**Mapping a Performance-Portable Global Shallow Water Model onto**

**Field Programmable Gate Arrays Using Advanced Compiler Tools**

David Bennett1, Greg Daughtry1,

Richard Loft2, Miller2

*1*

*2National Center for Atmospheric Research, Boulder CO USA*

**Abstract**

In this paper, we describe and analyze the computational performance of a field-programmable gate array (FPGA) on a standard two-dimensional benchmark for atmospheric science, namely solving the shallow water equations (SWE) on a rotating sphere. The benchmark, which is written in C, accesses the FPGA hardware via a novel compiler toolchain being developed by CacheQ, Inc. The benchmark is implemented without significant changes, thus overcomes a significant barrier to adoption of FPGA for high-performance computing (HPC) applications like weather and climate modeling. The performance of the Xilinx U250 FPGA is compared that of contemporaneous Intel Xeon multicore and Nvidia Tesla GPU systems. we find that the Xilinx U250 FPGA achieves ?? times the performance of an Intel SkyLake, and ?? times the performance of the NVIDIA Tesla V100 on the largest problem considered (??k nodes). The power efficiency of the FPGA is also evaluated, showing distinct advantages over traditional architectures.

1. **Introduction**

Quoting from Elliott, et al., *“The 2D shallow water equations (SWE) are a set of hyperbolic partial differential equations (PDE) derived from depth integration of the Navier-Stokes equations. The SWEs are a valid approximation of the Navier-Stokes equations when the horizontal scale is much larger than the vertical scale [4], such is the case in modeling atmospheric dynamics. When testing numerical methods for atmospheric modeling, implementation of a SWE solver is a logical stepping stone towards modeling the full, 3-dimensional Navier-Stokes equations.”* Numerical methods based on radial basis functions (RBF) for solving PDEs are gaining in popularity because of their simplicity and their inherent “meshless” nature. The numerical capabilities of RBF-FD methods for modeling the SWE’s have been established showing convergence comparable to other high-order methods such as discontinuous Galerkin (DG) [1].

When developing an HPC algorithm, scalability, performance and portability are essential. As of November 2016, Intel multicore architectures compose over 90% of the TOP 500 systems’ processor share. Of these systems, Intel many-core (Xeon Phi) and NVIDIA GPUs compose over 95% of the accelerator share [3]. Thus, to claim performance portability, models must demonstrate scalability and performance on all three architectures. Many of the current standard atmospheric models lack these desirable properties. Most are implemented using MPI and OpenMP and optimized strictly for standard CPU systems, resulting in either bad performance or complete incompatibility on accelerated architectures. Some atmospheric models have been partially converted to CUDA for compatibility with NVidia GPUs[Huang], while others have adopted a strategy of creating locally developed domain specific tools and frameworks[REFMeteoSwiss]. Both approaches appear to require significant software development and maintenance investments, as well as the risk of a good deal of duplication of effort between centers with different models. To avoid these issues, we utilized OpenACC and OpenMP, which provide a directive-based solution for generalization of thread-level parallelism. This addresses the portability of the algorithm while still providing computational efficiency on a variety of HPC architectures.

In this paper, we extend the computational capabilities of RBF-FD methods for atmospheric simulation beyond Intel and NVIDIA architectures by demonstrating performance portability of an RBF-FD SWE solver on an FPGA architecture. In section 2 we begin with a description of the numerical methods and the RBF-FD formulation of the SWEs on a sphere. In section 3 we describe the implementation and algorithmic optimizations of the model on FPGA using the CacheQ software toolchain. In section 4 we present the performance results on Intel Skylake, NVIDIA Tesla V100, and Xilinx U250 architectures, as well as power-performance results.

* 1. **Related Work (in FPGA & similar applications)**

A long-standing issue with applying Field-Programmable Gate Arrays (FPGAs) to HPC problems have been the number of resources on the chip and their programmability. However,

Quoting from Lin Gan: *“In recent years, we start to see some FPGA-based acceleration for modules within a global model ([9],[10],[11]), and for regional weather predictions ([11]).*

*Smith et al. [10] accelerate the Parallel Spectral Transform shallow water model using ORNL’s SRC Computers. Only some subroutines (FFT or LT) is deployed on the FPGA and a small speedup is gained over CPUs. Wilhelm et al. [11] analyze a high-level approach for programming pre- conditioners for an ocean model in climate simulations on FPGAs but do not manage actual acceleration. Oriato et al. [12] accelerate a realistic dynamic core of LAM model using FPGAs. It is a successful trial on reducing resource usage through fixed-point arithmetic.”*

Paper [9] proposed a hybrid CPU-FPGA algorithm that applies single and multiple FPGAs to compute the upwind stencil for the global shallow water equations. Achieving dramatic speedups over CPUs by using mixed-precision calculations.

Paper [8] showed that, using a set of code structure optimizations, OpenCL designs can be compiled to Radix-2 FFT pipelines which outperform IP core-based designs for the same throughput, and that  HDL generated by the OpenCL compiler couls be isolated and seamlessly integrated into existing 3D FFT shells to reduce implementation effort, and that OpenCL-generated FFT compute pipelines used substantially fewer ALMs and 1.6x fewer DSPs compared corresponding IP core versions.

1. **Numerical Methods**
   1. *Shallow Water Equations (SWE) on a Unit Sphere*

The following formulation of the SWE are described in [5]. The SWE in a 3D Cartesian coordinate system for a rotating fluid are given by

where *f* is the Coriolis force, is the velocity vector, *h* is the geopotential height and is the position vector. We need to confine this motion to the surface of a sphere. To do this we first define the projection operator ***P*** by

To confine the motion to the surface of a sphere we confine the gradient and divergence operators to the surface as well by redefining in equations (2.1.1-2) above. We then project the entire right hand side of equation (2.1.1) onto the surface of the sphere as well. For the equation, this gives us

.

Similarly,

with *RHS* as is given in equation 2.1.1.

* 1. *Calculating Radial Basis Function Generated Finite Difference Methods (RBF-FD) Weights*

The solver uses radial basis function finite difference (RBF-FD) methods, which are characterized by large stencils on unstructured grids. The following describes a generalization of RBF-FD methods for approximation of a linear operator acting on a function at a given location . This generalization will then be applied to equations 2.1.4-7 for a RBF-FD formulation of the SWE on a unit sphere.

Given a set of *n* function values, evaluated on a set of *n* nodes, , we would like to approximate with a linear combination of the given function values and a constant. Note that the constant is mathematically unnecessary but helps avoid rounding errors in practical calculations.

The weights, are uniquely determined by enforcing that this linear combination is exact for the linear operator acting on an RBF interpolant of , , with basis functions centered at the node locations . denotes the Euclidian norm and is referred to as the shape parameter. This leads to the linear system

(2.2.2),

which can be solved to determine the weights, . Using these weights, we can approximate with the weighted sum of the given function values by the relation given in equation (2.2.1). Note that equation (2.2.1) approximates with operations.

* 1. *RBF-FD Formulation of the SWE on a Unit Sphere*

The following formulation is described in [2]. Suppose we are given function values and at node points Let be any RBF centered at . The projected gradient operator can be represented by

where is the position in conventional spherical coordinates. Define the *n-*point stencil , which is composed of the *n* nearest node points to the point including itself (we let ) and let be the RBF expansion of a function on this stencil. Then the *n*-point RBF-FD approximation of the operator is given by

where is the vector and *A* is the resulting matrix in equation 2.2.2 using the nodes . The operators and are defined in the same way with their respected dimensions. Let be the *N* by *N* sparse matrix with nonzero entries equal to the corresponding value in the vector . Define and similarly using and . Then the discretization of the *RHS* defined in equation 2.1.4, denoted by , is given by

where is element by element multiplication and and are simply the vectors containing the values of *x, y, z, u, v, w* and *h* at each of the node points . Let . Using this our discretization of equations 2.1.4-7 becomes

Let be *H* at a time *t* and be our time step. This system is then advanced in time using 4th order Runge Kutta by

where

* 1. *Test Case: Flow over an Isolated Mountain*

For verification and performance benchmarking of the RBF-FD SWE solver, we use the standard *Flow Over an Isolated Mountain* test case as described in [6] on an icosahedral node set. The surface of the mountain is described using conventional spherical coordinates by

where , , and with and . This term is added to *h* on the right-hand side of equation 2.3.4 as a forcing term to enforce this surface. The initial conditions are given by

where is the reference geopotential height, is the standard gravitational constant, is the mean radius of the earth, is the rotation rate of Earth and . The icosahedral node set is produced by successive subdivision of the triangular faces of an icosahedron into four new triangular faces and projecting the intersection points onto the surface of a sphere. Figure (2.1) shows an example icosahedral node set.



*Figure (2.1): Example Icosahedral node set on a sphere*

* 1. *Benchmark Workflow*

*Figure 2.1: RBF-FD Solver Workflow Diagram*

Figure 2.1 All the time is spent in the blue section, basically calculating the RHS four times (for RK4).

1. **Implementation and Optimization on FPGAs**

**Field Programmable Gate Arrays (FPGAs) are programmable logic devices that provide custom hardware capability with lookup tables (LUTs) which can implement gate-level logic equations, digital signal processing (DSP) blocks for highly efficient computation and local random access memory that is very high bandwidth. Modern FPGAs are extremely complex devices with very high levels of integration of many of these types of resources that can be configured to exectute in a highly parallel manner. FPGAs typically operate at a much lower core clock frequency, typically 100-700MHz than modern CPU which operate at core clock frequencies in the multiples of GHz range (3-4 GHz). The FPGAs make up for the slower clock rates by having architectures of many orders of magnitude greater numbers of compute processors, memory, and logic with custom interconnect to implement much greater parallelism than is achievable even in superscalar microprocessors with relatively small numbers of ALUs executing speculatively sequential machine code. FPGAs are typically programmed with Hardware Description Languages (HDL) such as Verilog and VHDL, and require specialized skills and expertise to use.**

**High Performance Compute (HPC) applications typically have code which implements an algorithm that iterates over a dataset to implement an algorithm. Performance is typically gated by the innermost loop it is executed the most times. If a loop is of length C and executes N times, then the first order approximation of the runtime (Rs) is given by**

**If the architecture of the machine executing this loop can be implemented in a parallel manner using a custom compute pipeline, then the runtime can be reduced. Given a loop is of length C and executes N times, then the pipeline runtime (Rp) is given by**

**These formulas are simplistic and optimistic. If there are dependencies between data within the loops, the engines must stall until the data can be fetched and moved to where it is needed. Modern CPUs execute at high frequency and have superscalar engines which extract thread- level parallelism. VLIW, GPU or vector processing engine architectures implement SIMD architectures to parallelize operations of the same instruction on many pieces of data at the same time. Generally speaking, FPGA performance wins when the algorithm and dataset can be mapped to the hardware resources, removing as many loop carry dependencies (LCDs) as possible or mitigating their effects.**

**This is very challenging when programming with HDL. There has been research as well as commercial tools developed to move the programming model of FPGAs from HDLs to what are called High Level Synthesis (HLS) tools, which use languages such as C and C++ to allow coding of FPGA behavior. [TBD references]. However, the approaches all require significant FPGA hardware expertise and utilization of low-level specific APIs that do not lend themselves well to an algorithm developer**

**CacheQ toolchain**

1. **Results and Analysis**

In this paper we consider the properties of a single device by holding the number of cores/accelerators constant, and scaling up the problem size. This workload scaling has the effect of exercising the memory subsystem in two different ways: smaller problems are more latency sensitive and larger problems will be less localized in memory and less able to take advantage of caching. In this section, we first analyze the performance characteristics of the various optimizations discussed in [12] as well as those for FPGA’s discussed in section 3. Only after establishing the optimal configurations for running the RBF-FD SWE solver on each architecture, including the memory layout, compiler, optimization flags, etc., are comparisons made.

* 1. *Benchmark Systems and Hardware Implications*

Table 4.1 describes relevant information about the benchmark systems. Note that the V100 has a large shared L2 cache across the entire GPU whereas the Intel architectures have dedicated L2 caches. Therefore, we would expect data locality optimizations to have a much more significant effect on the Intel architectures than on the P100 GPUs as cache thrashing is likely to be much more apparent. We also expect vectorization

optimizations to have a less significant effect on Intel architectures since they currently support noncoalesced vector operations.

* 1. *Benchmark and Runtime Configurations*

The performance results presented throughout section 4 were performed using the *Flow over a Mountain* test case on varying sized icosahedral node sets as is described in Section 2.4. Each case was run for 100 timesteps and the presented results are that of the average of 10 separate runs. Each simulation result was verified at the end of the run using established high-resolution results from alternative high order models (see [1] for further verification methods). For each study, the optimal configuration for all other parameters and runtime settings were used to ensure relevancy of the performance results for each optimization study. All experiments were run in single precision. <<Explain about double precision on FPGAs>>.

Choices that led to the baseline configurations for the results presenting in this section. Skylake node, V

*Intel Skylake* It is essential to use optimal thread binding and affinity settings OpenMP models on Intel architectures. In Table 1 shows the thread binding and affinity selections that were used on the Intel Xeon Skylake. One OpenMP thread per core and static OpenMP scheduling. The specific environment settings used on SkyLake to achieve this are displayed in table 4.2.

*NVIDIA Tesla V100* For the V100 system we found that it was best to let the OpenACC implementation determine the optimal runtime parameterizations.

*Table 4.1: Test System Hardware Fact Sheet*

|  |  |  |  |
| --- | --- | --- | --- |
| Target Architecture | XiLinx U250 | Xeon | Tesla GPU |
| System | CacheQ, Inc. | NCAR HPC Lab | NCAR Casper Cluster |
| Node Composition | ??? | Dual Socket Intel SkyLake CPU | 4X Tesla V100 GPUs with NVLINK |
| Scalability | Up to 72 nodes | 1 node | Up to 8 nodes (32 GPUs) |
| Interconnect | N/A | N/A | 56 GB/s FDR |
| Memory Bandwidth | ??? GB/s per socket | ???GB/s | 720 GB/s |
| Peak DP Performance | ??? | ?? TFLOPS | 5.3 TFLOPS |
| L1 Cache |  |  |  |
| L2 Cache |  |  |  |
| L3 Cache |  |  |  |

**CPU -** Intel Skylake/OpenMP -

**Power:** look into HPC Futures Lab power sensing.

Date chip GA:???

Fab:???

**GPU - NVIDIA** Tesla V100/GPU -

power:SMI toolchain

Date chip GA: ???

Fab: ???

**FPGA -** Xilinx Alveo U250 - The Alveo U250 offers 1.728M LUTs, 12,288 DSP slices, 3,456K registers, 64GB of DDR4 memory with 77 GB/sec, dual 100Gbps network interfaces. The internal SRAM bandwidth is 47 TB/s. Power and thermal specs are: 225W, with typical draw of 110W

Power: overclocking ???

Date chip GA: ???

Fab: ???

*Workload Scaling Results*

Figure 4.X shows the workload scaling performance results for each architecture.

1. **Conclusions**

In this paper, we have described an RBF-FD formulation of the SWEs on a rotating sphere for two-dimensional approximation of atmospheric dynamics. We then described the parallelization and algorithmic optimization methods used to implement this formulation on an FPGA.

1. **Acknowledgements**

**References**

[1] Flyer, N., Lehto, E., Blaise, S., Wright, G.B., St-Cyr, A., “A guide to RBF-generated finite differences for nonlinear transport: Shallow water simulations on a sphere”, J. Comput. Phys. **231**(11), 4078–4095 (2012)

[2] Flyer, N., Wright, G.B., “A radial basis function method for the shallow water equations on a sphere”, Proc. Roy. Soc. A 465 (2009) 1949–1976.

[3] List Statistics. (n.d.). Retrieved December 13, 2016, from <https://www.top500.org/statistics/list/>

[4] Pedlosky, J. (1987). *Geophysical fluid dynamics*. New York: Springer-Verlag.

[5] Fornberg, B., & Flyer, N. (2015). “A primer on radial basis functions with applications to the geosciences”, Philadelphia: SIAM, Society for Industrial and Applied Mathematics.

[6] Williamson, D.L., Drake, J.B., Hack, J.J., Jakob, R., Swarztrauber, P.N., “A standard test set for numerical approximations to the shallow water equations in spherical geometry”, J. Comput. Phys. 102 (1992) 211–224.

[7] Liu, Wai-Hung, and Andrew H. Sherman. "Comparative Analysis of the Cuthill–McKee and the Reverse Cuthill–McKee Ordering Algorithms for Sparse Matrices." *SIAM Journal on Numerical Analysis* 13.2 (1976): 198-213. Web.

[8] Sanaullah, A., Herbordt. M.C., “FPGA HPC using OpenCL: Case Study in 3D FFT”, in Proceedings of the 9th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies (HEART 18), (2018), Toronto ON, Canada; ACM, New York, NY, Article No. 7, pages 7:1—7:6, isbn 978-1-4503-6542-0, doi>10.1145/3241793.3241800.

[9] Gan, L., Fu, H., Luk, W., Yang, C., Xue, W., Huang, X., Zhang, Y., Yang, G., “Accelerating Solvers for Global Atmospheric Solvers Through Mixed-Precision Data Flow Engine”,???

[10] M. C. Smith, J. S. Vetter, and X. Liang, “Accelerating scientific applications with the SRC-6 reconfigurable computer: Methodologies and analysis,” in *IPDPS*. IEEE, 2005, pp. 157b–157b.

[11] D.-M. F. Wilhelm and N. ad Weinstraße, “Parallel preconditioners for an ocean model in climate simulations.” ???

[12] D. Oriato, S. Tilbury, M. Marrocu, and G. Pusceddu, “Acceleration of a Meteorological Limited Area Model with Dataflow Engines,” in *2012 Symposium on SAAHPC*, 2012, pp. 129–132.

[13] Elliott, S., Ta, T., Kumar, R.R.P., Flyer, N., Loft, R.D., “Implementation of a Scalable, Performance Portable Shallow Water Equation Solver Using Radial Basis Function Generated Finite Difference Methods”, Inter. Jour. of High Perf. Comp. App., Volume: **33** issue: 4, page(s): 619-631.