Performance Benchmarks and Programmability of the IBM/Sony/Toshiba Cell Broadband Engine Processor

Luke Cico, Robert Cooper, Jon Greene, Michael Pepe
Mercury Computer Systems, Inc.
lcico@mc.com, rcooper@mc.com, jgreene@mc.com, mpepe@mc.com

Abstract

Cell Broadband Engine™ (CBE) processor-based systems have been available in various prototype forms for at least a year. This has given early adopter users the ability to implement their algorithms and assess the performance and programmability of the technology. In this paper we present application-specific benchmarks for intensive algorithms targeting the Cell processor including measurements on real hardware.

The mathematical primitives explored in the benchmarking portion of this paper include fast Fourier transforms (FFTs), Fast Convolution, matrix multiplication, matrix factorization and image filters. This sampling of key algorithm primitives represents significant components in a number of applications of relevance to the HPEC community.

We will also present a programming framework developed by Mercury Computer Systems that is being used by early adopters to maximize the productivity of applications targeting the CBE.

An example will be provided to 'pull together' the ideas presented in the programming framework and how it can be utilized to achieve high levels of sustained performance and high productivity. The high-level design process for implementing a complex algorithm on the CBE is to:

1. decompose the mathematical problem and map it to the node's architecture,
2. describe the data and processing flow in terms of a programming model,
3. implement the model using the applicable software tools.

In this example we will decompose the mathematics of a radar adaptive nulling processor to the Cell processor and describe the data flow and computational mapping within the framework of the Multi-Core Framework (MCF) programming model. Also presented will be a solution of the mapping of this 3D problem using the MCF software tools (Application Programming Interface [API] and library). The example will include performance of the algorithm primitives required by the application.

Algorithm Performance

This paper presents sustained performance and performance/watt benchmark results on CBE hardware for various processing primitives:

- 1D and 2D complex Fast Fourier Transforms for N=64 to 1M samples
- 1D and 2D complex Fast Convolutions for N=64 to 1M samples
- Complex matrix multiplication
- Matrix Factorization (QR)
- Image filters

The benchmark results are showing at least 10X-improvement in performance per chip relative to current PowerPC/AltiVec™ technology-based solutions. Furthermore, the results show improvements in sustained performance per watt by 2-3X relative to the current PPC/AltiVec technology.

Some preliminary results for 1D FFTs are shown in Table 1. These have been measured on 2.4 GHz and 3.0 GHz Cell processor systems and scaled up to 3.2 GHz to reflect the performance of production hardware. The Freescale 7448 numbers are from a Mercury-optimized Scientific Algorithm Library (SAL) implementation while the other results are taken from the FFTW benchmarking website (www.fftw.org/benchfft/).

Table 1. FFT Performance (GFLOPS)

Processor	1K	8K	64K
Cell BE 3.2 GHz	170.7	176.8	90.8
IBM 970 (G5) 2 GHz	9.1	5.3	3.2
Pentium 4 Xeon 3.6 GHz 2 MB L2	8.0	6.3	6.1
Opteron Model 275 32bit 2.4GHz	4.7	3.5	3.0
FreeScale7448 975 MHz	4.0	2.6	3.0

The MultiCore Framework Programming Toolkit

This paper also presents a software toolkit for programming heterogeneous multi-core processors, such as the CBE processor, that contain explicit non-cached memory hierarchies. Emerging heterogeneous multicores contain specialized processing elements that do not possess traditional cache hardware. Instead, multiple levels of the memory hierarchy must be explicitly managed by software. The MCF toolkit provides an abstract view of this hardware oriented toward computation of *n*-dimensional matrix data sets. High performance and ease of programming are the primary goals. Through the toolkit, the programmer has explicit control over how data and processing is divided up among processing and memory elements, while being insulated from specific hardware details. Further, MCF separates data organization from computation. This enables the problem size and resource allocation to be changed without touching highly optimized inner loop code.

MCF facilitates multi-buffered strip mining of data between a large memory (XDR memory on the Cell) and small worker memories (SPE local store on the Cell) in such a way as to insulate worker code from the details of the data organization in main memory.

Conclusions

This paper presents benchmark results to date for computationally intensive algorithms on the STI CBE. The results indicate that the device should deliver exceptional improvements in throughput and latency performance relative to currently fielded processors. Furthermore, improvements in the delivered performance per watt by several factors relative to the current generation of high-performance, low-power embedded RISC+DSP chips are likely.

This paper also presents a programming model and MCF software toolkit that greatly eases the distributed programming tasks of the designers/developers and works through a larger example to illustrate the design process for producing high-performance Cell applications.