Max Out Your Multis (An Embedded Perspective)

Radhika Thekkath

April 17, 2008

At the core of the user experience.®

Topics

- Sorting out the terminology: multi-threading and multi-cores
- Enabling embedded multi-cores
- Adding hardware multi-threading to an embedded processor core
- Maxing out your multis
- Summary

All Those Words and Terminology

Homogenous

SMVP

Multiprocessors

Multi-Threading

Coherent

Heterogenous

CMP

Multi-Cores

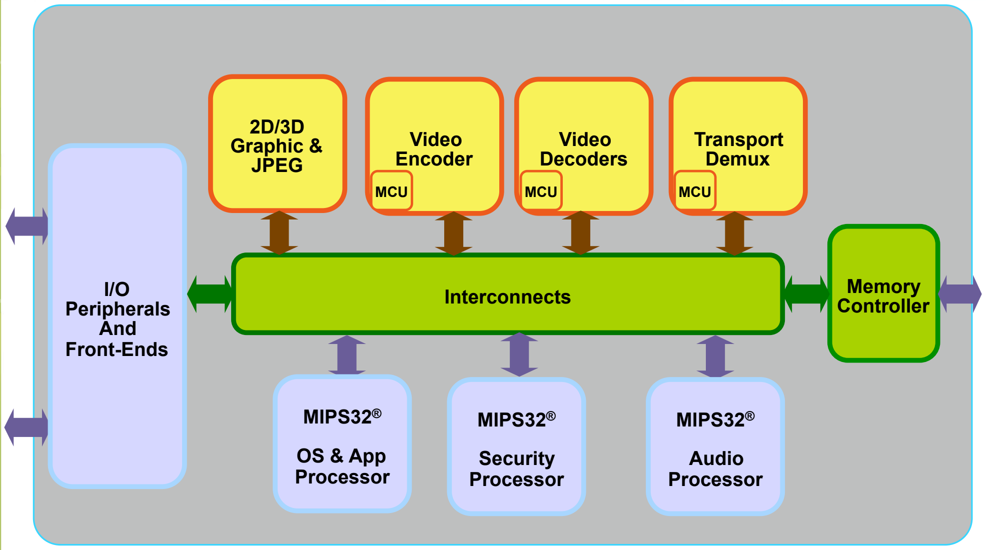
SMP

Non-Coherent

Multiprocessing—Before And Now

- **Multiprocessing is old stuff**
  - Remember the Sequent, the CM-2, the Exemplar, etc.?
- **Are we re-inventing this stuff all over again?**
- **What are we re-inventing?**
- **Multi-processing in the embedded domain (as opposed to the desk-top world)**
- **Even this is not new: embedded systems have always required multiple processing units, host control, audio, video, comm., etc.**
- **Single-chip multi-cores: homogeneous core-1 and core-2, video, etc.**
  - Audio and communication folded into core-1 and core-2

Example 1: a *Simplified* MIPS-Based™ SoC for DTV, DVD and STB Designs

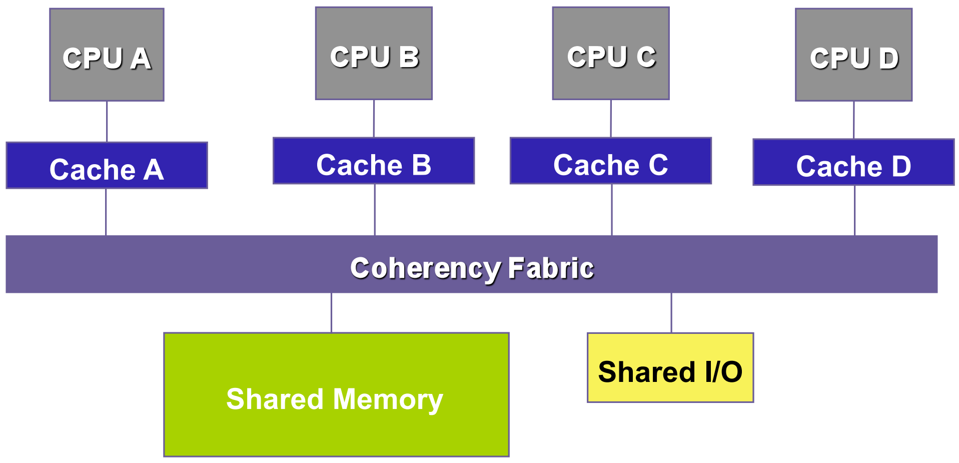


MIPS IP | 3rd Party IP | Customer IP

Embedded Multiprocessing

- Embedded multiprocessor designs usually connected point to point or sometimes on a single bus structure
- Evolutionary path leading to…
- Coherent multi-cores
- With these there is an immediate customer expectation of 2x or 4x performance gain
  - Comes with a substantial increase in die area and power
  - Cannot always get this type of performance increase, very dependent on:
    - target application and parallelism obtainable
    - Implementation design (competent or poor)

Multi-core SoC—An Example

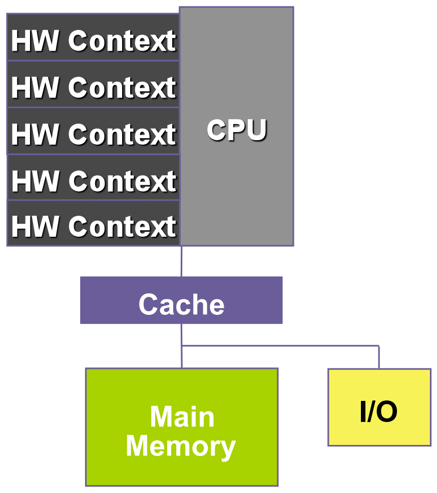


Embedded Multi-threading

- Multi-threading: software and hardware perspectives
- A software multi-threading architecture can execute multiple threads on a single CPU with a single hardware context by context switching the single hardware resource
- Hardware multi-threading implements multiple hardware contexts (registers mostly)
- Trade-off of some extra hardware for efficiency
- Focus on hardware-based multi-threading
- Gain in CPU efficiency and task throughput

At the core of the user experience.®

Multi-threaded Core—An Example



At the core of the user experience.®

Topics

- Sorting out the terminology: multi-threading and multi-cores
- **Enabling embedded multi-cores**
- Adding hardware multi-threading to an embedded processor core
- Maxing out your multis
- Summary

Making SoC-based Multi-cores Possible

- Come-together time for a bunch of ideas and direction
- Shrinking process technologies
- System architectural innovation
- Micro-architectural and implementation techniques
- Adaption of software tools and methodologies for embedded multi-core implementations

Shrinking Process Technologies

- Smaller implies
- More transistors on a die needed for:
  - Logic—implementation multiple everything, pipelines, functional units, etc.
  - Memory—each core needs its own first-level caches, scratchpad memories, etc. to be effective
- Dramatic performance advantage in the short term
  - Run multiple task and applications simultaneously—this is the easiest
  - Run multiple threads (parallelize) the application

System Architectural Innovation

- Optimization of coherence protocols—not necessarily related to the embedded world, but can be useful in certain contexts, e.g., data delivery short-cuts may be possible in a single-chip implementation because of some assumptions about latencies
- Interrupt protocol and its interaction with the DMA engine and I/O block
- Tracing mechanisms that span system boundaries

At the core of the user experience.®

Micro-architectural and Implementation Techniques

- Where did all the cycles go? Reducing the overhead of coherence protocols
  - Tightening up the path through the core to the coherence manager block and back
- Lock and sync implementation
  - Traveling the same path as data, but for control purposes
- Interrupt processing—optimizing for the embedded application
  - Although major components may look similar, the requirements could be very different compared to the desktop system
- Take short-cuts in data delivery—sometimes take liberties with the coherence protocol

At the core of the user experience.®

MIPS TECHNOLOGIES

At the core of the user experience.®

Software Tools for Embedded Multi-cores

- Tools for trace and debug must be enhanced
- Operating systems must understand the existence of multiple execution units
- Tools that can guide users on existing parallelism in applications
- Tools that can parallelize applications
- Performance analysis tools
- Look at the software track of any popular symposium (Multi-core expo, ESC, etc.), and you will trip over dozens of companies with tools and software offerings

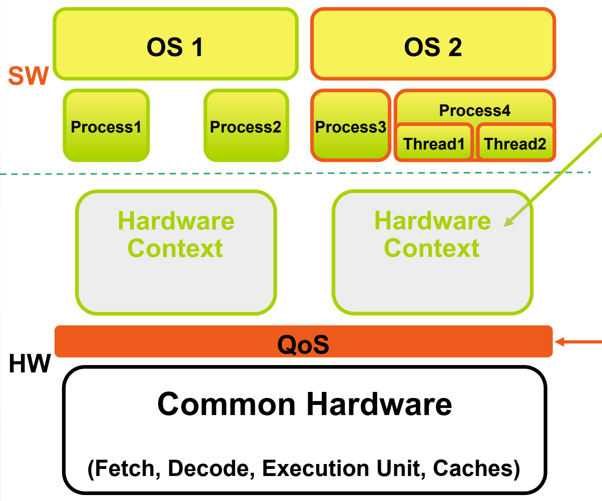
MIPS TECHNOLOGIES

At the core of the user experience.®

Topics

- Sorting out the terminology: multi-threading and multi-cores
- Enabling embedded multi-cores
- **Adding hardware multi-threading to an embedded processor core**
- Maxing out your multis
- Summary

Multi-threading—An Example Implementation

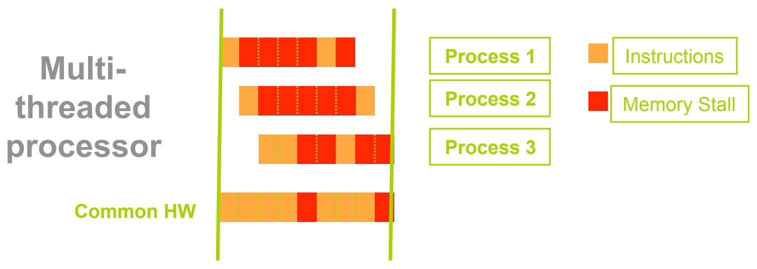


Virtual Processing Element
- Looks like a complete processor to SW
- Can run an OS or other software thread
- Build-time option: 1 or 2

Quality of Service
- Schedules resources between multiple contexts

- **Example implementation: MIPS32® 34K® core**
  - **Can run dual operating systems**
  - **Inherent cache coherency**
  - **Legacy applications run unmodified**

Where Do Stalls Go?

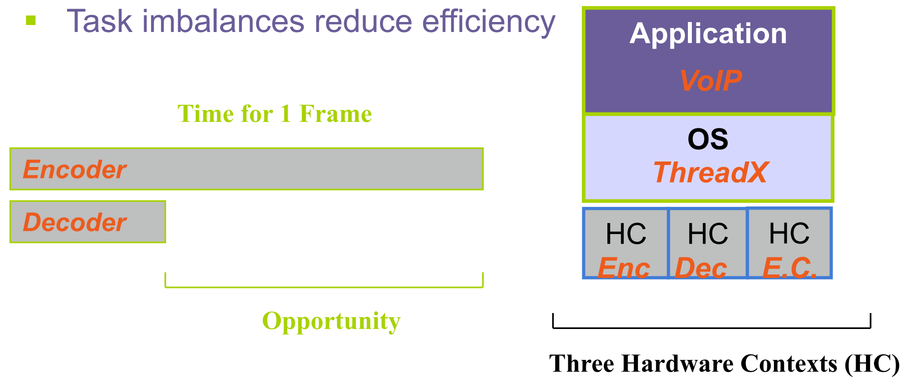


Why Multi-Threading?

- Multi-threading can eliminate or reduce stalls
- Stalls are a problem: memory latency continues to be an issue
- Other stalls include:
  - Synchronization
  - Pipe dependencies
  - Long latency operations
- Higher frequencies imply higher power consumption —can you do the same work at lower frequencies? Yes, if you are more efficient
- Multi-threading enables efficient behavior
- Power consumption is becoming a real issue with wired consumer devices like DTVs, STBs, etc. as they include more functionality and the SoC size grows

Why Multi-Threading for Applications?

- Contrast with multi-tasking
  - Targets “Application-level parallelism”
    - Diverse workloads
    - Example: VOIP
  - Task imbalances reduce efficiency



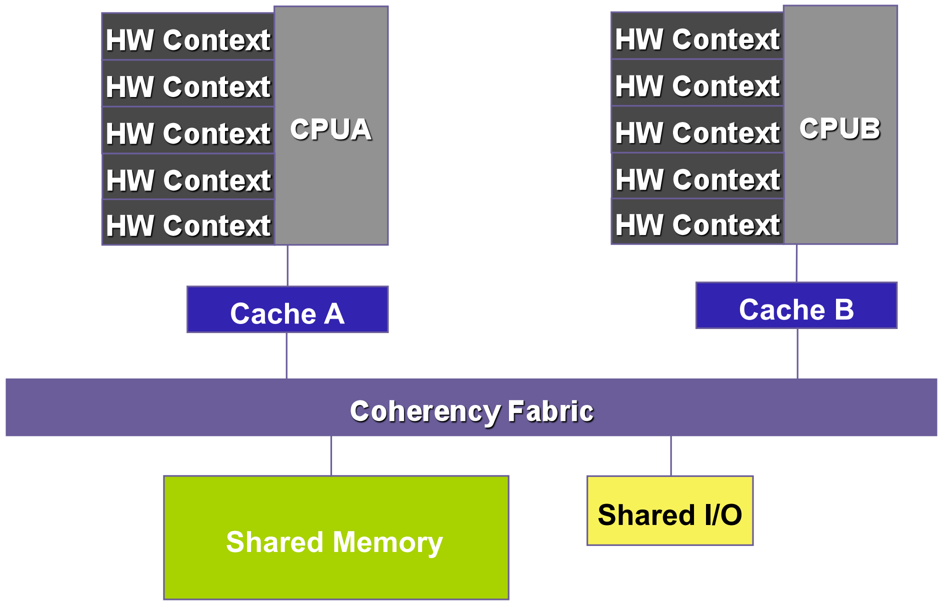
At the core of the user experience.®

Topics

- Sorting out the terminology: multi-threading and multi-cores
- Enabling embedded multi-cores
- Adding hardware multi-threading to an embedded processor core
- **Maxing out your multis**
- Summary

At the core of the user experience.®

Multi-threaded Multi-Core SoC—An Example

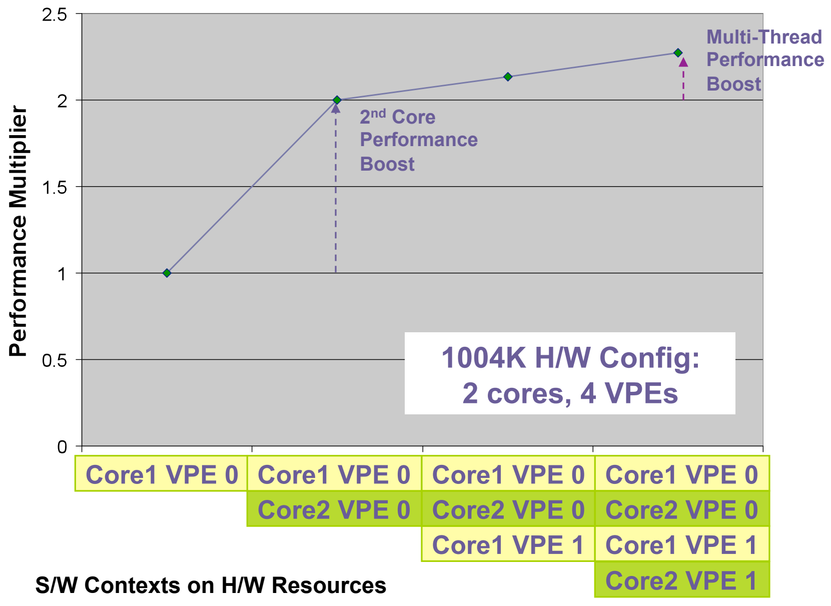


Why Multi-threaded Multi-cores?

- **Multi-threading and multiprocessing are complementary techniques**
  - Combination achieves pipeline and power efficiency with higher system performance levels—memory efficiency
  - Efficiency for the doubled or quadrupled hardware—leverage the multi-core hardware a step further
  - Both use the same parallel programming model, so software transition is seamless

Allocating S/W on a Multi-threaded Multi-processor

EEMBC Multi-core Benchmark Suite JPEG Decode – Typical Prelim Results



MIPS TECHNOLOGIES

At the core of the user experience.®

Summary

- Traditional multi-processing is here in the embedded, consumer world
- In the end, multi-threading and multi-core designs exist to make memory usage more efficient
- Combine them to get the power, efficiency, and performance boost