ROME 2008
LEVERAGING TECHNOLOGY

P055

Accelerating Seismic Computations on FPGAs From the Perspective of Number Representations

H. Fu* (Imperial College London), W. Osborne (Imperial College London), R.G. Clapp (Stanford University) & O. Pell (Maxeler Technologies)

SUMMARY

Conventional seismic processing is performed on a CPU with 32 or 64-bit precision for all operations. In certain cases, using a reduced precision produces equivalent result within acceptable tolerances. However, as CPUs do not support configurable bit-widths, reducing precision brings no benefits to performance. In contrast, Field Programmable Gate Arrays (FPGA) enable application specific number representations. With hardware support for reconfigurable number format and bit-width, reduced precision can greatly decrease the area cost and I/O bandwidth of the design, thus multiplying the performance with concurrent processing cores on an FPGA. In this paper, we present a tool to determine the most appropriate number format for a given seismic application. We demonstrate the methodology on a source-receiver downward continuous based migration. The acceleration device is a Maxeler Technologies MAX-1 card containing a Xilinx Virtex-4 FPGA and connects to a PC over a PCI Express bus. Using optimized number representations, we can implement two concurrent processing cores on the FPGA, and achieve a speedup of 14 times compared to a Intel Xeon 1.86GHz CPU. Moreover, with sufficient bandwidth between the CPU and FPGA, we show that a further increase to 48x speedup is possible.

Introduction

Reducing the time for converting collected data into seismic images is becoming critical for the production cycle of the oil industry. Compared to common processors, reconfigurable computing devices, such as Field Programmable Gate Arrays (FPGA), provide another architecture with inherent parallelism in both computation and data I/O, and have achieved up to 20 times acceleration of pre-stack Kirchhoff time migration (He, C. et al. 2004) and 40 times for subsurface offset gathers (Pell, O., and Clapp, R.G. 2007).

Conventional seismic processing is performed on a CPU with 32 or 64-bit precision for all operations. In certain cases, using a reduced precision produces equivalent result within acceptable tolerances. However, as CPUs do not support configurable bit-widths, reducing precision brings no benefit on performance. In contrast, FPGAs enable application specific number representations. With hardware support for reconfigurable number format and bit-width, reduced precision can greatly decrease the area cost and I/O bandwidth of the design, thus multiplying the performance with concurrent processing cores on an FPGA. In this paper, we propose a tool to determine the most appropriate number format for seismic applications. We also demonstrate the acceleration achieved with optimized number representation with a FPGA design for complex exponential calculation in downward continued based migration.

Background

FPGAs support reconfigurable computation units and connection routes between them and can use arbitrary number formats. In this paper, we focus on the two most commonly used number representations, fixed-point and floating-point numbers.

- A fixed-point number consists of an integer part and a fractional part, shown as follows:

Integer part: m bits	Fractional part: f bits
$x_{m-1}x_{m-2}\cdots x_0$	$x_{-1}x_{-2}\cdots x_{-f}$

  When it uses a sign-magnitude format, its value is given by $(-1)^{x_{m-1}}\sum_{i=-f}^{m-2} x_i \cdot 2^i$.

- According to IEEE-754 standard, a floating-point number consists of three parts, the sign bit, the exponent, and the significand, shown as follows:

Sign: 1 bit	Exponent part: m bits	Significand part: f bits
S	M	F

  The significand part is an unsigned fractional number, with an implied '1' to the left of the radix point. The exponent uses a biased format, which represents the sum of the actual value and the bias value $2^{m-1}-1$. Thus, the value of a floating-point number is given by $(-1)^S \cdot 1.F \cdot 2^{M-2^{m-1}+1}$.

Method

Our tool currently works on seismic programs written in Fortran. Firstly, we manually partition the Fortran program into the part remaining in software and the part to be moved onto FPGA (target code). The partition is based on two metrics: (1) the target code shall consume a large portion of processing time in the entire program, otherwise the acceleration does not bring enough performance improvement to the entire application; (2) the target code shall be suitable for a streaming implementation on FPGA, thus highly probable to accelerate.

After partition, the next step is to profile the target code, and acquire information about the ranges and distributions of the variables. Based on the range information, we map the target code into a hardware design with configurable number format and bit-widths. The design is described using A Stream Compiler, ASC (Mencer, O. 2006). As a high-level hardware programming tool, ASC supports basic arithmetic units for fixed-point and floating-point

numbers with configurable bit-widths. The ASC description is then translated into bit-accurate simulation code, and merged into the original Fortran program to provide a value simulator for the original application. This value simulator enables us to perform explorations with configurable settings such as different number representations and different bit-widths.

Besides basic arithmetic operations, evaluation of elementary functions (such as square root and trigonometric functions) takes a large part in seismic applications. To map these functions into efficient units on the FPGA board, we use a table-based polynomial approximation approach (Lee, D. et al. 2005) to generate optimized hardware function evaluation units.

To ensure that designs with reduced precision still produce equivalent results within acceptable tolerances, the exploration of number formats requires an automated mechanism to evaluate the accuracy of generated seismic images. To judge whether a generated image contains accurate pattern information, we compare it to a 'true' image, which is calculated with a double-precision implementation of the target code and contains the correct image pattern. To perform a pattern comparison automatically, we use techniques based on Predication Error Filters (PEFs) (Claerbout, J. 19949) to highlight differences between two images. The basic workflow to compare image *A* (the 'true' image) with image *B* is as follows:

1. Divide image *A* into overlapping small regions of 40x40 pixels, and estimate PEFs for these small regions.
2. Apply these PEFs to both image *A* and image *B* to get the results *A'* and *B'*.
3. Apply algebraic combinations of the images *A'* and *B'* to acquire a value indicating the image differences.

By the end of the above workflow, we achieve a single value which describes the difference from the generated image to the 'true' image. For convenience of discussion, we describe this value as 'Difference Indicator' (DI).

Case Study: Complex Exponential in Downward Continued Based Migration

Downward continued migration comes in various flavours including common azimuth migration, shot profile migration, source-receiver migration, planewave or delayed shot migration, and narrow azimuth migration. The different techniques have varying cost profiles but all share two meaningful computational bottlenecks: transforming to-and-from the wavenumber domain (FFT) and applying the Single Square Root (SSR) or Double Square Root (DSR) condition (the complex exponential step). In this case study, we use to our tool to analyze the most appropriate number formats for the complex exponential step.

The governing computation for the complex exponential step is the DSR Equation, which can be written as:

$$U(w,k_s,k_g,z+\Delta z) = \exp[-iwv(\sqrt{1-\frac{vk_g}{w}}+\sqrt{1-\frac{vk_s}{w}})]\cdot U(w,k_s,k_g,z) \qquad (1)$$

where w is frequency. The implementation takes the approach of building a priori a relatively small table of the possible values of vk/w, then performing a table lookup that converts a given vk/w value to an approximate value of the square root. The entire design consists of three parts, square root calculation, sine/cosine evaluation, and complex multiplication.

In exploration of fixed-point formats, as the range of variables in the three parts are quite different from each other, we apply a different bit-width in each part, specified as SQRT (square root), SINE (sine/cosine evaluation) and WMUL (wave-field complex multiplication) bit-width respectively. We keep two of the bit-widths constant, and change the other bit-width gradually to find out its effect on the accuracy of the entire application. Figure 1(a) shows an example where we keep SINE and WMUL bit-widths as 20 and 30, and change the SQRT bit-width from 6 to 20. Large bit-widths are used for the other two parts so that they do not contribute much to the errors and the effect of changes in SQRT bit-width can be extracted. In our example, when the SQRT bit-width increases to 10 bits, the DI value shows a clear

change from 10^5 to 10^2. The fast improvement in accuracy is also demonstrated in the generated seismic images. The image generated with 8-bit design contains totally different patterns, while the image from 10-bit design already shows the same pattern as the 'true' image. Thus, a suggested value for the SQRT bit-width is 10.

Applying a similar method to other parts, we achieve the suggested values 12 and 16 for SINE and WMUL bit-widths. The suggested values are acquired with two of the three bit-widths kept very large. Thus, when we combine them together, we need to slightly increase the suggested values to avoid accuracy degradation. Using these values as a starting point, we use our tool to automatically enumerate nearby cases, and find out the minimum bit-widths to provide acceptable image quality are 12, 16, 16 for SQRT, SINE and WMUL.

For floating-point designs, we explore combinations of different exponent and significand bit-widths. Similarly, when we investigate one of them, we keep the other one with a constant high value. In the example shown in Figure 1(b), we use a significand bit-width of 24, and gradually increase the exponent bit-width from 3 to 10. There is also a clear change at a bit-width value of 6, where the DI value drops from 10^5 to 10. Using these values as initial guess, we again apply enumerations for values nearby, and determine that the minimum exponent and significand bit-widths are 6 and 16 for a floating-point design.

Based on the exploration results, the fixed-point design with bit-widths of 12, 16, and 16 for three different parts is selected as our hardware implementation. The design produces images with the same patterns as the double-precision floating-point software implementation, and consumes the least cost on a Maxeler Technologies MAX-1 Card with Xilinx Virtex-4 FX100 FPGA. Under the constraint of the PCI Express X8 (2 Gigabytes per second) bandwidth between CPU and FPGA, we can currently support two concurrent computation cores and achieve 13.7 times speedup compared to the software implementation on Intel Xeon 1.86GHz CPU. With sufficient communication bandwidth, we can put 6 concurrent cores on the FPGA with the minimized bit-width, and achieve up to 48 times speedup.

Conclusion

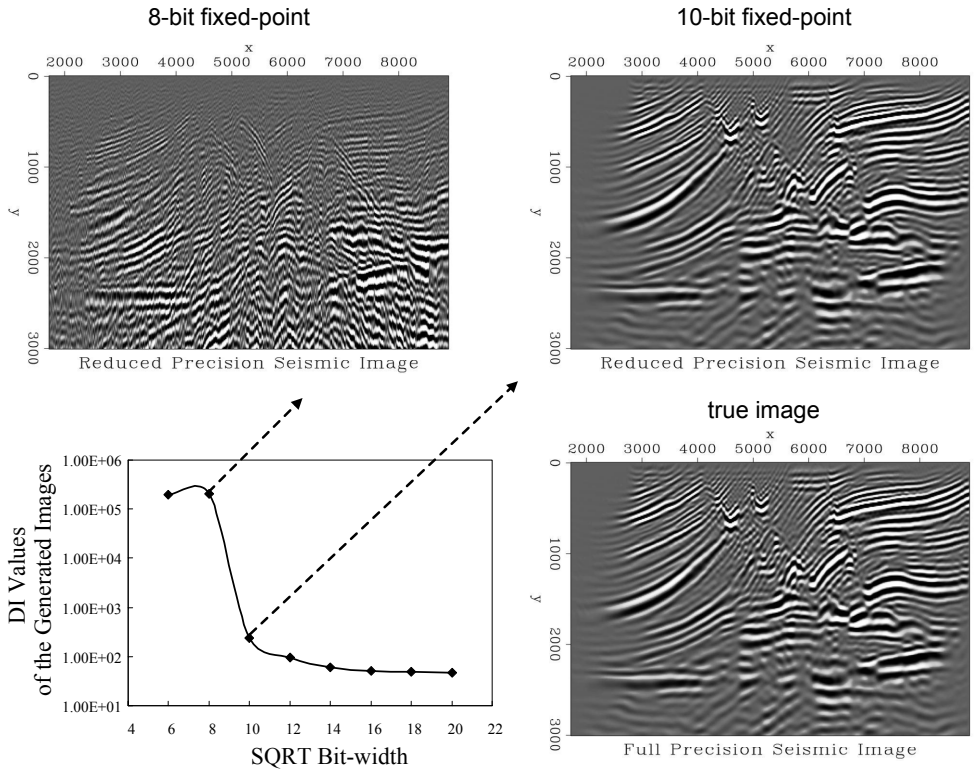
This paper presents our work on accelerating seismic applications on FPGAs. We present a tool to determine the most appropriate number format for a given seismic application. We demonstrate the methodology on a source-receiver downward continuous based migration. Using optimized number representations, we greatly reduce the area cost and communication bandwidth of the application, and achieve up to 13.7 times speed up with two concurrent processing cores on FPGA. Moreover, with sufficient bandwidth between the CPU and FPGA, we can achieve a further increase to 48x speedup with 6 concurrent cores.

Acknowledgements

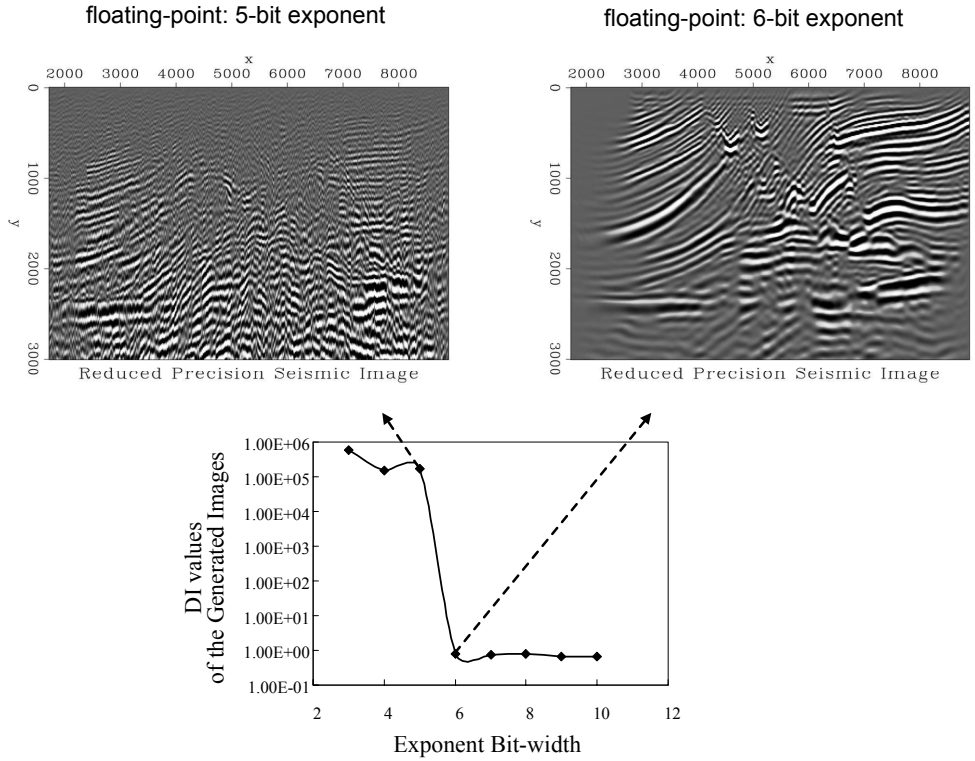
The support from the Center for Computational Earth and Environmental Science, Stanford Exploration Project, and Maxeler Technologies is gratefully acknowledged. We also thank Professor Martin Morf and Professor Michael Flynn for their support and advice.

References

He, C., Lu, M., and Sun, C. [2004] Accelerating Seismic Migration Using FPGA-based Coprocessor Platform. Proc. FCCM, 207-216.
Pell, O., and Clapp, R.G. [2007] Accelerating Subsurface Offset Gathers for 3D Seismic Applications using FPGAs. Society of Exploration Geophysicists, Extended Abstract.
Mencer, O. [2006] ASC, A Stream Compiler for Computing with FPGAs. IEEE Transactions on CAD, 25(9), 1603-1617.
Lee, D., Gaffar, A.A., Mencer, O., and Luk, W. [2005] Optimizing Hardware Function Evaluation. IEEE Transactions on Computer, 54(12), 1520-1531.
Claerbout, J. [1999] Geophysical Estimation by Example: Environmental Soundings Image Enhancement: Stanford Exploration Project. http://sepwww.stanford.edu/sep/prof/.
Claerbout, J. [2000] Basic Earth Imaging (BEI). http://sepwww.stanford.edu/sep/prof/.



(a) DI values for different SQRT bit-widths in a fixed-point design.



(b) DI values for different exponent bit-widths in a floating-point design.

Figure 1. Exploration of fixed-point and floating-point designs with different bit-widths.