Energy Efficient Synchronization and Parallelism on the VirtualSoC Simulator

Callen Rain[1], Peng Zhao[1], Dimitra Papagiannopoulou[2], Tali Moreshet[1], Iris Bahar[2], Maurice Herlihy[2]
Department of Engineering, Swarthmore College, Swarthmore PA 19081
1. Swarthmore College 2. Brown University

Background

Embedded Platform

Embedded systems are highly constrained in size and battery life. High performance mobile devices such as smart phones require architectures that are designed with energy efficiency in mind.

From Single Core to Many-Core

- Consumer demand for higher computing capability encourages a shift from **single core architectures** to **multi-core architectures** (2-8 cores)
- This shift means that tasks must be **parallelized** among cores
- While primarily implemented in research environments, **many-core** systems can achieve massive scalability by **clustering cores** and organizing a network interface within and between clusters

Hazards

When attempting to run applications on systems with many cores, issues arise with how memory is shared between processors. Shared data structures need to be protected in some way to prevent hazards from occurring. Unsynchronized cores may unknowingly add **duplicate or incorrect information** to shared memory.

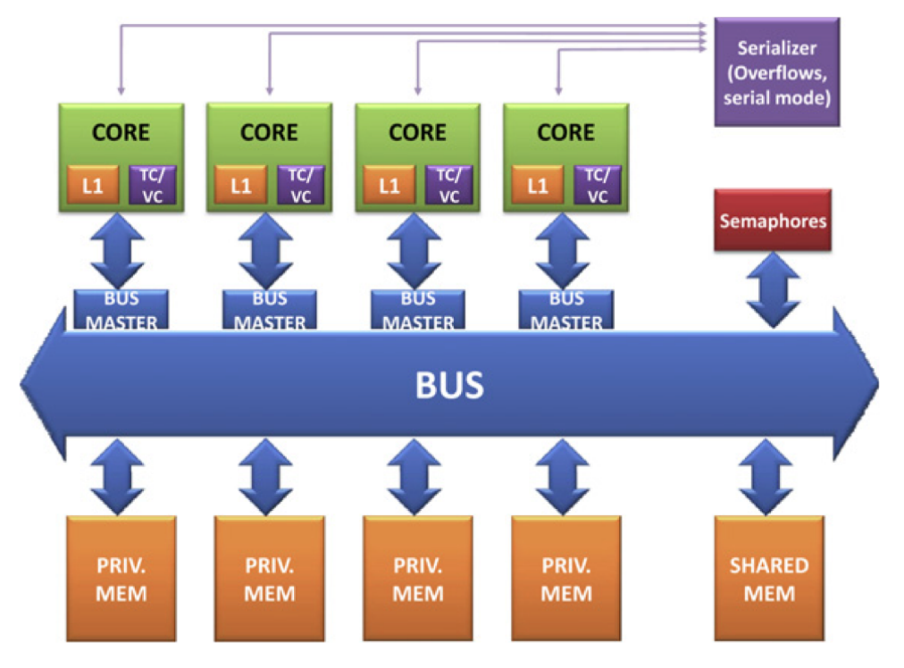
Synchronization

- To maintain synchronization between cores, shared data structures can be protected with locks or transactions
- **Locks** force attempts to access memory to serialize so that no hazards can occur
  - If an application spends much of its time accessing shared data, using locks will waste the advantages of a parallel system
- **Transactions** are bundles of operations that are denoted by the programmer
  - Transactions are speculatively executed in parallel but are tracked in hardware to avoid conflicts
  - If a conflict is detected, the transaction aborts and memory is returned to the state it was in before the transaction

Virtual Platforms

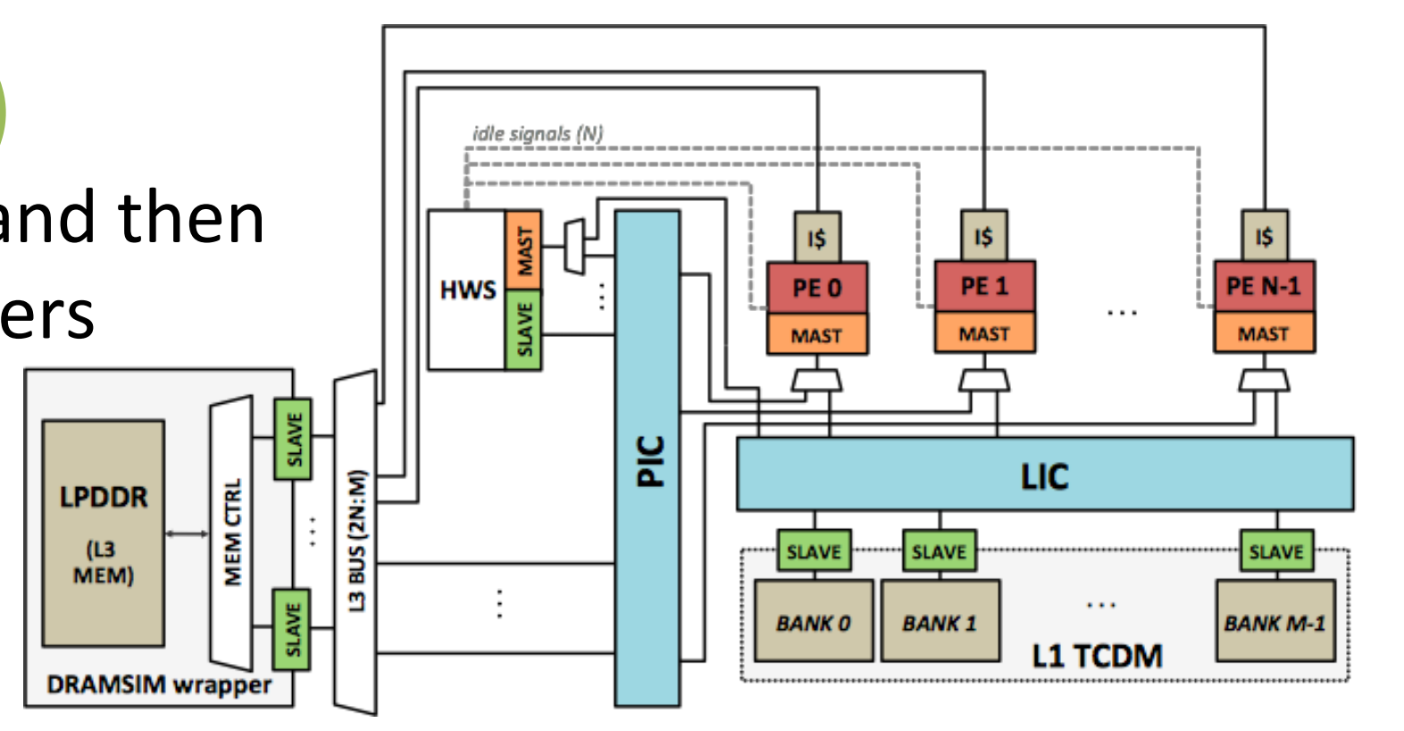
- Allow testing of architectures that would be expensive to fabricate
- No operating system
- Written using the SystemC library
- Developed by researchers in University of Bologna

MPARM (Multi-Processor ARM)



- Previously used platform for transactional memory research; runs many benchmarks
- Bus-based architecture only supports up to 16 cores
- Bus becomes bottlenecked when more cores are added

VSoC (Virtual System-On-Chip)



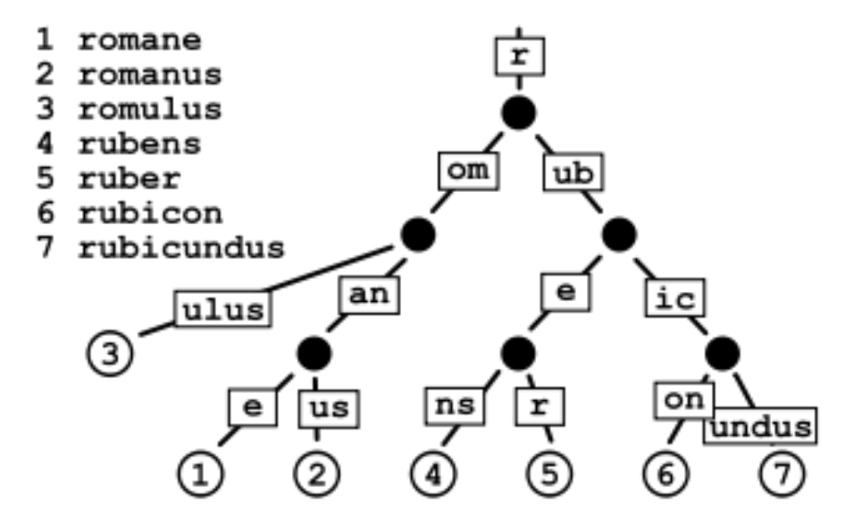
- Offers scalability by clustering cores and then running a network between the clusters
- Originally ran very few parallel benchmarks
- Contains a Tightly Clustered Data Memory (TCDM), a shared memory bank connected to the cores through a logarithmic interconnect

Porting Benchmarks

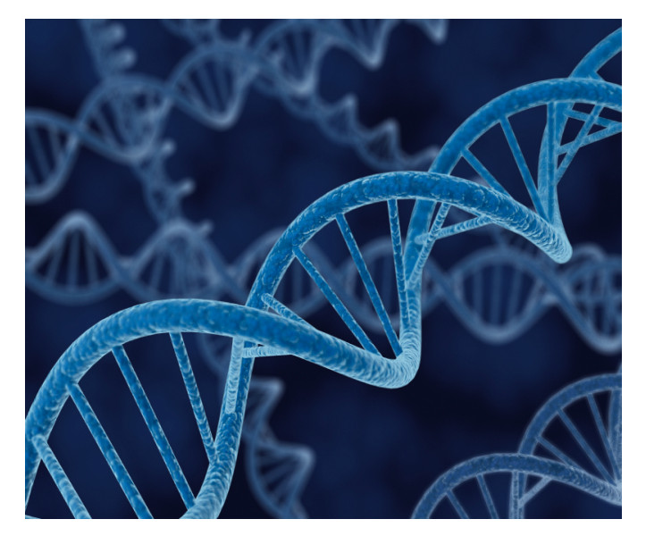
Goals

- A **benchmark** is an application on the simulator that is used to model how real computer applications would interact with shared data
- We would like benchmarks to utilize large shared data structures.
- The work done on the data structures should overlap with other cores, creating possible conflicts that test the system

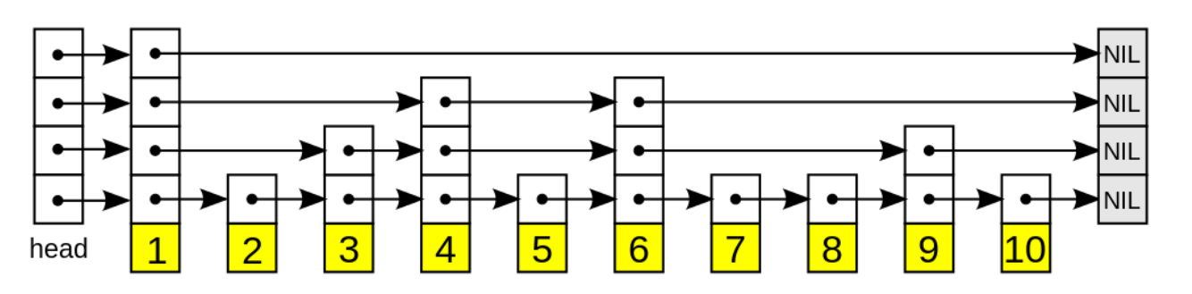
Benchmarks

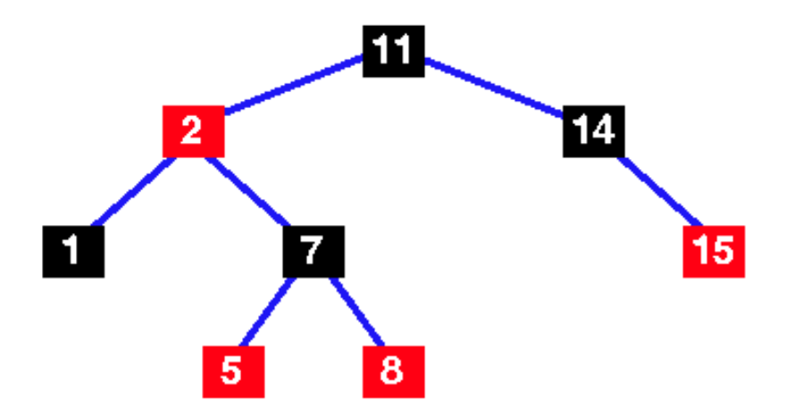


- **Patricia -** Inserts small number of IP addresses into patricia trie (radix tree) and then searches for addressed from larger list.
  - Shared work involves cores adding nodes to the trie
- **STAMP Suite -** Developed at Stanford
  - **Vacation -** Checks clients into vacation reservation system.
  - **Genome -** Creates unique list of DNA segments from large dataset and attempts to construct a complete genome.
  - **K-means -** Carries out k-means clustering on a set of floating point vectors
  - **Labyrinth -** Finds a path through a maze

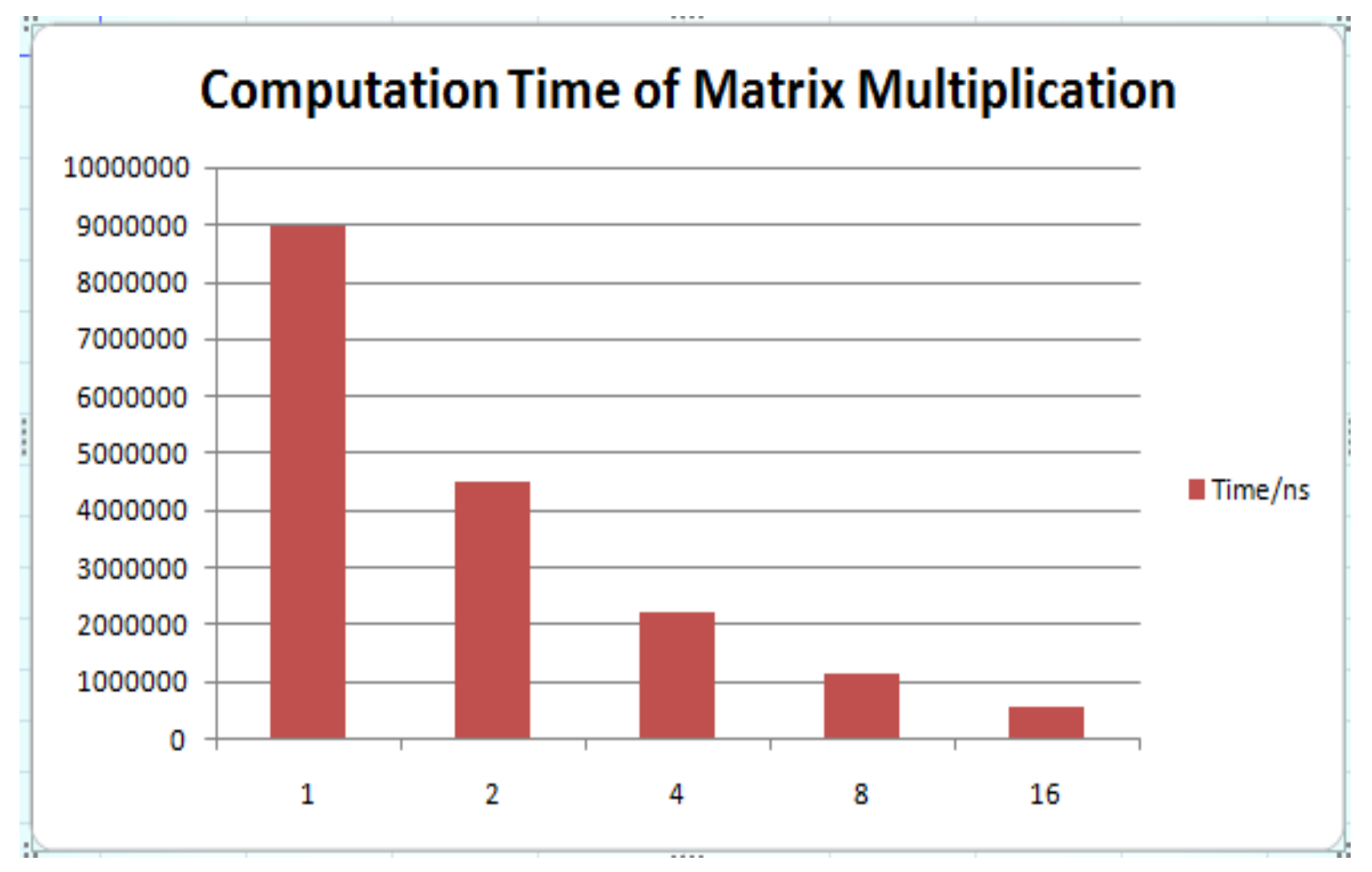


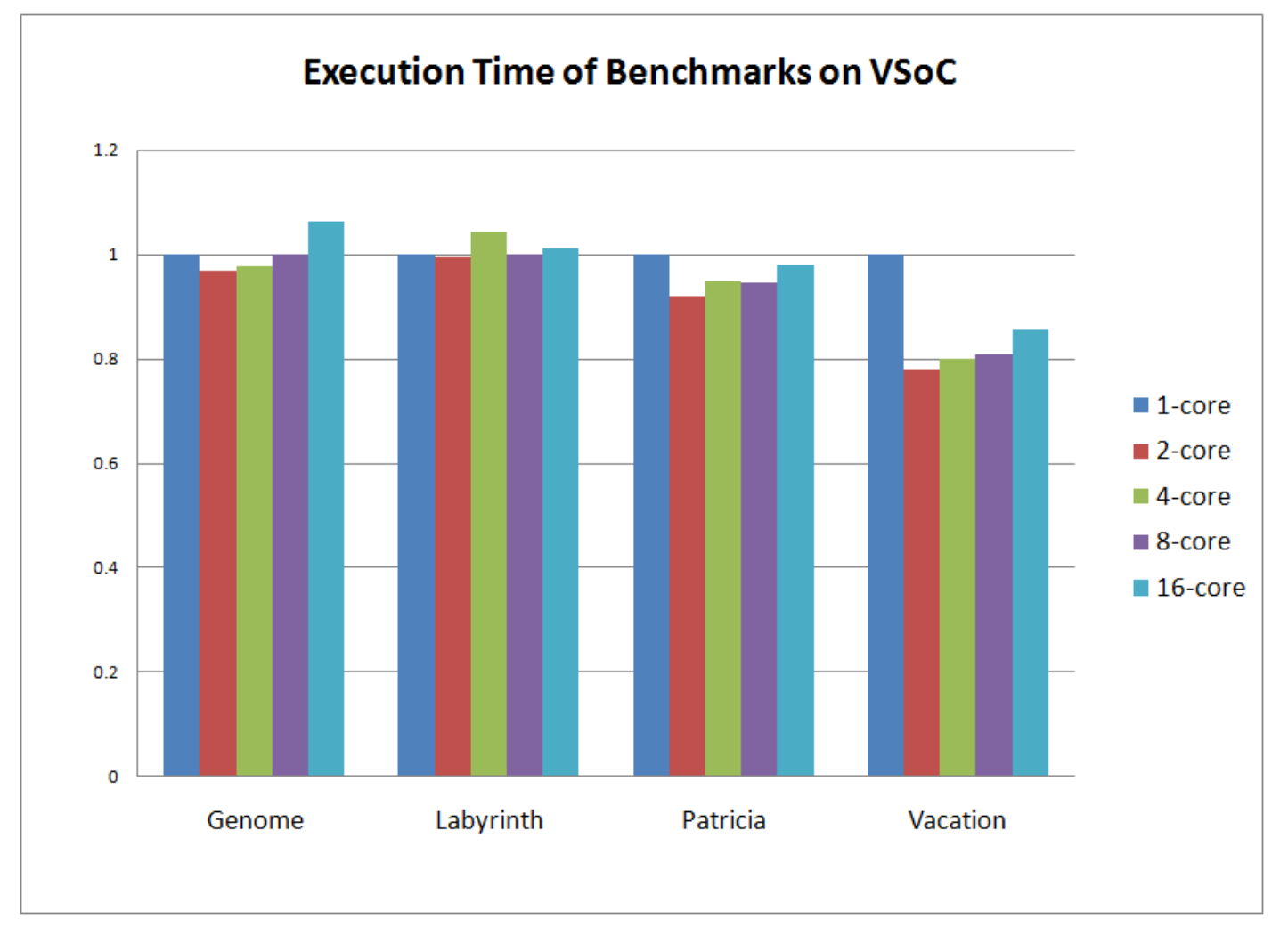
- **Red-Black / Skiplist -** inserts random memory addresses into private and local data structures





Results

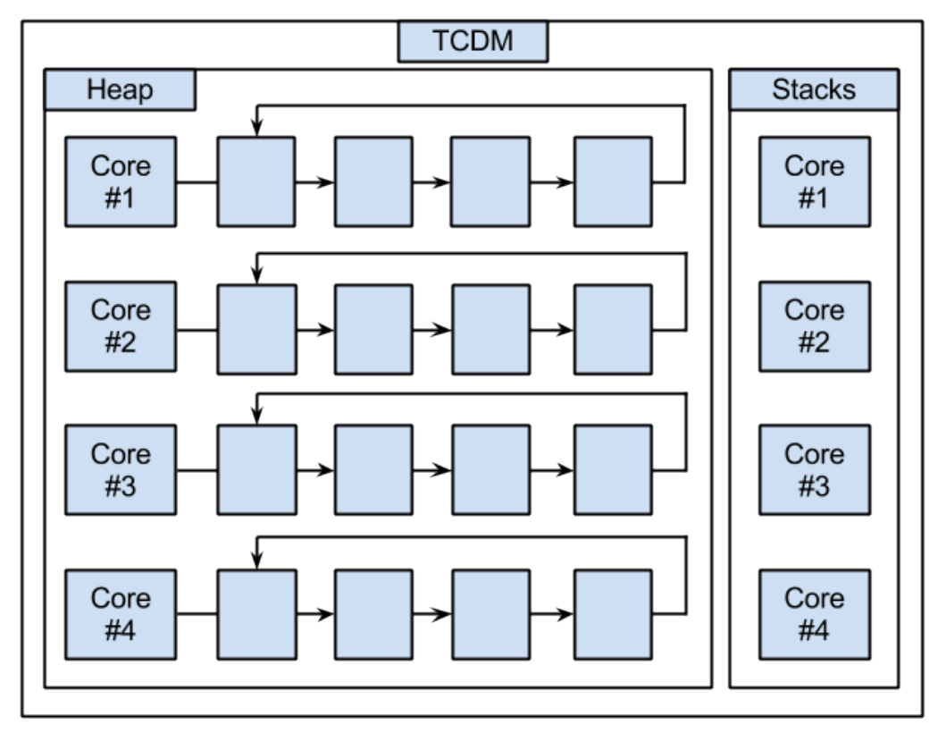




Application Support

- We expanded the interface between the applications and system that could support parallel benchmarks
- We added the MPARM supporting functions to VSoC for initialization flags, barriers, and global pointers

Shared Memory Allocation (Shmalloc)



- We created our own system tailored for VSoC that can allocate and free shared memory
- Relies on several linked links that connect free sections of memory
- Contains counters that keep track of how many free sections there are of different memory sizes

Initialization Flags

- In the initialization phase of most benchmarks, global data structures need to be allocated by one core only
  - The other cores will watch a flag in memory to be set by the first core

Barriers

- Barriers are checkpoints that all the cores must reach before any of them can move along with the rest of the benchmark
- Implemented in memory as a small counter

Global Pointers

- Once the global data structures are allocated, each of the other cores accesses a special location in memory that is initialized by the memory allocation system

Future Work

- Complete transactional memory support for the VSoC simulator
- Optimize shared memory allocation
- Run entire benchmark suite with transactional memory support and evaluate the system's performance and energy efficiency

Selected References

- Herlihy, Maurice, and J. Eliot B. Moss. Transactional memory: Architectural support for lock-free data structures. Vol. 21. No. 2. ACM, 1993.
- Holla, Aditya G., and Maurice Herlihy. "Lock Elision for Memcached: Power and Performance analysis on an Embedded Platform."
- Ferri, Cesare, et al. "Embedded-TM: Energy and Complexity-Effective Hardware Transactional Memory for Embedded Multicore Systems 6."
- Ferri, Cesare, et al. "SoC-TM: integrated HW/SW support for transactional memory programming on embedded MPSoCs." Proceedings of the seventh IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis. ACM, 2011.