Multi-core and System Coherence Design Challenges

Agenda

- Cortex A15
- AXI Coherency Extensions (ACE)
- Cache Coherent Interconnect (CCI-400)
- big.LITTLE
- Q & A

Cortex-A15: Next Generation Leadership

Target Markets

- High-end wireless and smartphone platforms
- tablet, large-screen mobile and beyond
- Consumer electronics and auto-infotainment
- Hand-held and console gaming
- Networking, server, enterprise applications

Cortex-A class multi-processor

- 40bit physical addressing (1TB)
- Full hardware virtualization
- AMBA 4 system coherency
- ECC and parity protection for all SRAMs

Advanced power management

- Fine-grain pipeline shutdown
- Aggressive L2 power reduction capability
- Fast state save and restore

Significant performance advancement

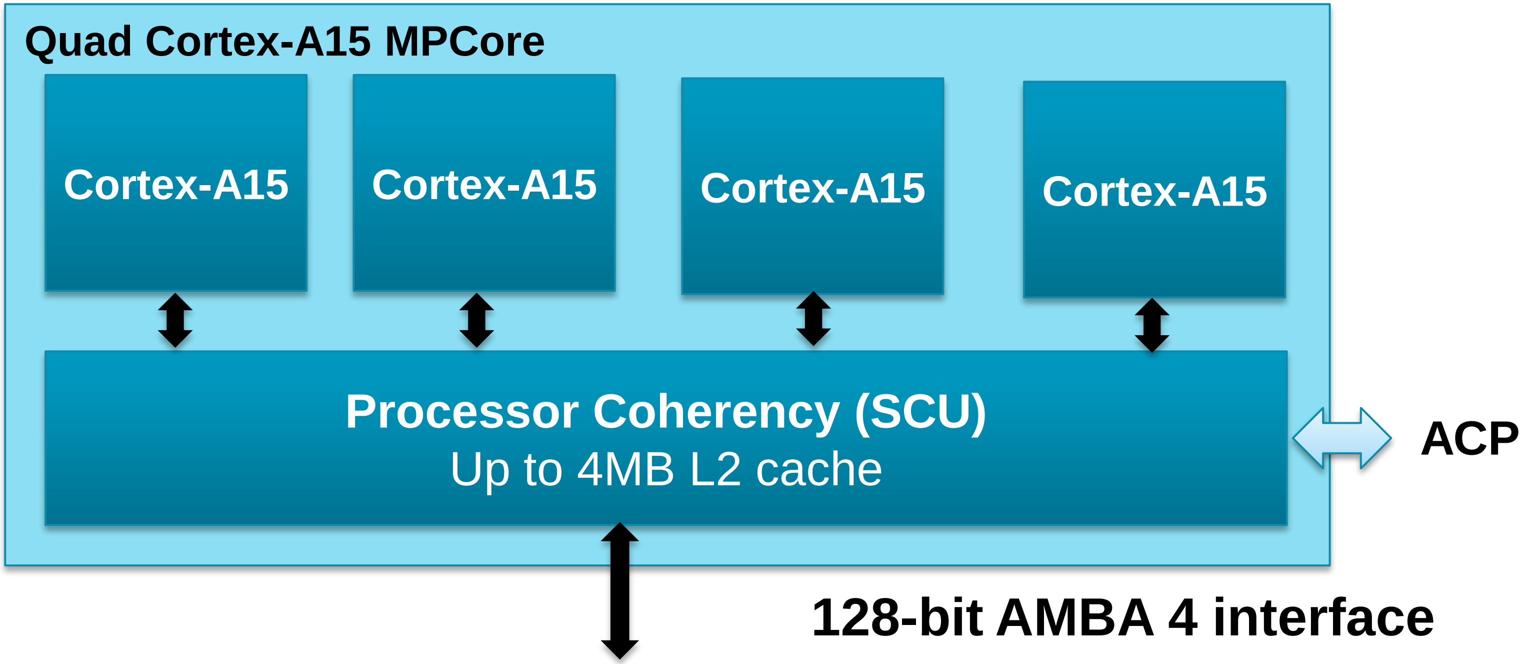
- Improved single-thread and MP performance

Targets 1.5 GHz in 32/28 nm LP process

Targets 2.5 GHz in 32/28 nm G/HP process

Cortex-A15 Multiprocessing

- ARM introduced up to quad MP in 2004 with ARM11 MPCore
- Multiple MP solutions: Cortex-A9, Cortex-A5, Cortex-A15
- Cortex-A15 includes
  - Integrated L2 cache with SCU functionality
  - 128-bit AMBA 4 interface with coherency extensions



128-bit AMBA 4 interface

Scaling Beyond Four Cores

Introducing AMBA 4 coherency extensions

- Coherency, Barriers and Memory management

Software implications

- Hardware managed coherency simplifies software
- Processor spends less time managing caches

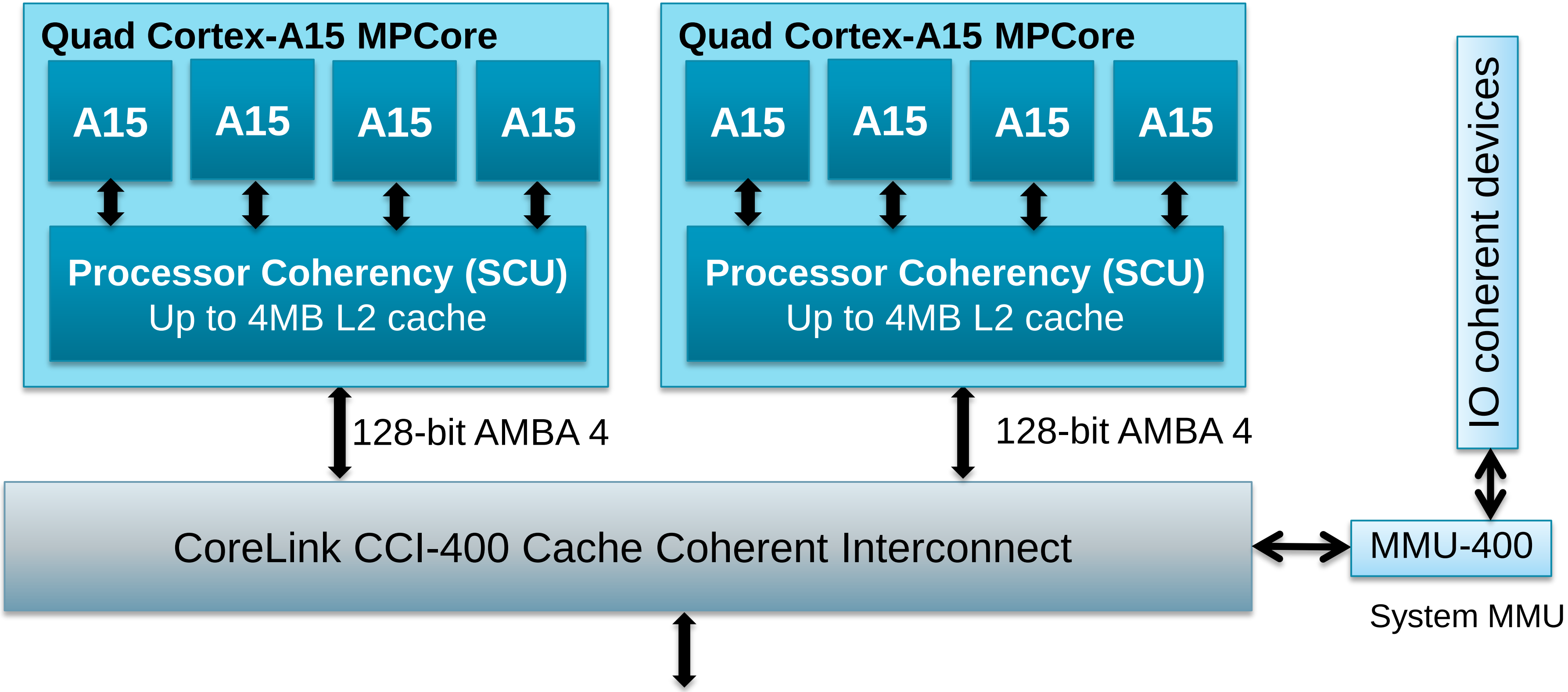
Coherency types

- I/O coherency
  - Devices snoop into processor caches (but processors do not snoop into the device)
- Full cache coherency
  - Cache snooping in both directions

Cortex-A15 System Scalability

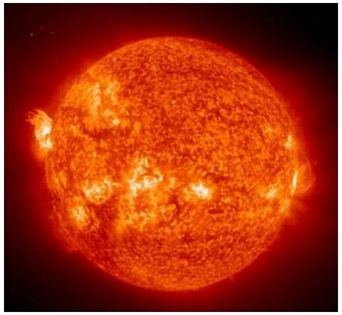
Introducing CCI-400 Cache Coherent Interconnect

- Processor to Processor Coherency and I/O coherency
- Memory and synchronization barriers
- TLB and cache maintenance



Memory Error Detection/Correction

Error Correction Control on all software writable memories

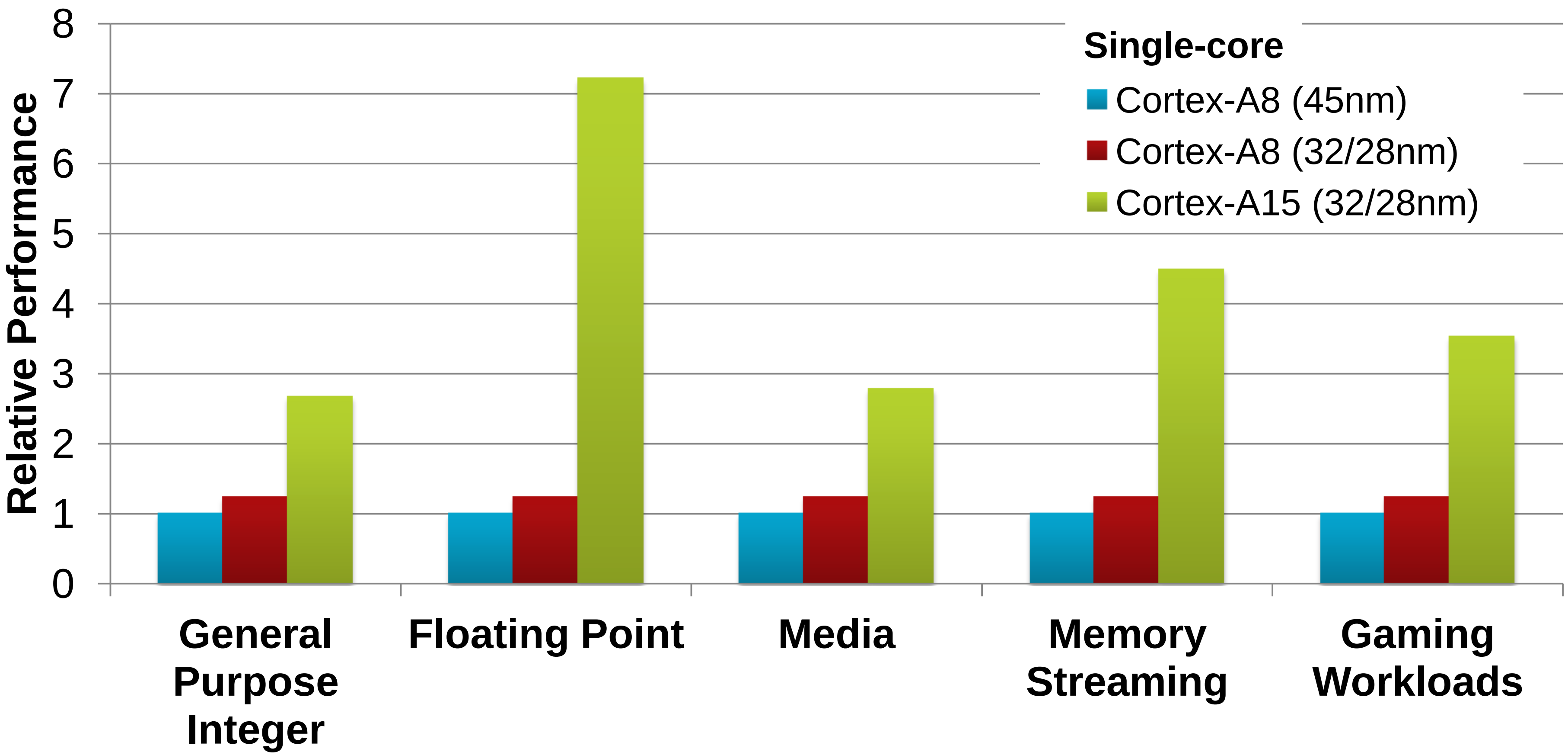


- Single error correct, 2 error detect
- Multi-bit errors rare
- Protects 32 bits for L1, 64 bits for L2
- Error logging at each level of memory
- Leveraging out-of-order mechanisms for no performance impact
- Icache and TLB RAMs protected with precise parity (no fault required)

Primarily motivated by enterprise markets

- Soft errors predominantly caused by electrical disturbances
- Memory errors proportional to RAM and duration of operation
- Servers: MBs of cache, GBs of RAM, 24/7 operation
    - Highly probability of error eventually happening
- If not corrected, eventually causes computer to crash and affect network

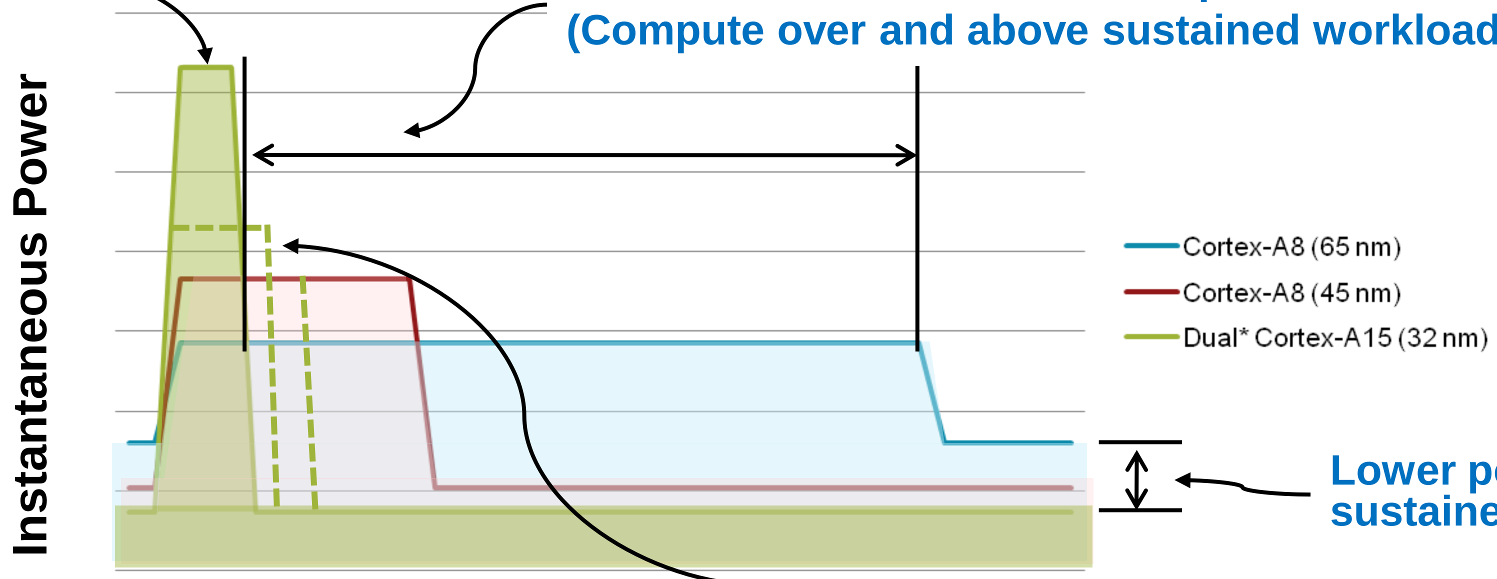
High-end Single Thread Performance



- **Both processors using 32K L1 and 1MB L2 Caches, common memory system**
- **Cortex-A8 and Cortex-A15 using 128-bit AXI bus master**

Note: Benchmarks are averaged across multiple sets of benchmarks with a common real memory system attached
Cortex-A8 and Cortex-A15 estimated on 32/28nm.

Performance and Energy Comparison



A15 dual-core power at peak

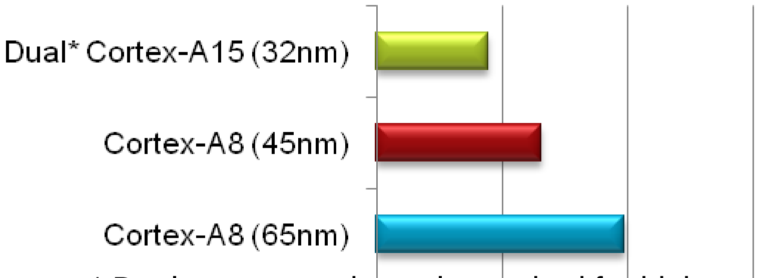
Much faster execution time for performance critical task (Compute over and above sustained workload)

Lower power on sustained workload

Performance at tighter thermal constraints

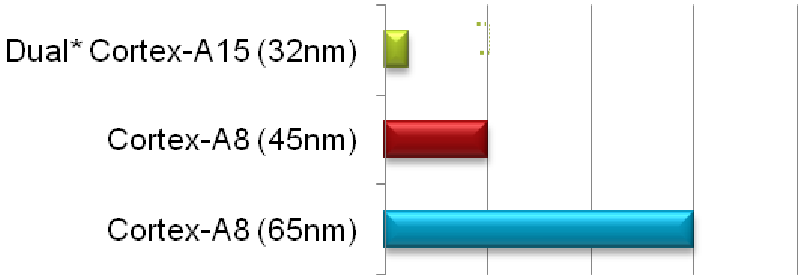
Energy consumed

(lower is better)



Execution Time for critical task

(lower is better)



* Dual-core operation only required for high-end timing critical tasks. Single-core for sustained operation

Where We Started: Early Goals

Large performance boost over A9 in general purpose code

- From combination frequency + IPC
- Performance is more than just integer
  - Memory system performance critical in larger applications
  - Floating point/NEON for multimedia
  - MP for high performance scalability

Straightforward design flow

- Supports fully synthesized design flow with compiled RAM instances
- Further optimization possible through advanced implementation
  - Power/area savings

Minimize power/area cost for achieving performance target

Where to Find Performance: Frequency

Give RAMs as much time as possible

- Majority of cycle dedicated to RAM for access
- Make positive edge based to ease implementation

Balance timing of critical “loops” that dictate maximum frequency

- Microarchitecture loop:
  - Key function designed to complete in a cycle (or a set of cycles)
    - cannot be further pipelined (with high performance)
- Some example loops:
  - Register Rename allocation and table update
  - Result data and tag forwarding (ALU->ALU, Load->ALU)
  - Instruction Issue decision
  - Branch prediction determination

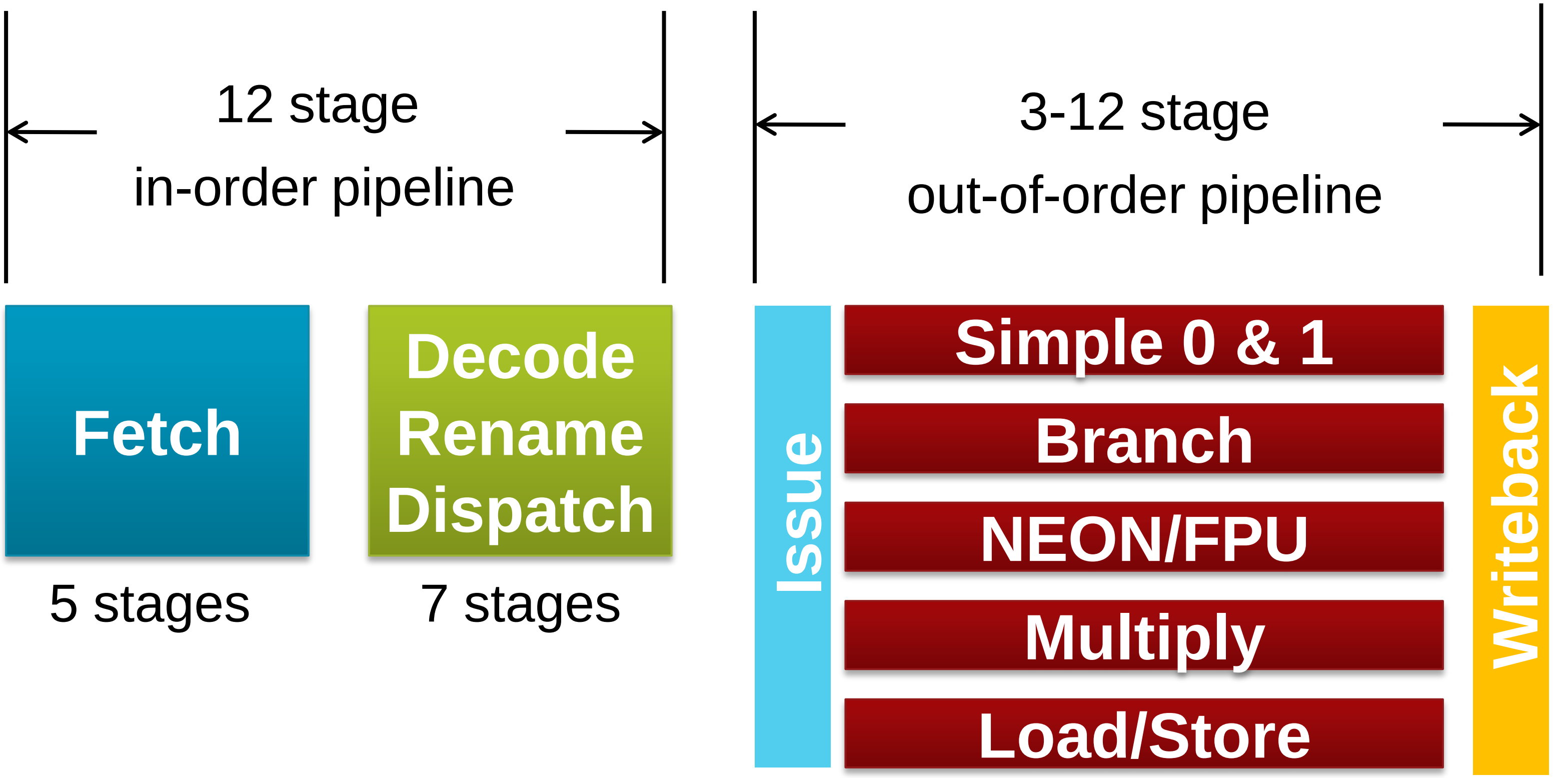
Feasibility work showed critical loops balancing at about 15-16 gates/clk

Where to Find Performance: IPC

- Wider pipelines for higher instruction throughput
- Larger instruction window for out-of-order execution
- More instruction types can execute out-of-order
- Improved memory system performance

Cortex-A15 Pipeline Overview

- 12 stage in-order fetch and decode
- 3-12 stage out-of-order execute



Improving Branch Prediction

Similar predictor style to Cortex-A8 and Cortex-A9:

- Large target buffer for fast turn around on address
- Global history buffer for taken/not taken decision

Global history buffer enhancements

- 3 arrays: Taken array, Not taken array, and Selector

Indirect predictor

- 256 entry BTB indexed by XOR of history and address
- Multiple Target addresses allowed per address

Out-of-order branch resolution:

- Reduces the mispredict penalty
- Requires special handling in return stack

Fetch Bandwidth: More Details

Increased fetch from 64-bit to 128-bit

- Full support for unaligned fetch address
  - Enables more efficient use of memory bandwidth
  - Only critical words of cache line allocated

Addition of microBTB

- Reduces bubble on taken branches
- 64 entry target buffer for fast turn around prediction
- Fully associative structure
- Caches taken branches only
- Overruled by main predictor when they disagree

Out-of-Order Execution Basics

Out-of-Order instruction execution is done to increase available instruction parallelism

The programmer's view of in-order execution must be maintained

- Mechanisms for proper handling of data and control hazards
  - WAR and WAW hazards removed by **register renaming**
  - **Commit queue** used to ensure state is retired non-speculatively
- Early and late stages of pipeline are still executed in-order
- Execution clusters operate out-of-order
  - Instructions issue when all required source operands are available

Register Renaming

Two main components to register renaming

- Register rename tables
  - Provides current mapping from architected registers to result queue entries
  - Two tables: one each for ARM and Extended (NEON) registers
- Result queue
  - Queue of renamed register results pending update to the register file
  - Shared for both ARM and Extended register results

Execution Clusters

- **Simple cluster**
  - Single cycle integer operations
  - 2 ALUs, 2 shifters (in parallel, includes v6-SIMD)
- **Complex cluster**
  - All NEON and Floating Point data processing operations
  - Pipelines are of varying length and asymmetric functions
  - Capable of quad-FMAC operation
- **Branch cluster**
  - All operations that have the PC as a destination
- **Multiply and Divide cluster**
  - All ARM multiply and Integer divide operations
- **Load/Store cluster**
  - All Load/Store, data transfers and cache maintenance operations
  - Partially out-of-order, 1 Load and 1 Store executed per cycle
  - Load cannot bypass a Store, Store cannot bypass a Store

Increasing Out-of-Order Execution

Out-of-order execution improves performance by executing past hazards

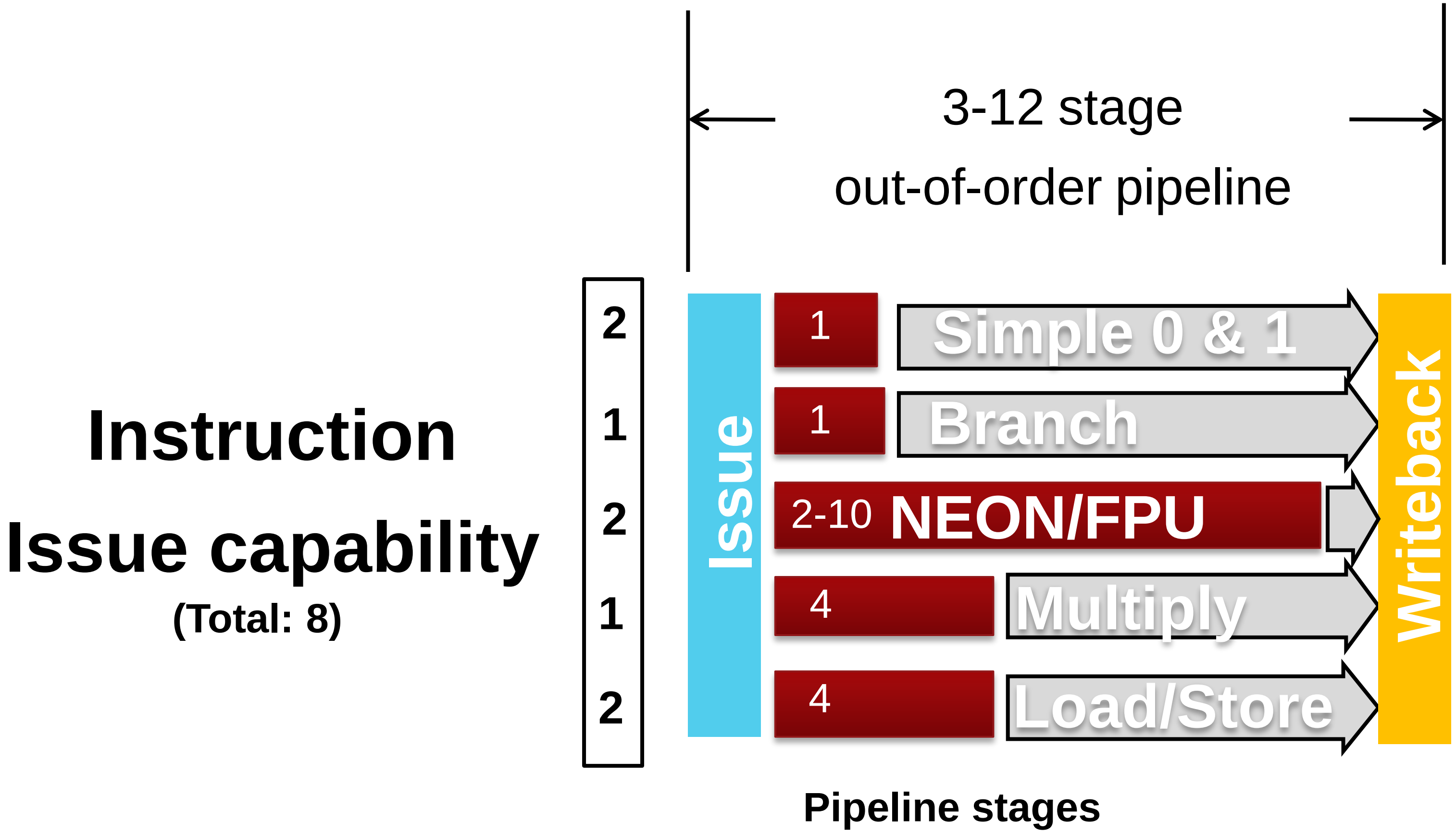
- Effectiveness limited by how far you look ahead
  - Window size of 40+ operations required for Cortex-A15 performance targets
- Issue queue size often frequency limited to 8 entries

Solution: multiple smaller issue queues

- Execution broken down to multiple clusters defined by instruction type
- Instructions dispatched 3 per cycle to the appropriate issue queue
- Issue queues each scanned in parallel

Cortex-A15 Execution Clusters

- Each cluster can have multiple pipelines
- Clusters have separate/independent issuing capability



Floating Point and NEON Performance

Dual issue queues of 8 entries each

- Can execute two operations per cycle
- Includes support for quad FMAC per cycle

Fully integrated into main Cortex-A15 pipeline

- Decoding done upfront with other instruction types
- Shared pipeline mechanisms
- Reduces area consumed and improves interworking

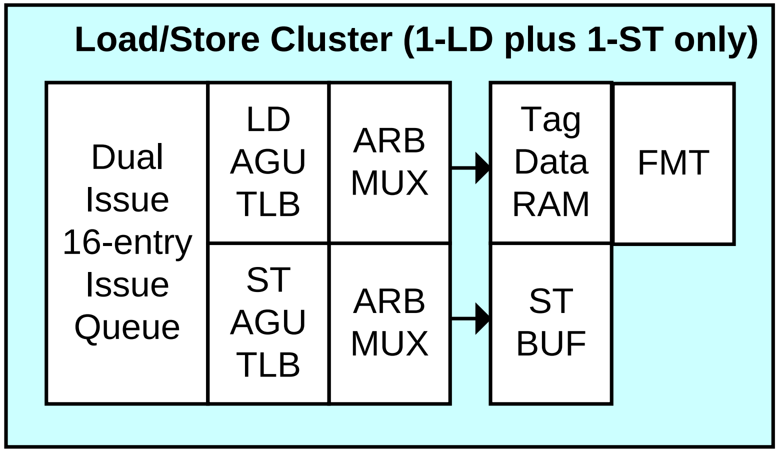
Specific challenges for Out-of-order VFP/Neon

- Variable length execution pipelines
- Late accumulator source operand for MAC operations

Load/Store Cluster

16 entry issue queue for loads and stores

- Common queue for ARM and NEON/memory operations
- Loads issue out-of-order but cannot bypass stores
- Stores issue in order, **but only require address sources to issue**



Load/Store Cluster (1-LD plus 1-ST only)
Dual Issue 16-entry Issue Queue
LD AGU TLB
ARB MUX
Tag Data RAM
FMT
ST AGU TLB
ARB MUX
ST BUF

4 stage load pipeline

- 1st: Combined AGU/TLB structure lookup
- 2nd: Address setup to Tag and data arrays
- 3rd: Data/Tag access cycle
- 4th: Data selection, formatting, and forwarding

Store operations are AGU/TLB look up only on first pass

- Update store buffer after PA is obtained
- Arbitrate for Tag RAM access
- Update merge buffer when non-speculative
- Arbitrate for Data RAM access from merge buffer

The Level 2 Memory System

Cache characteristics

- 16 way cache with sequential TAG and Data RAM access
- Supports sizes of 512kB to 4MB
- Programmable RAM latencies

MP support

- 4 independent Tag banks handle multiple requests in parallel
- Integrated Snoop Control Unit into L2 pipeline
- Direct data transfer line migration supported from cpu to cpu

External bus interfaces

- Full AMBA4 system coherency support on 128-bit master interface
- 64/128 bit AXI3 slave interface for ACP

Other key features

- Full ECC capability
- Automatic data prefetching into L2 cache for load streaming

Other Key Cortex-A15 Design Features

Supporting fast state save for power down

- Fast cache maintenance operations
- Fast SPR writes: all register state local

Dedicated TLB and table walk machine per cpu

- 4-way 512 entry per cpu
- Includes full table walk machine and cache walking structures

Active power management

- 32 entry loop buffer
- Completely disables Fetch and part of the Decode stages of pipeline

ECC support in software writeable RAMs, Parity in read only RAMs

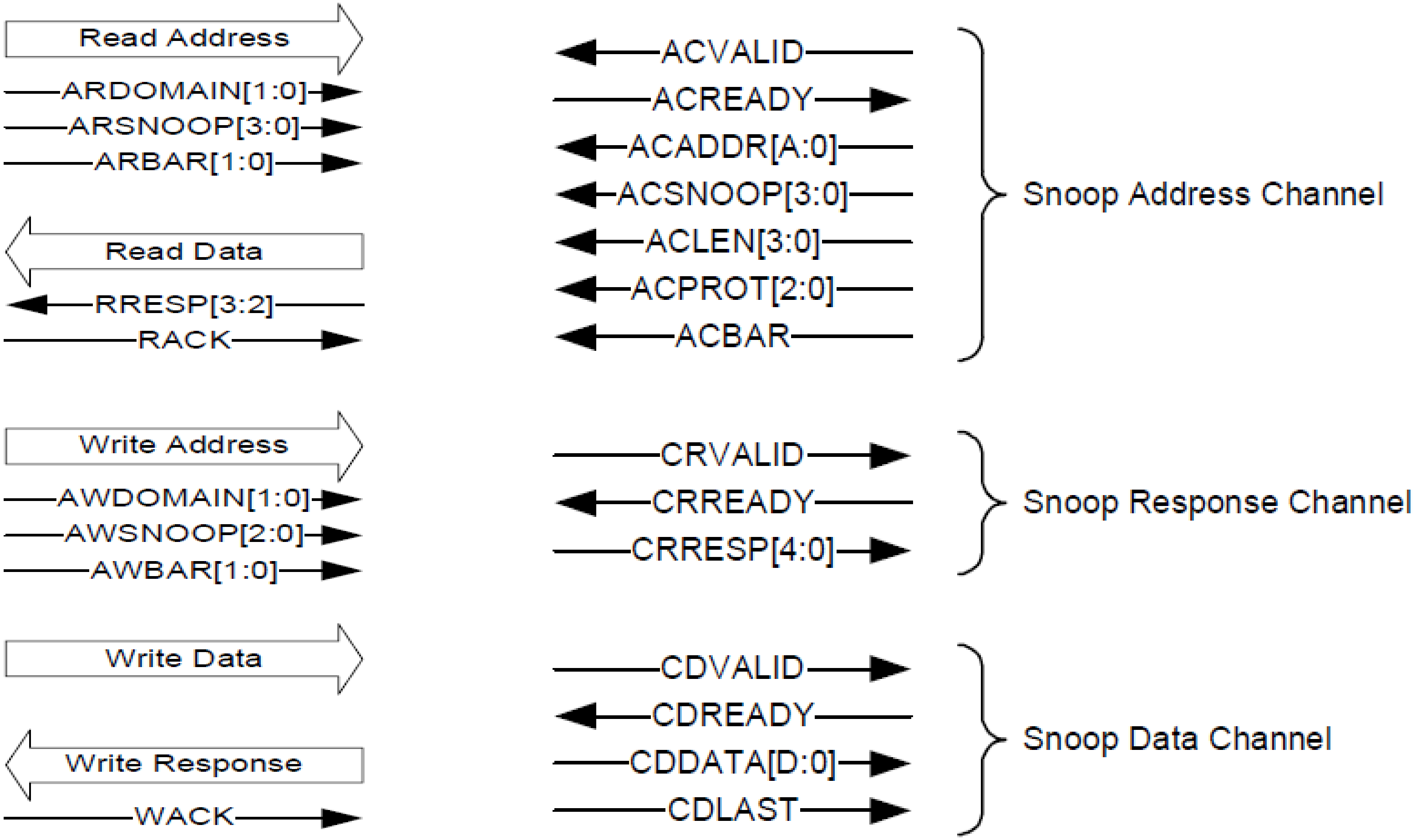
- Supports logging of error location and frequency

Agenda

- Cortex A15
- AXI Coherency Extensions (ACE)
- Cache Coherent Interconnect (CCI-400)
- big.LITTLE
- Q & A

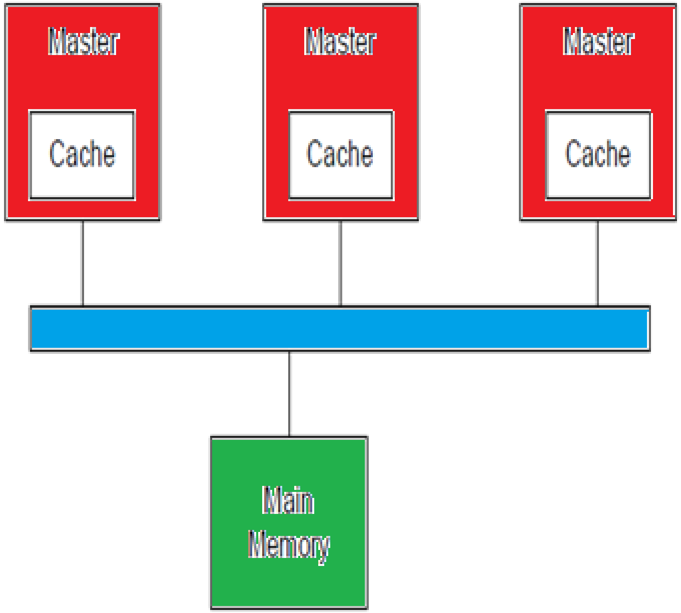
AXI Coherency Extensions

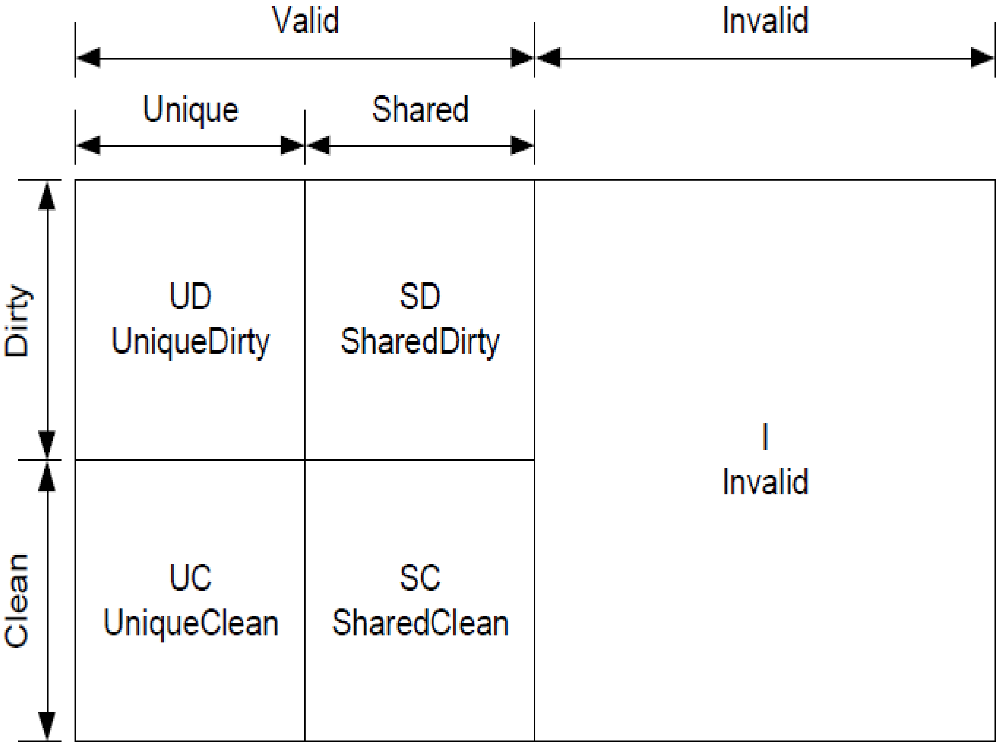
- Extends upon existing non-coherent interface by adding support for new transactions and snoop channel



Coherency Model

- Multiple masters with local caches, protocol supports up to 5 state cache model

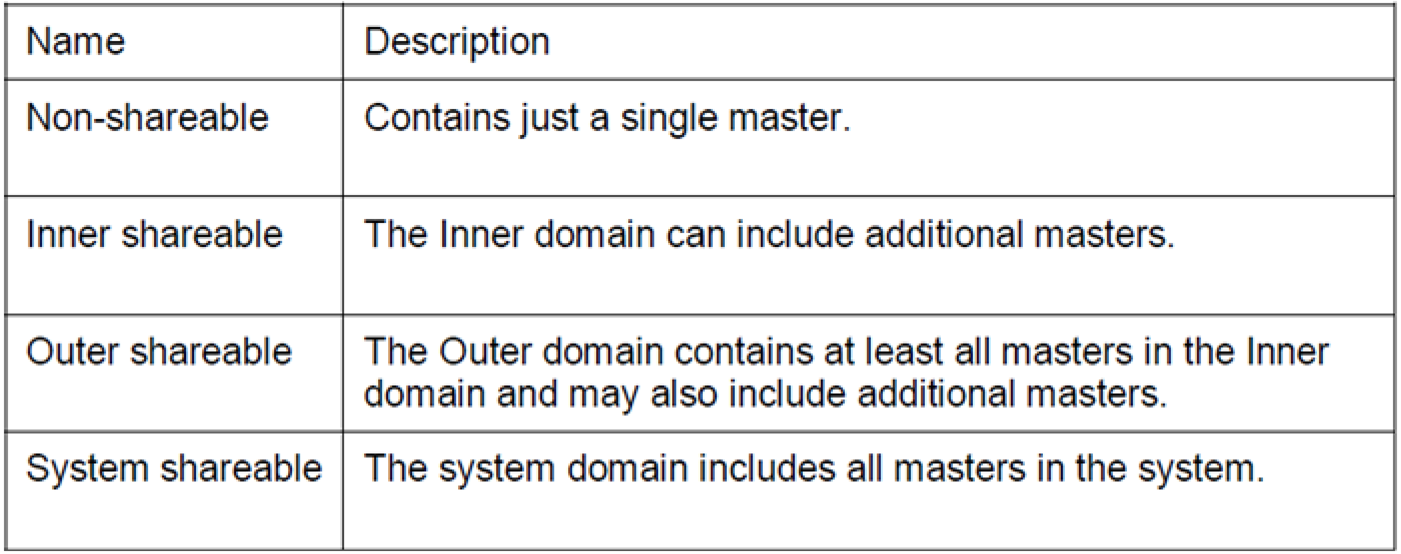




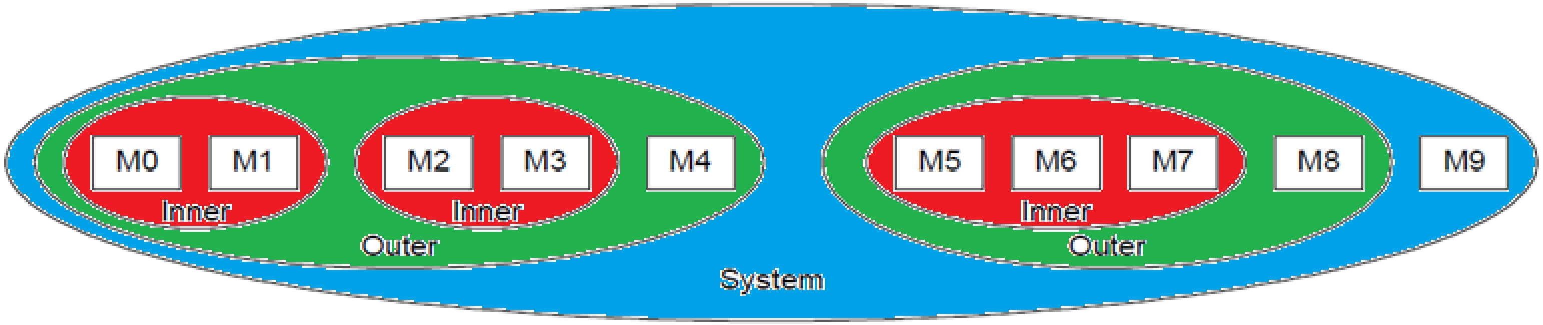
	Valid		Invalid
	Unique	Shared	
Dirty	UD UniqueDirty	SD SharedDirty	I Invalid
Clean	UC UniqueClean	SC SharedClean	

Domains

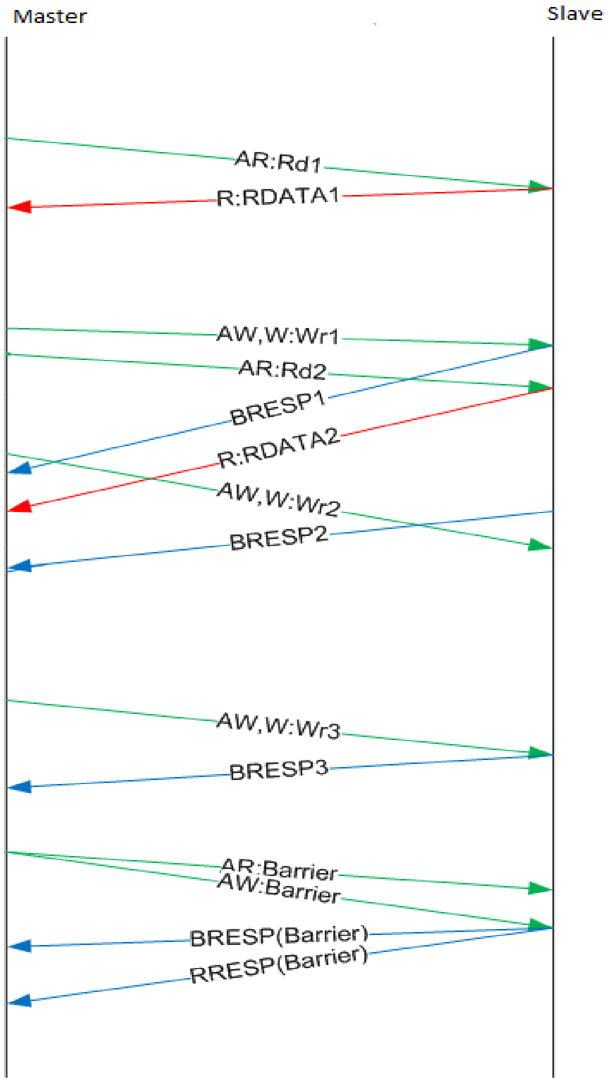
- Allows for course-grained filtering of snoops in a system with partitioned memory

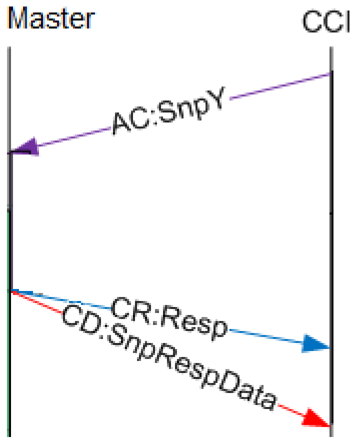


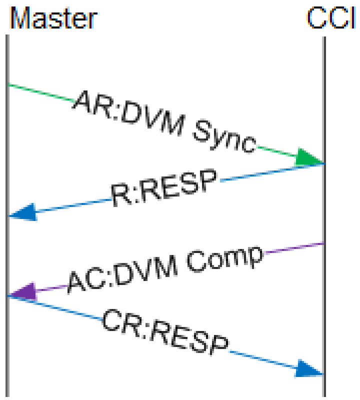
Name	Description
Non-shareable	Contains just a single master.
Inner shareable	The Inner domain can include additional masters.
Outer shareable	The Outer domain contains at least all masters in the Inner domain and may also include additional masters.
System shareable	The system domain includes all masters in the system.



Example Transactions



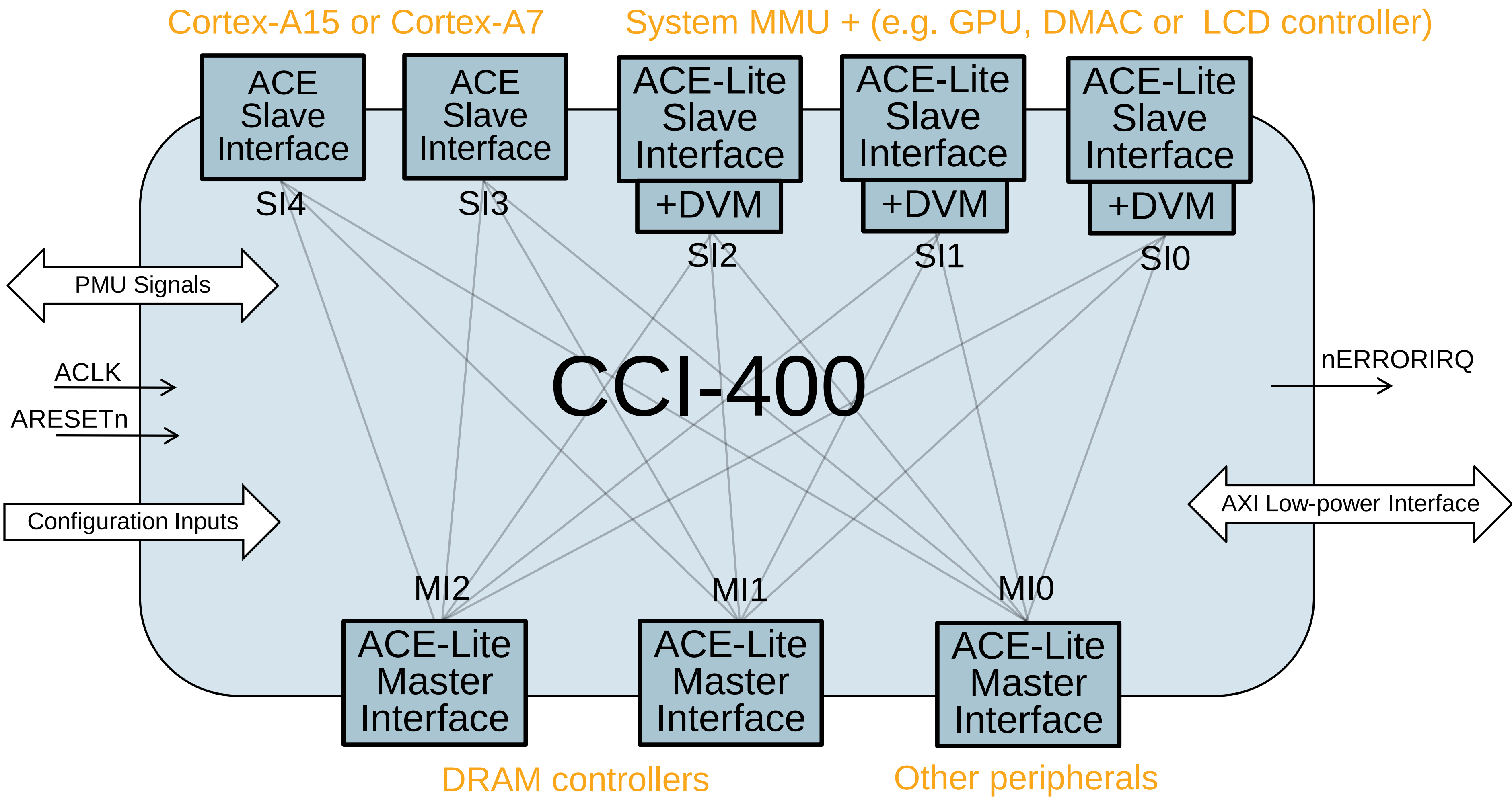




Agenda

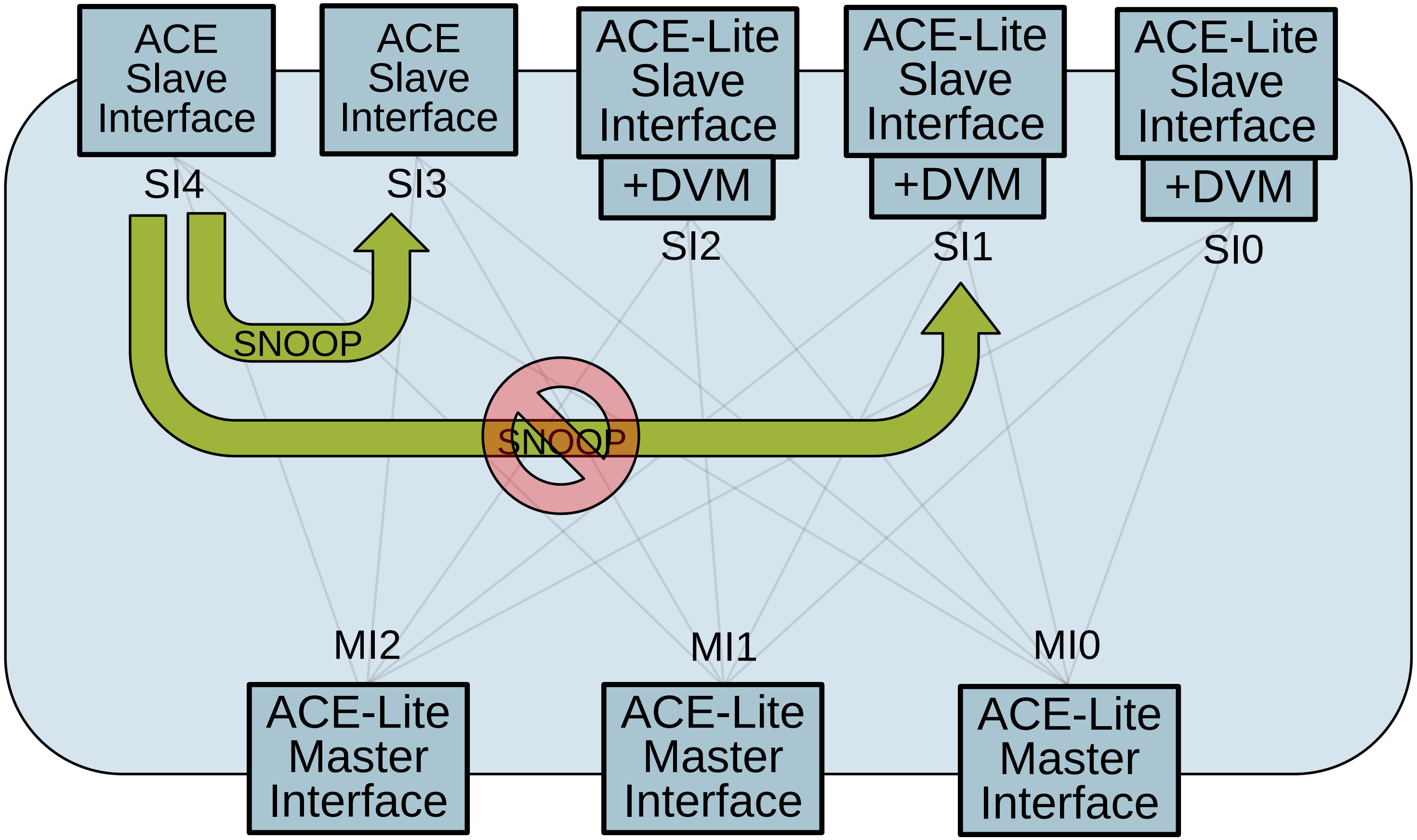
- Cortex A15
- AXI Coherency Extensions (ACE)
- Cache Coherent Interconnect (CCI-400)
- big.LITTLE
- Q & A

Block diagram



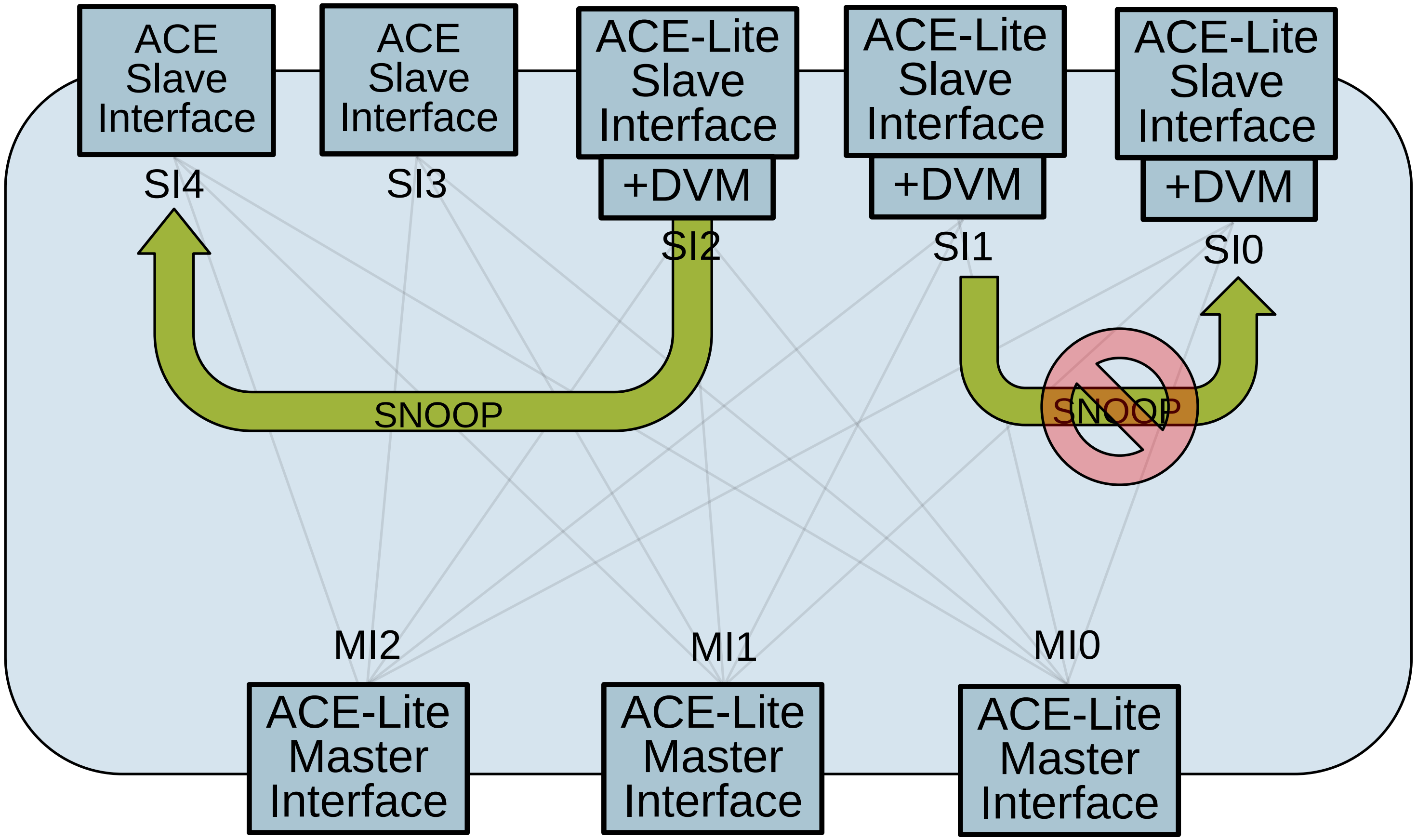
ACE master snooping transactions

- **Transactions from ACE masters can snoop other ACE masters' caches**
- **Transactions from ACE masters cannot snoop ACE-Lite masters' caches**



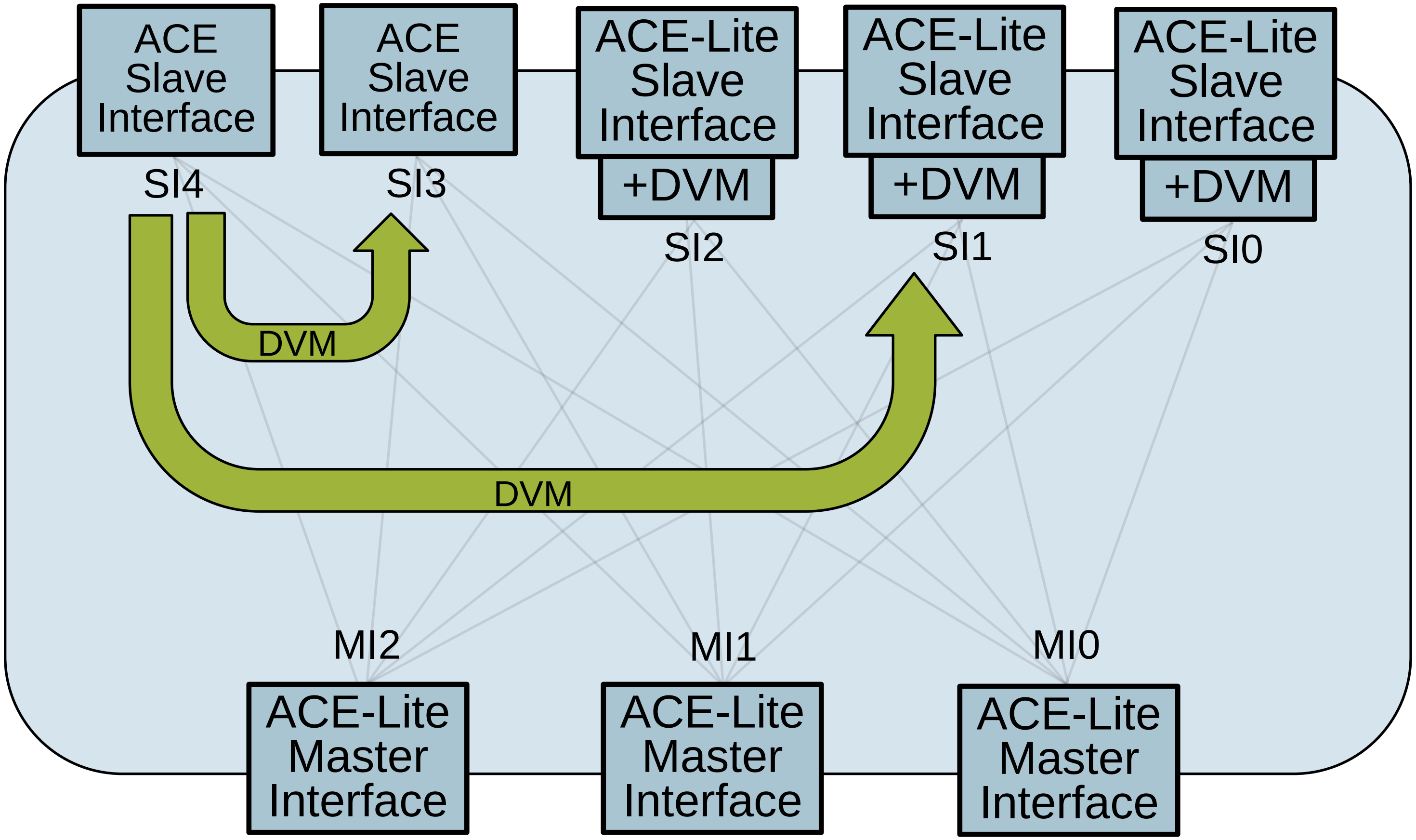
ACE-Lite master snooping transactions

- **Transactions from ACE-Lite masters can snoop other ACE masters' caches**
- **Transactions from ACE-Lite masters cannot snoop ACE-Lite masters' caches**



Distributed Virtual Memory messages

- ACE masters can accept DVM messages, per the ACE protocol
- CCI-400 DVM extensions allow ACE-Lite masters to receive DVMs messages
- ACE-Lite masters cannot generate DVM messages

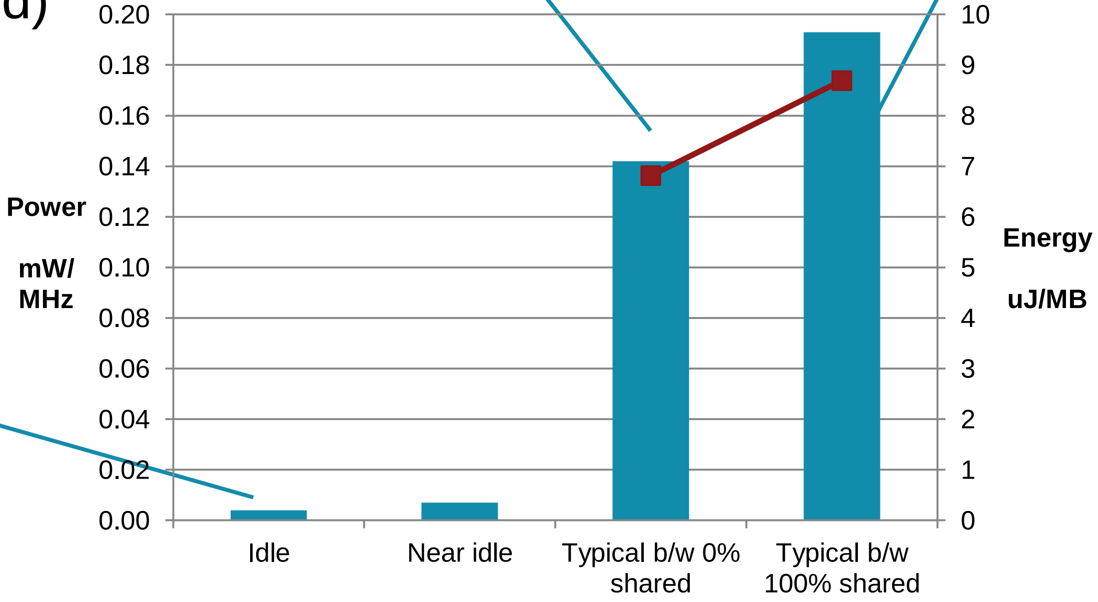


Power Estimation

- Power estimated using Synopsys Primetime-PX using netlist simulations under different traffic scenarios
- Conditions:
- Active power shown below:
  - as mW/MHz (blue bars)
  - as uJ/MB (red)

With typical high-end scenario, 100% shared traffic at 533MHz, CCI-400 consumes ~100mW

Energy per MB transferred is approx 8 uJ/MB.
e.g. transferring a total of 10GB in 1s would consume ~80mW



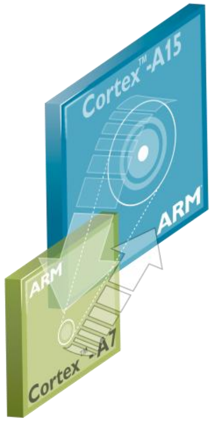
Modest *idle* and *near idle* power demonstrates effective clock gating

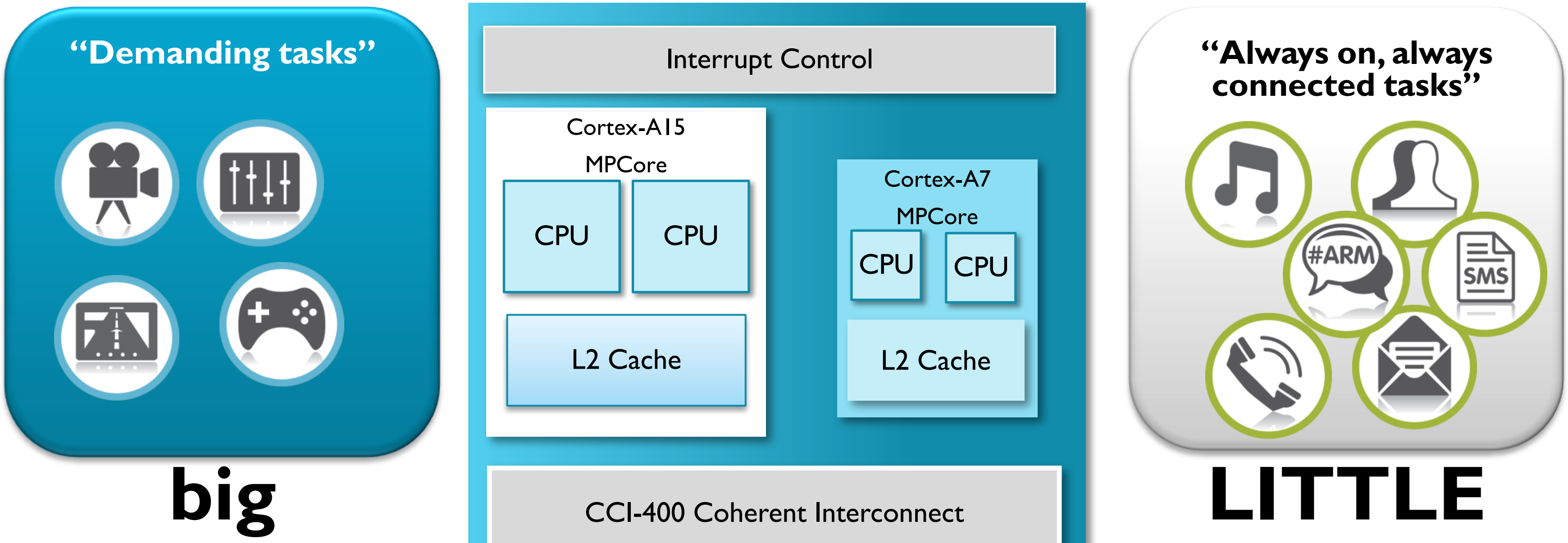
Agenda

- Cortex A15
- AXI Coherency Extensions (ACE)
- Cache Coherent Interconnect (CCI-400)
- big.LITTLE
- Q & A

Introducing big.LITTLE Processing

- Uses the right processor for the right job
- Up to 70% energy savings on common workloads
- Flexible and transparent to apps – seamless software handover
- Best of both worlds solution for high performance and low power





"Demanding tasks"

big

Interrupt Control

Cortex-A15 MPCore

CPU CPU

L2 Cache

Cortex-A7 MPCore

CPU CPU

L2 Cache

CCI-400 Coherent Interconnect

"Always on, always connected tasks"

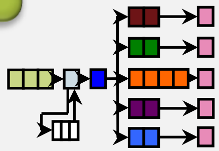
LITTLE

Performance and Energy-Efficiency

LITTLE

Most energy-efficient applications processor from ARM

- Simple, in-order, 8 stage pipeline
- Performance better than today's mainstream, high-volume smartphones

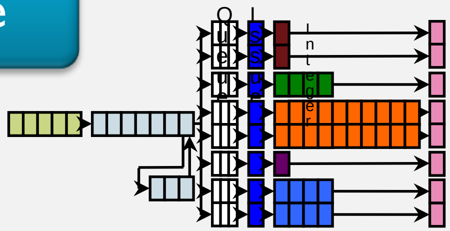


Cortex-A7

big

Highest performance in mobile power envelope

- Complex, out-of-order, multi-issue pipeline
  - Up to 5x the performance of today's mainstream, high-volume smartphones

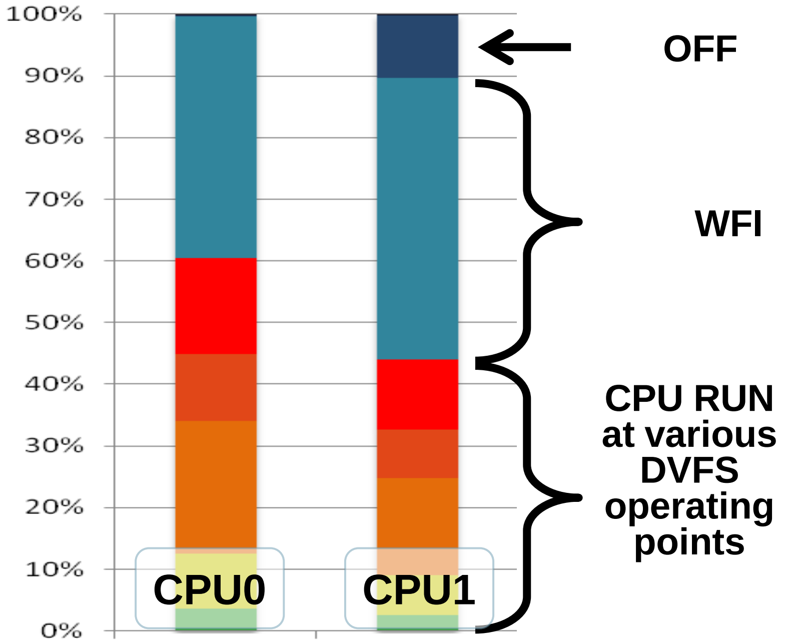


Cortex-A15

Power Analysis: Setting Up big.LITTLE

Android app used to gather CPU Activity information from a device

(DVFS states and per-CPU RUN / WFI / OFF)



Y-Axis is the % of the total runtime for the use case

Blue areas represent 'idle' states:
WFI, OFF, and Cluster OFF

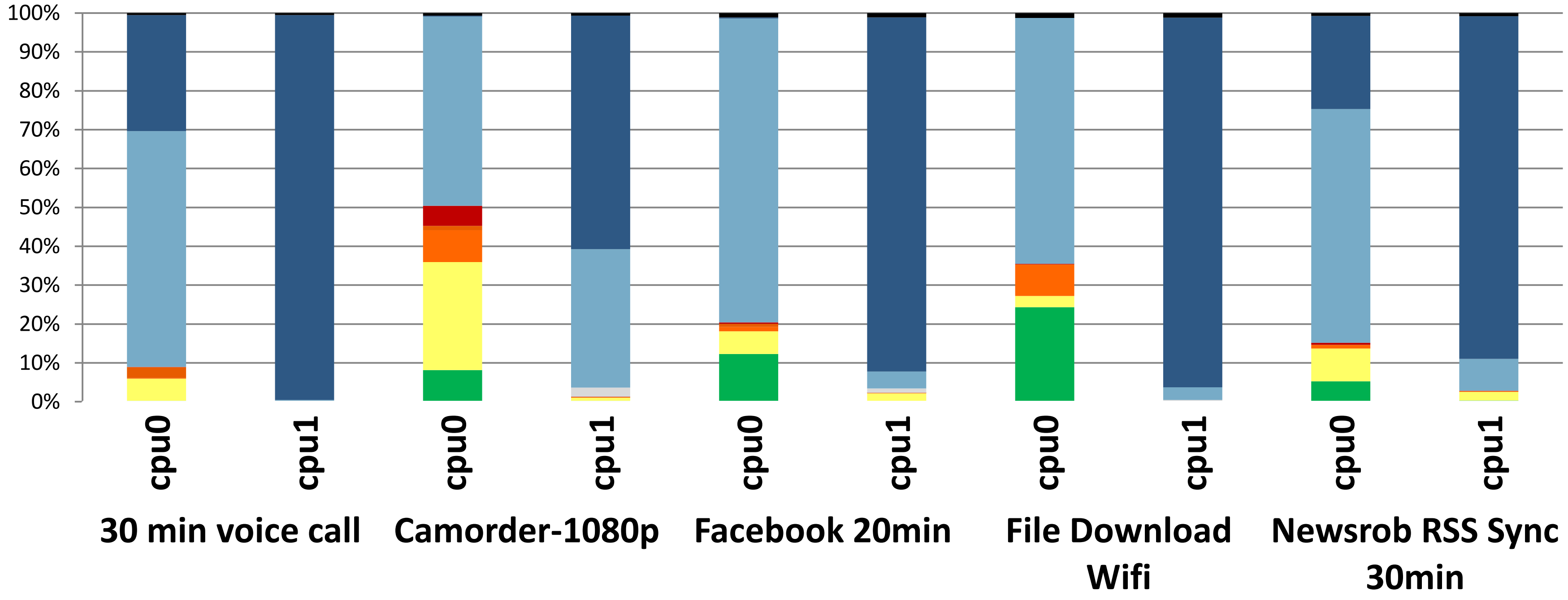
All other colors represent CPU RUN, at some DVFS OPP
Red is 'hottest' OPP

Per-CPU Data collected on MP systems

Note: Cortex-A7 should handle all except the red workloads (we expect it will even be able to handle a portion of today's red workloads)

Basic Workloads Today

All data from Cortex-A9 based mobile handset



- Basic activity (phone calls, etc.) can be handled almost entirely in "LITTLE" core

CPU Migration

- Per-Core Granularity
  - Each CPU software context can switch to big or LITTLE
  - Each CPU context switches independently of other CPU’s context
- CPU migration is driven by OS power management
  - DVFS algorithm monitors per CPU load
  - Operating point selection is done independently per CPU
  - When a LITTLE CPU cannot service the incumbent load a switch to an available big CPU is performed
  - The LITTLE processor within the cluster is switched off
  - When a lower performance is required, the software context will switch back to the LITTLE CPU
- If no CPUs are active within a cluster, then its L2 can also be powered off

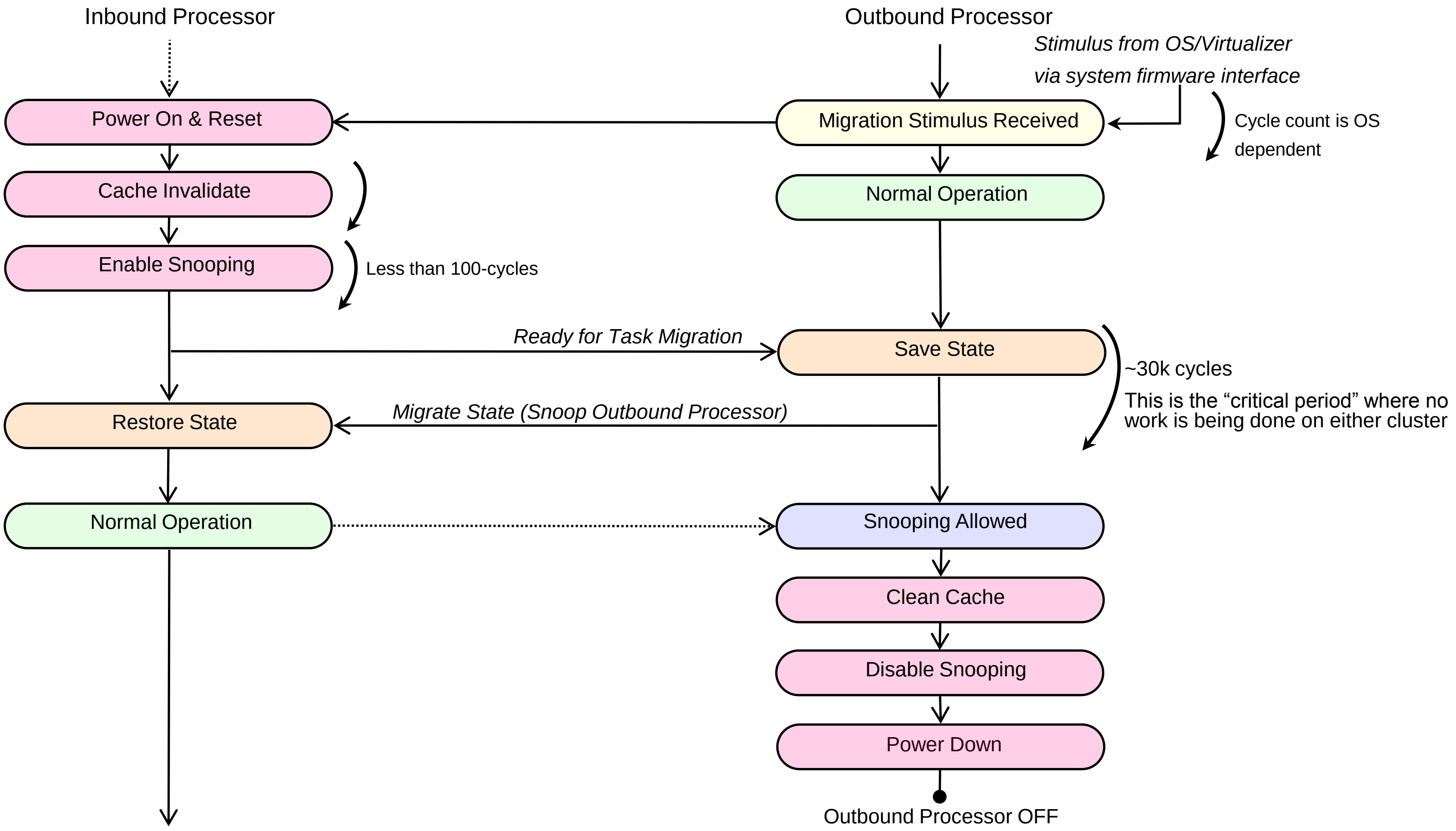
Cluster Migration

- Only one cluster is ever active
  - End-to-end “Interrupt off” switch time ~30K cycles

- Cluster selection driven by OS power management
  - DVFS algorithm selects a suitable operating point
  - A switch from the Cortex-A7 cluster to Cortex-A15 cluster is an extension of the DVFS strategy

- Load monitoring is done at the cluster level
  - Linux cpufreq samples load across all CPU in cluster
  - Selects a cluster operating point using the most loaded CPU
  - Switches cluster at an appropriate point on the DVFS curve

big.LITTLE Cluster Migration Mechanics



- State transfer is fast and atomic
- The mechanism is invisible to the payload software

Summary

- The combination of Cortex A15, ACE and CCI-400 offers customers a complete system coherent solution for the first time in ARM’s history
- Adding the low power Cortex A7 enables the big.LITTLE ecosystem to handle dynamic power/performance demands
- Many tradeoffs were necessary to achieve this while maintaining low power leadership with decent performance

Some Light Reading

- ARM Architecture Reference Manual ARMv7-A and ARMv7-R Edition
  - http://infocenter.arm.com
  - ARM Architecture → Reference Manuals
- DesignStart IP
  - http://www.arm.com/support/university/ip/index.php
- Careers
  - http://www.arm.com/about/careers/students/student-vacancies.php
  - http://www.arm.com/about/careers/graduates/vacancies.php

Q & A

ARM®

THE ARCHITECTURE FOR THE DIGITAL WORLD™