Accelerating an iterative Helmholtz solver with FPGAs

Art Petrenko, Felix J. Herrmann
Department of Earth, Ocean and Atmospheric Sciences, University of British Columbia

Diego Oriato, Simon Tilbury
Maxeler Technologies

Tristan van Leeuwen
Centrum Wiskunde & Informatica,
formerly with EOAS UBC

January 6, 2013

Abstract

We implement the Kaczmarz row-projection algorithm [Kaczmarz, 1937] on a CPU host + FPGA accelerator platform using techniques of dataflow programming. This algorithm is then used as the preconditioning step in CGMN, a modified version of the conjugate gradients method [Björck and Elfving, 1979] that we use to solve the time-harmonic acoustic isotropic constant density wave equation. Using one accelerator we achieve a speed-up of over 2× compared with one Intel core.

1 Introduction

Using reconfigurable hardware (FPGAs) to handle the computationally intensive kernel of an algorithm is attractive. Unless the algorithm is limited by the bandwidth of an interconnect (such as a link to memory), speed of execution is directly proportional to the amount of data to be processed. As long as the kernel fits onto the FPGA its complexity is irrelevant, since all the operations on an FPGA happen in the same clock tick. In contrast, on a CPU core, more complex algorithms require more clock ticks as instructions are executed sequentially, or with only a small amount of parallelism. We demonstrate hardware acceleration with FPGAs applied to seismic wave propagation simulation, a large computational burden during full-waveform inversion (FWI).

In the frequency domain, and specializing to the acoustic isotropic constant density case, simulating wave propagation corresponds to solving a large linear system of the form

$$H(\mathbf{m}, \omega)\mathbf{u} = \mathbf{q}, \tag{1}$$

where H is the $N \times N$ Helmholtz operator that depends on the earth model $\mathbf{m}$ and the angular frequency ω, $\mathbf{u}$ is the (complex) Fourier transform of the pressure wavefield and $\mathbf{q}$ is the source. We take $\mathbf{m}$ to be the squared slowness ($1/\mathbf{v}^2$), and use a 27-point cube stencil to calculate the entries of H for the 3D case, as in [Operto et al., 2007]. A perfectly matched layer is used at the boundaries of the domain to eliminate reflection artifacts.

Because the matrix H is large (up to $10^9 \times 10^9$) and sparse (only a few non-zero diagonals), an iterative algorithm like the method of conjugate gradients (CG) is well suited to solve Equation 1. However CG cannot be used directly because H is not symmetric positive semidefinite. Instead we use the Kaczmarz algorithm to transform Equation 1 into an equivalent system that does have these properties. The Kaczmarz algorithm [Kaczmarz, 1937] is an iterative method for solving generic linear systems $A\mathbf{x} = \mathbf{b}$ that projects its iterate $\mathbf{x}_k$ onto the hyperplane defined by a row $\mathbf{a}_i$ of the matrix A:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + w(b_i - \langle \mathbf{a}_i, \mathbf{x}_k \rangle)\frac{\mathbf{a}_i^*}{\|\mathbf{a}_i\|^2}. \tag{2}$$

Here w is a relaxation factor we set equal to 1.5 and b_i is the i^{th} element of the right hand side. We refer to one application of Equation 2 as a Kaczmarz iteration, and to the projections onto each row from first to last as a forward Kaczmarz sweep (FKSWP). Backward sweeps correspond to projecting onto the rows in reverse order, and a double sweep (DKSWP) is a forward sweep followed by a backward sweep (k: $1 \rightarrow 2N$; i: $1 \rightarrow N, N \rightarrow 1$). Equation 1 is transformed into the equivalent system

$$\begin{aligned} \text{DKSWP}(H, \mathbf{u}, \mathbf{q}, w) = \mathbf{u} = Q\mathbf{u} + R\mathbf{q} \\ (I - Q)\mathbf{u} = R\mathbf{q}, \end{aligned} \tag{3}$$

where the action of matrices Q and R is computed using a double Kaczmarz sweep. Equation 3 is then solved with CG, an algorithm known as CGMN and due to [Björck and Elfving, 1979].

Our contribution is implementing the double Kaczmarz sweep for a CPU host + FPGA accelerator platform. Related work has been done by [Elble et al., 2010], who have examined the improvements that Kaczmarz and other algorithms experience on Graphical Processing Units (GPUs).

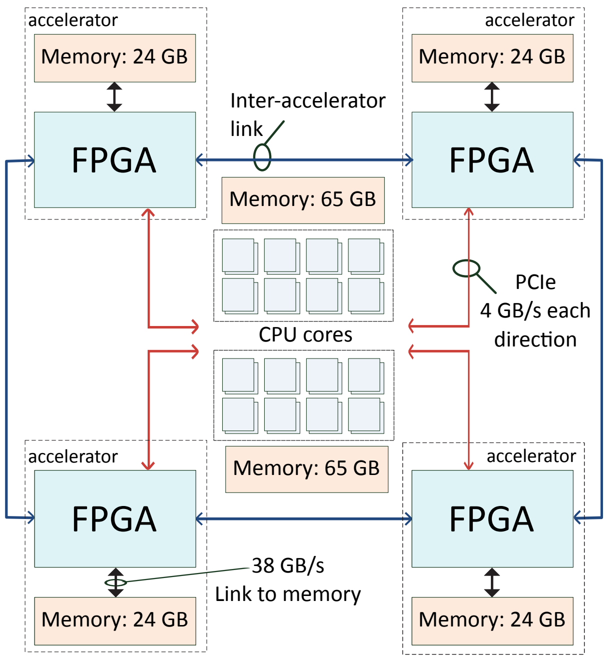

Figure 1: System of CPU host + FPGA accelerators; adapted from [Pell et al., 2013].

2 Implementation

The target platform for our implementation is described in [Pell et al., 2013] and shown in Figure 1. Four dataflow engines (DFEs), consisting of FPGAs and their associated large memory (LMem), are connected via a PCIe bus to the CPU. Each FPGA also has 4.6 MB of fast on-chip RAM (FMem). Coarse-grained parallelization by solving problems with different right hand sides $\mathbf{q}$ on each DFE is a straightforward extension but has not yet been attempted by the authors; currently only one accelerator is used. The DFE is operated via a compiled binary that is called from a high-level numerical computing environment using an automatically generated C intermediate layer.

Execution of the CGMN algorithm using the DFEs proceeds in three stages. First the matrix $H(\mathbf{m}, \omega)$ is copied to LMem. It will be read twice during each CGMN iteration (for each forward and backward sweep), but will not be modified. The second stage consists of the forward Kaczmarz sweep, during which the current CGMN search direction $\mathbf{p}$ is streamed to the FPGA from the CPU host, and the result $\text{FKSWP}(H, \mathbf{p}, 0, w)$ is stored to LMem. The third stage is the backward Kaczmarz sweep, which reads the results of the forward sweep from LMem and streams $\text{DKSWP}(H, \mathbf{p}, 0, w)$ back to the CPU. The CPU carries out the elementwise and reduction operations necessary, and stages 2 and 3 are repeated until CGMN converges.

If the rows of H are processed from 1 to N, each Kaczmarz iteration depends on the results of the last. Furthermore, each iteration computation takes many ticks of the FPGA clock. This is the latency L, and is chiefly due to the pairwise summation of values in $\langle a_i, \mathbf{x}_k \rangle$ (Equation 2). To avoid having to wait for L ticks between iterations, L independent iterations used to fill the computational pipeline. Two sources of parallelism can be exploited. First, multiple

Helmholtz problems sharing the same matrix H but different right hand sides $\mathbf{q}$ can be solved on the same DFE. Second, the order in which DKSWP accesses the rows of H can be changed such that consecutive Kaczmarz iterations update disjoint sets of iterate elements. At the present we use the latter method exclusively.

3 Preliminary Results

In Figure 2, computational times for 100 CGMN iterations running on the CPU + accelerator platform (Vectis model at 100 MHz) are compared with times on one Intel Xeon E5-2670 core. A speed-up of over 2× is seen when CGMN is run on the accelerator. At the current stage of development there are several factors that limit the performance of the accelerator, including inefficient use of the FPGA's memory controller. Once these are addressed, we expect a speed-up of 12×.

Speed-up of the pipelined accelerated Kaczmarz sweeps is further limited by the fact that the sweeps (when implemented on an Intel core) take up only about 95% of the total time used by CGMN. To see a more significant improvement we are implementing all of CGMN on the accelerator, from which we expect a speed-up of 32×. At that point the link between the FPGA and LMem that is used to stream the matrix H will become the bottleneck. To avoid this bandwidth limitation, the elements of the Helmholtz matrix will be generated on the FPGA from the earth model $\mathbf{m}$ as they are needed, rather than being read in from LMem.

4 Conclusion

We have demonstrated a time-harmonic iterative wave equation solver that off-loads its computational kernel onto a reconfigurable hardware accelerator. Work on this project is ongoing, and while further development is needed in order to better realize the potential for acceleration inherent in the platform, our preliminary results give us reason to be optimistic.

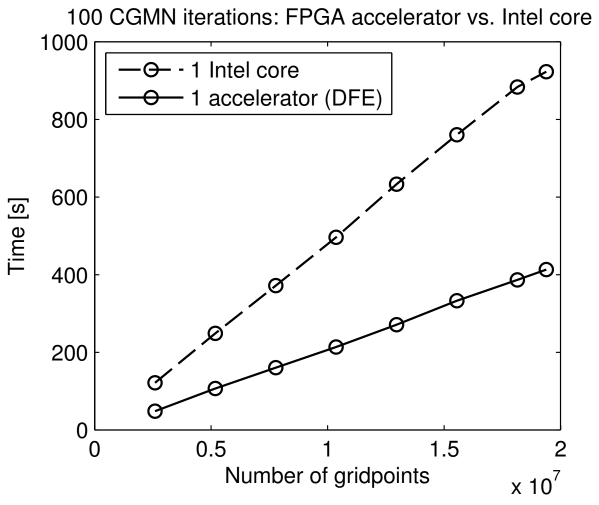

Figure 2: Comparison of one accelerator (DFE) with an Intel Xeon core. The system solved is one part of the SEG/EAGE Overthrust model [Aminzadeh et al., 1997].

5 Acknowledgements

We would like to thank Henryk Modzelewski for his support of the hardware and software used by the three authors affiliated with UBC, Eddie Hung for stimulating discussions about implementation, Rafael Lago for guidance about Krylov solvers, and Maxeler Technologies for allowing one of the authors to visit for a two-week period. This work was in part financially supported by the Natural Sciences and Engineering Research Council of Canada Discovery Grant (22R81254) and the Collaborative Research and Development Grant DNOISE II (375142-08). This research was carried out as part of the SINBAD II project with support from the following organizations: BG Group, BGP, BP, CGG, Chevron, ConocoPhillips, ION, Petrobras, PGS, Total SA, WesternGeco, and Woodside.

References

[Aminzadeh et al., 1997] Aminzadeh, F., B. Jean, and T. Kunz, 1997, 3-D salt and overthrust models: Society of Exploration Geophysicists.

[Björck and Elfving, 1979] Björck, Å., and T. Elfving, 1979, Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations: BIT Numerical Mathematics, **19**, 145–163.

[Elble et al., 2010] Elble, J. M., N. V. Sahinidis, and P. Vouzis, 2010, GPU computing with Kaczmarz's and other iterative algorithms for linear systems: Parallel Computing, **36**, 215–231.

[Kaczmarz, 1937] Kaczmarz, S., 1937, Angenäherte auflösung von systemen linearer gleichungen: Bulletin International de l'Academie Polonaise des Sciences et des Lettres, **35**, 355–357.

[Operto et al., 2007] Operto, S., J. Virieux, P. Amestoy, J.-Y. L'Excellent, L. Giraud, and H. B. H. Ali, 2007, 3D finite-difference frequency-domain modeling of visco-acoustic wave propagation using a massively parallel direct solver: A feasibility study: Geophysics, **72**, SM195–SM211.

[Pell et al., 2013] Pell, O., J. Bower, R. Dimond, O. Mencer, and M. J. Flynn, 2013, Finite-difference wave propagation modeling on special-purpose dataflow machines: Parallel and Distributed Systems, IEEE Transactions on, **24**, 906–915.