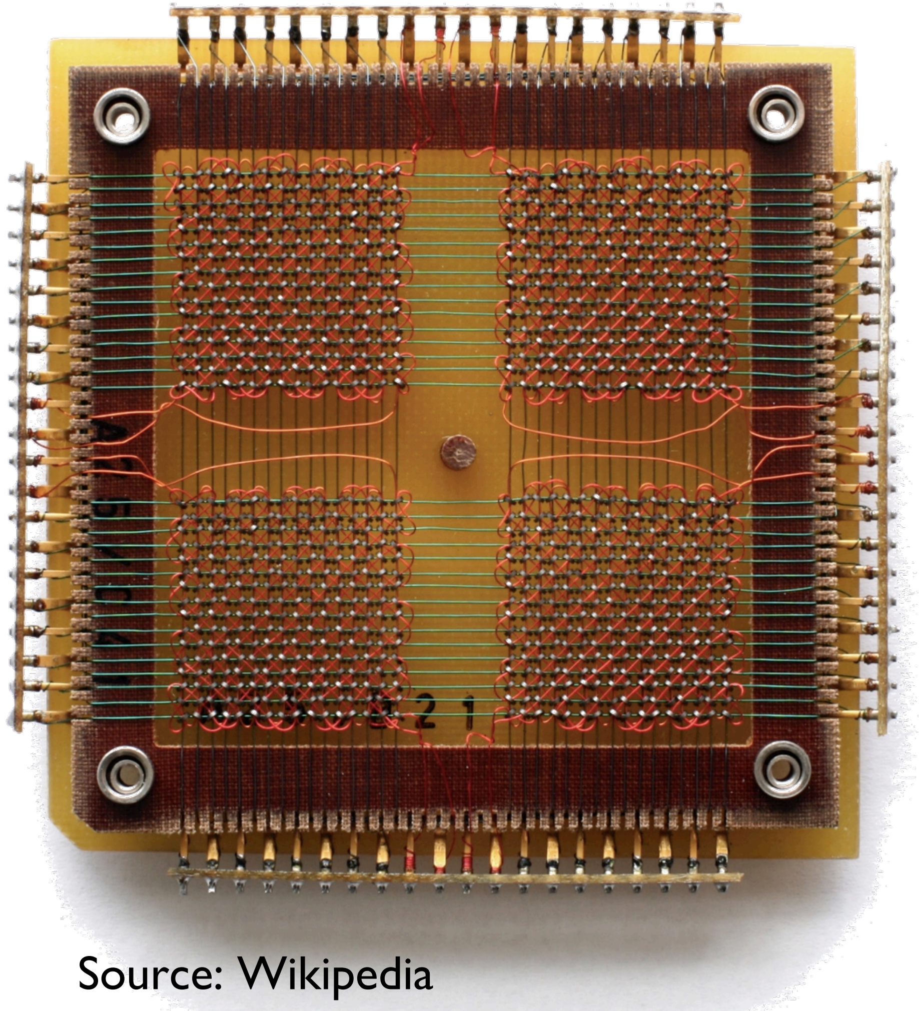


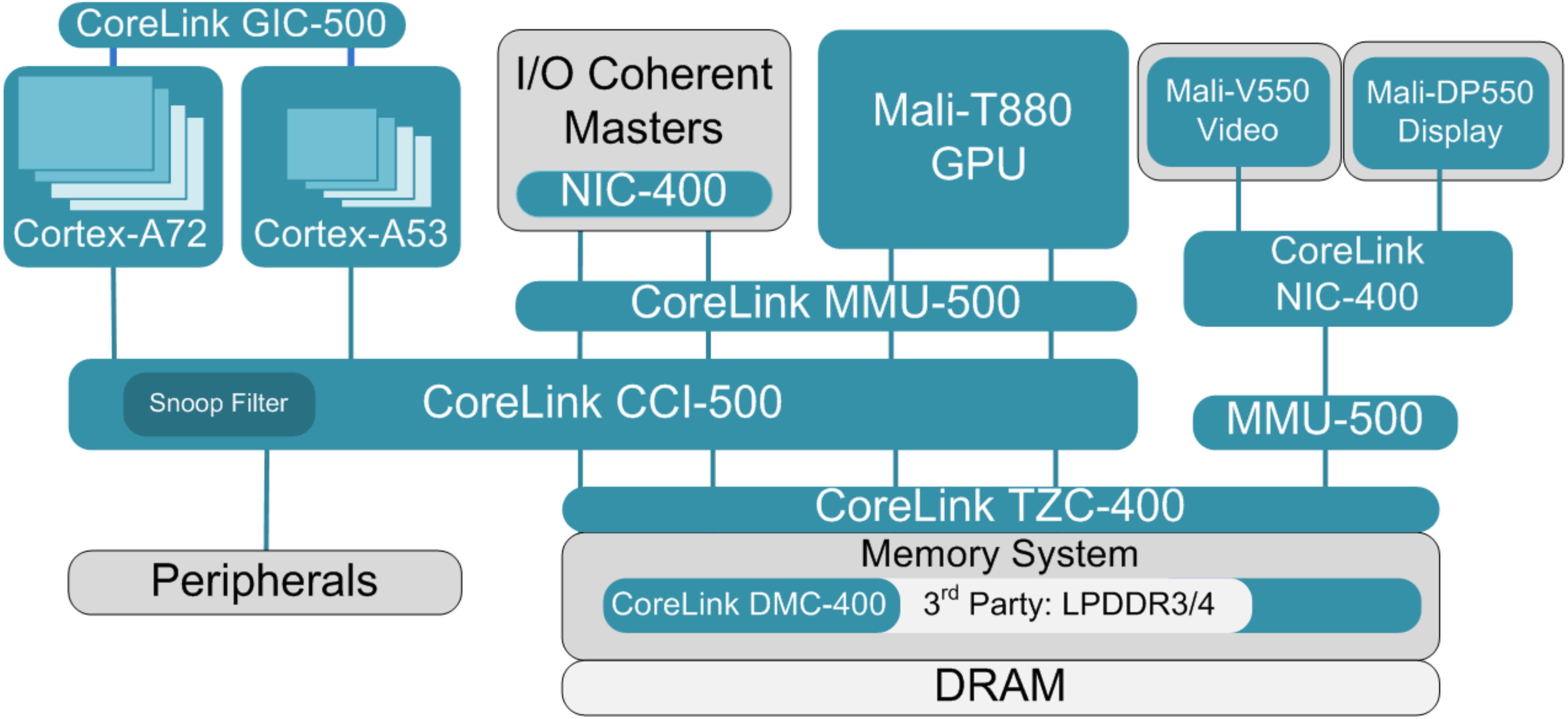
Source: Wikipedia

Classic Memory System
Revisited

Andreas Hansson
ARM Research

gem5 User Workshop 2015

What are we modeling?

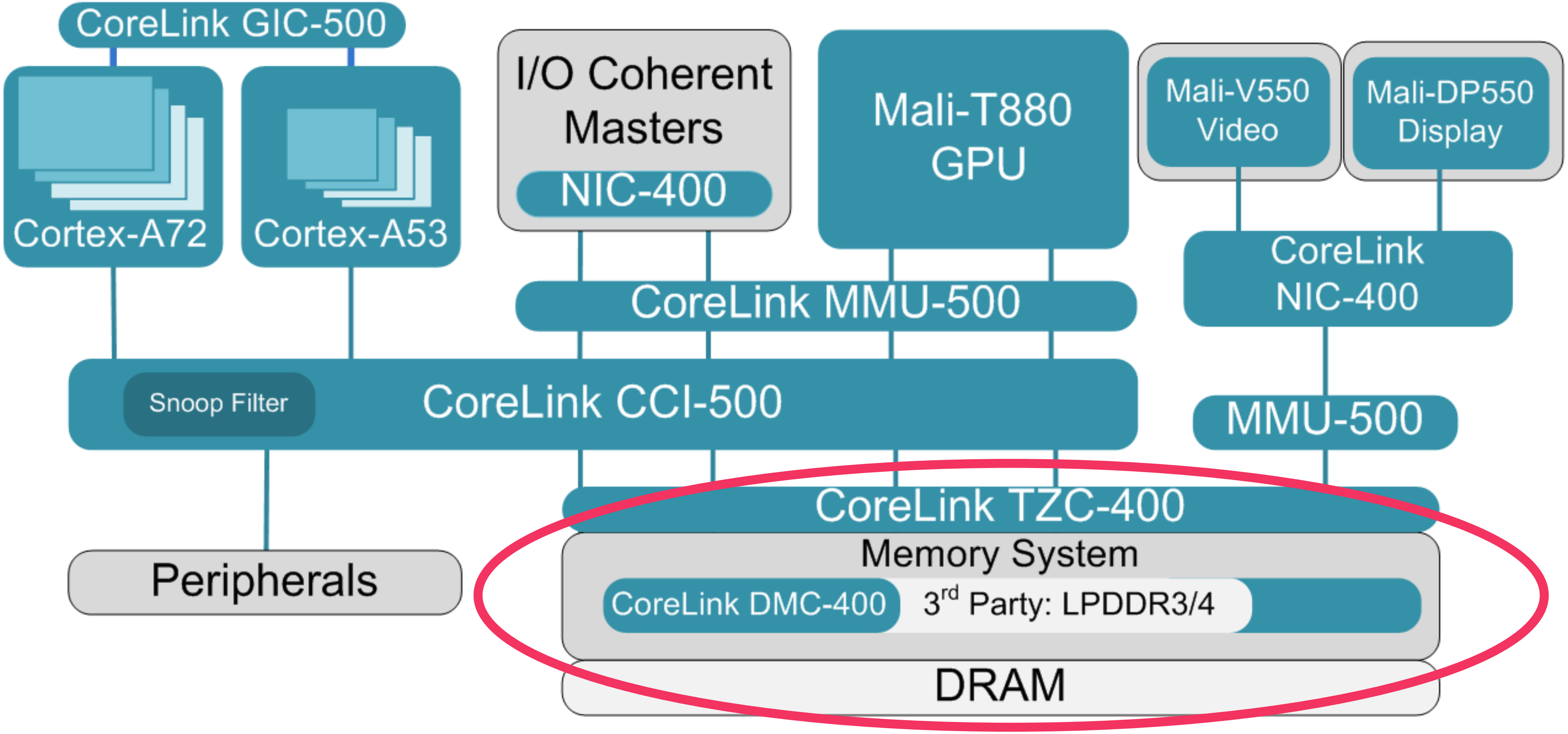


Source: ARM

Key changes and additions

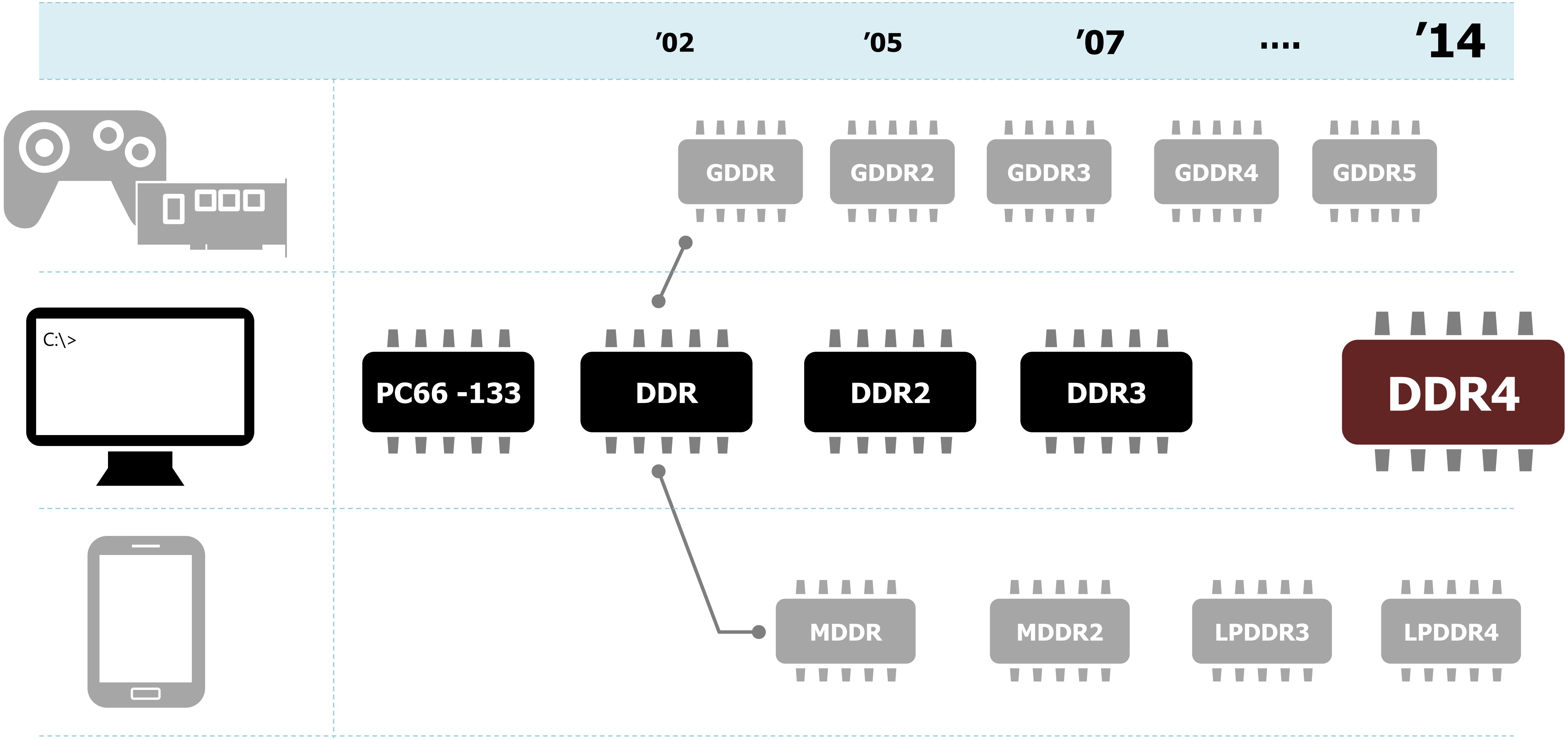
- DRAM controller refinements
  - New DRAM features, power modeling
- Crossbar extensions
  - Interleaving and hashing
- Snoop filter addition
  - Steering snoops, tracking evictions
- Correctness checking
  - Memory-model checker and soak tests
- Performance tuning
  - Transaction support, cache latencies

What are we modeling?



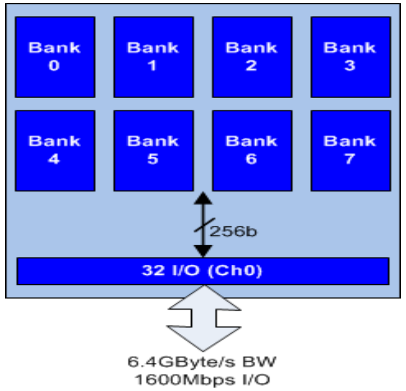
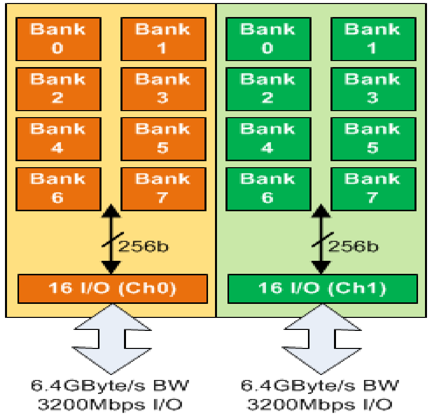
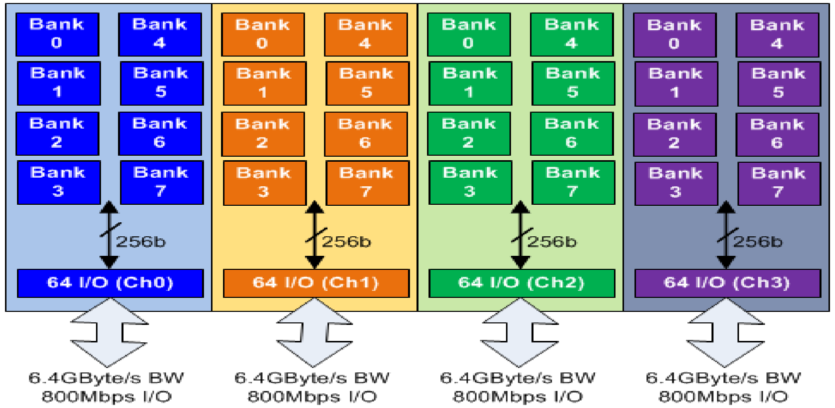
Source: ARM

DRAM evolution



Source: Samsung

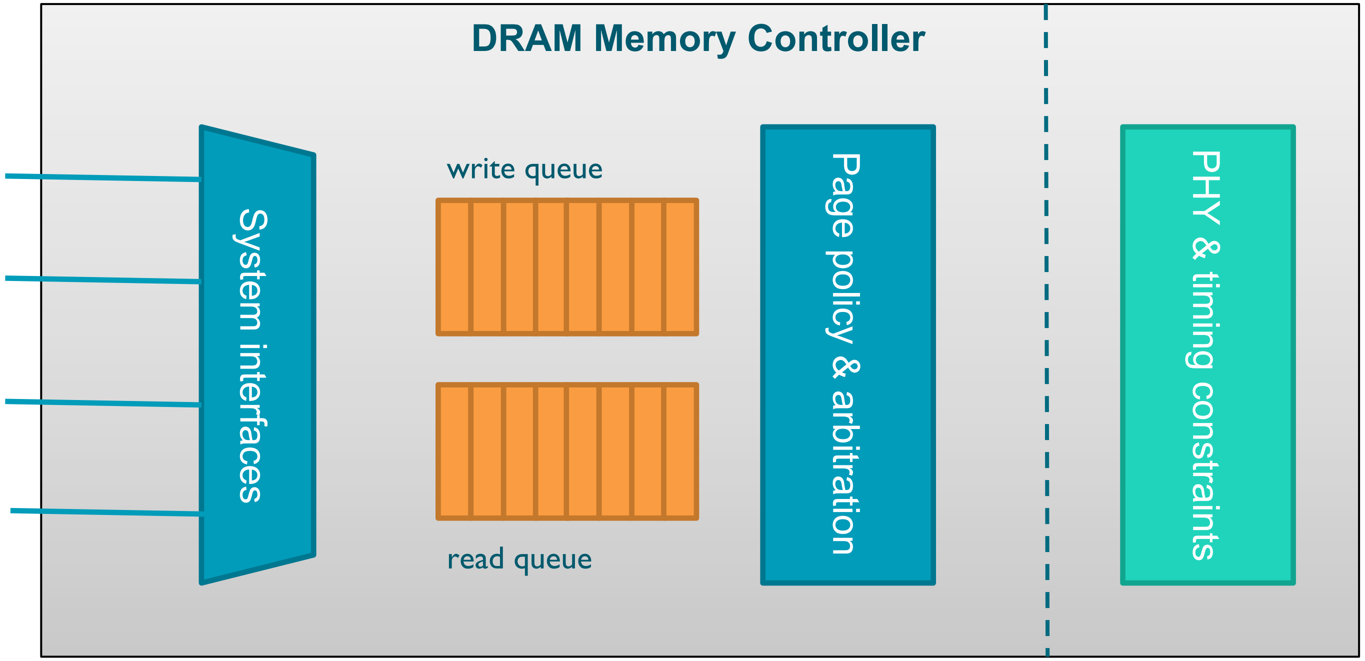
Same same…but different

	LPDDR3 & LPDDR3E	LPDDR4	Wide IO2
Die Organization	1ch X 8 banks X 32 IO  Bank 0 Bank 1 Bank 2 Bank 3 Bank 4 Bank 5 Bank 6 Bank 7 256b 32 I/O (Ch0) 6.4GByte/s BW 1600Mbps I/O	2ch X 8banks X16 IO  Bank 0 Bank 1 Bank 2 Bank 3 Bank 4 Bank 5 Bank 6 Bank 7 256b 16 I/O (Ch0) Bank 0 Bank 1 Bank 2 Bank 3 Bank 4 Bank 5 Bank 6 Bank 7 256b 16 I/O (Ch1) 6.4GByte/s BW 3200Mbps I/O 6.4GByte/s BW 3200Mbps I/O	4ch X 8banks X 64 IO  Bank 0 Bank 4 Bank 1 Bank 5 Bank 2 Bank 6 Bank 3 Bank 7 256b 64 I/O (Ch0) Bank 0 Bank 4 Bank 1 Bank 5 Bank 2 Bank 6 Bank 3 Bank 7 256b 64 I/O (Ch1) Bank 0 Bank 4 Bank 1 Bank 5 Bank 2 Bank 6 Bank 3 Bank 7 256b 64 I/O (Ch2) Bank 0 Bank 4 Bank 1 Bank 5 Bank 2 Bank 6 Bank 3 Bank 7 256b 64 I/O (Ch3) 6.4GByte/s BW 800Mbps I/O 6.4GByte/s BW 800Mbps I/O 6.4GByte/s BW 800Mbps I/O 6.4GByte/s BW 800Mbps I/O
Channel #	1	2	4 & 8
Bank #	8	8 per channel (16 per die)	32 per die
Density	4Gb – 32Gb	4Gb – 32Gb	8Gb – 32Gb
Page Size	4KByte	2KByte	4KByte (4ch die), 2KB (8ch die)
Max BW per die	6.4GB/s, 8.5GB/s (overclocking)	12.8GB/s, 17GB/s (overclocking)	25.6GB/s & 51.2GB/s 34GB/s & 68GB/s(overclocking)
Max IO Speed	2133Mbps	4266Mbps	1066Mbps
Signal Pin #	62 per die	66 per die	~430 per die (4ch die), ~850 per die(8ch die)
Package	POP, MCP	POP, MCP	KGD,

Source: Qualcomm

Top-down controller model

- Don't model the actual DRAM, only the timing constraints
  - DDR3/4, LPDDR2/3/4, WIO1/2, GDDR5, HBM, HMC, even PCM
  - See *src/mem/DRAMCtrl.py* and *src/mem/dram_ctrl.{hh, cc}*



Device width
Burst length
#ranks, #banks
Page size

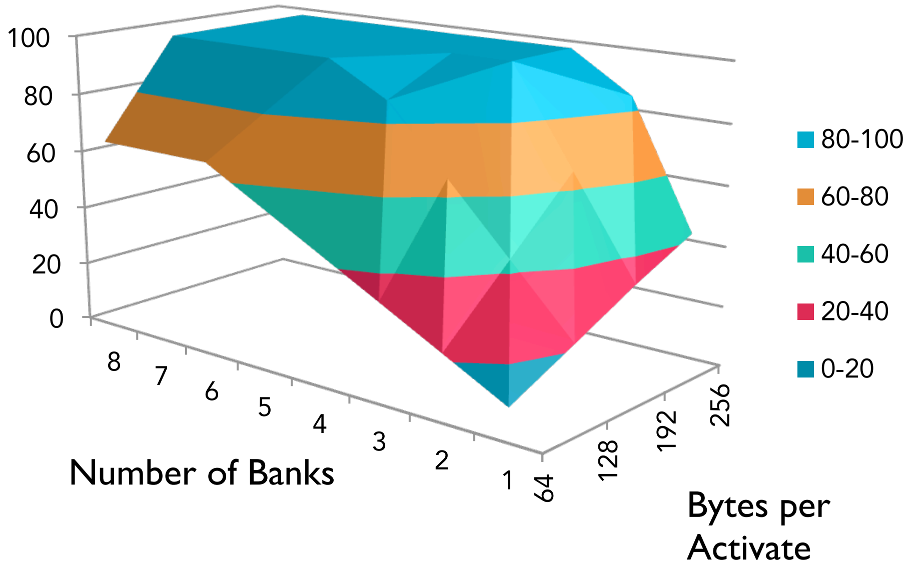
tRCD
tCL
tRP
tRAS
tBURST
tRFC & tRFEI
tWTR
tRRD
tFAW/tTAW
…

Hansson et al, *Simulating DRAM controllers for future system architecture exploration,* ISPASS'14

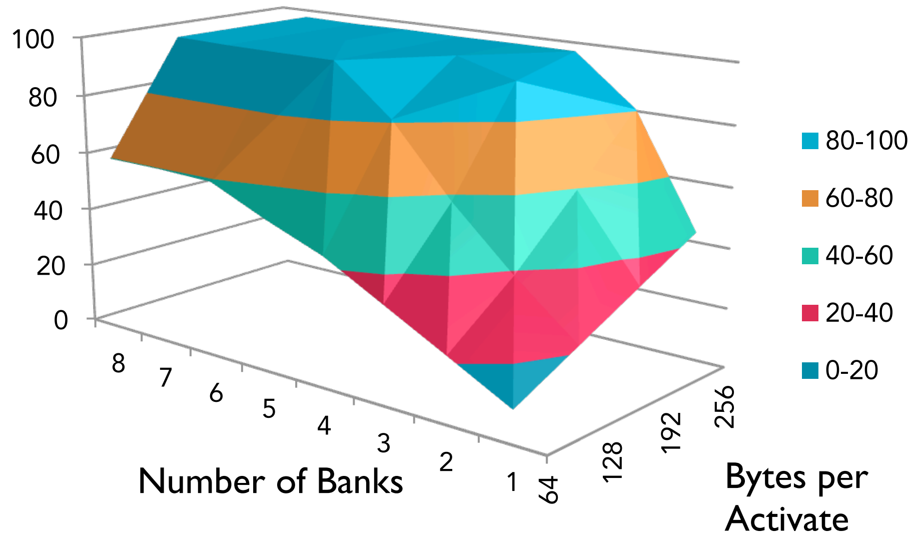
Controller model correlation

- Comparing with a real memory controller
  - Synthetic traffic sweeping bytes per activate and number of banks
  - *See configs/dram/sweep.py* and *util/dram_sweep_plot.py*

gem5 model

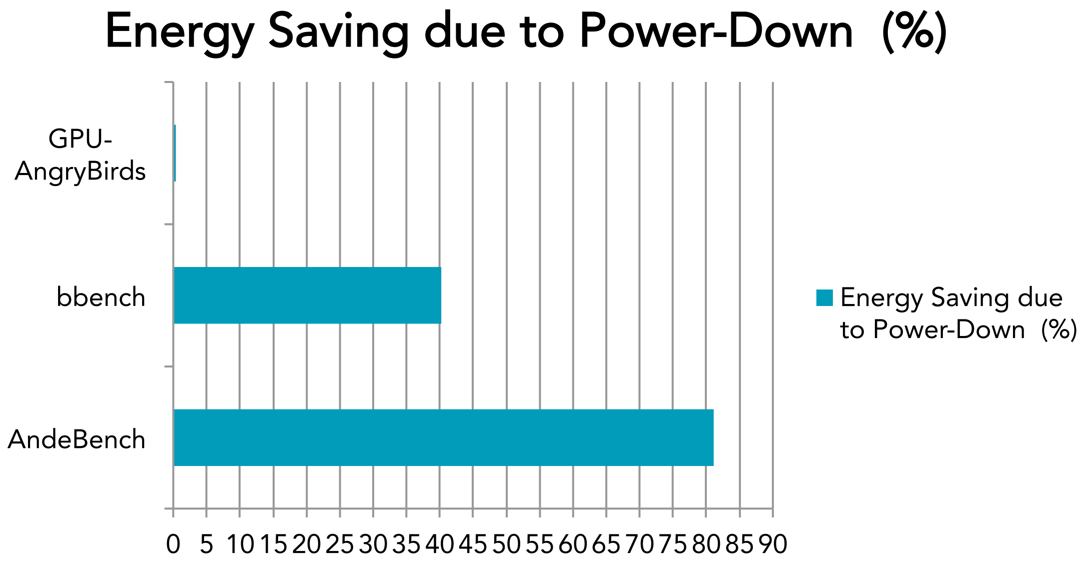


Real memory controller

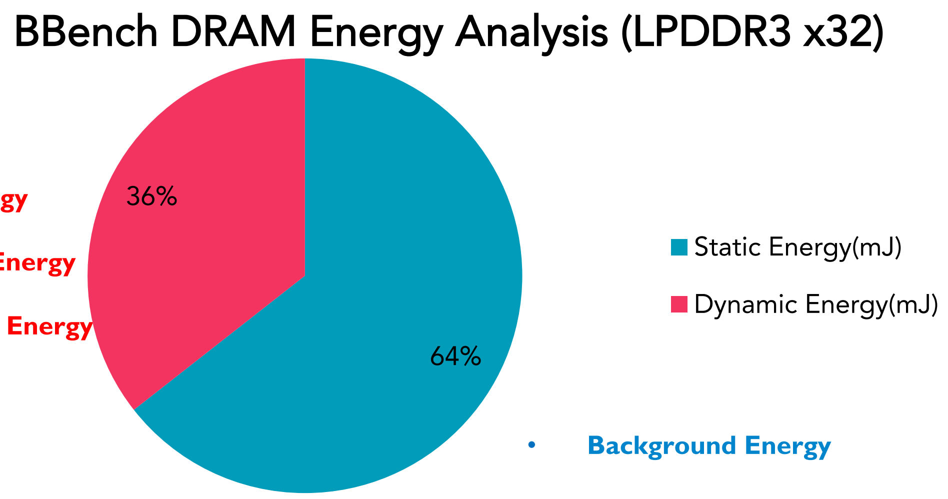


DRAM power modeling

- DRAM accounts for a large portion of system power
  - Need to capture power states, and system impact
- Integrated model opens up for developing more clever strategies
  - DRAMPower adapted and adopted for gem5 use-case



Energy Saving due to Power-Down (%)



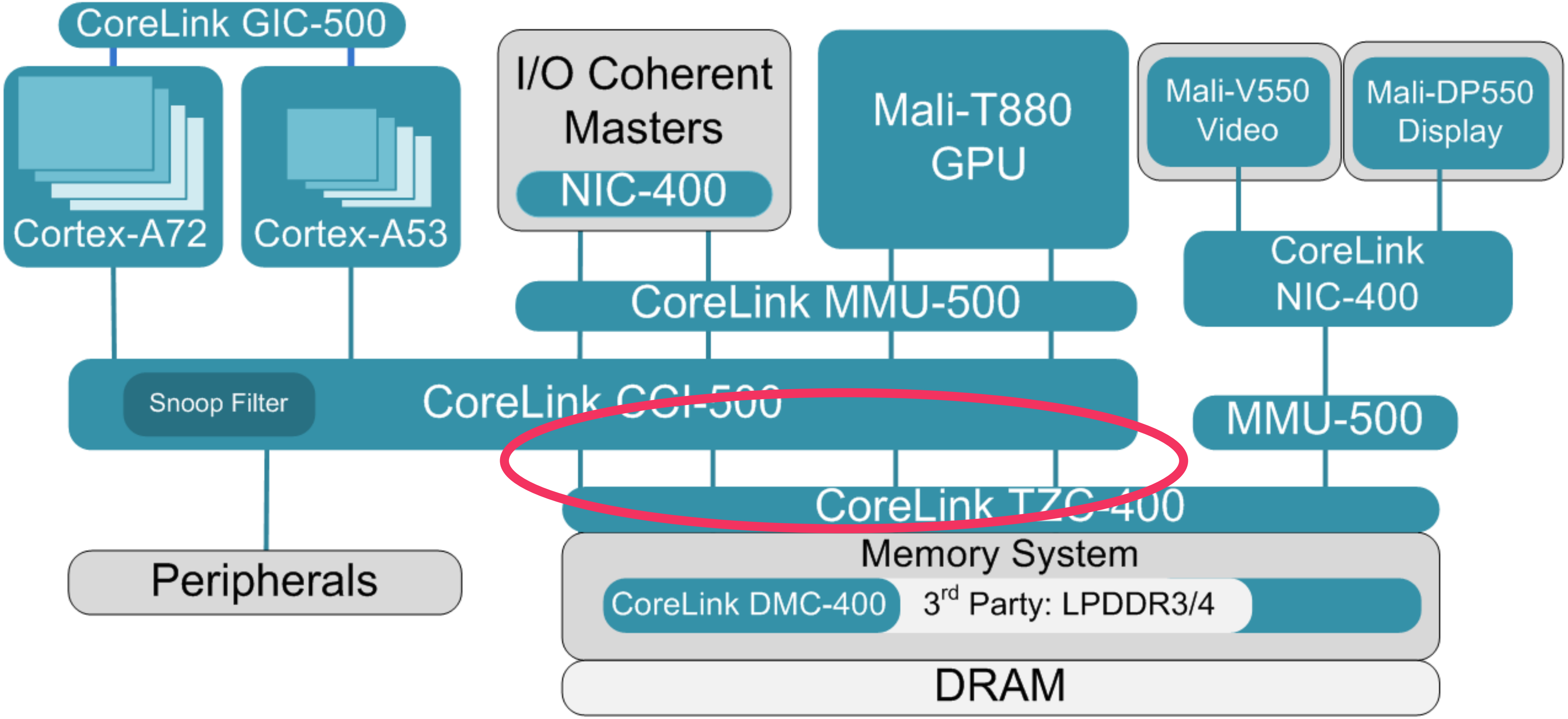
BBench DRAM Energy Analysis (LPDDR3 x32)

- Active Energy
- Precharge Energy
- Read/Write Energy

- Background Energy
- Refresh Energy

Naji et al, *A High-Level DRAM Timing, Power and Area Exploration Tool,* SAMOS'15

What are we modeling?



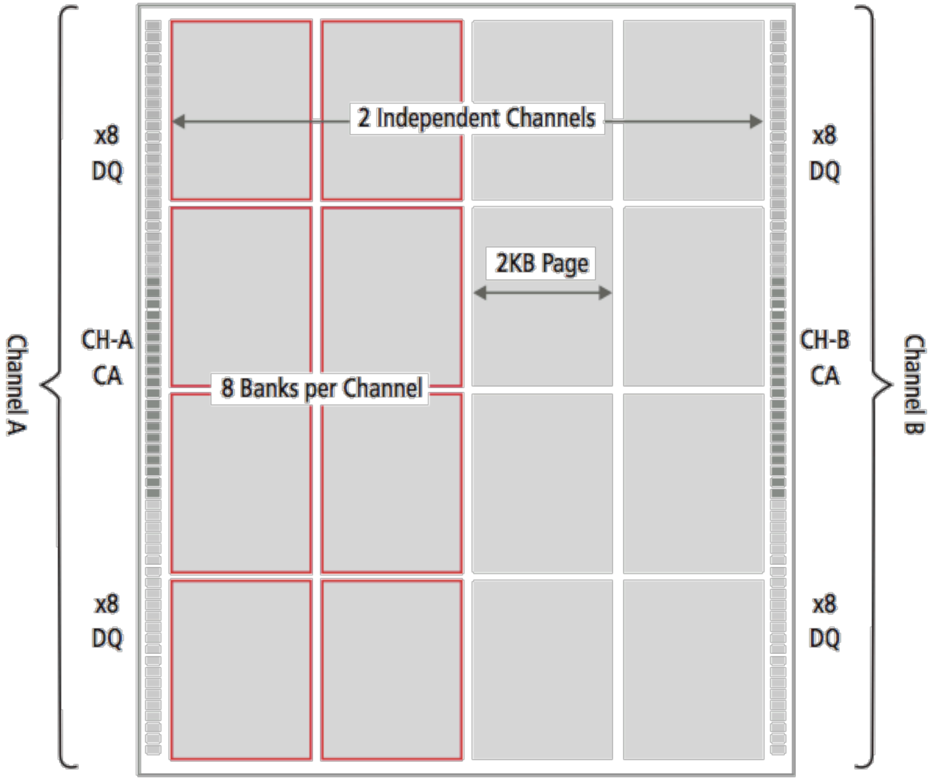
Source: ARM

Key changes and additions

- DRAM controller refinements
  - New DRAM features, power modeling
- Crossbar extensions
  - Interleaving and hashing
- Snoop filter addition
  - Steering snoops, tracking evictions
- Correctness checking
  - Memory-model checker and soak tests
- Performance tuning
  - Transaction support, cache latencies

Address interleaving

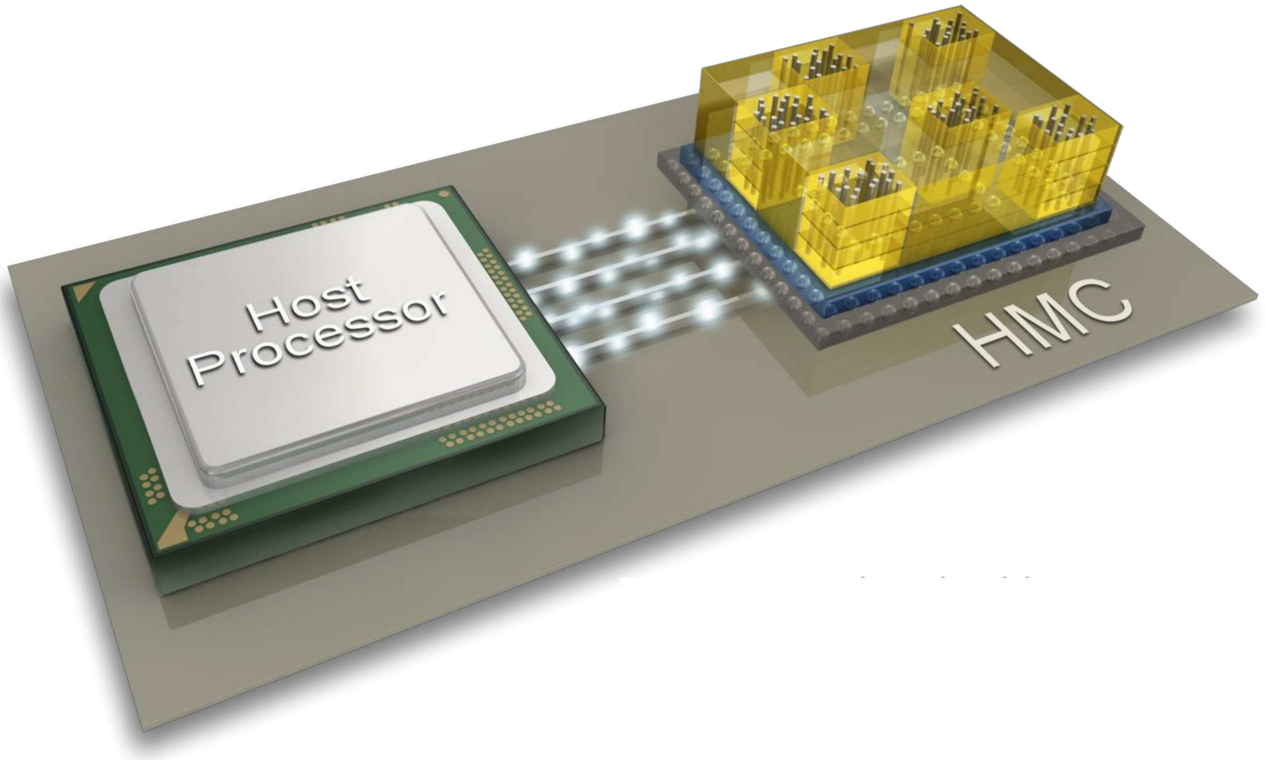
- Multi-channel memory support is essential
  - Emerging DRAM standards are multi-channel by nature (LPDDR4, WIO1/2, HBM1/2, HMC)
- Interleaving support added to address range
  - Understood by memory controller and interconnect
  - See *src/base/addr_range.hh* for matching and *src/mem/xbar.{hh, cc}* for actual usage
  - Interleaving not visible in checkpoints
- XOR-based hashing to avoid imbalances
  - Simple yet effective, and widely published
  - See *configs/common/MemConfig.py* for system configuration



Source: Micron

With a bit of creativity…

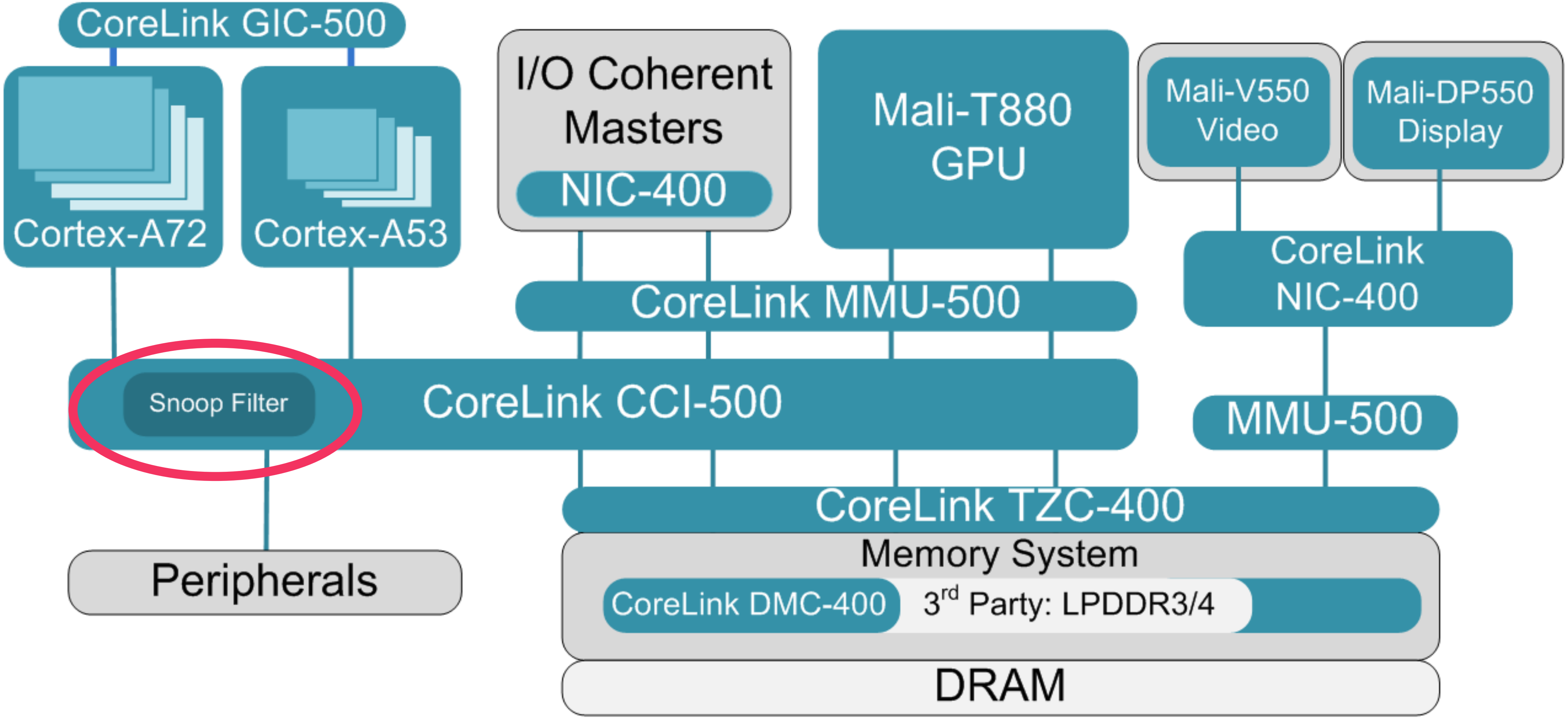
- Hybrid Memory Cube (HMC) vaults
  - 32 channels of DRAM
  - HMC DRAM configuration
- HMC base layer
  - 4 non-coherent crossbars
  - HMC interleaving configuration
- HMC links
  - Bridges or custom link classes
  - Link interleaving on the host side
- …only using what is already part of gem5



Source: Micron

ARM

What are we modeling?



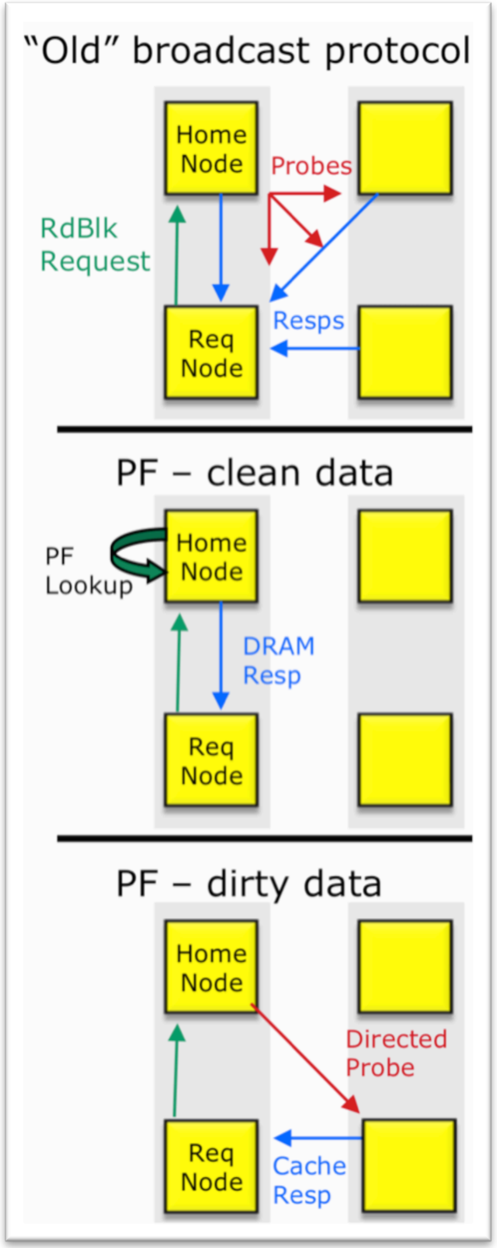
Source: ARM

Key changes and additions

- DRAM controller refinements
  - New DRAM features, power modeling
- Crossbar extensions
  - Interleaving and hashing
- Snoop filter addition
  - Tracking evictions, steering snoops
- Correctness checking
  - Memory-model checker and soak tests
- Performance tuning
  - Transaction support, cache latencies

Snoop (probe) filtering

- Broadcast-based coherence protocol
  - Incurs performance and power cost
  - Does not reflect realistic implementations
- Snoop filter goes one step towards directories
  - Track sharers, based on writeback and clean eviction
  - Direct snoops and benefit from locality
- Many possible implementations
  - Currently ideal (infinite), no back invalidations
  - Can be used with coherent crossbars on any level
  - *See src/mem/SnoopFilter.py* and *src/mem/snoop_filter.{hh, cc}**



Source: AMD

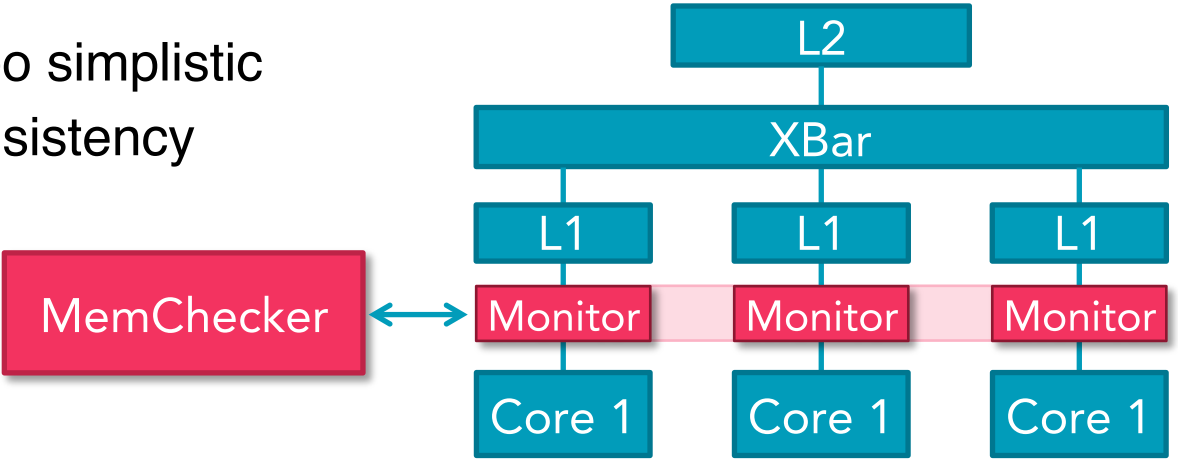
* Clean eviction patches are still on reviewboard

Key changes and additions

- DRAM controller refinements
  - New DRAM features, power modeling
- Crossbar extensions
  - Interleaving and hashing
- Snoop filter addition
  - Steering snoops, tracking evictions
- Correctness checking
  - Memory-model checker and soak tests
- Performance tuning
  - Transaction support, cache latencies

Memory system verification

- Check adherence to consistency model
  - Notion of functional reference memory is too simplistic
  - Need to track valid values according to consistency model



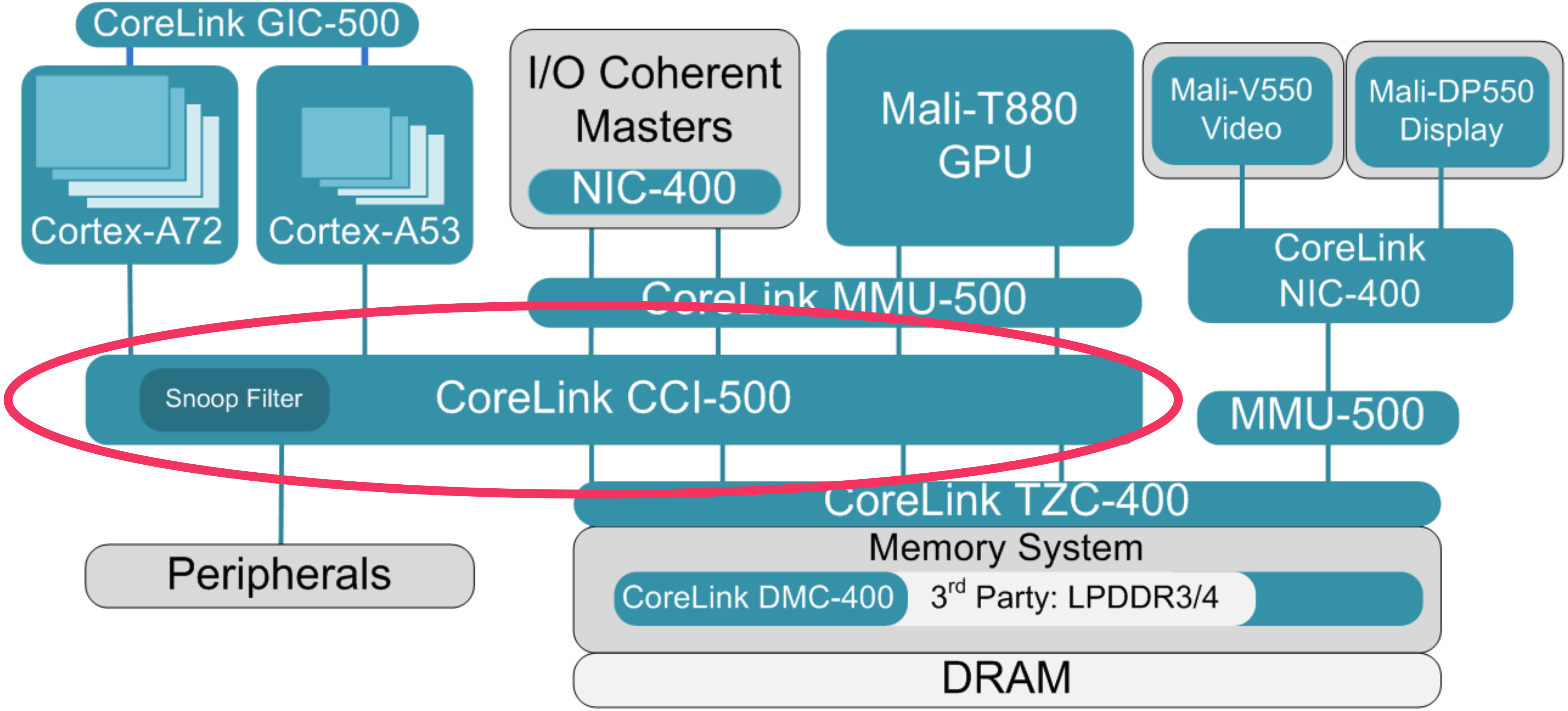
- Memory checker and monitors
  - Tracking in *src/mem/MemChecker.py* and *src/mem/mem_checker.{hh, cc}*
  - Probing in *src/mem/mem_checker_monitor.{hh, cc}*

- Revamped testing
  - Complex cache (tree) hierarchies in *configs/examples/{memtest, memcheck}.py*
  - Randomly generated soak test in *util/memtest-soak.py*
  - For any changes to the memory system, please use these

Key changes and additions

- DRAM controller refinements
  - New DRAM features, power modeling
- Crossbar extensions
  - Interleaving and hashing
- Snoop filter addition
  - Steering snoops, tracking evictions
- Correctness checking
  - Memory-model checker and soak tests
- Performance tuning
  - Transaction support, cache latencies

What are we modeling?



Source: ARM

A more complete picture

- More control in device and cache interactions
  - Aligned with AMBA terminology and SystemC TLM
  - See *src/mem/packet.{hh, cc}*

- Extended set of supported transactions
  - Whole line writes without need for read exclusive*
  - Reads for non-dirty data and non-cacheable reads*
  - Proper handling of uncacheable transactions
  - See *src/mem/cache/cache.{hh, cc}*

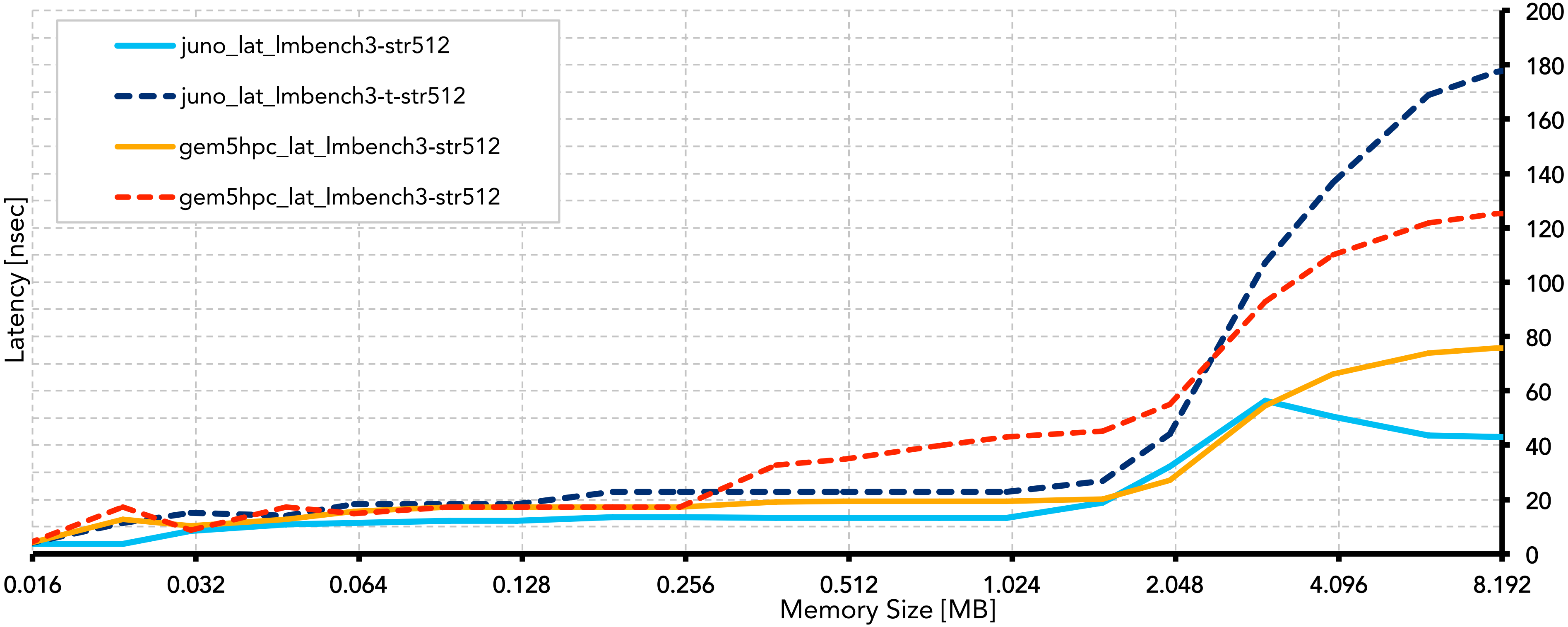
AMBA®

Interconnect Standards from ARM

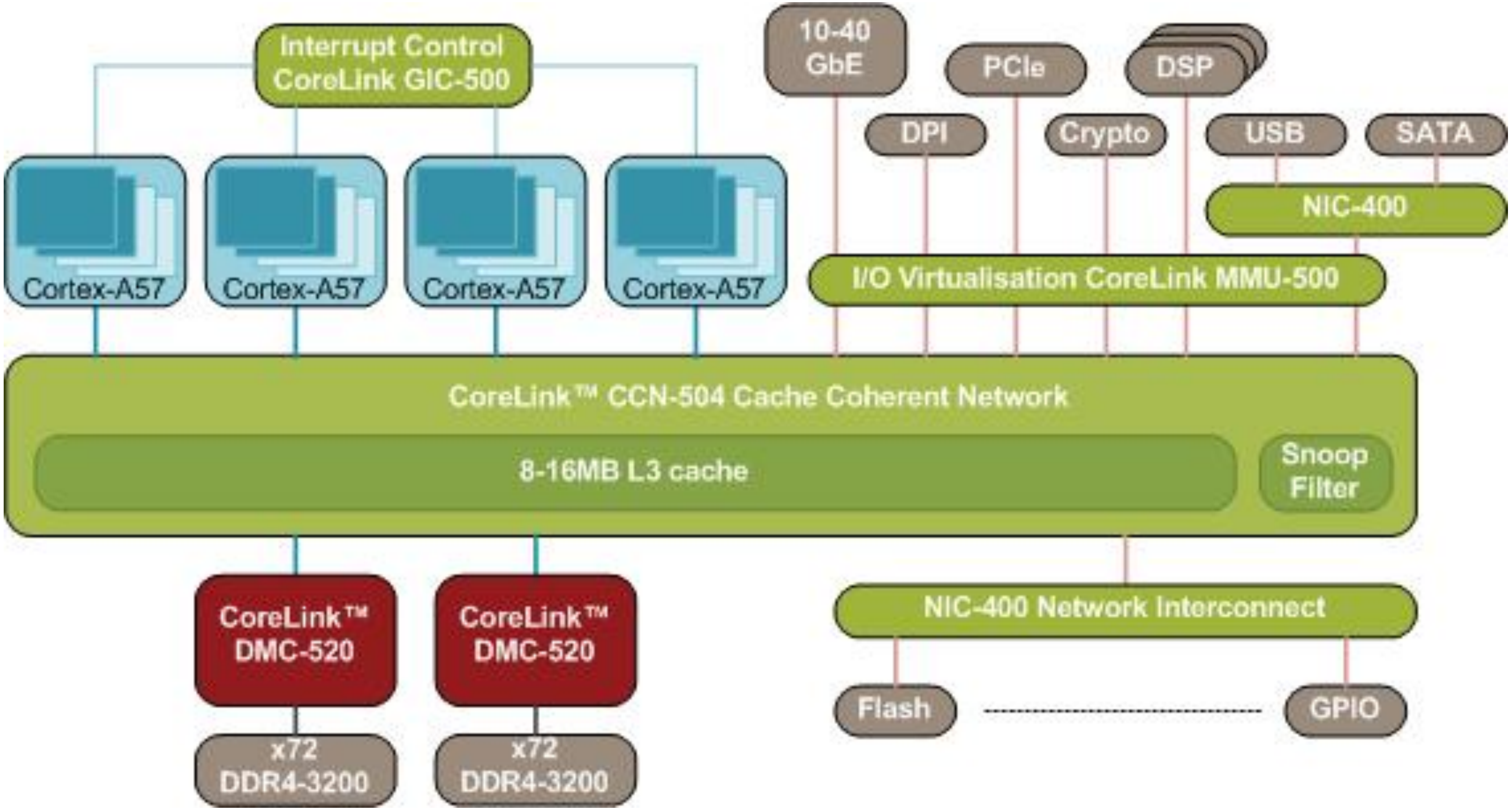
*Transaction support patches are still on reviewboard

Performance tuning

- Cache and crossbar latencies refined
  - Enable more representative behaviour with split into request/response/snoop flows
  - Allow caches with longer and asymmetric read/write latencies
  - See *src/mem/cache/cache.{hh, cc}* and *src/mem/xbar.{hh, cc}*



Where to next?



Source: ARM

ARM

What about Ruby?

- Slow
  - No support for atomic, and a clear bottleneck in timing mode
- Unnecessarily complex
  - Many times there is no need to explore coherency protocols
- Meta programming
  - C++ as text, making development inconvenient
- Compatibility issues
  - Need more flexibility in terms of address ranges, I/O devices, etc

Questions?