Exploring the Design of the Cortex-A15 Processor

ARM's next generation mobile applications processor

Travis Lanier

Senior Product Manager

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

- 1 TB physical addressing
- 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
- Extremely fast state save and restore

Large 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 HP process

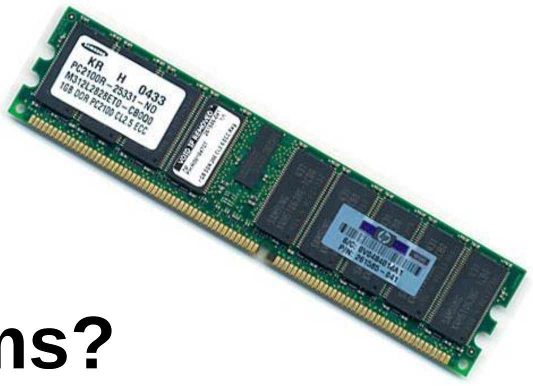
Agenda

- Architectual Updates and Key New Features
  - Large physical addressing
  - Virtualization
  - ISA extensions
  - Multiprocessing and AMBA 4
  - ECC
- Comparisons
- Microarchitecture
  - Frequency optimization
  - Pipeline IPC optimization

Large Physical Addressing – LPA

Cortex-A15 introduces 40-bit physical addressing

- 1 TB of memory
- 32-bit limited ARM to 4GB



What does this mean for ARM systems?

- More memory per core in an MP system
- More applications at the same time
- Applications can be wired into OS to take advantage directly
- Virtualization/multiple operating system instantiations

Virtualization

Seamlessly migrate OS instances between servers

- Run multiple OS instances simultaneously on same CPU
- Speeds recovery and migration
- Allows isolation of multiple work environments and data
- Power management under low loads

Hypervisor Partners

Open Kernel Labs™

vmware®

virtualLogix

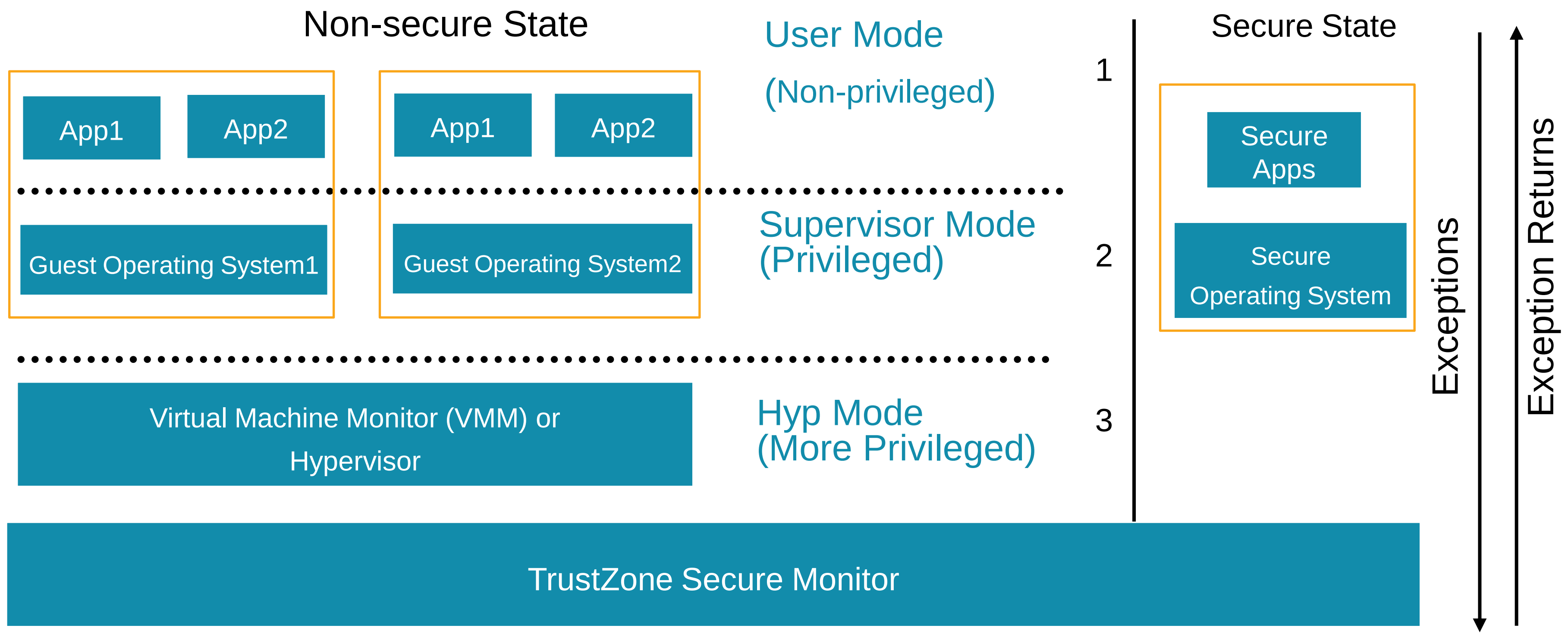
Builds on ARM TrustZone extensions

- Hypervisor privilege level
- Two level address translation
- Supports execution of existing binaries
- Includes support for I/O

Virtualization Extension Basics

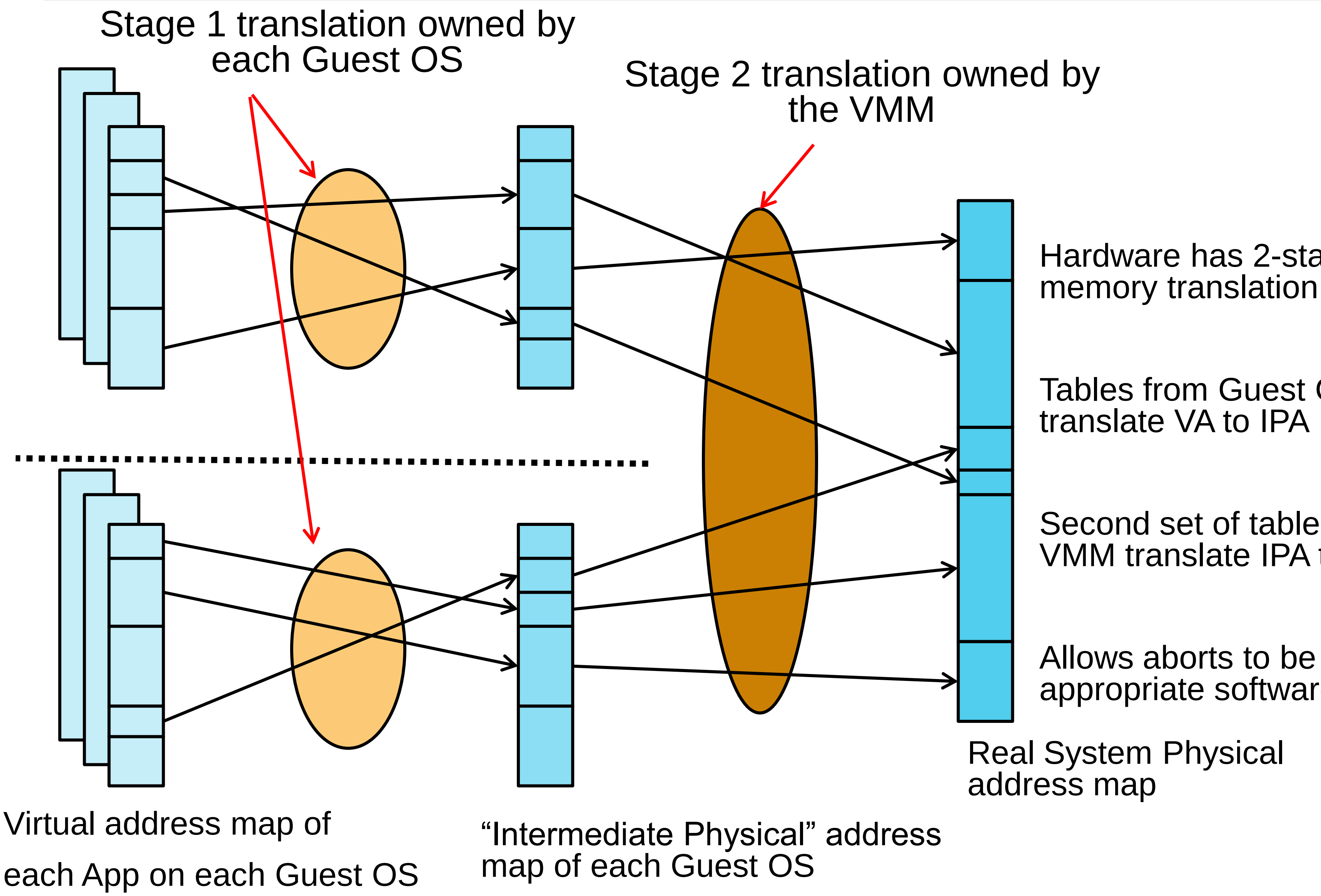
- New Non-secure level of privilege to hold Hypervisor
  - Hyp mode
- New mechanisms avoid the need Hypervisor Intervention for:
  - Guest OS Interrupt masking bits
  - Guest OS page table management
  - Guest OS Device Drivers due to Hypervisor memory relocation
  - Guest OS communication with the GIC
- New traps into Hyp mode for:
  - ID register accesses; WFI/WFE
  - Miscellaneous “Difficult” System Control Register cases
- New mechanisms to improve:
  - GuestOS Load/Store emulation by the Hypervisor
  - Emulation of Trapped instructions

Virtualization: A Third Layer of Privilege



- Guest OS same privilege structure as before
    - Can run the same instructions
- New Hyp mode has higher privilege
- VMM controls wide range of OS accesses to hardware

Virtual Memory in Two Stages


Stage 1 translation owned by each Guest OS
Stage 2 translation owned by the VMM
Virtual address map of each App on each Guest OS
"Intermediate Physical" address map of each Guest OS
Real System Physical address map

Hardware has 2-stage memory translation

Tables from Guest OS translate VA to IPA

Second set of tables from VMM translate IPA to PA

Allows aborts to be routed to appropriate software layer

ISA Extensions

Instructions added to Cortex-A15

(and all subsequent Cortex-A cores)

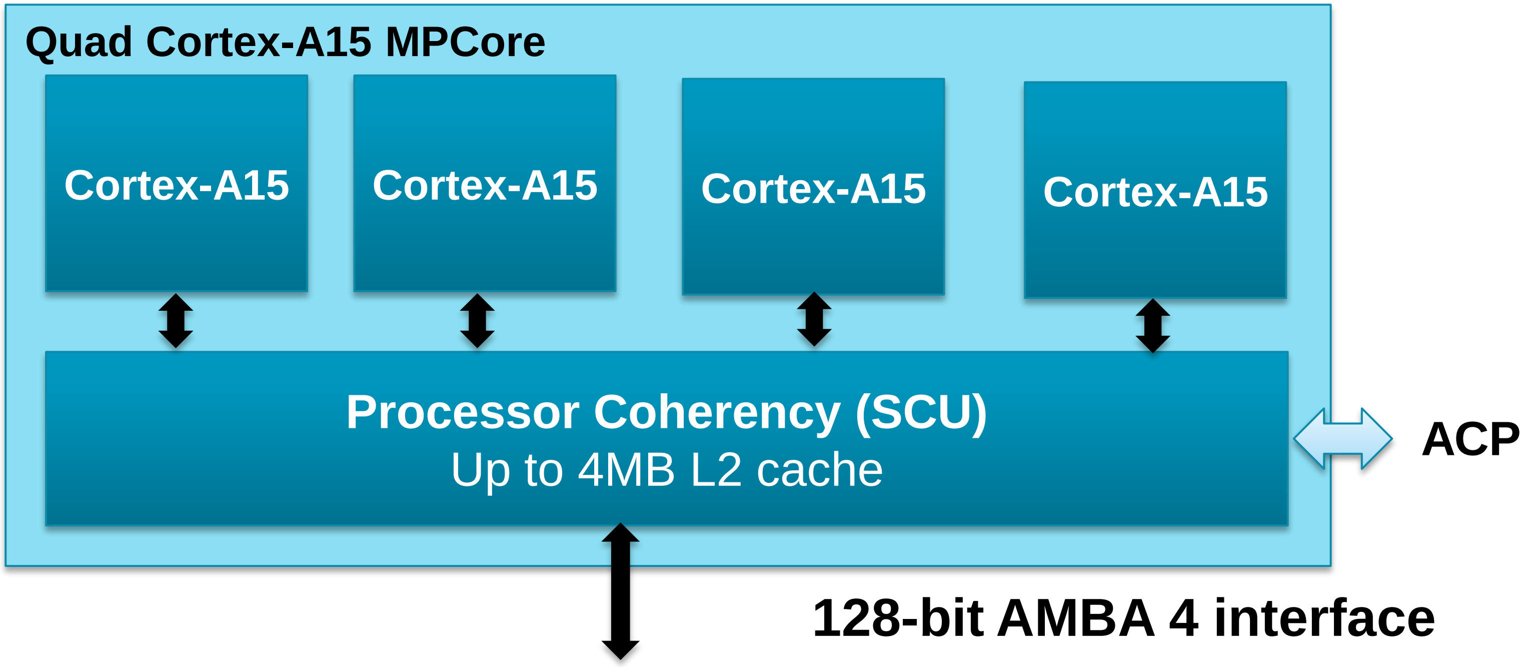
- **Integer Divide**
  - Similar to Cortex-R, M class (driven by automotive)
  - Use getting more common
- **Fused MAC**
  - Normalizing and rounding once after MUL and ADD
  - Greater accuracy
  - Requirement for IEEE compliance
  - New instructions to complement current chained multiply + add

Hypervisor Debug

- Monitor-mode, watchpoints, breakpoints

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



Scaling Beyond Four Cores

Introducing AMBA 4 coherency extensions

- Coherency, Barriers and Virtualization signalling

Software implications

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

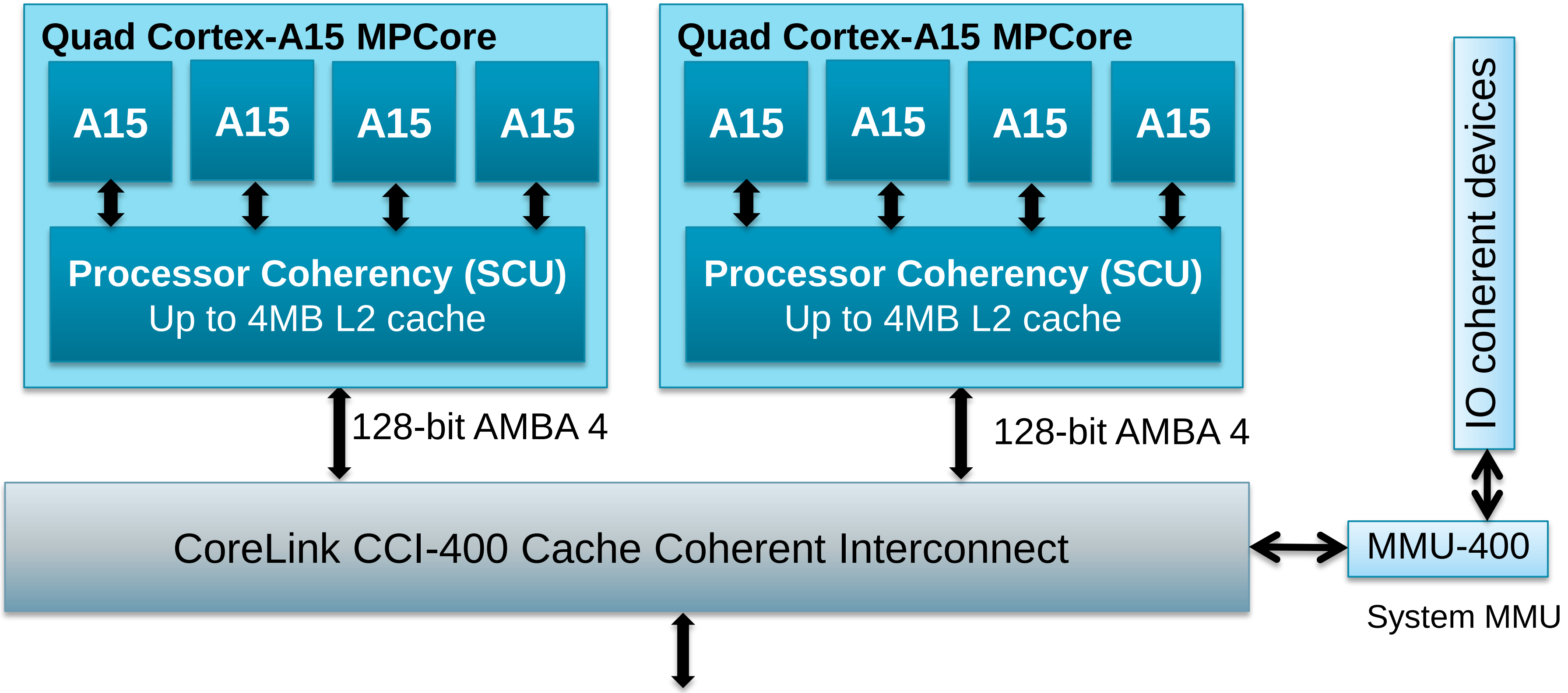
Coherency types

- Within a MPCore cluster: existing SCU SMP coherency
- Between clusters: AMBA 4 ensures coherency with snoops
- I/O coherent devices can read processor caches

Cortex-A15 System Scalability

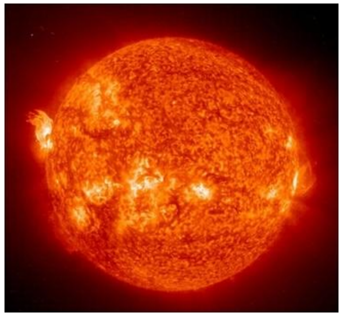
Introducing CCI-400 Cache Coherent Interconnect

- Processor to Processor Coherency and I/O cohency
- Memory and synchronization barriers
- Virtualization support with distributed virtual memory signalling



Memory Error Detection/Correction

Error Correction Control on L1 and L2 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
- Optimize for common case – so correction not in critical path

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

Cortex-A15 Microarchitecture

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

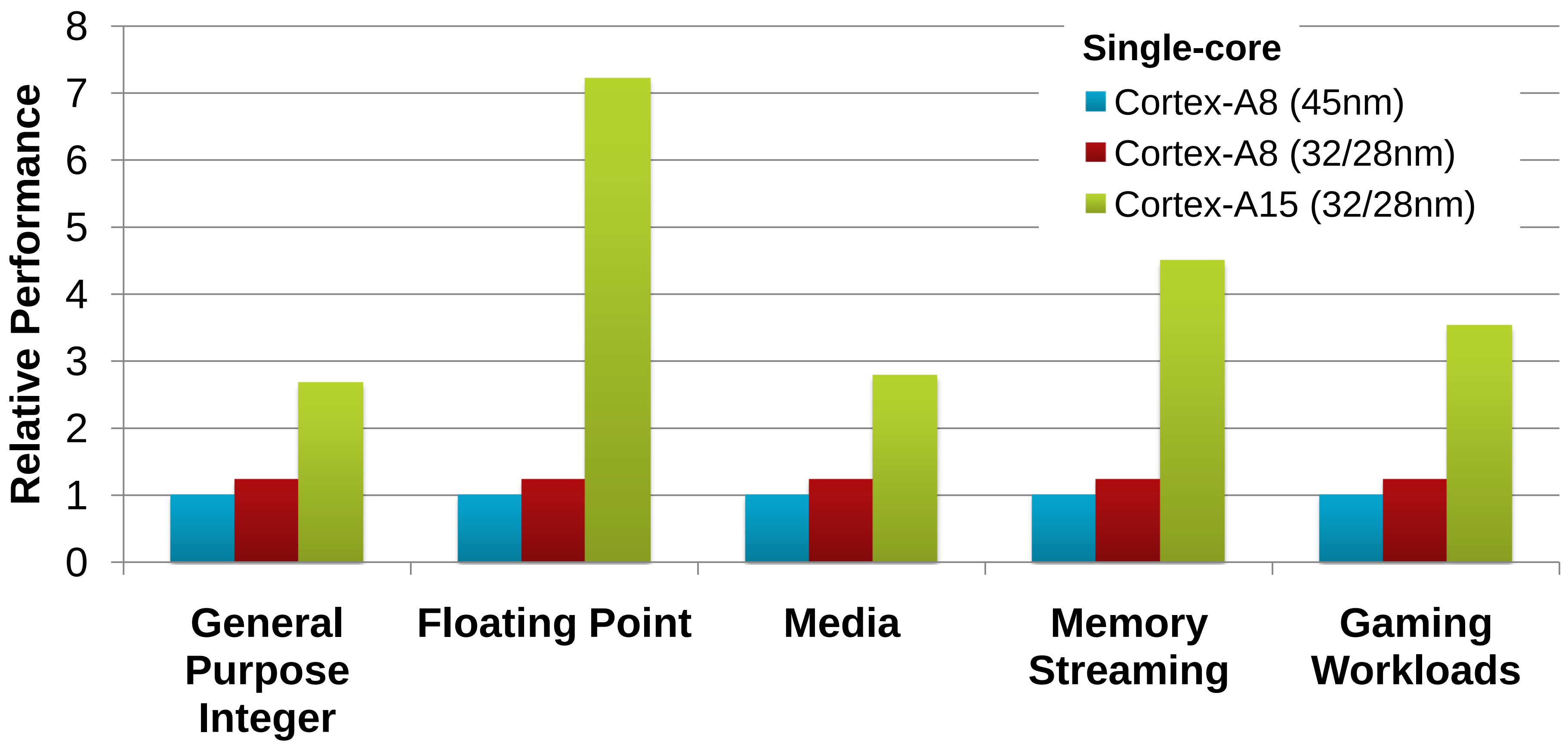
- Microarchitecture loop:
  - Key function designed to complete in a cycle (or a set of cycles)
    - cannot be further pipelined (with high performance)
- Some example loops:
  - Register Rename allocation and table update
  - Result data and tag forwarding (ALU->ALU, Load->ALU)
  - Instruction Issue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

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

Where to Find Performance: IPC

- Improved branch prediction
- Wider pipelines for higher instruction throughput
- Larger instruction window for out-of-order execution
- More instruction types can execute out-of-order
- Tightly integrated/low latency NEON and Floating Point Units
- Improved floating point performance
- Improved memory system performance

High-end Single Thread Performance

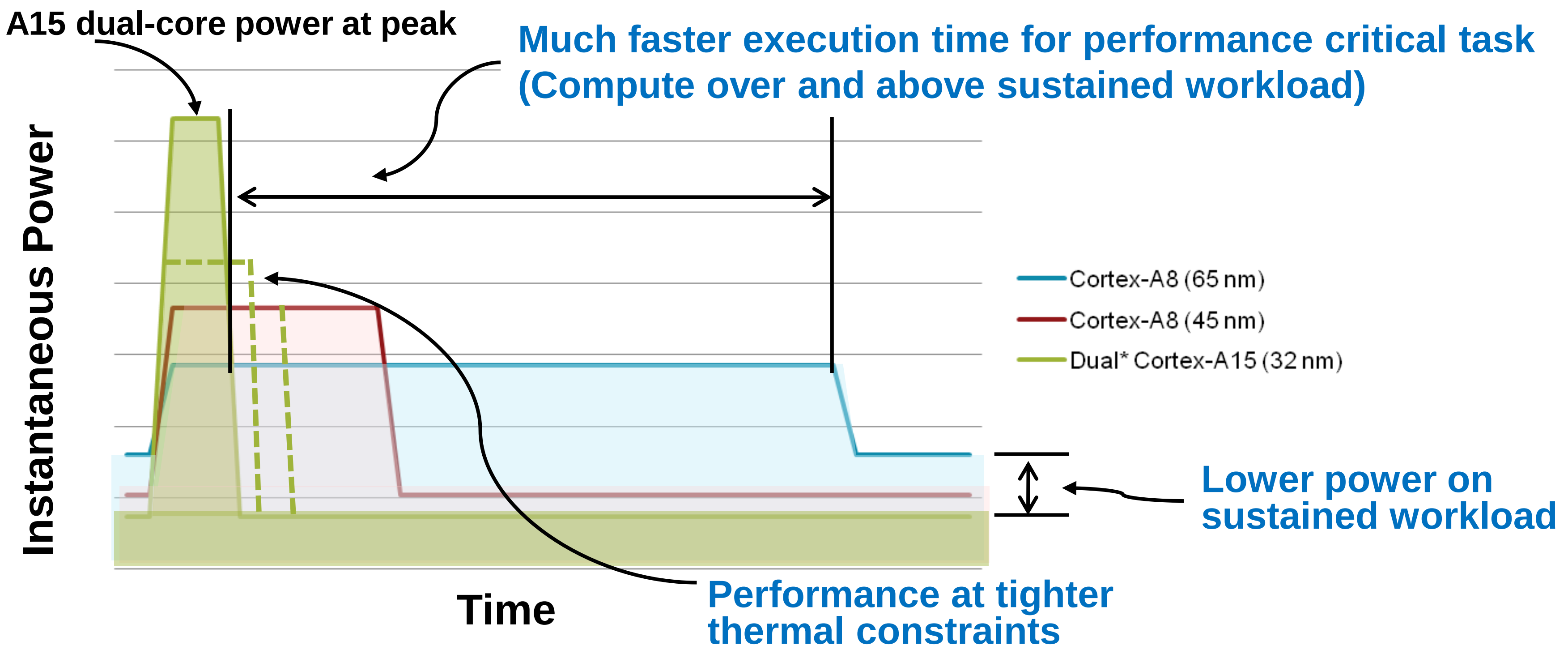


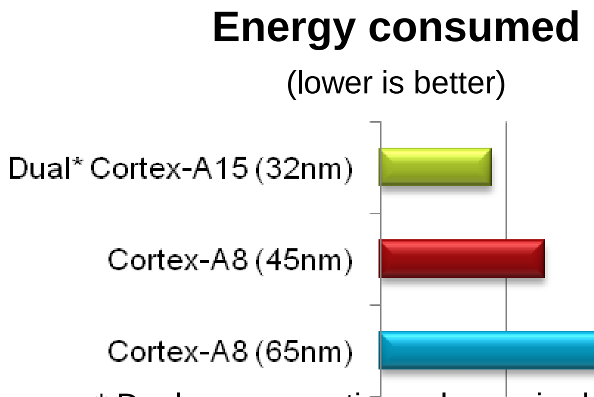
- **Both processors using 32K L1 and 1MB L2 Caches, common memory system**
- **Cortex-A8 andCortex-A15 using 128-bit AXI bus master**

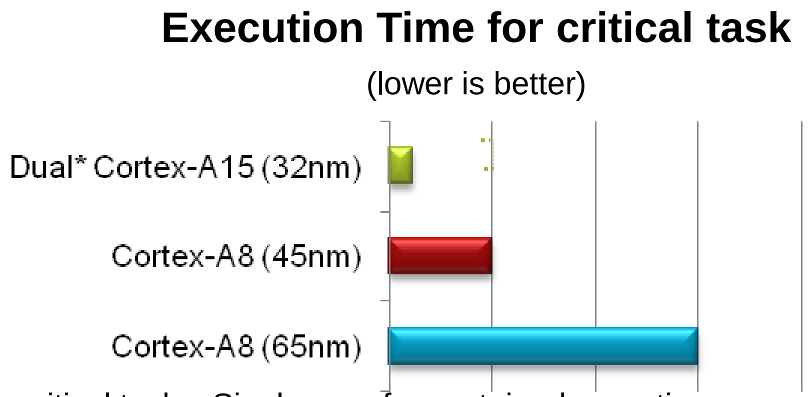
Note: Benchmarks are averaged across multiple sets of benchmarks with a common real memory system attached

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)
Instantaneous Power
Time
Cortex-A8 (65 nm)
Cortex-A8 (45 nm)
Dual* Cortex-A15 (32 nm)
Lower power on sustained workload
Performance at tighter thermal constraints



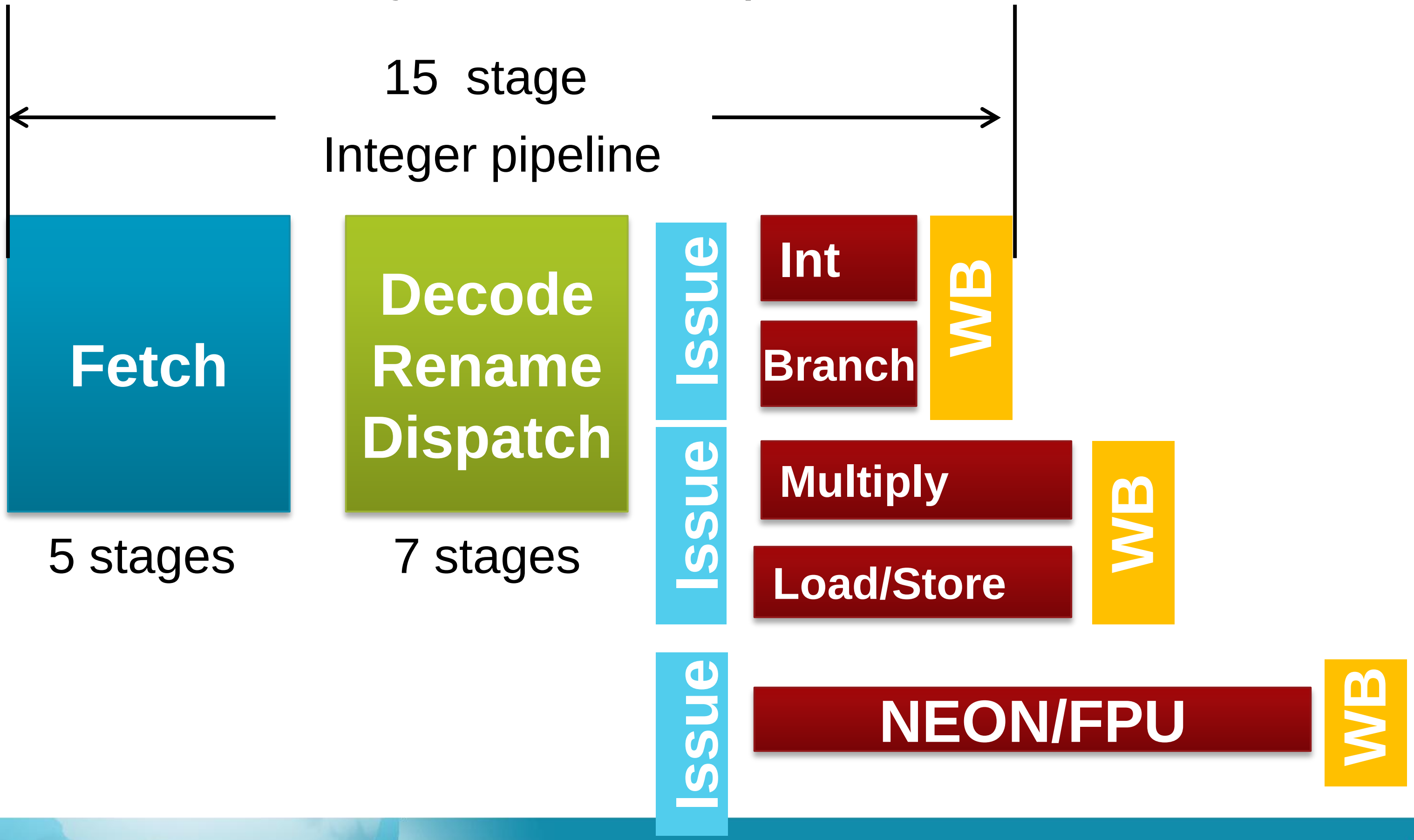


* Dual-core operation only required for high-end timing critical tasks. Single-core for sustained operation

Cortex-A15 Pipeline Overview

15-Stage Integer Pipeline

- 4 extra cycles for multiply, load/store
- 2-10 extra cycles for complex media instructions



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

The rename loop

- Destination registers are always renamed to top entry of result queue
  - Rename table updated for next cycle access
- Source register rename mappings are read from rename table
  - Bypass muxes present to handle same cycle forwarding
- Result queue entries reused when flushed or retired to architectural state

Increasing Out-of-Order Execution

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

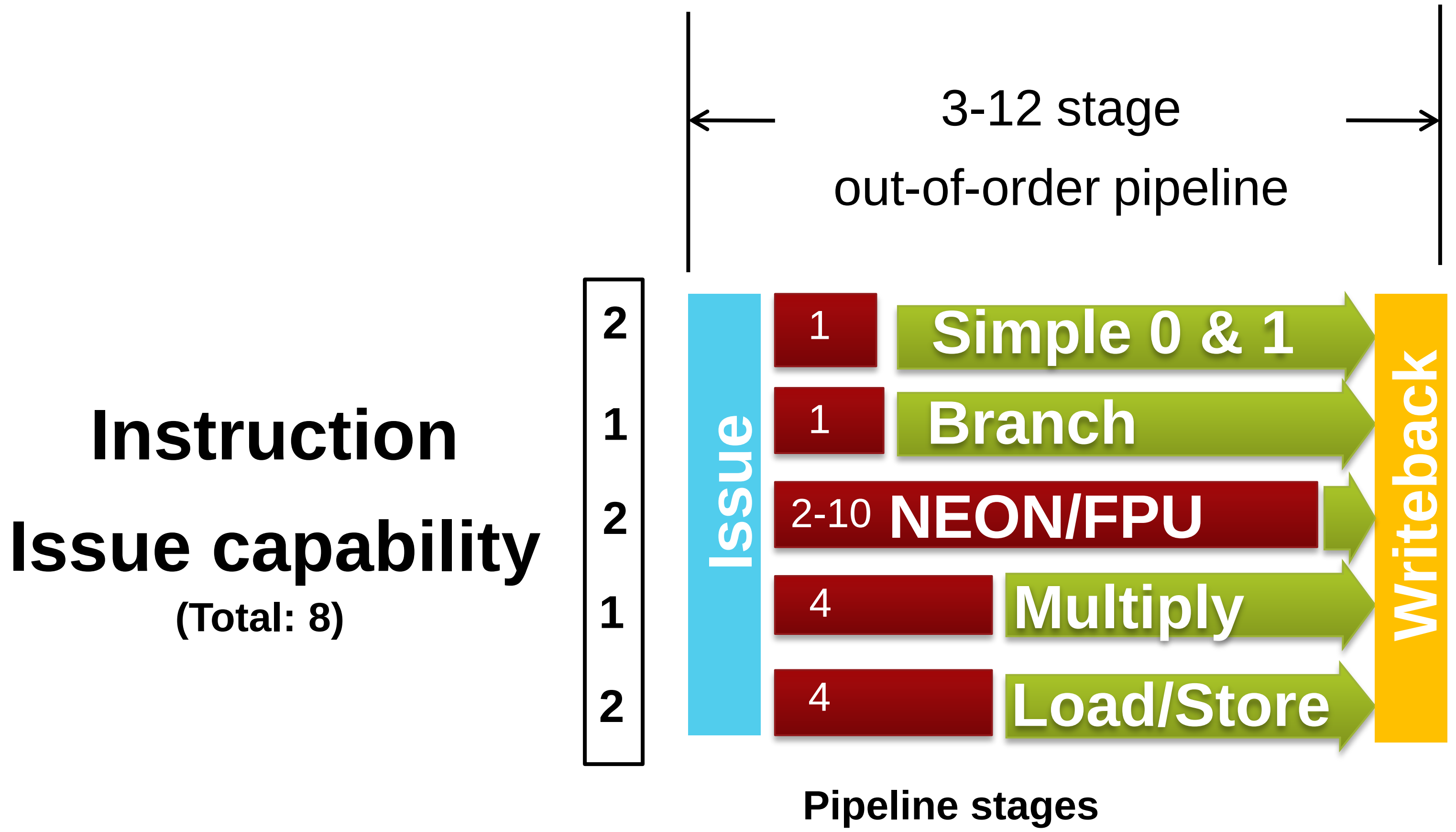
- Effectiveness limited by how far you look ahead
  - Window size of 40+ operations required for Cortex-A15 performance targets
- Issue queue size often frequency limited to 8 entries

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



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

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

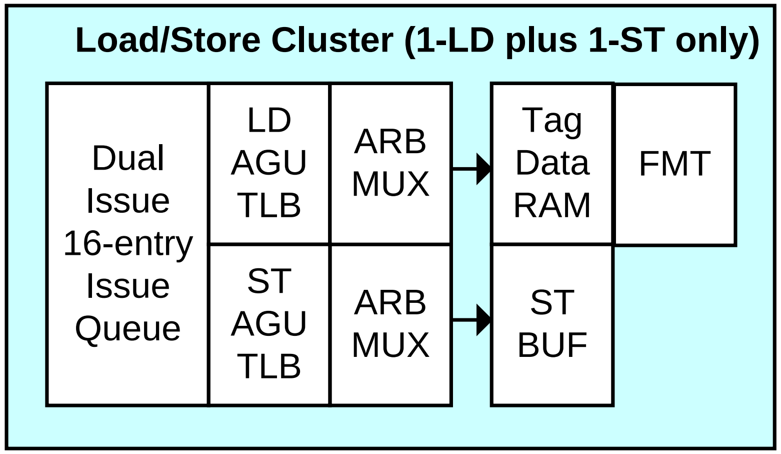
Specific challenges for Out-of-order VFP/Neon

- 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
- Includes walking cache structures

Active power management

- 32 entry loop buffer
- Loop can contain up to 2 fwd branches and 1 backwards branch
- Completely disables Fetch and most of the Decode stages of pipeline

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

- Supports logging of error location and frequency

Overall Summary

- The Cortex-A15 extends the application processor family with
  - Dramatic increase in single-thread and overall performance
  - Compelling new features, functionality enable exciting OEM products
  - Scalability for large-scale computing and system-on-chip integration

- Cortex-A15 has strong momentum in mobile market

SAMSUNG ST ERICSSON TEXAS INSTRUMENTS NVIDIA. BROADCOM.

- ARM Cortex-A family provides broadest range of processors
  - Ultra-low cost smartphones through to tablets and beyond
  - Full upward software and feature-set compatibility
  - Address cloud computing challenges from end to end

Thank You

Please visit www.arm.com for ARM related technical details

For any queries contact <Salesinfo-IN@arm.com>