Build-A-PIM: A Modular Framework for Exploring the "What-if's" of Processing in Memory

William Bradford, Farzana Ahmed Siddique, Nebil Ozer, Kevin Skadron, Sandhya Dwarkadas

University of Virginia

- Processing in Memory (PIM) accelerates memory-bottlenecked applications by placing small processor cores inside modern DRAM technology
- Memory technology is relatively consistent between PIM devices, compute technology *differs dramatically*
- Prior PIM simulators developed independently for fixed:
  - Individual applications (e.g. SAIT's PIMSimulator)
  - Device structures (e.g. PIMeval)
- Extension to new microarchitectural features difficult
- **Our Goal**: separate cores from memory simulation for maximal code reuse in bank-level PIM design

→ *Build-A-PIM* enables the design of diverse microarchitectural functionality

- Hardware implementation not enforced
- Easily customizable & interchangeable simulated elements (see Fig A):
  - PIM Device Locus of Control (CPU-driven or device-driven)
  - PIM Core ISA & Logic (Based on prior work: PIMeval)
  - PIM Device Topology
  - Misc. hardware additions

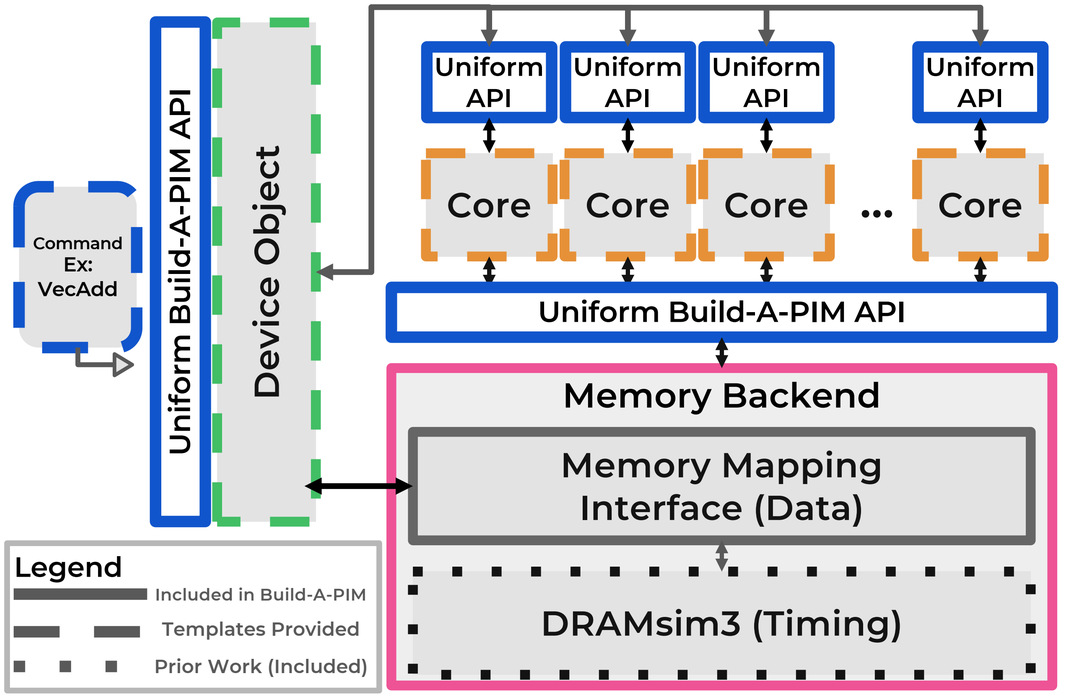

Fig B: The overall architecture of a Build-A-PIM device. High-level logic (e.g. vector addition) is handled through a bulk-synchronous device class, which communicates with each core individually. Cores are responsible for communicating with the memory backend independently.

Semiconductor Research Corporation

PRISM