High-Performance Linear Algebra Processor using FPGA *

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

1 Extended Abstract

With recent advances in FPGA (Field Programmable Gate Array) technology it is now feasible to use these devices to build special purpose processors for floating point intensive applications that arise in scientific computing. FPGA provides programmable hardware that can be used to design custom hardware without the high-cost of traditional hardware design. In this talk we discuss two multi-processor designs using FPGA for basic linear algebra computations such as matrix multiplication and LU factorization. The first design is a purely hardware solution for dense matrix computations, and the second design uses a hardware/software solution for sparse matrix computations. The hardware solution uses the regular structure available in dense linear algebra computations to design custom processors with hard-wired communication patterns. The hardware/software solution uses embedded processors with the flexibility to program the irregular communication patterns required by sparse matrix computations.

The dense matrix processor utilizes a distributed memory architecture connected in a ring topology, with hardwired control for communication. Each processing element consists of pipelined multiply-accumulate hardware, and local memory to store part of the input and output matrices.

*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

Extra buffers are available to provide overlapped communication and computation so that while a computation is being performed the next inputs can be downloaded from the host computer. This allows the FPGA to be used as a hardware accelerator as part of a larger matrix computation using various block algorithms. In fact, the matrix processor supports BLAS routines such as GEMM (http://www.netlib.org/blas/).

The sparse matrix processor also uses a distributed memory architecture, however, it uses embedded processor cores which execute parallel programs written in C. A ring interconnect was designed and special instructions were added to the processor cores to support interprocessor communication. In addition, hardware support for floating point computations were added which are accessible to the processor through extra instructions. A parallel algorithm for LU factorization was implemented.

The resulting processor for sparse LU factorization is being designed for applications in power computations, where very large sparse systems arising from load-flow computation are solved as part of the Newton-Raphson method for solving a system of non-linear equations. These systems have problem specific structure which can be incorporated into the processor design. While the current investigation centers around this application, many other problems could benefit from a high-performance processor geared towards both sparse and dense matrix computations.

In this work parameterized designs, allowing a scalable number of processors, were created using the hardware definition language VHDL. The resulting design was synthesized and testing using the cyclone development board from Altera (http://www.altera.com/). Embedded NIOS processors were used for the processor for sparse LU factorization. While this board allowed verification of the design, it does not have sufficient hardware resources to adequately demonstrate the potential performance. Current FPGA devices have built-in hardware multipliers and large amounts of on chip memory, in addition to a large number of programmable logic blocks. These resources are crucial to obtaining high-performance, especially for computations with floating point numbers. For example, the Xilinx Virtex 2 (XC2V8000), see www.xilinx.com, has 168 pipelined 18-bit multipliers which can be used to implement 42 single-precision multiply-accumulate units.

Consequently a high-level performance model was created to estimate the performance using the high-end FPGA devices that are currently available and those that will be available in the next few years. The predicated performance was compared to efficient software solutions on high-speed workstations and clusters of workstations. Using clock speeds and the number of processors that can be implemented on the Virtex 2, a

400×400 matrix can be multiplied in 11ms, which corresponds to 11,570 MFLOPS. This compares to 10,000 MFLOPS obtained using a four processor 800 MHz Itanium processor with the Intel Math Kernel library (see http://www.intel.com/software/products/mkl/mkl52/specs.htm).

A similar performance model was used for the sparse linear solver. Assuming eight processors running at 400MHz with an 80 MHz floating point unit, which is currently possible using Xilinx FPGAs with a floating point core sold from Digital Core Design (http://www.dcd.com.pl/), the linear system needed to solve the 7917-bus power system could be solved in .084 seconds as compared to .666 seconds using the WSMP sparse linear solver (http://www.research.ibm.com/math/OpResearch/wsmp.html) on a 1.4GHz Pentium IV. While the times reported here are estimates based on a performance model, and hence should not be taken at face value, they show enough promise that the use of high-end FPGA devices, on a board with multiple FPGAs, to build a special-purpose linear algebra processor should be seriously investigated.