ABSTRACT

1. Motivation and Background
Heterogeneous computing with FPGAs (field-programmable gate arrays) is a potential area of interest for next generation supercomputers for solving exascale problems. The amount of additional effort invested into integrating FPGA device programming into existing workflows, for accelerating compute intensive kernels, is significant. Writing scientific code to run on FPGAs using HDLs (Hardware Description Languages) is complex. The Intel OpenCL SDK converts OpenCL code to FPGA executable code. The OpenARC compiler [1, 2] provides easier FPGA programmability by translating OpenACC directive based code into OpenCL optimized for FPGAs. Directive based programming models like OpenACC hide low level language complexities.

In this work, we port a GAMESS [3, 4] computational chemistry kernel containing the Hartree-Fock procedure to FPGA enabled machines using OpenARC and evaluate the performance results. The GAMESS-SIMINT Hartree-Fock quantum chemistry method is used both to compute molecular properties and as a starting point for higher accuracy, more computationally demanding methods. The computational bottleneck of the Hartree-Fock procedure is construction of the Fock matrix, which requires computation of many electron repulsion integrals (ERIs). The SIMINT integral package [5] is a highly vectorized, high performance implementation of the Obara-Saika ERI evaluation method.

2. Methodology and Results
Since OpenARC takes only C code as input, we translated the GAMESS-SI-MGMS kernel to pure C code by hand. We then inserted OpenACC directives to parallelize the code. We used OpenARC to transform the code to OpenCL optimized for FPGAs. The final step was to use the Intel SDK for OpenCL to compile the code to an FPGA executable. We verified that the FPGA executable produced the same results as the C++ version of the code running on a CPU. Figure 1 compares execution times of the kernel with increasing problem size, where problem size is the number of basis set functions, on a Nalltech Stratix V FPGA accelerator board and on an Intel(R) Xeon(R) E5520 CPU. Figure 2 indicates the speedup achieved for different problem sizes. We achieve up to 9.5X speedup on the FPGA. The tested FPGA (Stratix V) does not contain dedicated floating point cores but the floating point units are synthesized from existing building blocks. We plan to measure the speedup on the newer Nalltech Arria 10, which has hardened floating point units and thus offers higher floating point performance. We will also use the Intel SDK Quartus power estimation tool to estimate power consumption and will compare power and energy consumption of the CPU and FPGA implementations.

Acknowledgments
The paper has been co-authored by a contractor of the U.S. Government under Contract No. DE-AC05-00OR22725. According to the U.S. Government retains a non-exclusive, royalty free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes: Partial support for this work was also provided through AFOSR Grant FA9550-12-1-0476.

References