Scalable SAR with Sourcery VSIPL++ for the Cell/B.E.

Benchmarking open-architecture programming approaches to multi-core architectures.

J. Bergmann, M. LeBlanc, D. McCoy, B. Moses, S. Seefeld
CodeSourcery, Inc.
jules@codesourcery.com

Introduction

Scalable SAR is part of the HPEC Challenge Benchmark Suite [5][6][7]. It is a synthetic application that accurately models the computation performed in Synthetic Aperture Radar (SAR) image formation [6]. It is designed to be representative of the types of embedded processing that occur in aerospace, medical, and reconnaissance processing.

Sourcery VSIPL++ for the Cell/B.E. [1][2] is an implementation of the open standard VSIPL++ signal and image-processing API [1] on the IBM Cell/B.E. multi-core processor architecture [4]. It is suitable for implementing high-performance signal-processing applications that take full advantage of the Cell/B.E. processor throughput, without sacrificing programmer productivity or application portability.

This paper presents an implementation of the Scalable SAR synthetic application in VSIPL++. This implementation will allow productivity, performance, and portability comparisons to be made, illustrating the potential benefits of Sourcery VSIPL++ as an open-architecture programming approach to complex multi-core architectures such as the Cell/B.E.

Scalable SAR

Scalable SAR consists of a scalable data generator, four separate computational and I/O kernels, and a validation component. For this comparison, the kernel performing SAR image formation was implemented in VSIPL++. The kernel performs fast-time filter, bandwidth expansion, matched filtering, interpolation, and 2D FFT steps (Figure 1). These steps use a range of signal processing functions, including 1D FFT, interpolation, and element-wise operations. In between several of the steps, data must be rearranged in memory through a “corner-turn” for more efficient processing.

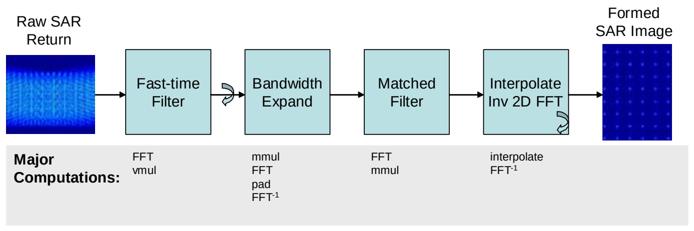

Figure 1: Processing steps in SAR image formation.

Sourcery VSIPL++ for Cell/B.E.

The Cell/B.E. is an asymmetric, multi-core processor architecture developed by IBM, Sony, and Toshiba. It is described as “supercomputer on a chip” capable of over 200 peak single-precision GFLOP/s on a single chip with 9 cores. For more detailed descriptions of its architecture, refer to [4].

Numerous authors have commented on the challenge of programming the Cell effectively. Sacco [10] lists nearly a dozen concerns brought about by architectural features that a programmer must consider when trying to program the hardware directly at a low-level. Addressing these concerns is reported to increase code size by a factor of 16.

Sourcery VSIPL++ for the Cell/B.E. simplifies Cell/B.E. programming by presenting a simple programming model that can make optimal utilization of the Cell/B.E.’s capabilities. The PPE control processor runs the application while the SPE accelerators are used as high-performance computation engines. IBM’s Acceleration Library Framework (ALF) [9] manages the SPEs, handling initialization and double-buffered data transfer to hide communication latency behind computation.

Computations which can be mapped to the SPEs are recognized by Sourcery VSIPL++’s dispatch engine. Compile-time and run-time attributes control how SPEs are allocated for a computation. A variety of factors are considered, including the data layout, the operation being performed, and the ratio of computation to communication. Application attributes can also be used to tune the SPE allocation.

Existing VSIPL++ applications can take advantage of the Cell/B.E. simply by recompiling for the new target. Performance can be tuned by modifying data structure attributes to influence resource allocation, and by using fused operations to create locality and optimization potential.

In previous work [1], a high performance VSIPL++ implementation of fast convolution was demonstrated, which achieved 40% of peak performance across Cell/B.E., Xeon, and multiple PowerPC architectures without requiring system or architecture specific code.

Results Preview

The following results will be presented at HPEC 2008.

First, productivity results will compare the number of lines of code required to implement the synthetic SAR computation kernels in VSIPL++ with other available implementations, including the reference Matlab implementation. Our experience in previous comparisons of this type is that VSIPL++ applications have approximately 60% fewer lines of code than comparable C-VSIPL/MPI applications.

Second, performance results will present the computation throughput of the VSIPL++ synthetic SAR application. These measurements will be compared against the theoretical peak for the architecture, the algorithmic theoretical peak (architectural theoretical peak adjusted to represent limiting factors in the algorithm, such as computation/communication ratio, special instruction requirements, and so on), and the performance of other available implementations. Performance results will be presented for the Cell/B.E. architecture and other commodity architectures used in the DOD signal and image processing space. Our previous experience in benchmarking VSIPL++ applications is that they achieve performance that is close to the algorithmic limits, and comparable to that of applications written in C-VSIPL/MPI and vendor libraries.

Third, portability results will measure the amount of work necessary to port the VSIPL++ synthetic SAR application across multiple architectures. Portability will be measured using two metrics. First, the porting effort will be measured by the number of lines of code necessary to change to port the application to a new platform. Second, the porting quality will be measured by the performance achieved by the ported application. Our previous experience in porting VSIPL++ applications is that very few lines of code need to be modified, and that similar performance is achieved in terms of percentage of peak utilizations.

Conclusion

Sourcery VSIPL++ for the Cell/B.E. provides an open architecture approach for high-performance programming the Cell/B.E. Applications are written with a simple programming model, achieve high performance, and remain portable to other architectures. Implementing the HPEC Scalable SAR synthetic application in VSIPL++ enables these claims to be verified by comparisons with other Scalable SAR implementations.

References

[1] J. Bergmann, M. Mitchell, D. McCoy, S. Seefeld, A. Salama, F. Christensen, R. Pancoast, and T. Steck. “Sourcery VSIPL++ for the Cell/B.E.” *HPEC Workshop*, Lexington, MA, 2007.

[2] CodeSourcery, Inc. *VSIPL++ Specification 1.0.* Georgia Tech Res. Corp. 2005 [online] Available: http://www.hpec-si.org.

[3] CodeSourcery, Inc. Sourcery VSIPL++. [online] Available: http://www.codesourcery.com/vsiplplusplus.

[4] M. Gschwind, et al. “Synergistic Processing in Cell’s Multicore Architecture.” *IEEE Micro*, March 2006.

[5] R. Haney, J. Kepner, T. Meuse. “The HPEC Challenge Benchmark Suite.” *HPEC Workshop*, Lexington, MA, 2006.

[6] R. Haney, T. Meuse, J. Kepner, and J. Lebak. *The High Performance Embedded Computing (HPEC) Challenge Benchmark Suite.* MIT/LL 2005. [online] Available: http://www.ll.mit.edu/HPECChallenge.

[7] R. Haney, T. Meuse, J. Kepner, and J. Lebak. “The High Performance Embedded Computing (HPEC) Challenge Benchmark Suite.” *HPEC Workshop*, Lexington, MA, 2005.

[8] High Performance Embedded Computing Software Initiative. [online] Available http://www.hpec-si.org/.

[9] IBM Cell Broadband Engine Software Development Kit. [online] Available: http://www.alphaworks.ibm.com/tech/cellsw?open&S_TACT=105AGX16&S_CMP=DWPA

[10] S. Sacco, G. Shrader, and J. Kepner. “Exploring the Cell with HPEC Challenge Benchmarks.” *HPEC Workshop*, Lexington, MA, 2006.