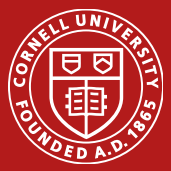


Cornell University

CSL

Efficient Data Supply for Hardware Accelerators with Prefetching and Access/Execute Decoupling

Tao Chen and G. Edward Suh
Computer Systems Laboratory
Cornell University

Accelerator-Rich Computing Systems

- Computing systems are becoming **accelerator-rich**
  - General-purpose cores + a large number of accelerators
- **Challenge**: Design and verification complexity
  - Non-recurring engineering (NRE) cost **per accelerator**
- **Manual efforts** are a major source of cost
  - Create computation pipelines → High-Level Synthesis (HLS)
  - Manage data supply from memory

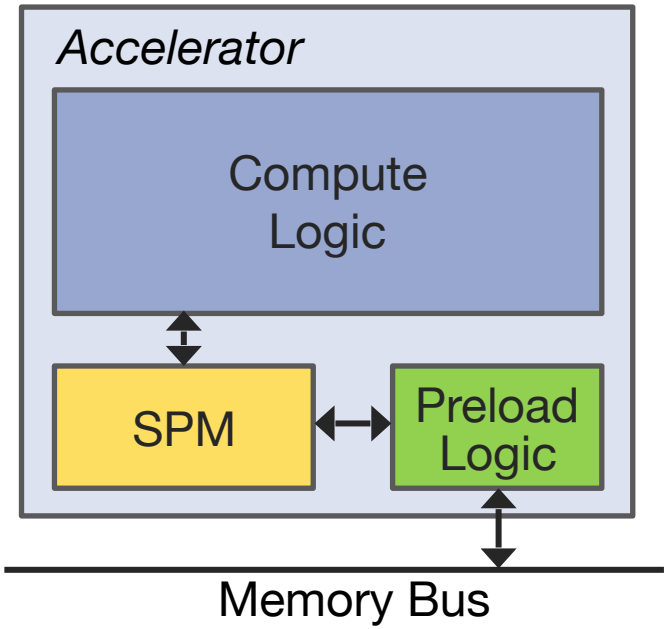
↓

This work: An automated framework for generating accelerators with efficient data supply

Inefficiencies in Accelerator Data Supply

Scratchpad-based accelerators

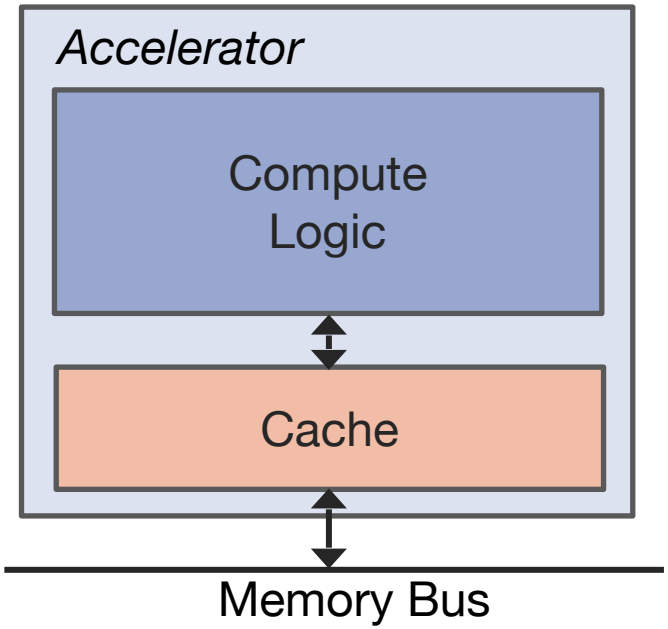
- On-chip scratchpad memory (SPM)
- Manually designed logic to move data between SPM and main memory
- **Pros**: Good performance
- **Cons**: High design effort, accelerator-specific, not reusable



Inefficiencies in Accelerator Data Supply

Scratchpad-based accelerators

- On-chip scratchpad memory (SPM)
- Manually designed logic to move data between SPM and main memory
- **Pros**: Good performance
- **Cons**: High design effort, accelerator-specific, not reusable

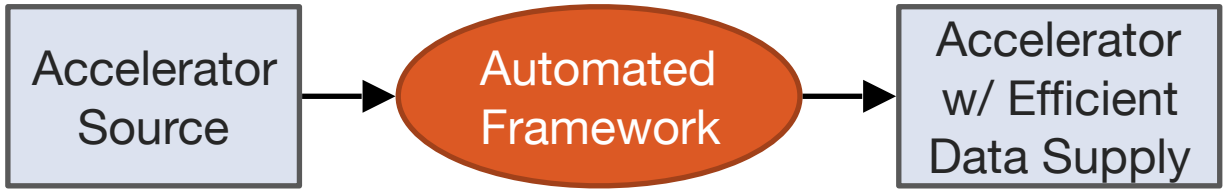


Cache-based accelerators

- **Pros**: Low design effort, cache can be reused
- **Cons**: Uncertain memory latency impacts performance

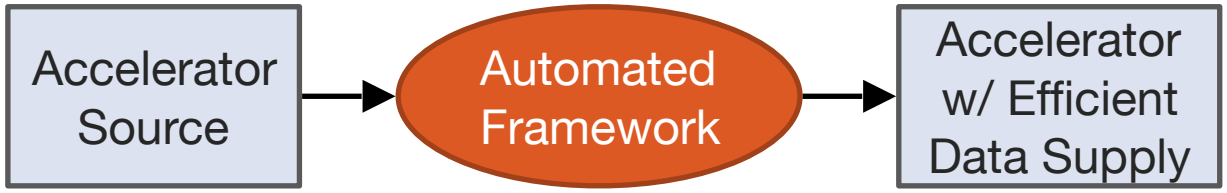
Optimize Data Supply for Cache-Based Accelerators

Approach: **automated framework** for generating accelerators with efficient data supply



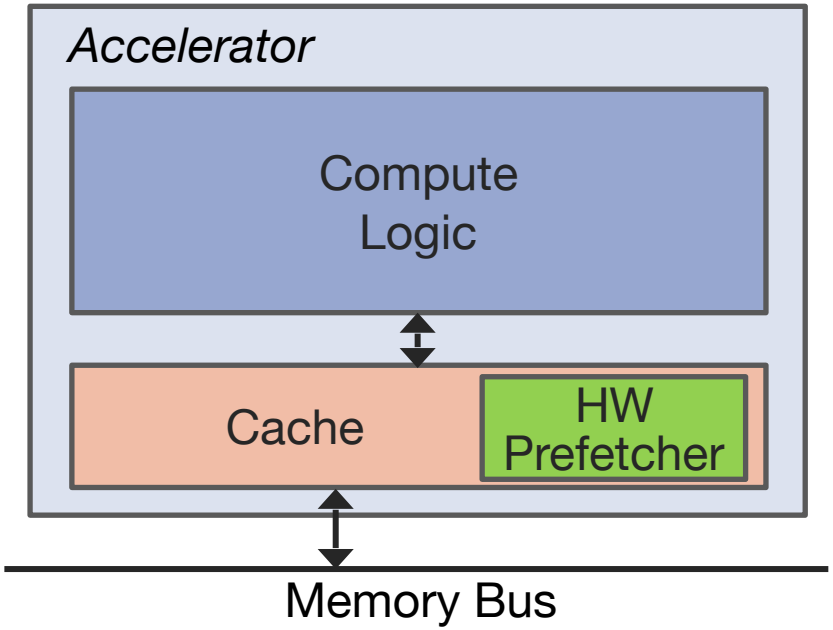
Optimize Data Supply for Cache-Based Accelerators

Approach: **automated framework** for generating accelerators with efficient data supply



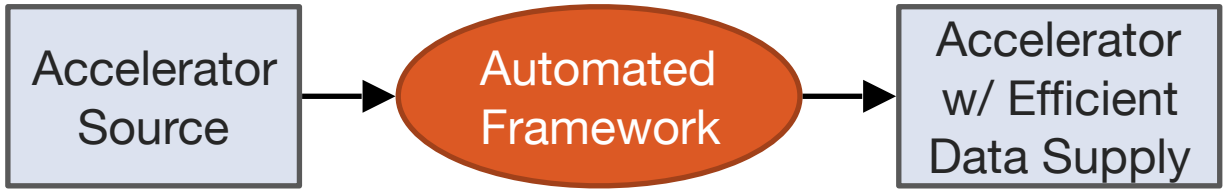
Techniques

- **Prefetching**
  - Tagging memory accesses



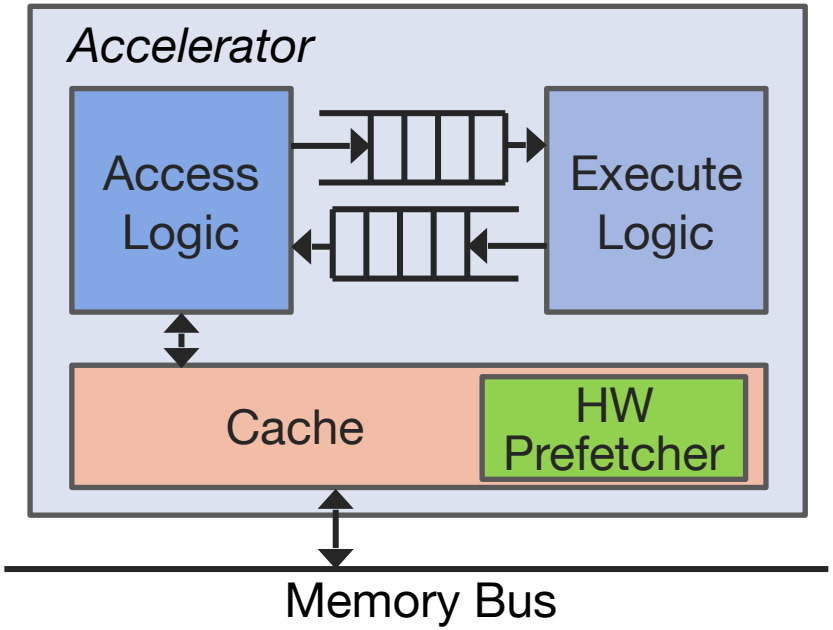
Optimize Data Supply for Cache-Based Accelerators

Approach: **automated framework** for generating accelerators with efficient data supply



Techniques

- **Prefetching**
  - Tagging memory accesses
- **Access/Execute Decoupling**
  - Program slicing + architecture template

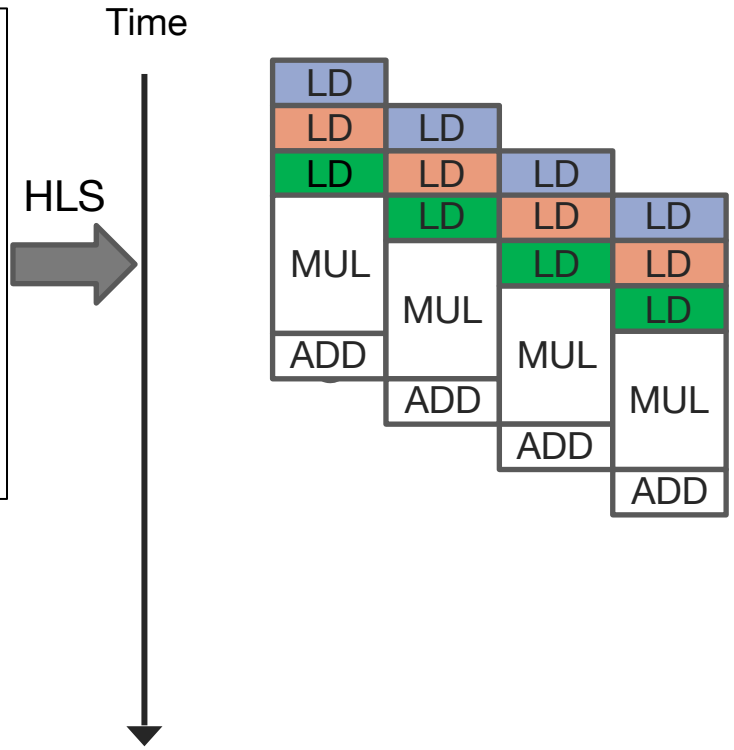


Impact of Uncertain Memory Latency

- **Example**: Sparse Matrix Vector Multiplication (spmv)
  - Pipeline generated with High-Level Synthesis (HLS)

```
// inner loop of sparse matrix
// vector multiplication

for (j = begin; j < end; j++) {
    #pragma HLS pipeline
    Si = val[j] * vec[cols[j]];
    sum = sum + Si;
}
```



Impact of Uncertain Memory Latency

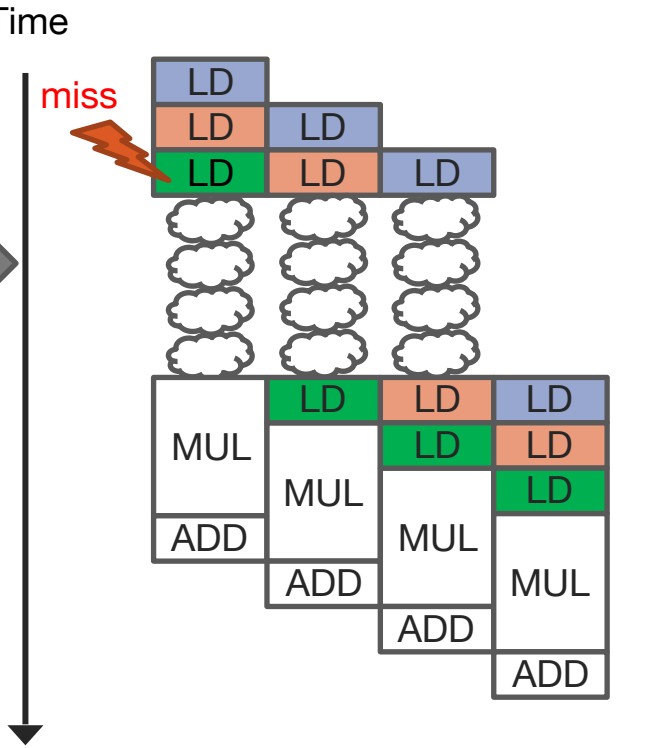
- **Example**: Sparse Matrix Vector Multiplication (spmv)
  - Pipeline generated with High-Level Synthesis (HLS)

```
// inner loop of sparse matrix
// vector multiplication

for (j = begin; j < end; j++) {
    #pragma HLS pipeline
    Si = val[j] * vec[cols[j]];
    sum = sum + Si;
}
```

Regular stride | Irregular | Regular stride

HLS



- Reduce cache misses for **regular** accesses
  - Prefetch data into the cache
- Tolerate cache misses for **irregular** accesses
  - Access/Execute Decoupling

A cache miss stalls the entire accelerator pipeline

Hardware Prefetching

- Predict future memory accesses
- **PC** is often used as a hint
  - Stream localization
  - Spatial correlation prediction

```
for (j = begin; j < end; j++) {
    Si = val[j] * vec[cols[j]];
    sum = sum + Si;
}
```

Global Addr Stream

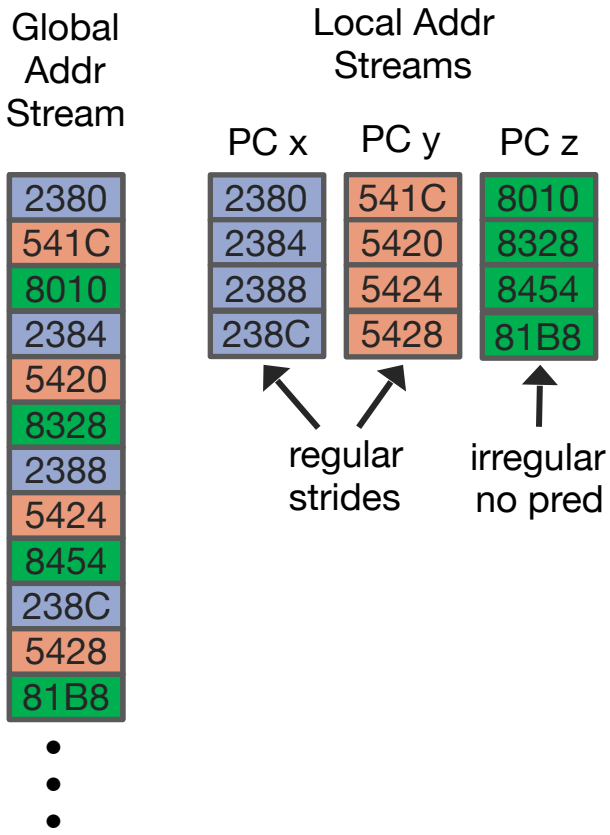
2380
541C
8010
2384
5420
8328
2388
5424
8454
238C
5428
81B8

Hardware Prefetching

- Predict future memory accesses
- **PC** is often used as a hint
  - Stream localization
  - Spatial correlation prediction

```
for (j = begin; j < end; j++) {
    Si = val[j] * vec[cols[j]];
    sum = sum + Si;
}
```

- **Problem**: accelerators lack a PC
- **Solution**: generate PC-like tags for accelerator memory accesses

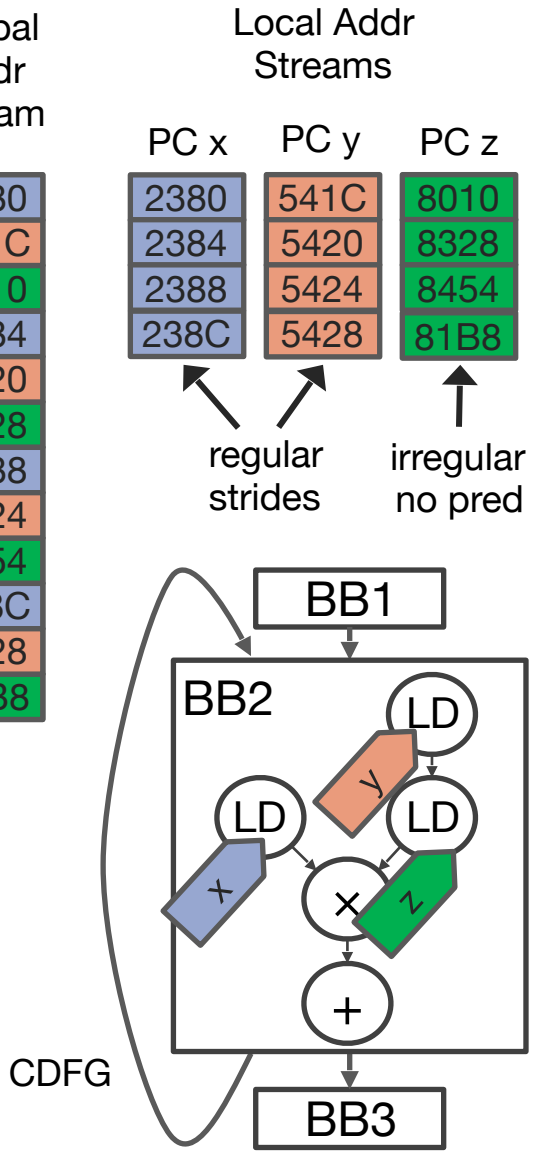


Hardware Prefetching

- Predict future memory accesses
- **PC** is often used as a hint
  - Stream localization
  - Spatial correlation prediction

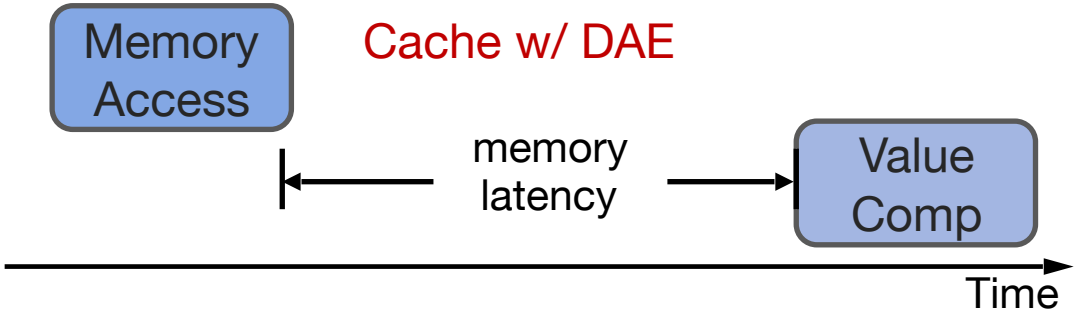
```
for (j = begin; j < end; j++) {
    Si = val[j] * vec[cols[j]];
    sum = sum + Si;
}
```

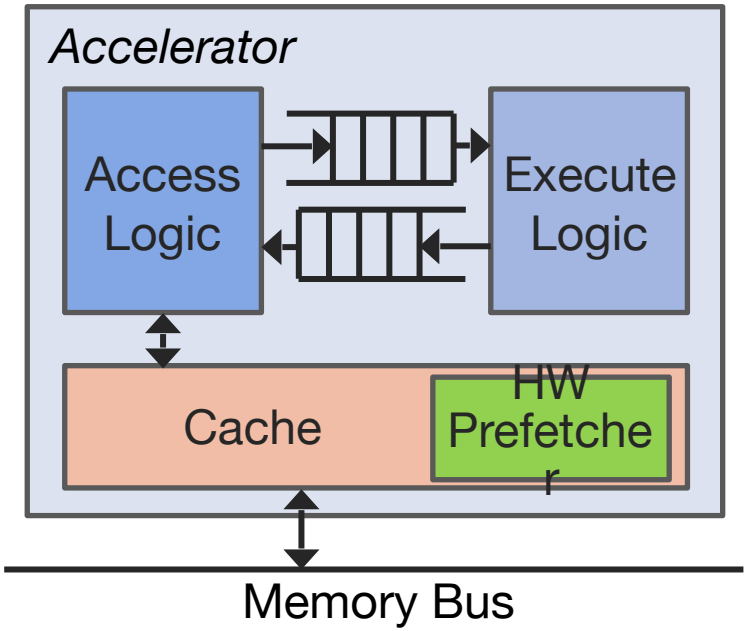
- **Problem**: accelerators lack a PC
- **Solution**: generate PC-like tags for accelerator memory accesses



Decoupled Access/Execute (DAE)

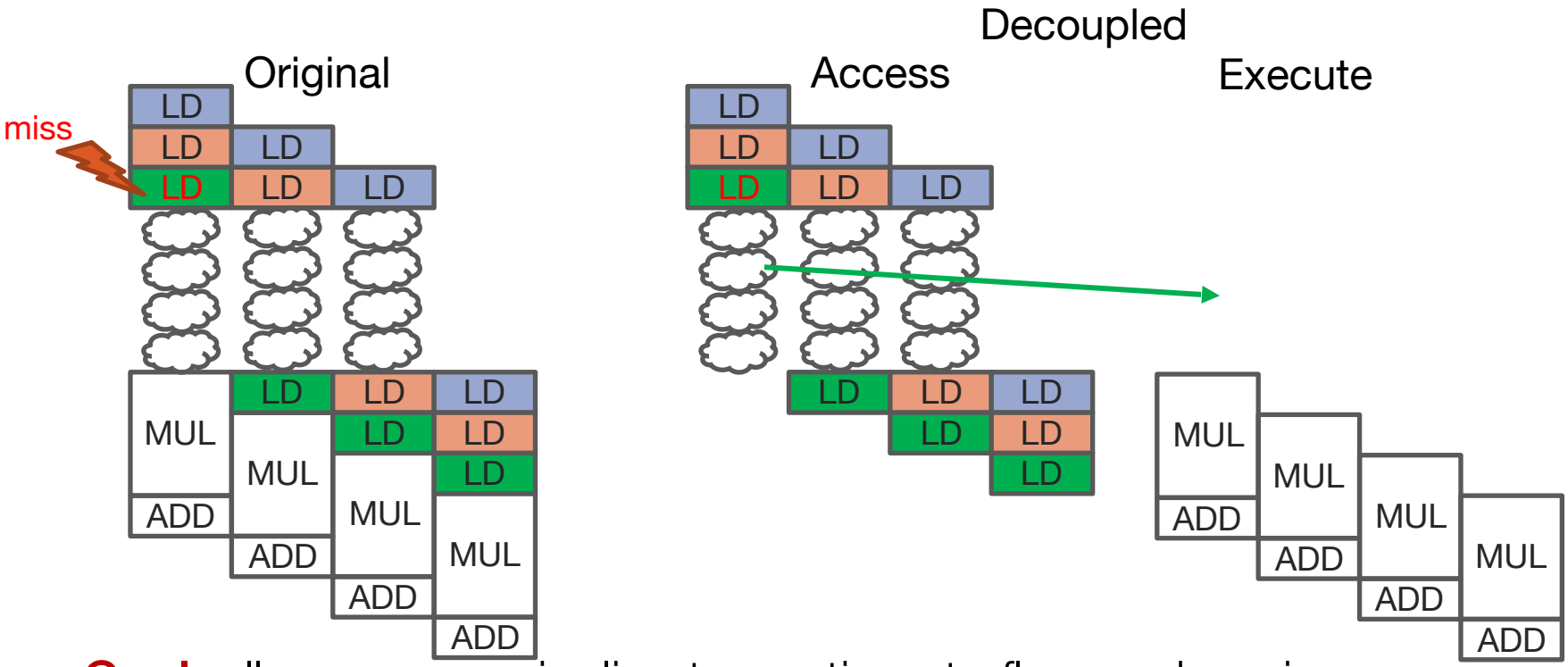
- Limitations of Hardware Prefetching
  - Not accurate for **complex patterns** / Needs **warm-up time**
  - Fundamental reason: lack of **semantic information**
- Decoupled Access/Execute
  - Allow memory accesses to **run ahead** to preload data





Traditional DAE is not Effective for Accelerators

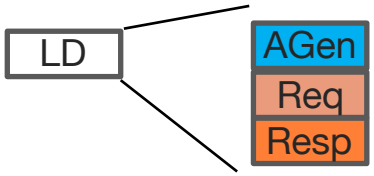
- **Traditional DAE**: access part forwards data to execute part
- **Problem**: access pipeline stalls on misses
  - Throughput is limited by access pipeline

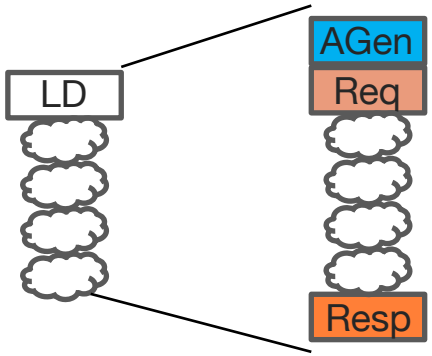


- **Goal**: allow access pipeline to continue to flow under misses

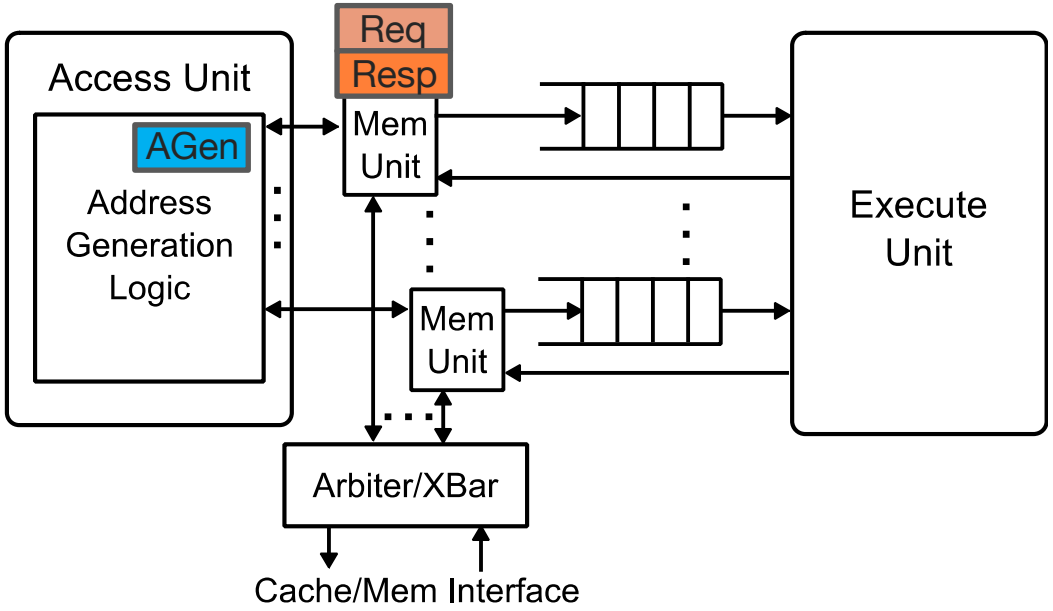
DAE Accelerator with Decoupled Loads

- Anatomy of a load



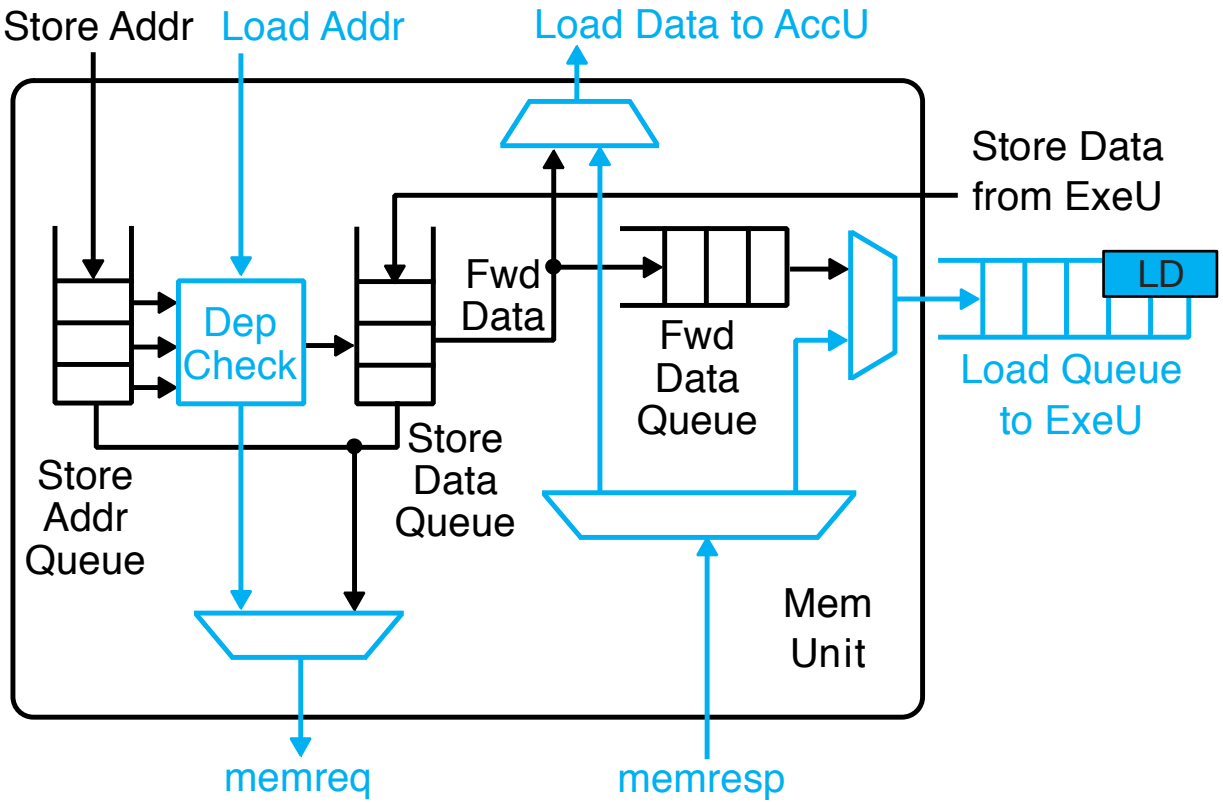


- **Solution**: Delegate request/response handling



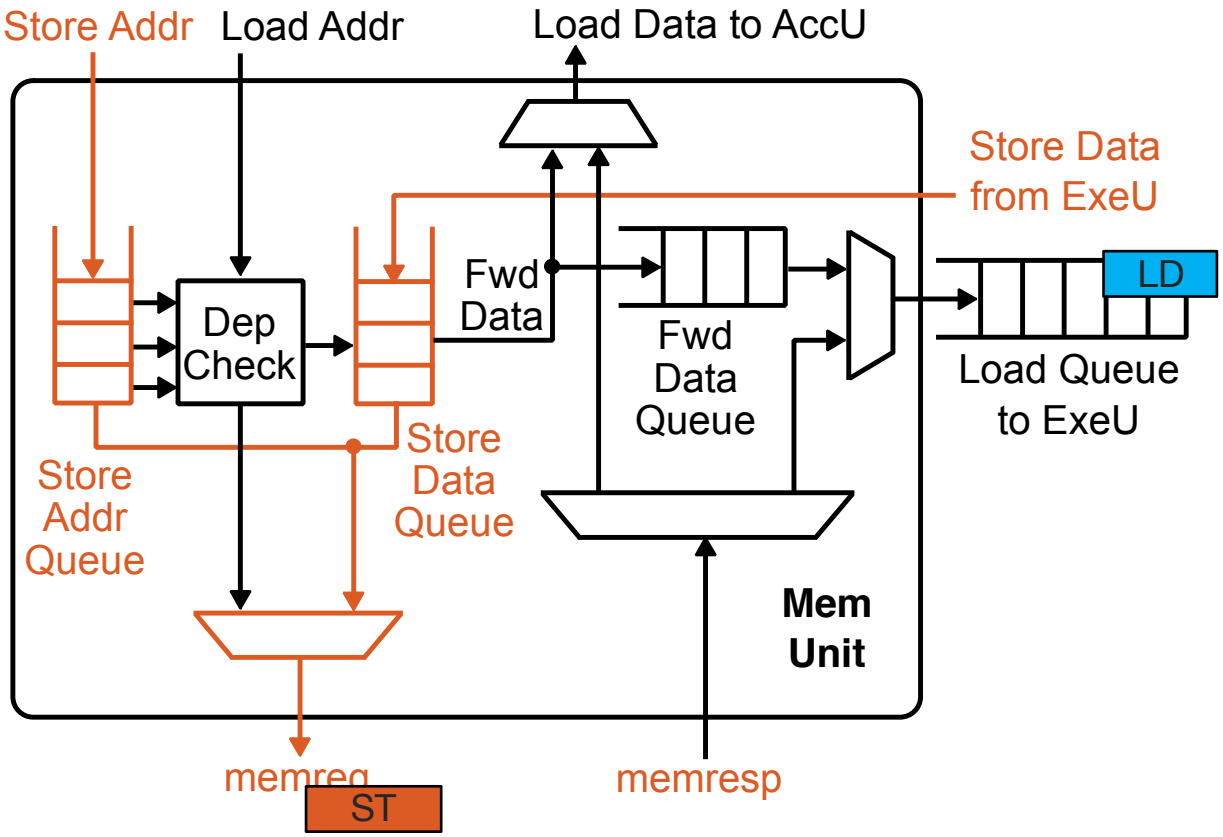
Memory Unit

- Proxy for handling memory requests and responses
  - Supports response reordering and store-to-load forwarding



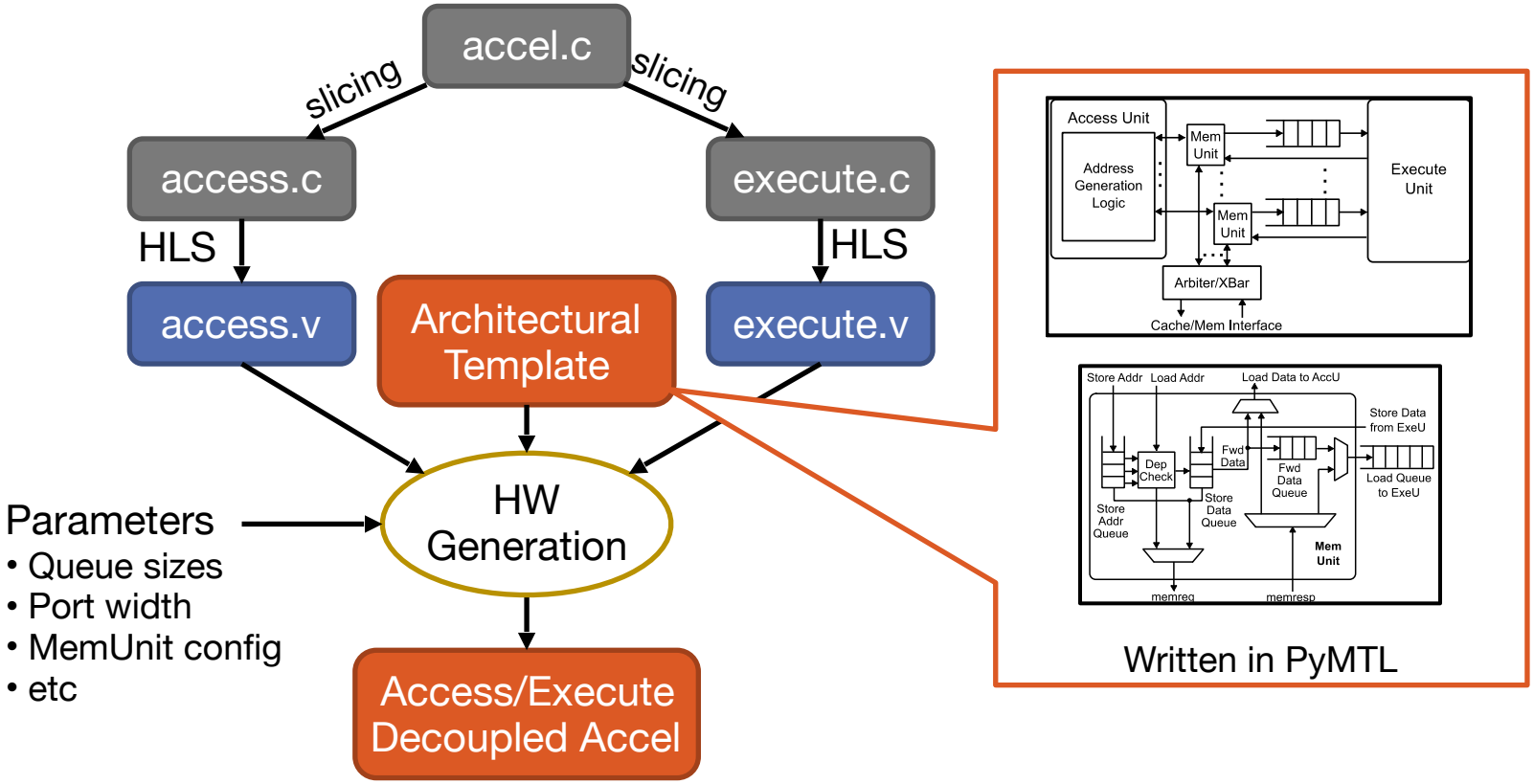
Memory Unit

- Proxy for handling memory requests and responses
  - Supports response reordering and store-to-load forwarding



Automated DAE Accelerator Generation

- Program slicing for generating access/execute slices
- Architectural template with configurable parameters

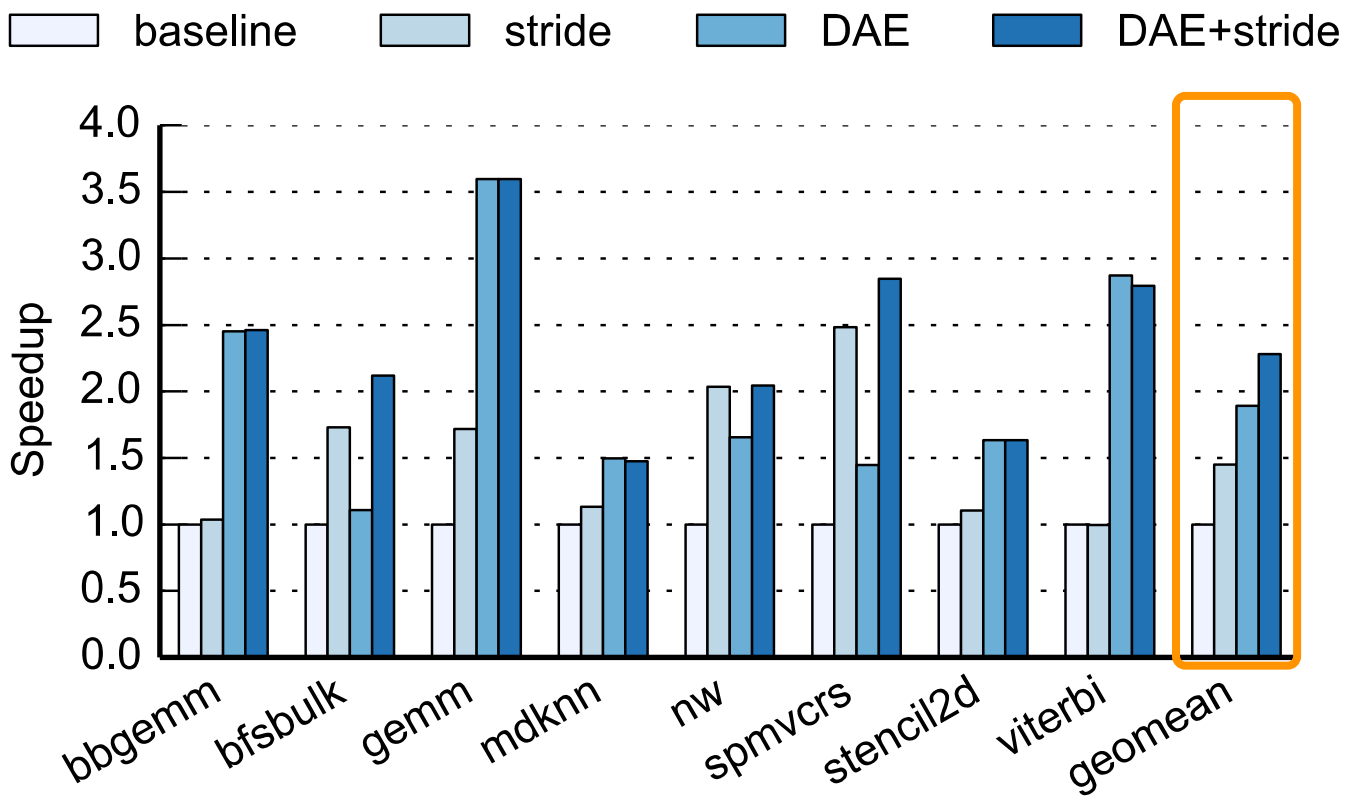


Evaluation Methodology

- Vertically integrated modeling methodology
  - **System components**: cycle-level (gem5)
  - **Accelerators**: register-transfer-level (Vivado HLS, PyMTL)
  - **Area, power and energy**: gate-level (commercial ASIC flow)
- Benchmark accelerators from MachSuite

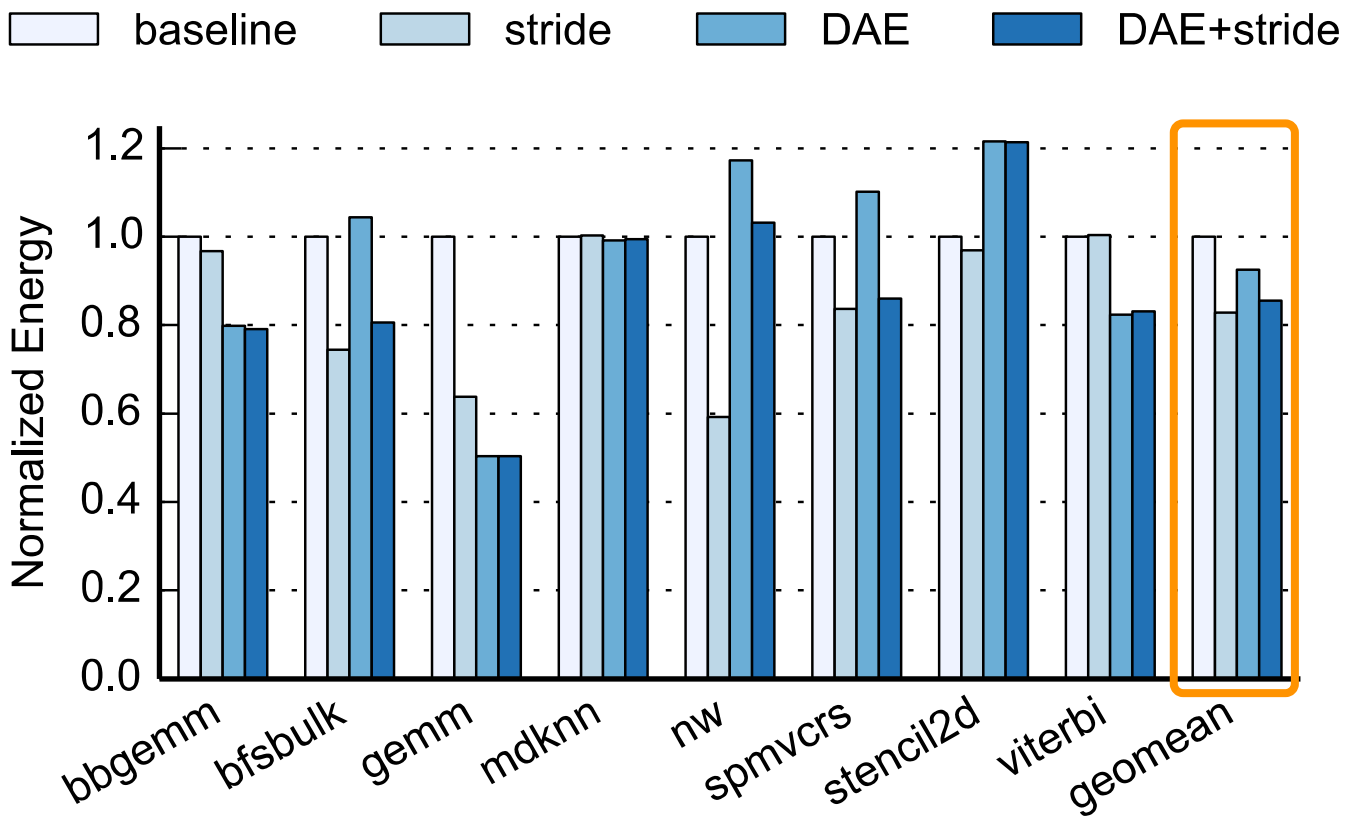
Name	Description
bbgemm	Blocked matrix multiplication
bfsbulk	Breadth-First Search
gemm	Dense matrix multiplication
mdknn	Molecular dynamics (K-Nearest Neighbor)
nw	Needleman-Wunsch algorithm
spmvcrs	Sparse matrix vector multiplication
stencil2d	2D stencil computation
viterbi	Viterbi algorithm

Performance Comparison



- **2.28x** speedup on average
- Prefetching and DAE work in synergy

Energy Comparison



- **15%** energy reduction on average because of reduced stalls
- MemUnits/queues only consume a small amount of energy

More Details in the Paper

- Deadlock Avoidance
- Customization of Memory Units
- Baseline Validation
- Power and Area Comparison
- Energy, Power and Area Breakdown
- Sensitivity Study on Varying Queue Sizes
- Design Space Exploration: Queue Size Customization

Summary

Cache-based accelerators

- Avoid the high design cost of manual data movement logic
- **Problem**: Inefficient in handling uncertain memory latency

Approach: Automated program analysis and architectural template to generate accelerators with efficient data supply

- **Tagging memory requests** to enable prefetching
- **Decoupling** to enable memory accesses to run ahead

Results: High-performance cache-based accelerators with minimal manual efforts