On Sparse Matrix-vector Multiplication with FPGA-based System

Hossam ElGindy and Yen-Liang Shue
School of Computer Science and Engineering
University of New South Wales
Sydney, NSW 2052, Australia

Abstract

*In this paper we report on our experimentation with the use of FPGA-based system to solve the irregular computation problem of evaluating $y = \mathbf{A}x$ when the matrix $\mathbf{A}$ is sparse. The main features of our matrix-vector multiplication algorithm are (*i*) an organization of the operations to suit the FPGA-based system ability in processing a stream of data, and (*ii*) the use of distributed arithmetic technique together with an efficient scheduling heuristic to exploit the inherent parallelism in the matrix-vector multiplication problem. The performance of our algorithm has been evaluated with an implementation on the* Pamette *FPGA-based system.*

1. Introduction

Many practical systems have been modelled by large systems of linear equations, and in many cases such systems are sparse. Ef cient parallel algorithms, which takes advantage of sparsity, for manipulating such matrices have been presented in [1, and references within] for the EREW-PRAM model of computation. Schmeck et al [2] presented constant time algorithms for multiplying different types of weakly sparse matrices for the recon gurable mesh architecture which is an abstract model of computation that permits the change of functionality of each node and the interconnection between nodes at the beginning of each clock cycle as basic operations. Practical recon gurable systems (e.g., FPGA-based systems) provide the same capability as the abstract model, but with a varying cost for achieving the different recon guration types. In this paper we report on our work-in-progress into the use of FPGA-based systems to manipulate irregular structures like sparse matrices. In section 2 we describe the system to be used for implementing our algorithm which is then described in section 3. The performance of our implementation is presented in section 4. Finally we conclude with a discussion of our experiment and directions for future research.

2. Hardware Platform

A constant-coef cient multiplier [4], which is the heart of our solution for evaluating $y = \mathbf{A}x$ when the matrix $\mathbf{A}$ is sparse, is a hardware-effective method for performing multiplication when one of the multiplicands is a constant compared to general multiplier designs. However the penalty of a long delay time to reload multipliers with different constants into an FPGA device via a con guration bit stream, even partially recon gurable devices, would far outweigh any bene ts in its ef cient use of the hardware. Fortunately such devices use Look-Up Tables (LUT) to enable the implementation of combinatorial logic functions. A LUT can also be used as a RAM device whose content may be changed with very little time penalty. This type of devices can thus be used to achieve runtime logic recon guration, with no interconnect recon guration, with reasonable time delays which is ideal for implementing algorithms that require re-loading of constant-coef cient multipliers. In this scheme one multiplicand is used to con gure the multiplier (equivalently, it is used to store appropriate values into the LUTs), and the other multiplicand is used to address the LUTs' inputs to produce the desired product [3]. We use the PCI-Pamette, a PCI board developed by Compaq Custom Systems department, to assess the performance of our algorithms.

3. The Algorithm

In this section we present a new approach for evaluating $y = \mathbf{A}x$, when the matrix $\mathbf{A}$ is sparse, for the Pamette system described earlier. Our algorithm assumes that the matrix $\mathbf{A}$ is given in a column-major representation along with the number of non-zero entries in each column. The main features of the algorithm are:

(i) The use of a self-con guring constant-coef cient multiplier [3] which can be ef ciently implemented on the available FPGA device. Therefore we can implement a number

Proceedings of the 10 th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02)
1082-3409/02 $17.00 © 2002 **IEEE**

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: National Univ of Defense Tech. Downloaded on December 07,2020 at 13:49:37 UTC from IEEE Xplore. Restrictions apply.

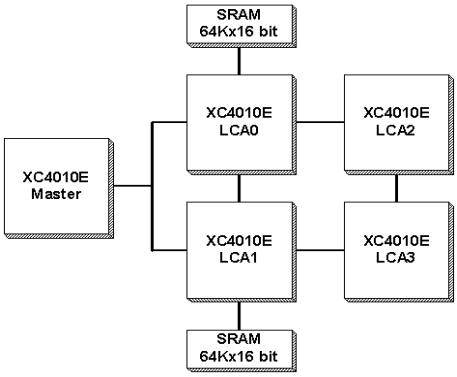

Figure 1. Pamette Board Layout

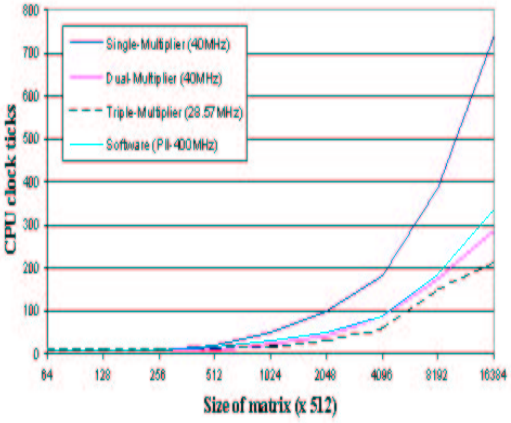

Figure 2. The core multiplier performances of the various systems.

of such multipliers in the available FPGA devices and use the parallelism inherent in the problem.

(ii) The use of a computationally simple scheme to schedule the recon guration and use of constant-coef cient multipliers. The problem is formulated as a bin-packing problem with the number of bins equivalent to the number of available multipliers. The NP-hard nature of this problem is not a major obstacle in our situation. For this paper we have used the pseudo-polynomial partition algorithm to compute an exact optimal solution to schedule operation for a dual-multiplier system. This component of the our algorithm is executed on the host computer. For triple-multiplier system, we have used brute-force search to compute the optimal schedule. A complete description of the procedure is detailed in [5].

(iii) The use of Pamette memory modules to partially overcome delays imposed by the communication subsystem between the host and the Pamette board. We use one SRAM module to store the multiplicands and the second SRAM module to store the results of multiplications. This method allows for the computations on the Pamette board to proceed without the need to communicate with the host over the PCI-bus.

4. Performance Results

Various results were taken to measure the times required for different components of the system. In this paper we focus on the core multiplication component which refers to the time taken within the Pamette unit to perform the multiplication using data already loaded into the SRAM module. Figure 2 shows the relative performances of the different hardware multiplier systems. The behavious of a software multiplier emulator is also included in the graph.

The slowest performance was that of the single multiplier which required around twice the time of the dual-multiplier system. However, as the size of the matrix increased, the time required by the single-multiplier became more than twice that of the dual-multiplier. The big deviation at larger size matrices can be explained by the fact that the pre-processing of data had caused the host machine to become low on memory resources, using nearly all of the available physical memory.

5. Concluding Remarks

Our future work will address the scalability of our method with larger number of multipliers implemented to operate in parallel, and will investigate different host-to-FPGA communication strategies.

References

[1] C.P. Kruskal, L. Rudolph and M. Snir, "Techniques for parallel manipulation of sparse matrices," *Theoretical Computer Science* **64** (1989), pp. 135 – 157.

[2] M. Middendorf, H. Schmeck, H. Schroder and G. Turner, "Multiplication of matrices with different sparseness properties on dynamically recon gurable meshes," *VLSI-Design*.

[3] M. Wojko and H. ElGindy, "Self con guring binary multipliers for LUT addressable FPGAs," *Proceedings of 6th PART Conference* , Adelaide, 1998.

[4] Xilinx homepage, http://www.xilinx.com.

[5] Yen-Liang Shue, "Advanced matrix-vector processing with FPGA-based systems," *Bachelor of Computer Engineering Thesis*, University of New South Wales, 2001.

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: National Univ of Defense Tech. Downloaded on December 07,2020 at 13:49:37 UTC from IEEE Xplore. Restrictions apply.