Future architectures of MPSoC Platforms

John Goodacre
Senior Program Manager
ARM Processor Division

1st Workshop on
Mapping of Applications to MPSoCs

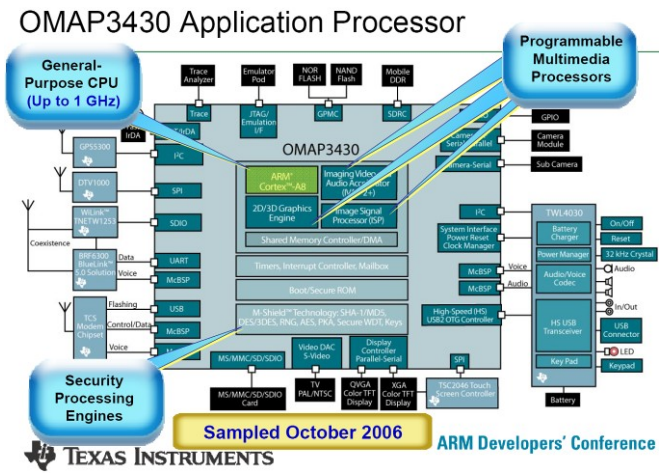
June 16th 2008

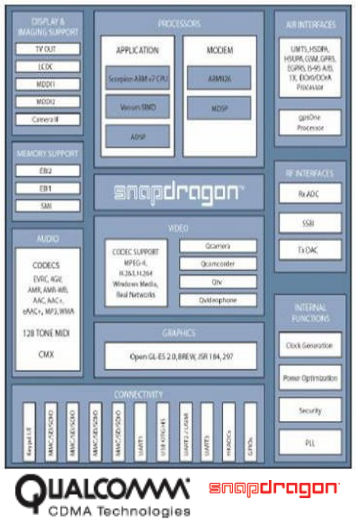
The Language (I use) for “MPSoC”

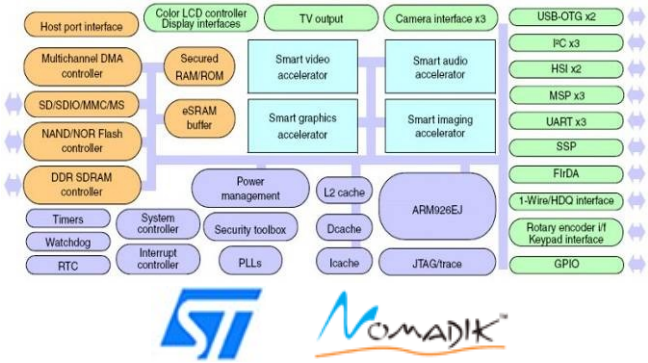
- Multiprocessor
  - A computing platform utilizing more than one processor unit
    - MPSoC: A multiprocessor delivered within a single system on chip
  - Each processing unit is associated with a single domain of software
    - -> Asymmetric software design paradigm
  - Typically utilizing heterogeneous processors

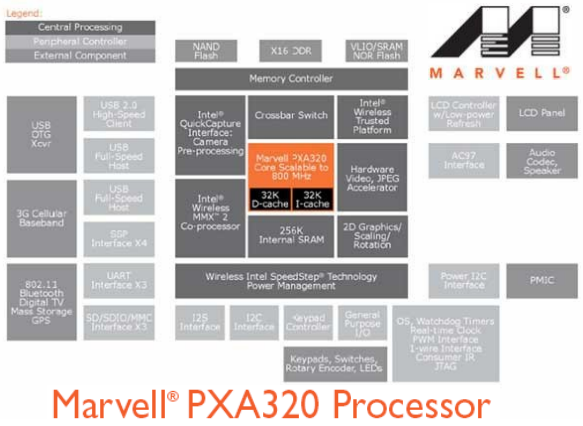
- Multicore processor
  - A processing node that can support a single domain of software across multiple processing units
    - -> Symmetric software design paradigm
  - Typically homogeneous processors (but not necessarily)
    - Always an unified instruction set architecture

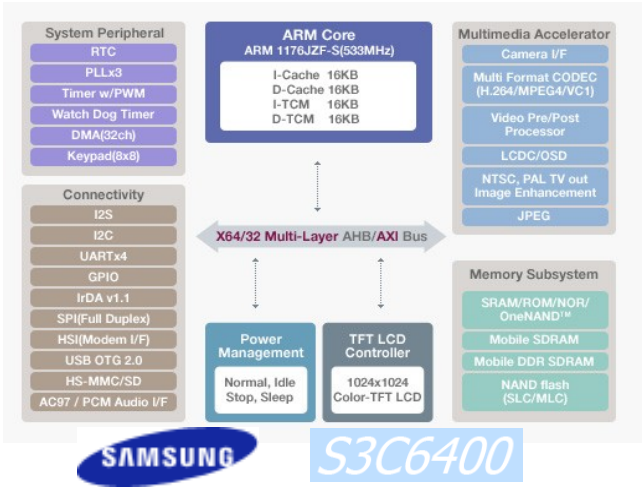
Today's Mobile Multiprocessors

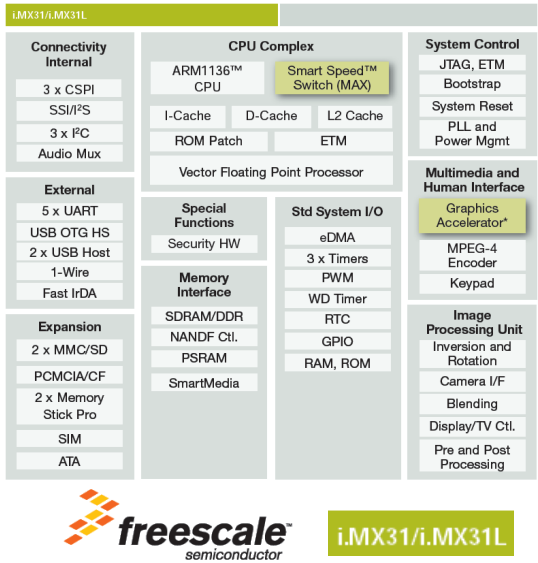






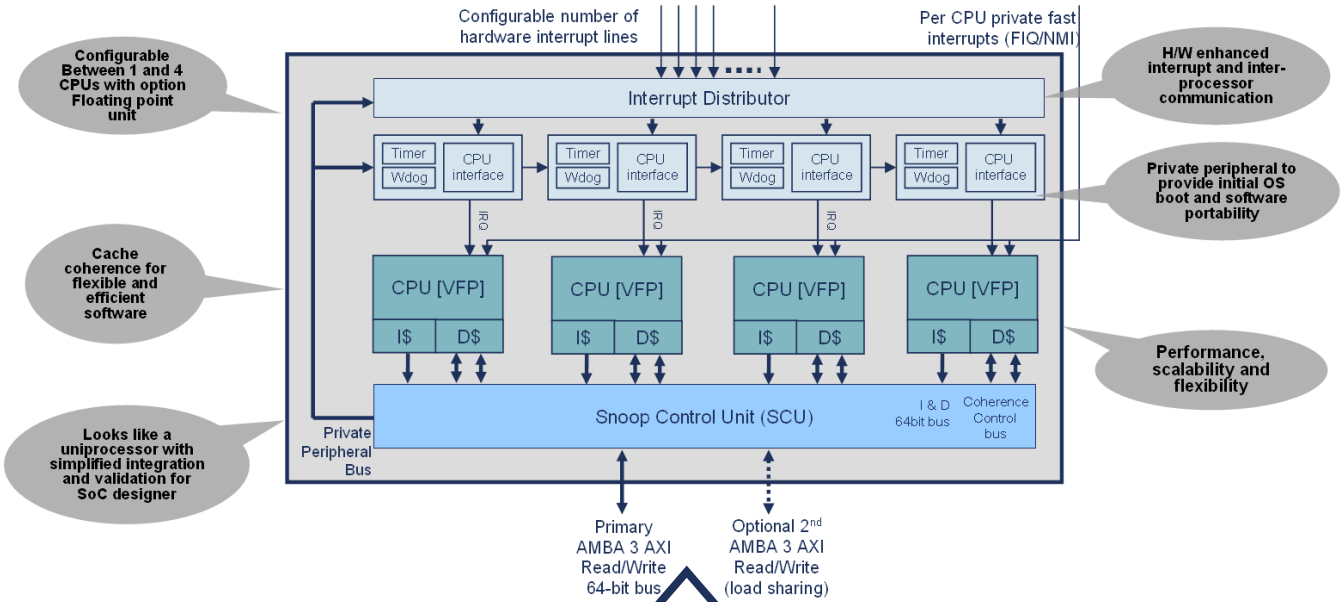




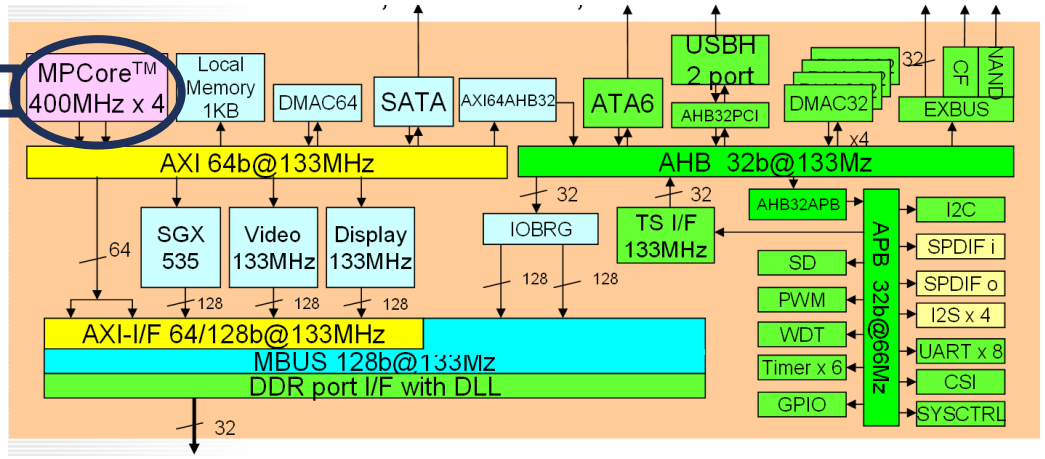


Includes full integration and multiple processors

Example ASSP multicore multiprocessor

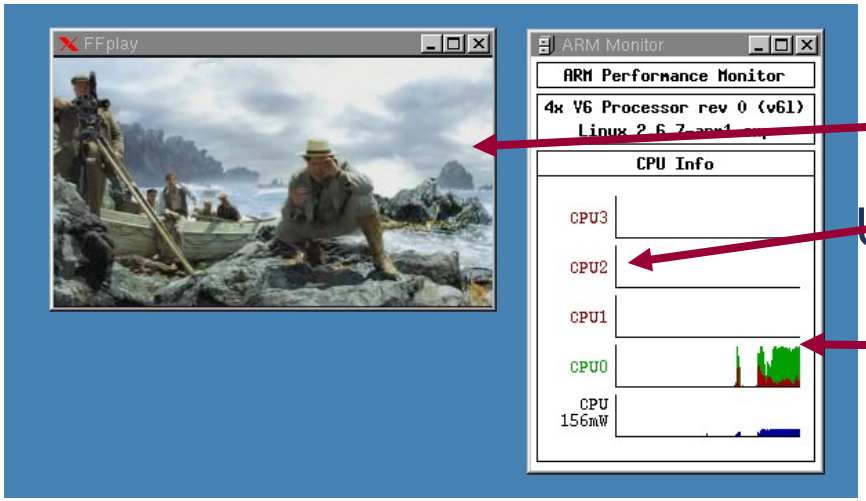


- Utilizing the ARM11 MPCore multicore processor to extend the host processor node (application software domain) to execute across multiple homogeneous cores



NaviEngine®の特徴 NEC

Why bring multicore to the host node?

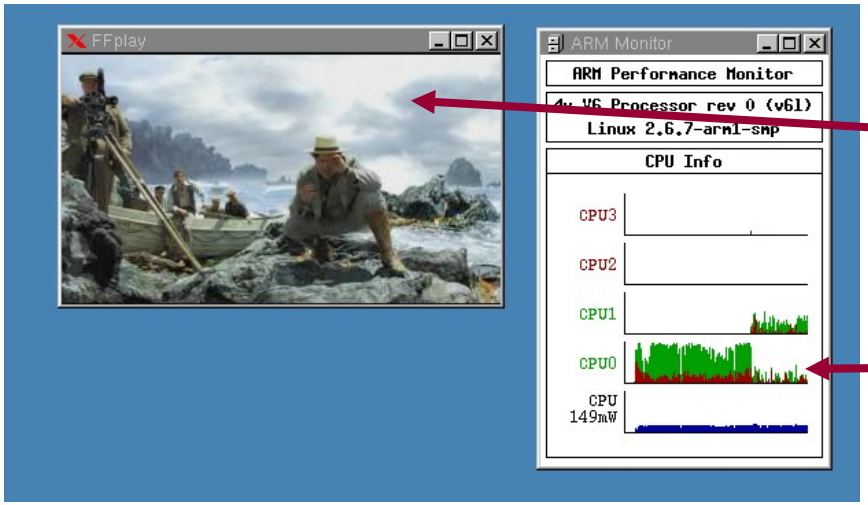


Single CPU

For a given workload requirement

Unused processors are 'turned off'

Single CPU @ 260MHz, **Testchip** consuming ~160mW



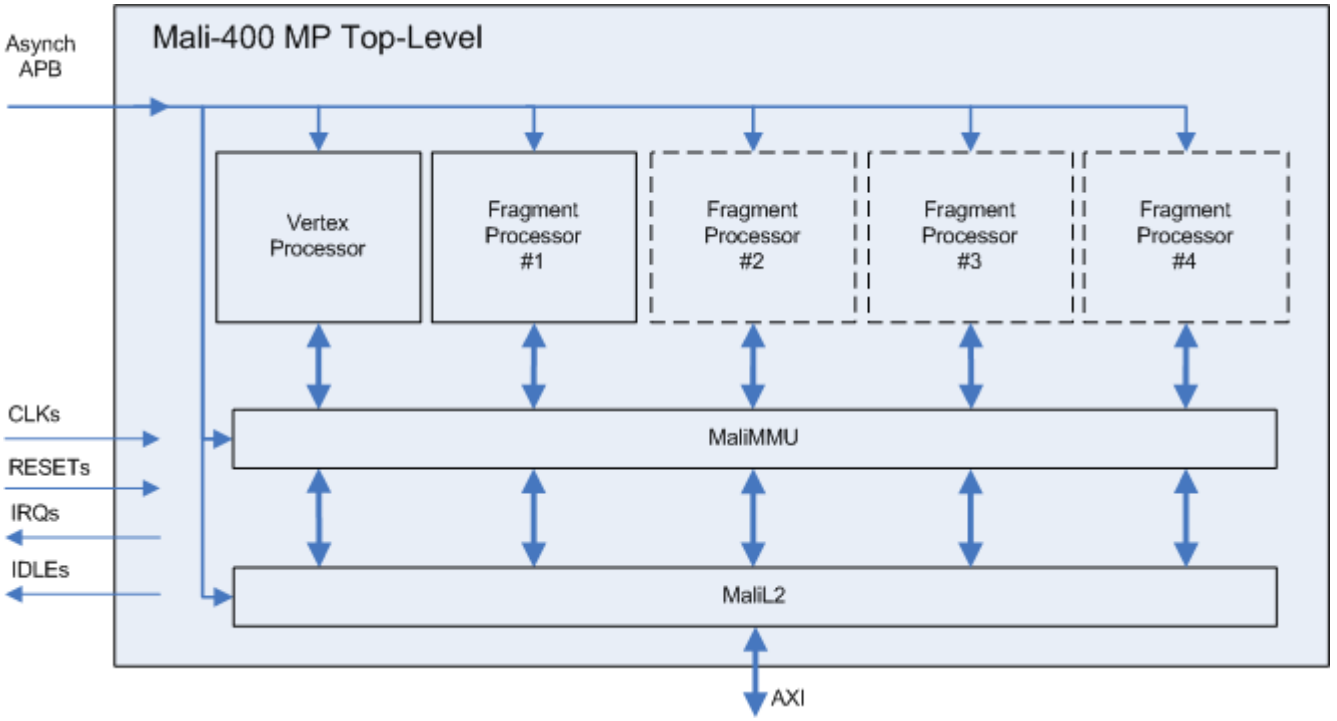
Dual CPU (same MHz, same voltage)

Same workload level leaves headroom on CPUs

Alternatively, up to 50% energy saving potential with voltage and frequency scaling

Multiprocessing offers more performance at lower MHz

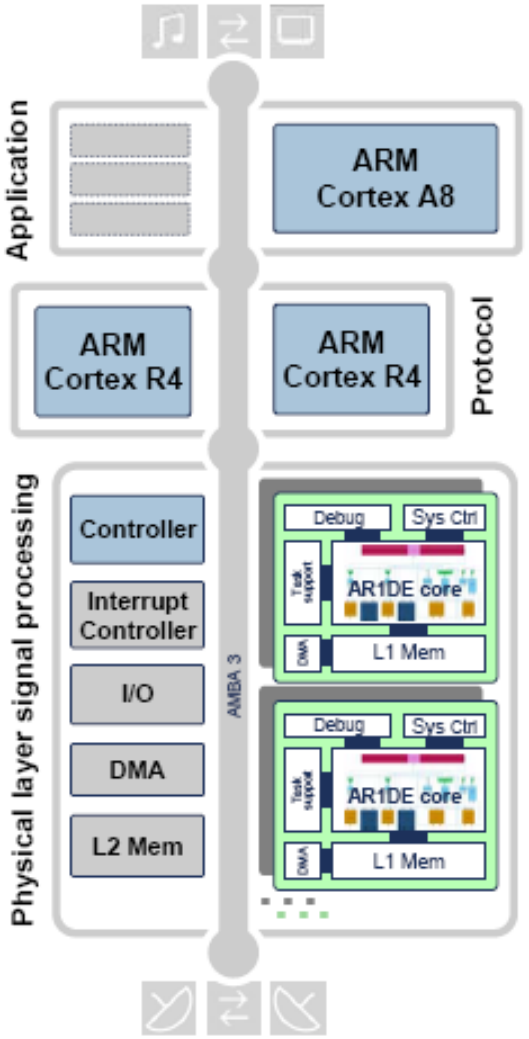
Equally valuable for other SW domains



ARM's Mali 3D processing engines also brings multicore technology for scalable performance and improved energy efficiency

Summary of ARM MPSoC Today

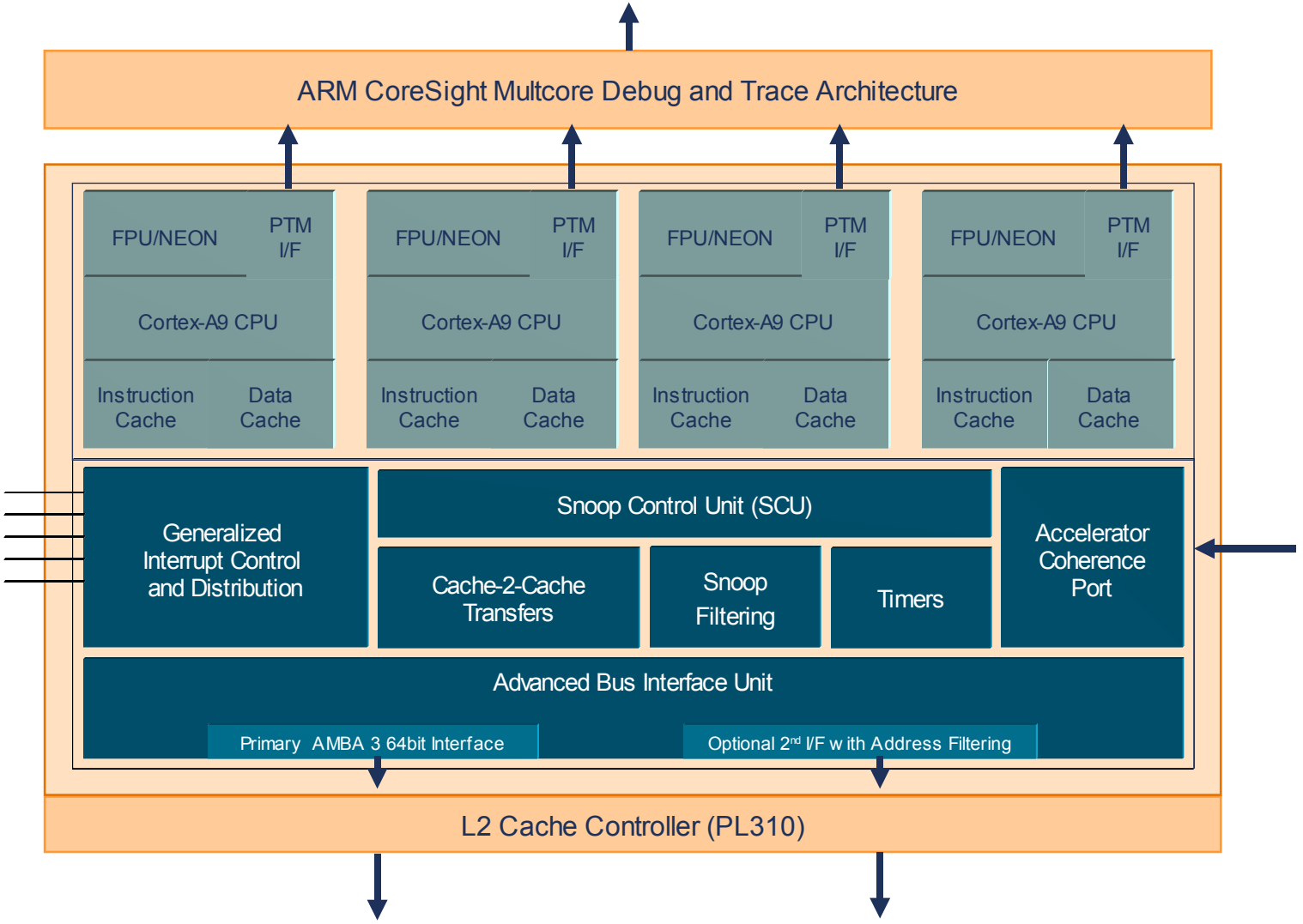
- Various multiprocessor SoC solutions are in the market today
  - Heterogeneous, each processor having an application specific purpose
  - Homogeneous, each processor scaling some defined function
- Just arriving in market are ARM11 MPCore based multicore devices
  - Automotive “general purpose” infotainment
  - Scalable Network device control plane processors
  - Application specific functions within camera, printers partitioned across each processor
- The next-gen Cortex-A9 multicore was announced October 2007 and is bringing new MPSoC structures to SoC design
  - End market products 2009-2010



MPSoC Integration Challenges

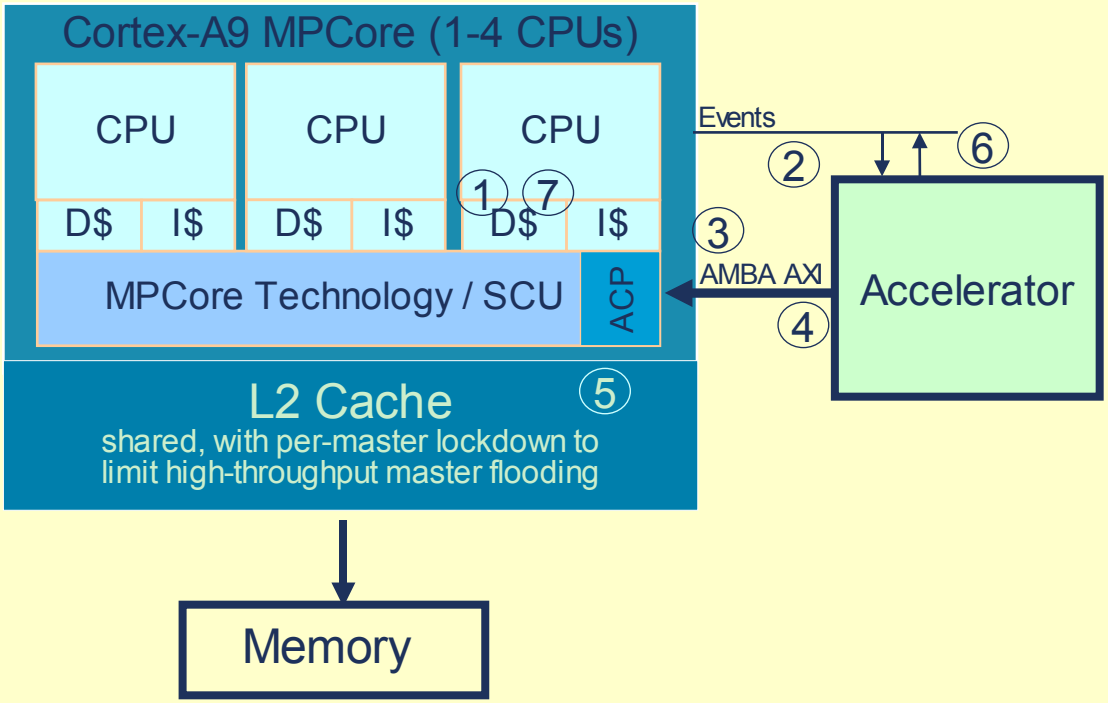
- The complexity (hence size, power and speed) of system interconnects required to support large MPSoC
  - Designs moving toward subsystems utilizing multiple localized interconnects with shared backbone connectivity.
    - Technologies such as NoC appearing in products
  - Management of memory between the growing number of masters
- ARM MPCore technology provides a localized (coherent) interconnect within the multicore macroblock (the SCU)
  - Pre-validated, with same SoC verification complexity of a single core
- The growth of both specific subsystems, and the total number of subsystem (hence cost) along with SW legacy are key

Cortex-A9 MPCore Processor Structure



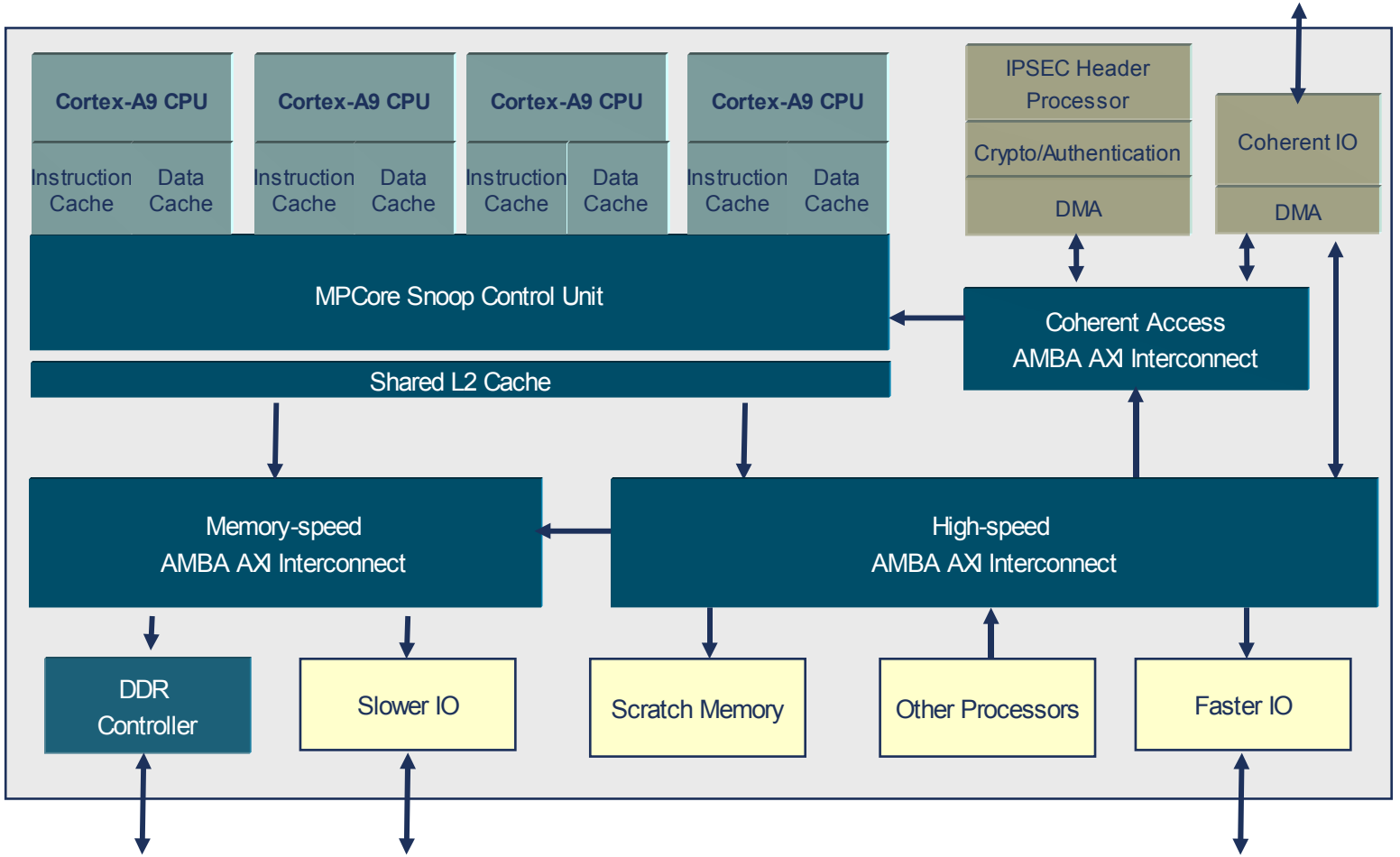
Enhanced Accelerator SoC Integration

- 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

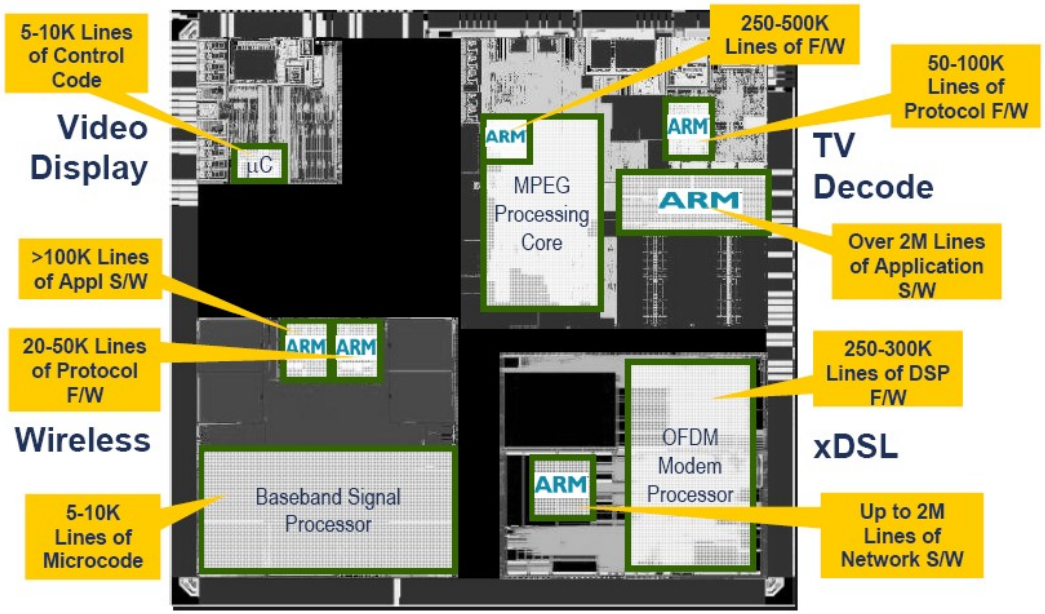
Example next-gen multicore MPSoC



(expect to see such structures in devices by 2009-2012)

Observation on current approach to SW

- Most designs have major legacy / code reuse demands
  - Especially in open platforms
    - …but thankfully mostly use a pre-emptive multitasking OS today
  - Deeply embedded and fix function devices are more open to change
    - …and due to the typical priority based scheduling used, must change to partition code over multiple processors



- Current MPSoC typically using explicit communication schemes between otherwise independent processor specific domains of software
  - With most developers thinking this is the only way to program for MP !!

Current SW Techniques

- Multiprocessors
  - Each processor programmed using a myriad of different single thread paradigms with programmer statically assigning tasks to processors
  - Typically interrupt communication mailbox via scratch or off-chip RAM
    - Occasionally core supports direct FIFO messaging path
  - Increasing complexity associated with programming a growing number of processors while still maintaining effective silicon utilization

- Multicores
  - A number of designs are directly migrating existing SW framework from previous generation multiprocessing solution
    - But enabling coherence for enhanced L1 level communication
  - Various multicore aware OS/RTOS providing various levels of AMP vs. SMP scheduling, (AMP/Bounded/Blended/SMP/etc)

Vision for the future

- New tools to help with the manual partitioning and communication of tasks in heterogeneous multiprocessors
  - Eg. More of the same, but a little easier

- Consolidation of various independent multiprocessor software domains onto fewer multicores (but keeping their isolation)
  - Easier and cheaper to build than many-processor multiprocessors
  - Potential for virtualization to support more virtual processors than physically supported by the multicore

- Gradual adoption of coherent multicores for a limited number of software domains that utilize dynamic migration of tasks
  - Driven by open platform requirements and the need to conserve power
  - Programming limited to implicit concurrency and task-level threads

Mapping of Applications to MPSoC

- Please, help address one of my biggest MPSoC challenges…
  - We need clarity! clarity of definition, clarity over the problem space, and clarity of any proposed solution
  - I doubt there is a single solution to “Mapping of applications to MPSoC”

Multiprocessing has its problems,

Multicore has its problems,

MPSoC using both have another set of problems,

The biggest problem is almost always software investment legacy

Followed by the problem of designers being unable to classify their problems and listing all the problems anyone has ever said of multi-anything a show stopper