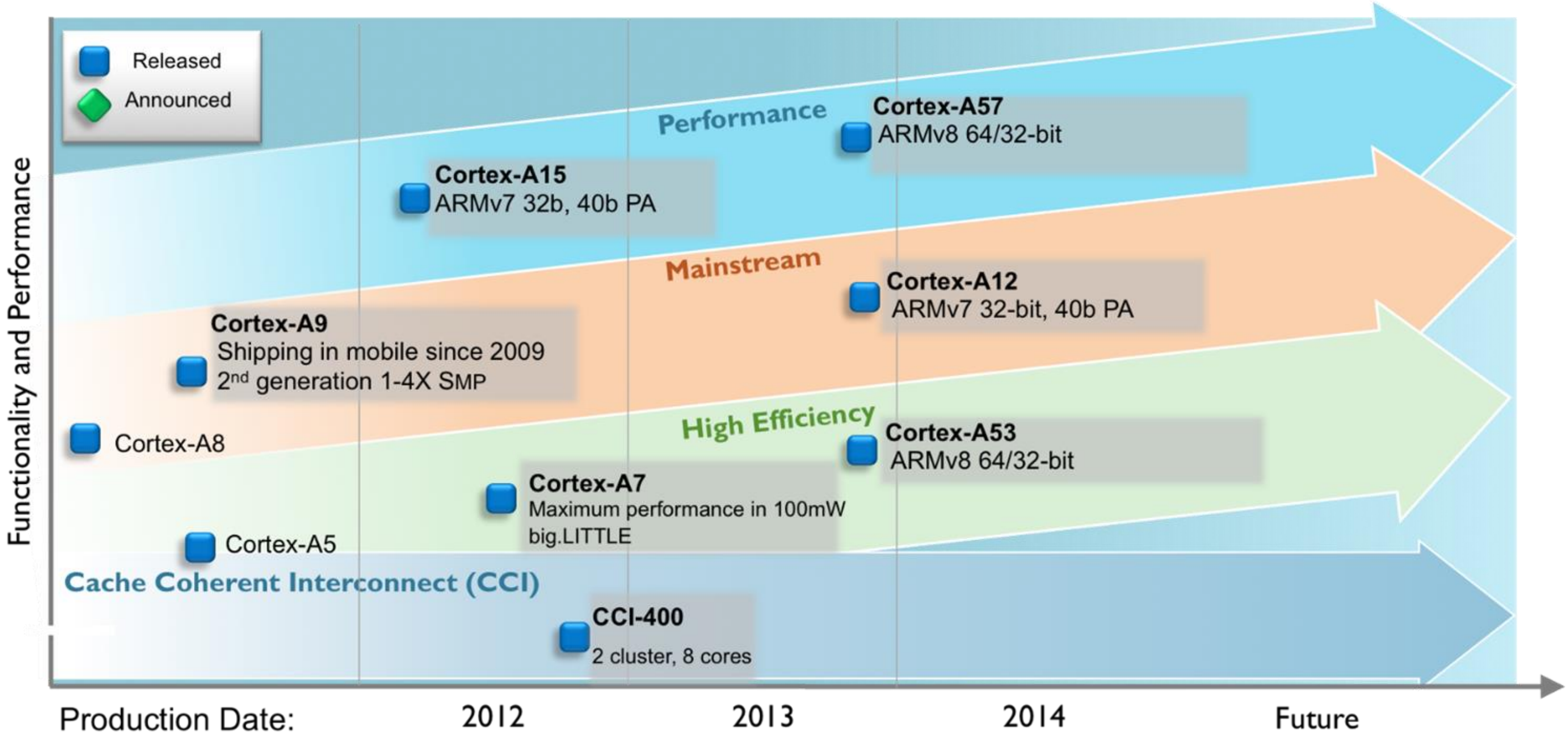
ARM Cortex-A*

"Any sufficiently advanced technology is indistinguishable from magic" - Arthur C. Clarke

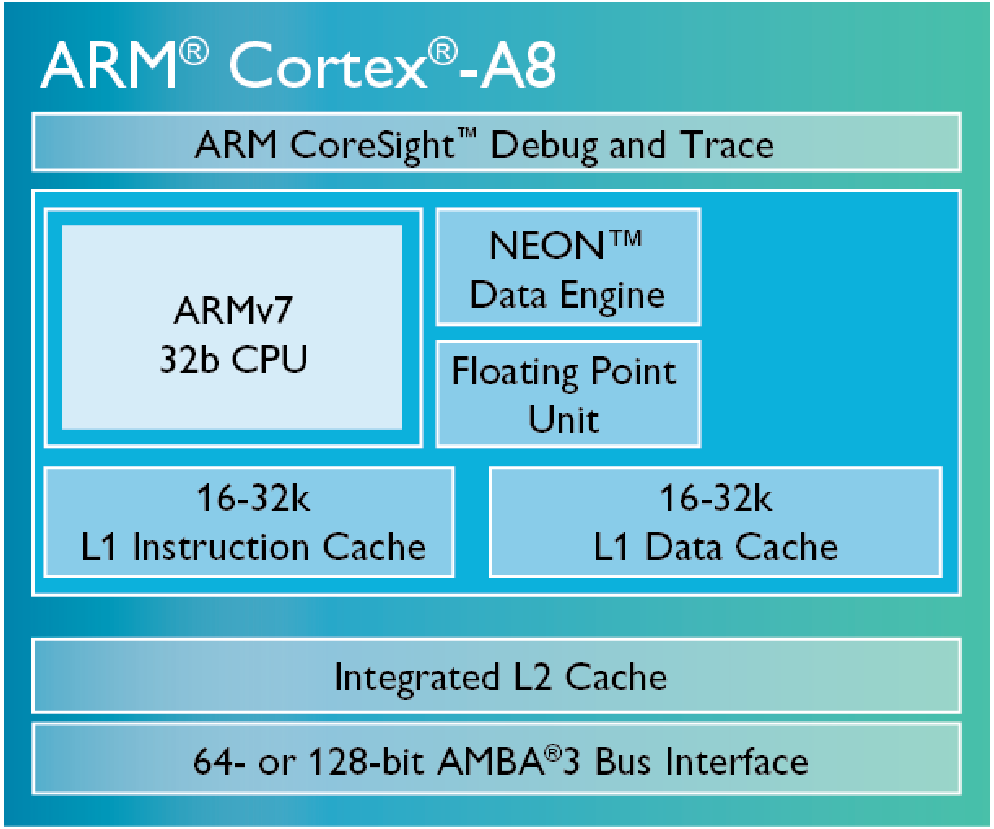
Brian Eccles, Riley Larkins, Kevin Mee, Fred Silberberg, Alex Solomon, Mitchell Wills

Images and information courtesy of *Computer Architecture A Quantitative Approach (5th edition)* by Hennesy and Patterson, and *ARM® Cortex®-A57 MPCore Processor Technical Reference Manual Revision: r1p3* published by ARM

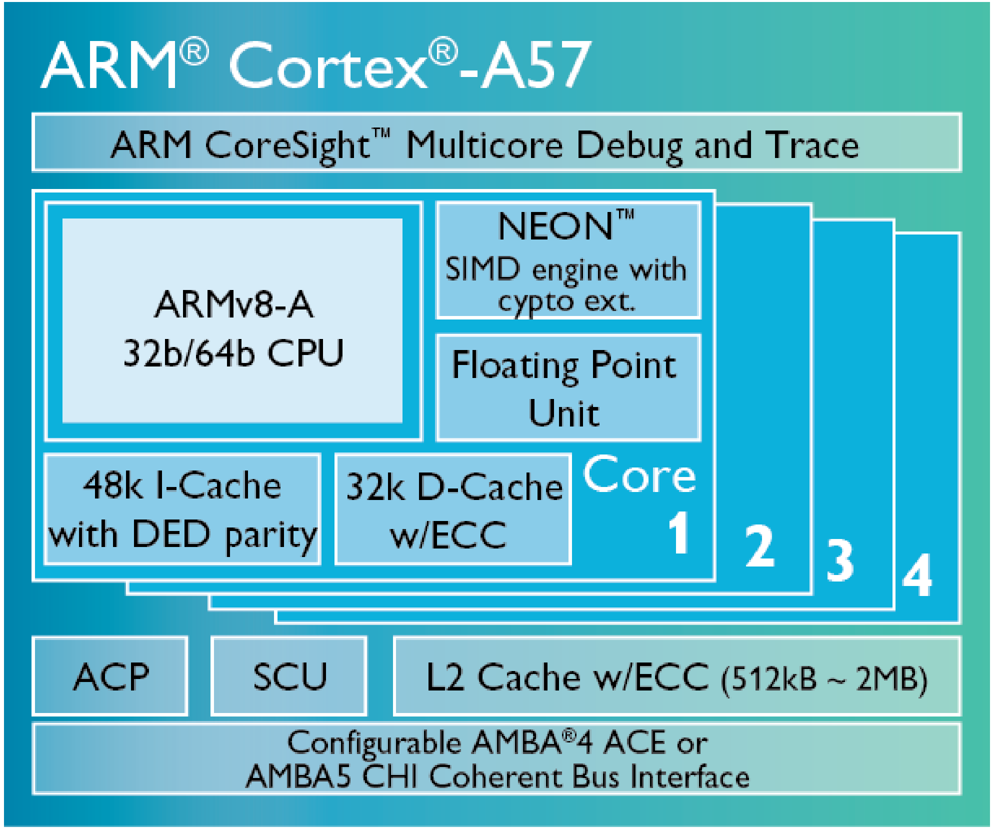
ARM Cortex History



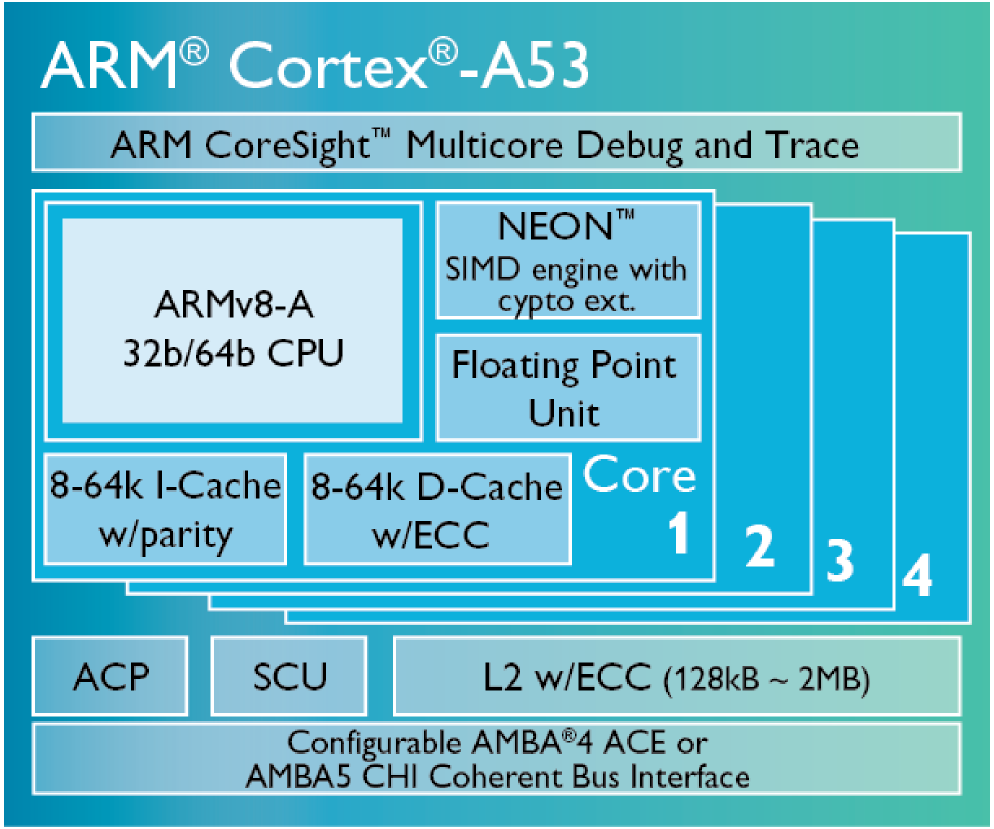
ARM Cortex-A8



ARM Cortex-A57



ARM Cortex-A53



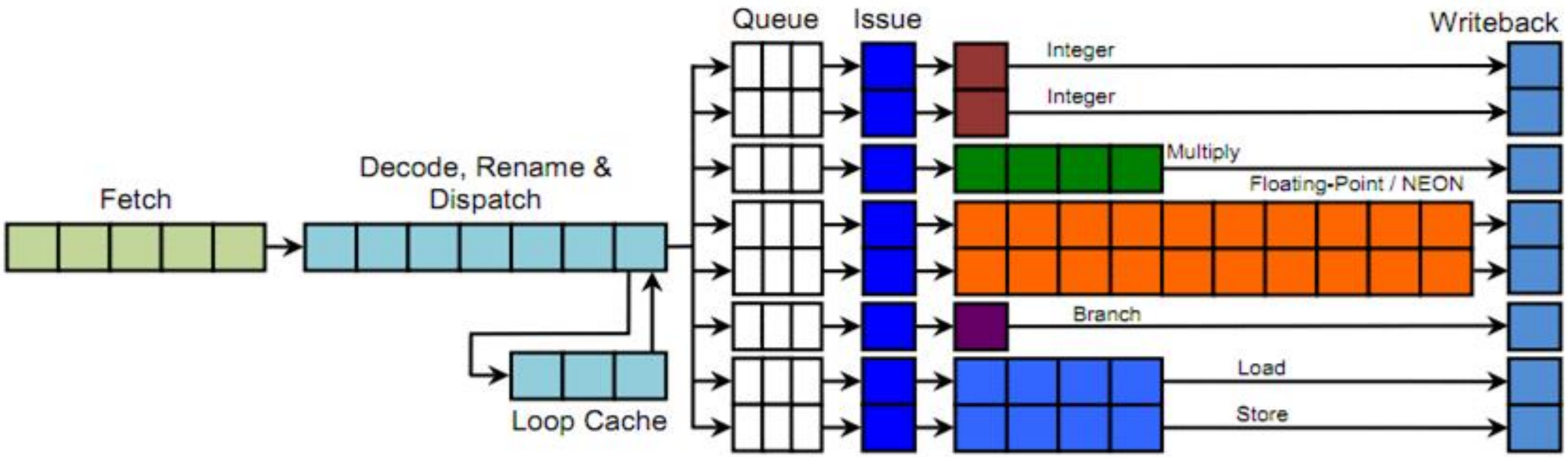
Overview A57

- 1-4 Cores
- 64-bit memory addressing
- Up to Three Instructions per Cycle
- 12 Stage In-Order Pipeline
- 3-15 Stage Out-Of-Order Pipeline

Cortex A53

- Most power-efficient ARMv8 processor
- Supports 32-bit and 64-bit
- Highly scalable
  - single multi-core CPU cluster
  - multi-cluster enterprise system

A57 - Pipeline Overview



A57 - In Order Pipeline

- 5 Stage Instruction Fetch
- 7 Stage Instruction Decode and Register Renaming

Instruction Fetch

- Fetches instructions from L1 instruction cache
- Sends up to 3 instructions per cycle to decode
- Branch prediction
  - 2-level dynamic predictor with Branch Target Buffer
  - Return stack
  - Indirect predictor - non-return type
  - Static predictor - unconditional branches
  - Buffer invalidated on context switch

Instruction Decode

- Instruction Sets
  - A32
  - T32
  - A64
  - SIMD, Floating-point and cryptography instructions
- Performs register renaming
  - allow for out of order instruction execution
  - removes write after write and write after read
- 3 instructions per cycle

A32

- Fixed length 32-bit instructions
- ARMv7
- Executes in AArch32 execution state (32-bit)
- Previously called ARM instruction set
- For high performance applications
- Most instructions can be conditional
  - negative, zero, carry, overflow

T32

- Variable length instructions (16-bit or 32-bit)
- ARMv7
- Executes in AArch32 execution state (32-bit)
- Previously called Thumb instruction set
- Higher code density

A64

- Fixed length 32-bit instructions
- ARMv8
- Executes in AArch64 execution state (64-bit)
- Fewer conditional instruction
- No named access to program counter

A57 - Out of Order Pipeline

- 8 Parallel pipelines
- Dispatch + 1-10 Stages + WriteBack
- Simple Integer 0/1, Branch
- Integer Multi-cycle Load, Store
- Floating Point/SIMD 0/1

Dispatch Stage

- Three micro-operations per cycle
- Operations are queued for each execution pipeline

Branch

- 1 Stage
- Some operations also use simple integer

Simple Integer Pipeline

- Two pipelines: Integer 0, Integer 1
- Add, subtract, bitwise operations
- 1 cycle of latency
- 2 operations per cycle
- Some SIMD operations

Multi-cycle Integer Pipeline

- Integer Multiply, Divide, Shift
- 4 Stages
- Variable latency
  - 4-36 Cycles for divide
- Some operations block all stages while active

Load/Store

- One load, one store per cycle
- Many operations also use a simple integer pipeline

Floating Point/ASIMD

- FP multiply, add
- ASIMD basic operations
- F0 supports ASIMD integer multiply, FP divide, crypto operations
- F1 supports ASIMD shift operations

Exception Levels

- EL0-EL3
- Restrictions based on:
  - Exception level
  - Security state
  - Execution state

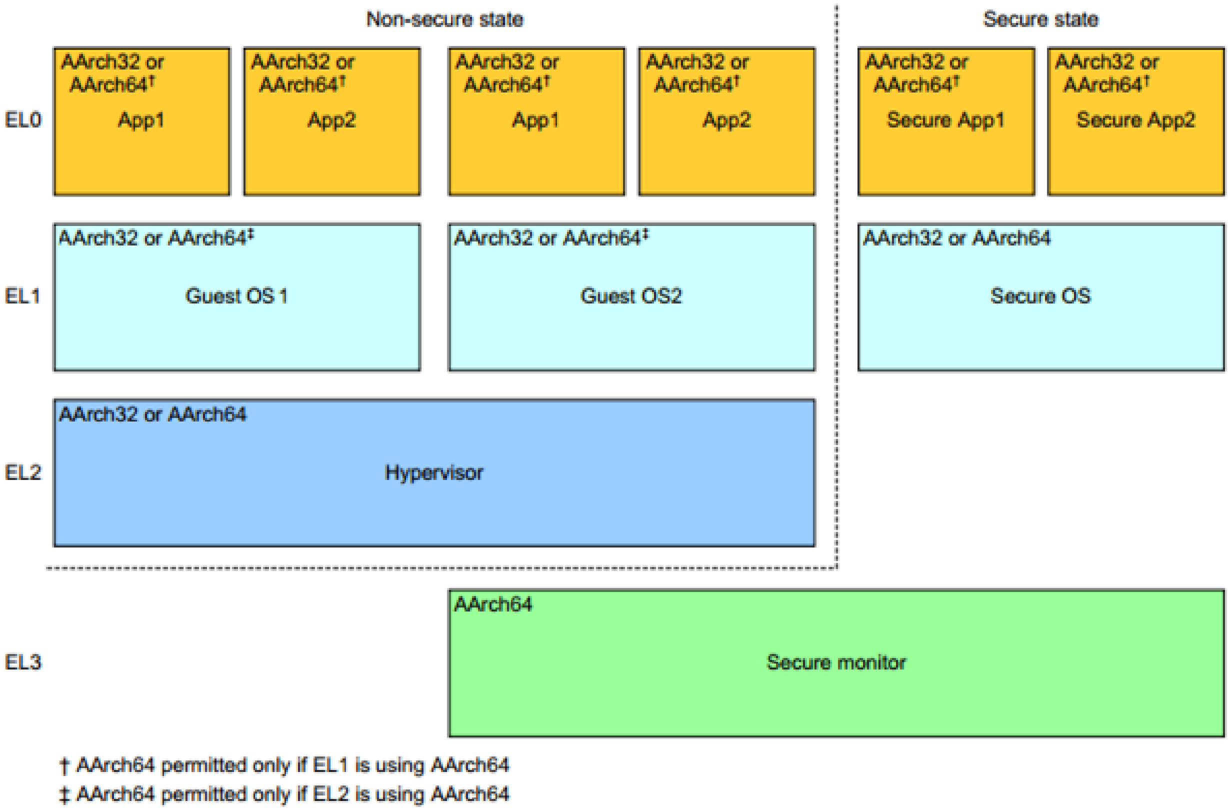
Exception Handling

- EL0 - Application Mode
- EL1 - OS Kernel
- EL2 - Hypervisor
- EL3 - Secure Mode

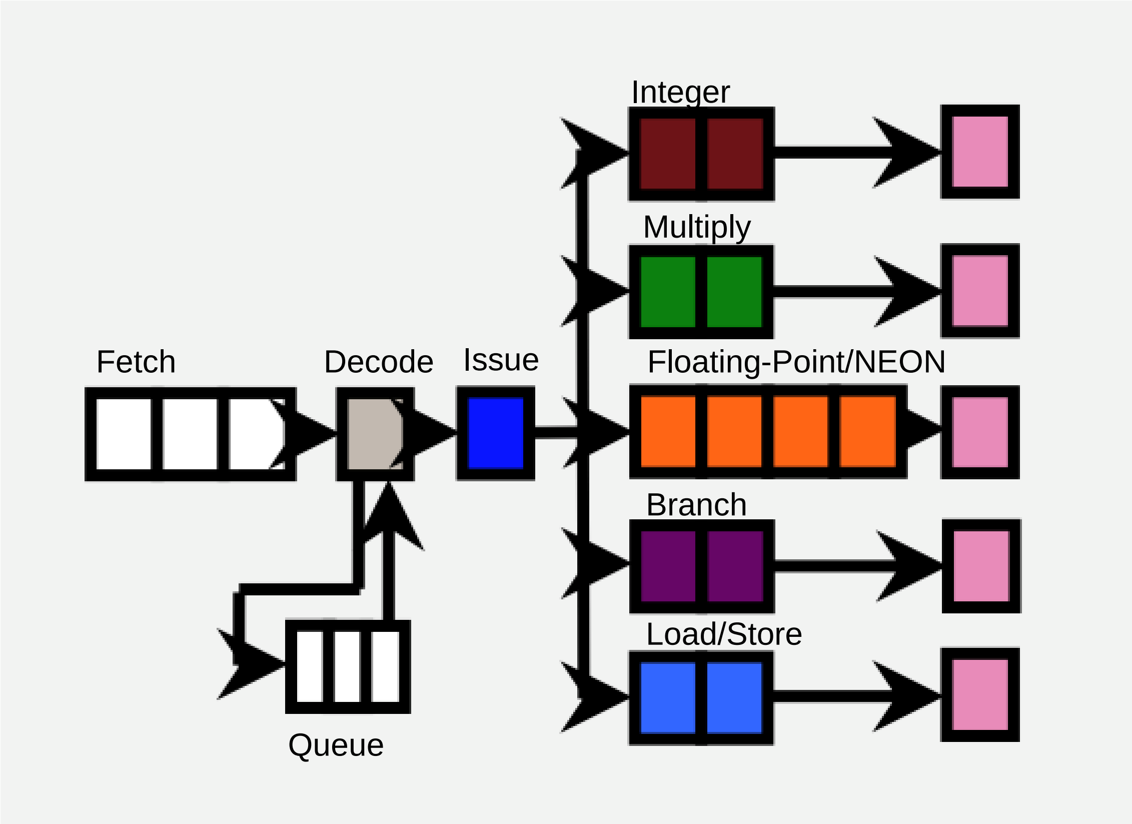
Changing Execution States

- Can only change on exceptions
- On increase in exception level:
  - Remain the same
  - AArch32 to AArch64
- On decrease in exception level:
  - Remain the same
  - AArch64 to AArch32

Example Exception Uses



A53 - Pipeline Overview



Memory Management

- Controlled by Memory Management Unit (MMU)
- Separate L1 data and instruction caches
- L2 cache shared by all cores
- 2 Level Translation Lookaside Buffer (TLB) for address translation

L1 Instruction Cache Comparison

	Cortex-A8	Cortex-A57	Cortex-A53
Size	16-32 KB	48 KB	8-64 KB
Associativity	4 way set associative	3 way set associative	2 way set associative
Block Size	64 bytes	64 bytes	64 bytes
Redundancy	1 parity bit per byte	1 parity bit per 2 bytes	1 parity bit per byte
Tagging	VIPT	PIPT	VIPT
Replacement Policy	Random	Least Recently Used	Pseudo-random

L1 Data Cache Comparison

	Cortex-A8	**Cortex-A57**	**Cortex-A53**
Size	16-32 KB	32 KB	8-64 KB
Associativity	4 way set associative	2 way set associative	4 way set associative
Block Size	64 bytes	64 bytes	64 bytes
Redundancy	1 parity bit per byte	ECC	ECC
Tagging	PIPT	PIPT	PIPT
Replacement Policy	Random	Least Recently Used	Pseudo-random

L2 Shared Cache Comparison

	Cortex-A8	Cortex-A57	Cortex-A53
Size	0 KB - 1 MB	512 KB - 2 MB	128 KB - 2 MB
Associativity	8 way set associative	16 way set associative	16 way set associative
Block Size	64 bytes	64 bytes	64 bytes
Redundancy	Optional parity or ECC	ECC	Optional ECC
Tagging	PIPT	PIPT	PIPT
Replacement Policy	Random	Random	Pseudo-random

TLB Comparison

	Cortex-A8	Cortex-A57	Cortex-A53
Level 1 Instruction	32 entry, fully associative	48 entry, fully associative	10 entry, fully associative
Level 1 Data	32 entry, fully associative	32 entry, fully associative	10 entry, fully associative
Level 2 Combined	None	1024 entry, 4 way associative	512 entry, 4 way associative

A57 and A53 TLB Entries

Each entry contains:

- Virtual Address
- Physical Address
- Page size
- Memory type
- Permissions
- Application Specific Identifier (ASID)
- Virtual Machine Identifier (VMID)
- Exception level

A57 and A53 TLB Match Conditions

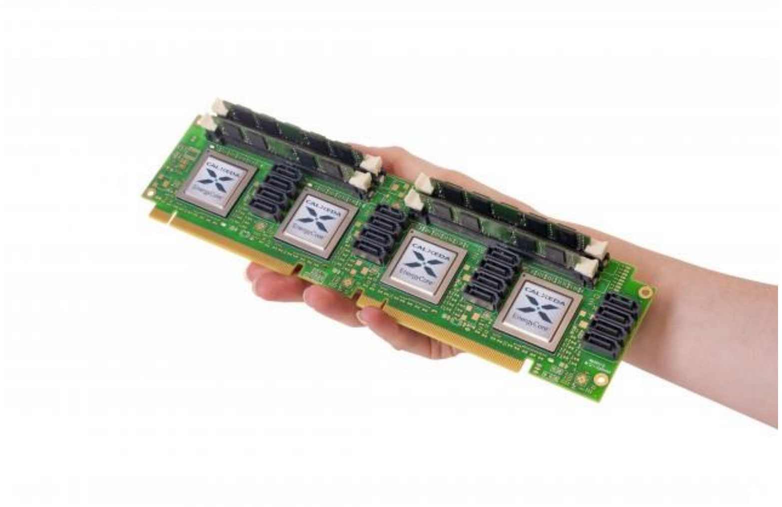
- The VA matches the VA in the entry
- The memory space of the entry matches the memory space of the request
- The ASID in the entry matches the ASID in the CONTEXTIDR register or is global
- The VMID in the entry matches the VMID in the VTTBR register

A57 Memory Access Sequence

1. Attempt to match the provided VA to an entry in the correct Level 1 TLB
   a. On a miss, attempt to match the provided VA to an entry in the Level 2 TLB
   b. On a miss, perform a table walk in main memory
2. Check the entry’s permission bits
   a. Issue a Permission Fault on failure
3. Check the security state of the entry
4. Return the translated PA
5. Check the corresponding L1 cache for the PA
   a. On a miss, check the L2 cache
   b. On a miss, issue a request to main memory

Virtualization

- Adds support for hardware assisted virtualization
- TLB entries contain ASID and VMID to permit context and VM switches without flushing the TLB
- Brings ARM into low power server processor market

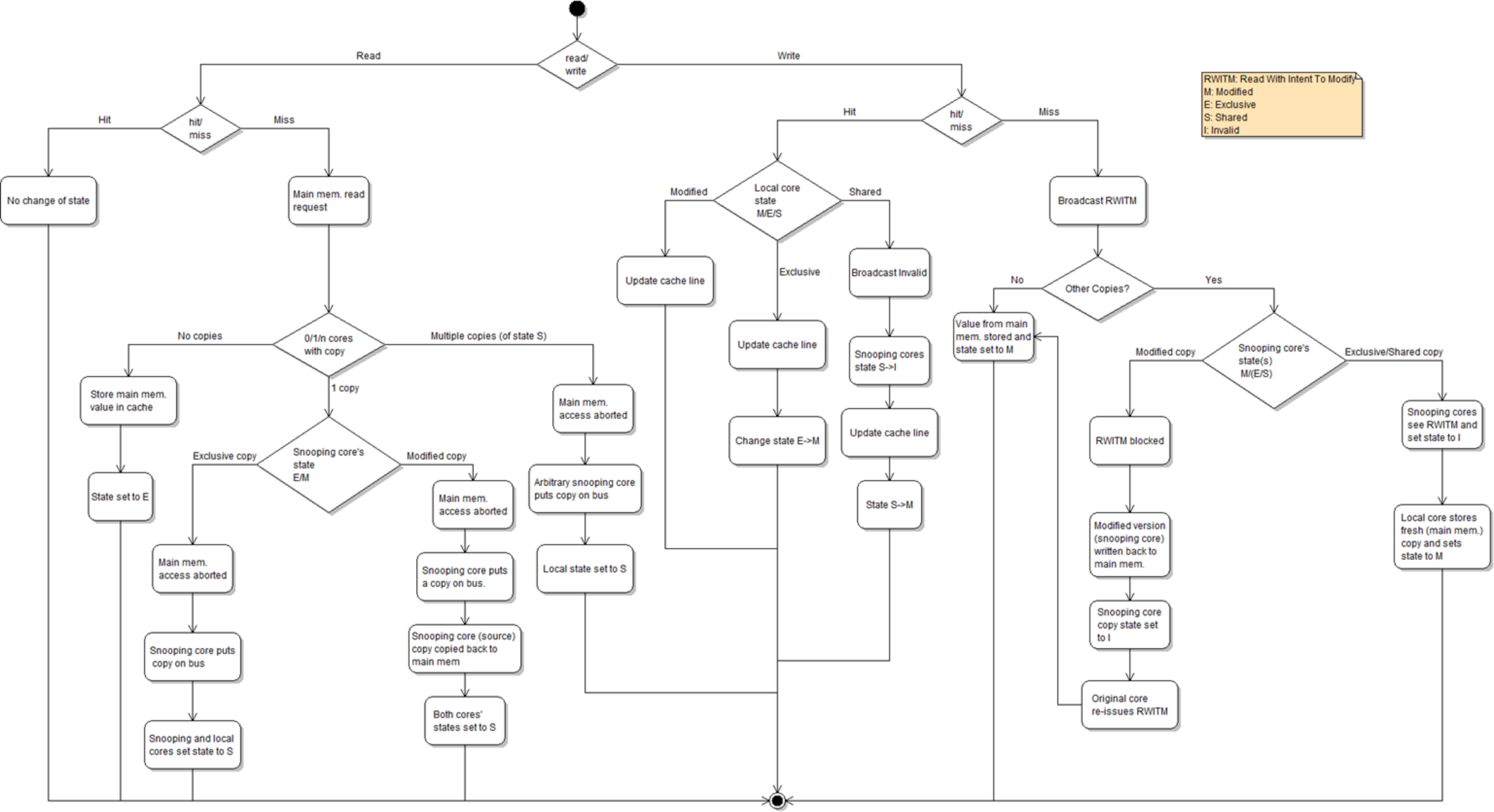


Snooping

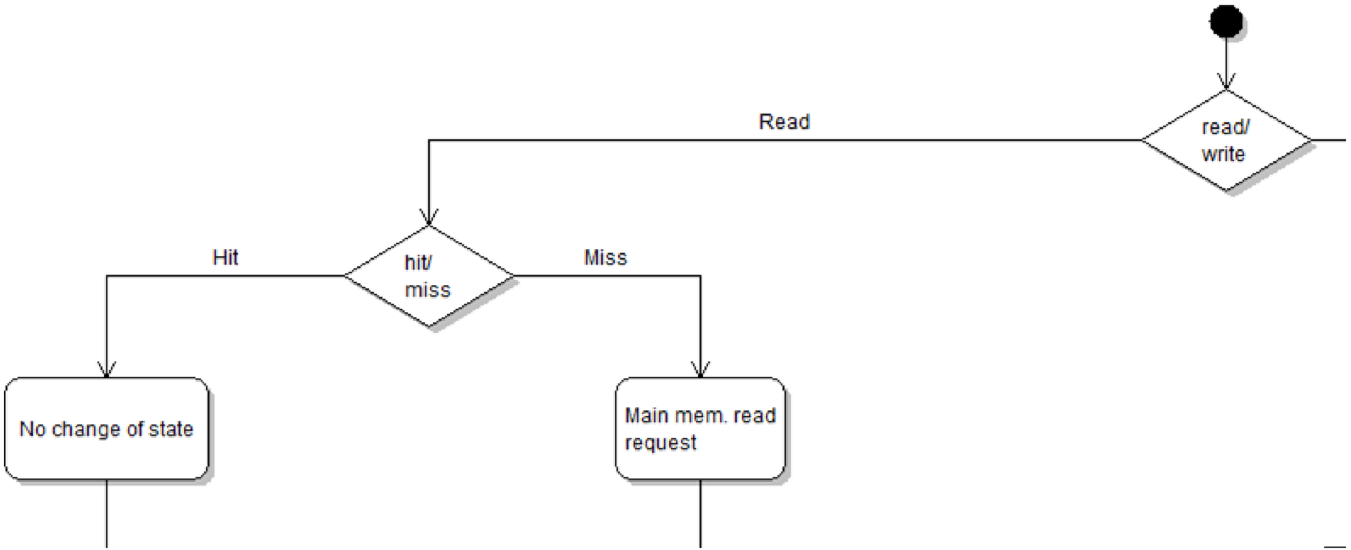
- Caches monitor address access by other caches for addresses they are interested in
- When other caches attempt to access addresses this cache knows about, it can respond by invalidating local caches or writing back modified data
- Allows caches to share data directly

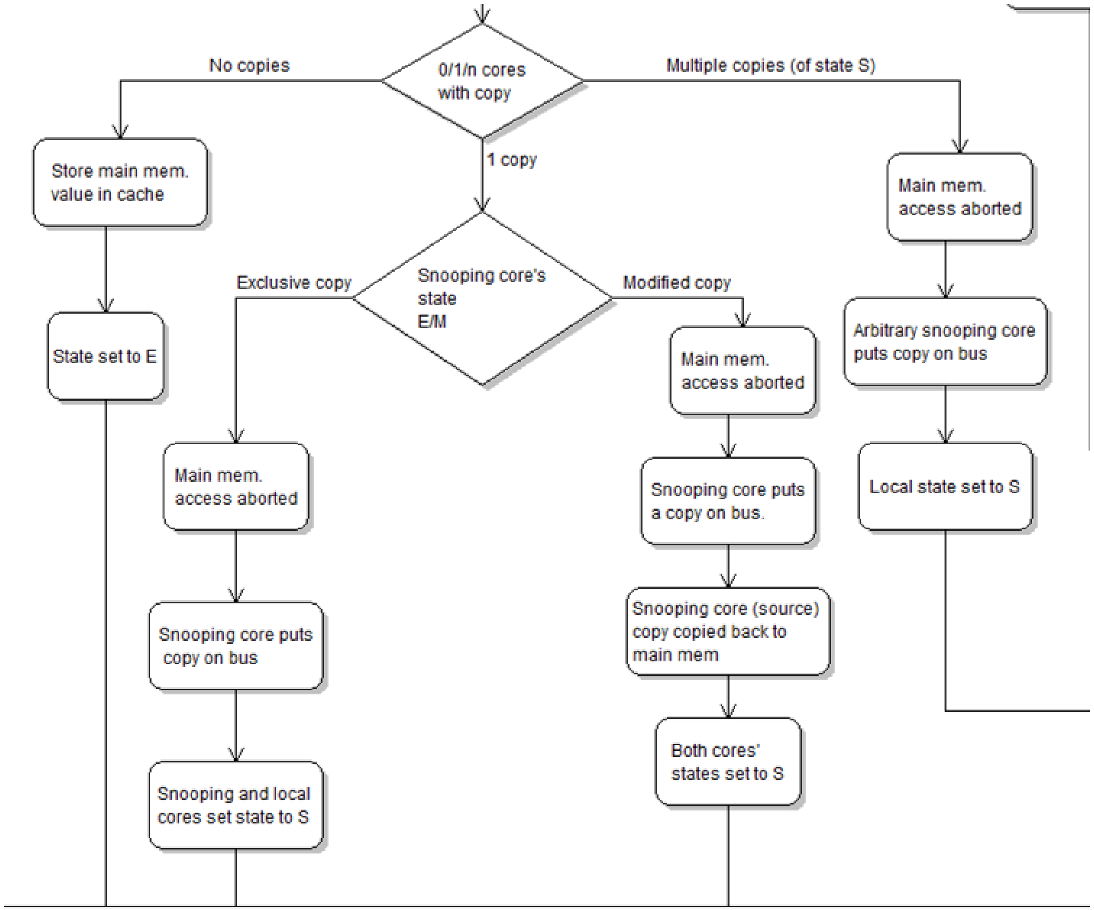
Cache Coherency - MESI

- Much bigger problem with multiple cores
- Standard Coherency Protocol is MESI:
  - M: Modified
  - E: Exclusive
  - S: Shared
  - I: Invalid
- Used in Cortex-A57

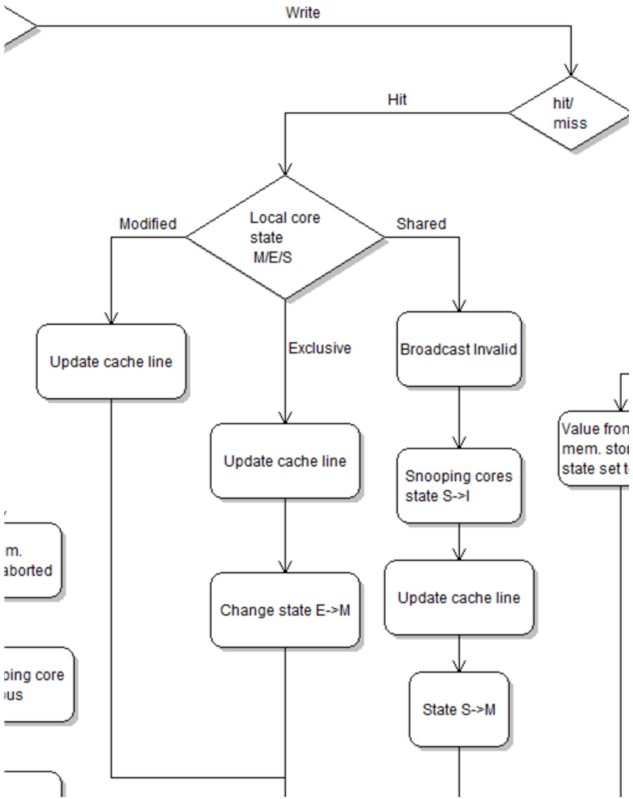


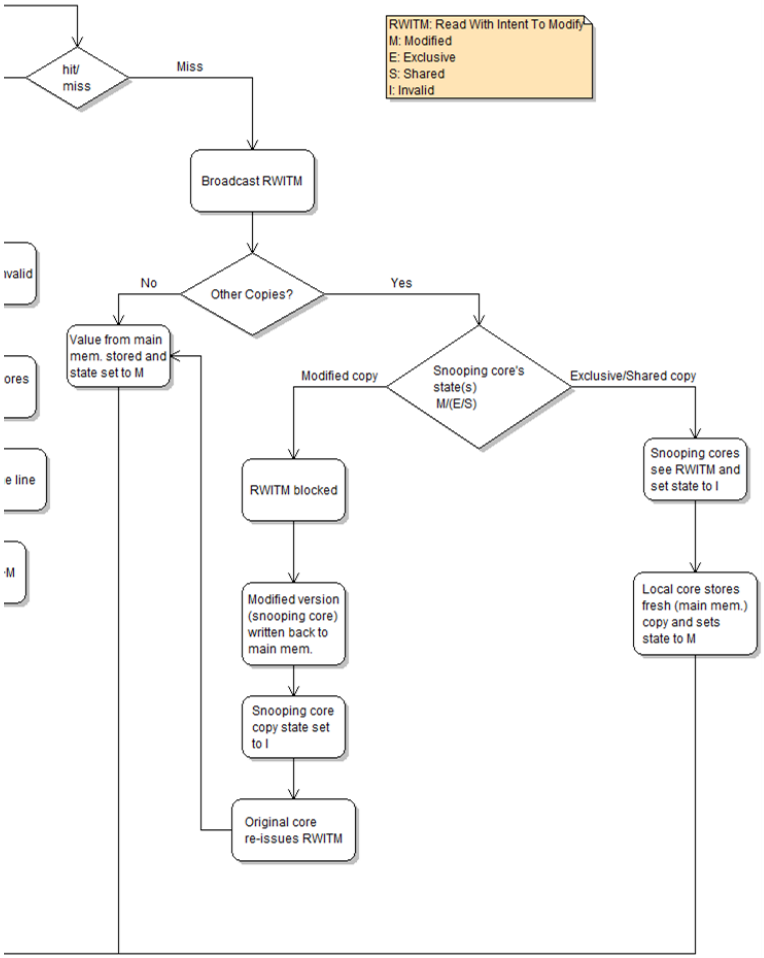
read/ write
Read
Write
hit/ miss
Hit
Miss
No change of state
Main mem. read request
0/1/n cores with copy
No copies
1 copy
Multiple copies (of state S)
Store main mem. value in cache
State set to E
Snooping core's state E/M
Exclusive copy
Modified copy
Main mem. access aborted
Snooping core puts copy on bus
Snooping and local cores set state to S
Main mem. access aborted
Snooping core puts a copy on bus.
Snooping core (source) copy copied back to main mem
Both cores' states set to S
Main mem. access aborted
Arbitrary snooping core puts copy on bus
Local state set to S
hit/ miss
Hit
Miss
Local core state M/E/S
Modified
Exclusive
Shared
Update cache line
Update cache line
Change state E->M
Broadcast Invalid
Snooping cores state S->I
Update cache line
State S->M
Broadcast RWITM
Other Copies?
No
Yes
Value from main mem. stored and state set to M
Snooping core's state(s) M/(E/S)
Modified copy
Exclusive/Shared copy
RWITM blocked
Modified version (snooping core) written back to main mem.
Snooping core copy state set to I
Original core re-issues RWITM
Snooping cores see RWITM and set state to I
Local core stores fresh (main mem.) copy and sets state to M
RWITM: Read With Intent To Modify
M: Modified
E: Exclusive
S: Shared
I: Invalid


Read
read/
write
Hit
hit/
miss
Miss
No change of state
Main mem. read
request



0/1/n cores with copy
No copies
1 copy
Multiple copies (of state S)
Store main mem. value in cache
State set to E
Snooping core's state E/M
Exclusive copy
Modified copy
Main mem. access aborted
Snooping core puts copy on bus
Snooping and local cores set state to S
Main mem. access aborted
Snooping core puts a copy on bus.
Snooping core (source) copy copied back to main mem
Both cores' states set to S
Main mem. access aborted
Arbitrary snooping core puts copy on bus
Local state set to S


Write
hit/ miss
Hit
Local core state M/E/S
Modified
Shared
Exclusive
Update cache line
Broadcast Invalid
Update cache line
Snooping cores state S->I
Change state E->M
Update cache line
State S->M
Value fron mem. stoi state set t
m. aborted
ping core ous


RWITM: Read With Intent To Modify
M: Modified
E: Exclusive
S: Shared
I: Invalid
hit/ miss
Miss
Broadcast RWITM
Other Copies?
No
Yes
Value from main mem. stored and state set to M
Snooping core's state(s) M/(E/S)
Modified copy
Exclusive/Shared copy
RWITM blocked
Modified version (snooping core) written back to main mem.
Snooping core copy state set to I
Original core re-issues RWITM
Snooping cores see RWITM and set state to I
Local core stores fresh (main mem.) copy and sets state to M

Cache Coherency - MOESI

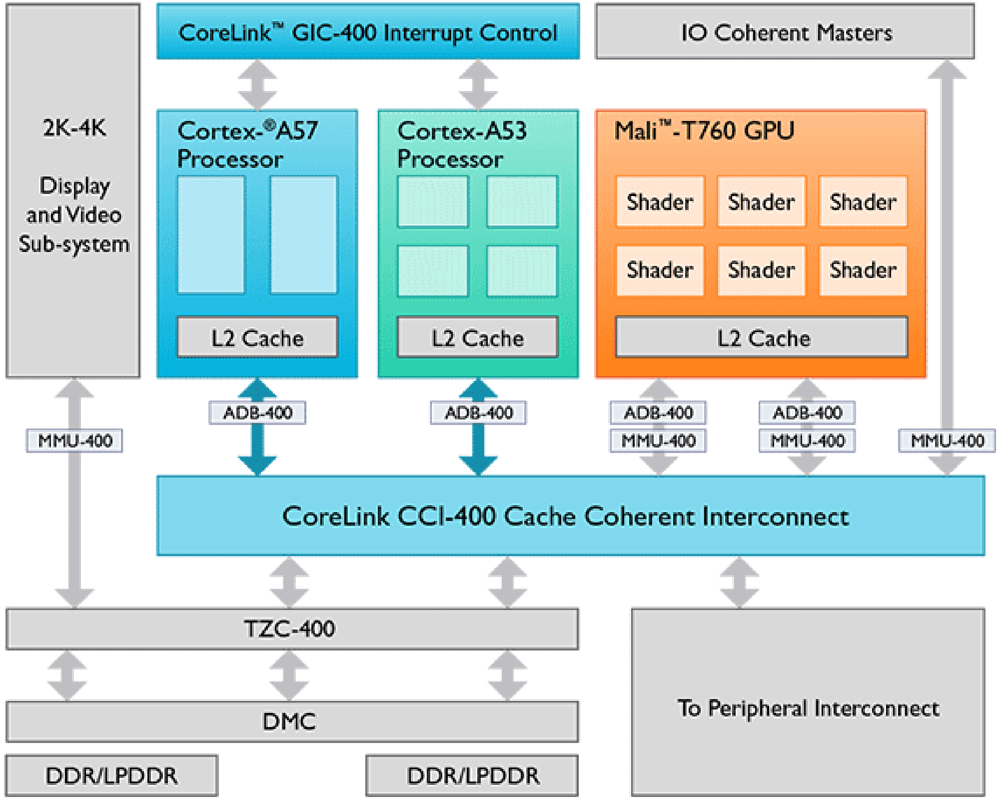
- Used in the A53
- All the same, except:
    - O - Owned. Possibly shared to other cores, but is dirty, and this core has exclusive modify access
    - Shared - Can be clean or dirty

big.LITTLE

- Combines high performance Cortex-A57 cores with low power Cortex-A53 cores
- Can seamlessly move processes between cores based on needs
- Supported by Linux 3.11

CoreLink CCI-400

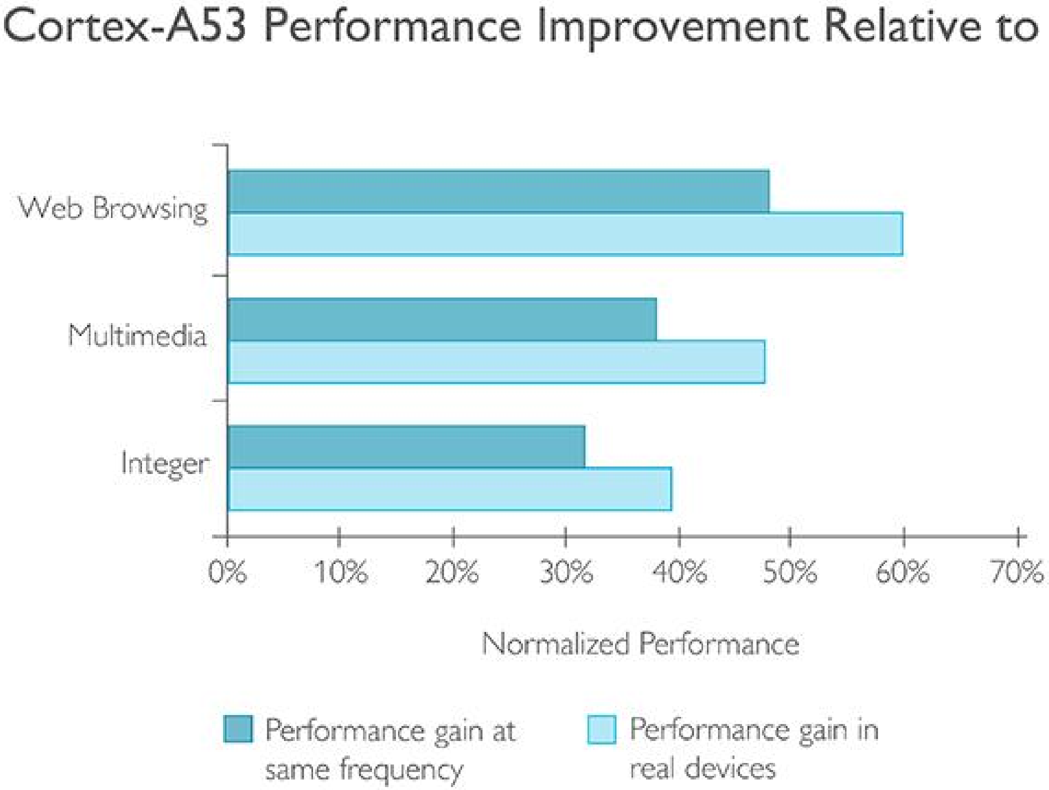
- Manages cache coherency across big and LITTLE cores
- Supports 128 bit wide data at 10 GB/s



Cortex A53 - Applications

- Smartphones (big.LITTLE)
- wireless networking infrastructure
- low-power servers
- smart TVs

A53 vs A7



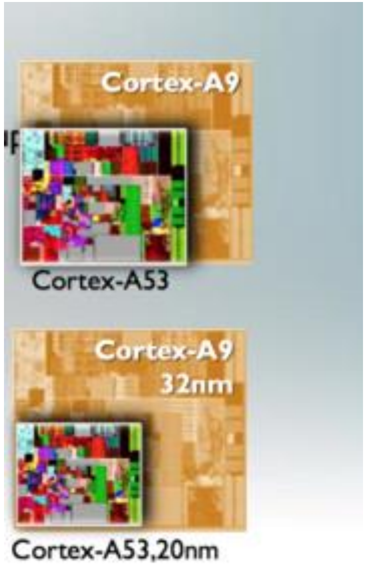
Cortex-A53 Performance Improvement Relative to Cortex-A7
Web Browsing
Multimedia
Integer
0%
10%
20%
30%
40%
50%
60%
70%
Normalized Performance
Performance gain at same frequency
Performance gain in real devices

A57 - Applications

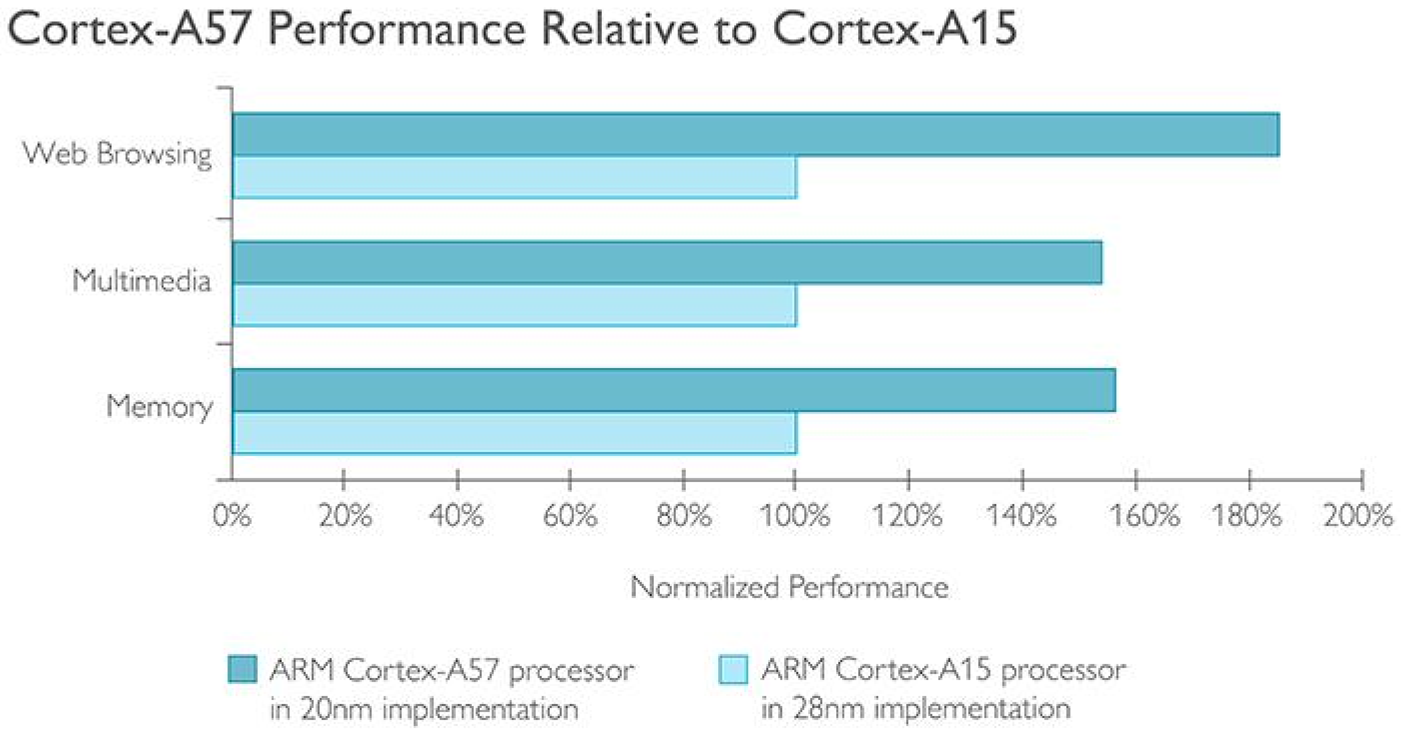
- Premium smartphones
- enterprise servers
- home servers
- wireless infrastructure
- digital tv

Comparison- A53 vs A9

- A53 is the same performance
- 40% smaller
- 4x as efficient for matched performance

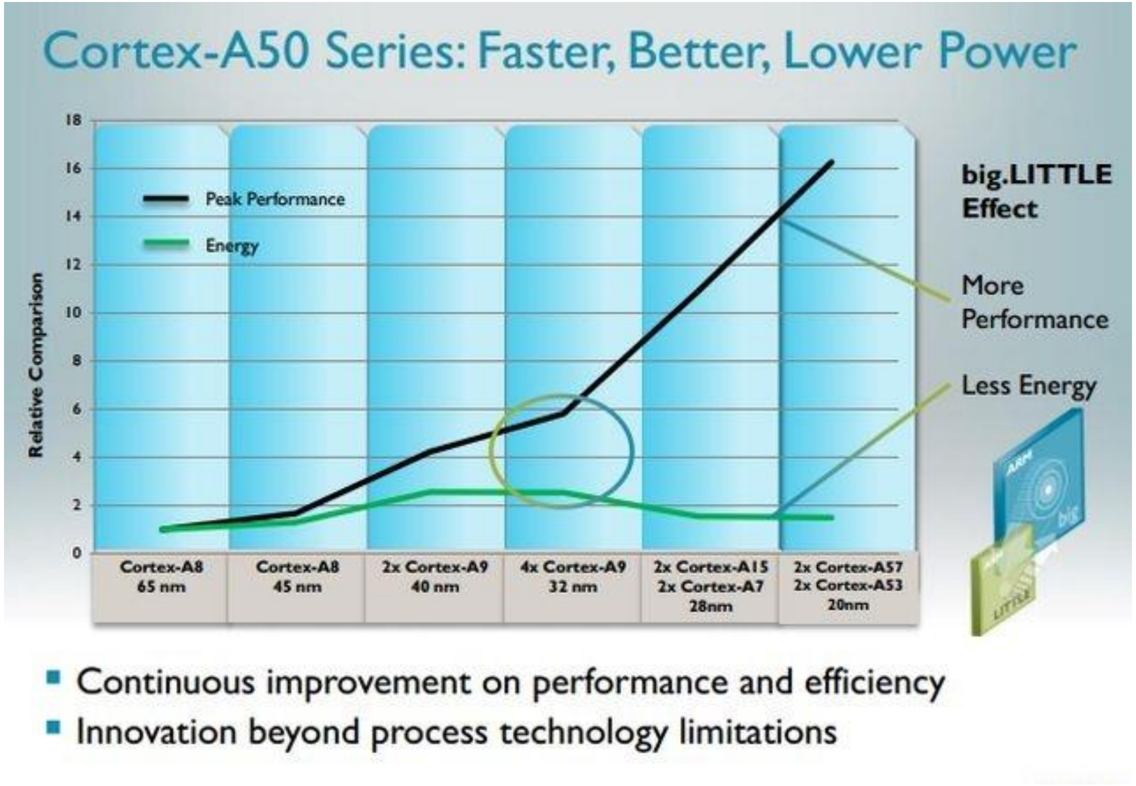


A57 vs A15



Cortex-A57 Performance Relative to Cortex-A15
Web Browsing
Multimedia
Memory
0%
20%
40%
60%
80%
100%
120%
140%
160%
180%
200%
Normalized Performance
ARM Cortex-A57 processor in 20nm implementation
ARM Cortex-A15 processor in 28nm implementation

Cortex-A50 Series



AMD Opteron A1100

- Codename Seattle
- Announced January 2014
- Based around ARM-A57
- Networking and I/O over raw performance

General Architecture

"SEATTLE" SOC OVERVIEW

AMD

Power Efficient Cores
- Up to Eight ARM Cortex-A57 cores
- Up to 4MB shared L2 cache total

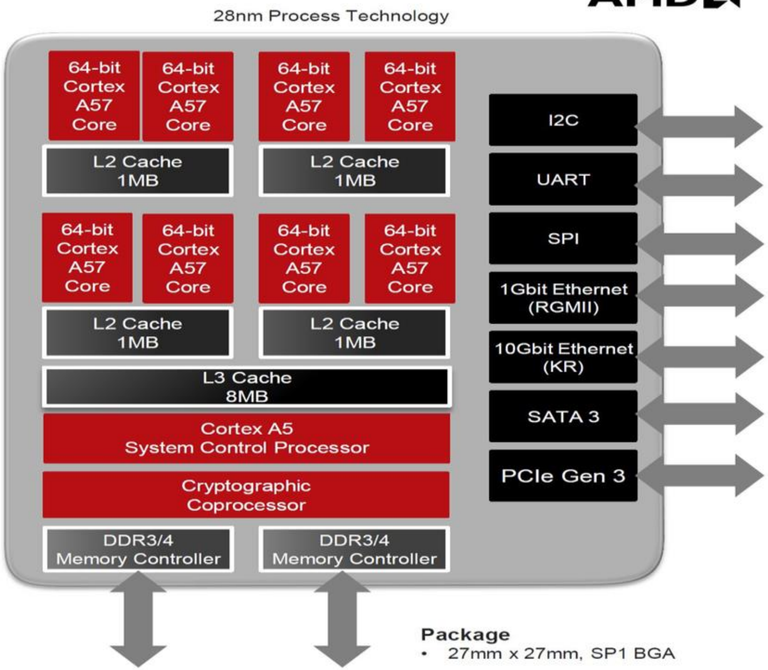
Cache Coherent Network
- Full cache coherency
- 8MB L3 cache
- SMMU: I/O address mapping and protection

High Performance, Flexible Memory
- Two 64-bit DDR3/4 channels with ECC
- Two DIMMs/channel up to 1866Mhz
- SODIMM, UDIMM, RDIMM support
- Up to 128GB per CPU

Highly Integrated I/O
- 8x SATA 3 (6Gb/s) ports
- Two 10GBASE-KR Ethernet ports
- 8 lanes PCI-Express® Gen 3, supports x8, x4, x2

System Control Processor
- TrustZone® technology for enhanced security
- Dedicated 1GbE system management port (RGMII)
- SPI, UART, I2C interfaces

Cryptographic Coprocessor
- Separate Cryptographic algorithm engine for offloading encryption, decryption, compression, decompression computations

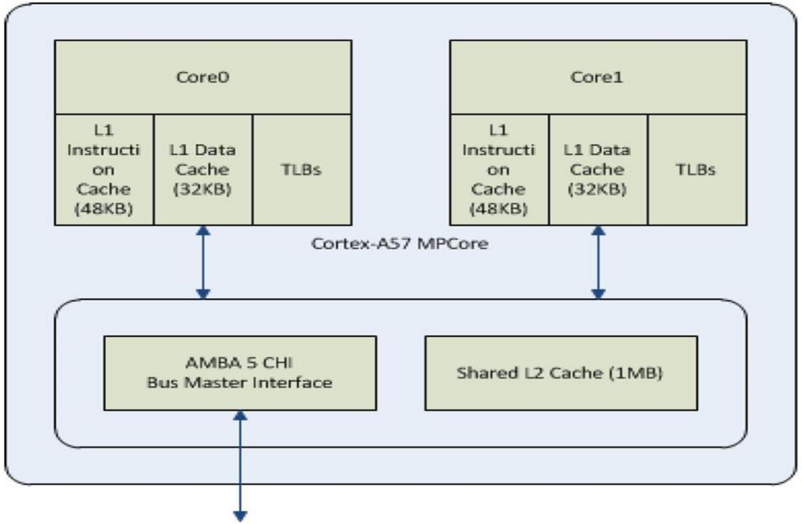


Package
- 27mm x 27mm, SP1 BGA

Basic Core Architecture

“SEATTLE” CORTEX-A57 MPCORE, L1/L2/L3 CACHES

AMD



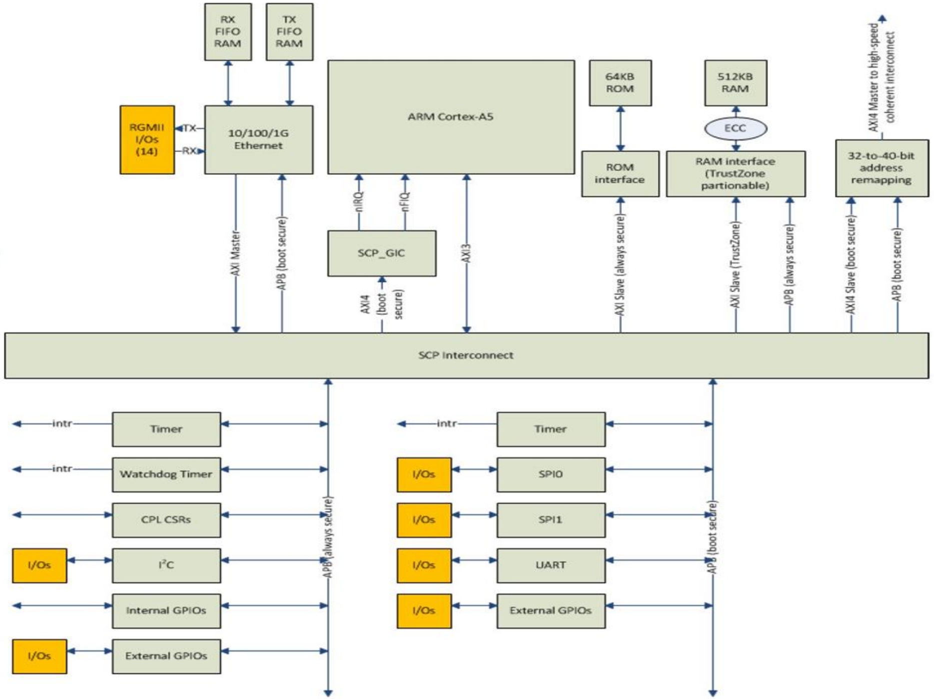
- 2 X A57 cores plus shared 1MB L2 cache
- ARMv8-A architecture
- Caches (64 byte cache line size)
  - 48KB Level 1 Instruction Caches, 3-way set associative, parity protected
  - 32KB Level 1 Data Caches, 2-way, ECC protected
  - 1MB shared Level 2 Cache, 16-way, ECC protected
  - 8MB shared Level 3 Cache, 16-way, ECC protected (Snoop filter integrated with L3 cache)
- Cryptographic instructions included
- AMBA 5 CHI interface to rest of system
- CoreSight debug, Cross-Trigger Interface (CTI) and Embedded Trace Macrocell (ETM) also in MPCore
- Interface to Generic Interrupt Controller (GIC) also in MPCore

System Control Processor

SYSTEM CONTROL PROCESSOR

AMD

- System Control Processor (SCP) is an ARM Cortex-A5 processor with attached ROM, RAM and I/O devices
- SCP is used to control power, configure the system, initiate booting, and act as a service processor for system management functions
- SCP is effectively a small system-on-a-chip (SOC) within the larger "Seattle" SOC
- SCP looks like an I/O device to the rest of the system



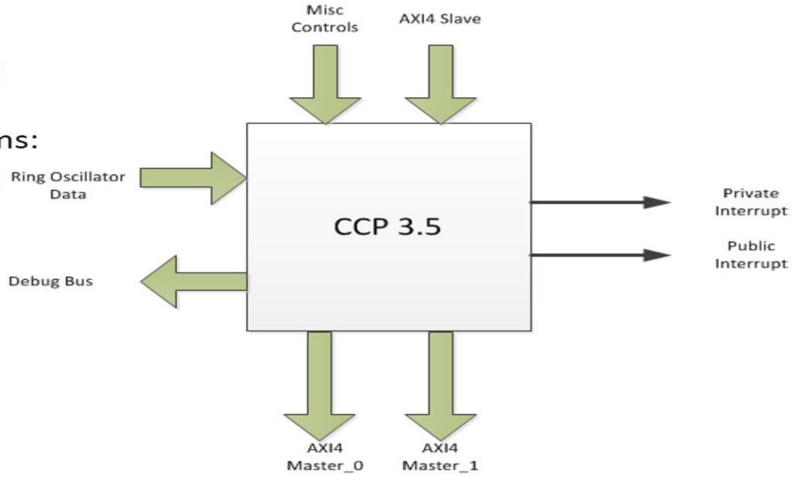
Cryptographic Processor

CRYPTOGRAPHIC COPROCESSOR (CCP)

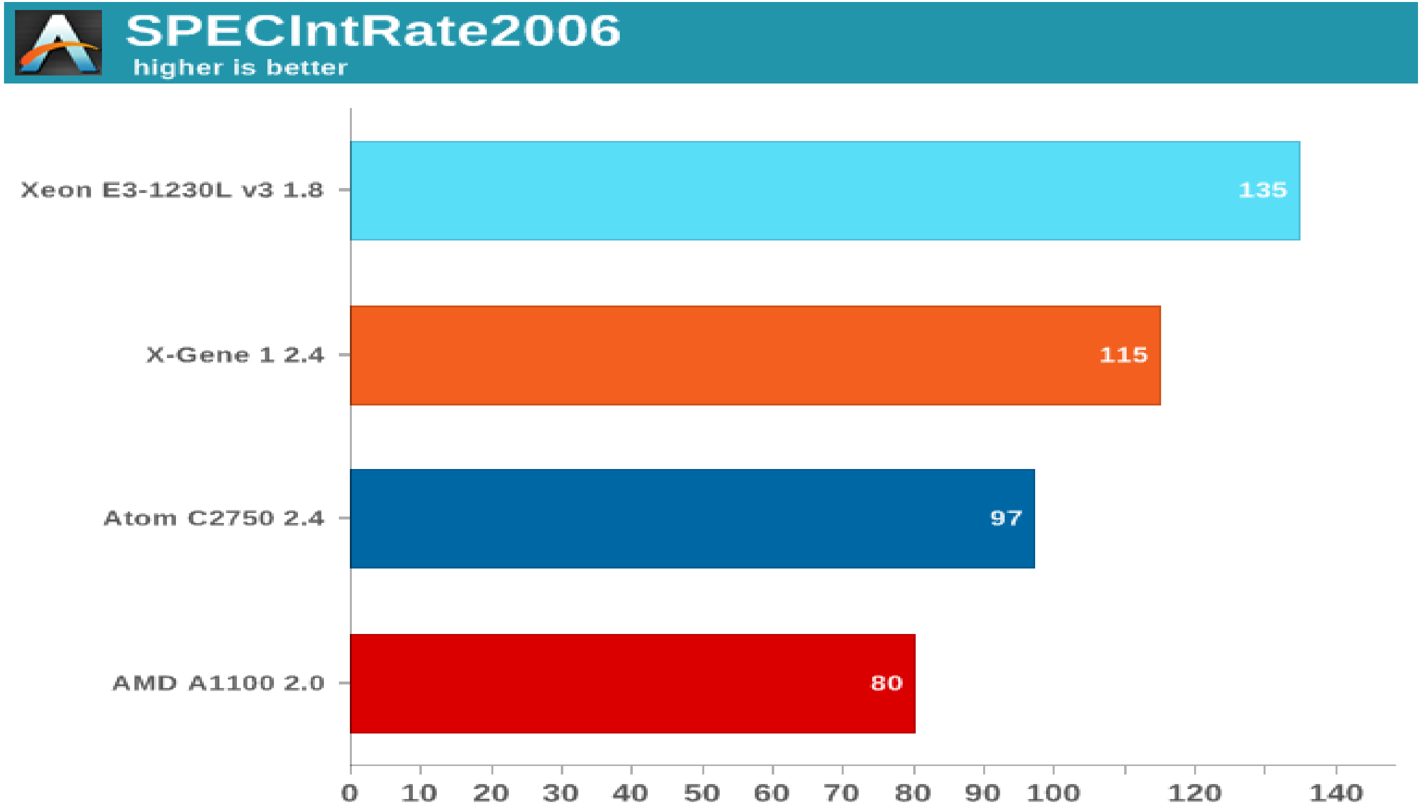
COMPUTE OFFLOAD HARDWARE

AMD

- The Cryptographic Coprocessor is a dedicated accelerator for the following encryption/decryption and compression/decompression algorithms:
  - Advanced Encryption Standard (AES)
  - Elliptic Curve Cryptography (ECC)
  - RSA
  - Secure Hash Algorithm (SHA)
  - Zlib compression
  - Zlib decompression
  - True Hardware Random Number Generator
- Available to the System Control Processor (SCP) for secure and non-secure processing
- Available to the Cortex-A57 cores for non-secure processing



Performance Comparison



Adjusted Performance Comparison