Seismic Modeling and RTM Migration on unconventional Hardware

de Bragança, R.S.N. and Silva, R.W.G. PETROBRAS, Lima, Manoel Eusebio et al. UFPe

Abstract

FPGA's (Field Programmable Gate Arrays) are silicon chips with billions of transistors like current microprocessor chips but, unlike these, all the interconnection of its internal logic elements are completely reconfigurable. This unique characteristic allows a user to instantiate electronic circuits inside an FPGA by configuring the interconnection in order to execute a specific computational task. The inherent parallel nature of the circuits thus allow the FPGA to perform such task several times faster than current CPU's, making them suitable for high performance computing. In this work FPGA's are configured to perform seismic acoustic 2D modeling and 2D acoustic RTM migration. The computational performance and energy efficiency are then compared with the traditional CPU approach.

Introduction

Seismic imaging in geologically complex areas, especially in sub-salt regions, demands the application of algorithms that require high computational power, such as RTM migration. The traditional approach is the use of large clusters of computers and, more recently, the use of the GPU's for processing. The goal of this work is to study a new technology based on FPGA chip's that demonstrates promising for implementing these types of algorithms which may, in a proper hardware, to achieve superior performance than GPU's and with the advantage of doing it with less energy consumption.

The kernel of the acoustic modeling and RTM migration implemented in this work is the propagation of the seismic signal, which is governed by the acoustic wave equation

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} = 0$$

A numerical solution of this equation is obtained by finite difference. The classic second order in time, fourth order in space approximation (2-4 operator) was used and a future pressure field could be expressed by

$$p^{k+1}{}_{i,j} = -2\,p^{k}{}_{i,j} - p^{k-1}{}_{i,j} - \frac{c^2(\Delta t)^2}{(\Delta x)^2}\,p^{k}{}_{i+1,j} - \frac{c^2(\Delta t)^2}{(\Delta y)^2}\,p^{k}{}_{i,j+1} + \left(\frac{2c^2(\Delta t)^2}{(\Delta x)^2} + \frac{2c^2(\Delta t)^2}{(\Delta y)^2}\right)p^{k}{}_{i,j} - \frac{c^2(\Delta t)^2}{(\Delta x)^2}\,p^{k}{}_{i-1,j} - \frac{c^2(\Delta t)^2}{(\Delta y)^2}\,p^{k}{}_{i,j-1}$$

As imaging condition for RTM migration the criterion of maximum energy near the first break[2] was used. The image is constructed by propagating a Ricker pulse in a softened estimated velocity model and building up an array of travetimes that record for each grid point, the time at which the maximum amplitude occurs. The seismic image is obtained by reverse propagating the seismograms from the receiver coordinates until the time recorded in each array element is reached and collecting the amplitudes of this reverse propagation in their respective grid points. To prevent unwanted reflections on the edges of the model, Cerjan's[3] edge-absorbing technique has been implemented.

Those algorithms was implemented in a FPGA which is composed by a set of two-dimensional computational elements known as logical blocks that communicate through a programmable interconnection network, as illustrated in Figure 1.

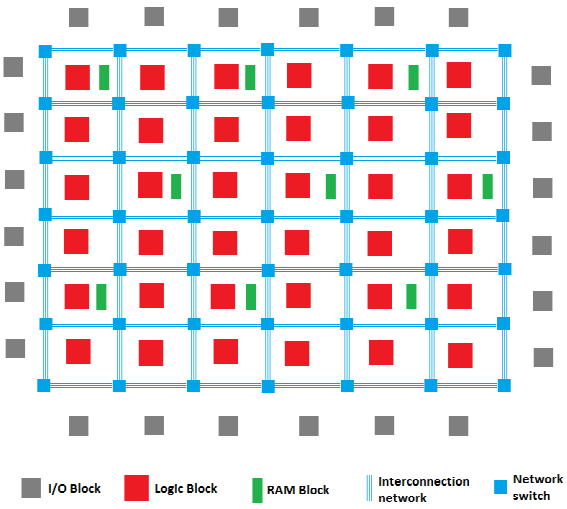

Figure 1: Internal FPGA structure

Each logical block or CLB is able to perform basic logic functions and store some data bits.

The interconnection network is fully programmable, so that it is possible to literally interconnect the outputs of a logic block to the input of any other. Both the logic function of each block and the routing therebetween are programmable via a data sequence supplied to the device during its initialization. Special input/output blocks are connected to the device's pins and are used to connect the logic to the memory or the computer bus. In addition to the blocks shown there are additional RAM and specialized multiplication blocks that significantly increase the computing power of the device.

Method

The development took place on two different fronts: Arithmetic and architecture.

The arithmetic front treats the finite difference operator inner works while the architecture deals with all the remaining design responsible for driving the data from memory to the operator, and vice versa.

We know that the solution of the wave equation by finite differences requires a large amount of computations and also a large volume of data traffic with memory. These are the key elements that need to be optimized. To obtain the best possible performance, we maximize various design parameters both in arithmetic and in architecture fronts.

Arithmetic:

High performance of the finite difference operator can be achieved basically maximizing the number of mathematical operations per unit of time and this can be done in two ways: Increasing the clock speed and parallelizing the operator computation. In order to increase the clock rate we must simplify the operator. A very significant simplification is to use fixed-point math operations instead of floating point computation, commonly used in microprocessors. To accomplish this, a detailed study[5] was made to ensure that the fixed point representation does not introduce significant errors in the results. The use of fixed-point operators reduces the amount of logic required for the computation of finite differences, contributing positively to the increased clock frequency of the operator and also allowing a greater number of concurrent operators to be implemented. Some smart tricks were employed to achieve further simplifications of the operator. We can cite for example, the way to perform a simple by 60 multiplications, which is necessary for the finite differences operator. Instead of performing a multiplication by 60 which consumes considerable resources of logic, it was decided to perform two multiplications: one for 64 and another for 4 and then to subtract the results. The gain is the fact that multiplying by powers of 2 is basically a simple bit shift, and a complex multiplication circuit was replaced by a much more simple subtraction.

Other front is to parallelize the operator mathematical computation. Tasks that would be performed sequentially on a microprocessor are then executed concurrently by independent circuits. With this approach, after the initial loading of the data, the operator shall be able to compute a grid point per clock cycle. This set of optimized circuits, able to solve arithmetic finite difference operator is called PE (Processing Element). Due to all simplifications made to the PE, there was a significant reduction in the amount of resources used by it, which allows to instantiate up to 16 PE's on a single FPGA, and thus 16 grid points can be processed simultaneously at each clock cycle.

Architecture:

There are two basic strategies for achieving performance in Architecture: Parallelism and data reuse. We know that the finite difference operator requires a large volume of memory data access, and often the same data has to be read several times, which makes the performance highly dependent on the memory access bandwidth. This fact is so critical that becomes a bottleneck in the execution of tasks, i.e., the processing capability of the 16 PE's ends up being greater than the ability we have to drive the data to them. Therefore, pure and simple parallelism is not enough; we need to think of an efficient data flow, avoiding unnecessary memory access. The first approach in this direction is to group the PE's in-line, so that neighbor PE's shares some of the same input data. This strategy is limited by the memory access bandwidth, because as we increase the number of in-line PE's, more and more input data is needed do feed them. Given the characteristics of Stratix III board available, we reach the maximum number of 4 PE's in-line, which is far less than the resources available in the FPGA logic, sufficient for the creation of up to 16 simultaneous PE's. A major challenge then arises: How to calculate more grid points without increasing bandwidth memory access? To answer this question, we need to rethink the finite difference implementation[4]. To apply the finite difference operator, traditionally for each grid point calculation of the future pressure field P2, one previous pressure field grid point at P0 and the present pressure field P1 region surrounding the point in question must be read from memory plus the value of the velocity field at the same point, as illustrated in Figure 2.

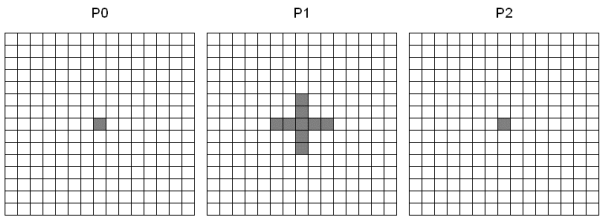

Figure 2: Finite difference stencil for the 2-4 operator.

For the computation of each time step, usually the same grid point is read several times from memory during the algorithm execution. Using the RAM blocks available in the FPGA, a buffer is instantiated in a first-in/first-out (FIFO) configuration in order to store 4 lines of the current pressure field. This allow us to read each grid point from memory just once and making them available to the finite difference operator along the entire time step, thus avoiding extra memory I/O. That is done in groups of 4, as we have 4 PE's in-line. Figure 3 illustrates this approach.

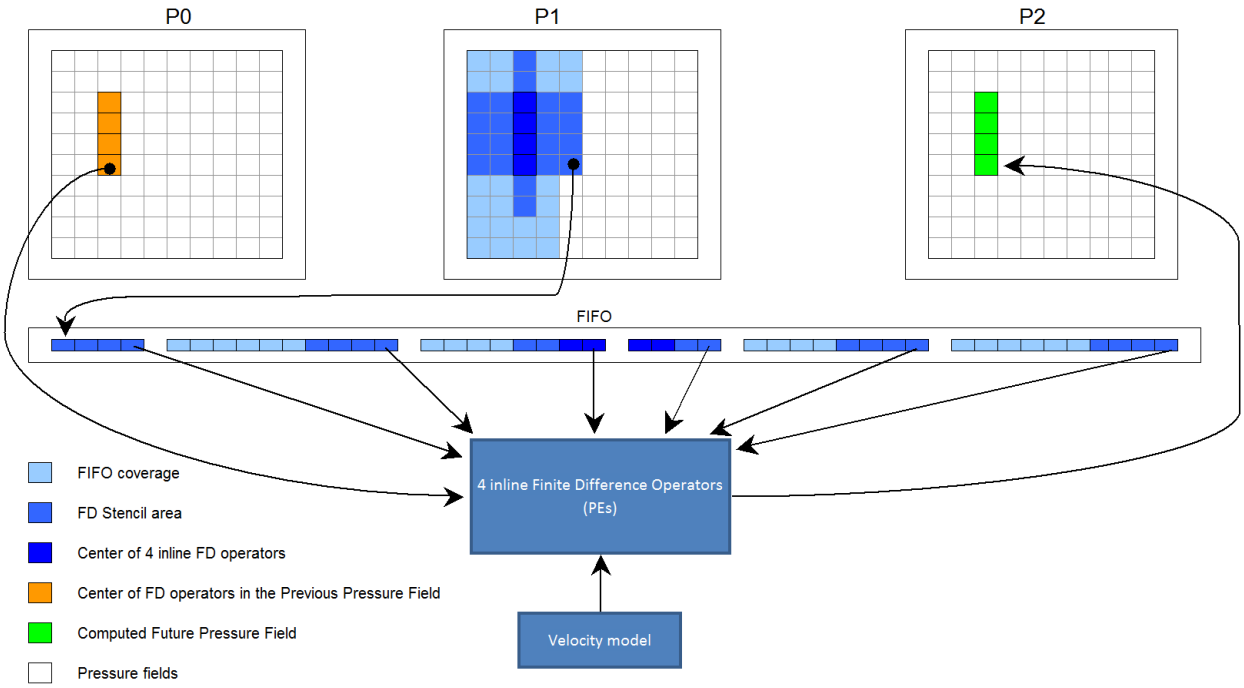

Figure3: Using FIFO's to calculate a time step with just one read of each current pressure field grid point.

In resume, the FIFO approach leads to a module which can read the entire current pressure field data just once and produce as output, the entire future pressure field data. One very clever use of this fact is that we can daisy chain such modules in order to calculate several time steps almost simultaneously. The number of time steps that can be evaluated with this approach is limited by the model size, the amount of block RAM's and the amount of logic necessary to construct the PE's. The FPGA used in this work allowed us to compute 4 time steps at the same time, calculating 16 grid points of the future pressure field on each clock pulse after the FIFO´s are filled.

Seismic Modeling and RTM implementations:

Once created the finite difference machine design, it needs to be inserted into a larger context for the realization of Seismic Modeling and RTM Migration. At this point, the implementation is divided into three: the Direct Seismic Modeling; whose purpose is the generation of synthetic seismograms, the Forward Modeling for Migration; whose goal is to generate the travel time matrix and the Reverse modeling; whose goal is to generate the migrated section by reverse propagating the field data seismograms. Only one of these implementations is loaded into the FPGA at a time, according to the desired functionality. Each one of these implementations requires additional calculations to those already considered in the finite difference machine. Fortunately, in contrast to software solutions, all these calculations are executed in parallel, including the Ricker source term and the Cerjan damping edges calculation.

After all these elements are instantiated, the FPGA is ready to perform the modeling or the migration. It is important to remember that the FPGA board needs to exchange data with a host computer, which in turn, reads and writes to disk. So, additional software and drivers was written and ran on the host to handle this data.

Results and Data Analysis

We carried out the designs of seismic modeling and 2D RTM acoustic machines. They were tested in simulation environment and were also deployed on a GIDEL PROCe III with 1 ALTERA Stratix III FPGA running at 50 MHz.

Were modeled and successfully migrated several synthetic geological models, including the Marmousi and Hess model, which input parameters are listed in Table 1.

	Marmousi	Hess
Model dimensions with damping zone	2301 x 751	3601 x 1501
Damping zone	80	80
Cut frequency	60 Hz	45 Hz

Table 1: Modeling and migration parameters

We implemented three different designs: Direct Seismic Modeling (synthetic seismogram generation), Modeling for Migration (generation of travel times and amplitude arrays) and Reverse Migration (generation of the migrated section). Each version had its runtime measured. Table 2 compares the FPGA runtime with the runtime performed by software on an Intel Xeon E5430

2.66 GHz quad-core. In summary, the FPGA modeled Marmousi about 30 times faster and migrated about 20 times faster than the CPU.

Design Version	CPU (s)	FPGA (s)	# Timesteps
Direct Seismic Modeling	546	18	8500
Modeling for Migration	272	11	4500
Reverse Modeling	476	28	8500

Table 2: CPU vs. FPGA Marmousi runtime comparison

Table 3 shows the execution times for each of the required steps to perform a direct modeling with seismogram generation. It can be seen that considerable time is spent on data flow outside the FPGA, burdening the total execution time by 46.3%. This result is due to the lack of software optimization on the host side, responsible for sending, receiving and processing the data exchanged with the FPGA board.

	Execution time (s)	Execution time (%)
FPGA Configuration	0,553	3,2
Sending velocity model and Ricker pulse to board	0,117	0,7
FPGA runtime (8500 timesteps)	9,408	53,7
Returning results to host	2,667	15,2
Host post processing	4,771	27,2

Table 3: runtime of each step for Marmousi seismogram modeling

Table 4 shows the execution performance (expressed in billions of grid samples processed per second - GS/s) of direct modeling of Marmousi and Hess models. The result is slightly better for the Hess model primarily due to its larger size which improves the FPGA runtime ratio compared with the remaining tasks.

	Marmousi 217 shots (GS/s)	Hess 99 shots (GS/s)
Direct Seismic Modeling	0,88	1,11
Modeling for Migration	0,73	0,77
Reverse Modeling	0,55	0,66

Table 4: Total performance in Giga Samples per second for Marmousi and Hess models on FPGA platform

The power consumption was also evaluated considering a real machine running different velocity models and different scenarios. All experiments were performed taking the average values of several voltage and current measurements. Table 5 shows the results comparing the energy efficiency of modeling and migration of the Marmousi model.

Marmousi Model 8500 timesteps	Runtime (s)	Mean processing power (Watts)	Total Energy (Joules)	Relative energy efficiency
Modeling CPU	546	200,32	109375,8	1,00
Modeling FPGA	18	207,86	3741,5	29,23
RTM CPU	748	199,93	149547,9	1,00
RTM FPGA	39	205,71	8022,6	18,64

Table 5: CPU vs. FPGA Marmousi energy efficiency comparison

In summary, considering the Marmousi model, the FPGA based approach consumes approximately 29 times less energy to generate synthetic seismogram and approximately 19 times less energy to produce a migrated section compared with the CPU based approach.

Figures 4 and 5 shows respectively the migrated sections of Marmousi and Hess models, computed by the FPGA. Due to the precision study conducted, the difference between the numerical results obtained by the proposed method and those obtained by software running on CPU are negligible.

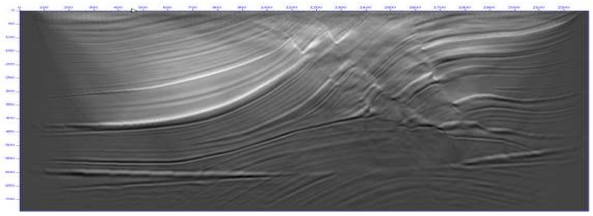

Figure 4: Marmousi migrated section

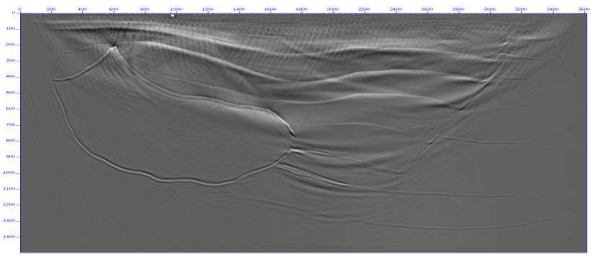

Figure 5: Hess migrated section

Conclusions

This paper presented the development and implementation of 2D seismic modeling and RTM migration in a hybrid platform composed of a generic CPU (host) and a platform accelerator device based reconfigurable FPGA (Altera ES260E). All equations, control units, arithmetic unit (PE) etc.. relevant to the algorithm were implemented in hardware, all described in Verilog, synthesized and validated, leading to the following conclusions:

• The proposed platform was able to perform seismic modeling and RTM migration with better computational performance and better energy efficiency than the current computer clusters

The design ran with a relatively low frequency due to limited I/O resources of the board used (basically memory I/O bandwidth).

• The proposed system is scalable. It is ease to expand the computational power with the addition of more FPGA's on a multi-platform FPGA. In simulations the performance reached 4.4 GSamples/s for a board with four FPGA's.

• The computational performance could be further increased by the use of a suitable FPGA board with better memory bandwidth and host software optimizations. As such board does not exist in the market, for better performance is mandatory the development of a dedicated board with sufficient I/O bandwidth.

• The FPGA based power efficiency can also be improved further, as no optimization was implemented yet in order to reduce consumption.

• The major drawback is the fact that the complexity of VERILOG development led to a high development time and the maintenance of designs are difficult, requiring specialized professionals.

Expectations

• A high-level language OPEN-CL is being ported to FPGA's, which will greatly reduce the skills required and the total development type of the designs.

• The growth rate of FPGA internal resources is very high. This fact allied with the low power profile of FPGA´s and the use of OPEN-CL, leads us to believe that this technology could have a solid presence in the high performance computing scenario in near future.

Future work

Partitioning the model into multiple FPGA's

Acoustic Modeling and RTM migration of 3D models

Development of a suitable hardware platform

Development in OPEN-CL high-level language

References

[1] Eduardo Lopes de Faria, 1986, “Migração antes do Empilhamento Utilizando Propagação Reversa no Tempo” , Dissertação de Mestrado, PPPG – Universidade Federal da Bahia.

[2] André Bulcão, 2004, “Modelagem e Migração Reversa no Tempo Empregando Operadores Elásticos e Acústicos”, Tese de Doutorado, COPPE – Universidade Federal do Rio de Janeiro.

[3] Cerjan,C., Kosloff, D., Reshef, M., 1985, “A Nonreflecting Bondary Condition for Discrete Acoustic and Elastic Wave Equations, Geophysics, v. 50, pp. 705-708.

[4] Medeiros, V W C ; Rocha, R. C. F. ; Pyetro, A. ; Dutra, B. ; Barros, A. C. ; Liborio, J. C. ; Barbosa, J. P. F. ; Barros Júnior, S. J. ; Menezes, G. ; Silva Filho, A. G. ; Lima, M. E., High performance FPGA-based implementation of the seismic modeling of the RTM algorithm 2011. In: SC.2011, Seattle, WA,2011

[5] Barros, Abner Correia ; Dutra, Bruno H. T. C. ; Brito, Vinícius V.; Lima, Manoel Eusebio; Silva Filho Abel Guilhermino da. Modelo para Avaliação de Desempenho na Redução de Precisão para FPGAs Aplicados a Modelagem Sísmica. In: WSCAD-SSC 2011, Vitória – ES