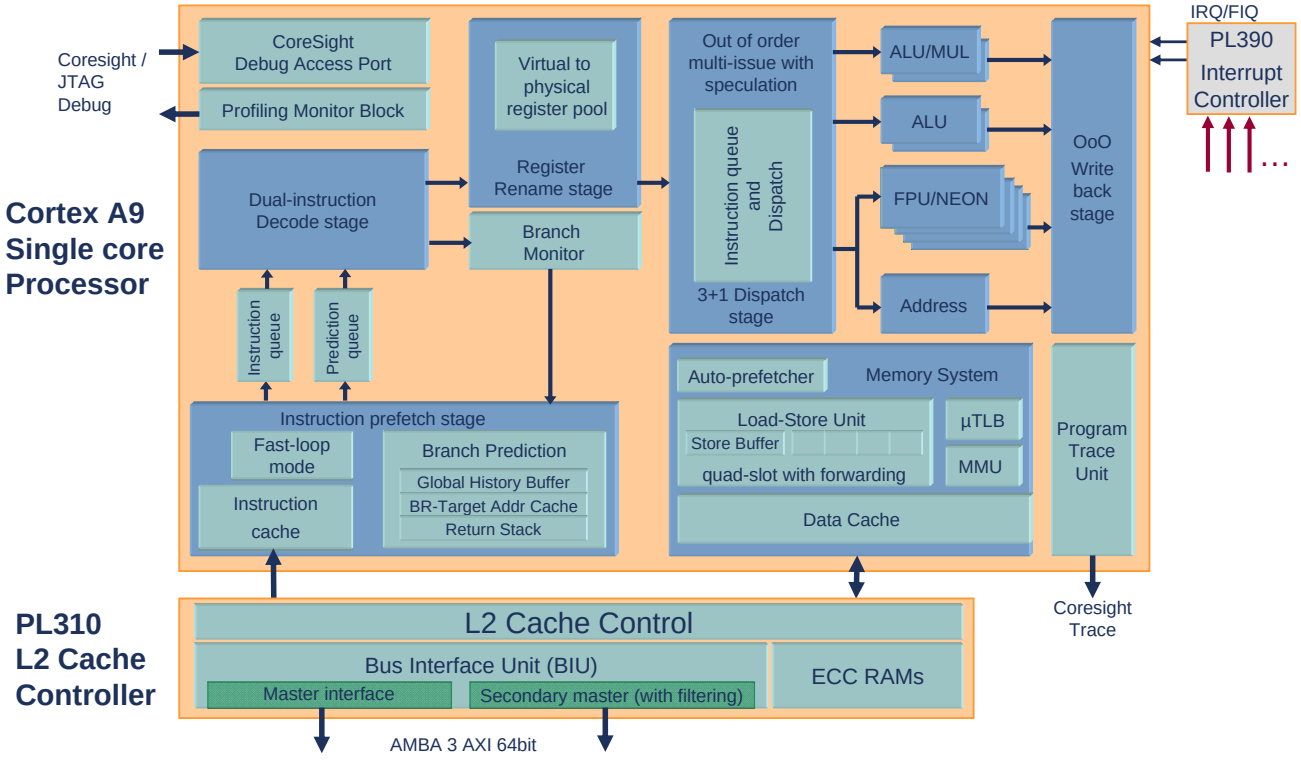
The Effect and Technique of System Coherence in ARM Multicore Technology

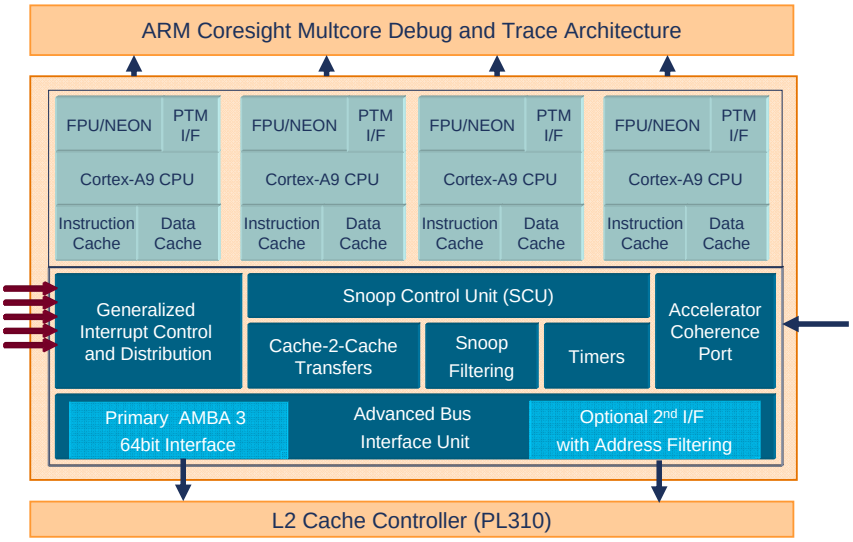
John Goodacre
Senior Program Manager
ARM Processor Division

Cambridge, UK

Cortex™-A9 Microarchitecture (single core variant)



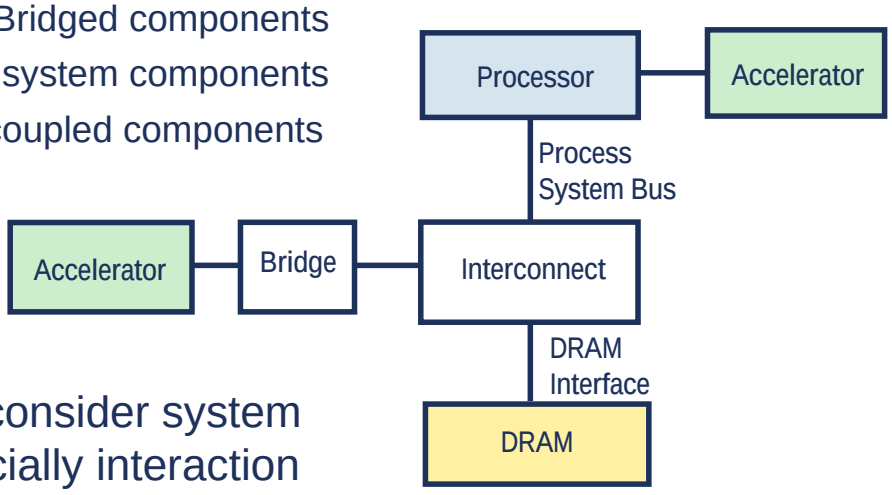
Cortex-A9 MPCore™ Processor Structure



- Tightly coupled multiple core configuration enables range of AMP and SMP configurations, configurable at boot time
- Modified MESI coherency protocol for cache-to-cache transfers and direct data intervention
- Accelerator coherence port (ACP) enables acceleration engines and peripherals to share multi-core optimized coherency design
- Event bus across cores for fine grained communication within coherency domain

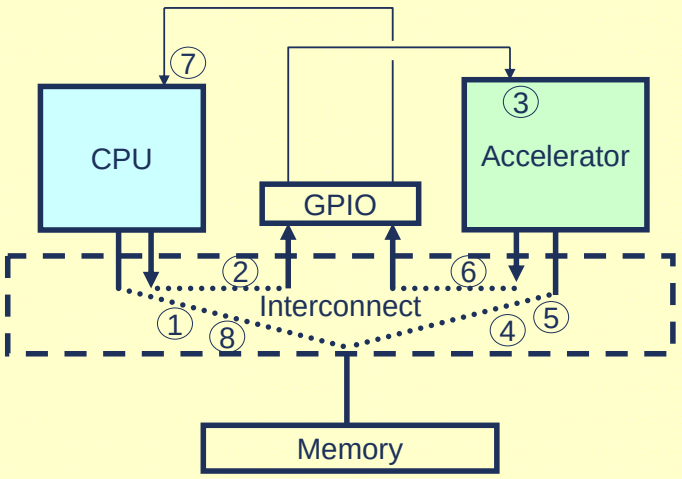
Addressing system performance

- System performance is not only defined by the processor
  - Speed/width and latency of DRAM and processor system interface
  - Connectivity and efficiency interacting with system components
    - Offchip Bridged components
    - On chip system components
    - Tightly coupled components



- Important to consider system design, especially interaction with other system components such as DMA and accelerators

Traditional Accelerator SoC Integration

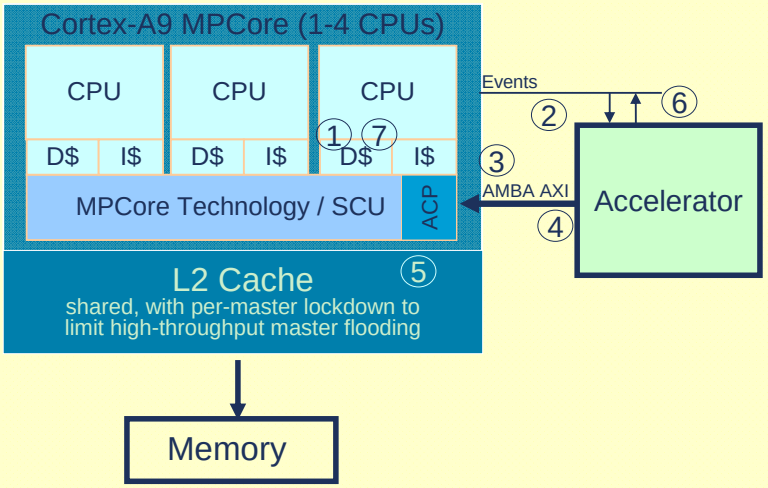


1. CPU flushes data from cache to make visible to accelerator.
2. Writes to mailbox to indicate availability of data
3. Interrupt serviced by accelerator
4. Read data from memory
5. Writes result back to memory
6. Notifies availability of result data
7. CPU services interrupt
8. Reads accelerator result data

- Analysis of traditional SoC accelerator SoC integration
  - Inefficient usage of CPU cache
  - Significant performance and power implications from data movement
  - High signalling latencies due to mailbox access and interrupt latencies

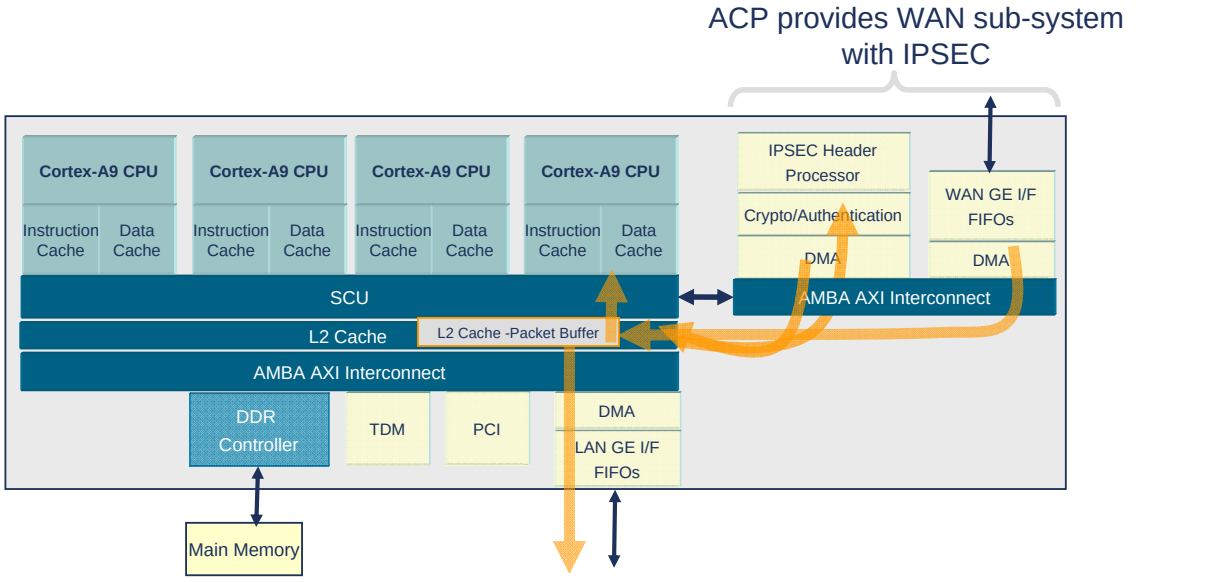
Enhanced Accelerator SoC Integration

- ARM MPCore: Accelerator Coherence Port (ACP)
  - Simplified software and reduces cache flush overheads
  - Accelerators gain access to CPU cache hierarchy, increasing system performance and reducing overall power
  - Uses AMBA® 3 AXI™ technology for compatibility with standard un-cached peripherals and accelerators



1. CPU leaves data in the cache
2. Next-cycle notify to accelerator to check and process data
3. Accelerator issues read - Data may return from L1, L2, or from main memory
4. Accelerator issues write of result, L1 coherence ensured
5. and may be configured to allocated into L2 cache
6. Accelerator raises event to CPU to check for result data
7. CPU issues read, may hit in L2

IPSEC Acceleration Using I/O Coherency



1. WAN port receives packets and allocates packet into L2 coherently

2. Core looks at header, decides it is encrypted and interrupts IPSEC block

3. IPSEC block processes the packet and modifies coherent memory

4. ARM core performs NAT/QoS/Firewall and sends packet through LAN port

ACP - Access to Shared Caches

- Example: CRC engine for TCP packet forwarding on 64 byte packet
  - Using typical system latency, ignoring common processing overhead
  - Assumed writes are fully buffered

Algorithm Stage	Approximate Cycle Counts	
Design style	Traditional shared memory with mailbox communication	ACP attached accelerator with synchronous event
Packet received and processed by CPU	0	0
Flush cache to make data visible to accelerator	20	0
Accelerator notified of data availability	> 4 [write to mailbox GPIO]	1 [Send Event]
Accelerator Reads data from cache line	Typical - 120 [read data from off-chip]	10 [read from L1/L2]
Accelerator Write data (assuming buffered)	8	8
Processor reads processes cache line	Typical - 120	12 [from L2]
Total latency overhead	**~272 cycles**	**~31 cycles**

ACP solution is appropriate for cycle-offload accelerators executing in 100's of cycles with cache resident workloads.
For example in low latency situations required by audio echo cancelation

Summary

- ARM Cortex-A9 has now exposed the MPCore coherence technology to the wider SoC
  - Interface is known as “Accelerator Coherence Port (ACP)”
  - We’re hearing about ~25% reduction in memory transactions due to reduction in cache flushing
- Software no longer needs to be concerned with cache flush, which can be particularly troublesome on a multicore

Q1’08

- First devices expected be in sample around end of this year