Accelerating an Iterative Helmholtz Solver with FPGAs

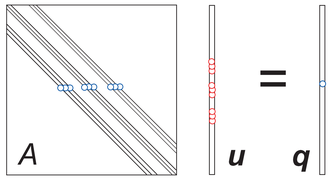
Art Petrenko[1], Tristan van Leeuwen[2], Diego Oriato[3], Simon Tilbury[3] and Felix J. Herrmann[1]

a place of mind
THE UNIVERSITY OF BRITISH COLUMBIA

Introduction

Seismic exploration aims to characterize the Earth's subsurface to assist the discovery and production of hydrocarbon resources. One algorithm used in this context is full-waveform inversion (FWI) (Herrmann et al. (2013)). FWI compares real wavefield data measured in a seismic survey with synthetic data generated by numerical wave simulations. Based on the difference, it updates its current guess at the subsurface parameters and repeats the process until convergence. The high computational burden of simulating seismic waves limits the number of FWI iterations that can be performed and hence the accuracy of the subsurface parameter estimate. Our contribution is acceleration of a wave equation solver via its implementation on a computing platform consisting of a CPU host and a reconfigurable hardware accelerator.

Solving the Wave Equation: Mathematical Background



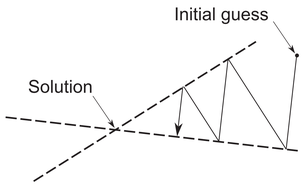
In the frequency domain, and specializing to the acoustic isotropic constant density case, simulating wave propagation corresponds to solving a large linear system, schematically illustrated at left. (Only 9 of 27 non-zero diagonals of A are shown).

$A = (\omega \boldsymbol{m} + \Delta)$ is the $N \times N$ Helmholtz operator that depends on the earth model $\boldsymbol{m}$ and the angular frequency ω.

$\boldsymbol{u}$ is the (complex) Fourier transform of the pressure wavefield, $\boldsymbol{q}$ is the time-harmonic source and Δ is the Laplacian.

We take $\boldsymbol{m}$ to be the squared slowness ($1/v^2$), and use a 27-point cube stencil to calculate the entries of A for the 3D case, as in Operto et al. (2007). A perfectly matched layer (PML) is used at the boundaries of the domain to eliminate reflection artifacts.

Because the matrix A 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 the wave equation above. However CG cannot be used directly because A is not symmetric positive semidefinite (SPSD, i.e. it has both positive and negative eigenvalues). Instead we use the Kaczmarz algorithm as a preconditioner to transform the wave equation into an equivalent system that does have these properties.



The Kaczmarz algorithm (Kaczmarz (1937)) is an iterative method for solving generic linear systems $\boldsymbol{Ax} = \boldsymbol{b}$ that projects its iterate $\boldsymbol{x}_k$ onto the hyperplane defined by a row $\boldsymbol{a}_i$ of the matrix A:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \lambda(b_i - \langle \boldsymbol{a}_i, \boldsymbol{x}_k \rangle)\, \boldsymbol{a}^*_i / \|\boldsymbol{a}_i\|^2$$

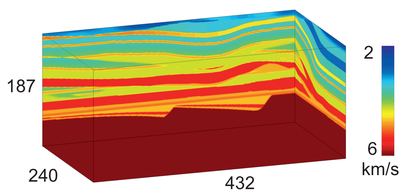
Here λ is a relaxation factor we set equal to 1.5, b_i is the ith element of the right-hand side, and $\langle \cdot, \cdot \rangle$ is the dot product. We refer to one application of the above equation as a Kaczmarz iteration, and to the projections onto each row from first to last as a forward Kaczmarz 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: $1 \rightarrow N$, $N \rightarrow 1$). The double Kaczmarz sweep can be represented in matrix form:

$$\text{DKSWP}(A, \boldsymbol{u}, \boldsymbol{q}, \lambda) = Q\boldsymbol{u} + R\boldsymbol{q}$$

The wave equation is transformed into an equivalent system for the fixed point of DKSWP:

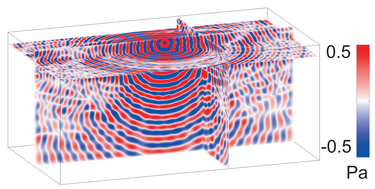
$$(I - Q)\boldsymbol{u} = R\boldsymbol{q}$$

This equivalent system is SPSD and is solved with CG, an algorithm known as CGMN and due to Björck and Elfving (1979). A parallel version of CGMN (dubbed CARP-CG) has recently been used by van Leeuwen et al. (2012).



Subsurface Model

The largest part of the SEG/EAGE overthrust velocity model (Aminzadeh (1997)) used in the experiments. The axes are labelled with the corresponding number of grid points.

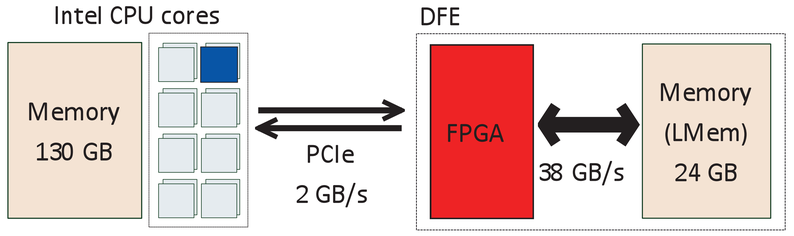


Example Wavefield

The real part of the Fourier transform of the pressure wavefield $\boldsymbol{u}$ that results from a time-harmonic point source. The solution shown took 2703 CGMN iterations (9826 s using one accelerator) to converge to a relative residual norm of 10^{-6}.

What are FPGAs?

Field-programmable gate arrays (FPGAs) are computer chips that consist of electronic building blocks that can be configured and re-configured by the user to represent algorithms in hardware. A conventional CPU can be thought of as a pipeline for processing instructions: the more complex the algorithm, the more instructions there are, the more clock ticks are required to execute it. In contrast, an FPGA can be thought of as a pipeline for processing data: 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 algorithm fits onto the FPGA, its complexity is irrelevant, since all the operations on an FPGA are hard-wired and happen in the same clock tick.



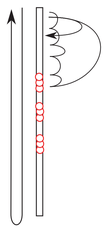
The target platform for our implementation is described in Pell et al. (2013) and shown above. A dataflow engine (DFE), consisting of an FPGA and its associated large memory, is connected via a PCIe bus to the CPU. The FPGA also has 4.6 MB of fast on-chip RAM (FMem). Although the machine the first author has access to has four DFEs, currently only one accelerator is used. The DFE is operated via a compiled binary that is called from a high-level numerical computing environment (MATLAB) using an automatically generated C intermediate layer. Related work has been done by Grüll et al. (2012), who implement an algorithm related to the Kaczmarz sweep (the simultaneous algebraic reconstruction technique, SART) on the same platform.

Dataflow During a Kaczmarz Row Projection

Execution of the CGMN algorithm using the DFEs proceeds in three stages. First the matrix $A(\boldsymbol{m}, \omega)$ 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 $\boldsymbol{p}$ is streamed to the FPGA from the CPU host, and the result FKSWP(A, $\boldsymbol{p}$, 0, λ) is stored to LMem. The third stage is the backward Kaczmarz sweep, which reads the results of the forward sweep from memory and streams DKSWP(A, $\boldsymbol{p}$, 0, λ) back to the CPU. The CPU carries out the element-wise and reduction operations necessary, and stages 2 and 3 are repeated until CGMN converges.

Overcoming Dependency of Sequential Projections

If the rows of A are processed from 1 to N, each Kaczmarz row projection depends on the results of the last. Furthermore, each projection's computation takes many ticks of the FPGA clock. This is the latency L, and is chiefly due to the pairwise summation of values in $\langle \boldsymbol{a}_i, \boldsymbol{x}_k \rangle$ in the Kaczmarz row projection equation. To avoid having to wait for L ticks between iterations, L independent iterations are used to fill the computational pipeline. Two sources of parallelism can be exploited. First, multiple Helmholtz problems sharing the same matrix A but different right-hand sides $\boldsymbol{q}$ can be solved on the same DFE. Second, the order in which DKSWP accesses the rows of A can be changed such that consecutive Kaczmarz iterations update disjoint sets of iterate elements. Access to the rows (and to the elements of the Kaczmarz iterate and right-hand side) is strided by 3 grid-points in the fast dimension, as illustrated at right. When the end of the fast dimension is reached the index wraps around to rows skipped the first time around. At the present we use the latter pipelining method exclusively.



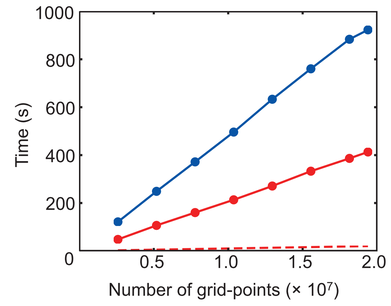
Current Challenges and Future Work

After acceleration, as shown in the Results section, the Kaczmarz sweeps are no longer the hot-spot of CGMN. A large amount of time is spent in MATLAB pre-processing the complex Kaczmarz iterate. To speed up CGMN further, we will implement all of it on the accelerator. This has the added benefit of off-loading the computational complexity of CGMN from the CPU to the accelerator, which decreases the CPU execution time, but does not increase the execution time on the FPGA, since the amount of data to process (problem size) remains the same.

At an FPGA operational frequency of 145 MHz, CGMN-on-the-chip will be bandwidth-limited by the link to the DFE's large dedicated memory, used to stream the matrix A. To avoid this bandwidth limitation, the elements of the Helmholtz matrix will be generated on the FPGA from the earth model $\boldsymbol{m}$ as they are needed, rather than being read in from memory. We expect a speed-up of 42× compared to a single Intel core from the CGMN-on-the-chip, matrix-on-the-fly version.

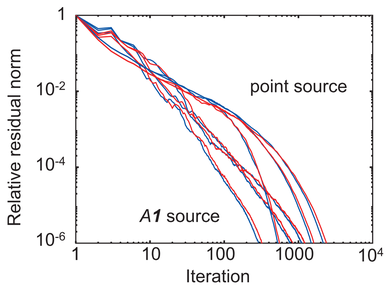
Further work will include domain decomposition to enable solution of large systems. The 4.6 MB on-chip memory limits the domain block size to approximately 300×300 in the two fast dimensions. Coarse-grained parallelization by solving problems with different right-hand sides $\boldsymbol{q}$ on each DFE is another straight-forward extension.

Results



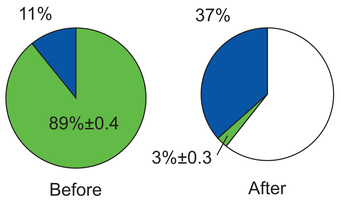
Time for 100 Iterations

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. The dashed red line represents the theoretical maximum performance of the accelerator, an improvement of more than 16× relative to the current version of the accelerated kernel.



Iterations to a Specified Tolerance

To overcome the latency of each individual row projection, elements of the Kaczmarz iterate must be accessed out of order when the Kaczmarz sweeps are run on the accelerator. This has an effect on the number of iterations required for CGMN to converge. The figure at left compares the two different access orders discussed: sequential (1 to N) and strided by three grid-points along the fast dimension.
Convergence histories of three different system sizes are shown: 432×240×{25,100,187}, and two different right-hand sides:
• a point-source (smooth down-curving lines)
• a source equal to $A\boldsymbol{x}$, where $\boldsymbol{x}$ is a vector of ones (jagged lines).
The difference between sources far outweighs the difference due to iterate element access order.



Where Time is Spent

The effect on computational time of porting the Kaczmarz sweeps to the CPU + accelerator platform is broken up into two parts:
• The time for the sweeps is reduced compared to the implementation on one Intel core.
• The time spent in the rest of CGMN is increased due to having to interleave real and imaginary parts of vectors for the accelerator.
The proportions are consistent across different system sizes, an average over sizes is shown.

Conclusions

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.

References

Aminzadeh, F., Jean, B. and Kunz, T. [1997] 3-D Salt and Overthrust Models. Society of Exploration Geophysicists.

Björck, A and Elfving, T. [1979] Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. BIT Numerical Mathematics, 19(2):145–163. ISSN 0006-3835. doi: 10.1007/BF01930845.

Grüll, F., Kunz, M., Hausmann, M. and Kebschull, U. [2012] An implementation of 3D electron tomography on FPGAs. Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on, 1–5, doi: 10.1109/ReConFig.2012.6416732.

Herrmann, F. J., Calvert, A. J., Hanlon, I., Javanmehri, M., Kumar, R., van Leeuwen, T., Li, X., Smithyman, B., Takougang, E.T. and Wason, H. [2013] Frugal full-waveform inversion: from theory to a practical algorithm. The Leading Edge, 32(9):1082–1092, 09. doi: http://dx.doi.org/10.1190/tle32091082.1.

Kaczmarz, S. [1937] Angenäherte Auflösung von Systemen linearer Gleichungen. Bulletin international de l'academie polonaise des sciences et des lettres, 35, 355–357.

Operto, S., Virieux, J., Amestoy, P., L'Excellent, J.Y., Giraud, L. and Ali, H.B.H. [2007] 3D Finite-difference frequency-domain modeling of visco-acoustic wave propagation using a massively parallel direct solver: A feasibility study. Geophysics, 72(5), SM195–SM211, doi:10.1190/1.2759835.

Pell, O., Bower, J., Dimond, R., Mencer, O. and Flynn, M.J. [2013] Finite-difference wave propagation modeling on special-purpose dataflow machines. Parallel and Distributed Systems, IEEE Transactions on, 24(5), 906– 915, ISSN 1045-9219, doi:10.1109/TPDS.2012.198.

van Leeuwen, T., Gordon, D., Gordon, R. and Herrmann, F. J. [2012] Preconditioning the Helmholtz equation via row-projections. In EAGE technical program. EAGE, 01.

Acknowledgements

We would like to thank Henryk Modzelewski, Eddie Hung, Rafael Lago, Brendan Smithyman, and everyone at Maxeler Technologies. 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, Statoil, Total SA, WesternGeco, and Woodside.

1: Department of Earth, Ocean and Atmospheric Sciences, University of British Columbia
2: Centrum Wiskunde & Informatica, formerly with UBC
3: Maxeler Technologies

apetrenko@eos.ubc.ca