Sparse Matrix-Vector Multiplication for Finite Element Method Matrices on FPGAs

Yousef El-Kurdi, Warren J. Gross, and Dennis Giannacopoulos
Department of Electrical and Computer Engineering, McGill University
Montreal (QC), H3A 2A7, Canada
yousef.el-kurdi@mail.mcgill.ca, wjgross@ece.mcgill.ca, dennis.giannacopoulos@mcgill.ca

Abstract

We present an architecture and an implementation of an FPGA-based sparse matrix-vector multiplier (SMVM) for use in the iterative solution of large, sparse systems of equations arising from Finite Element Method (FEM) applications. The architecture is based on a pipelined linear array of processing elements (PEs). A hardware-oriented matrix "striping" scheme is developed which reduces the number of required processing elements. Our current 8 PE prototype achieves a peak performance of 1.76 GFLOPS and a sustained performance of 1.5 GFLOPS with 8 GB/s of memory bandwidth. The SMVM-pipeline uses 30% of the logic resources and 40% of the memory resources of a Stratix S80 FPGA. By virtue of the local interconnect between the PEs, the SMVM-pipeline obtain scalability features that is only limited by FPGA resources instead of the communication overhead.

1. Introduction

Making good use of the reprogrammability feature of FPGAs motivates devising specialized algorithms and hardware designs optimized for accelerating computations for specific application areas. The Finite Element Method (FEM) is a widely used engineering analysis tool based on obtaining a numerically approximate solution for a given mathematical model of a structure. The application of FEM requires the solution of a large system of linear equations repeatedly. The resulting linear system is characterized by the system matrix (A) which is usually large and sparse. Iterative solvers, mainly the Conjugate Gradient (CG) method, are typically used to solve such systems due to their low storage requirements. The dominant operation in each iteration of the CG algorithm is the Sparse-Matrix Vector Multiplication (SMVM) which is computed as $Y^{(k)} = A \times X^{(k)}$, where (k) is the k'th iteration of the CG algorithm, A is a $N \times N$ sparse matrix, and X and Y are $N \times 1$ dense vectors.

FPGAs have been shown to outperform general-purpose CPUs in sustained and peak floating-point performance [1]. More so, there is a great degree of pipelineability and parallelism inherent to the SMVM computation that can be greatly exploited by FPGAs. SMVM computation involving FEM matrices requires large memory storage and bandwidth due to the large size of the FEM mesh. Our approach does not require storing the matrix A inside the FPGA chip before starting the SMVM computation which results in efficient computational resource scalability as the FEM matrix size increases. As a result, our architecture can rely on external SDRAM for storage capacity while utilizing the FPGAs computational resources in a stream-through approach. Our stream-through approach also is inherently suited for the iterative application of the SMVM as required by the CG method. It has been shown in [2] that the elements of an FEM matrix can be covered by a bounded number of stripes. We have developed our own striping algorithm [3] which results in an efficient utilization of the SMVM-pipeline.

2. SMVM-Pipeline Implementation

We have implemented a linear array pipeline architecture, SMVM-pipeline, to efficiently perform the SMVM computation targeted for the FEM application area. The SMVM-pipeline is shown in Figure 1. The SMVM-pipeline computes the overall SMVM operation in parallel by implementing a pipeline of Processing Elements (PEs) each containing a cascade of pipelined floating-point arithmetic units made of an adder and a multiplier. The matrix elements are striped using our striping algorithm and fed to each PE from the top, while the X and Y vectors are fed into the pipeline horizontally from the same side.

14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'06)
0-7695-2661-6/06 $20.00 © 2006 IEEE

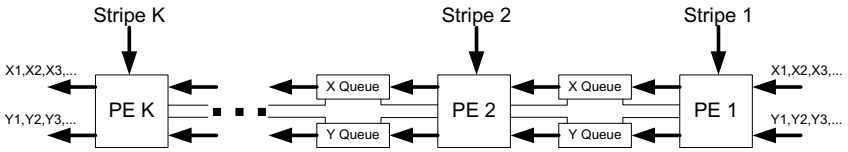

Figure 1. SMVM-pipeline.

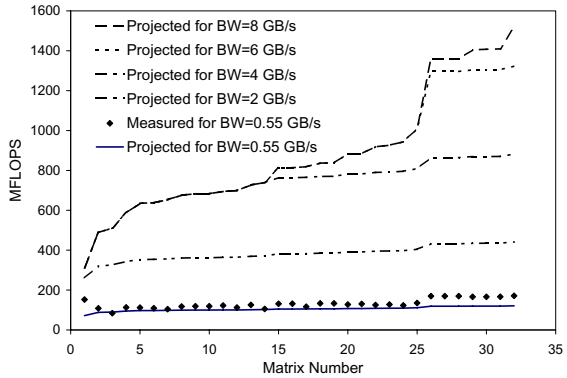

Figure 2. MFLOPS performace as the memory bandwidth varies.

3. Results

The implemented SMVM-pipeline contains 8 PEs and is clocked at 110 MHz obtaining a peak performance of 1.76 GFLOPS. However, the actual SMVM-pipeline performance depends on the available SDRAM bandwidth. The effective SDRAM bandwidth available to the current SMVM-pipeline is limited to 0.55 GB/s. Future upgrades to the SMVM-pipeline will utilize more SDRAM bandwidth when fully implemented on the TM4 [4] system which provides up to 16 GB/s of SDRAM bandwidth for four FPGAs.

Figure 2 shows the MFLOPS results for FEM test matrices along with projected results modeled by our performance analysis equation as the memory bandwidth scales from 0.55 to 8 GB/s. Table 1 shows performance comparison to other computing platforms. The SMVM-pipeline prototype achieves comparable MFLOPS performance while utilizing only 30% of the FPGA logical resources and 40% of internal RAM. Future work will demonstrate increased performance as the number of PEs is further scaled-up.

4. Conclusion

In this paper we analyzed the acceleration of SMVM computation on FPGAs for FEM and demonstrated an SMVM-pipeline architecture that can obtain high performance for very large sparse FEM matrices. In addition, we developed our own pipelinable striping scheme, improving the overall utilization of our design. Our stream-through type architecture provides the added advantage of enabling an iterative implementation of the SMVM computation for the CG method.

Table 1. SMVM performance comparison.

Processor	Peak MFLOPS	Utilization	Ref.
Pentium 4	3000	14.29%	[5]
Power 4	5200	16.67%	[5]
Sun Ultra 3	1800	6.25%	[5]
Itanium	3200	10.00%	[5]
Itanium 2	3600	33.33%	[5]
V2 6000-4	2240	66.67%	[5]
V2-Pro 70	2880	20% – 75%	[6]
Stratix S80	1760	17.74%–86.4%	[this work]

References

[1] K. D. Underwood and K. S. Hemmert, "Closing the gap: CPU and FPGA trends in sustainable floating-point BLAS performance," in *FCCM '04: Proceedings of the 12th Annual IEEE Symposium on Field Programmable Custom Computing Machines*, 2004, pp. 219–228.

[2] R. Melhem, "Determination of stripe structure for finite element matrices," *SIAM Journal on Numerical Analysis*, vol. 24, no. 6, pp. 1419–1433, 1987.

[3] Y. El Kurdi, W. J. Gross, and D. Giannacopoulos, "Hardware acceleration for finite element electromagnetics: Efficient sparse matrix floating-point computations with field programmable gate arrays," in *CEFC '06: Proceedings of the 12th Biennial IEEE Conference on Electromagnetic Field Computation*, 2006.

[4] "The Transmogrifier-4 project," http://www.eecg.utoronto.ca/~tm4/, 2005, University of Toronto.

[5] M. DeLorimier and A. DeHon, "Floating-point sparse matrix-vector multiply for FPGAs." in *FPGA '05: Proceedings of the 2005 ACM/SIGDA 13th International Symposium on Field-Programmable Gate Arrays*, 2005, pp. 75–85.

[6] L. Zhuo and V. K. Prasanna, "Sparse matrix-vector multiplication on FPGAs," in *FPGA '05: Proceedings of the 2005 ACM/SIGDA 13th International Symposium on Field-Programmable Gate Arrays*, 2005, pp. 63–74.