Porting a **GAMESS Computational Chemistry Kernel** to **FPGAs**

Uma Klaassen – University of Texas at El Paso  
Shirley Moore – Oak Ridge National Lab  
Mark S. Gordon, Kristopher Keipert – Iowa State University  
Jeffrey Vetter, Seyong Lee – Oak Ridge National Lab
1. Motivation

- Additional effort invested into integrating FPGA device programming to existing workflows is significant for accelerating compute intensive kernels.

- Writing code for scientific applications in HDLs (Hardware Description Languages) is complex.

- The **OpenARC compiler** provides easier FPGA programmability.
  - OpenARC translates OpenACC directive based code into OpenCL optimized for FPGAs.
  - The Intel OpenCL SDK converts OpenCL code to FPGA executable code.

- OpenACC and other directive based programming models hide low level language complexities.
2. Background

- We port a GAMESS kernel to FPGA enabled machines using OpenARC and evaluate the performance results.

- GAMESS computational chemistry kernel contains the Hartree-Fock procedure.

- The GAMESS-SIMINT Hartree-Fock quantum chemistry method computations
  - Compute molecular properties
  - A starting point for higher accuracy, for more computationally demanding methods.

- The computational bottleneck of the Hartree-Fock procedure
  - Construction of the Fock matrix.
  - Requires computation of many electron repulsion integrals (ERIs).

- The SIMINT integral package is a highly vectorized, high performance implementation of the Obara-Saika ERI evaluation method.
3. Methodology

- OpenARC takes only C code as input.
- We translated the GAMESS-SIMGMS kernel to pure C code by hand.
- We then inserted OpenACC directives to parallelize the code.
- We used OpenARC to transform the code into OpenCL optimized for FPGAs.
- Intel SDK for OpenCL compiles the code to an FPGA executable.
4. Results

- Figure 1 compares execution times of the kernel with increasing problem size.

- Problem size is the number of basis set functions, on a Nallatech 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.

Figure 1: Runtime on FPGA vs. CPU (logarithmic scale).

Figure 2: Speedup obtained on FPGA vs. CPU.
5. Future/Ongoing work

• We plan to measure the speedup on the newer Nallatech 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.

Figure 1: Runtime on FPGA vs. CPU (logarithmic scale).

Figure 2: Speedup obtained on FPGA vs. CPU.
Questions?

Thank you!