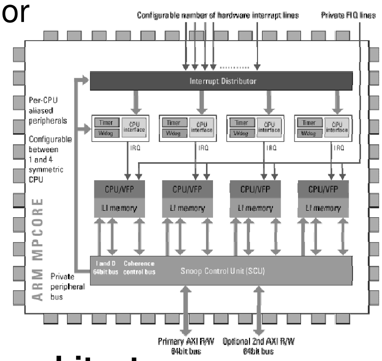
Extending the Cortex ARMv7 architecture for next generation multicore

John Goodacre,
Program Manager, Multiprocessing,
ARM Ltd

Summary of ARM Multicore Technology

- First announced in May 2004 with the release of the **ARM11 MPCore** multicore processor
  - Applied ARM MPCore technology to an updated ARM11 processor
- **MPCore enabled processors licensed to over 10 partners**
  - Broad section of applications, including networking, consumer, wireless and traditional embedded devices

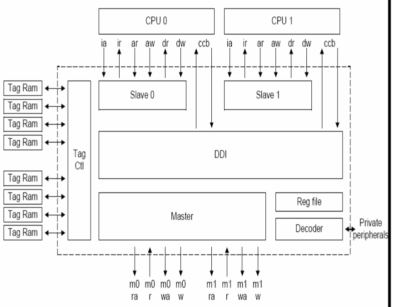


- **ARM11 MPCore uses the ARMv6 architecture**
  - Offering ARM application compatibility
  - A pre-integrated multicore processor offering scalable performance and advanced power management techniques for reduced power consumption

MPCore: Optimized MESI Snooping

Increased power efficiency and increased scalability by avoiding system accesses:

- Duplicated Tag RAMs acting as a local Directory
  - Stored in Snoop Control Unit of the multicore processor
    - Checks if data is in cache without interrupting CPU
    - Filters access to only CPU that are sharing data
    - Keeps power lower since directory is local
  - Allows independent tasks to run at full single thread performance resulting in linear scalability
- Direct Data Intervention (cache-2-cache transfer)
  - Copy **clean** data from one CPUs cache to another
  - Removing need for main memory accesses reducing associated power
- Migratory Lines
  - Move **dirty** data between CPUs and skips shared state.
  - Avoids power and latency associated with write back
- Read/Write cache allocation
  - With adaptive back-off for temporally inappropriate write allocation such as during a memset()



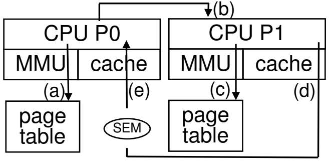
ARM MPCore Snoop Control Unit

- ❖ Clocked at CPU frequency for lower latency lookups and filtered access to CPU
- ❖ Keeps data within processor permitting lower power consumption than if time-sliced on a uniprocessor
- ❖ Power aware allowing per CPU logic and cache shutdown for advance power management.

ARMv7MP Adds h/w processor coherence

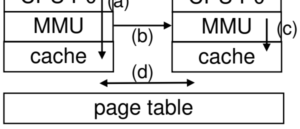
- **Memory Management Unit maintenance**
  - Processes may span multiple processors
  - Virtual memory map must be maintained so as to provide a common and consistent view of memory for each thread of a process on each CPU
  - Today's solution broadcasts an interrupt to all CPU requesting update and blocking until all processors respond
  - ARMv7MP multicore processors contain hardware to ensure coherence of MMU
- **Cache maintenance**
  - For non-coherent devices, it is necessary to 'flush' the appropriate cache regions prior to the device access the associated memory
  - Today's solution also use a broadcast interrupt based notification
  - ARMv7MP provides hardware to synchronize cache operation

Today's s/w processor coherence



(a) P0 executes local request
(b) Sends interrupt to P1
(c) P1 executes local request
(d) P1 notified update completed
(e) P0 waits until semaphore clear

ARMv7MP Processor Coherence



(a) P0 executes local request
(b) h/w echoes request to P1
(c) Without s/w, P1 executes request
(d) Cache coherence maintains table

Cost of today's s/w processor coherence

Measuring effect on a ARM quad-core during
Linux 2.6.19 OS boot to X11 shell

~ 50,000 occurrences of software controlled processor coherence
Each CPU interruption cost each other CPU
measured around 2000 cycles of normal software execution

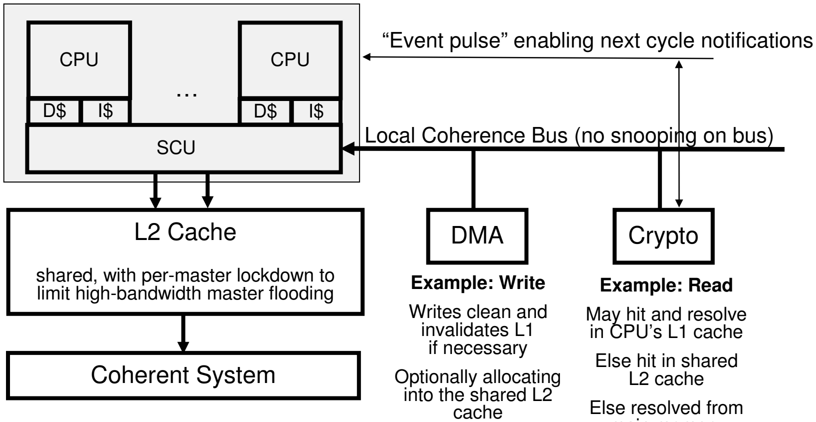
Cost of software based processor coherence:
(50,000 occurrences) x (3 other CPUs) x (2000 cycles lost) =

300,000,000 wasted CPU cycles

ARMv7MP Processor coherence expected to notably reduce the time of Linux SMP boot in addition to removing the need for complex inter-processor synchronization software

Supporting Accelerator Coherence

Sharing benefits of the ARM SCU optimized coherency implementation
Accelerators gain access to CPU cache hierarchy, lowering system power and increasing overall performance


CPU
CPU
…
D$
I$
D$
I$
SCU
"Event pulse" enabling next cycle notifications
Local Coherence Bus (no snooping on bus)
L2 Cache
shared, with per-master lockdown to limit high-bandwidth master flooding
Coherent System
DMA
Crypto

Example: Write

Writes clean and invalidates L1 if necessary

Optionally allocating into the shared L2 cache

Final write back

Example: Read

May hit and resolve in CPU's L1 cache

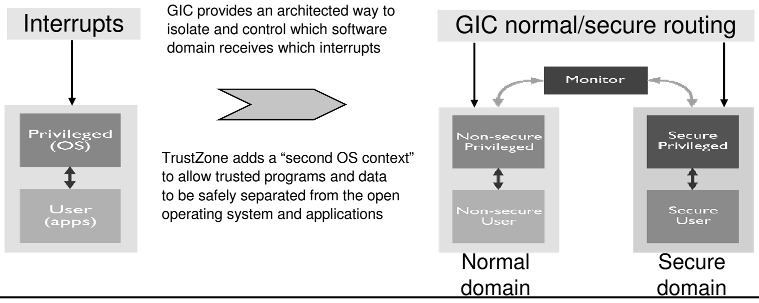
Else hit in shared L2 cache

Else resolved from main memory

ARM TrustZone® and GIC Architectures

- Previously extended the ARM architecture to include additional privilege levels for the execution of an additional OS within the secured software domain
- Along with ARMv7MP the architecture has been further extended to manage and secure processor interrupts between the normal and secured software domains
  - Known as the ARM Generic Interrupt Controller (“**GIC”**) Architecture

- Together, a TrustZone and GIC architecture enabled ARMv7MP processor can:
  - Restrict and control interrupts between the two software domains
  - Partition and control access to system resource between the normal/secure domains
  - Host ‘management’ software within the secured domain without altering the normal OS



GIC provides an architected way to isolate and control which software domain receives which interrupts

TrustZone adds a “second OS context” to allow trusted programs and data to be safely separated from the open operating system and applications

ARMv7MP: Improved Paravirtualization

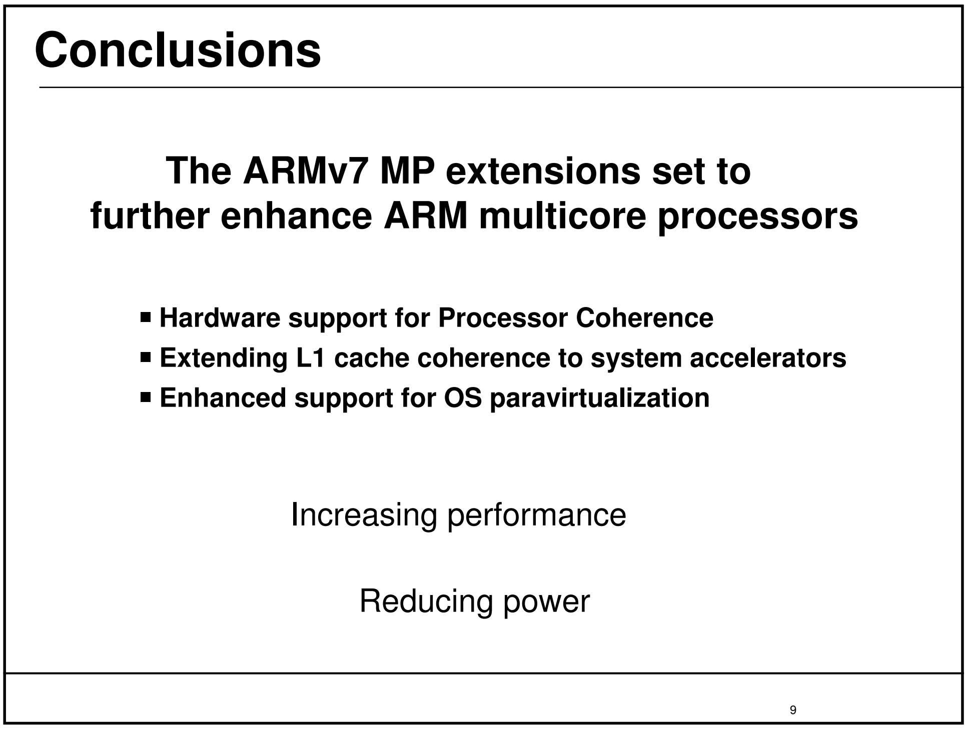
- Today’s Paravirtualization solutions
  - Provide the ability to run multiple independent OS/RTOS on a single processor
  - Requires a notable modification of a OS port to appropriately defer all privileged operations to a virtual machine manager (VMM) managing processor resource sharing

- ARMv7MP processors
  - Enable concurrent execution of multiple paravirtualized operating systems
  - Providing improved real-time response and dynamic load balancing

- ARM TrustZone Architecture
  - Allows the open OS to maintain their User and Privilege states and run the VMM in the privilege contexts of the secured software domain
  - Provides an AMBA architecture mechanism to signal TrustZone context accesses to peripherals and any system based memory protection units
  - Cache state is maintained and secured between the open and TrustZone OS context during virtualization traps and request forwarding

- ARM GIC Architecture
  - Allows the VMM to manage and arbitrate access to drivers for each open OS

- Together providing an accelerated and simplified paravirtualization solution



Conclusions

The ARMv7 MP extensions set to further enhance ARM multicore processors

- **Hardware support for Processor Coherence**
- **Extending L1 cache coherence to system accelerators**
- **Enhanced support for OS paravirtualization**

Increasing performance

Reducing power