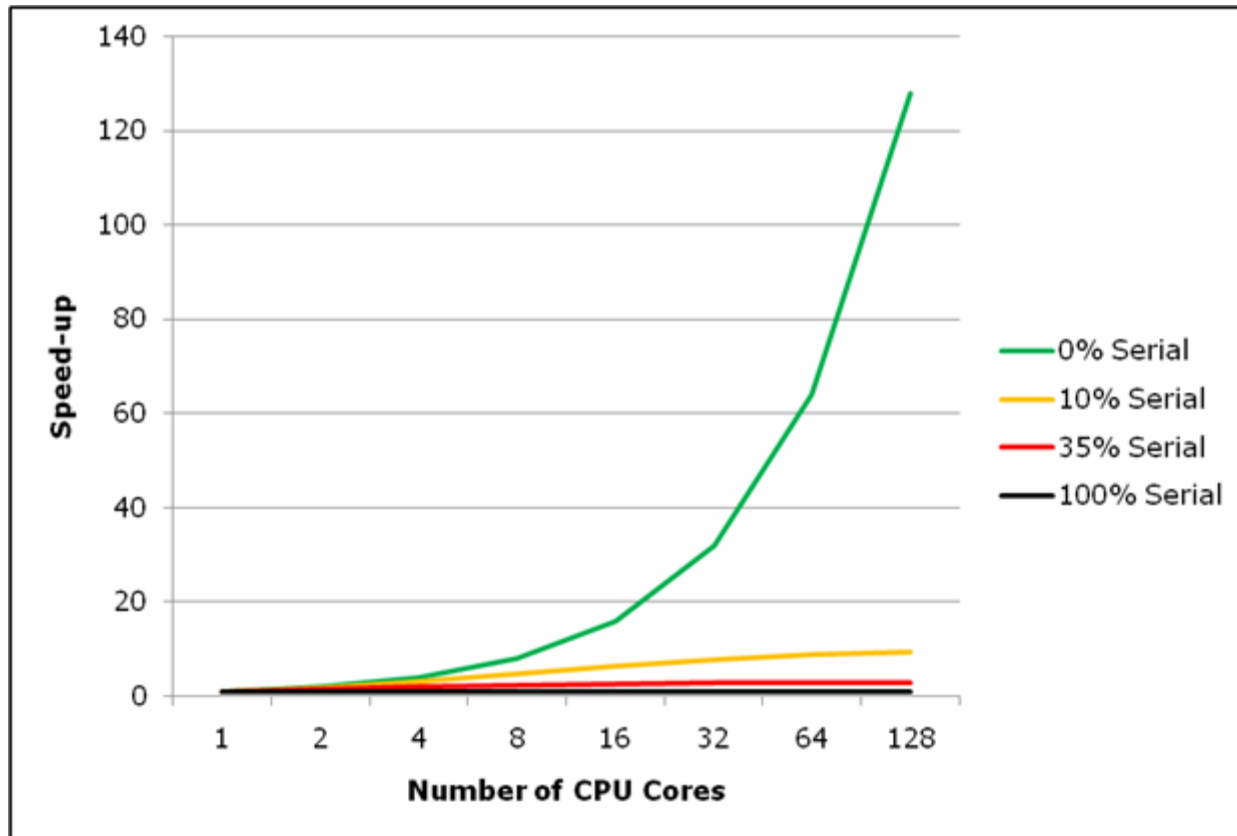
$$\text{Speed-up} = \frac{1}{S_W + (1 - S_W) / N}$$

S_W : % Serial Work

N : Number of processors



Amdahl's Law – Zoom out a bit ...



What about Throughput Computing?

- Measure Performance as *throughput* (vs. *turn-around-time*)
 - How long does it takes to run N “independent” tasks?
 - Multiple cores/threads should be faster than just one
- For basic multi-program throughput, the OS is the “serial component”
 - In some sense, the OS “offloads” work onto available cores
 - As some point, OS scalability becomes the bottleneck
- More advanced applications take on that role themselves
 - Modern databases
 - Some HPC apps turn data parallelism into task-level throughput
 - Future: Managed runtime environments → User mode scheduling
- Can we exploit large scale throughput computing?

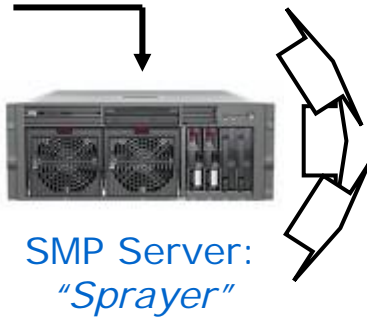


Large Scale Throughput *Systems*

- In these, there are far more threads than HW thread-slots
 - A **Centralized Controller** helps *dispatch/juggle* among threads

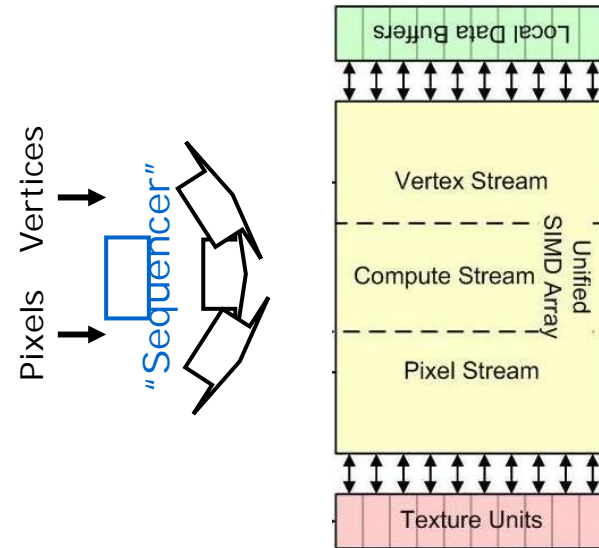
Two-Tiered Web Server

Massive amounts
of Internet *Clicks*



Blade Servers

Modern GPU



Cooperative *Heterogeneous* Computing!



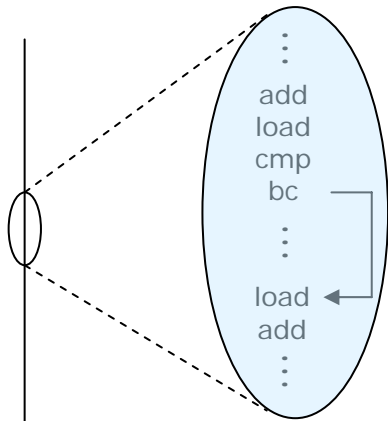
Characteristics of Throughput Computing

- The real goal is to saturate the pipeline to memory
 - Or any other very long latency pipeline, for that matter ...
 - Measure effectiveness by how well memory BW is used
- Hardware (Microarchitecture) Implications
 - Support of large number of active and standby threads
 - under-threaded designs won't saturate the pipeline to memory
 - Very robust memory controller - *lots of outstanding references*
- Software / Workload Implications
 - Separate natural parallelism into large # of independent threads
 - Applies to data parallel workloads and task parallel workloads
- What happens if the "Central Controller" gets backed up?
 - It is effectively the serial component in these topologies
 - A macroscopic version of Amdahl's Law kicks in



Data-level Parallelism and Throughput

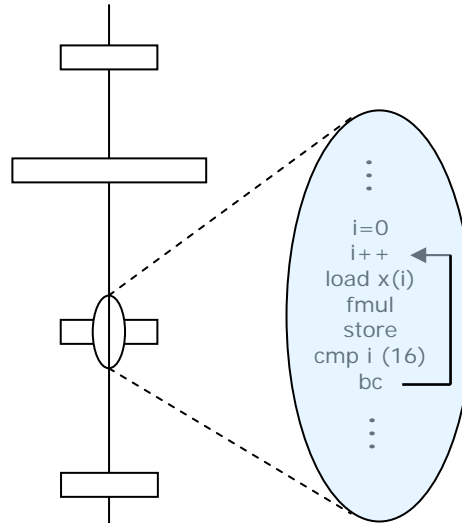
Mostly Serial Code



Typical control flow structure in most code

Some opportunity for instruction-level parallelism

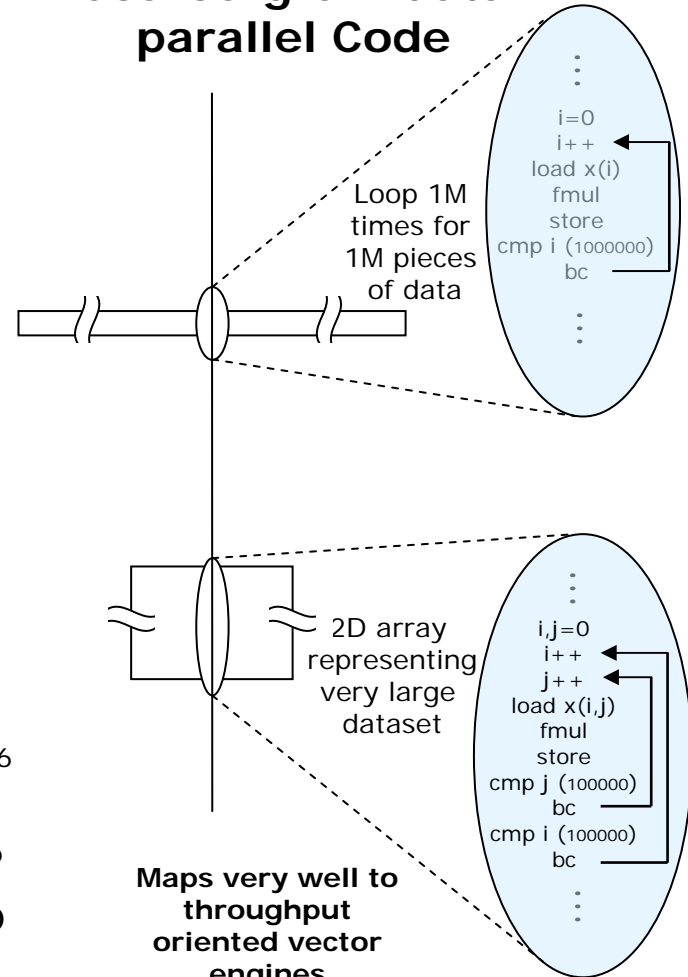
Fine-grain data parallel Code



Loop 16 times for 16 pieces of data

Maps very well to integrated SIMD dataflow (ie: SSE)

Coarse-grain data parallel Code



Loop 1M times for 1M pieces of data

Maps very well to throughput oriented vector engines



What does a General Purpose Throughput Computing solution look like?

- Start with a Powerful, yet efficient, Uniprocessor
 - Perhaps a next generation *Opteron* would be a good choice ☺
 - Deploy one or more of these for excellent general purpose legacy support, and as for the Centralized Controller for throughput
- Add a larger number of small, power-efficient, domain optimized compute offload engines
 - Not all such engines can afford an x86 compatible front-end!
- Build an optimized memory system
 - Shared among all the processors
 - Optimized communication, synchronization & data transfers
- Embrace a programming model abstraction that favors simplicity and programmer productivity



Distributed VS. Parallel Computing

Attribute	Parallel Computing		Distributed Computing
Example system topology	SMP	MPP	1) Internet 2) Data Center
Industry leaders	IBM, Sun, HP servers	HPC Niche	Google, Yahoo, MS Live
Programming	P-threads OpenMP	Message Passing (MPI)	HTTP, XML/SOA Mash-up APIs
Scalability	Poor	Moderate	Outstanding
Flexibility	Homogeneous	Homogeneous	Heterogeneous
Fault Tolerance	Failures are feared	Failures are feared	Failures are <i>assumed</i> in the architecture
20 year Progress assessment	Very poor	Moderate	Substantial



Optimized SW for System-level Solutions

- **Long history of Software optimizations for HW “characteristics”**
 - ▶ Optimizing compilers
 - ▶ Cache / TLB blocking
 - ▶ Multi-processor coordination: communication & synchronization
 - ▶ Non-uniform memory characteristics: Process and memory affinity
- **System-level Integration Era will demand even more**
 - ▶ Many Core: user mode and/or managed runtime scheduling?
 - ▶ Heterogeneous Many Core: capability aware scheduling?
- **SW productivity versus optimization dichotomy**
 - ▶ Exposed HW leads to better performance but requires a “platform characteristics aware programming model”
- **Scarcity/Abundance principle favors increased use of Abstractions**
 - ▶ Abstraction leads to Increased productivity but costs performance
 - ▶ Still allow experts burrow down into lower level “on the metal” details



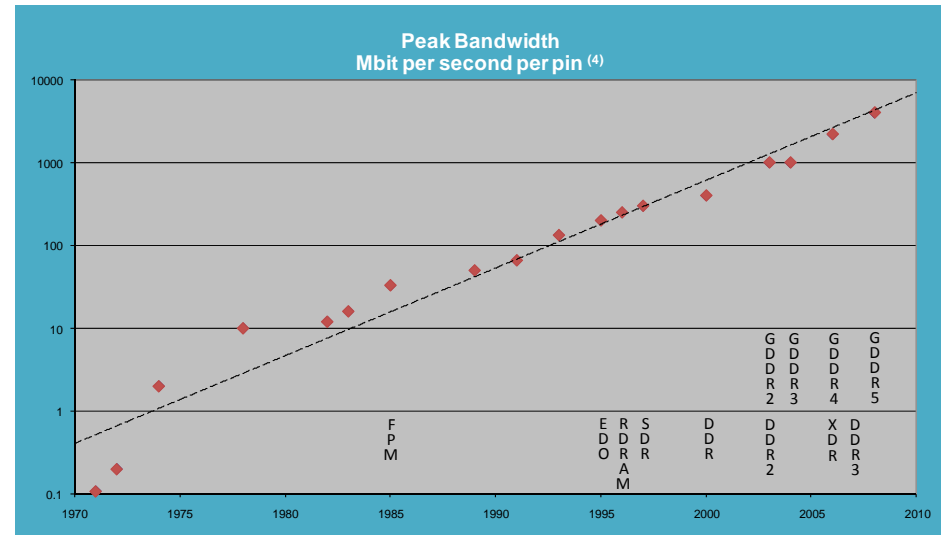
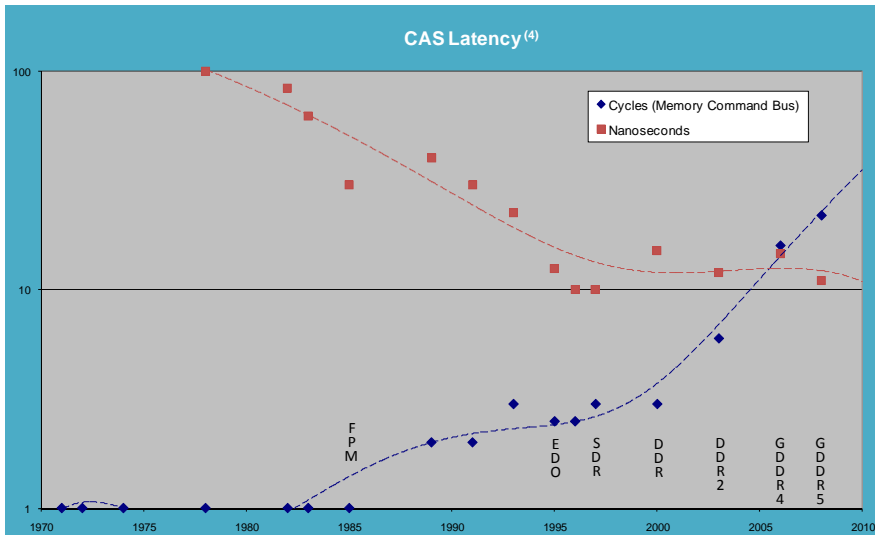
Challenges in the System-level Integration Era

- Complexity management
 - Principles for managing exponential growth
 - Development expense and TTM
 - Exploitation of available parallelism
 - Single thread performance outlook
 - Parallel threads, Throughput and Distributed computing
 - Optimized SW for System-level Solutions
- Memory system balance
 - The Power Wall issues keep getting *worse*



Memory System Balance

- Tricky balance between Latency, BW, Power, Capacity & Cost
 - CPU cores and caches very sensitive to latency
 - Many-cores, vector engines, GPGPUs all demand very high BW
 - Multimedia data types and modern SW demand increasing capacity
 - Everyone wants lower power and lower cost



(4) Data collected by author from various sources and are intended to illustrate the trends



The Memory Wall – *getting thicker*

There has always been a Critical Balance between
Data Availability and *Processing*

Situation	When?	Implication	Industry Solutions	
<u>DRAM vs CPU Cycle Time Gap</u>	Early 1990s	Memory wait time dominates computing	Non-blocking caches O-o-O Machines	☺
<u>SW Productivity Crisis: Round 1</u> Object oriented languages; Managed runtime environments	Early 1990s	Larger working sets More diverse data types	Larger Caches Cache Hierarchies Elaborate prefetch	☹
<u>Frequency and IPC Wall</u> CMP and Multi-Threading	2005 and beyond	Multiple working sets! Virtual Machines! More memory accesses	Huge Caches T'put Architectures Elaborate MemCtrls	☹!
<u>SW Productivity Crisis: Round 2</u> Increased abstraction layers Image/Video as basic data types	2009 and beyond	Even larger working sets Larger data types	Stream Computing <i>Chip Stacking?</i>	<i>TBD</i>



The Power Wall

- Easy prediction: *Power will continue to be the #1 design constraint in the System-level Integration Era*
- Why? Several conditions have worsened:
 - V_{\min} will not continue tracking Moore's Law
 - Thermal Design Points (TDPs) in all markets continue to drop
 - Lightly loaded and idle power characteristics are increasingly important in key markets
 - Integration of system-level components also consume chip power
 - A well utilized 100GB/sec DDR memory interface consumes ~15W for the I/O alone!
 - Percent of total U.S. energy consumed by computing devices continues to grow year-on-year



The Power Wall

- Another easy prediction: *Escalating multi-core designs will crash into the power wall just like single cores did due to escalating frequency*
- Why?
 - In order to maintain a reasonable balance, core additions must be accompanied by increases in other resources that consume power (on-chip network, caches, memory and I/O BW, ...)
 - Spiral upwards effect on power
 - The use of multiple cores forces each core to actually slow down
 - At some point, the power limits will not even **allow** you to activate all of the cores at the same time
 - Small, low-power cores tend to be very weak on general purpose workloads
 - The transition to compelling general purpose parallel workloads will not be a fast one
 - Customer value proposition will demand excellent performance on general purpose workloads



The Power Wall -- Implications

- Chip Multiprocessors (CMPs) will evolve to be heterogeneous
 - First, integration of cores with different capabilities
 - Second, integration of alternative programmable devices with superior performance/watt characteristics when running workloads of interest
 - Third, integration of extremely power efficient dedicated hardware assists for very commonly used functions
- Most meaningful metrics are (or will be) a ratio with power
 - Processor Cores and Chips: **Perf/Watt** and **Perf/Watt/\$\$**
 - GPUs & other data parallel throughput solutions: **FLOPS/Watt**
 - I/O SERDES PHYs: **mW/Gbit/sec**
- Very sophisticated next generation power management
 - Fully embrace power as a critical platform-level resource
 - Provision power based on real time monitoring and/or explicit requests
 - SOC components all on separable voltage and clock domains
 - Programmable uController for the base power management controller



Summary – *The System Level Integration Era*

- Chip Multiprocessors are just a first step in this bigger picture
 - Expect to see increased levels of system integration, heterogeneous cores and dedicated accelerators
- Large opportunity space for Microarchitecture innovation
 - Identification of the appropriate components to integrate
 - Establishing system-level balance between compute and integrated functions
 - Improvements in on-chip communication and synchronization
 - Scalable chip-level infrastructure
- Imperatives for the System-level Integration Era
 - Modularity and re-use
 - Never forget the uniprocessor! Continue improving each core.
 - The role of SW and abstractions for completing the picture



Thank You – Have a Great Conference!

Questions?

©2008. Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, the AMD Fusion logo, AMD Opteron, and any combinations thereof, are trademarks of Advanced Micro Devices, Inc.

Other names are for informational purposes only and may be trademarks of their respective owners.

