Dynamic Delay Unit for the Logic in Memory Emulator(LiME)

Prateek Srivastava*, Scott Lloyd, Ivy Peng, and Maya Gokhale

*Department of Intelligent Systems Engineering, School of Informatics, Computing, and Engineering, Indiana University at Bloomington

Abstract

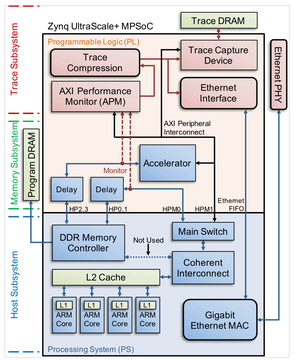
Modern memory systems have a broad range of bandwidth, latencies, and capacities. LiME provides a platform for system developers to emulate(not simulate) different architectures and get cycle accurate results with only 20x slower than real time execution speeds. However, it is orders of magnitude faster than simulators. This project introduces a dynamic delay unit which incorporates the latency distribution of a memory system by running real applications thus, emulating new memory technologies more accurately.

Introduction

LiME is a flexible emulator that runs on a Zynq Ultrascale+ MPSoC FPGA(Fidus Sidewinder or ZCU102) with 4 ARM cores along with the programmable logic, which allows users to model current and future memory systems or near memory accelerators. For instance, a proposed memory system such as the Intel Optane has a different latency compared to a conventional DRAM. Even a single memory technology could have multiple latencies due to memory bank conflict resolution. For a hardware designer, the freedom to control these latencies using a latency distribution curve acts as an extremely powerful tool.

The current LiME implementation has a fixed delay memory model where a memory request can be delayed over a wide latency range. However, memory systems show high variability with respect to latency under contention. This work enables dynamic latency at runtime.

LiME has three major parts – Loopback, accelerator, and trace capture. Loopback emulates a processor's interaction with memory system, accelerator emulates near memory acceleration functions, and the trace capture captures the memory events occurring in the loopback as well as the accelerator.



Fidus Sidewinder boards are Xilinx FPGA boards that have a Zynq Ultrascale+ FPGA with quadcore ARM processors.

Figure 1 : LiME Architecture[1] and an Xilinx FPGA board
https://fidus.com/wp-content/uploads/2018/11/Website-HAPS-Connection-top_540x477-e1552082194687.png

Methodology

There are two segments to the methodology. The first segment entails gathering latency distribution for the proposed memory system. It is done as follows:

1. Run real application with Intel Optane
2. Sample memory latency for the running application i.e. XSBench (XL problem size, 8 threads)

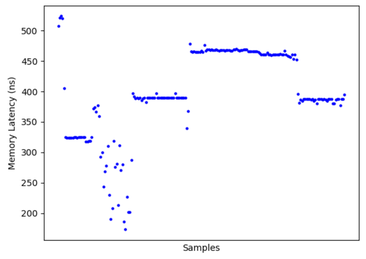


Figure 2 : Memory latency graph of the proposed system

3. Create a histogram of the latency using the sampled data

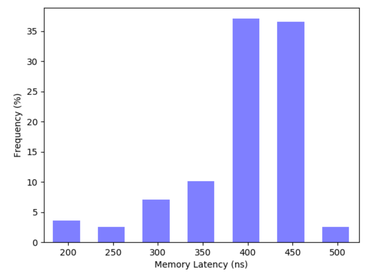


Figure 3 : Histogram of memory latency

4. Normalize the sampled data to a 2K sized vector maintaining the frequency distribution

The second part includes using the dynamic delay unit to emulate the proposed memory technology.

LiME Services

Function	Stake holders
Loopback	ARM ↔ Delay Unit ↔ Memory
Accelerator(ACC)	ACC ↔ Delay Unit ↔ Memory
Trace Capture	ARM, ACC → Trace Memory

Dynamic Delay Unit Architecture

The delay unit consists of 5 channels – address read(AR), read data(R), address write(AW), write data(W), and write response(B).

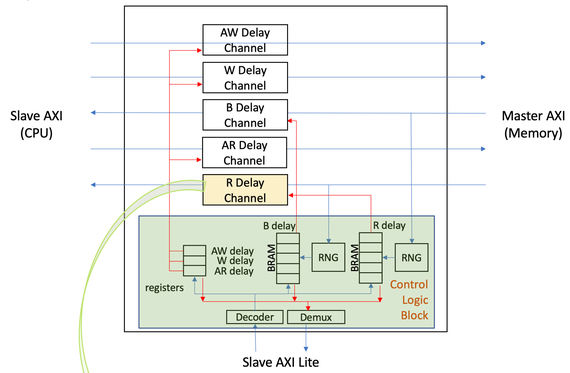


Figure 4 : Delay Unit with a Control Logic block

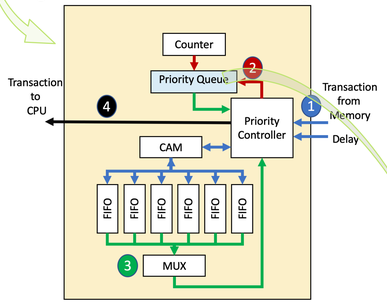


Figure 5 : Delay Channel (Priority Queue)

Each delay channel consists of a custom hardware priority queue which takes in the transaction and the delay picked from the distribution[2]. The priority controller stores the transaction data in the block RAM and sends the index along with its delay to the priority queue. After the delay, the transaction is popped from the delay channel, sending a response back to the ARM cores, thus, completing the read/write operation.

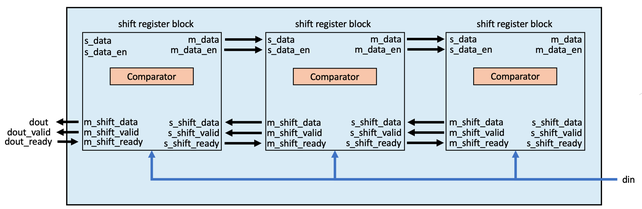


Figure 6 : Priority Queue

Simulation Results

- The control logic block has been engineered
- Detailed architecture for the delay channel along with the priority queue has been designed
- The waveforms for the control logic block have been tested with synthetic data

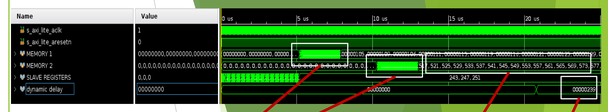


Figure 7 : Waveforms for the Control Logic Block

Initialization of Latency Table

Once the initialization of the latency table is completed, the memory has constant values and is never changed. The random number generator(RNG) then picks one of the values and delays the transaction respectively

Dynamic delay = 239

Synthesis Results

Site Type	Used	Available	Utilization %
CLB LUTs	1466	274080	0.53%
CLB Registers	4289	548160	0.78
Carry8	166	34260	0.48
Global Clock Buffers	2	404	0.5
Block RAM Tile	2	1824	0.00

References

[1] A. K. Jain, S. Lloyd and M. Gokhale, "Microscope on Memory: MPSoC-Enabled Computer Memory System Assessments," *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Boulder, CO, 2018, pp. 173-180.

[2] Moon, Sung-Whan & Rexford, Jennifer & Shin, K.G.. (2000). Scalable hardware priority queue architectures for high-speed packet switches. Computers, IEEE Transactions on. 49. 1215 - 1227. 10.1109/12.895938.

LLNL-POST-781284