Implementation of a Correlation Algorithm on the Cyclops-64 Architecture

Jeremy Pedersen, Jürgen Ributza, Guang R. Gao

Computer Architecture and Parallel Systems Laboratory

Problem

The tools in use by radio astronomers are beginning to change. Whereas individual large dishes were the preferred tools for much of the last century, a shift towards the use of large arrays of much smaller, omnidirectional antennas is now underway. These arrays are more prone to radio frequency interference than highly directional dishes, and a great deal of signal processing must be done to pull useful signals from the noise. The algorithm used is called correlation, and it is a very I/O intensive algorithm, requiring one floating point operation per byte of data to be correlated. Furthermore, correlation needs to be performed in real time. This requires massive parallel computer hardware with enormous bandwidth both on-chip and to main memory.

Past Approaches

The correlation algorithm has been implemented, and its performance measured, on a variety of architectures [1]. A graph of performance for several architectures relative to theoretical peak performance is shown below. As a result, future work will focus on two approaches to fiding data.

Why Cyclops-64?

As can be seen, the architectures whose performance was closest to their theoretical peaks were the Cell/B.E. and BG/P. These architectures both provide a lot of cache and have relatively high on-chip bandwidth and bandwidth to main memory. Below is a comparison of C-64 to the Cell/B.E. and BG/P. These architectures both provide a lot of cache and have relatively high on-chip bandwidth and bandwidth to main memory. Below is a comparison of C-64 to the Cell/B.E. and BG/P. These architectures both provide a lot of cache and have relatively high on-chip bandwidth and bandwidth to main memory. Below is a comparison of C-64 to the Cell/B.E. and BG/P. These architectures both provide a lot of cache and have relatively high on-chip bandwidth and bandwidth to main memory.

C-64: Overview

Cyclops-64 is a very unique architecture. Unlike a GPGPU, it provides sophisticated hardware for synchronizing threads, and allows all of its cores (called Thread Units or TUs) to access all locations in on and off-chip memory via a non-uniform global address space. Unlike Cell/B.E. or BG/P, it uses chip store real-estate that would otherwise be used for complex caches and large CSC cores to provide 160 simple in-order RISC cores (the TU) paired with 5-MB of completely user-managed SRAM. Below is a diagram of the basic layout of C-64.

Correlation on C-64

Two implementations of the correlation algorithm were written for Cyclops-64. The first ran in SPMD (Single Process Multiple Data) mode, and the second ran in SPSD (Single Process Single Data) mode. In SPSD mode, all TUs run the exact same code, and threads are created automatically. In SPMD mode, threads are created by the user using the TnT Threads (TNT) library, and need not all be the same. The SPSD code was a naive implementation, providing a baseline against which the performance of optimized implementations like the SPMD code could be judged.

SPMD vs. SPSD

Below are a few bullet points illustrating the major differences between the SPMD and SPSD implementations:

- **SPMD**
  - Station data is loaded and stored directly to DRAM.
  - All threads are identical at the instruction level.
  - **SPSD**
  - Station data is loaded into a ring buffer in SRAM before use.
  - 8 threads are "producers," moving data from DRAM to SRAM.
  - 1–150 threads are "consumers," correlating data from SRAM and storing it back to DRAM.

Results

The SPMD code was expected to outperform the SPSD code in all cases, as its producer/consumer model overlaps memory operations with computation, and allowed consumer threads to fetch their data from the much faster SRAM (20 cycles to access, as opposed to 59 for DRAM). Although these were several cases in which the SPMD version had higher or equivalent performance to the SPSD version, the SPSD version was generally faster. The average speedup over the SPSD version was about 3x. See the graphs of wall time (on the C-64 simulator) and speedup in the next column.

Future Work

Although a decent speedup was achieved simply by overlapping memory operations with computation, it is expected that a much greater improvement in performance is possible by reducing the overall number of loads performed. As a result, future work will focus on two approaches to finding data.

- Tiling in the Scratchpad memory.
- Tiling in registers.

Cyclops-64: Major Features

Cyclops-64 has some interesting features which make it especially suitable for a task like correlation, in which increasing arithmetic intensity through data reuse (locality) is key to improving performance:

- Scratchpad memory: each TU has access to a small local memory called the scratchpad (SP). It can be accessed without going through the crossbar to do so if accessing an SP other than its own. The default size of each SP is 15 KB.
- Large global SRAM: 2400 KB (not counting the assumed 15 K SPs).
- High bandwidth to global SRAM through the crossbar: 640 GB/s.
- Large number of registers for each TU: 64 registers per TU, of which one is a stack pointer, one is a global pointer, one is an interrupt register, and one is a permanent zero (this leaves 60 for use by the program).
- Hardware support for threads: hardware barriers, large set of atomic operations.
- Optimized load/store instructions: LDM/STM for more efficient crossbar utilization.

References


Acknowledgements

I would like to thank Robert Pavel, Joshua Landwehr, and Aaron Landwehr for their help in debugging early versions of my SPMD code, and for offering useful criticisms of the code. I also thank the Undergraduate Research Program at the University of Delaware for funding me this summer and giving me an opportunity to try my hand at research.

Additional information

My e-mail address: jpedersen@udel.edu The CAPSL Website: http://www.capsl.udel.edu/