# LiME: An Open Source Memory System Emulation Platform

Abhishek Kumar Jain
Lawrence Livermore National Laboratory, Livermore, CA
jain7@llnl.gov

Scott Lloyd
Lawrence Livermore National Laboratory, Livermore, CA
lloyd23@llnl.gov

Maya Gokhale
Lawrence Livermore National Laboratory, Livermore, CA
gokhale2@llnl.gov

## ABSTRACT

We present a lightning demo of Logic in Memory Emulator (LiME), an open source hardware/software library that can emulate memory latencies ranging from 10's of nanoseconds to microseconds. LiME also can non-intrusively capture and time stamp every memory read and write command presented to the external memory system by an application running on the SoC's CPU. LiME addresses the shifting focus of high performance computing systems from CPU centric to memory and data centric. LiME allows emulation with significant savings in FPGA logic and development time by leveraging fixed IP modules such as the ARM cores, caches, and memory controllers within Xilinx SoC/MPSoC devices.

**ACM Reference Format:**
Abhishek Kumar Jain, Scott Lloyd, and Maya Gokhale. 2017. LiME: An Open Source Memory System Emulation Platform. In *Proceedings of Workshop on Open Source Supercomputing*. ACM, New York, NY, USA, 2 pages.
https://doi.org/XX.XXX/XXX

## 1 MOTIVATION

For memory intensive workloads, modern high performance multi-core processors usually spend a significant amount of time waiting for memory to serve requests, leaving processor cycles idle. Types and capacities of memory are proliferating (For example, HBM, DDRx, SCM varieties, Flash (w/DRAM buffer)) and there is a significant need to assess these emerging memory architectures and configurations on realistic application access patterns. Investigations of memory performance have typically been conducted using trace-driven simulation in architecture simulators [3, 6]. Due to the detailed software simulation of CPU microarchitecture and cache hierarchy, these simulators are very slow. Programmable hardware such as FPGAs have been used historically for performing system simulations by emulating complex CPU, cache hierarchy and memory controller [2, 7]. This approach is advantageous as emulation is closer to real-time, and low level emulation may improve the accuracy of the predictions. However, it takes tremendous resources and development time to create the emulation architecture.

## 2 APPROACH

Inclusion of hard IP modules, such as embedded ARM processors, cache hierarchy and memory controller, now makes modern SoC platforms [1] an excellent choice for exploring architectural changes in memory hierarchy and composition. Our approach of developing the emulation platform around an SoC platform allow us to use these IP modules, thus saving FPGA logic and development time [4, 5]. However, interfacing fixed IP blocks with programmable logic blocks requires coordination of multiple clocks to accurately model the desired system. We have developed a method to route memory traffic through programmable logic to capture each request, and carefully coordinate the clock domains in a modern MPSoC, Zynq Ultrascale+ platform. With this system, we can faithfully emulate the latencies of a wide range of existing and proposed memories, while running the application orders of magnitude faster than full software simulation. To model execution of a 2.75GHz CPU causes only 20x slowdown from actual execution.

However, using fixed CPU and cache also limits the emulator to a specific processor and cache hierarchy. To assess the impact of fixed CPU, we compare emulated performance profiles from two LiME platforms, the Zynq 7000 with 32-bit dual ARM A9 and the Zynq Ultrascale+ with 64-bit quad ARM A53. We show that though absolute performance numbers differ, the performance trends track closely between the two platforms.

## 3 METHOD

Figure 1 shows the architecture of Zynq Ultrascale+ MPSoC with emulation framework in which the ARM cores can run at a clock frequency of up to 1100 MHz. Since the FPGA logic on the emulation board is limited to a maximum clock frequency of about 300 MHz, the ARM core is slowed to run at a comparable frequency. For the experiments in this paper, the ARM runs at 137.5 MHz and when multiplied by a scaling factor of 20, represents the emulated CPU frequency of 2.75 GHz. The off-chip memory runs at its normal rate of 1900 Mt/s. In the emulation infrastructure, memory transactions are delayed to scale the latencies of the emulated memory relative to the emulated CPU clock frequency. Delay units in the FPGA fabric are programmable at a range of 0–262 us in 0.25 ns increments. The application's load/store requests not satisfied in cache are routed to the delay units, and from there through a hard IP memory controller to off-chip DDR4 memory, crossing multiple clock domains as shown in Figure 2.

## 4 RESULTS AND DISCUSSION

The Figure 3, 4 and 5 present results of emulating varying latencies of memory on various latency sensitive workloads on the two emulation platforms. While the 64-bit processor has faster run times, the trend lines track closely for a very wide range of emulated latencies, demonstrating that for purposes of memory latency emulation, the trends are valid despite significant differences in CPU (out-of-order, 32-bit vs. in-order, 64-bit), cache hierarchy and cache line size.


*Workshop on Open Source Supercomputing, 2017, Denver, Colorado, USA*

ACM ISBN XXX-XXXX-XX-XXX/XX/XX…$XX.00
https://doi.org/XX.XXX/XXX

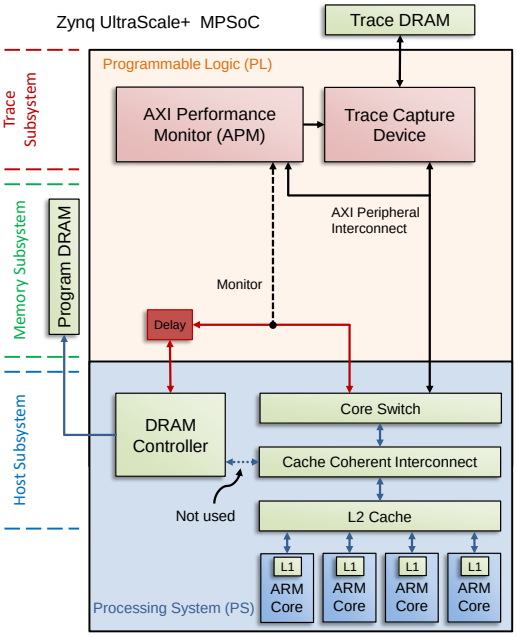


**Figure 1: Zynq Ultrascale+ with emulation framework**

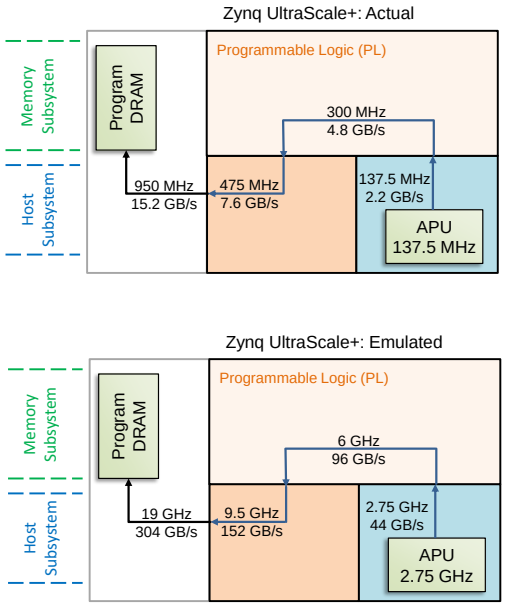


**Figure 2: Different clock domains in the loopback path for Zynq Ultrascale+**

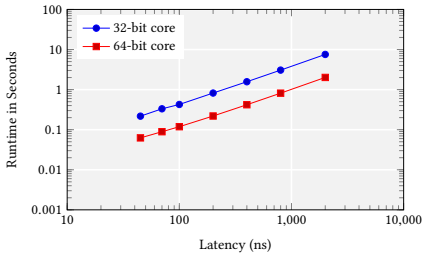


**Figure 3: STREAM-triad execution time at varying latencies.**

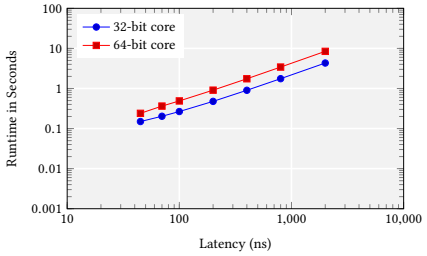


**Figure 4: Random Access at varying latencies.**

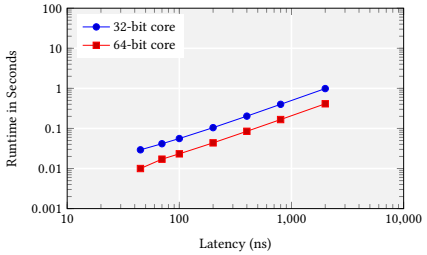


**Figure 5: Image difference at varying latencies.**

## REFERENCES


[1] Sagheer Ahmad, Vamsi Boppana, Ilya Ganusov, Vinod Kathail, Vidya Rajagopalan, and Ralph Wittig. 2016. A 16-nm multiprocessing system-on-chip field-programmable gate array platform. *IEEE Micro* 36, 2 (2016), 48–62.

[2] Hari Angepat, Derek Chiou, Eric S Chung, and James C Hoe. 2014. FPGA-accelerated simulation of computer systems. *Synthesis Lectures on Computer Architecture* 9, 2 (2014), 1–80.

[3] Todd Austin, Eric Larson, and Dan Ernst. 2002. SimpleScalar: An infrastructure for computer system modeling. *Computer* 35, 2 (2002), 59–67.

[4] Maya Gokhale, Scott Lloyd, and Chris Macaraeg. 2015. Hybrid memory cube performance characterization on data-centric workloads. In *Proceedings of the 5th Workshop on Irregular Applications: Architectures and Algorithms*. ACM, 7.

[5] Scott Lloyd and Maya Gokhale. 2016. Evaluating the feasibility of storage class memory as main memory. In *Proceedings of the Second International Symposium on Memory Systems*. ACM, 437–441.

[6] Milo MK Martin, Daniel J Sorin, Bradford M Beckmann, Michael R Marty, Min Xu, Alaa R Alameldeen, Kevin E Moore, Mark D Hill, and David A Wood. 2005. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. *ACM SIGARCH Computer Architecture News* 33, 4 (2005), 92–99.

[7] Graham Schelle, Jamison Collins, Ethan Schuchman, Perrry Wang, Xiang Zou, Gautham Chinya, Ralf Plate, Thorsten Mattner, Franz Olbrich, Per Hammarlund, et al. 2010. Intel nehalem processor core made FPGA synthesizable. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays (FPGA)*. ACM, 3–12.