

Design space analysis for the acoustic wave equation implementation on FPGA MAXELER Technologies

This is a summary of a paper presented at the 70th European Association of Geoscientists and Engineers (EAGE) Conference, Rome, Italy, June 2008.

About the Authors

Tamas Nemeth and Joe Stefani are Geophysicists at Chevron.

Oliver Pell is VP of Engineering at Maxeler Technologies.

Ray Ergas is a Geophysicist, formerly at Chevron.

Summary

Seismic modeling and data processing operates on gigabyte to terabyte-size datasets and involves large CPU clusters running applications for extended periods of time. Accelerating these applications would make a big impact and the emerging new co-processor hardware architectures need to be carefully tested for this purpose.

However, algorithms typically fail to exploit the strengths of such massively parallel stream processors. To illustrate the algorithm redesign that accompanies a transition from x86-64 to FPGA, we consider the problem of 3D acoustic forward modeling.

The run-time of the x86-64 implementation is dominated by a small, computationally intensive kernel. This kernel performs many independent math operations per memory access, indicating a potential for profitable FPGA reimplementations. It consists of 5 components, every one of which must be accelerated to achieve the intended 100x performance improvement (see [Figure 1](#)). We assume that all data will reside in DRAM on the FPGA card and evaluate alternatives in terms of how they use the platform's computational capability, memory capacity, and memory bandwidth.

Option 1 is a straightforward translation of the x86-64 algorithm. The software kernel performs six 11pt uni-axial convolutions, two in each of the X, Y, and Z axes. This incurs six passes through the dataset, each of which uses so little FPGA mem-

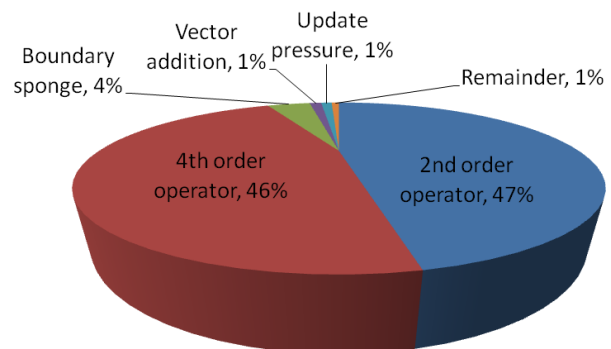


Figure 1: 5 sub-components of acoustic modeling kernel.

ory and is so computationally inexpensive that the primary cost is that of repeatedly reading the dataset into the FPGA from DRAM.

Option 2 replaces three uni-axial convolution operators with a single 23pt tri-axial operator that is applied in two passes through the dataset. The larger operator increases the amount of computation per memory access, alleviating the bottleneck of option 1, but it also exceeds the caching capacity of the FPGA and therefore requires expensive decomposition of the dataset.

Option 3 uses less FPGA internal memory by replacing the 23pt operator from option 2 with a 12pt operator applied in four passes through the dataset. Here the domain decomposition overhead is reduced but additional DRAM is required to store the results of intermediate convolutions.

Option 4 makes three passes through the dataset. On each pass, two uni-axial operators, as in option 1, are applied. The two large-stride Z-axis convolutions are done together to reduce the amount of FPGA internal memory required. This makes domain decomposition unnecessary while maintaining a high ratio of computation to memory access (see [Table 1](#)).

Careful evaluation of the possible software implementations to discover these tradeoffs in compute capability, memory capacity and memory bandwidth is crucial to providing significant acceleration using FPGA technology.

Option	Operator	Passes	CPU FLOPS	FPGA FLOPS	Temp. Arrays	FPGA memory req.
1	1D, 11pt	6	32	31	1	Low
2	3D, 23pt	2	200+	93	1	Very high
3	3D, 12 pt	4	54	51	4	High
4	Pseudo-2D, 11 pt	3	-	59	1	Moderate

Table 1: Summary of the four different algorithm implementation options.