Sparse Matrix-Vector Multiplication Kernel on a Reconfigurable Computer

Sreesa Akella
Department of Computer Science and Engineering,
University of South Carolina
Columbia, SC 29208
akella@engr.sc.edu

Melissa C. Smith, Richard T. Mills,
Sadaf R. Alam, Richard F. Barrett, Jeffrey S. Vetter
Oak Ridge National Laboratory
Oak Ridge, Tennessee, 37831-6173
{smithmc | rmills | alamsr | rbarrett | vetterjs}@ornl.gov

Abstract

The SRC reconfigurable computer provides the capability of obtaining application-specific driven performance for high data bandwidth, computationally intensive applications. It has high-density FPGA devices with local distributed memory banks that can be utilized to obtain high performance for floating point applications. The floating-point Sparse Matrix Vector (SpMatVec) multiplication, a key computation kernel in many scientific applications does not run at peak performance on general purpose microprocessors. The high I/O bandwidth and avoidance of cache-hierarchy architecture in this reconfigurable computer allow us to efficiently implement the floating-point SpMatVec kernel on the SRC platform. In this paper we investigate the implementation of a floating-point SpMatVec kernel on the SRC MAPstation and benchmark its performance against other general-purpose microprocessor-based implementations.

Introduction

The floating point SpMatVec multiplication kernel forms an integral part of several scientific and engineering applications such as iterative matrix equation solvers. The cache hierarchy based general purpose microprocessors exhibit low performance in implementing this kernel for two main reasons. Firstly, the irregular memory access patterns due to poor data locality of sparse matrices causes large number of cache misses. Secondly, the high ratio of load/store operations to floating- point operations result in low utilization of the floating-point units [1]. Several techniques and optimizations have been proposed to overcome these two problems but depend on the sparsity information of the matrices [2, 3].

Recent advances in FPGA logic and capacity now provide the means to effectively implement floating point applications. It has been shown that current FPGAs with abundant on-chip memory and I/O pin resources provide peak floating-point performance surpassing that of microprocessors [4]. In [5, 6] Prasanna et al. implemented floating-point cores and dense matrix-vector multiplication on FPGA devices and compared the performance with that of general purpose microprocessors.

Recently some amount of work has been conducted with SpMatVec operations on FPGAs. In [7] ElGindy et al. implemented fixed point SpMatVec multiplication on the PCI-Pamette FPGA-based system. They evaluate single, dual and triple constant-coefficient hardware multiplier systems that can be reconfigured at run-time through a host computer-based scheduler. In [8] Wang et al. implemented a parallel LU factorization of sparse matrices on the Altera Nios development board and the SOPC development board. Their work focuses on integration of floating point multiplication, addition and division units with the Nios soft IP RISC processor core and the implementation of a multiprocessor parallel machine on both the development boards. They evaluate the performance of these machines using the bordered diagonal block sparse matrix LU factorization application.

In contrast, there has been limited work in the field of floating-point SpMatVec multiplication using application-specific designs on FPGAs. In [9] the authors look at an implementation of an optimized SpMatVec multiplication kernel on a Virtex-II Pro FPGA device. They analyze and compare its performance in GFLOPS to that of an Itanium® 2 processor. In [10] a multi-FPGA based implementation is explored and its performance is compared to a general purpose microprocessor-based multiprocessor system. These authors show that FPGAs can be utilized to achieve high performance on SpMatVec multiplication kernel; however they do not deal with the real issues of implementing the kernel on a FPGA-based high performance computing system.

Our implementation of a floating-point SpMatVec multiplication kernel on the SRC MAPstation reconfigurable platform aims to provide actual performance numbers by including the issues of 1) FPGA configuration time and, 2) data communication time between the host processor, on which the main application resides, and the FPGA, on which the SpMatVec multiplication kernel resides. This approach provides a realistic view of the advantages and disadvantages of implementing scientific applications for which the floating point SpMatVec multiplication is an integral part, on a FPGA-based high performance computer.

The SRC MAPstation

The SRC MAPstation, by SRC Computers [11], removes the programmer from the details of underlying hardware architecture and allows one to focus on the function implementation. This approach reduces the time to solution by facilitating software development by programmers and mathematicians.

The SRC MAPstation architecture includes dual 2.8 GHz Pentium 4 Intel® microprocessors (µPs) running the Linux operating system. The reconfigurable logic resource, or MAP®; MAP® board normally comes in pairs attached to these µPs. Each MAP® as shown in Figure 1 consists of two Xilinx® Virtex XCV6000 FPGA chips running at 100 MHz, a control processor, and six 4 MB SRAM banks referred to as OnBoardMemory (OBM). Code for the µPs is written in

standard C or Fortran. Code for the MAP® hardware is also written in C or Fortran and compiled by an SRC-proprietary compiler that targets the MAP® components. Calls to execute on the MAP® are function/subroutine calls from the standard C or Fortran modules.

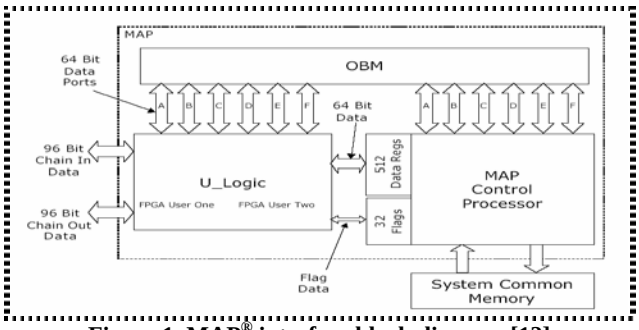

Figure 1. MAP® interface block diagram [12]

The SRC MAPstation C Map Implementation

The SpMatVec multiplication requires computing the matrix-vector product $y = Ax$. Where A is $m \times m$ sparse matrix with nz number of non-zero elements and x and y are vectors of length m.

The sparse matrix is represented in a compressed storage format that stores only the non-zero elements. The Compressed Row Storage (CRS) format is one of the simplest and most efficient formats that does not make assumptions on the sparsity structure of the matrix [13]. We employ this format for storing the matrix in the memory. The format represents the matrix as three vectors: NZ of length nz, for the non-zero floating-point elements; CO for the column indices of the non-zeros of length nz; and PT of length $m+1$ for the pointers to the non-zero elements in NZ that start a column. From the PT vector we generate an additional vector CL that stores the size of each column or the number of non-zeros in each column.

The basic architecture of the implementation is given in Figure 2. It consists of n parallel 64-bit floating point multiplier accumulators (MACs) working on n different matrix elements. Each MAC works on one row of the matrix to obtain one element of the output vector OV.

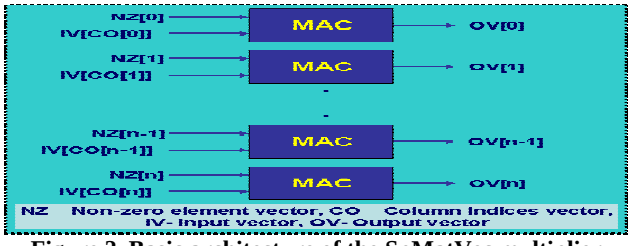

Figure 2. Basic architecture of the SpMatVec multiplier

The elements of each column are read in sets of n. Each column is thus partitioned into sub-columns of size n. Thus a column of size k would be partitioned to $\lceil k/n \rceil$ sub-columns. Each sub-column is read from the OBMs or the BRAMs and fed to the MACs in parallel. The MACs are iteratively fed all the first sub-columns of every column to produce the first n elements of the output vector OV. The indexing of the first sub-columns is done by using the PT array which stores the index to the first non-zero element of each column. The above step is repeated until all sub-columns of the every column are processed to obtain all elements of the output vector OV. The matrix, the input and output vectors are stored in the available six OBMs on the MAP and on-chip BRAMs in the FPGA.

Performance Analysis

In our evaluation of floating-point SpMatVec on the SRC reconfigurable computing architecture, we analyze the performance of the implementation and evaluate its MFLOPS performance. We benchmark these results against other implementations on traditional microprocessor-based systems. These implementation results are applicable to the more general use of SpMatVec kernels in the implementation of iterative solvers for matrix equations as used in scientific and engineering applications. Future plans are to scale this implementation to multi-FPGA and multiprocessor systems such as the Cray XD1 [14].

References

[1] S. Toledo, "Improving Memory-System Performance of Sparse Matrix-Vector Multiplication," IBM Journal of Research and Development, 41(6), pg 711-725, 1997.

[2] E. J. Im, K.A. Yelick, R. Vuduc, "SPARSITY: Optimization framework for sparse matrix kernels," International Journal of High-Performance Computing Applications, 18(1) pg 135-58, 2004.

[3] W. D. Gropp, D.K. Kaushik et al., "Performance Modeling and Tuning of an Unstructured Mesh CFD Application," Proceedings of SC2000: High Performance Networking and Computing Conference. (electronic publication), 2000.

[4] K. D. Underwood, and K. S. Hemmert, "Closing the GAP: CPU and FPGA Trends in Sustainable Floating-Point BLAS Performance," In Proceedings of 2004 IEEE Symposium on Field-Programmable Custom Computing Machines, 2004.

[5] G. Govindu, et al., "Analysis of High-Performance Floating-Point Arithmetic on FPGAs," In Proceedings of the 11th Reconfigurable Architectures Workshop, 2004.

[6] L. Zhuo, V. K. Prasanna, "Scalable and Modular Algorithms for Floating-Point Matrix Multiplication on FPGAs," In Proceedings of the 18th International Parallel & Distributed Processing Symposium, 2004.

[7] H. A. ElGindy, Y. L. Shue, "On Sparse Matrix-Vector Multiplication with FPGA-based System," In Proceedings of the 10th IEEE Symposium on Field-Programmable Custom Computing Machines, 2002.

[8] X. Wang, S. G. Ziavras, "Performance Optimization of an FPGA-Based Configurable Multiprocessor for Matrix Operations," In Proceedings of IEEE International Conference on Field Programmable Technology, 2003.

[9] L. Zhuo, V. K. Prasanna, "Sparse Matrix-Vector Multiplication on FPGAs," In Proceedings of the 13th International Symposium on Field Programmable Gate Arrays, 2005.

[10] M. deLorimier, A. DeHon, "Floating-point Sparse Matrix-Vector Multiply for FPGAs," In Proceedings of the 13th International Symposium on Field Programmable Gate Arrays, 2005.

[11] SRC Computers, Inc., www.srccomp.com.

[12] SRC MAP Hardware guide, Version 1.9, SRC Computers, Inc.

[13] R. Barrett, et al., "Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods", 2nd edition, SIAM, 1994.

[14] Cray, Inc., www.cray.com