Lower Precision for Higher Accuracy: Precision and Resolution Exploration for Shallow Water Equations

James Stanley Targett*, Xinyu Niu*, Francis Russell*, Wayne Luk*, Stephen Jeffress†, Peter Düben†
* Department of Computing, Imperial College London, United Kingdom
† University of Oxford, AOPP, Department of Physics, Oxford, United Kingdom

***Abstract*—Accurate forecasts of future climate with numerical models of atmosphere and ocean are of vital importance. However, forecast quality is often limited by the available computational power. This paper investigates the acceleration of a C-grid shallow water model through the use of reduced precision targeting FPGA technology. Using a double-gyre scenario, we show that the mantissa length of variables can be reduced to 14 bits without affecting the accuracy beyond the error inherent in the model. Our reduced precision FPGA implementation runs 5.4 times faster than a double precision FPGA implementation, and 12 times faster than a multi-threaded CPU implementation. Moreover, our reduced precision FPGA implementation uses 39 times less energy than the CPU implementation and can compute a 100x100 grid for the same energy that the CPU implementation would take for a 29x29 grid.**

I. Introduction

Accurate climate forecasts have many important applications. For example, governments institutions rely upon global climate simulations to make critical policy decisions. However climate model accuracy is limited by available computing power. State of the art climate simulators run on the fastest super computers in the world (e.g. [1]), yet we still lack the computing power to represent important small scale physics that contribute significantly to model uncertainty. Exascale CPU based computing technologies are on the horizon and may provide a solution, but it is expected that the energy demands of these machines will be beyond that which government institutions can sustainably afford [2]. Thus there is a clear need for alternative computing architectures with improved energy efficiency to allow more accurate climate simulations at higher resolution.

An alternative computing technology showing promise as a potential solution is the data flow engine (DFE) technology based on field programmable gate arrays (FPGAs). FPGA based DFEs operate in a manner that is drastically different from a standard CPU. DFEs are customized deep pipelines of concurrent arithmetic operations. They achieve accelerated and energy efficient computation by creating parallelism both in space and in time. Another reason for the energy efficiency of DFEs is the minimization of data movements. This is accomplished both by minimizing the number of bits used for variable precision and also by shortening the total path-length of the data flow. In general, reducing representation bitwidth will reduce computational accuracy. Minimizing numerical precision will save computing resources. If savings are re-invested into larger/faster supercomputers climate simulations can be performed at higher resolution and this will eventually improve forecast accuracy.

978-1-4673-9091-0/15/$31.00

While a complete reduced-precision FPGA data flow implementation of a production quality climate model is a massive undertaking, we have in recent years seen inspiring results with simplified models of climate system components. Most previous studies have focused primarily on the atmosphere (see Section II). However in addition to the atmosphere, the ocean is a dominant driver of global heat transport and a critical part of a climate model.

In this paper we design, develop and test an FPGA data flow implementation of the Shallow Water Equations with an oceanic double gyre test case. The Shallow Water Equations are a set of simplified fluid dynamics equations commonly used for conceptual understanding and to test innovative numerical methods (see section III).

The major emphasis of this paper is a detailed study of the relationship between design performance, design precision, and application resolution of the shallow water equations in an oceanic double gyre configuration. The specific contributions of the study are:

- An FPGA-based DFE based implementation of the shallow water equations.
- A design approach to verify computational accuracy based on data representations, computational resolution, and available resources.
- A hardware design optimised with the proposed approach, showing large performance improvements with acceptable accuracy.

II. Background

In recent years we have seen the beginning of an exciting collaboration between climate scientists and computer scientists focusing on precision optimization and FPGA DFE implementations of climate models. There has been both a top-down approach, where precision analysis is performed on full scale climate models, and also a bottom-up approach where simplified models are implemented on FPGA based DFEs.

The top-down approach has revealed that full scale, operational models of weather and climate are making inefficient use of computational resources due to the prevalent use of double precision floating point arithmetic. Double precision floating point requires a mantissa width of 53 bits. Düben et al. [3] find that 84% of the floating point operations in the Intermediate Global Climate Model (IGCM) could be performed using mantissa widths of only 6 bits without a significant change in climate type simulations. In [4] a similar precision analysis was performed for weather type forecasts, revealing that 98% of calculations currently performed in double precision could

be performed using mantissa widths of only 8 bits without significant loss of forecast accuracy.

The bottom-up approach shows that simplified and limited area atmospheric models can be accelerated while achieving greater energy efficiency using FPGA DFE technology. Oriato et al. [5] accelerate the dynamical core of a limited area meteorological model on a DFE platform and report a 74 times speed up compared to a 12-core multi-threaded CPU implementation. Moreover, reduced precision is widely used in various applications fields, such as seismic imaging [6] and atmospheric modelling [7]. These FPGA designs with reduced precision lead to significant improvements in throughput and power efficiency, compared with supercomputers such as Blue Gene, Cary XK6, and Tianhe-2. However, these designs lack a formal approach to verify whether a design with reduced precision can still meet application accuracy requirements. Russell et al. [8] use the Hellinger distance to verify the difference between the two probability density function that describe the dynamics of the degrees-of-freedom of the Lorenz '95 system. In this paper, we propose a new approach to verify the design accuracy of more complex applications, such as the shallow water setup, that are, in contrast to the idealised Lorenz '95 model, not isotropic in space and need to be evaluated for individual grid points.

III. Approach Overview

A. Shallow Water Model

The shallow water equations describe a shallow layer of fluid which is bounded below by a rigid container and above by a free surface. The equations can be derived from the Navier–Stokes equations by assuming the height of the fluid is negligible compared to its width, and integrating vertically [9]. We use the following form of the shallow water equations in this study.

$$\begin{aligned}
&\frac{\partial u}{\partial t} - \left(f + \overline{\zeta}^{y}\right)\overline{v}^{xy} + \frac{\partial B}{\partial x} = \nu\nabla^2 u \\
&\frac{\partial v}{\partial t} + \left(f + \overline{\zeta}^{x}\right)\overline{u}^{xy} + \frac{\partial B}{\partial y} = \nu\nabla^2 v + \tau \\
&\frac{\partial h}{\partial t} + \frac{\partial}{\partial x}\left(\overline{h}^{x} u\right) + \frac{\partial}{\partial y}\left(\overline{h}^{y} v\right) = 0.
\end{aligned} \tag{1}$$

u: velocity East-West
v: velocity North-South
h: height of the fluid column
f: Coriolis parameter
ν: viscosity
τ: wind forcing
ζ: relative vorticity = $\partial v/\partial x - \partial u/\partial y$
B: Bernouli potental = $gh + (\overline{u}^{x2} + \overline{v}^{y2})/2$
g: gravitational acceleration

The domain size and parameter values of the model are chosen following [10] to create the double gyre flow. The horizontal length is 3,480 km in the North-South (y) and East-West (x). The average fluid column depth is 500 metres. The horizontal surface is discretized using 100×100 gridpoints of the Arakawa C-grid. The grid is staggered such that u, v,and h do not reside on the same gridpoint. The overbars (e.g. $\overline{v}^{xy}$) in equation (1) indicate averages between adjacent grid points in the x and y direction. We use a viscosity ν= 470.23 m^2/s. The Coriolis parameter f varies in the North-South direction and is approximated using a betaplane, f = 4.46e-5+(2.e-11)y. The wind forcing τ is non-zero only in the East-West direction and ranges from 0.2 to -0.4 Pa in a sinusoidal structure as shown in figure 2 of [10]. We use a third-order Adams-Bashforth time stepping scheme with a timestep of 25 seconds.

B. Hardware Architecture

The discretized shallow water equations are mapped to hardware as a 4-point start stencil. Each iteration uses the current values for h, u, and v (collectively called $d(t)$), as well as the change in those values at the previous time step ($d(t-1)$) and the time-step previous to that ($d(t-2)$). The change in the values for the current time-step is computed. All the values are then combined using the Adams-Bashforth method. Stencil operations in the calculation and Adams-Bashforth modules are pipelined, and iterate through C-grid space one point per cycle. The on-chip memory buffer stores three rows of input data (corresponding to the three labelled rows in memory layout) to maximise data reuse. At each clock cycle, the kernel streams in one data, updates the on-chip buffer, and generates one result.

In order to optimize the hardware design, we duplicate the application kernel through **Kernel pipelining**. Kernel pipelining involves connecting the input and output of multiple kernels to process multiple time steps per clock cycle. Instead of writing data for the next time step $d(t+1)$ back into memory, the following kernel uses $d(t+1)$ as input, and calculates the results for the time step after (i.e. $d(t+2)$). Therefore we eliminate the necessity to write to and read from external memory for the intermediate time steps. This significantly reduces memory bandwidth requirements.

IV. Accuracy Analysis

Besides the number of pipeline stages, other design parameters affecting performance are precision and resolution. Precision indicates the applied data representation, and resolution indicates the width of the grid measured in the number of points. This section covers the impacts of precision and resolution on resource usage, design throughput, and model accuracy.

A. Accuracy Verification Approach

The shallow water equations show chaotic behaviour in the form of eddies. Therefore, two simulations with slightly different but correct model implementations will diverge from each other when started from the same initial conditions. It is therefore very difficult to compare the quality of different model simulations. However, taking mean values of fields allows us to characterise the dynamics of a model in a climate-type way and enables us to compare the dynamics of different implementations. Equation 2 shows how to calculate the mean values for each grid cell.

$$\text{mean-field}(D)_{i,j} = \frac{\sum_{t=1}^{\text{number of time-steps}} D_{i,j,t}}{\text{number of time-steps}} \tag{2}$$

Algorithm 1 Accuracy verification Algorithm.

1: **for** $D \in$ reference designs $(D_0, D_1, \ldots, D_n)$ **do**
2: **for** $(i, j) \in$ (width, height) **do**
3: **for** $k \in N$ time steps **do**
4: mean-field$(D)_{i,j}$ += state$(D)_{i,j,k}$
5: sd-field$(D)_{i,j}$ += state$(D)_{i,j,k}$
6: **end for**
7: mean-field$(D)_{i,j}$ = $\frac{\text{mean-field}(D)_{i,j}}{N}$
8: sd-field$(D)_{i,j}$ = $\frac{\text{sd-field}(D)_{i,j}}{N}$
9: **end for**
10: **end for**
11: **for** $D \in$ reference designs $(D_0, D_1, \ldots, D_n)$ **do**
12: **for** $(i, j) \in$ (width, height) **do**
13: mean-error(D) += $(\text{mean-field}(D_0)_{i,j} - \text{mean-field}(D)_{i,j})^2$
14: sd-error(D) += $(\text{sd-field}(D_0)_{i,j} - \text{sd-field}(D)_{i,j})^2$
15: **end for**
16: mean-error(D) = $\frac{\sqrt{\text{mean-error}}}{\text{width}\cdot\text{height}}$
17: sd-error(D) = $\frac{\sqrt{\text{sd-error}}}{\text{width}\cdot\text{height}}$
18: **end for**
19: acceptable-mean-accuracy = max(mean-error(reference-designs))
20: acceptable-sd-accuracy = max(sd-error(reference-designs))

To get more information about the dynamics of different model setups beyond mean values, we also calculate the standard deviations of the different fields.

$$\text{sd-field}(D)_{i,j} = \frac{\sqrt{\left(\sum_{t=1}^{\text{number of time-steps}} (D_{i,j,t})^2\right) - (\text{mean-field}(D)_{i,j})^2}}{\text{number of time-steps}} \quad (3)$$

If there are no eddies we expect a steady solution and the standard deviation should be zero at all points. If the solution is unsteady, eddies will be present that change their position with time and the standard deviation of the fields will be non-zero.

To find how far precision can be reduced we must find metrics to set limits on what is acceptable. If precision can be reduced without breaching these limits then the reduction of precision is acceptable. The metric we will use to compare implementations is the L2 Norm of the absolute errors of the mean per grid cell (Equation 4), or *mean-error* for short. The metric to compare standard deviation-fields, sd-error, is calculated accordingly.

$$\text{mean-error}(\text{ref}, \text{test}) = \frac{\sqrt{\sum_{i=1}^{\text{width}} \sum_{j=1}^{\text{height}} (\text{mean-field}(ref)_{i,j} - \text{mean-field}(test)_{i,j})^2}}{\text{width} \cdot \text{height}} \quad (4)$$

We build the reference accuracy level by calculating *mean-field* and *standard deviation-field* of various reference designs (lines 1−10 of Algorithm 1), finding the mean-error and sd-error (lines 11−18), and taking the largest error (lines 19 and 20). Given a design to test, we compare the *mean-field* and *standard deviation-field* with the reference design, and using the L2 Norm of the differences as the metrics to verify whether the accuracy of the tested design is less than the acceptable error given by the algorithm.

All our data are taken from 100x100 grids with a time-step of 25 seconds. We run the model for 25 million iterations (approximately 20 years). The state of the model is output every 25 thousand iterations. The first half of this data is discarded as a conservative measure to prevent the spin up of the model affecting the data. The remaining 500 fields are then used to construct mean-fields and standard deviation-fields of the variables.

We have two forms of reference implementation to find the error inherent in the model. The model normally starts with a flat ocean that gets spun up to a normal state. If we add zero-mean noise to the initial state the final mean should not be affected. Therefore the error relative to the normal version provides an estimate of the measurement error at the given level of statistics. We also know that since floating point is not associative or commutative the order of operations that the compiler chooses will affect the result. Therefore using implementations created with different compilers and different optimisation settings will allow us to create more references.

B. Precision Modelling

Reducing the precision (i.e. the number of bits representing a datum) in a design has two impacts on the design. On the one hand, reduced precision decrease the resource usage and bandwidth requirements of a design, since less bits need to be transferred and processed. Given the same amount of available resources, this increases the number of pipeline stages. On the other hand, designs with reduced precision will have higher computation errors compared with double-precision reference designs.

We calculate the accuracy of a design with precision (e, m) by taking the *mean-error* and *sd-error* of the design. We call the error metrics mean-error$_{e,m}$ and sd-error$_{e,m}$ respectively. Therefore, the accuracy constraints can be expressed as:

$$\text{mean-error}_{e,m} \leqslant \text{acceptable-mean-accuracy} \quad (5)$$

$$\text{sd-error}_{e,m} \leqslant \text{acceptable-sd-accuracy} \quad (6)$$

If performance is improved by reducing precision we can then increase resolution while keeping the orginal runtime the same. We always use a square grid doubling the resolution-width causes the number of grid points to increase by a factor of four. Another effect of doubling the resolution width is that the length of each time-step must be halved to satisfy the Courant–Friedrichs–Lewy (CFL) condition for stability. This means that the overall time required to simulate a certain period is increased by a factor of eight. We see that the computation time increases as $\mathrm{O}(r^3)$ where r is resolution width.

V. Results

Our CPU tests are run on a computer with Dual Intel Xeon E5-2640 processors with hyper-threading disabled and 64GB of DDR3-1333 RAM. Our FPGA tests are run on a machine with a quad-core i7 870, processor, 16GB of DDR3-1600 RAM, and a Maxeler MAX3 card with a DFE based on Xilinx Virtex-6 FPGA technology.

A. Error Results

To find the acceptable error we run the CPU implementation with noise added to the initial height field as a uniform distribution between $\pm$0.5, 10, 20, 40, and 80 $\cdot 10^{-3}$ metres. We also run the model as compiled with ICC with optimisations

Implementation	Clock rate	Thousand iterations per second	Speedup relative to		Power	Joules per thousand iterations	Energy reduction relative to	
			12 thread CPU	53 bit FPGA			12 thread CPU	53 bit FPGA
1 thread CPU		1.273	0.16x		411W	323	0.176x	
12 thread CPU		7.948	1x		453W	57.0	1x	
53 bit FPGA	190 MHz	18.252	2.30x	1x	140W	4.96	11.5x	1x
24 bit FPGA	150 MHz	42.874	5.39x	2.34x	137W	3.20	17.8x	1.55x
14 bit FPGA	150 MHz	97.876	12.3x	5.35x	141W	1.44	39.6x	3.44x

TABLE I. PERFORMANCE RESULTS

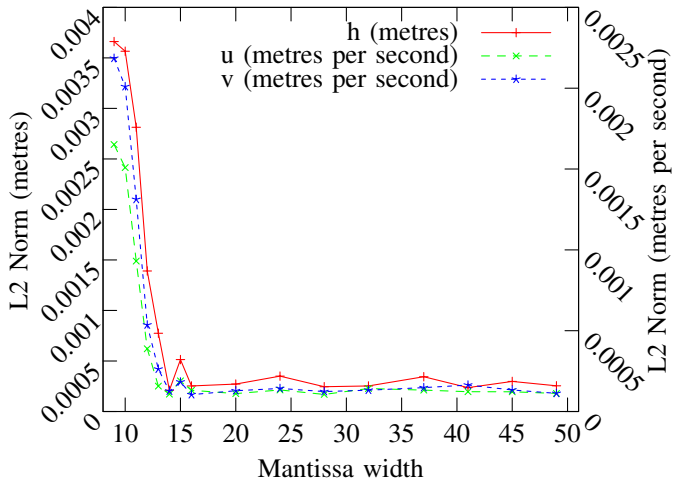

Fig. 1. L2 norm of absolute errors in standard deviation-fields with respect to mantissa width.

on and off, with GCC, and the double precision FPGA implementation. We use ICC with the highest optimisations as our main reference, and use the error relative to the other versions to characterise the error in the model.

For the initial noise results we see that the mean-error for h ranges between 0.1×10^{-4} m and 0.2×10^{-4} m. Similarly we see the mean-error for u and v ranges between 0.8×10^{-5} m s^{-1} and 8×10^{-5} m s^{-1}. The sd-error for h ranges between 2.5×10^{-6} m and 4.5×10^{-6} m and the sd-error for u and v ranges between 0.5×10^{-6} m s^{-1} and 2.0×10^{-6} mm s^{-1}.

We see that the mean-errors and sd-errors caused by the differences between the implementations are of a similar scale to those caused by adding noise to the initial state. This is the case for all three variables.

The acceptable-mean-error allows for reduced precision implementations with mantissas as short as 12 bits. The sd-error for the reduced precision implementations can be seen in Figure 1. The sd-error for h, u and v only remains in the acceptable range for mantissas as short as 14 bits. We can therefore use 14 bit mantissas in an acceptable reduced precision implementation of the shallow water model.

B. Performance results

Due to the amount of logic resources the compilation tools are able to increase the clock rate of the double precision implementation to 190MHz. The timing results are taken from processing a 100x100 grid for 25 million iterations.

The performance results are presented in Table I. We see that the 14 bit FPGA runs 12.3 times faster than the 12 thread CPU version and uses 39.6 times less energy. For the CPU version to complete a simulation of the same length in the same time the resolution would have to be reduced by $\sqrt[3]{12.3} = 2.3$ times to a 43x43 grid. For the simulation to be completed using the same energy the resolution would need to be reduced by $\sqrt[3]{39.6} = 3.4$ times to a 29x29.

VI. Conclusion & Discussion

This paper explores the FPGA acceleration of a C-grid shallow water model through reduced precision techniques. The novel aspects of this work include accuracy analysis based on precision modelling, resolution modelling, and performance modelling. Current and future research includes support for further optimisations such as the adoption of non-uniform word-lengths and run-time reconfiguration, and extension of our approach for speeding up more complex climate models.

Acknowledgement

Acknowledgement. The support of EPSRC grant EP/I012036/1 and the European Union Horizon 2020 Programme under Grant Agreement Number 671653 is gratefully acknowledged. Peter Düben and Stephen Jeffress receive funding from an ERC grant (Towards the Prototype Probabilistic Earth-System Model for Climate Prediction, project reference 291406).

References

[1] T. Shimokawabe, T. Aoki, J. Ishida, K. Kawano, and C. Muroi, "145 TFLOPS performance on 3990 GPUs of tsubame 2.0 supercomputer for an operational weather prediction," *Procedia Computer Science*, vol. 4, pp. 1535–1544, 2011.

[2] T. Palmer, "Climate forecasting: Build high-resolution global climate models," *Nature*, vol. 515, no. 7527, pp. 338–339, Nov 2014. [Online]. Available: http://dx.doi.org/10.1038/515338a

[3] P. D. Düben, H. McNamara, and T. Palmer, "The use of imprecise processing to improve accuracy in weather & climate prediction," *Journal of Computational Physics*, vol. 271, pp. 2–18, 2014.

[4] P. D. Düben and T. Palmer, "Benchmark tests for numerical weather forecasts on inexact hardware," *Monthly Weather Review*, vol. 142, no. 10, pp. 3809–3829, 2014.

[5] D. Oriato, S. Tilbury, M. Marrocu, and G. Pusceddu, "Acceleration of a meteorological limited area model with dataflow engines," in *in Symposium on Application Accelerators in High Performance Computing (SAAHPC)*. IEEE, 2012, pp. 129–132.

[6] X. Niu, J. G. F. Coutinho, Y. Wang, and W. Luk, "Dynamic stencil: Effective exploitation of run-time resources in reconfigurable clusters," in *in International Conference on Field-Programmable Technology (FPT)*, 2013, pp. 214–221.

[7] L. Gan, H. Fu, C. Yang, W. Luk, W. Xue, O. Mencer, X. Huang, and G. Yang, "A highly-efficient and green data flow engine for solving euler atmospheric equations," in *24th International Conference on Field Programmable Logic and Applications (FPL)*, 2014.

[8] F. P. Russell, P. D. Düben, X. Niu, W. Luk, and T. N. Palmer, "Architectures and precision analysis for modelling atmospheric variables with chaotic behaviour," in *23rd IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2015, pp. 171–178.

[9] G. K. Vallis, *Atmospheric and oceanic fluid dynamics: fundamentals and large-scale circulation*. Cambridge University Press, 2006.

[10] F. C. Cooper and L. Zanna, "Optimisation of an idealised ocean model, stochastic parameterisation of sub-grid eddies," *Ocean Modelling*, vol. 88, pp. 38–53, 2015.