Microscope on Memory:

MPSoC-enabled Computer Memory System Assessments

FCCM 2018

Abhishek Kumar Jain, Scott Lloyd, Maya Gokhale
Center for Applied Scientific Computing

May 1, 2018

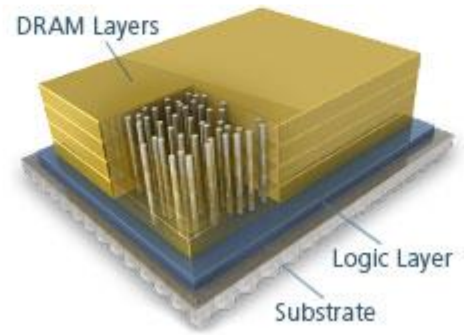
LLNL-PRES-750335
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Introduction

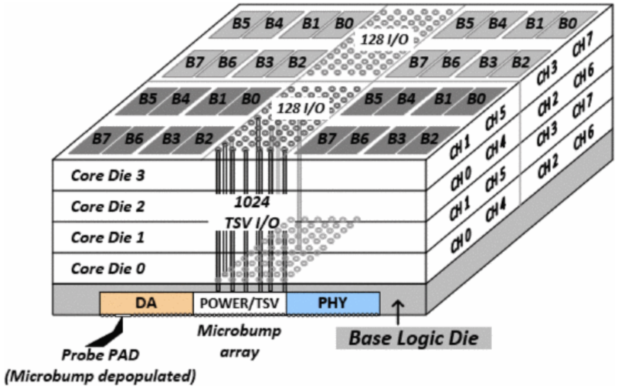
- Recent advances in memory technology and packaging
  - High bandwidth memories – HBM, HMC
  - Non-volatile memory – 3D XPoint
  - Potential for logic and compute functions co-located with the memory
  - Brought attention to computer memory system design and evaluation

HMC



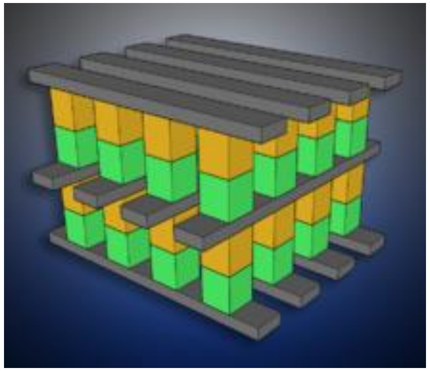
Micron Technology

HBM



Hongshin Jun, et. al. IMW 2017

3D XPoint

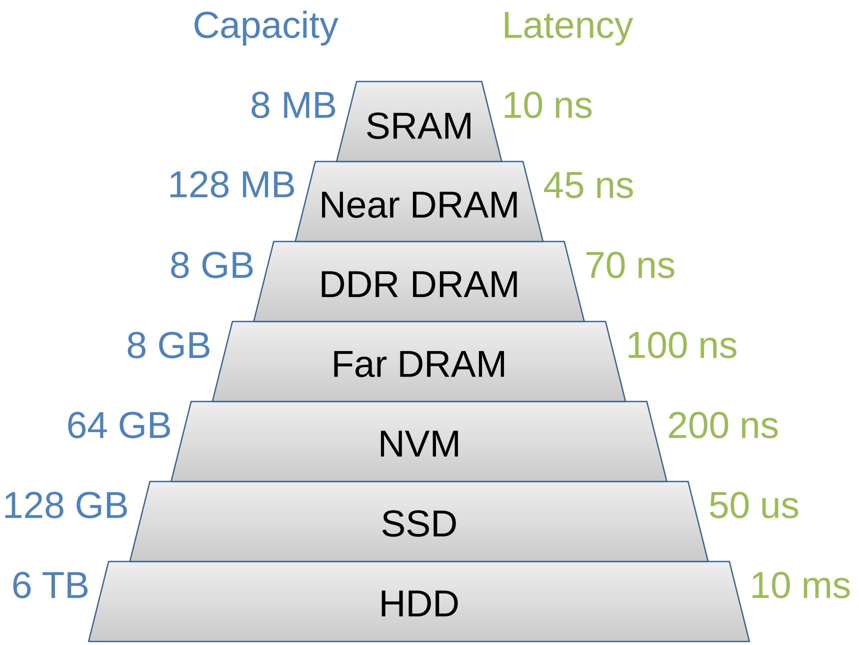


Creative Commons Attribution

Introduction

- Emerging memories exhibit a wide range of bandwidths, latencies, and capacities
  - Challenge for the computer architects to navigate the design space
  - Challenge for application developers to assess performance implications

Memory/Storage Hierarchy



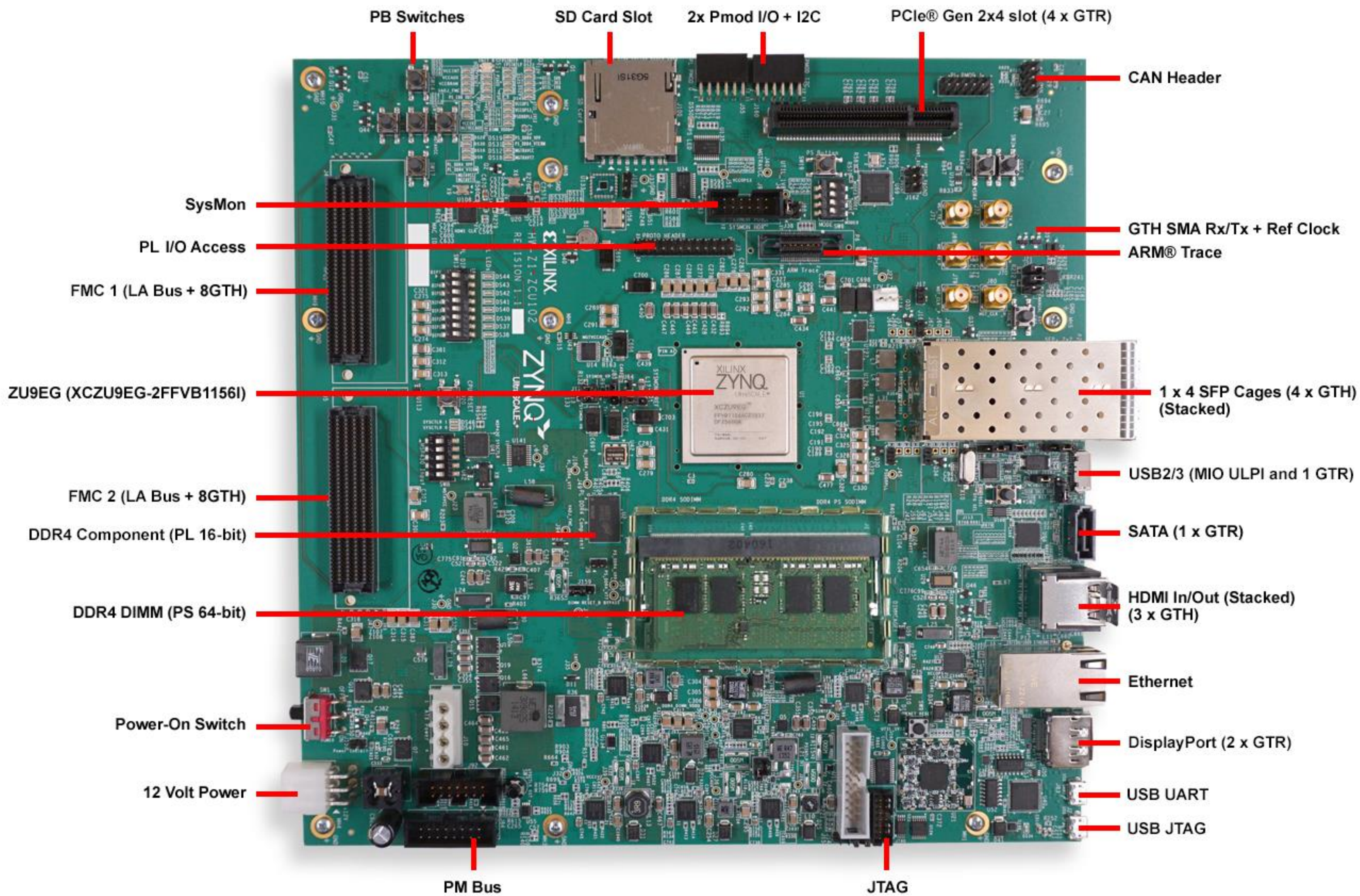
Introduction

- Need for ***system level*** exploration of the design space
  - Combinations of memory technology
  - Various memory hierarchies
  - Potential benefit of near-memory accelerators
  - Prototype architectural ideas in detail
- Need to quantitatively evaluate the performance impact on applications – beyond an isolated function
  - Accelerator communication overhead
  - Cache management overhead
  - Operating System overhead
  - Byte addressable vs. block addressable
  - Scratchpad vs. Cache
  - Cache size to working data set size
  - Latency impact

Background

- M. Butts, J. Batcheller and J. Varghese, “An efficient logic emulation system,” *Proceedings 1992 IEEE International Conference on Computer Design: VLSI in Computers & Processors*, Cambridge, MA, 1992, pp. 138-141.
  - Realizer System
- “Virtex-7 2000T FPGA for ASIC Prototyping & Emulation,” https://www.xilinx.com/video/fpga/virtex-7-2000t-asic-prototyping-emulation.html
  - Prototype ARM A9 processor subsystem (dual-core, caches) mapped into a single Virtex-7 2000T FPGA
- Our approach uses the native hard IP cores/cache hierarchy and focuses on external memory

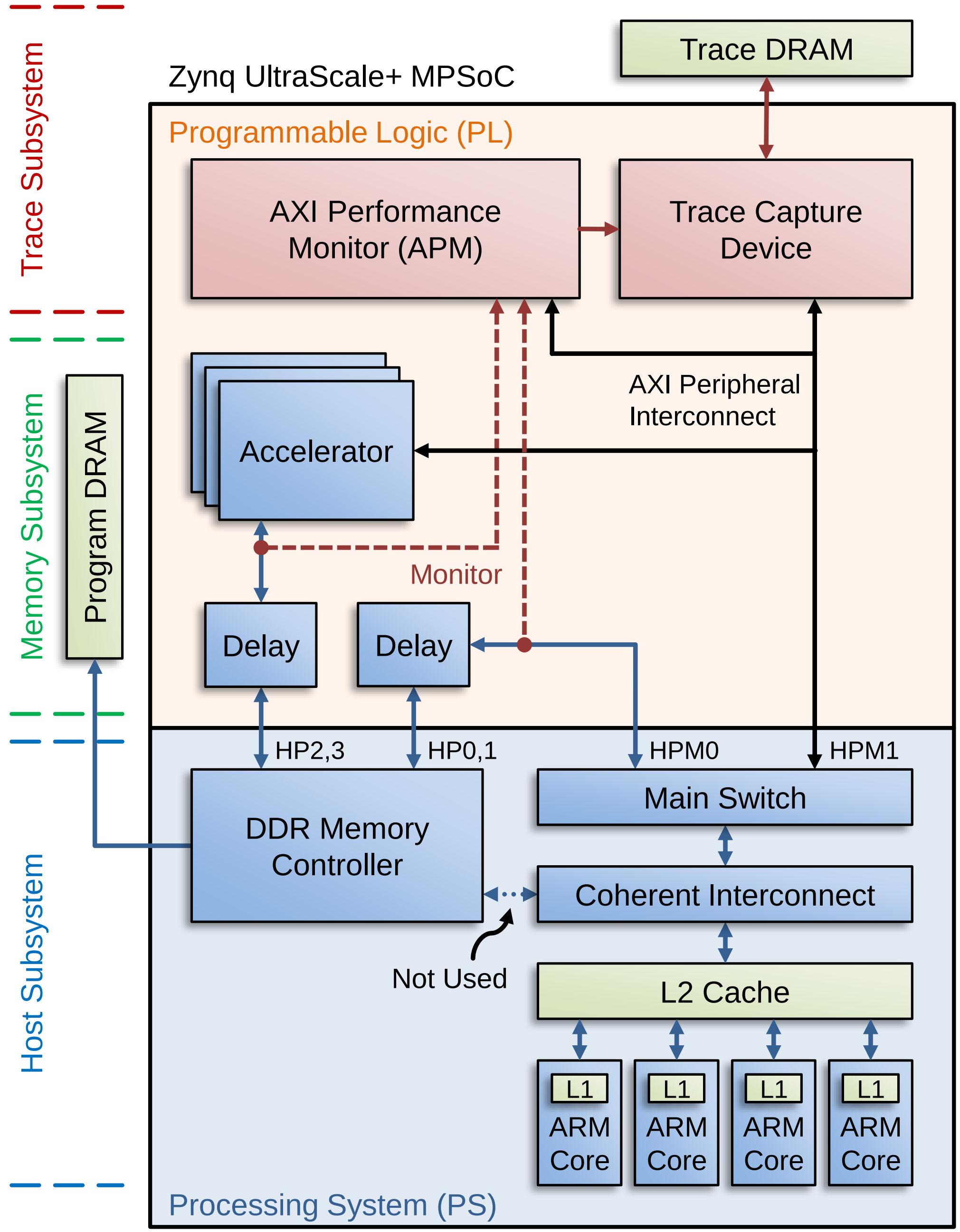
LiME (Logic in Memory Emulator)
ZCU102 development board with Xilinx Zynq UltraScale+ MPSoC device



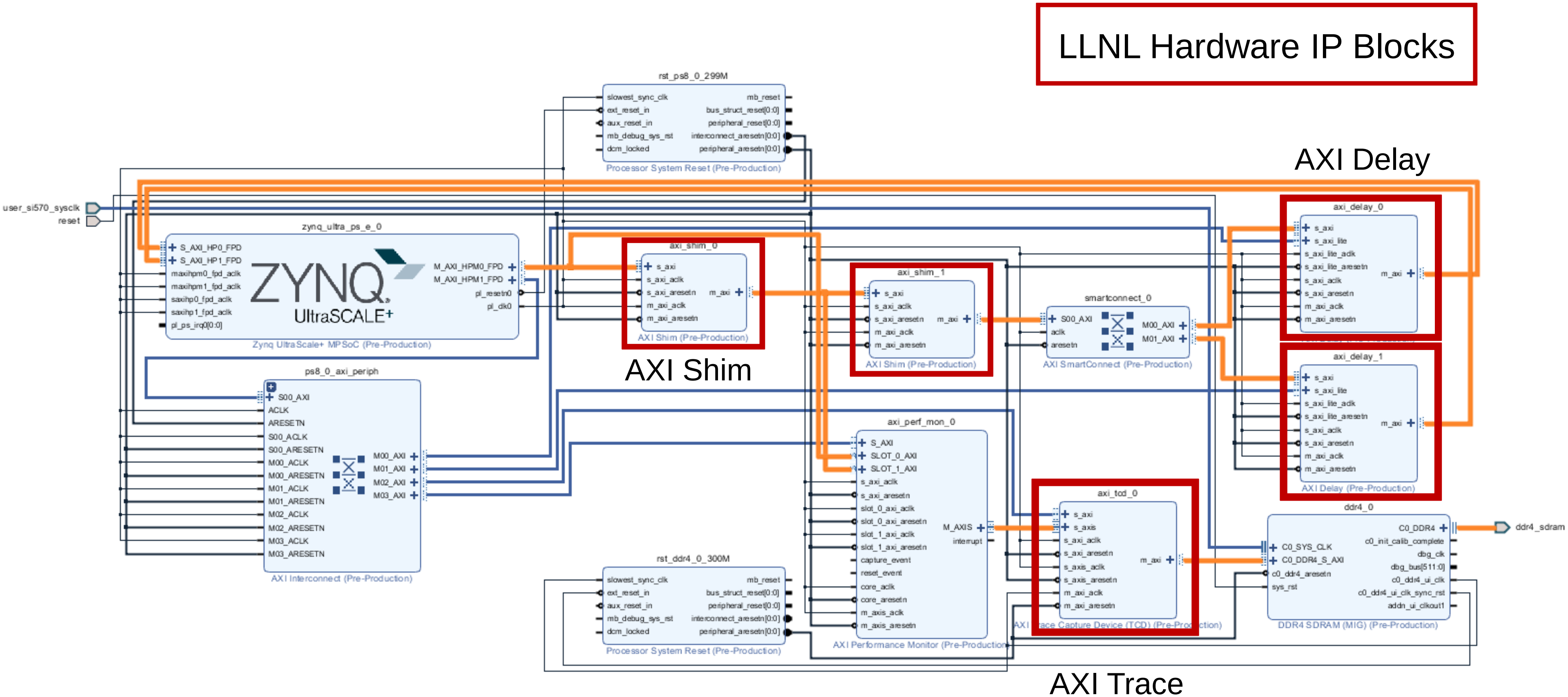
LiME (Logic in Memory Emulator) Implementation

- Use embedded CPU and cache hierarchy in Zynq MPSoC to save FPGA logic and development time
- Route memory traffic through hardware IP blocks deployed in programmable logic
- Emulate the latencies of a wide range of memories by using programmable delay units in the loopback path
- Capture time-stamped memory transactions using trace subsystem

Open Source:
http://bitbucket.org/perma/emulator_st/



LiME (Logic in Memory Emulator) Implementation

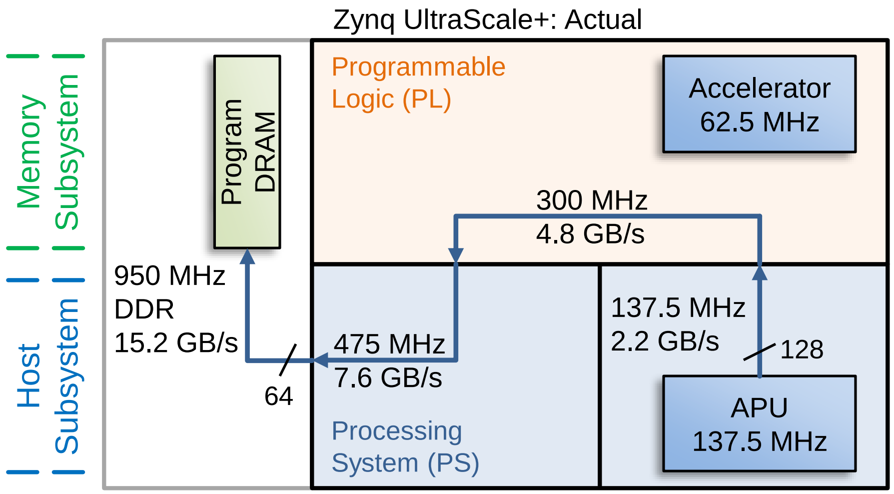


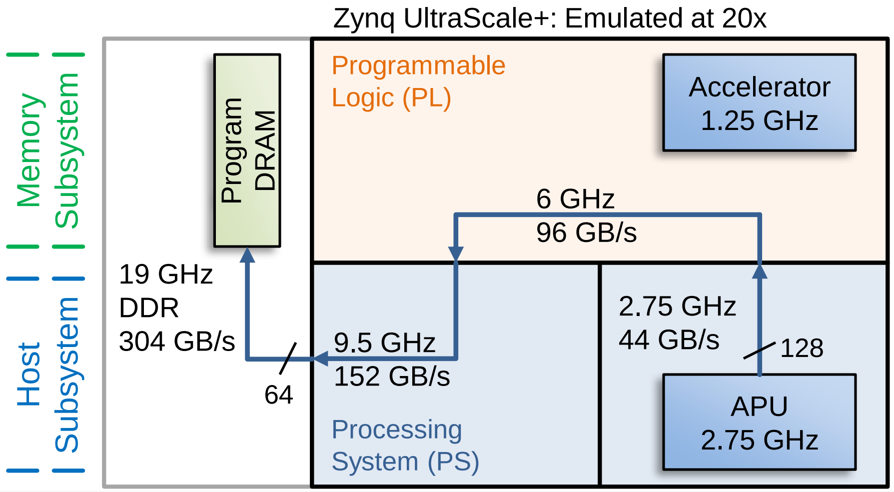
AXI Trace Capture Device

LiME uses only 13% of the device resources

Emulation Method
Clock Domains

- ARM cores are slowed to run at a frequency similar to programmable logic
- A scaling factor of 20x is applied to the entire system
- Other scaling factors can be used depending on the target peak bandwidth to memory
- CPU peak bandwidth is limited to 44 GB/s



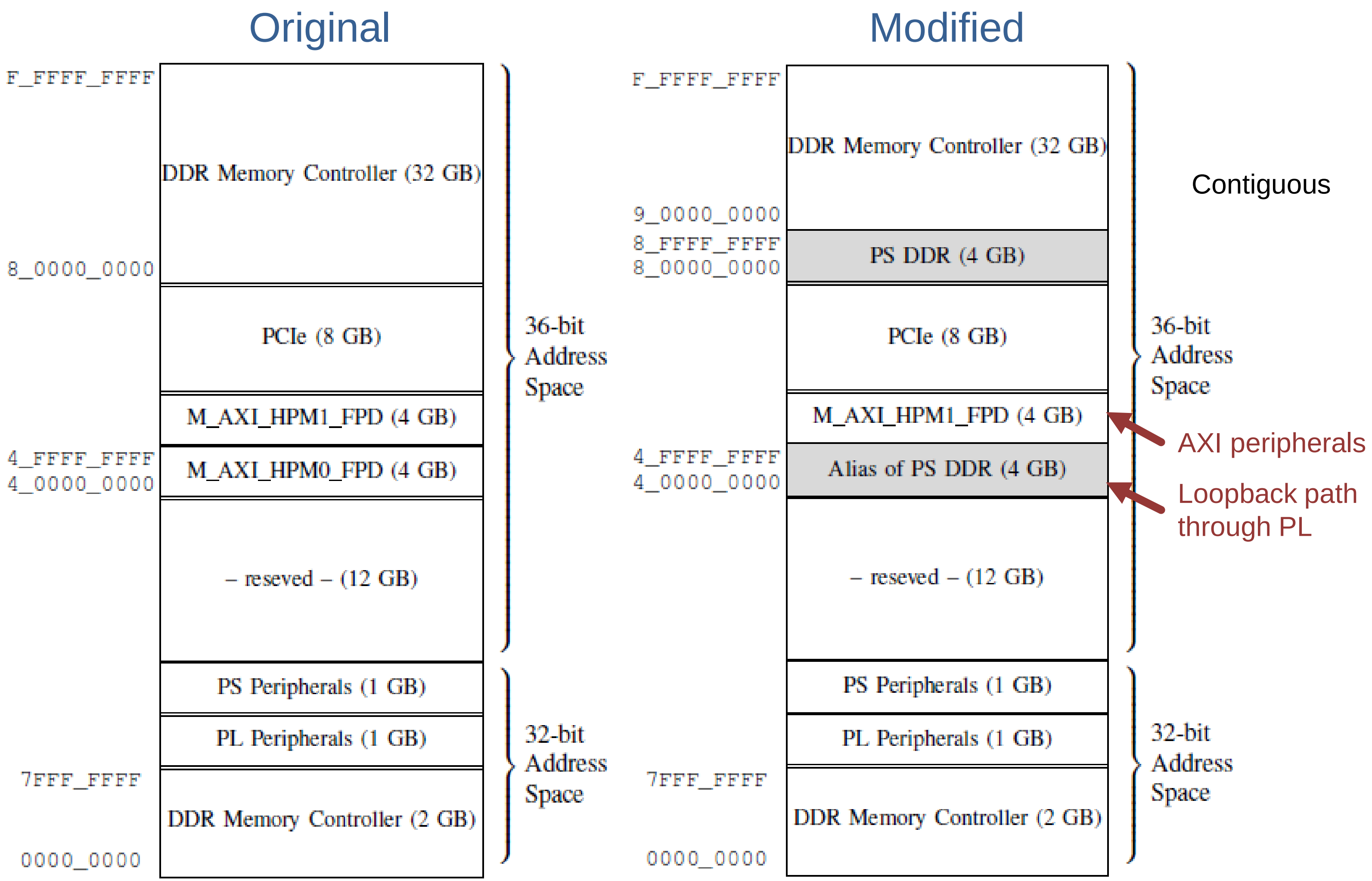


Emulation Method
Scaling by 20 Example

Component	Actual	Emulated
Memory Bandwidth (PL)	4.8 GB/s	96 GB/s
Memory Latency (PL)	230 ns	12 ns (too low)
Memory Latency (PL) w/delay	230 ns	12+88 = 100 ns
CPU Frequency	137.5 MHz	2.75 GHz
CPU Bandwidth	2.2 GB/s	44 GB/s
Accelerator Frequency	62.5 MHz	1.25 GHz
Accelerator Bandwidth	Up to 4.8 GB/s	Up to 96 GB/s

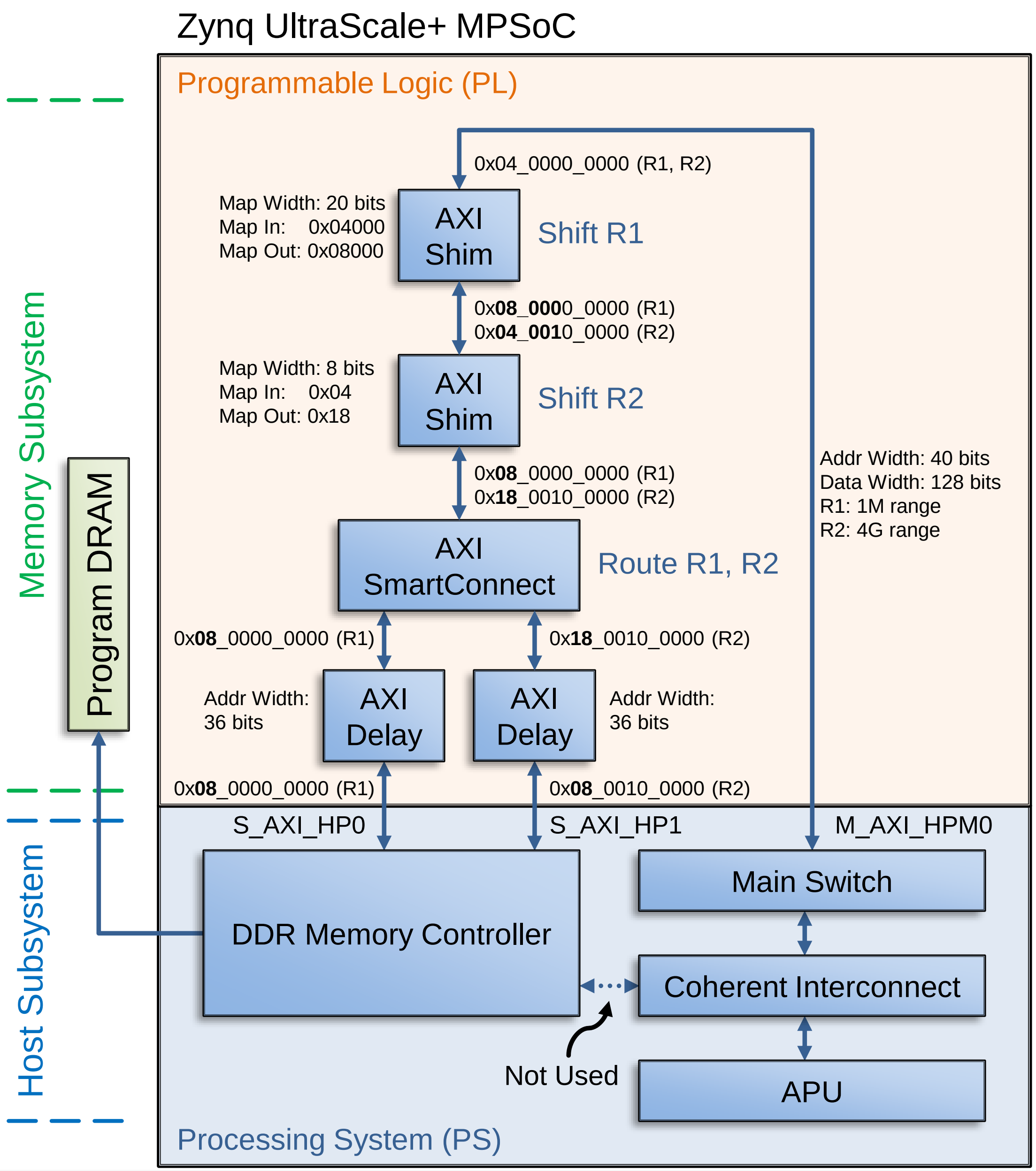
Delay is programmable over a wide range: 0 - 174 us
in 0.16 ns increments

Emulation Method
Address Space



Emulation Method
Delay & Loopback

- Address ranges R1, R2 intended to have different access latencies (e.g. SRAM, DRAM)
- Shims shift and separate address ranges (R1, R2) for easier routing
- Standard AXI Interconnect routes requests through different delay units
- Delay units have separate programmable delays for read and write access

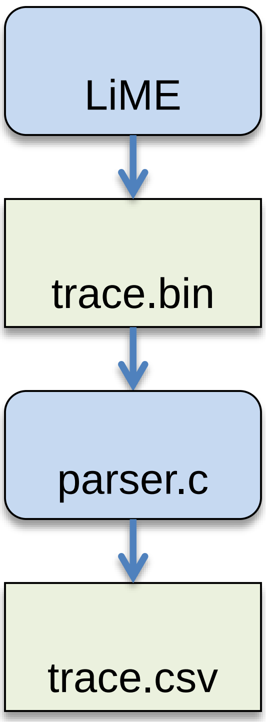


Emulation Method
Macro Insertion

- Insert macros at the start and end of the region of interest (ROI)
- CLOCKS_EMULATE/CLOCKS_NORMAL
  - Modify the clock frequencies and configure the delay units
- TRACE_START/TRACE_STOP
  - Trigger the hardware to start/stop recording memory events in Trace DRAM
- STATS_START/STATS_STOP
  - Trigger the hardware to start/stop the performance monitor counters
- TRACE_CAP
  - Save captured trace from Trace DRAM to SD card

```
    CLOCKS_EMULATE
    TRACE_START
    STATS_START
{
    /* --- MAIN LOOP --- repeat test cases NTIMES times --- */
    scalar = 3.0;
    for (k=0; k<NTIMES; k++)
    {
    times[0][k] = mysecond();
#ifdef _OPENMP
#pragma omp parallel for
#endif
    for (j=0; j<STREAM_ARRAY_SIZE; j++)
        c[j] = a[j];

    times[0][k] = mysecond() - times[0][k];

    times[1][k] = mysecond();
#ifdef _OPENMP
#pragma omp parallel for
#endif
    for (j=0; j<STREAM_ARRAY_SIZE; j++)
        b[j] = scalar*c[j];

    times[1][k] = mysecond() - times[1][k];

    times[2][k] = mysecond();
#ifdef _OPENMP
#pragma omp parallel for
#endif
    for (j=0; j<STREAM_ARRAY_SIZE; j++)
        c[j] = a[j]+b[j];

    times[2][k] = mysecond() - times[2][k];

    times[3][k] = mysecond();
#ifdef _OPENMP
#pragma omp parallel for
#endif
    for (j=0; j<STREAM_ARRAY_SIZE; j++)
        a[j] = b[j]+scalar*c[j];

    times[3][k] = mysecond() - times[3][k];
    }
}
    STATS_STOP
    TRACE_STOP
    CLOCKS_NORMAL
    TRACE_CAP
```

Memory Trace Capture

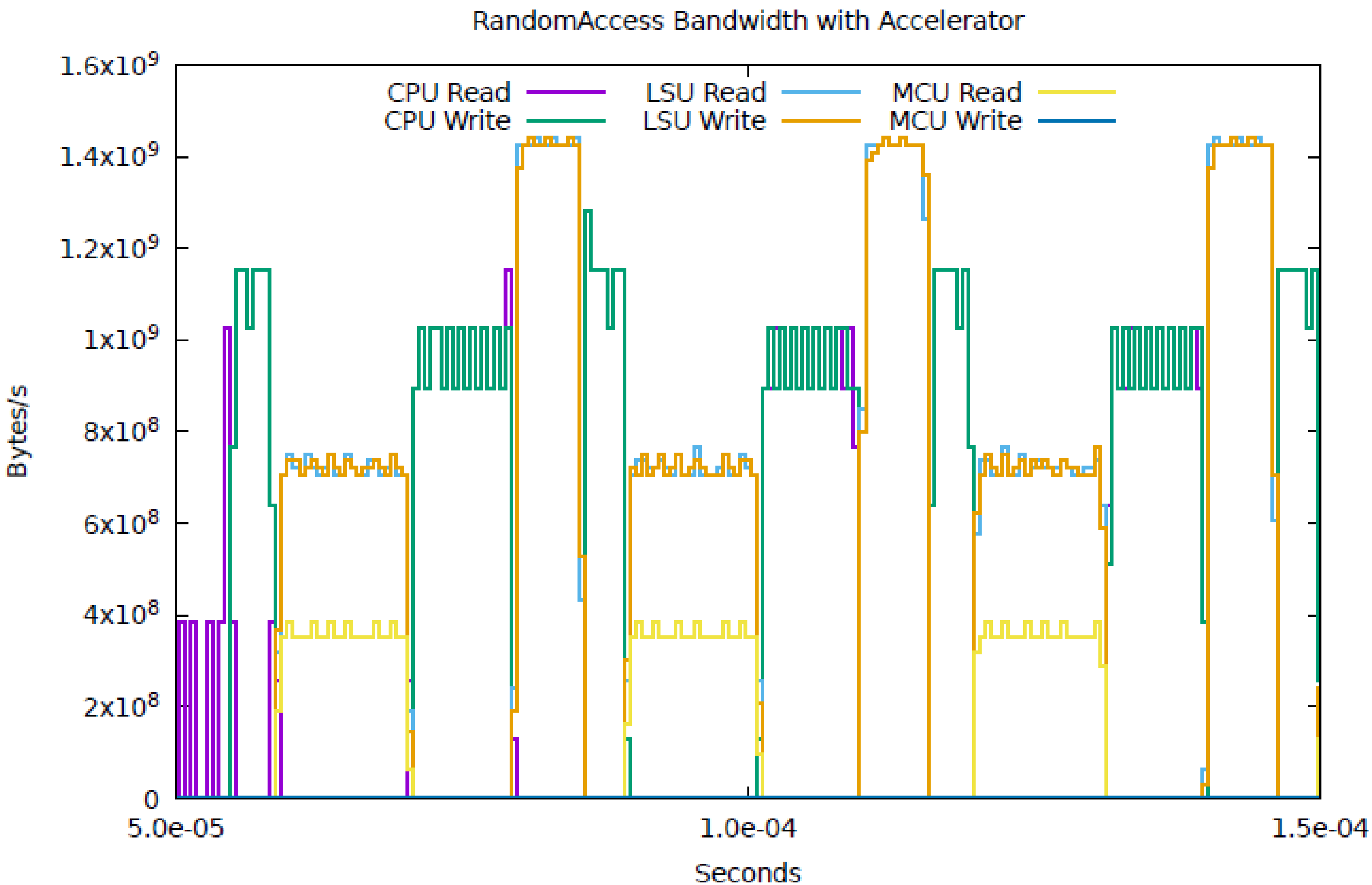


Source	Type	Address	Length	AXI ID	Time
0	W	0x400082F80	64	525	481545
0	W	0x400082FC0	64	525	482101
0	R	0x4002011C0	64	1165	482432
0	R	0x400201200	64	1293	482441
0	R	0x400201240	64	1037	487379
0	R	0x400201280	64	1037	492539
1	W	0x400080000	8	3	498523
1	R	0x400082000	16	0	493495
1	W	0x400080008	8	3	498557
1	W	0x400080010	8	3	503270
1	W	0x400080018	8	3	503304
1	W	0x400080020	8	3	503400

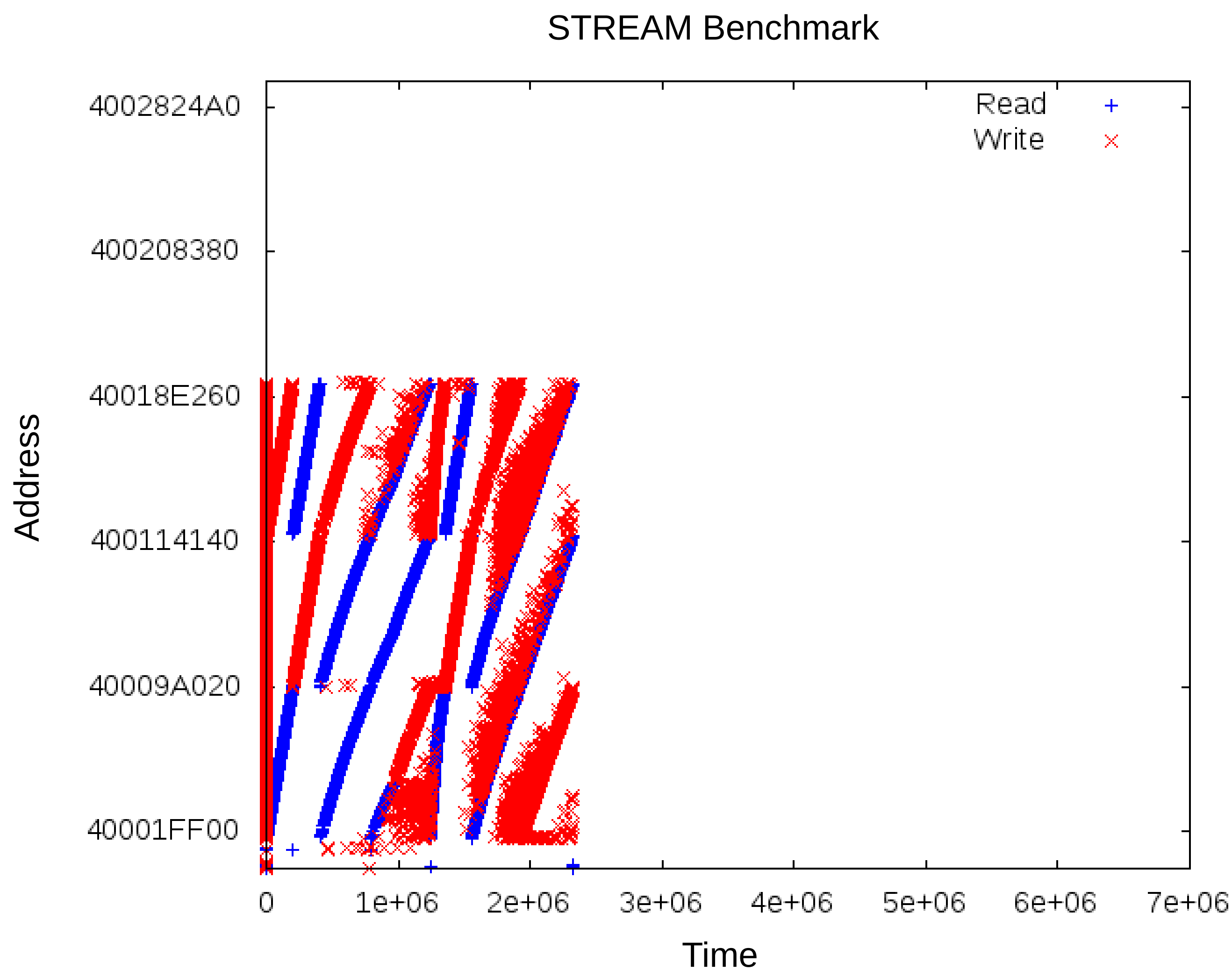
CPU = 0, Accelerator = 1

Each count represents 0.16 ns

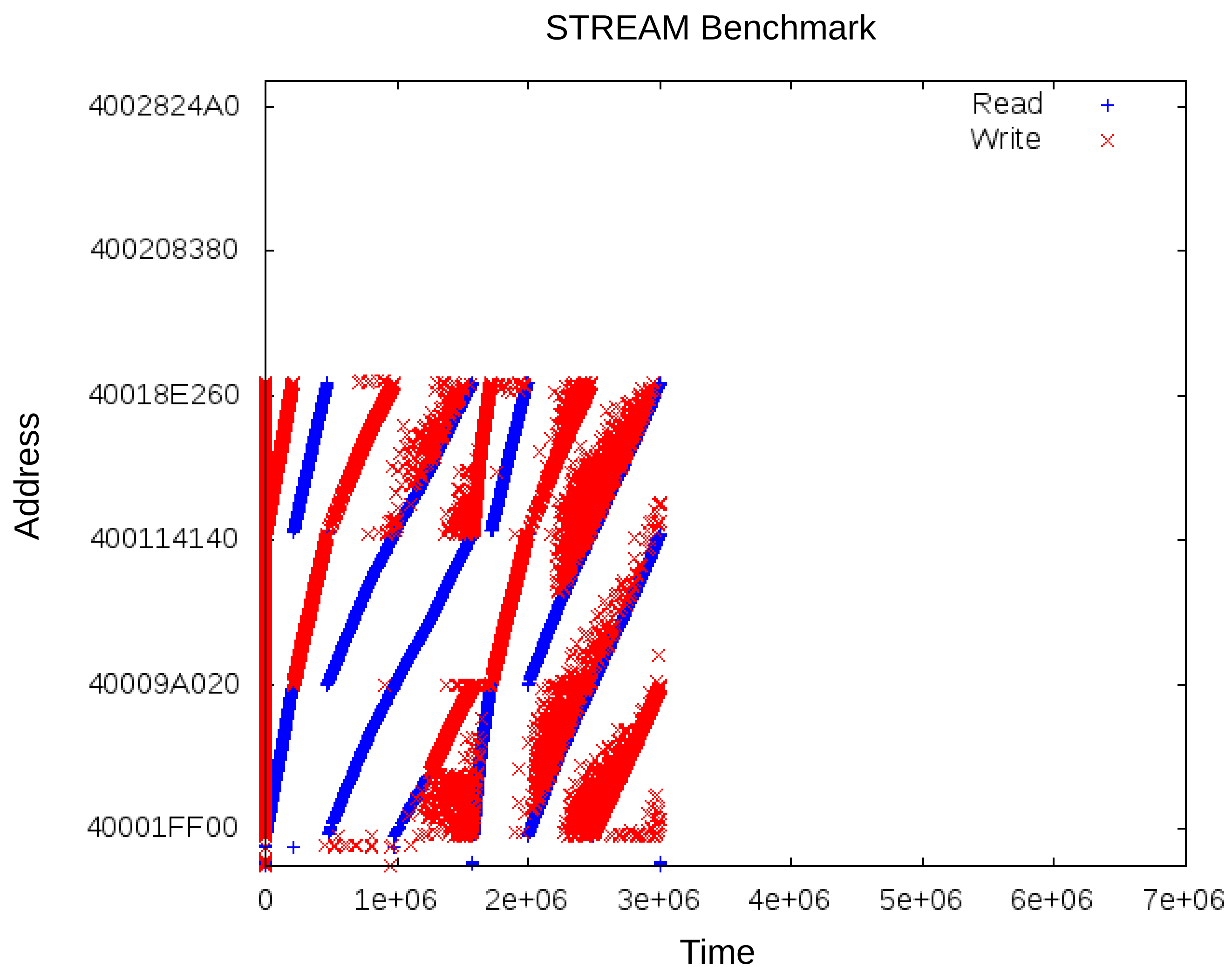
Use Cases
Bandwidth Analysis from Trace



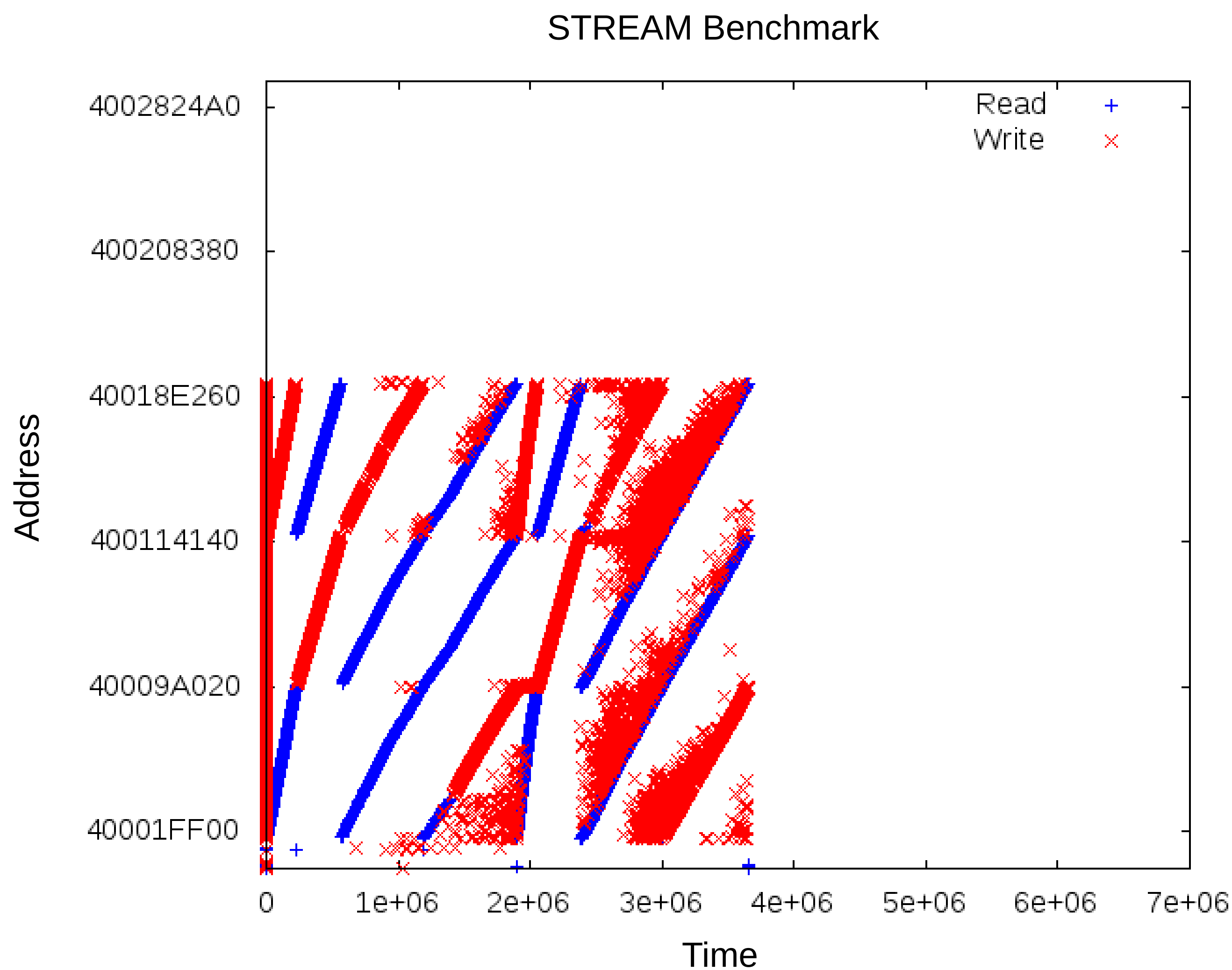
Use Cases
Access Pattern Analysis from Trace



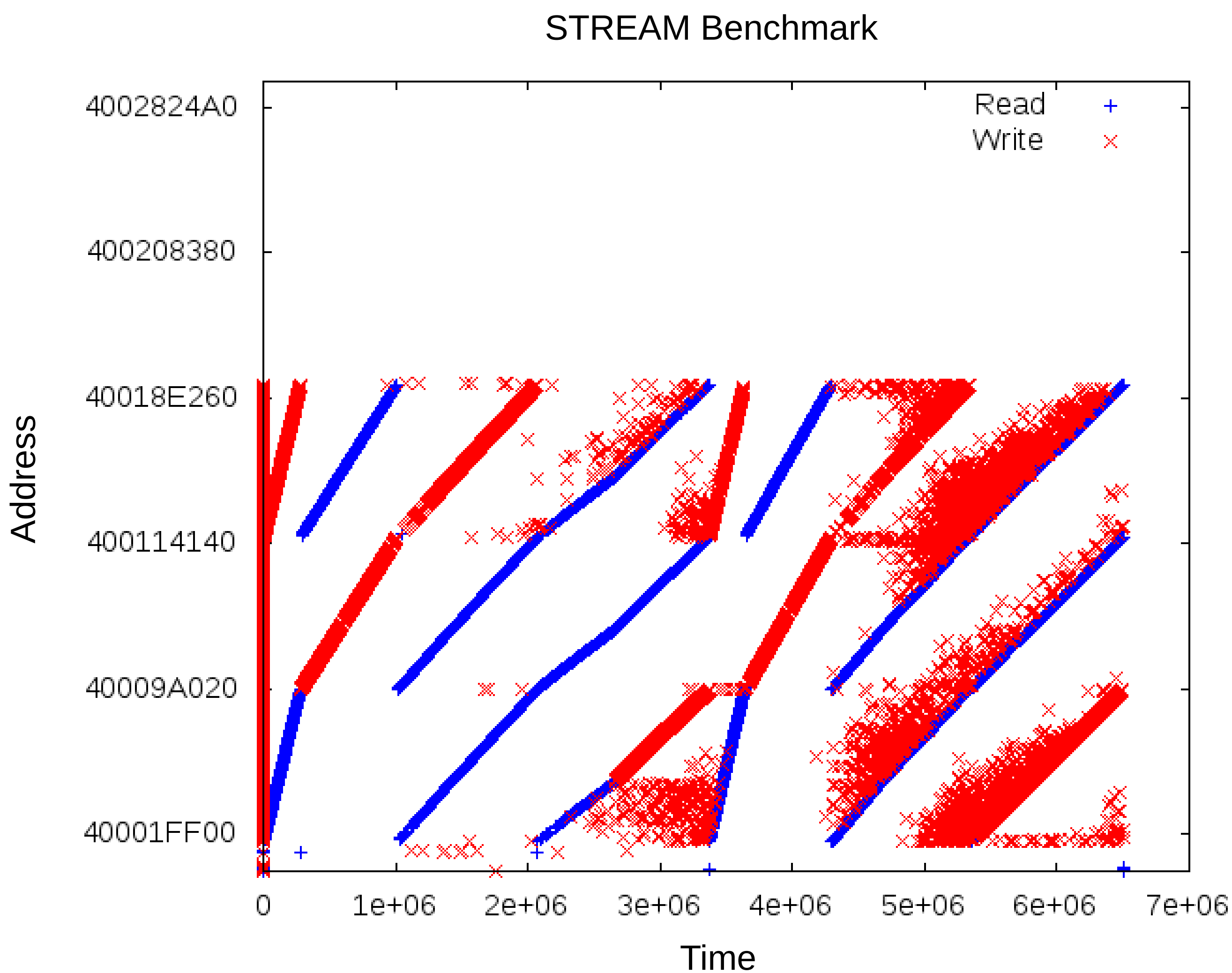
Use Cases
Access Pattern Analysis from Trace



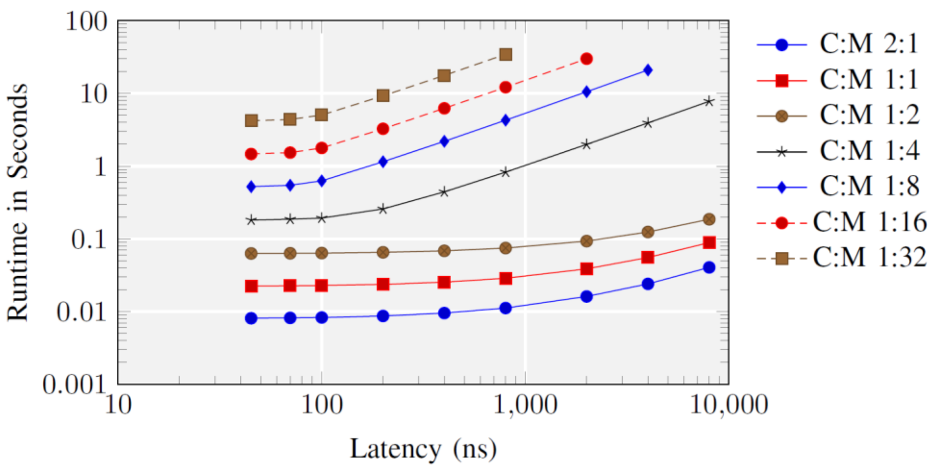
Use Cases
Access Pattern Analysis from Trace



Use Cases
Access Pattern Analysis from Trace

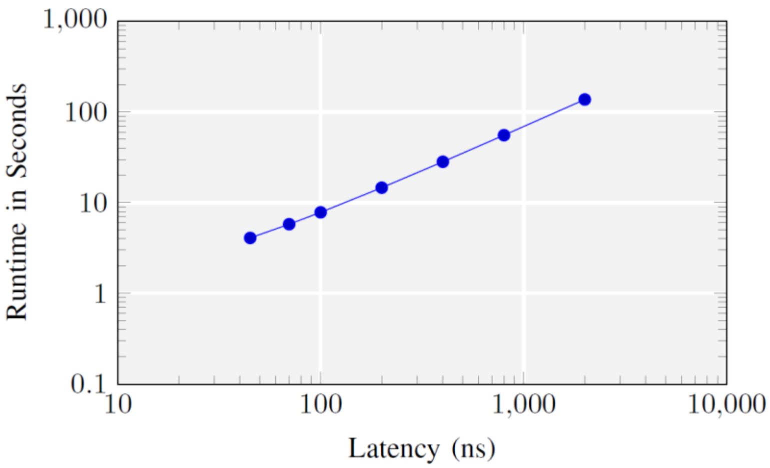


Use Cases
Evaluation of Future Storage Class Memory



DGEMM execution time on 64-bit processor at varying latencies and varying cache-to-memory ratios.

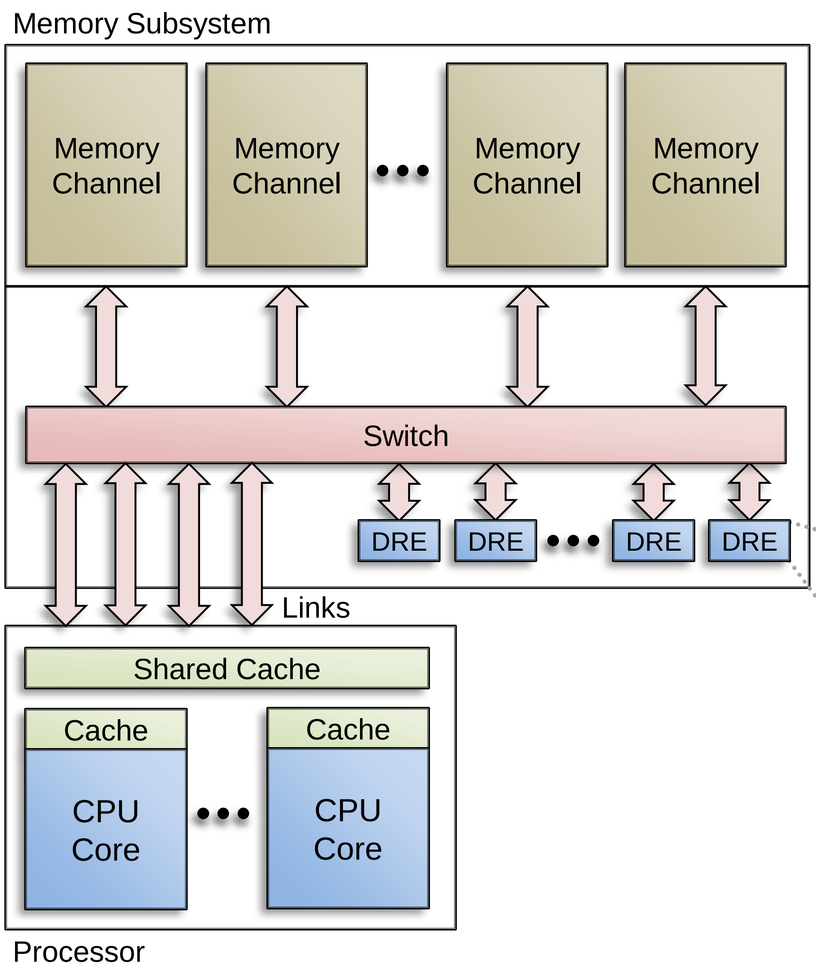
Cache can hide memory latency for a working set size up to twice the size of cache.



SpMV execution time on 64-bit processor at varying latencies with a cache-to-memory ratio of 1:112.

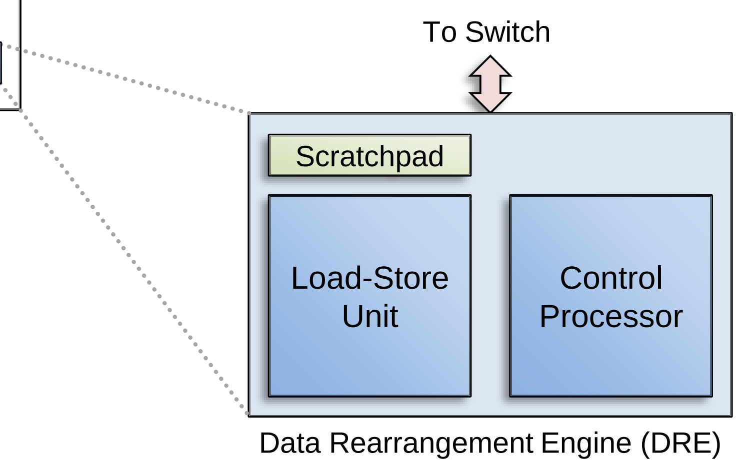
Latency has direct impact. Application will need a high level of concurrency and greater throughput to offset the loss in performance.

Use Cases
Evaluation of Near-Memory Acceleration Engines

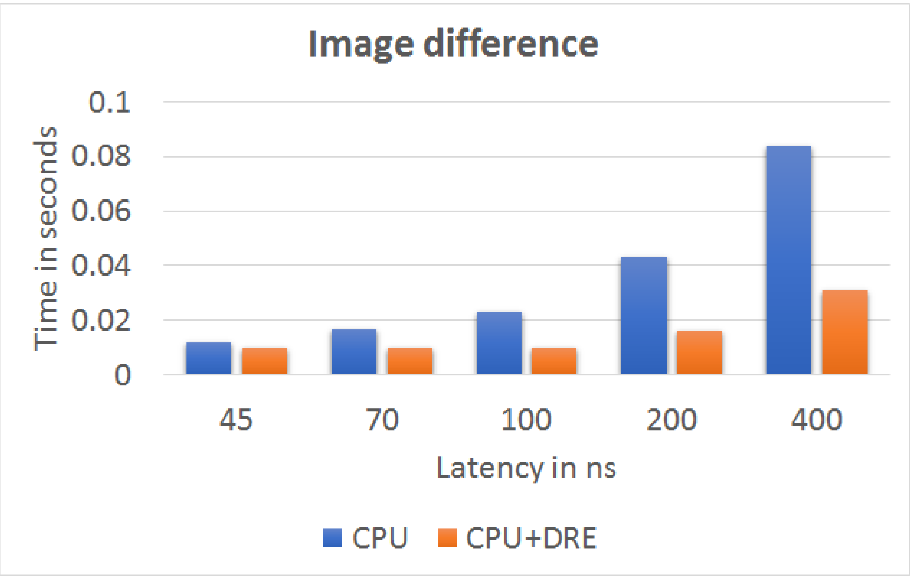


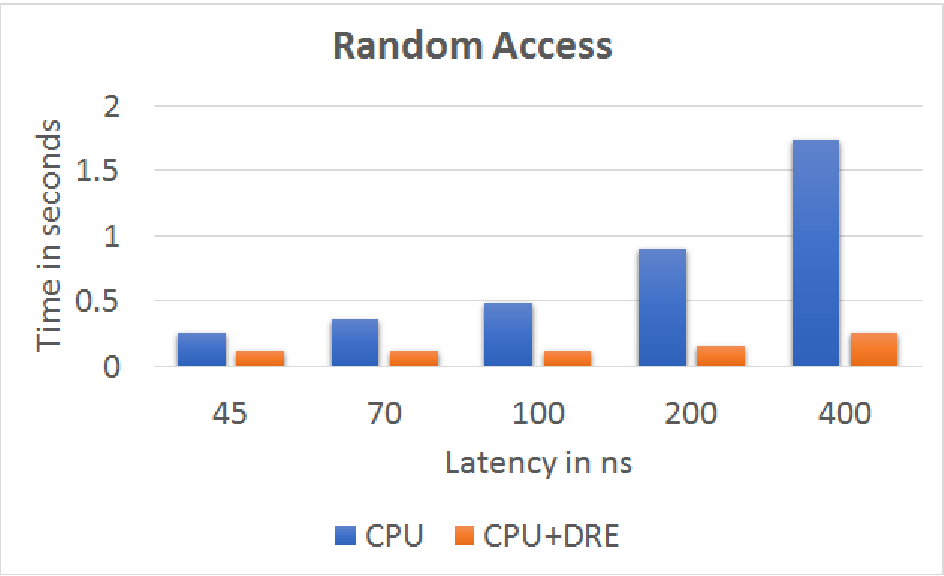
- Multiple Memory Channels
- Up to 16 concurrent memory requests

- DREs are located in the Memory Subsystem
- Scratchpad is used to communicate parameters and results between CPU and accelerator



Use Cases
Evaluation of Near-Memory Acceleration Engines



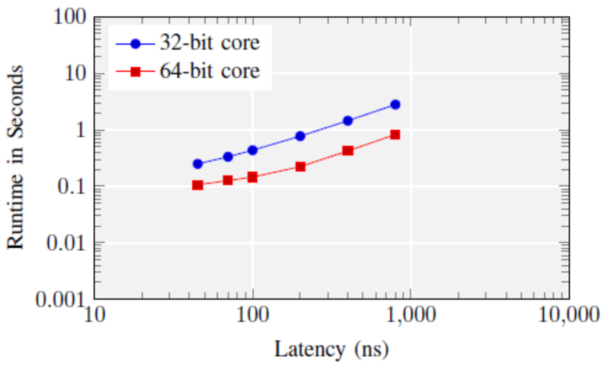


The results demonstrate that substantial speedup can be gained with a DRE due to the higher number of in-flight requests issued by the near-memory accelerator.

Use Cases
Comparing Performance Across CPUs

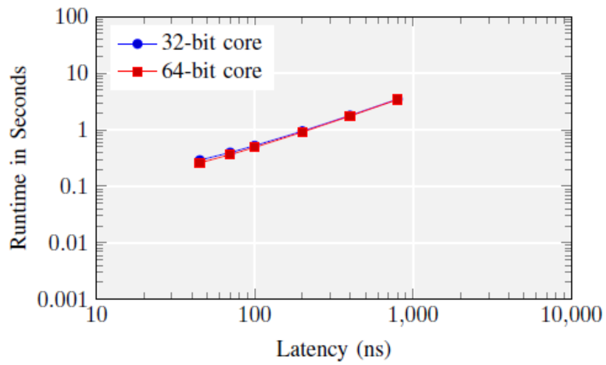
- 32-bit ARM A9 (Out-of-order with 11-stage pipeline) using Zynq 7000
  - L1 Cache: Two separate 32 KB (4-way set-associative) for instruction and data
  - L2 Cache: Shared 512 KB (8-way set-associative)
  - Cache Line Size: 32 Bytes

- 64-bit ARM A53 (In-order with 8-stage pipeline) using Zynq UltraScale+
  - L1 Cache: Two separate 32 KB (4-way set-associative) for instruction and data
  - L2 Cache: Shared 1MB (16-way set-associative)
  - Cache Line Size: 64 Bytes



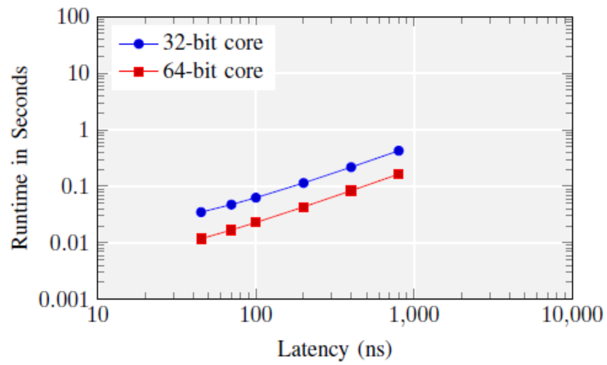
(a) STREAM-triad

Bandwidth-dominated STREAM-triad runs significantly faster on the 64-bit processor with wider data paths.



(b) Random Access

Random Access is mostly dependent on memory latency with little difference from CPU architecture.



(c) Image Difference

Image Difference requires some computation, giving the 64-bit core an advantage.

Summary & Conclusions

- LiME hardware/software infrastructure is available as open source.
- LiME takes a novel approach to evaluating memory systems from the perspective of application performance
  - Employ a state of the art MPSoC
  - Route CPU memory traffic through programmable logic for visibility
  - Use programmable delay units to model various memory technology
  - Store traces in a separate memory
- Emulate complex memory interactions in whole applications orders of magnitude faster than software simulation
  - Can search a larger design and parameter space
- Example Case Studies
  - Capture, replay and analysis of an application's memory behavior
  - Evaluate the use of emerging storage class memories
  - Emulate acceleration hardware co-located with the memory subsystem
  - Compare performance of 32-bit and 64-bit processors

Future Work

- Develop delay units with more sophisticated memory models
  - Use a statistical model
  - Implement more parameters (limit bandwidth, conflicts)
- Study full workloads with a mix of applications under Linux
- Evaluate performance of additional accelerators in programmable logic
- Explore synchronization and communication methods between CPU and accelerators
- Tools to associate memory trace addresses with program variables
- Add hardware compression to trace capture output for longer traces

References

- LiME Open Source Release, for ZC706 platform
  - "Logic in Memory Emulator" with benchmark applications available at http://bitbucket.org/perma/emulator_st

- S. Lloyd and M. Gokhale, "In-memory data rearrangement for irregular, data intensive computing," *IEEE Computer*, 48(8):18–25, Aug 2015.
- M. Gokhale, S. Lloyd, and C. Hajas, "Near memory data structure rearrangement," *International Symposium on Memory Systems*, pp. 283–290, Washington DC, Oct 2015.
- M. Gokhale, S. Lloyd, and C. Macaraeg, "Hybrid memory cube performance characterization on data-centric workloads," *Workshop on Irregular Applications: Architectures and Algorithms*, 7:1–7:8, Austin, TX, Nov 2015.
- S. Lloyd and M. Gokhale, "Evaluating the feasibility of storage class memory as main memory," *International Symposium on Memory Systems*, pp. 437–441, Alexandria, VA, Oct 2016.
- S. Lloyd, and M. Gokhale, "Near memory key/value lookup acceleration," *International Symposium on Memory Systems*, pp. 26–33, Alexandria, VA, Oct 2017.

Lawrence Livermore
National Laboratory