Leveraging Multicomputer Frameworks for Use in Multi-Core Processors

Yael Steinsaltz, Ph.D., Scott Geaghan, Myra Jean Prelle, Ph.D., Brian Bouzas
Mercury Computer Systems, Inc.
ysteinsa@mc.com, sgeaghan@mc.com, mjp@mc.com, bbouzas@mc.com

Abstract

Power and size limitations are two major roadblocks to continued speedup of single core microprocessors. To address these problems, most of the large chip producers have moved to multi-core processors. Those vary in architecture from 2 to 4 equivalent processors sharing many resources in a single die (Intel Xeon, Freescale 8641 and others) to a master-slave architecture where one processor manages the computations, and several others are math co-processors (IBM Cell Broadband Engine™ (BE) processor). These architectures are in fact, multicomputers on a single chip.

Mercury Computer Systems has been aggressively developing software for the IBM Cell processor for more than a year. There are some architectural features of Cell that make it challenging to program a single Cell chip with a standard API such as MPI. This fact pushed us to invent a new API tailored for architectures like Cell. However, we did not invent from scratch. We discovered that we could base our paradigm on a standard presented at multiple past HPEC conferences – www.data-re.org. We call the new API Multi-Core Framework (MCF). The purpose of this paper is to introduce this framework and why we found it necessary to invent MCF.

This paper makes multiple references to Mercury's prior implementation of www.data-re.org, a product named Parallel Application System (PAS). We borrow some of the techniques used for interconnectivity of multi-computers and apply them to these new multi-core architectures. However, we are not making a product pitch here – we are instead setting the groundwork for justifying deviating from established standards at the leading edge of technology – and potentially setting the basis for new standards that will evolve in the years that follow.

The framework we are using manages the data flow in a manager-worker fashion, and is most efficient for master-slave architecture, but can be implemented in any of these architectures, as long as one of the cores acts as a manager, it is worth reminding readers that the more established APIs will run in those chips as well.

We show, using an intercept receiver example, how leveraging past techniques enables us to achieve high efficiency, while doing minimal work when porting legacy code. Performance results for this application are shown for PowerPC (PPC using PAS), field programmable gate array ([FPGA]), manual optimization) and Cell technology-based hardware implementations.

MCF Features

Mercury's MCF provides a software Application Programming Interface (API) that is specifically designed for image and signal processing applications that are to be executed on heterogeneous or homogeneous multi-core architectures. It is based on the same principles that were used to build the PAS software, which is a commercial implementation of the data re-organization mechanism (many to many, N dimensions, see: www.data-re.org). Computationally demanding applications require a software development environment that supports high-performance, ease of use, and efficient processor, memory, and interconnection resource management. In the design of MCF, these goals were considered paramount.

MCF consists of a library of functions for managing concurrent processes and performing distributed computation. A heterogeneous multi-core architecture like the Cell BE processor consists of a general-purpose processor and a collection of math co-processors with their own small memories. This kind of architecture naturally lends itself to an environment where the general-purpose processor manages the small worker processors, which perform computations and move data. We call this the *Function Offload Engine Model*. However, MCF, like PAS, does *not* provide a model where the application developer writes a single program for the manager and tasks are automatically assigned to workers, although MCF can be a mechanism to enable that kind of model. It is the application developer's responsibility to provide a program for the manager (the general-purpose processor) and one or more programs for the tasks the workers (the math co-processors) must perform.

Data Movement

MCF tile channels provide multi-buffered, strip mining of N-dimensional data sets between a large main memory (XDR memory in the Cell BE processor) and the small worker memories, *local store*. In a heterogeneous multi-core architecture, the manager's main memory is probably large compared to the workers' local stores. In order for the workers to perform their tasks, they need to move data from the main memory into their own local memories. If we think of their local memories as caches, then the activity they need to perform is similar to strip mining from main

memory into a processor's cache. The MCF tile channels provide means for the manager program to define now the strip-mined pieces (i.e. tiles), these are to be assigned to workers – for example, bands of columns, bands of rows, bands of planes, round robin, or all tiles to every worker. More complicated tile assignments are also possible using functions that are modifying fields in the distribution object (e.g. tile and/or assignment overlap).

Signal Intercept Receiver Example

The original intercept receiver demo was implemented entirely with Power PC processors four years ago. It consists of two main components commonly found in the front-end of an intercept receiver: a channelizer, and a high-speed alarm (HSA). The real valued output from an analog-to-digital converter (ADC) feeds into the channelizer. The output of the channelizer feeds into the HSA. The HSA is designed to detect frequency hopping signals. Operation of the intercept receiver is demonstrated with an MSK modulated signal hopping at a rate of 250 hops per second. The hopping signal, called the Future Multi-Band, Multi-Waveform, Modular, Tactical Radio (FM3TR) waveform, was developed for the Software Defined Radio (SDR) Forum.

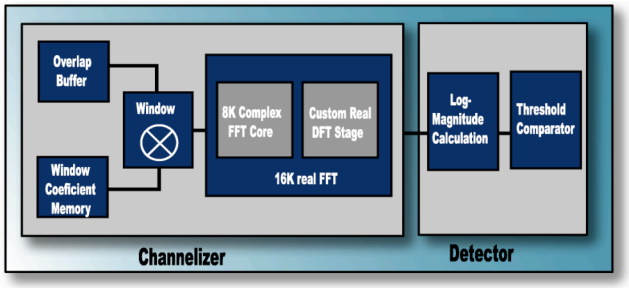

Figure 1: Application design.

The application is shown in Figure 1. The channelization is done using 16K real FFT with 75% overlap of the input. For the sake of simplicity, we use a simple threshold for each of the channels in the high-speed alarm. Still, the amount of processing required for the HSA is data dependent. An environment with many frequency hopping signals will require more processors than an environment with only one frequency hopping signal. The FPGA implementation, conducted a year and a half ago and presented at HPEC2005, uses the exact same algorithm, at the same speeds, as does the porting we are now doing onto the Cell BE processor-based blade.

This example requires the resources of a quarter of a dual VirtexIIPro70 FPGA board (MCJ6 FCN) and one 500MHz 7410 PowerPC (used for the clustering part of the hop detection) for an input sample rate of 80Msps. By comparison, the 500MHz 7410-based demo requires the resources of 20 PowerPCs (5 MCJ6 boards) for the channelizer, and 7 PowerPCs (1.75 MCJ6 boards) for the HSA. Using the Cell BE processor (data flow shown in Figure 2), a single one-processor blade can perform this algorithm, maintaining the same processing that was done in the multicomputer environment. The time estimated to port the application to the Cell BE processor, translating the PAS calls to MCF, and keeping the SAL (Scientific Algorithm Library) calls, is about two weeks, while porting to the FCN took about eight man months.

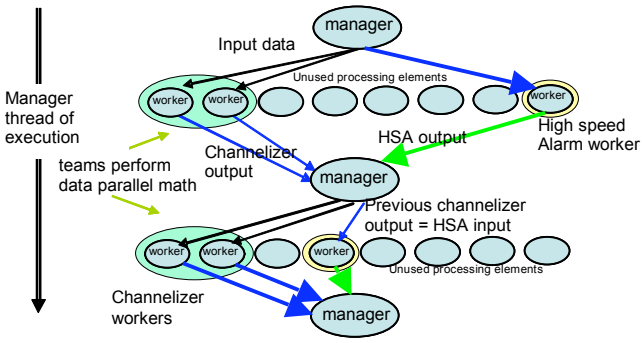

Figure 2: Data flow inside the cell BE processor

Summary

We are showing here that using multi-core processors with a framework that resembles an existing multiprocessor (distributed memory) one, lets us enjoy the best of all worlds: condensed processing environment (hardware), easy porting of legacy code, and the flexibility of general purpose processors. The resulting implementation achieves hardware savings comparable to those generated when porting to an FPGA, with a much easier legacy code porting effort.

These accomplishments are available because multi-core hardware, like the Cell technology, supports similar infrastructure as the multicomputer environment that preceded it, like PAS -> MCF, and the availability of the scientific algorithm library.

References

[1] Brian Bouzas, Jon Greene, Michael Pepe, Myra Jean Prelle, Ph.D., "MCF: Multi-Core Framework for Signal Processing Applications," Mercury Computer Systems, May 2005.

[2] Scott Geaghan, "Channel Recombination for Intercept Receivers," Mercury Computer Systems, 8/17/2005.

[3] Scott Geaghan, Sarah Leeper, "Intercept Receiver on FCN," Mercury Computer Systems, 6/7/2005.

[4] Sarah Leeper, Robert Frisch, Scott Geaghan, Scott Tetreault, Mike Vinskus, Erich Whitney, "FPGA based signal acquisition system," HPEC conference 2005.

[5] Brian Bouzas, Robert Cooper, Jon Greene, Michael Pepe, Myra Jean Prelle, Ph.D., "MultiCore Framework: API for Programming Heterogeneous Multicore Processors," First Workshop on Software Tools for MultiCore Sytems (STMCS), March 2006.