

Making Multicore Work and Measuring its Benefits



Markus Levy, president
EEMBC and Multicore Association

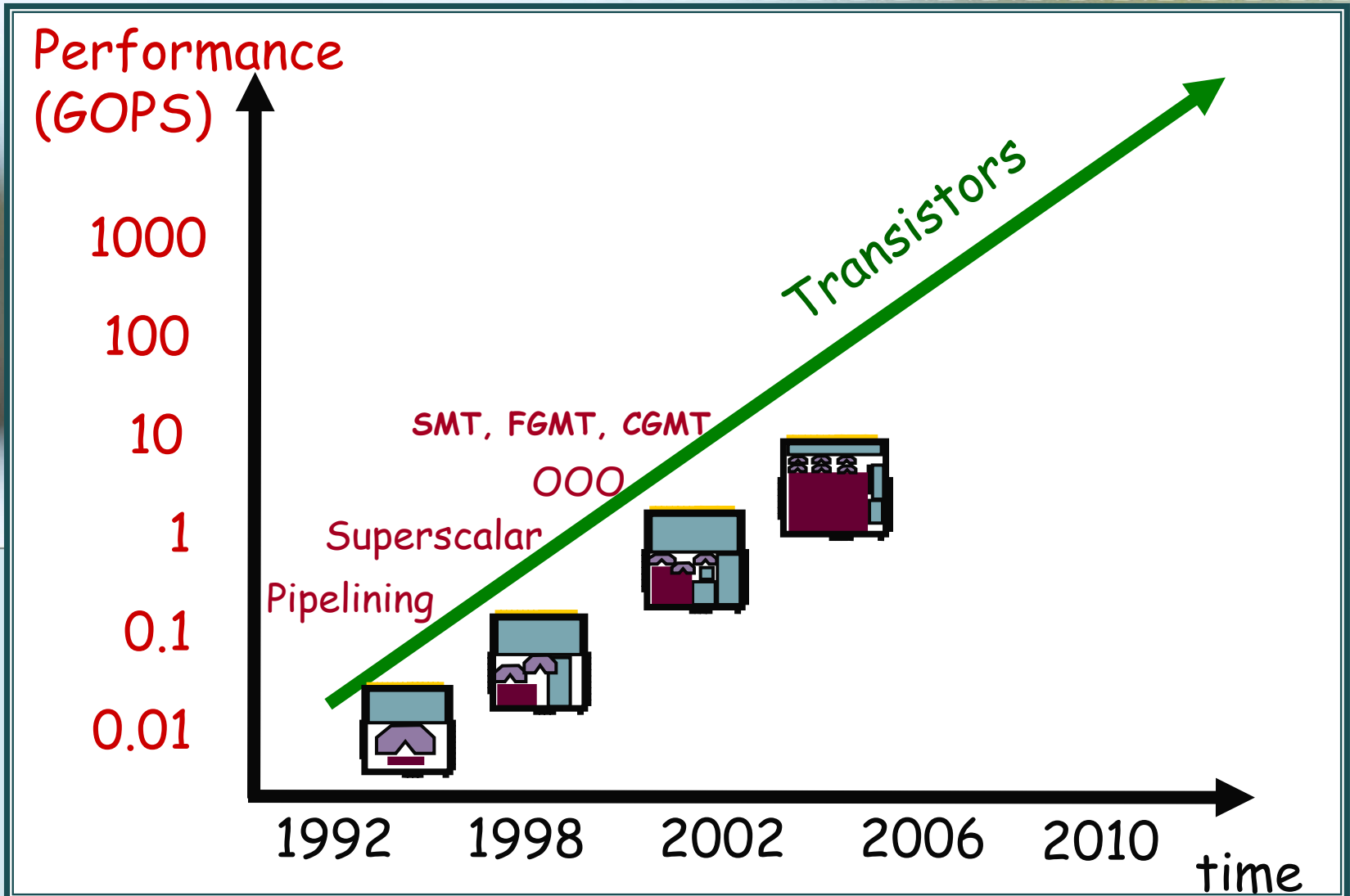
Agenda

- Why Multicore?
- Standards and issues in the multicore community
- What is Multicore Association?
- What is EEMBC?
- The multicore communication API
- Multicore performance and scalability
- Expected outcome in 2007

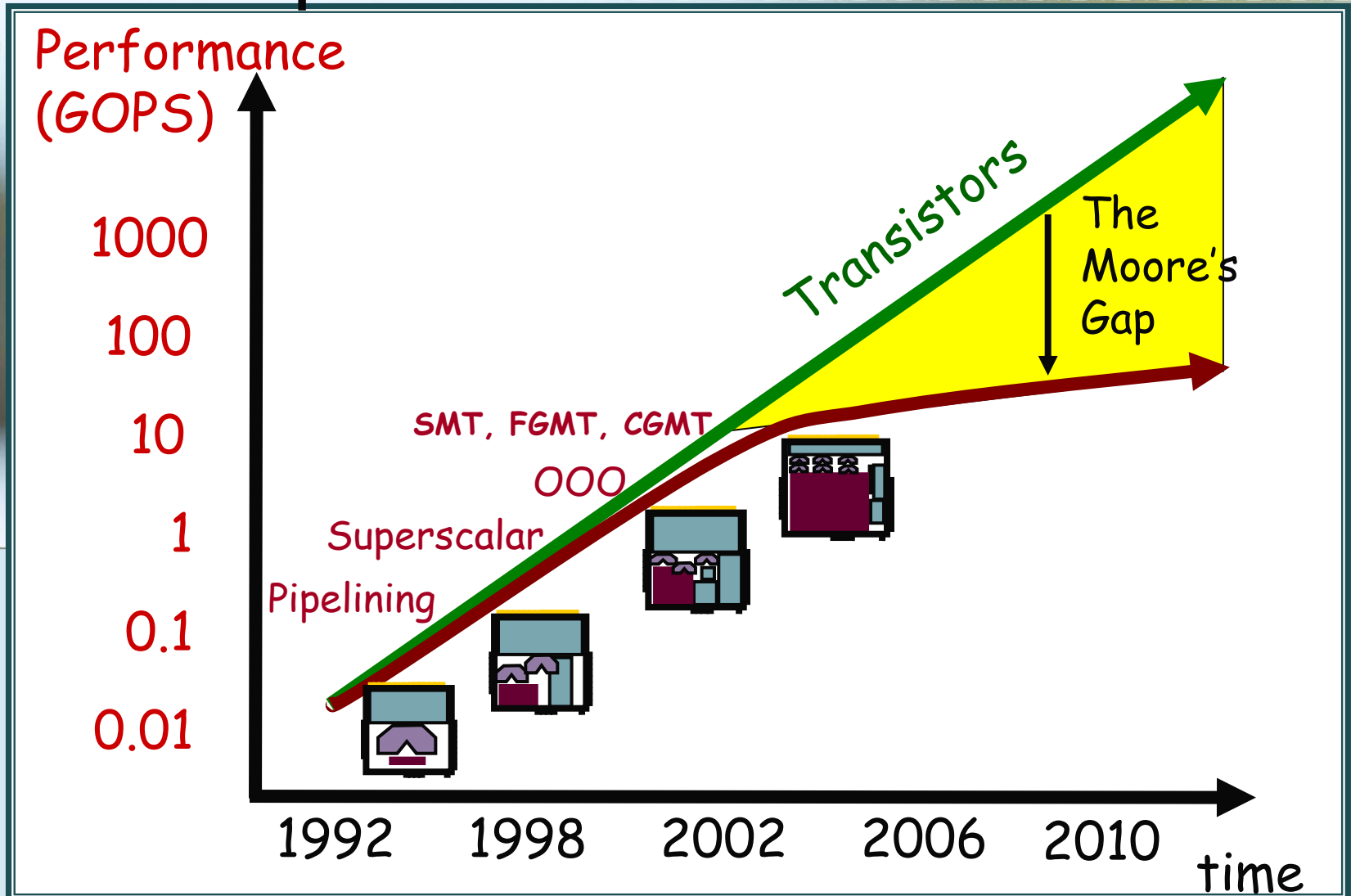
Multicore for Multiple Reasons

- Increase compute density
 - Centralization of distributed processing
 - All cores can run entirely different things; for example, four machines running in one package
- Functional partitioning
 - While possible with multiprocessors, benefits from proximity and possible data sharing
 - Core 1 runs security, Core 2 runs routing algorithm, Core 3 enforces policy, etc.
- Asynchronous multiprocessing (AMP)
 - Minimizes overhead and synchronization issues
 - Core 1 runs legacy OS, Core 2 runs an RTOS, other cores do a variety of processing tasks (i.e. things where applications can be optimized)
- Parallel pipelining
 - Taking advantage of proximity
 - The performance opportunity....

Ideal Trend of Transistor Utilization



“Moore’s Gap” Justifies More Cores



Decomposing Moore's Gap

- Diminishing returns from single CPU mechanisms
 - Pipelining
 - Caching
 - Functional units
- Wire delays
- Power envelopes

Transition Investment From Exponential to Linear

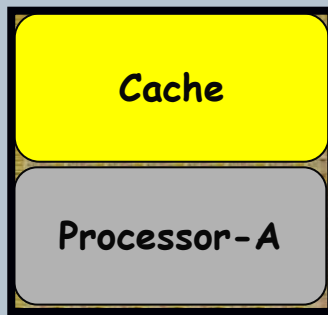
Closing Moore's Gap: Less is More

- Increase a resource only if for every 1% increase in area there is at least a 1% increase in performance
- Performance increase = Area increase
 - Percentage performance increase must be at least the same as the percentage increase in the area (power)
 - Remember the power of n : We can double performance by going from $n \rightarrow 2n$ cores
 - $2n$ cores have 2X the area (power)

“KILL Rule” for Multicore

Kill If Less than Linear

The Performance Opportunity



90nm

$$\text{CPI: } 1 + 0.01 \times 100 = 2$$

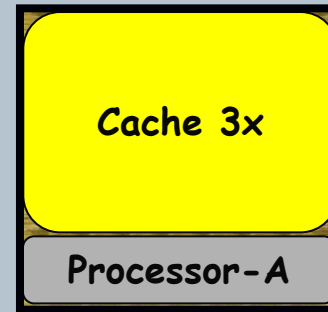
Miss rate = 1%

Latency to main memory = 100 cycles

90nm \rightarrow 65nm = 2x transistors

Smaller CPI (cycles per instruction)
is better

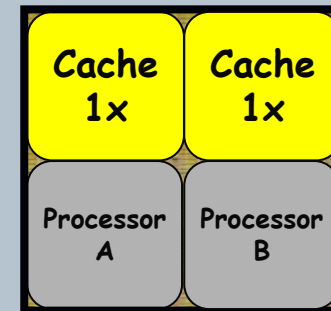
Push single core



65nm

$$\text{CPI: } 1 + 0.006 \times 100 = 1.6$$

Go multicore



65nm

$$\text{CPI: } (1 + 0.01 \times 100) / 2 = 1$$

Multicore Issues to Solve

- Hardware/software communication interconnect
- Modeling, simulation, and hardware/software co-validation
- Inter-core resource management
- Debugging: connectivity, synchronization
- Distributed power management
- Load balancing
- Algorithm partitioning
- Performance analysis



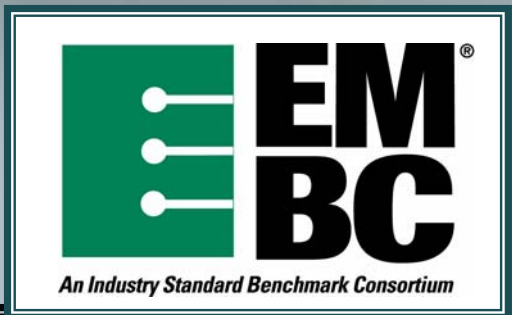
Enabling the Multicore Ecosystem

- Initial engagement began in May 2005
- Multicore ecosystem enablement
- Industry-wide participation
 - Including system developers, processor vendors, infrastructure developers, device manufacturers, EDA vendors, and software and application developers.
- Current efforts
 - Communications APIs
 - Debug API



Analyzing Multicore Processing Performance

- Embedded Microprocessor Benchmark Consortium (EEMBC)
- Industry standard benchmarks since 1997
- Evaluating current and future development of MP platforms
 - Uncovering MP bottlenecks
 - Comparing multiprocessor vs. uni-processor



The Abundance of Parallelism

- Networking/Telecomm
- Security
- General purpose: databases, web servers, multiple tasks
- Databases for differentiating services, etc
- Graphics and gaming
- Communications, cellphones
- Wireless
- Video imaging

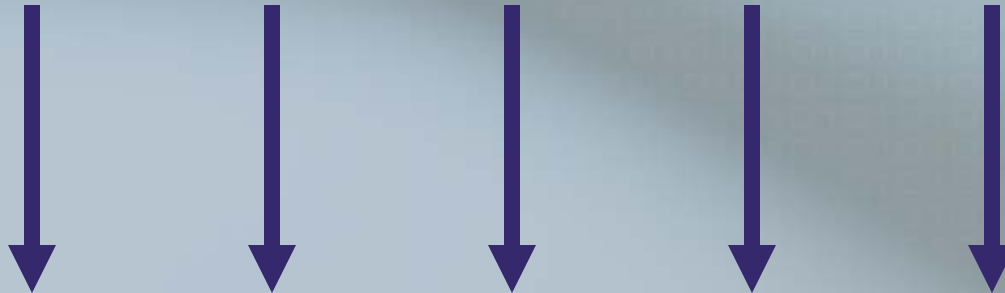
Wanted:
Lots of Computation Power

Possible Approaches

- Shared memory
 - Semantics of a thread-based API such as POSIX
- Message Based
 - Accelerator based asymmetric heterogeneous MP solutions today lack common programming paradigm
- Both approaches valid to support embedded applications which are supporting coherency and distributed memory architectures

Multicore Debug Challenges

- Need for higher level of abstraction with common debug semantics
 - No one would want 16 different debugger instances running to debug a 16 core system!
 - Need aggregate control and state inspection
- Dynamic debugging needed
 - Ability to look at things dynamically, without stopping the cores



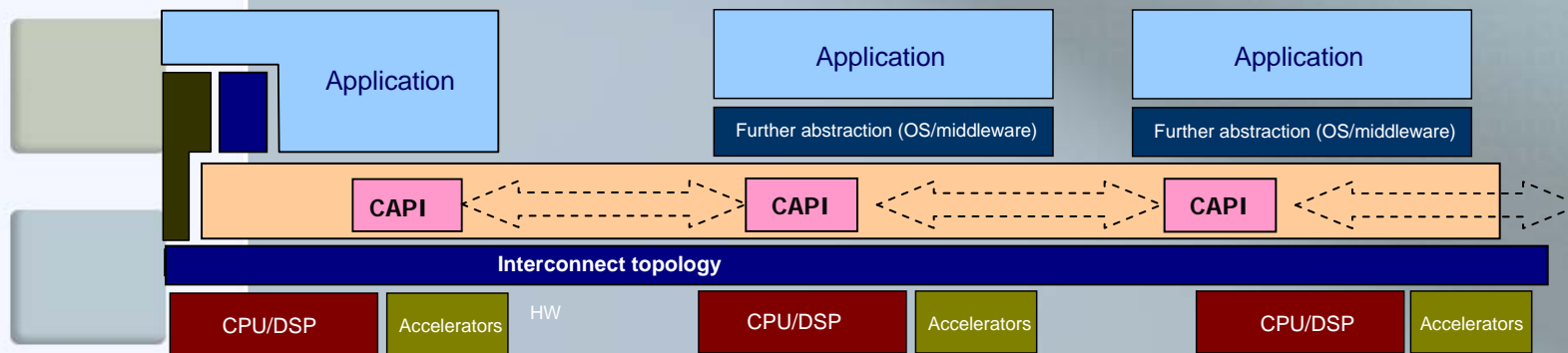
- Multicore Association plans to develop a standard debug protocol/API so vendors can support this type of development

Communication API Target Domain

- Referred to as MCAPI
- Embedded multi-processing requiring task to task communication
- Multiple cores on a chip & multiple chips on a board
 - Closely distributed and/or tightly-coupled systems
- Heterogeneous & homogeneous systems
 - Cores, interconnects, memory architectures, OSes
- Scalable: 2 – 1000's of cores
 - Assumes the 'cost' of messaging/routing < computation of a node
- Allows implementations with significantly lower latency, overhead & memory footprint than MPI and TIPC

Unified API for Closely Distributed Embedded System

- Forms layer on top of which other abstractions or applications may be built
- Provides OS vendor with a uniform foundation for present/future products and abstracts the platform capabilities
- Provides processor vendor with better determinism compared with MPI and reduced support burden
- Provides software developers with better ease of use and more powerful development methodologies



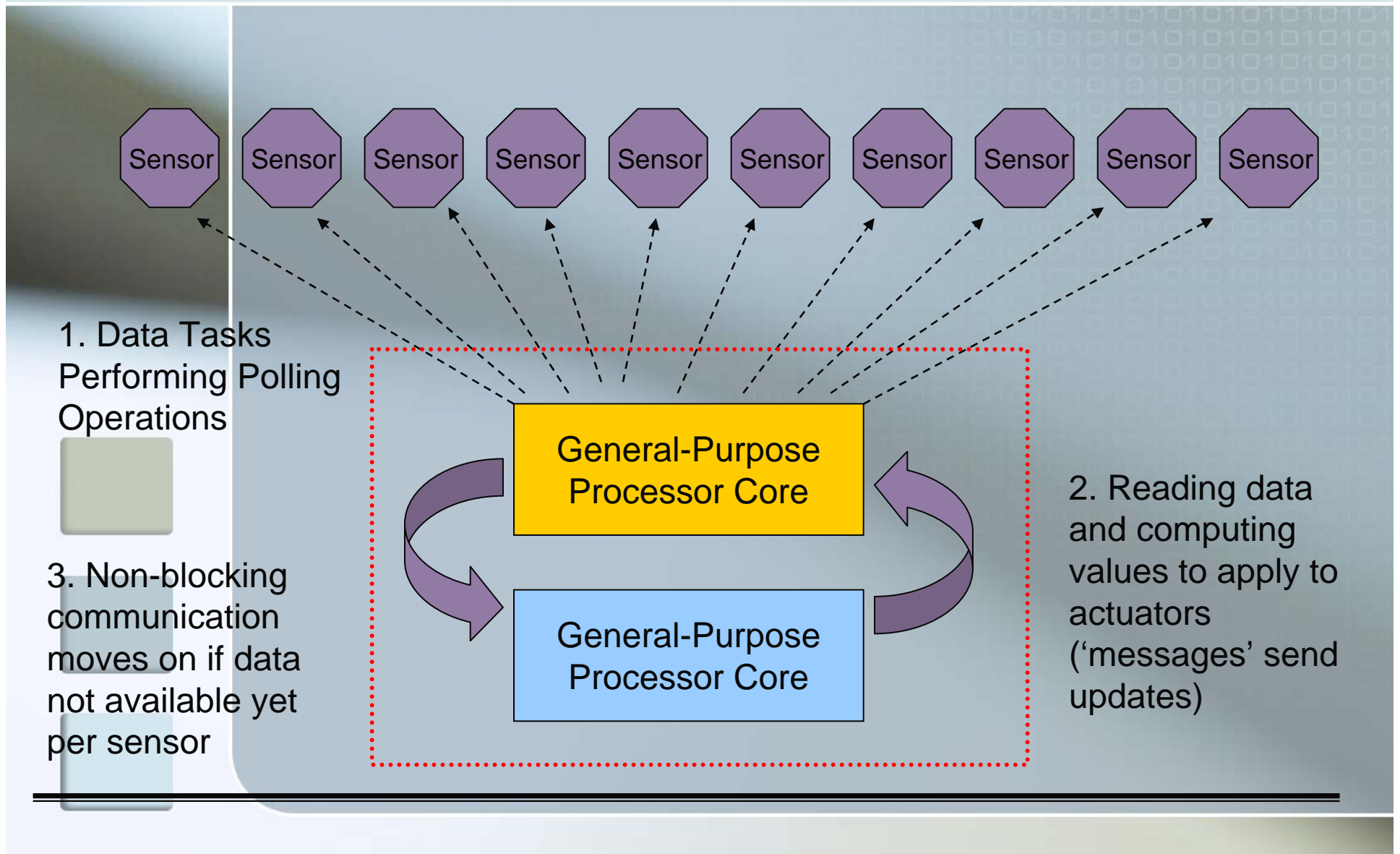
C-API Features and Performance Considerations

- Very small runtime footprint
 - OS ok, but not required
 - Significantly smaller code size
 - Significantly lower latency
- Low-level messaging API
- Low-level streaming API
- High efficiency
 - Enables efficient use of hardware accelerators
 - Zero copy features
 - No unnecessary data movement
- First draft of specification targeted at Q2 2007

Automotive Case Study in Engine Control

- Sensors are continuously monitored to determine the appropriate engine parameter settings
- General-purpose processor handles the data tasks polling the sensors
- Another GP processor handles the scheduled control task that reads the data collected by the data tasks and processes the data by computing values to apply to various actuators in the engine
- This workload and the associated transfer of data implies that the synchronization between control and data tasks must be minimal to avoid negative impacts on the latency of the control task

Fast Synchronization Between Data and Control Tasks



Packet Processing Case Study

- Streams contain packets 64 bytes to 1kbyte
- Example could be a TCP engine
- Multicore chip contains 3 to 100 cores
 - Depends on required bandwidth
 - Each core has small amount of local private memory or cache
 - Some cores may be specialized hardware accelerators
- Inter-core communication using on-chip interconnect
- Cores can also communicate through some common shared memory

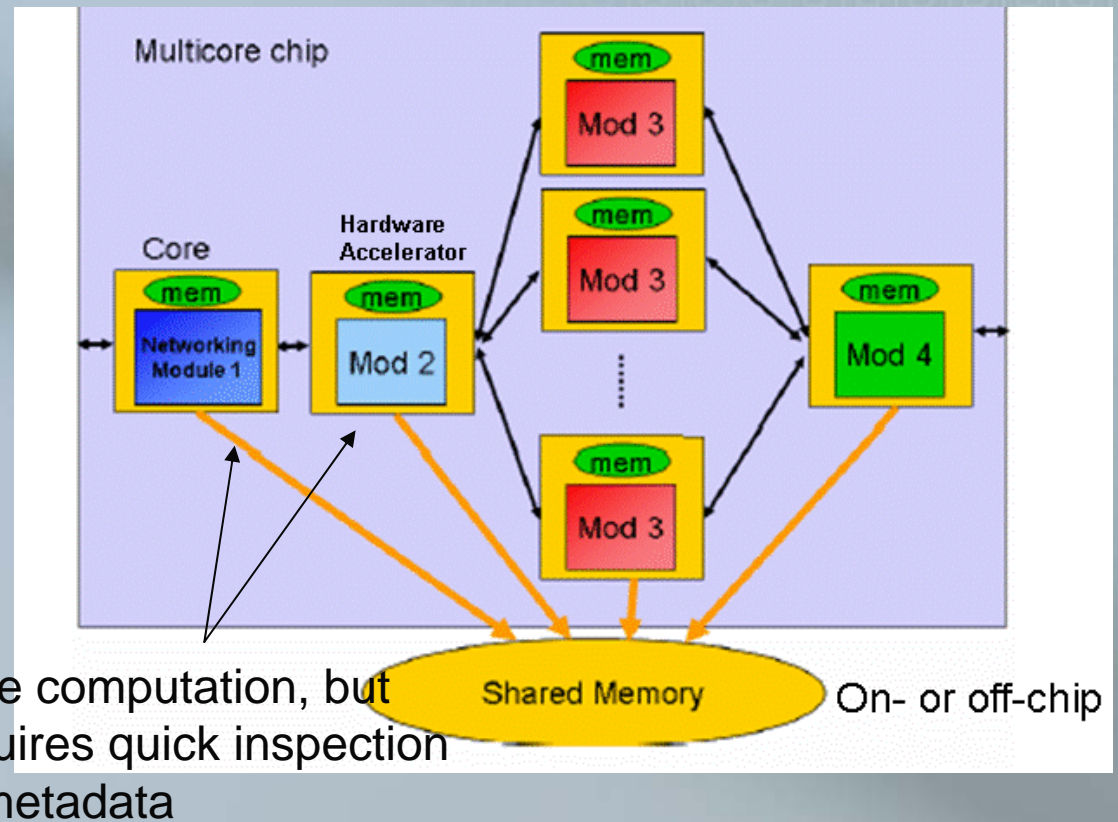
Packet Transfer Between Cores

- Mode 1: Packets streamed between the modules without going through external shared memory
- Mode 2: Packets placed in shared memory between modules
 - Modules accesses the packets from shared memory
- Hybrid mode: Packets placed in shared memory, packet descriptors and metadata are streamed directly between cores

Quick Synchronization

- Communication between the cores follows a stream pattern, occurring once or twice per packet
- Processing is both parallel and pipelined

Compute intensive modules



Benchmarking Multicore

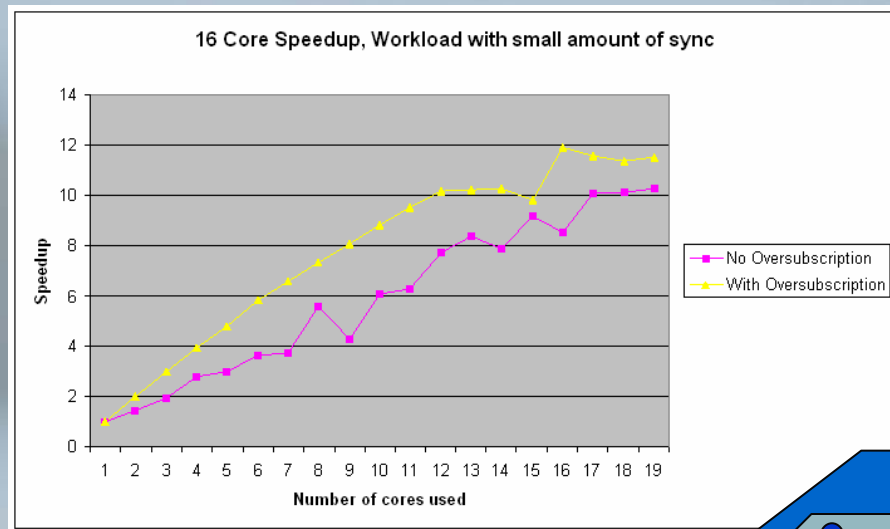
- Scalability
- Single versus multiprocessor
- Memory bandwidth
- OS support for scheduling

What's important?

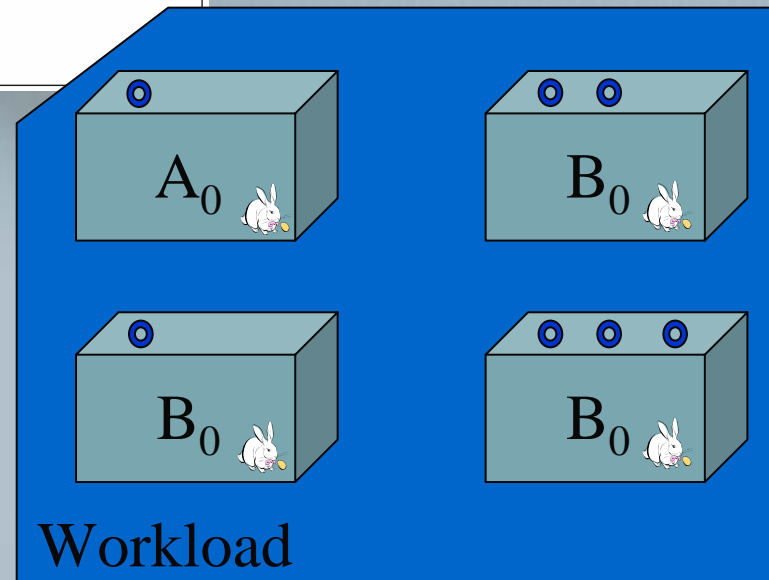
Benchmarking Multicore

- Software assumes homogeneity across processing elements
- Scalability of general purpose processing, as opposed to accelerator offloading
- Threads
 - Threading is the standard technique to express concurrency
 - Each thread has symmetric memory visibility
- Abstraction
 - Test harness abstracts the hardware into basic software threading model

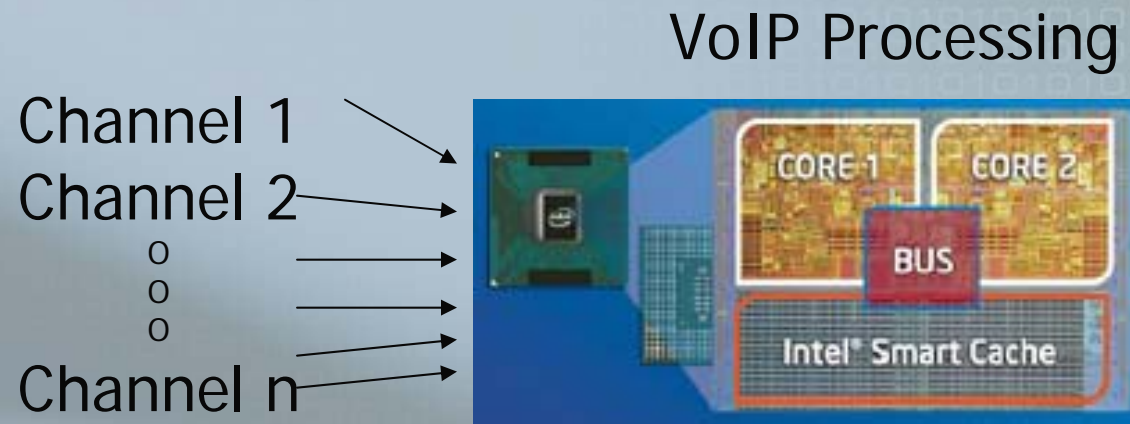
Scaling on multiple cores



Based on preliminary testing



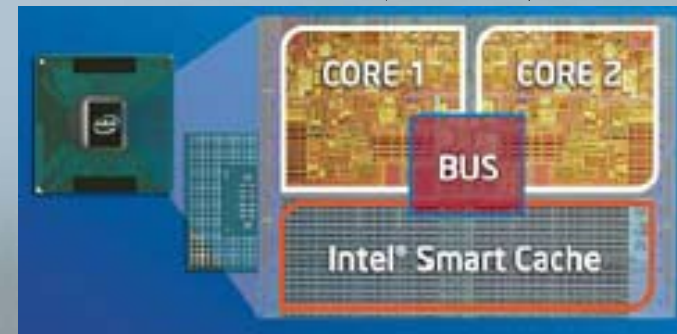
Processing of Multiple Data Streams



- Common code over multiple threads
- Shows how well a solution can scale over scalable data inputs

Decompose Single Task Into Multiple Subtasks

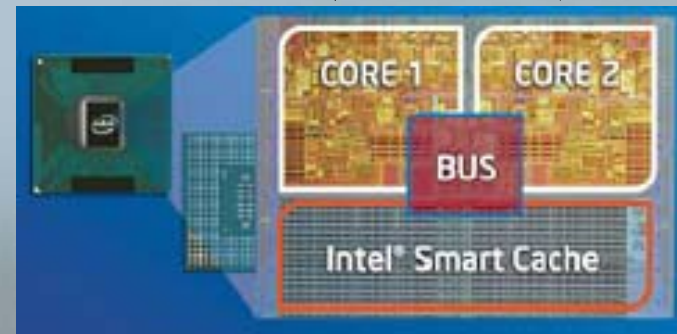
MPEG Decode



- Parallelize single algorithm to share workload between multiple processors
- Multiple threads work on common data set
- Demonstrates fine grain parallelism support

Execution of Multiple Different Algorithms

MPEG Decode MPEG Encode



- Concurrency over both the data and instructions
- Scalability of a solution for general purpose processing
- System (incl. OS) must handle workloads from multiple concurrent algorithms
 - Ex: mobile phone supporting a video call (Encode and decode algorithms, UI, possibly networking)

Multicore Opportunities and Challenges

- Multicore technology is inevitable
- It's time to begin implementing
- The Multicore Association will help ease the transition
- EEMBC will help analyze the performance benefits
 - Patent pending on new multicore technique