Fast 3D ZO CRS Stack

MAXELER Technologies

This is a summary of a paper presented at the 72nd European Association of Geoscientists and Engineers (EAGE) Conference, Barcelona, Spain, June 2010.

About the Authors

Paulo Marchetti is a scientist at ENI S.p.A. Exploration and Production Division, AESI Department. Oliver Pell is VP of Engineering at Maxeler Technologies. Diego Oriato is an acceleration architect at Maxeler Technologies.
Antonio Cristini and Daniela Theis are researchers in Imaging and Numerical Geophysics at CRS4.

Summary

This paper presents the results of accelerating the 3D-ZO-CRS stacking algorithm using a MAX2 FPGA accelerator card. The MAX2 connects to the CPU via PCI-Express and has 24GB of DRAM on board.
The purpose of the CRS algorithm is to obtain a non-migrated image from pre-stack seismic data. It does this by solving an optimization problem using conjugate directions to maximize the Semblance function within an 8 dimensional parameter space.
The computation time of the CRS application is dominated by the evaluation of the Semblance function (see *Figure 1*). The Semblance function takes a window of 16 complex data items out of each trace (in a location determined by the position in the parameter space) and performs computation with them. In a typical example we may have over 100,000 traces with 1500 complex numbers in each trace. The algorithm computes up to 18 Semblance functions on each iteration of the conjugate directions method and performs this process on 750 different starting values for the search.

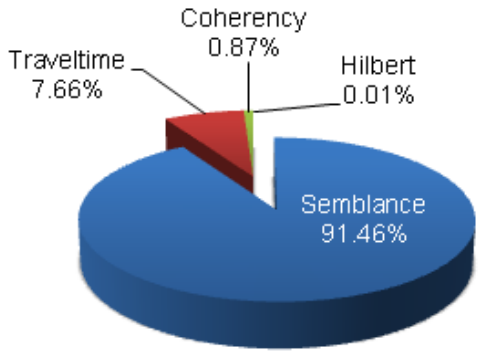

Figure 1: Percentage of computation time taken in each function in the original software.

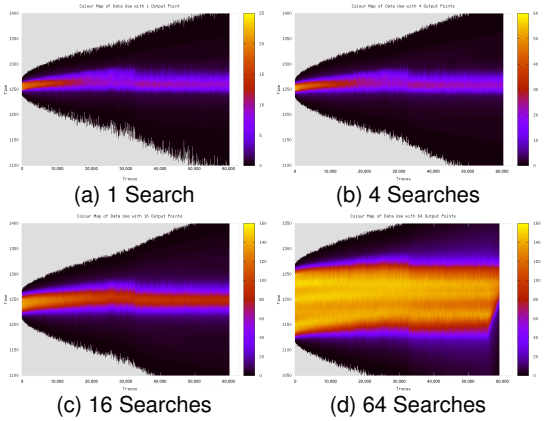

Figure 2: Usage of an input datum in CRS optimization, when computing 1, 4, 16 and 64 t0 output points.

The extra computation capacity of an FPGA versus a CPU means that in its original form the application would have become bound in its performance by the DRAM bandwidth of the MAX2, and thus the key to accelerating CRS is to reuse each piece of data that is streamed on to the FPGA multiple times. To do this, we perform multiple searches in parallel and store a selection of the current trace in a cache on the FPGA that can be accessed by each of the searches. The color maps in *Figure 2* show data reuse for different numbers of parallel searches. With 64 parallel searches the average data reuse is high enough that we are no longer memory bound.
At the start of the application the trace data is loaded from disk and loaded to the DRAM on the MAX2. After this point the application communicates with the FPGA via PCI-Express, allowing parameter values to be streamed into the FPGA, which then in turn uses these to compute Traveltimes (window locations) and the Semblance values. The values computed by the Traveltime kernel determine which section of the trace must be streamed from DRAM into the Semblance kernel. At the end of a single run of the FPGA it streams out Semblance values to the CPU.
With both Semblance and Traveltimes computation on the FPGA, over 99% of the total runtime from the CPU is accelerated. The final implementation gives a speedup of more than 200x compared to the FORTRAN software running on one CPU core.