An Optimized Finite Difference Computing Engine on FPGAs

Conference Paper - May 2006
DOI: 10.1109/FCCM.2006.24 - Source: IEEE Xplore

4 authors, including:

Guan Qin
University of Houston
64 PUBLICATIONS  954 CITATIONS

All content following this page was uploaded by Guan Qin on 27 May 2014.
The user has requested enhancement of the downloaded file.
An Optimized Finite Difference Computing Engine on FPGAs

Chuan He, Guan Qin, Mi Lu, and Wei Zhao
Texas A&M University, College Station, TX 77843
chuanhe@ee.tamu.edu, guan.qin@tamu.edu, mlu@ee.tamu.edu, w-zhao@tamu.edu

1. Introduction

Time domain or frequency domain Finite Difference (FD) methods are one of the most popular numerical
modelling techniques in the solution of scientific and engineering problems. However, these simulations are
still time-consuming and cannot be used routinely except in institutes that can afford the high cost of running and
maintaining supercomputers or large PC-cluster systems.

In this paper, we present an efficient implementation
of FPGA-based FD computing engine using acoustic
wave modeling problems as an example. Instead of
following the formal high-order FD expressions with
standard IEEE-754 compliant floating-point multipliers
and adders, we propose a new class of optimized FD
schemes, whose FD coefficients are optimized to be only
a few binary bits so that much fewer Logic Cell (LC)
resources or on-chip multipliers are needed without
deteriorating numerical accuracy criteria. Furthermore,
we simplify the implementation of following floating-
point summations by group-alignment technology. A
floating-point/fixed-point hybrid accumulator with similar
relative and absolute rounding errors now replaces the
conventional costly floating-point adder tree.

2. Optimized FD Schemes on RC Platform

The acoustic wave equation is represented in the form
of second-order Partial Differential Equations (PDEs) as:
\[
\frac{\partial^2 P(x,z,t)}{\partial t^2} - v^2(x,z) \Delta P(x,z,t) = f(x,z,t)
\] (2.1)

where \( P \) is pressure, \( v \) is acoustic velocity, and
\( \Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2} \), stands for the spatial Laplace operator.

By Taylor expansion, the spatial Laplace operator can be approximated systematically to \((2m)^{\text{th}}\) accurate
order by central \((2m+1)\)-point stencil along each spatial
axis [1] as follows,

\[
\frac{\partial^2 P}{\partial x^2}
\bigg|_x = \left[ \alpha_0^x \cdot P_{x} + \sum_{m=1}^{\infty} \alpha_m^x \cdot (P_{x+m} + P_{x-m}) \right] / (dx)^2 + O((dx)^n)
\] (2.2)

where \( \alpha_0^x = -2 \sum_{m=1}^{\infty} (-1)^{m-1} \frac{2m^2}{r!(m-r)!(m+r)!} \) (2.3)

and \( \alpha_m^x = (-1)^{m-1} \frac{2m^2}{r!(m-r)!(m+r)!} \) (2.4)

are chosen to maximize the order of the un-cancelled
Taylor expansion term.

This class of FD schemes provide excessive accuracy
within low wave-number band at the cost of rapidly-
worsen performance in high wave-number band. To
compensate this disadvantage, another approach is used
here to improve numerical dispersion relations rather than
pursue formal accuracy representations. This class of
methods also utilize central \((2m+1)\)-point stencil to
approximate each spatial second derivative term as
before. By dropping the requirement of maximizing the
order of un-cancelled truncation term, we now need only
the least nominal accuracy order to keep our FD schemes
numerically consistent to the original equation. All rest
FD coefficients in Equation (2.4) are chosen to minimize
the square of phase speed error of the discretized wave
equation inside its working wave-number band, which
leads to a frequency domain Least Square (LS) error
criterion.

Keeping the LS criterion in mind, we further
optimized these FD coefficients so that they can be
represented with only a few binary bits without
deteriorating numerical criterions. The advantage is
obvious: For FPGA devices without hardware multipliers,
fewer partial sums would lead to less occupation of LC
resources; FPGA devices with on-chip multipliers can
also benefit from this approach because of the reduction
of latency. In our design, the largest two coefficients in
the middle of FD stencils are represented in 18-bit fixed-
point format because their values are pivotal to the
numerical performance. (The number of 18 is chosen
because of the word-width of on-chip multipliers inside
Xilinx FPGAs.) For all other coefficients, we represent
them in floating-point format with only six mantissa bits
so that on-chip multipliers can always finish corresponding multiplications within one clock cycle.
Expressing exponents of these coefficients explicitly are
not necessary because underlying exponent adjustment
can be easily implemented by adding a constant to original exponents of the multiplicands.

3. Group-alignment based FP Summation

Floating-point summation is prone to so-called “catastrophic cancellations”. Also the sequence of additions will affect the final sum, which makes it impossible to produce a unique solution for the same input data set [2]. A group-alignment based summation unit is designed to accumulate those floating-point products produced by coefficient. This new approach results in similar or much better worst-case absolute and relative errors as the standard floating-point arithmetic with only a fraction of hardware resources consumed. Also the total number of pipeline stages required for the new FD computing engine is reduced significantly.

![Group-alignment based FP accumulator](image)

Figure 1. Group-alignment based FP accumulator

The basic idea of this new hardware-based algorithm is pretty straightforward: Instead of the original floating-point adder tree structure, where comparisons of exponents and additions of shifted mantissas are scattered in different adders, we now collect them up to form an exponent comparison tree in Step 2 and a fixed-point adder tree in Step 5. The hardware correspondences for other steps like 1, 3, and 4 are also clustered together so that each function step results in only one pipelining latency. The advantage of this arrangement is obvious: We now don’t have to round or normalize every intermediate summation result at the final stages of each floating-point adder but only one step at the end of this large summation unit. Correspondingly, we can save almost half of reconfigurable hardware resources and also the total number of pipeline stages is reduced significantly. Further optimization of this design is possible if we choose to make use of some new features of up-to-date FPGA chips.

4. Implementation on FPGA

As an example, the 9-point finite-accurate FD computing engine is implemented on an entry-level Virtex II Pro XUPV2P evaluation board [19]. The software development environments we used in this design are Xilinx ISE 7.1i and ModelSim 6.0 se.

A fully pipelined implementation of Equation (2.2) based on standard single-precision floating-point arithmetic will cost us 20 embedded 18X18 multipliers and over 3,300 Logic Slices. The total number of pipeline stages is about 50. Our new implementation costs only 6 embedded 18X18 multipliers and about 1560 Logic Slices, which corresponds to less than 10% of available RC resources on our evaluation board. Furthermore, the regular layout and localized interconnections of our design lead to much higher computational throughput especially for FD schemes with wide stencils. The clock rate applied to the computing engine is 167MHz for 13 pipeline stages or 200MHz for 15. Higher clock rate can also be achieved if more pipeline stages are introduced. Simplicity and scalability are other desirable properties of our design. Choosing more fraction bits into the new summation unit consumes negligible extra RC resources, but can significantly improve numerical error bounds.

![Structure of 1D Laplace operator for 8th-order finite-accurate optimized scheme](image)

Figure 2. Structure of 1D Laplace operator for 8th-order finite-accurate optimized scheme

References