Template-based Memory Access Engine for Accelerators in SoCs

Bin Li, Zhen Fang, and Ravi Iyer

Intel Corporation

ASP-DAC 2011

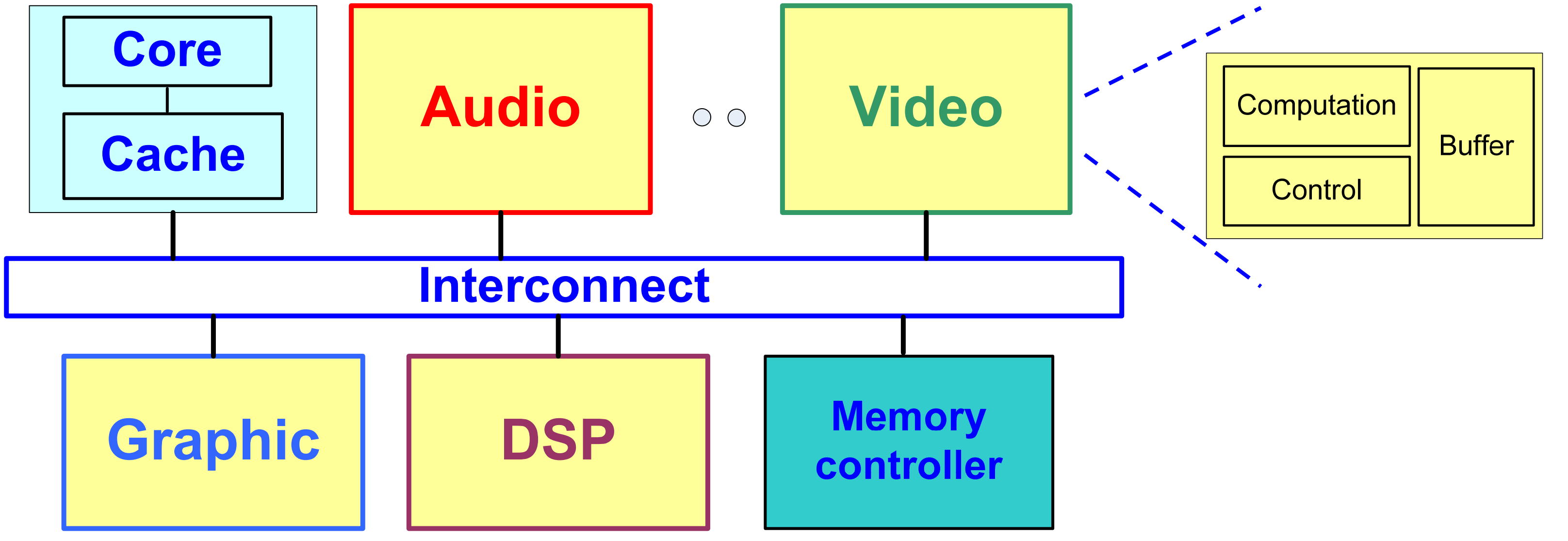
Outline

- Motivation
- Template-based Memory Access Engine
- Evaluations
- Conclusions

Outline

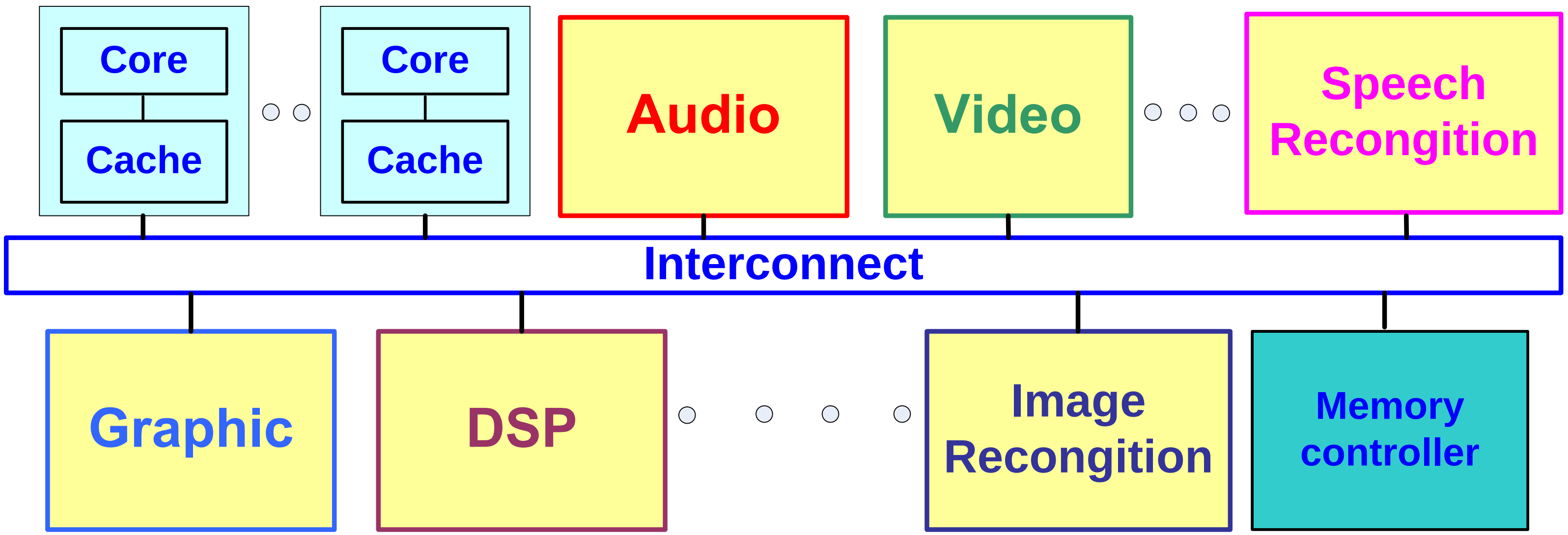
- Motivation
- Template-based Memory Access Engine
- Evaluations
- Conclusions

Systems-on-Chip Architectures



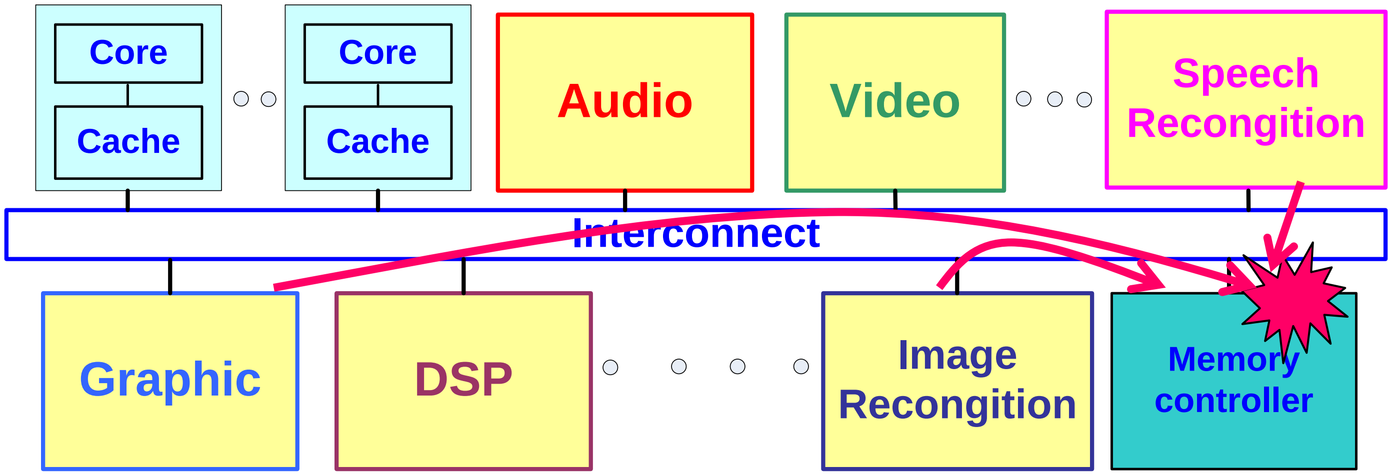
- Today's SoC Architecture
  - One or two cores
    - Employs caches to capture working set
  - Several accelerators
    - Each accelerator has its local buffer

Future SoC Architectures



- Increasing #. power-efficient small cores
- Large number of hardware accelerators
- Several applications run simultaneously

Contention for Memory Access



- Applications running simultaneous
  - Accelerator’s traffic arrives at memory in burst/stream
  - Memory reacts on demand
  - Increased memory access latency and jitter
  - Memory access latency unpredictable

Hide memory access latency is the key to performance improvement

Current Approaches

- Software based prefetch
  - Programmer insert prefetch instructions or compiler detection
  - Cannot issue prefetch instructions sufficiently far in advance
- Hardware based prefetch
  - Hardware detects access pattern and perform prefetch
  - No direct knowledge of future memory references
- Direct Memory Access (DMA)
  - Coordinate data movement between memory and accelerator’s local buffer
  - No global view of the contention

Accelerator Memory Access Characteristics

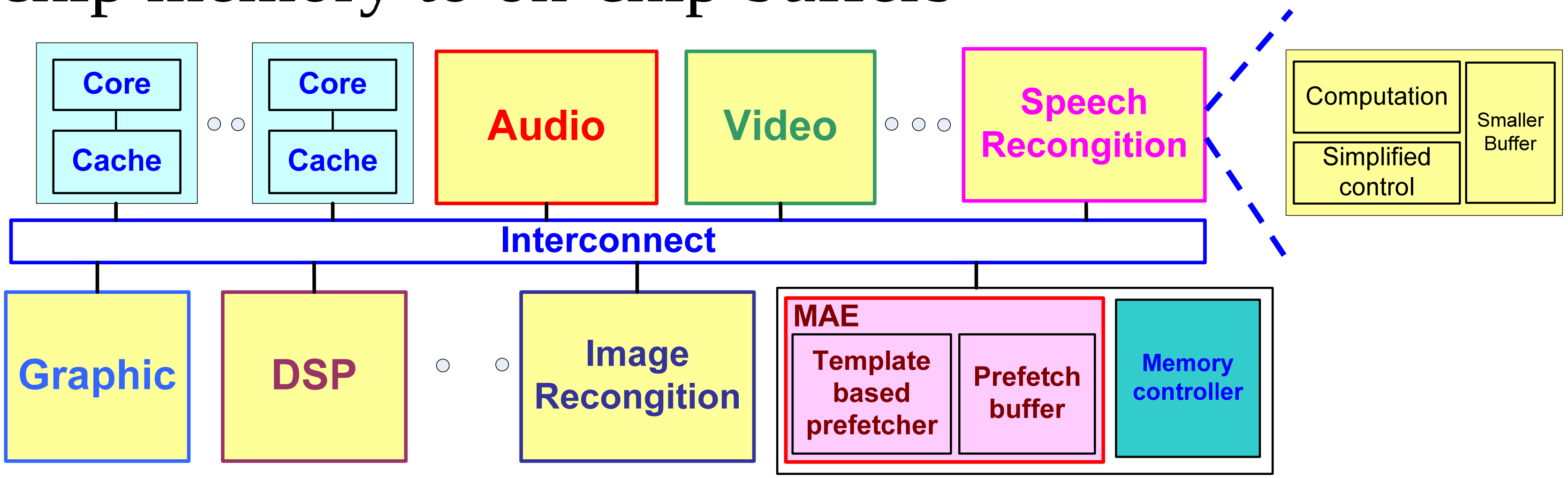
- Accelerators work on one or two tasks
- Memory access pattern known at design time
- Many accelerators have common memory access patterns
- Propose a template-based memory access engine (MAE) for accelerators

Outline

- Motivation
- Template-based Memory Access Engine
- Evaluations
- Conclusions

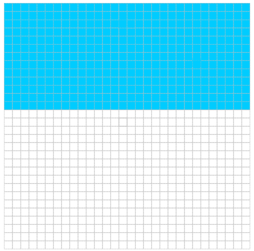
Template-based Memory Access Engine

- Target: Accelerators
- MAE locates next to memory controller
- Provides common memory access templates for accelerators
- Accelerator utilizes MAE to prefetch data from off-chip memory to on-chip buffers

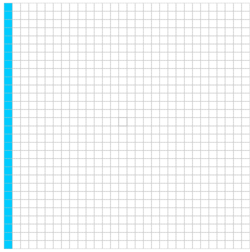


Common Memory Access Patterns

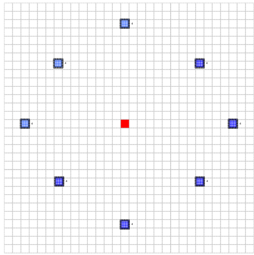
- Streaming
  - e.g., video post processors, display controllers



- Strided
  - e.g., matrix multiplication

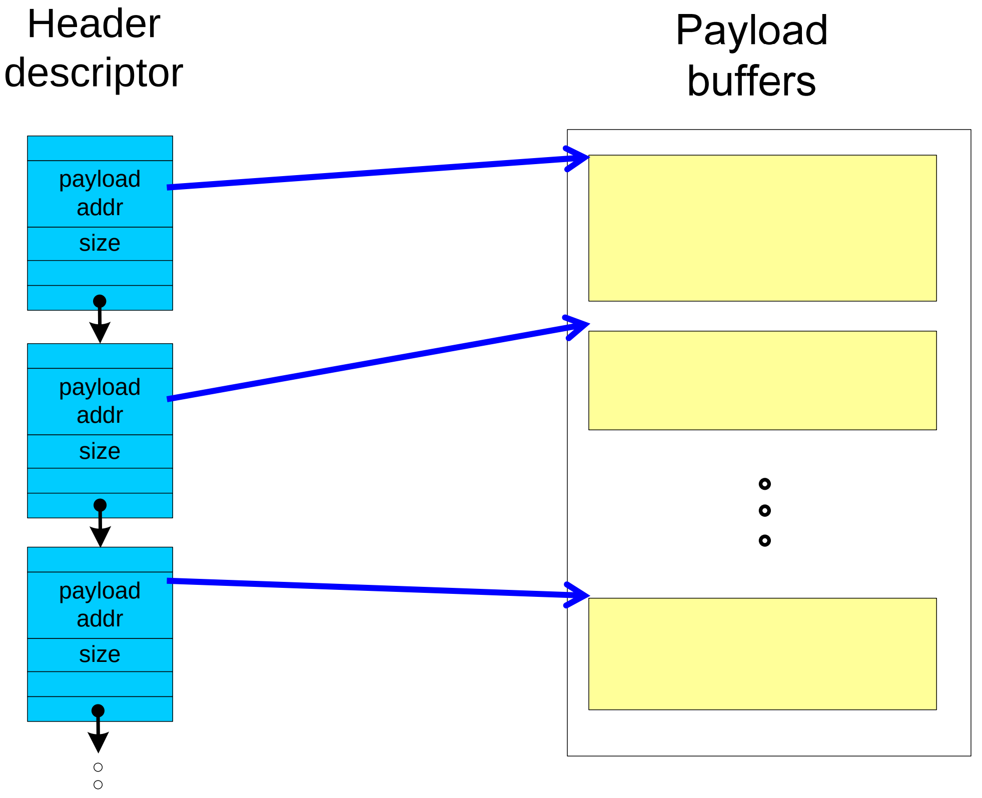


- Complex
  - Regular but complex
  - e.g., image recognition



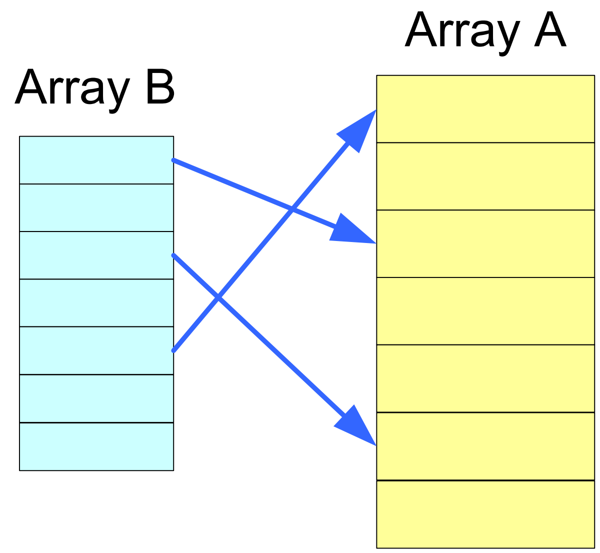
Common Memory Access Patterns (continued)

- Linked-list
  - e.g., network interface controller (NIC)

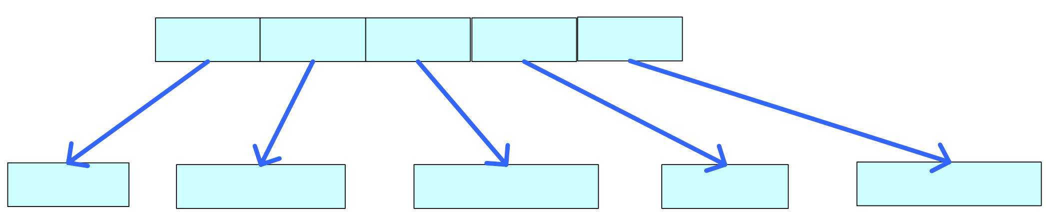


Common Memory Access Patterns (continued)

- Indirect array access
  - e.g., speech recognition, image processing



- Gather
  - Reads from non-contiguous locations
  - e.g., vector execution

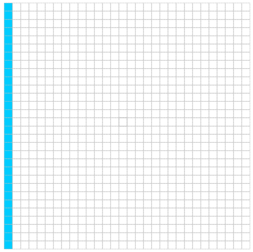


Template Descriptors

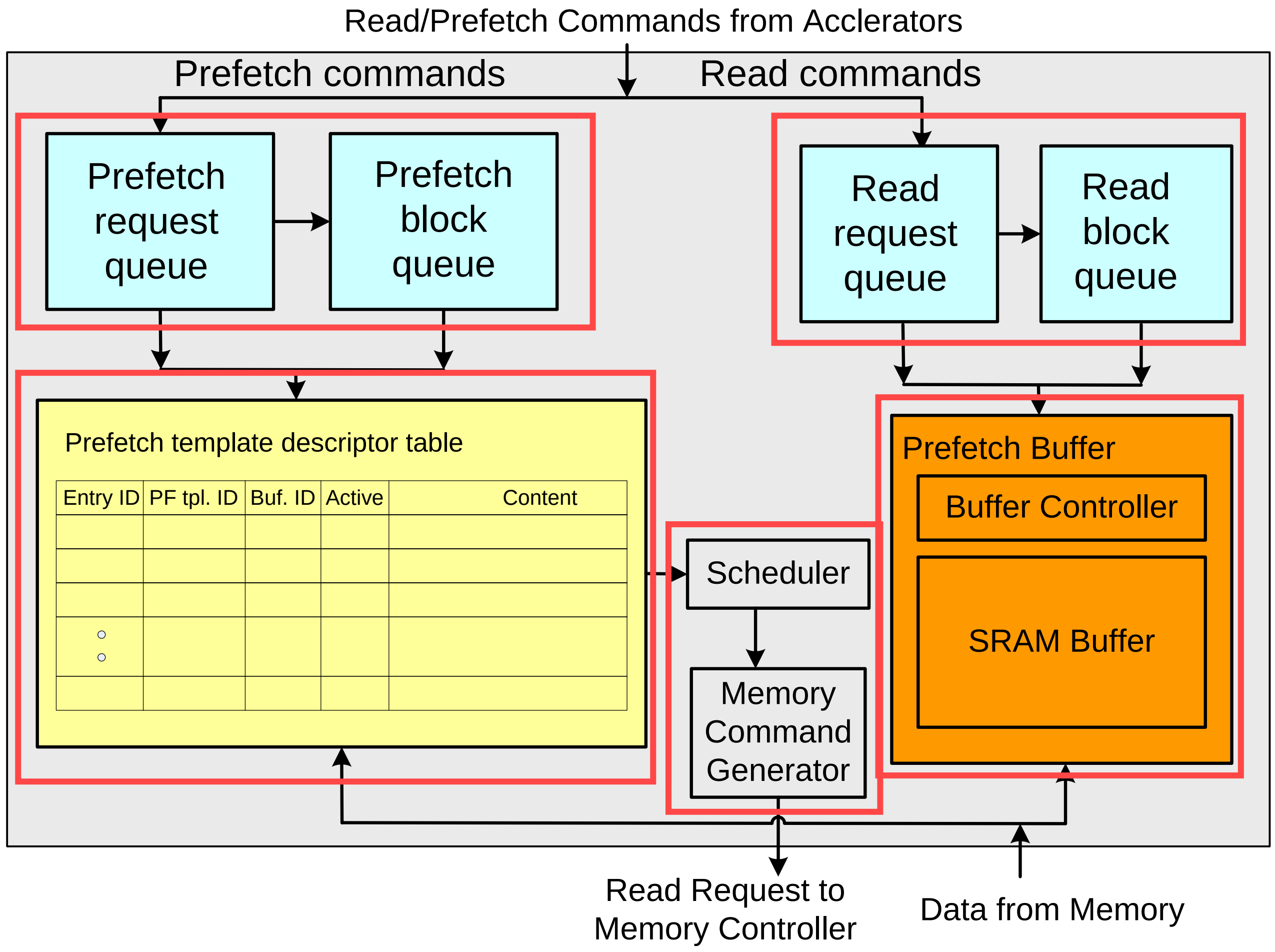
- Template Descriptors in MAE
  - Streaming
  - Strided
  - Complex
  - Linked-list
  - Indirect array
  - Gather

Example:

Prefetch template ID	Format			
ID 2 (Strided)	Start address	Byte count	Stride	Repeating times



Memory Access Engine Architecture

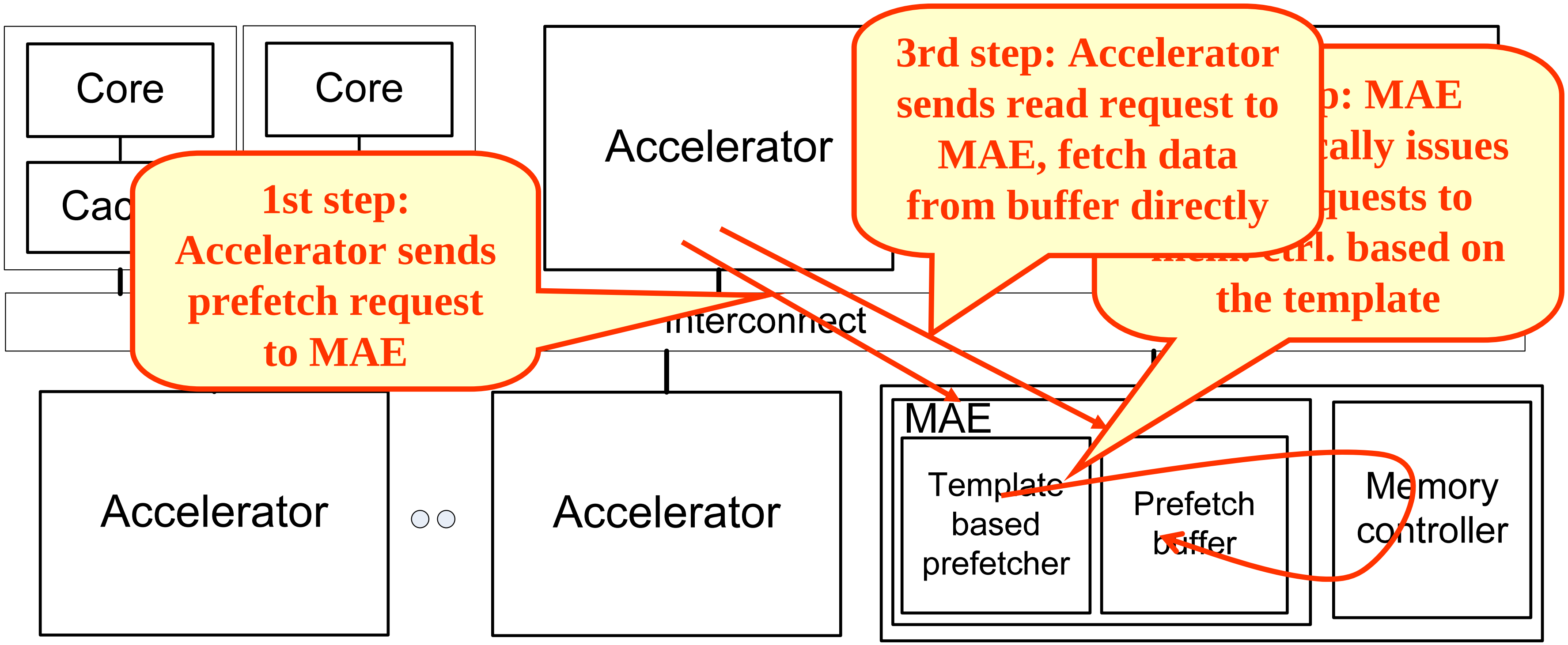


Memory Access Engine Usage

- Memory access classification for accelerators:
  - Data read with a known pattern
  - Data read w/o a pattern
  - Write access
- MAE services data read access with known patterns only
- For data read w/o a pattern and data write, accelerator sends requests to the memory controller directly

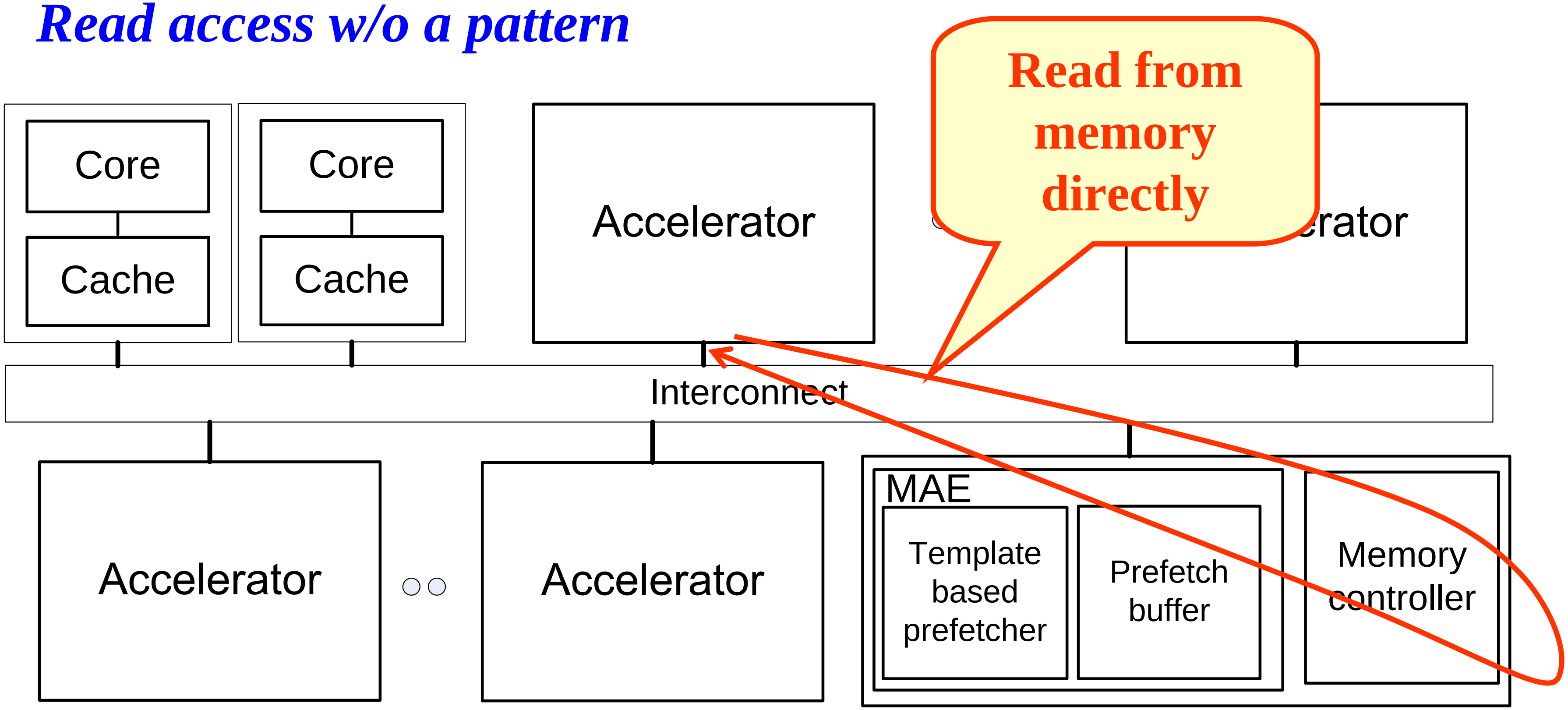
Memory Access Flows

Read access with known patterns



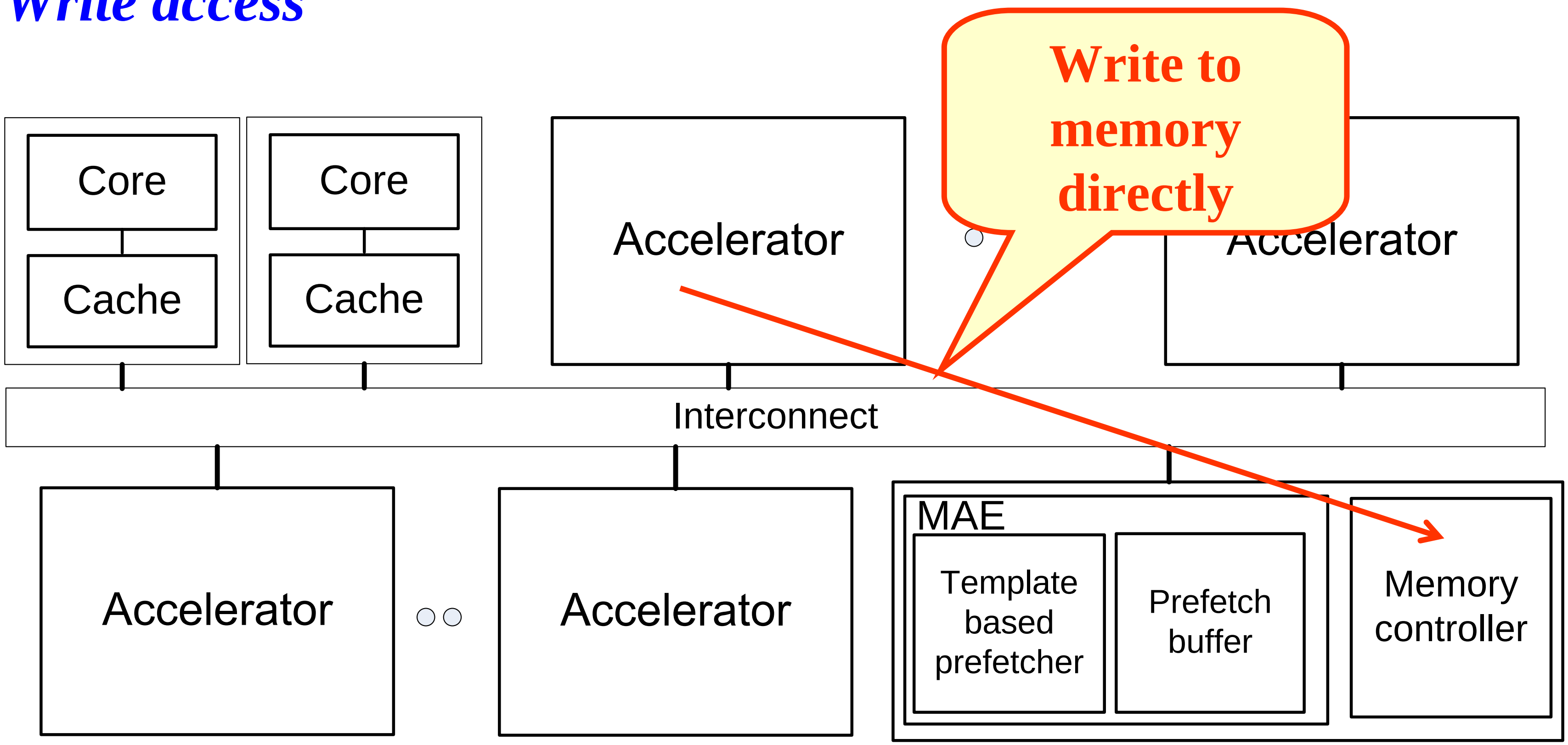
Memory Access Flows

Read access w/o a pattern



Memory Access Flows

Write access

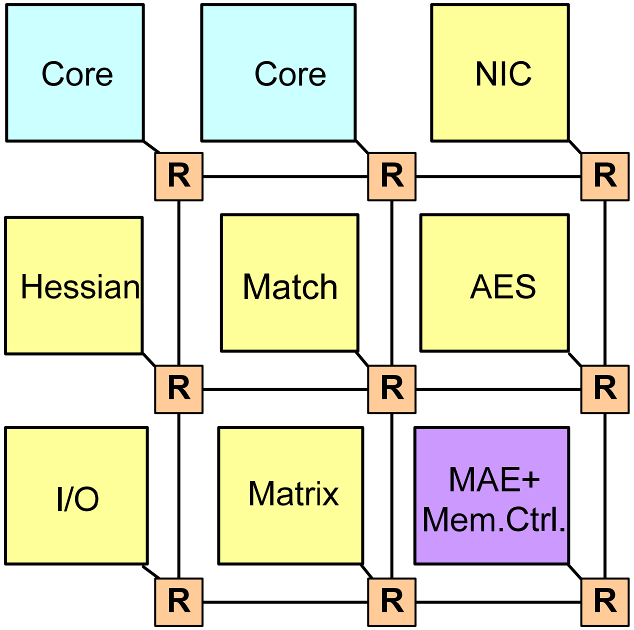


Outline

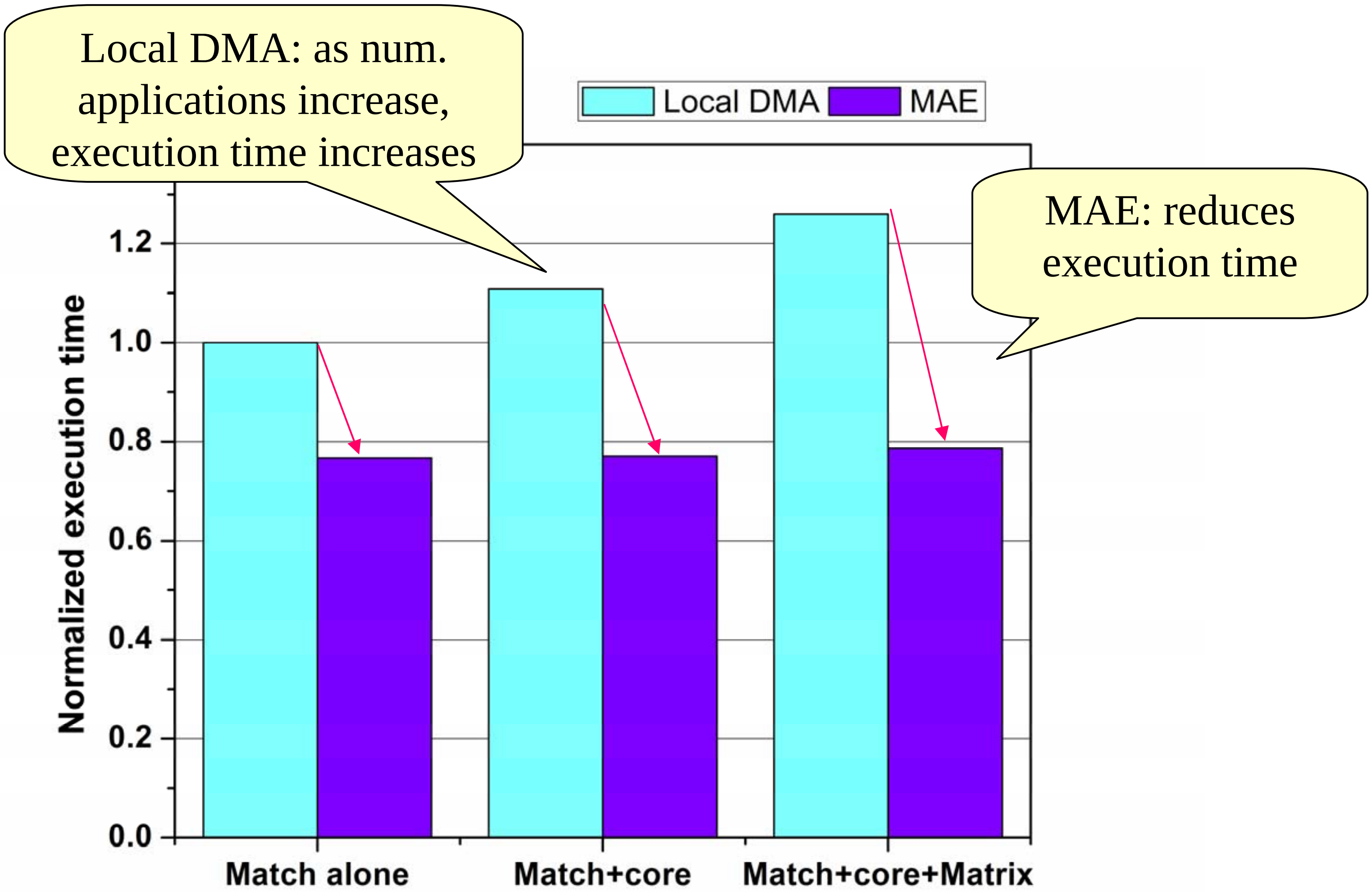
- Motivation
- Template-based Memory Access Engine
- Evaluations
- Conclusions

Experimental Setup

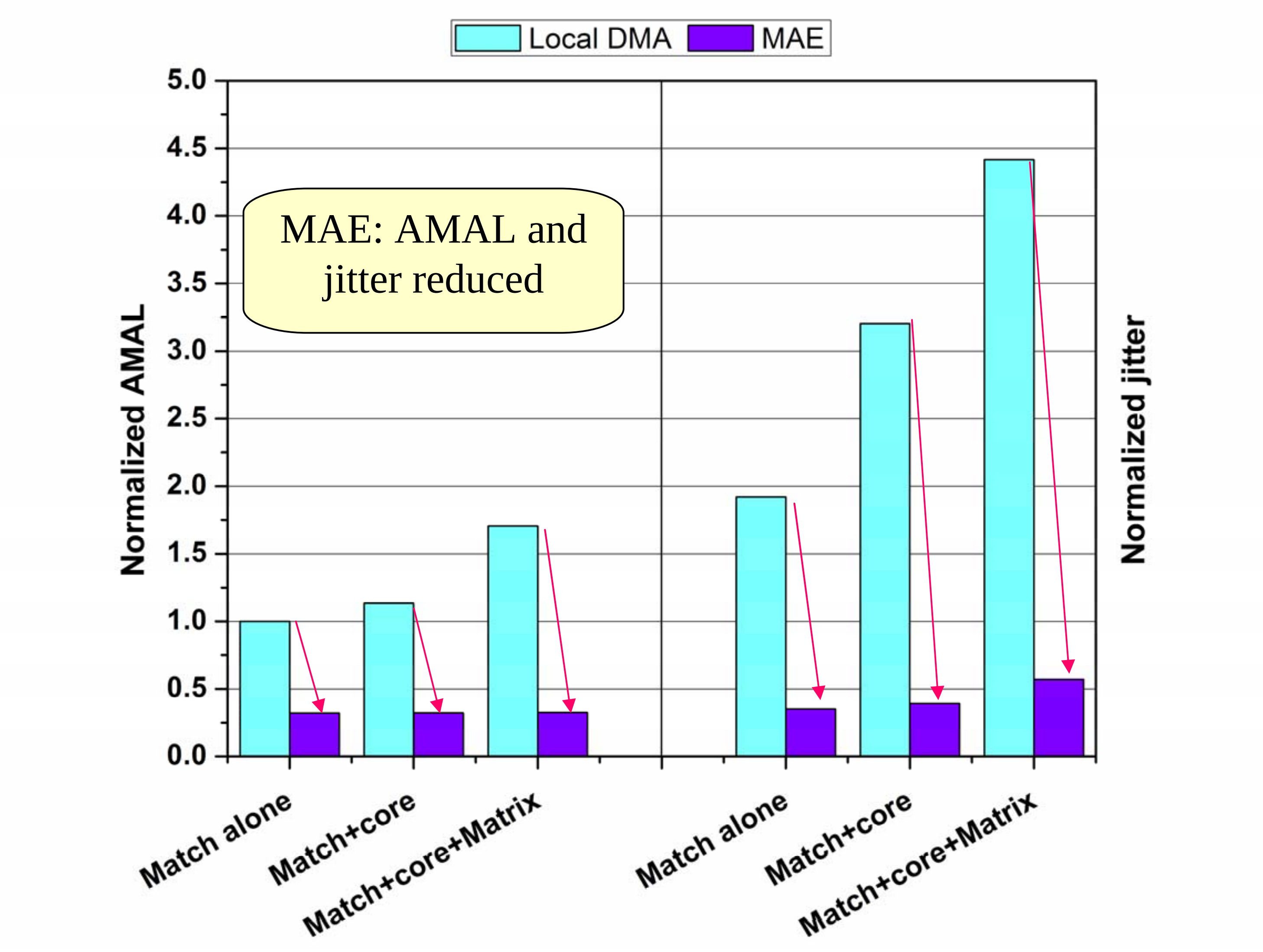
- Configurations:
  - Cores: 1GHz low power
    - Workloads: SPEC CPU 2000 (*art*, *mcf*)
  - NoC: 3 x 3 mesh NoC, X-Y routing
    - Router: 3 VCs, 5 buf/VC, 4 router pipelines
  - Memory:
    - bandwidth: 3.2GB/s maximum bandwidth
    - Access latency: 100 core clocks
  - Accelerators:
    - NIC, Hessian, Match, AES, Matrix
  - MAE:
    - Prefetch buffer: 32KB
    - Queue size: 8



Execution Time Comparison



Memory Access Latency and Jitter Comparison



AMAL: average memory access latency

Outline

- Motivation
- Template-based Memory Access Engine
- Evaluations
- Conclusions

Conclusions

- Accelerators exhibit common memory access patterns
  - Identity simple to complex memory access patterns
- Propose template-based MAE for accelerators
  - Accelerator sends simple prefetch command to MAE
  - MAE automatically fetches data to on-chip buffer
  - MAE has a global view of requests
  - Improves accelerator performance
  - Simplifies accelerator design
  - Effective for future SoC architectures

Thank you!