CANADIAN JOURNAL of EXPLORATION GEOPHYSICS
VOL. 36, NO. 1 (June 2011), P. 60-68

Fast Marching Method Seismic Traveltimes with Reconfigurable Field Programmable Gate Arrays

Xiaoning Zhang[1] and R. Phillip Bording[2]
Electrical Engineering Department-Earth Science Department
Memorial University of Newfoundland, Newfoundland, Canada

ABSTRACT

This paper presents a hardware based least time path fast marching method for seismic traveltime calculation. The hardware implementation uses field programmable gate array (FPGA) reconfigurable computing technology and applies it to a compute intensive seismic problem. The algorithm transformation process is described at a system level and at a detailed hardware level.

Keywords—Eikonal Equation, Seismic Traveltimes, Kirchhoff Migration, Fast Marching Method, Reconfigurable Computing, FPGA.

I. Introduction

Traveltime calculations play an important role in many methods of seismic data processing. For example, Kirchhoff migration and modeling of seismic data require calculating traveltimes and as with other seismic computations, it is very compute intensive. Typical seismic computer programs run on high performance multi-core clusters of parallel computers. These parallel computers are sometimes augmented with dedicated program accelerators' most commonly graphics processing units (GPU's). Here we consider the use of FPGA's for seismic processing algorithms. The comparison of FPGA and GPU implementations will be the basis for future research efforts. Our goal is to facilitate seismic computing on reconfigurable computing hardware, where a digital circuit is designed to implement the computing logic of a specific algorithm. Current and previous work on hardware seismic algorithms includes two ray tracing projects, a migration project, and an acoustic wave equation project by He (2011), Lu (2008), Chuan (2004), and Bording (1995), respectively.

In the past, high engineering development costs made it infeasible to develop a chip to work on a single algorithm of one specific field. These chips would have low production volumes which would not justify commercialization. To address these smaller markets and to improve the time-to-market the semiconductor industry now provides reconfigurable computing platforms, a FPGA chip which can be reprogrammed to create the needed logic.

The direct physical implementation of an algorithm can improve performance; here processing speed is no longer limited by the rigidity of computer architectures: arithmetic operation, data flow, and memory access can be optimized. The advances in reconfigurable computing by Lewis, et al., (1998) and the significant increase in logic cells available as discussed in Trullemans, et al., (2000) when to applied to a single algorithm are the driving forces to explore these technologies. These reconfigurable FPGA techniques are demonstrated for a complicated geophysical application.

Digital hardware design is a different form of software programming. The data movement and computation can be described in one of several hardware description languages (HDL's). In our research, a difference equation algorithm of the traveltime calculation is proposed, and is implemented using FPGA technology. The traveltime data are computed for a typical and realistic two dimensional model. The cost and current size of FPGA chips in terms of logic cells limits our efforts to two dimensional problems. Near term improvements in FPGA technology will provide substantial increases in capability which will make three dimensional traveltime hardware both fast and cheap.

II. Traveltime calculation

The use of seismic ray shooting methods followed by interpolation of traveltimes onto a regular grid are popular, robust, and presented in Cerveny (2003), Julian and Gubbins (1977), and Um and Thurber (1987). However, for complicated velocity models, rays may cross each other and/or not penetrate shadow zones; the interpolation is thus cumbersome and computationally expensive. These issues of ray tracing for seismic processing using FPGA's have been considered in the work of He (2011) where the bending ray method was used to overcome some of the difficulties describe above.

As an alternative to ray tracing, Vidale (1988) introduced a method directly solving the eikonal equation on a regular grid with a finite difference approximation. The eikonal equation is obtained from the elastic-wave equations by searching for plane harmonic solutions and applying the high frequency approximation of ray theory (Lecomte, et al., 2000). Vidale extended the finite difference schemes described by Reshef and Kosloff (1986) to compute traveltimes of first arrivals in isotropic media model.

$$\left(\frac{\partial t}{\partial x}\right)^2 + \left(\frac{\partial t}{\partial z}\right)^2 = s^2 \qquad 1.$$

The eikonal equation can be written as equation 1, the heterogeneous velocity model is presented as slowness, s, the reciprocal of velocity. The traveltime is t.

There are two keys to understanding Vidale's method, the local cell approximation to equation 1 and the structural expansion scheme for the grid which starts initially around the source point.

[1] zhangxiaoning01@gmail.com
[2] philbording@hotmail.com

A. Local Scheme of Extrapolation

With the geometry in Figure 1, traveltimes from source point to point A (t_0), B_1 (t_1), B_2 (t_2) are known; traveltime from point A to point C_1 (t_3) is sought.

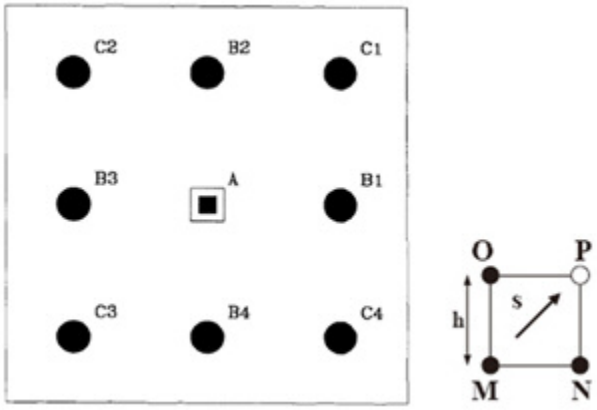

Figure 1. Finite difference grids for traveltime calculation, from Vidale, (1988).

$$\frac{\partial t}{\partial x} = \frac{1}{2h}(t_0 + t_1 - t_2 - t_3) \quad 2.$$

$$\frac{\partial t}{\partial z} = \frac{1}{2h}(t_0 + t_1 - t_2 - t_3) \quad 3.$$

$$t_3 = t_0 + \sqrt{2(hs)^2 - (t_2 - t_1)^2} \quad 4.$$

The finite difference approximation to the derivatives in the (x and z) directions are given in equation 2 and equation 3, and are the cell average derivative values. Assume that three cell corner times are known (t_0, t_1, and t_2) and one is unknown (t_3). Then equation 4, the extrapolation formula of Vidale's method can be derived.

$$t_3 = t_s + s\sqrt{(x_s - h)^2 + (z_s - h)^2} \quad 5.$$

Besides equation 4, Vidale also derived equation 5 to deal with the extrapolation of the wavefronts which have high curvature.The coordinates of the virtual source point are (Xs and Zs within a grid cell) of the circular wavefront with high curvature, and ts is the origin time for the virtual source (Lecomte, al., 2000).

Considering the wave generated from a source point, the wavefronts in the near-origin region have higher curvature than those in the far-origin region. The combined use of equation 5 in the near-origin region and equation 4 in far-origin region is called a "mixed" scheme by Vidale, which provides better accuracy. For purposes of simplicity and to achieve higher speed rather than great accuracy, equation 4 can be used exclusively in a 'simple' scheme.

B. Grid Expansion

Once the source point is fixed the Vidale method expands around the source grid point in a structured manner. The reader is referred to Vidale (1988). The basic process was to use an expanding rectangular grid scheme to decide how to spread and extend the computing to unknown grid points on the rectangle edges. This process is illustrated in Figure 2.

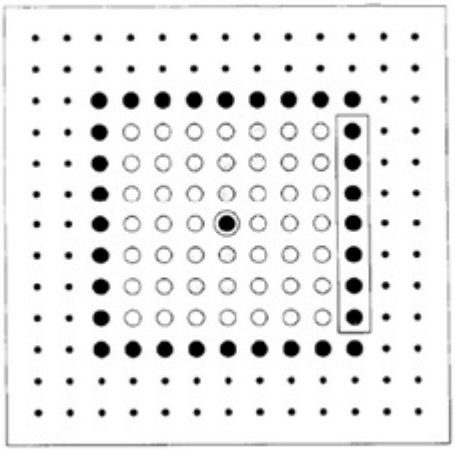

Figure 2. Computing traveltimes of the larger black points, from Vidale (1988).

In Figure 2, points inside the box of larger black points are the points with known traveltimes; the point in the centre of the box is the source point; and the smaller black points outside the box are the ones with yet to be computed traveltimes.

To expand the computed traveltimes to larger black points, the row of known traveltimes next to them must be used.

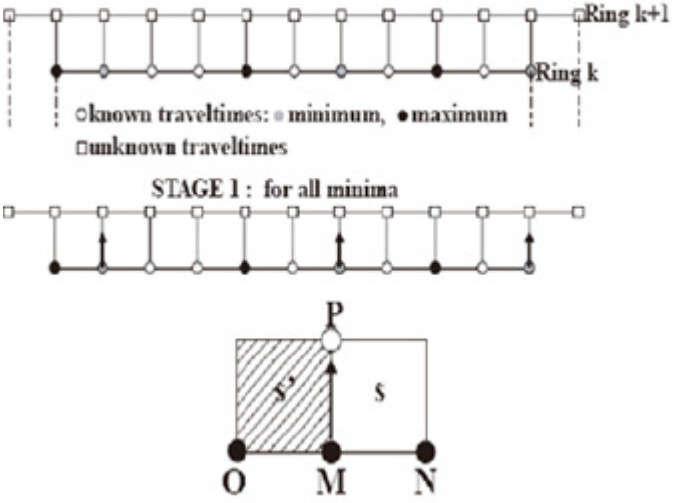

Figure 3. Step 1 in inductive scheme, calculate unknown traveltimes for next row starting from minima of the current known row.

On the first step, find out the relative minima traveltime values on the known row; and calculate the unknown traveltimes using equation 6:

$$t_3 = t_0 + \sqrt{(hs)^2 - 0.25(t_2 - t_1)^2} \quad 6.$$

In equation 6, t3 is the target traveltime on point P in Figure 3; t_0 is the traveltime on point M, which is a relative minimum on the row of known traveltimes; and t_1, t_2 are the traveltimes on O and N.

On the second step, the order of computing new traveltimes should be from relative minima to relative maxima by applying equation 4 of local scheme as illustrated in Figure 4.

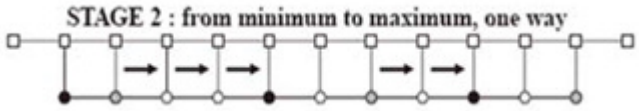

Figure 4. Step 2 in inductive scheme, solve new traveltimes from minima to maxima.

On the third step, reverse the order of solving in step two, still from relative minima to relative maxima as shown in Figure 5. The points in front of relative maxima are assigned two traveltimes from step two and three, only the smaller is kept.

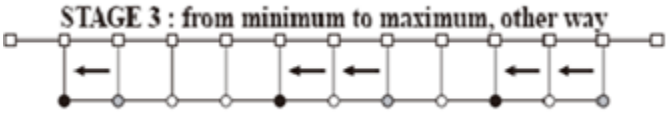

Figure 5. Step 3 in inductive scheme, solve to left from minima to maxima.

After solving the points on the four sides of Figure 2., the travel times on the corners can be computed by applying equation 4.

C. Limitations

A drawback of Vidale's method is that it may fail in the models with strong velocity variations; typically velocity changes in excess of 5 percent caused the method to have difficulties. The first derivative operations needed for the eikonal are based on the average of the two side derivatives, and this is problematic in that the traveltime contours have curvature and this averaging process has a higher error component than using the edge derivatives. To overcome these limitations new methods based on level sets and fast matching methods were developed by Van Trier and Symes (1991) and Sethian and Popovici (1991). Here we consider a method that is based on Fermat's principal of least time within the grid cell and uses the notion of others for expanding the next grid point from the active set of points in the computational domain. These ideas are discussed in great detail in the next section.

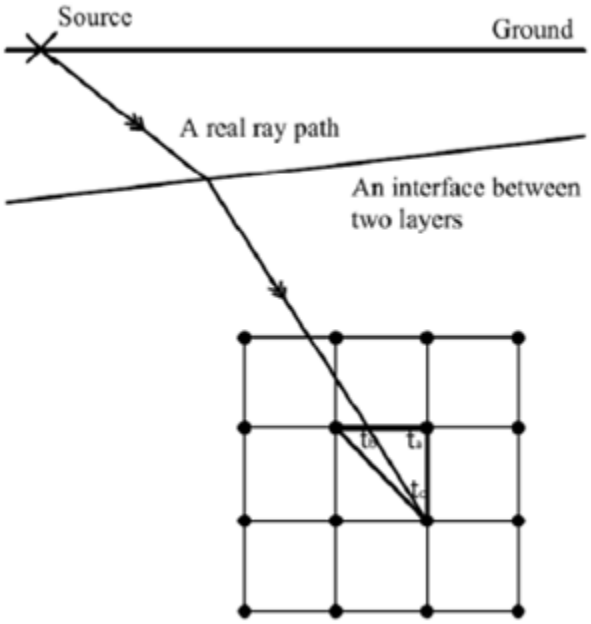

Figure 6. Ray shooting in local scheme numerical approximation.

III. Least time fast marching method

A. Introduction

Fermat's principle of least time is applied within the local cell and combined with the notion of expanding the grid based on a using the smallest time in all of the currently active set of points. The method is more robust as it allows for general velocity variations and does not breakdown.

As shown in Figure 6, ray path from a source point located on the ground crosses edge AB of triangle ABC.

$$\lambda = \frac{t_\lambda - t_a}{t_b - t_a} \quad 7.$$

$$t_\lambda = t_a + \lambda(t_b - t_a) \quad 8.$$

$$t_c = t_a + \lambda(t_b - t_a) + \sqrt{1 + \lambda^2}\cdot h\cdot s \quad 9.$$

A linear interpolation factor λ ranges from 0 to 1 is defined as equation 7 following the illustration of Figure 7. The diagonal approximation, equation 4, used in Vidale's method is compared with equation 9, which considers the relative position and angle of current finite difference grid to the source point, which makes the interpolation more accurate. The ray intersection traveltime is shown in Figure 8. The unknown traveltime point C can thus be expressed as equation 9 assuming traveltimes t_a and t_b are both known. In equation 9, h is the width and height of the finite difference grid cell, ***s*** is the slowness given by a known velocity model; if the only unknown

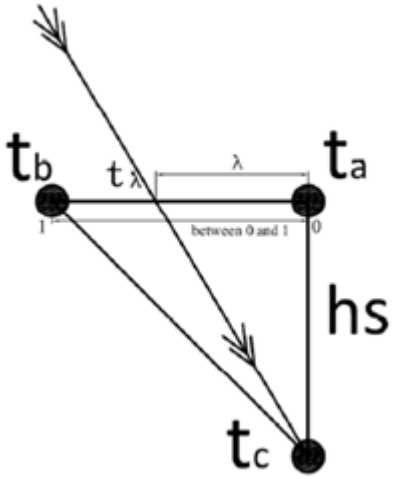

Figure 7. Ray shooting in local scheme numerical approximation.

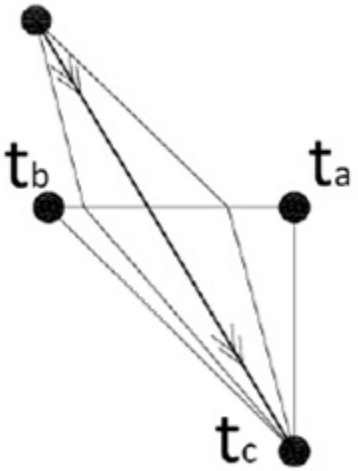

Figure 8. Least traveltime path of ray tracing.

factor λ can be expressed in terms of other known variables, this equation can be used as our local scheme of extrapolation, which is the case.

According to Fermat's Principle, the path taken between two points by a ray is the path that can be traversed in the least time. In nature, the chosen intersection point along the edge AB (in Figure 8) is the one which makes for the smallest ray traveltime.

$$\frac{\partial t_c}{\partial_\lambda} = (t_b - t_\alpha) + \frac{\lambda}{\sqrt{1+\lambda^2}} \quad \mathrm{h} \cdot \mathrm{s} = 0 \qquad 10.$$

The time as equation 9 is expressed in terms of an unknown λ value and the known data, and if we determine the triangle side ray intersection point we know λ. The first derivative of equation 9 shown here as equation 10 should be equal to zero when t_c is the least traveltime from source point to point C, and we can solve for λ.

$$\lambda^2 = \frac{(t_\alpha - t_b)^2}{h^2 s^2 - (t_\alpha - t_b)^2} \qquad 11.$$

$$t_c = t_\alpha + \sqrt{h^2 s^2 - (t_\alpha - t_b)^2}\,. \qquad 12.$$

From equation 10, equation 11 gives the interpolation factor λ which decides the least travel path of the ray. Solving equation 9 using equation 11 we arrive at equation 12 to calculate unknown traveltime at point C with the known data values of t_a, t_b, h, and s.

In a square grid, if the ray path crosses the edge AB to enter the triangle ABC, the angle α should be in the range of 0° to 45°; and the extrapolation formula equation 12 can only handle the ray with angle α in this range. If these conditions are not met, then an important step must be taken to limit the travel times at the angle boundaries. For all the rays with α not in this range, their traveltimes at point C would be counted as fixed approximate values, as shown in Figure 9. To make judgement on angle α, its tangent value is used:

When $t_a < t_b$ and $\alpha < 0°$,

$$t_c = t_\alpha + hs \qquad 13.$$

When $t_a > t_b + \sqrt{2}hs$ and $\alpha > 45°$,

$$t_c = t_b + \sqrt{2hs} \qquad 14.$$

In the application, incident ray may come from any direction from source point towards the grid points on the plane. To cover possible incident rays coming from 360° circumference, there can be 8 triangles surrounding a single grid point as shown in Figure 10; applying the least time triangle formula in every triangle, and then selecting the minimum result out of the 8 results computes the previously uncomputed grid point traveltime.

As shown in Figure 10, the computing unit for a grid point is named the Radial Incidence Computing Element (abbreviated as RICE, for its shape is like a Chinese character 'rice'). It is not necessary that all the traveltimes are known on the surrounding points; for any unknown t_a, t_b, simply put its value as 'infinity' when applying equation 12. In this way, the results from triangles containing unknown times will not contribute to the possible solution set.

B. Inductive Scheme

Fast Marching Method was developed for solving the eikonal equation by Sethian and Popovici (1999). In this method, a

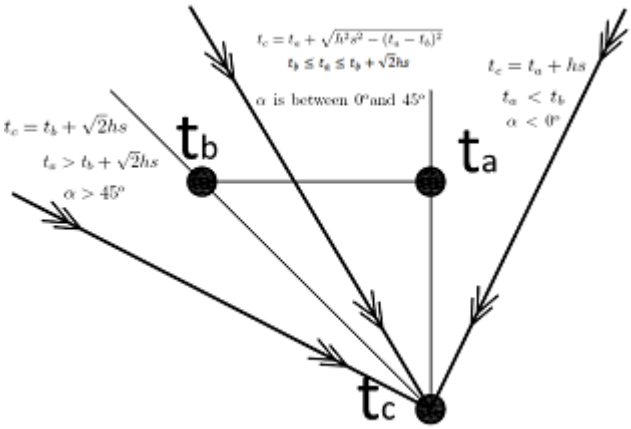

Figure 9. Interval of applying the formula.

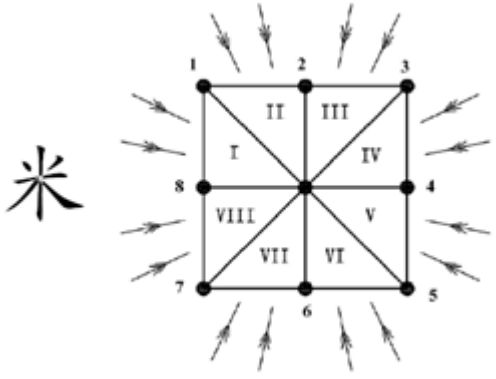

Figure 10. RICE computing model.

wavefront construction technique builds a narrow band around the traveltime wavefront. The computed traveltimes are stored in a sorting buffer, which always returns the smallest time grid point location. The wavefront always advances by using the grid point with the minimum of known traveltime to calculate next unknown traveltimes, and include the results into the narrow band of the wave front, the active set points, as illustrated in Figure 11. The incremental software cost of the buffer sorting operation is log (N), as this is an insertion step into an already sorted list, where N is the number of traveltime values in the narrow band of active grid points.

All the points on computational plane can be classified into three kinds: the points behind wavefront on the upwind side, whose traveltimes are computed and accepted; the points on the wavefront in the narrow band area, whose traveltimes have been computed, but have not accepted yet; and the points ahead of the wavefront on the 'downwind' side.

Adopting fast marching method in our traveltime computing, the algorithm is developed as following steps:

- Initialize traveltime values on all the grid points on the plane to be infinity. Assign initial value times to a small number of seed points surrounding source point, and store these traveltimes into the sorting buffer.
- Sort the minimum traveltime grid point out of the buffer as a basis point.

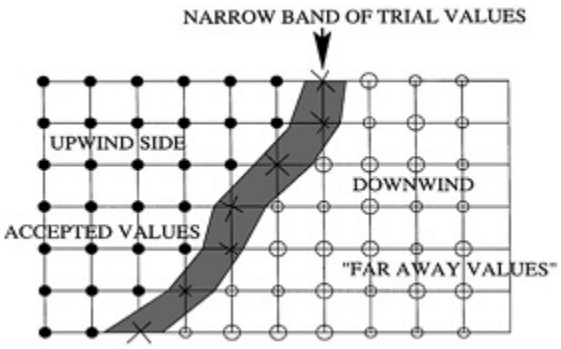

Figure 11. Fast marching method wavefront, from Lecomte, et al., (2000).

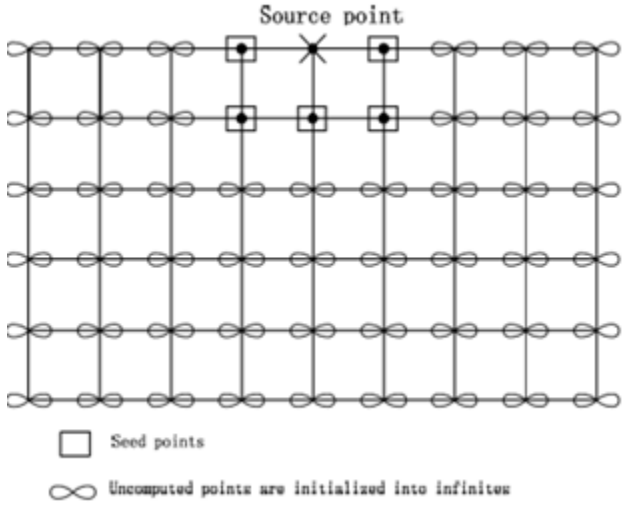

Figure 12. Initialization.

- Calculate and decide the eight neighbour points of this basis point.
- Apply rice computing on each neighbor point following local extrapolation scheme.
- Store the traveltime results and neighbour points into the sorting buffer (the active set buffer); and write the basis point into fixed set or called accepted set (upwind side) as an accepted point.
- Repeat step 2 to step 5 until all the points on plane become accepted

As illustrated in Figure 13, the initial set of seed points are the points close to source point; their traveltime values can be precalculated. The number of seed points can be decided according to the value of the specified seed radius around the source point

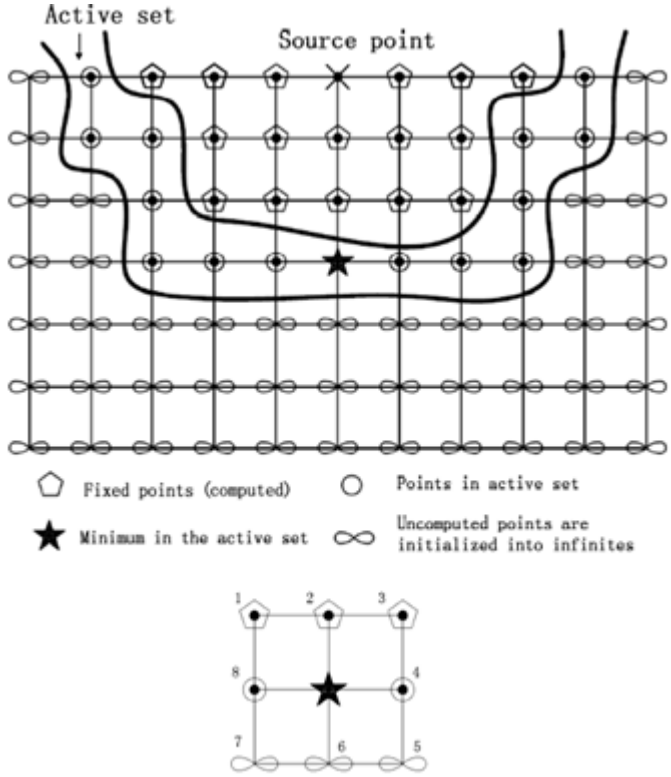

Figure 13. Initialization.

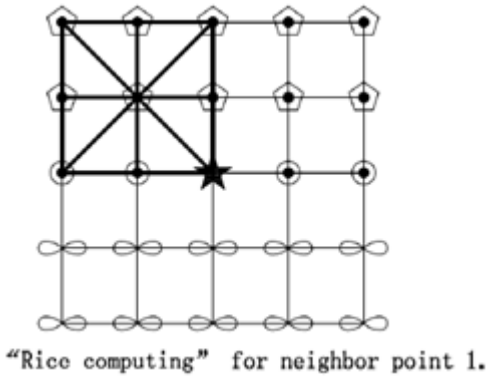

Figure 14. Rice computing on grid points.

grid. Except for seed points, all other unknown points are set infinity as their initial traveltime values.

In step 2, the sorting buffer can sort and output the point which has the smallest traveltime value among seed points pre-stored in the buffer during initialization. This point indicated by sorting buffer is used as a basis point to find out its eight neighbour points (Figure 13) by calculating corresponding coordinates. In step 3, apply rice computing algorithm developed in local scheme of extrapolation for each neighbour point; Figure 14 illustrates the rice computing on neighbour point 1. If the neighbour point is not yet in the accepted set and active set, namely, the points with unknown traveltimes such as point 5, 6, 7 in Figure 13, the points with computing results will be stored into sorting buffer as active set points. The basis point (the star in Figure 13) is stored into accepted set and removed from active set. Repeatedly applying step 2 to step 5, the wavefront composed of active set points will go through the plane, and every point will be computed.

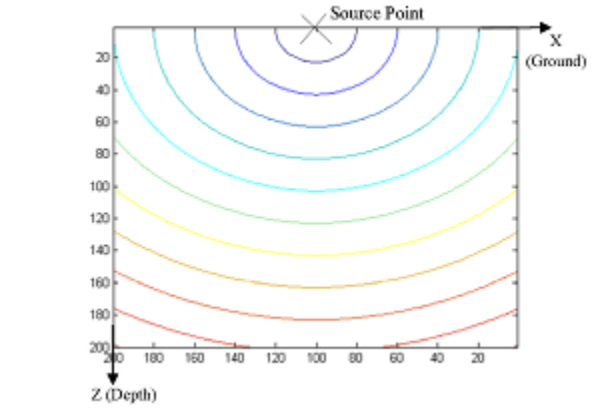

Figure 15. Rice computing on grid points.

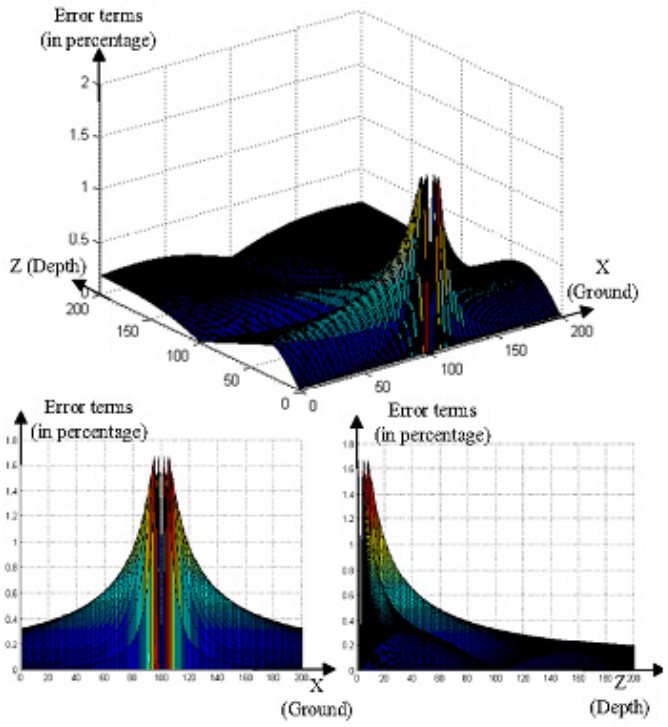

Figure 16. Rice Error plot generated by comparing the results of our method with the exact traveltimes using Pythagorean Proposition (Errors in percentage).

IV. Softward Simulation

To verify the algorithm, software simulation is run on different velocity models. Figure 15 shows simulation results that the algorithm runs on a constant velocity model. In this simple model, the velocity of seismic wave on each grid point is assumed a fixed number. Figure 15 shows the contours of computed traveltimes.

In this constant velocity model, it is not difficult to calculate the 'exact' traveltimes on each grid point according to Pythagorean proposition. Comparing the output from our numerical algorithm and the results by Pythagorean proposition, error distribution plot can be drawn as Figure 16 shows. Every point in this plot is the percentage calculated based on equation 15.

$$err = \frac{T - T_{exact}}{T_{exact}} \times 100\% \qquad 15.$$

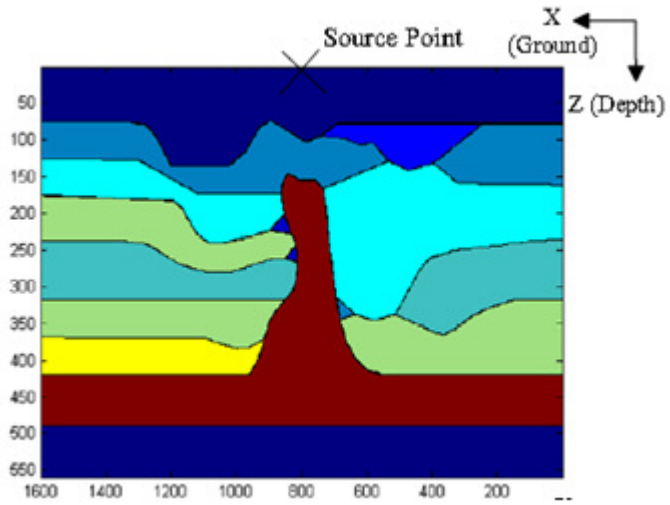

Figure 17. Dablain model.

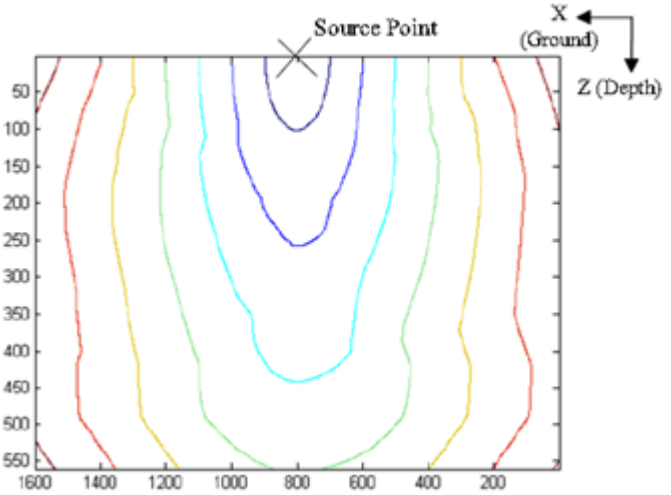

Figure 18. Travel time contours on Dablain model.

It can be seen that error distribution in source region is high. This is because seismic wavefront curvature in this area is high. Improvement in this error can be made by increasing the number of initial seed points, or developing 'mixed scheme'.

Figure 17 shows a more complex velocity model from Dablain with several layers of different velocities. The traveltime contours of Dablain model are shown in Figure 18.

V. Hardware

A. Reconfigurable Computing on FPGA with HDLs

A Field Programmable Gate Array (FPGA) is an integrated circuit chip designed to be configured by the customer or designer after manufacturing. FPGAs contain many programmable logic blocks and a hierarchy of reconfigurable interconnects that allow the blocks to be 'wired together' in different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. The logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.

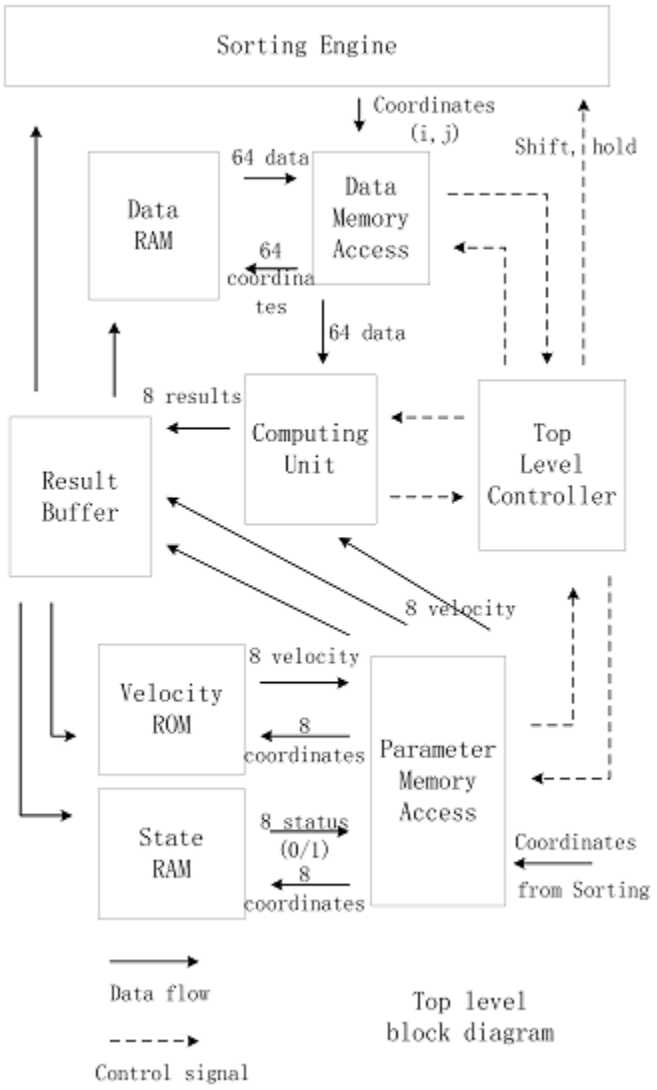

Figure 19. Top level block diagram.

To define the behavior of the FPGA, the user provides a HDL description specifying functions of the digital design. Using electronic design automation (EDA) tools to synthesize the design, a technology-mapped netlist is generated. The netlist can then be fitted to the actual FPGA architecture using a process called place-and-route (PAR). The user will validate the map, place and route results via timing analysis, simulation, and other verification methodologies. Once the design and validation process is complete, the binary bit file generated is used to configure the actual FPGA chip.

The most common HDLs are VHDL and Verilog, whose complexity is compared to the equivalent of an assembly language. In a typical design flow, the design is under simulations at multiple stages throughout the design process. Initially the register transfer logic (RTL) description in VHDL or Verilog is simulated by creating testbenches to simulate the system and observe results in logic level. After the synthesis engine has mapped the design to a netlist, the netlist is translated to a gate level description where simulation is repeated to confirm the synthesis proceeded without errors. Finally the design is laid out in the FPGA at which point propagation delays can be added and the simulation run again with these values back-annotated onto the netlist.

The ability to update the functionality after shipping, debug the design in the field, partial re-configuration of the portion of the design, shorter time to market, and the low non-recurring engineering costs make FPGA an ideal platform for reconfigurable computing.

B. Digital System Design

To solve traveltime computing problem with reconfigurable computing, we have to design a digital system to implement our least time fast marching method based on FPGA technology. Digital design defines datapath components and controller of a digital system. Datapath is a collection of functional units which perform data processing operations need by the target algorithm; controller usually a finite state machine (FSM) controls and schedules the data flowing through datapath components to be processed in correct order. Block diagram of the digital system to implement least time fast marching method is as shown in Figure 19.

C. Datapath

To implement algorithm steps described in section III, the system can be partitioned into following functional units: computing unit, sorting engine, memory access modules, and three pieces of memory. The array of traveltimes is stored in a random access memory (RAM). The two dimension coordinates (i, j) of a grid point is translated into a one dimensional memory address by data memory access module. Velocity array is stored in a ROM (read only memory); it records velocity value (actually the value hs) for every grid point. This ROM is actually rebuilt for every new model by redoing the FPGA programmable design process. The state RAM maintains an array of flag bits of every point; if the point is computed and has a known

traveltime value, the flag bit is set to '1', or otherwise set to '0'. The parameter memory access module controls access of these two storages by translating the grid logical coordinates into physical memory address. Result buffer is composed of registers which buffer the outputs from computing unit; it stores resulting traveltimes and coordinates of the points, which can be written back to the corresponding storages. The sorting engine is a functional block which buffers all the grid points (traveltime values and coordinates) of the active set.

1) Computing unit:

The computing unit shown in the top level block diagram is composed of eight rice computing units; each rice computing unit is composed of eight 'triangle computing units', as the structure shown in Figure 20.

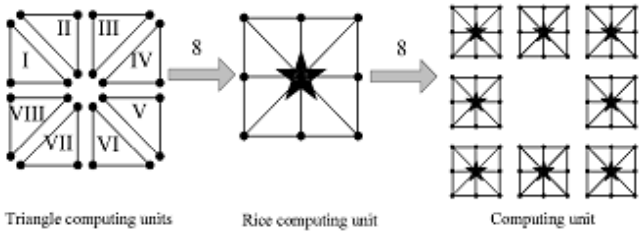

Figure 20. Structure of computing unit.

The triangle computing unit implements equation 12, equation 13, and equation 14. Single precision floating point number is chosen as the number representation in arithmetic operations. Floating point arithmetic is an IP (intellectual property) software core provided by the FPGA vender (Xilinx), which is a well designed and tested library unit can be reused in customer projects.

Configuring this IP core, addition, subtraction, multiplication, square root and comparison (==, <=) operations in equation 12, equation 13, and equation 14 can be implemented. Because division is a expensive hardware operation, it has been avoided in developing the equations. Figure 21 illustrates a design of triangle computing unit. T_a, t_b, and hs are inputs to this module; arithmetic units are connected according to logic order defined in equations; a multiplexer is used to select the correct output by making judgement of incident angles based on the calculation with t_a, t_b. During calculation, the algorithm requires the value 'infinity' which is represent by the largest number Xilinx floating point core can represent, '7f800000'.

A rice computing unit instantiates eight triangle computing units and seven comparators as shown in Figure 22. The inputs are the traveltimes of eight neighbor points plus the value hs. After going through triangle computing, the eight results go through three stages of comparison to determine the minimum – least time grid point.

According to the hierarchy shown in Figure 20, instantiations eight rice computing units into one block; the computing unit can take 25 points as inputs, and output eight traveltimes with eight rice computing at once.

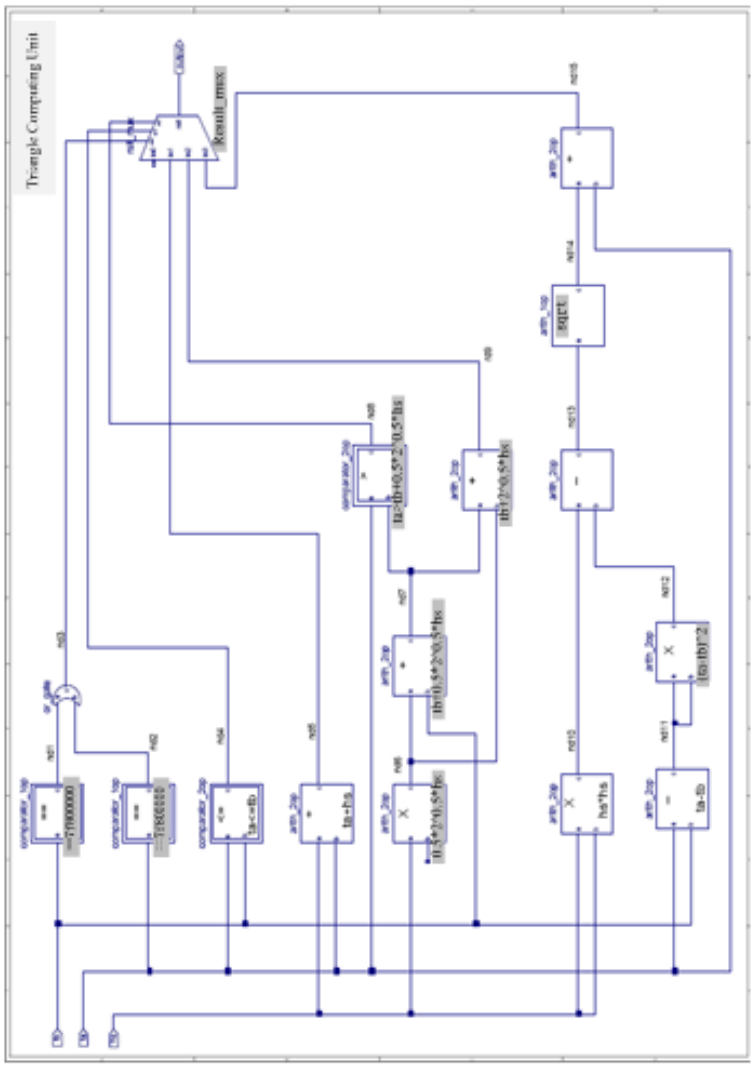

Figure 21. Implementation of equations with arithmetic units.

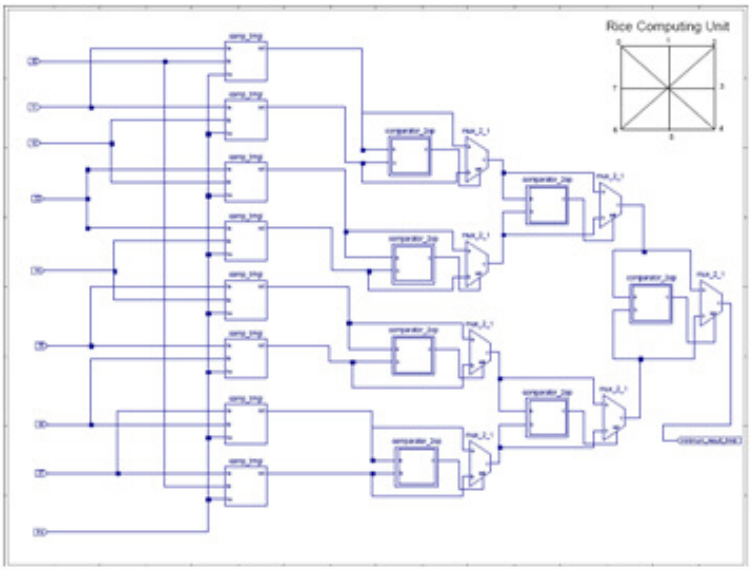

Figure 22. Rice computing with triangle computing units.

2) Sorting Engine:

The buffer holding active set points can sort traveltime values based on insertion sorting. This sorting engine is composed of an array of sorting cells Figure 23. The first sorting cell always stores the smallest traveltime point. The input data and grid coordinates can be 'seen' by every cell through the buses. Control signals (hold and shift) can control output of the first sorting cell which holds the minimal traveltime point. Each sorting cell contains three registers, three multiplexers and a comparator. Among sorting cells, there are comparisons between neighbor sorting cells: based on certain logic, input points can be inserted to right position, and other points in sorting cells can be 'shift' or 'hold' accordingly. In this way, sorting engine always maintains an ordered array of sorting cells.

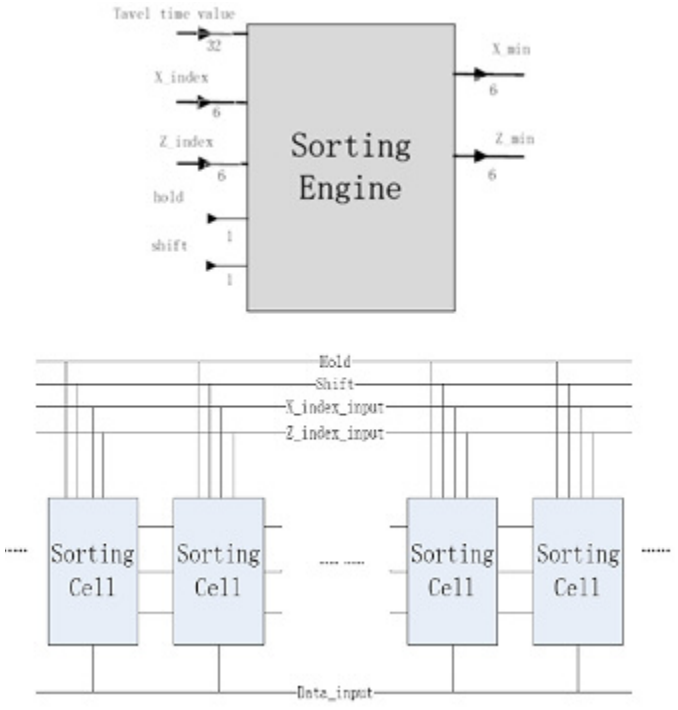

Figure 23. Sorting engine and sorting cells.

D. Controller

In a digital system, the operating sequence of datapath modules is scheduled by the Finite State Machine (FSM) controller. The digital system is synchronized up in the pace of a system clock; controller sequentially sends all sorts of control signals to every datapath unit to apply necessary operations. The many details of the FSM machine for this project are found in the MS Thesis of the first author, Zhang (2008).

VI. Conclusions

In the work, we solve a difficult seismic computing problem with reconfigurable computing technology. The least time path fast marching method is developed to calculate traveltimes, which is suited to implement on reconfigurable FPGA's. Software simulations were run to verify the algorithm. A digital system design was proposed, implemented, and tested to demonstrate the feasibility of this approach. We hope this work can be an inspiration to apply these ideas to other seismic computing problems.

Acknowledgments

The authors will like to thank Dr. Sam Gray and the other reviewers for their insightful comments and suggestions. This work was supported by generous grants from IBM, (especially many thanks to Dr. Kirk Jordan), and ACOA Grant 189320, and by the Husky Energy Chair in Oil and Gas Research held by Dr. Bording.

References

Bording, R. P., 1995, Wave Equation Difference Engine, PhD Dissertation, U. of Tulsa.

Cerveny, V., 2003, Seismic ray theory, Journal of Seismology, vol. 7, p. 543, Nov.

Chuan, H, Lu. M., and Sun, C. W., 2004, Accelerating seismic migration using fpga-based coprocessor platform, 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM04) Proceedings, pp. 207-216.

He, L., 2011, Travel Time Engine for Seismic Petroleum Applications using Evolving Methods, MS Thesis(in review), Memorial University of Newfoundland.

Julian, B. R., and Gubbins, D., 1977, Three-dimensional seismic ray tracing, Geophysics, vol. 43, pp. 95-114.

Lecomte, A. D. I., Gjoystdal, H., and Pedersen, O. C., 2000, Improving modelling and inversion in refraction seismics with a first-order eikonal solver, Geophysical Prospecting, vol. 48, pp. 437-454.

Lewis, D. M., Galloway, D. R., van Ierssel, M., Rose, J., and Chow, P., 1998, The transmogrifier-2: A 1 million gate rapid-prototyping system, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 6, no. 2, pp. 188-198, Jun.

Lu, S., 2008, Genetic Algorithm Design for Ray Tracing and Hardware Implementation, MS Thesis, Memorial University of Newfoundland.

Reshef, M. and Kosloff, D., 1986, Migration of common shot gathers, Geophysics, 51, 324-331.

Sethian, J. A., and Popovici, A. M., 1999, 3-d traveltime computation using the fast marching method, Geophysics, vol. 64, no. 2, pp. 516-523, Mar-Apr.

Van Trier, J., and Symes, W. W., 1991, Upwind finite-difference calculation of traveltimes, Geophysics, vol. 56 no. 6, pp. 812-821, Jun.

Trullemans, A.M., Ferreira, R., David, J. P., and Legat, J. D., 2000, A multi-fpga system for prototyping power conscious algorithms, 15th Design of Circuits and Integrated Systems Conference (DCIS 2000) Proceedings, pp. 41-46, Nov.

Um J., and Thurber, C. H., 1987, A fast algorithm for two-point seismic ray tracing, Bulletin of the Seismological Society of America, vol. 77(3), pp. 972-986.

Vidale, J. E., 1988, Finite-difference calculation of travel times, Bulletin of the Seismological Society of America, vol. 78, pp. 2062-2076, Dec.

Zhang, X., 2008, Least Time Path Fast Marching Method for Seismic Travel Time Computing: Theory and Implementation, MS Thesis, Memorial University of Newfoundland.