Sparse Linear Solver for Power System Analyis using FPGA *

J. R. Johnson† P. Nagvajara‡ C. Nwankpa§

1 Extended Abstract

Load flow computation and contingency analysis is the foundation of power system analysis. Numerical solution to load flow equations are typically computed using Newton-Raphson iteration, and the most time consuming component of the computation is the solution of a sparse linear system needed for the update each iteration. When an appropriate elimination ordering is used, direct solvers are more effective than iterative solvers. In practice these systems involve a larger number of variables (50,000 or more); however, when the sparsity is utilized effectively these systems can be solved in a modest amount of time (seconds). Despite the modest computation time for the linear solver, the number of systems that must be solved is large and current computation platforms and approaches do not yield the desired performance. Because of the relatively small granularity of the linear solver, the use of a coarse-grained parallel solver does not provide an effective means to improve performance. In this talk, it is argued that a hardware solution, implemented in FPGA, using fine-grained parallelism, provides a cost-effective means to achieve the desired performance.

Previous work [1, 2, 3] has shown that FPGA can be effectively used for floating-point intensive scientific computation. It was shown that high MFLOP rates could be achieved by utilizing multiple floating-point units,

*This work was partially supported by DOE grant #ER63384, PowerGrid - A Computation Engine for Large-Scale Electric Networks

†Department of Computer Science, Drexel University, Philadelphia, PA 19104. email: jjohnson@cs.drexel.edu

‡Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA 19104. email: nagvajara@ece.drexel.edu

§Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA 19104. email: chika@nwankpa.ece.drexel.edu

and FPGA could outperform PCs and workstations, running at much higher-lock rates, on dense matrix computations. The current work argues that similar benefit can be obtained for the sparse matrix computations arising in power system analysis. These conclusions are based on operation counts and system analysis for a collection of benchmark systems arising in practice.

Benchmark data indicates that between 1 and 3 percent of peak floating point performance was obtained using a state-of-the-art sparse solver (UMFPACK) running on 2.60 GHz Pentium 4. The solve time for the largest system (50,092 unknowns and 361,530 non-zero entries) was 1.39 seconds.

A pipelined floating point core was designed for the Altera Stratix family of FPGAs. An instantiation of the core on an Altera Stratix with speed rating (-5) operates at 200 MHz for addition and multiplication and 70 MHz for divion. Moreover, there is sufficient room for 10 units. Assuming 100% utilization of eight FPUs, the projected performance for the FPGA implementation is 0.069 seconds, which provides a 20-fold improvement. While it is optimistic to assume perfect efficiency, hard-wired control should provide substantially better efficiency than available with a standard processor. Moreover, analysis of the LU factorization shows that the average number of updates per row throughout the factorization is 20.3, which provides sufficient parallelism to benefit from 8 FPUs. An implementation, and more detailed model, is being carried out to determine the attainable efficiency.

References

[1] J. Johnson, P. Nagvajara, and C. Nwankpa. High-Performance Linear Algebra Processor using FPGA. In *Proc. High Performance Embedded Computing (HPEC 2003)*. Sept. 22-24, 2003.

[2] K. Underwood. FPGAs vs. CPUs: Trends in Peak Floating-Point Performance. In *Proc. International Symposium on Field-Programmable Gate Arrays (FPGA 2004)*. Feb. 22-24, 2004.

[3] Ling Zhuo and Viktor K. Prasanna. Scalable and Modular Algorithms for Floating-Point Matrix Multiplication on FPGAs. In *Proc. International Parallel and Distributed Processing Symposium (IPDPS 2004)*, 2004.