FD Wave Propogation Modeling on Special Purpose Dataflow Machines

MAXELER Technologies

This is a paper summer published in IEEE Transactions on Parallel and Distributed Systems, May 2013.

About the Authors

Oliver Pell is VP of HPC Solutions, Jacob Bower is VP of Acceleration Engineering, Robert Dimond is an Engineer, Oskar Mencer is CEO and Michael J. Flynn is Chairman at Maxeler Technologies and Professor Emeritus of Electrical Engineering at Stanford University.

Summary

Finite difference is a numerical method used to solve PDEs such as the wave equation. While finite difference has been highly optimized on conventional CPUs, it still has memory requirements in the hundreds of gigabytes and runtimes of weeks on thousands of CPUs. Our dataflow computing platform represents an evolution of concepts of dataflow computers which has the potential to meet these increasing memory and performance demands.

Simulation of wave propagation involves some of the most computationally intensive geoscience algorithms. When creating an image, a low frequency acoustic source is activated, reflected and recorded by thousands of receivers. The resulting dataset is tens or hundreds of terabytes.

Our hardware architecture consists of conventional CPUs coupled with dataflow engines (DFEs). An accelerated cluster is made up of special purpose compute nodes. Inside a node, CPUs are accompanied by DFEs, where each DFE has a direct interconnect to its neighbours called a MaxRing. A typical 1U configuration delivers 15-50 times the performance of a standard multicore x86 server for practical applications.

Our programming environment, MaxGenFD, is a domain specific compiler for 3D finite-difference applications. MaxGenFD handles complexities such as managing large datasets and boundary conditions. When MaxGenFD accelerates an application, the programmer splits the application into three components: the FDKernel, the FDKernel Configuration, and the Runtime. The MaxGenFD compilation process changes the user's program into a dataflow graph, which is then transformed into an application specific DFE configuration. MaxGenFD also constructs a Manager, which determines how the application kernel is interconnected to the PCI Express, DRAM and MaxRing controllers. The next stage of compilation is executed by the MaxCompiler, which compiles the kernel dataflow graph and manager graph into a chip configuration. Once compiled, the FDKernel is implemented by the accelerator chip as a function where I/O is configured at runtime.

Three levels of parallelization are utilized in MaxGenFD. The first is across multiple nodes where parallelization occurs across multiple shots. The next is across multiple DFEs in a single chip. Multiple chips can collaborate on a single shot, as shown in Figure 1. The final level of parallelization is within a single chip, where MaxGenFD can replicate the dataflow graph until the entirety of the chip's resources are exhausted. The amount of data required for finite difference modeling ranges from gigabytes to hundreds of gigabytes, ultimately resulting in the performance limiting factor on a conventional CPU. Memory requirements for the algorithm are addressed by compressing data to be stored in memory. Optimizing data layout to minimize access cost is also employed.

A GPU chip has a peak performance similar to chips used to implement DFEs; however, in real applications, the DFE vastly outperforms the GPU. Typical performance gains of a 1U node with 2 GPUs over a conventional CPU is eight to ten times, compared to 70 times for a 1U node with 4 DFEs.

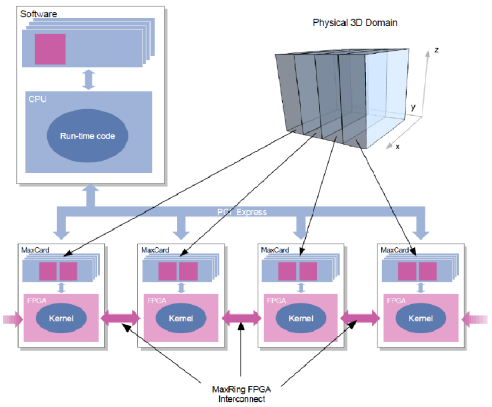

Figure 1: One-dimensional decomposition of the problem domain to parallelize across multiple DFEs linked with MaxRing