GENETIC ALGORITHM DESIGN FOR RAY TRACING AND HARDWARE IMPLEMENTATION

SHILIANG LU

Genetic Algorithm Design for Ray Tracing and Hardware Implementation

By

Shiliang Lu

A thesis submitted to the school of graduate studies

in partial fulfillment of the requirement for

the degree of mast of engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

OCTOBER 2008

ST. JOHN'S NEWFOUNDLAND AND LABRADOR CANADA

Acknowledgements

I am very grateful to my supervisor, Dr. R. Phillip Bording, for his constant guidance, feedback, encouragement, for keeping me focused in my research. During the past two years, Dr. Phillip not only has given me consistent trust and support, which has encouraged me to improve and finish my work, but also supports me when my life encounters the shock. He is not only a dependable supervisor in the academic studying, but also a trustable father in my life.

I also have to acknowledge Mrs. Michelle Butt. She has walked me through the writing of this thesis. Without her carefulness checking for the grammar and illuminating instruction for the technical writing, this thesis could not reach its present form.

I am thanking all the professors and Mrs. Moya Crocker in Faculty of Engineering and Applied Science for their selfless academic and personal help.

This is also a chance to thank all rest of the members of Music-π group in the Memorial University of Newfoundland during these years. Special thanks to Dr. Jingyi Chen, Dr.Andreas Atle, Liangjie He, Swastika Sharma, Md. Waheduzzaman Talukder, Lujie Zhan, Xiaoning Zhang for the precious friend ship and their co-operation as well as valuable suggestions.

I have to thank the funding of Memorial University, ACOA/AIF, Husky Energy and IBM for the financial support.

I am also indebted to my parents. Thank you for you continuous support and love during two years of my Master study.

I would like to thank my wife, Yanzhen Ou, for her selfless supporting and encouraging me to pursue this degree.

Abstract

Over the last 30 years, seismic ray tracing methods have played the important roles in the geophysical exploration and seismology. Various seismic ray tracing methods had been proposed, including ray bending and shooting methods based on the two-point ray tracing and graph theory based on the grids-algorithms. Each of these methods has its limitations, like shadow zone problem and nonlinear issues. In this thesis, we investigate the use of the genetic algorithms (GA), which are nonlinear global search algorithms, to improve upon these existing issues. Using a simpler continuous layer (polynomial based) function representation, ray tracing is accomplished by sampling each interface for a set of intersecting points. Based on these points, a ray path is traced from the shot point to a reflector interface back to the receiver. This process is similar to ray bending. The method for the generation of the interface points is a genetic algorithm and it finds the Fermat path of the least travel time. However, it is computational intensive. In order to improve the algorithm the run times are reduced by using the genetic algorithm to generate some of the interface layers points and using Snell's Law to bend the rays at other interface layers. We validate the suitability and correctness of the two proposed methods using seismic modeling and Pre-stack Kirchhoff migration. The results of Kirchhoff migration demonstrate that the reconstructed subsurface structures fit the real

model very well, and also prove that the proposed methods are very effective seismic ray tracing methods.

In addition, the hardware implementations are powerful approaches to accelerate our proposed ray tracing algorithms. Moreover, considering that the development of the hardware implementations did not attract much attention in geophysics, a purpose built, specific hardware algorithm is developed and a hardware engine is implemented in the low-cost field-programmable gate array (FPGA) device. The fixed-point arithmetic, the functional parallel design, the high efficiency sorting engine and the memoryless design for the velocity model work together to produce a comparable performance with IBM workstation.

All results mentioned above demonstrate that the Pre-stack Kirchhoff migration and the hardware implementations of our seismic ray tracing methods are all feasible and the proposed approaches may be further extended for the more complex media.

Table of Contents

List of Figures and Tables

List of Abbreviation

GA	Genetic Algorithm
2D	Two Dimensions
3D	Three Dimensions
MPI	Message Passing Interface
API	Application Programming Interface
ACEnet	Atlantic Computational Excellence Network
RAM	Random Access Memory
FPGA	Field-Programmable Gate Array
PCI	Peripheral Component Interconnect
FSM	Finite State Machine
RNGu	Random Number Generation unit
GLFSR	Galois Linear Feedback Shift Register
MTTM	Minimum Travel Time Method
MESLM	Minimum Errors of Snell's Law Method

Chapter 1

Introduction

Ray tracing is a method to simulate the paths of rays (waves or particles) through a system and, based on the reflected and transmitted points on the passed surfaces, to model the physical effect (Glassner, 1989). It traces rays propagation obeying Snell's Law, and decides rays bending directions when they travel through an interface. All the rays recorded at the destination are traced from the source (Cerveny, 2001), by which the energy and the frequency of the rays are determined. This method closely simulates the ray travel in nature. Thus, it is widely used in the computer graphics to produce the highly realistic 3D image (Peter and Keith, 2001) and used in the physics sciences to model and analyze the transmission system (Cerveny, 2001).

In the seismic exploration or seismology, seismic ray tracing is a very powerful approach for providing seismic travel time (refracted, direct, reflected travel time and so on) and seismic wave form information (Officer, 1974) by simulating seismic wave propagation in the assumed complex media, including isotropic and anisotropic media. Generally, the

seismic wave propagations are studied from source to receiver to locate the source position (if source is set underground) (Haslinger, 2001) and to determine physical parameters (p-wave velocity, s-wave velocity, density) (Wesson, 1971), as well as to determine geometry parameters (reflector depth) (Docherty, 1991) in the subsurface.

1.1. Seismic Modeling and Migration Using Ray Tracing

Seismic modeling and migration play major roles in seismic exploration. They are valuable tools for the seismic data interpretation and the subsurface structure reconstruction. Seismic modeling simulates waves propagation in the assumed structure of the subsurface model. Based on the travel time of waves propagation and the source function, seismic modeling generates the synthetic seismogram (Carcione *et al.*, 2002). Seismic migration utilizes the seismic data to construct the image of the subsurface structure and removes the effect of the seismic wave propagation from data to move seismic events towards their correct subsurface positions (Alaei and Pajchel, 2006).

One of cornerstones of seismic modeling and migration is the travel time calculation between two given points in the assumed subsurface model. Seismic ray tracing is one of several travel time calculation approaches that can be used in the geophysics. Given two points in the subsurface model, called a shot and a receiver (shown in Fig. 1.1), as an efficient method, seismic ray tracing traces the propagation of the seismic waves from the shot point to the receiver point based on Snell's Law, and calculates the travel time of a wave propagation (Keller and Perozzi, 1983).

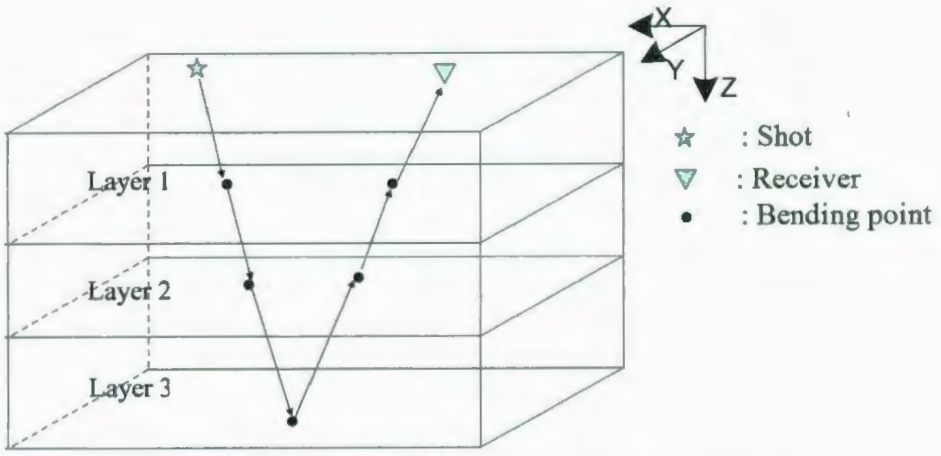

Fig. 1.1. The sketch map of seismic ray tracing in the 3 layers subsurface velocity model.

1.2. Seismic Ray Tracing Methods: Shooting and Bending Methods

Shooting and bending methods are two major seismic ray tracing methods in use today. In the shooting method, a seismic ray is firstly emitted with an assumed direction (take-off angle) from the shot point. Then, along the take-off angle direction, the intersection points with the interface layer, called the bending points, are calculated. Based on Snell's Law, the shooting method tries to match the calculated reflected ray to the receiver by iteratively improving the take-off angle (Sadeghi *et al.*, 1999). This method is a very efficient seismic ray tracing method, but, it has some known limitations.

As shown in Fig. 1.2, for a certain subsurface model and a shot-receiver pair, we may not find a ray path, which is shot from the shot point with any direction arriving at the receiver point. Therefore, the ray paths have to be interpreted to simulate the wave transmission and calculate the travel time. The interpretation introduces an error that only occurs at the receiver location. As one can notice that, the error could also be distributed

into the bending points on the interfaces. So that, the simulated ray paths and the calculated travel times are much closer to the real cases.

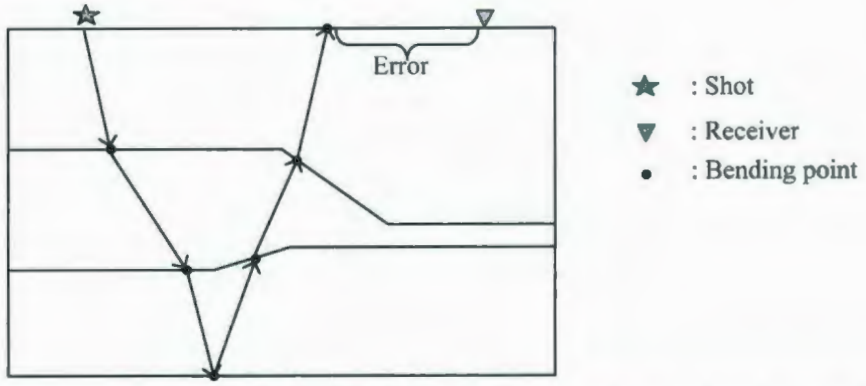

Fig. 1.2. The illustration of a failure of the seismic ray shooting method.

In the bending method, the bending points are assumed already on the interfaces, but the locations of the bending points are unknown. The method uses an algorithm to search all the points on the interfaces and finds the bending points, which satisfy Snell's Law and Fermat's Principle. If the subsurface model is too complex to find the bending points, the search algorithm returns a set of bending points that have the minimal errors (Lu and Bording, 2007, Zhao *et al.*, 2004). In this case, the total travel time error is the sum of errors of all the bending points on the interfaces.

1.3. The Search algorithm: Genetic Algorithm

The Monte Carlo method (Metropolis and Ulam, 1949), simulated annealing (Kirkpatrick *et al.*, 1983) and genetic algorithm (Holland, 1975) are the most popular search algorithms in used today. As the most powerful search algorithm, the GA is designed for very complex function optimization (Goldberg, 1989). The idea of GA comes from the biological evolution (Holland, 1975). It is based on several initial individuals (bending points sets in this thesis) to continuing evolution. During the continuing evolution, the

individuals who are suitable to the environment are remain and the others are eliminated through the contest. After several numbers of generations, the best individual is selected as the optimization result.

1.4. The Objective of the Research

The first focus of our research is to design the seismic ray tracing bending methods using GAs. The derivations of two ray tracing methods: the minimum travel time method (MTTM) and the minimum errors of Snell's Law method (MESLM) are discussed. The detailed designs of GAs for both methods are provided and they are compared with respect to the calculation efficiency, results quality, software parallelizability and hardware implementations. In order to validate the correctness of our bending methods, the 2D and 3D seismic modeling and Pre-stack Kirchhoff migration are implemented.

The error calculation of a bending point is the most computational intensive part in our algorithms. As the second focus of our research, to accelerate our ray tracing algorithms, a purpose built, specific digital hardware implementation is proposed. This hardware implementation is intended to work as an integrated card, which can be plugged into peripheral component interconnect (PCI) socket to supply the application-specific travel times. The ray tracing applications, e.g. seismic modeling and Pre-stack Kirchhoff migration program, could access the PCI interface to read/write the data and the control signals. In the hardware implementation, the fixed-point calculation runs faster and uses less chip space than the corresponding floating-point calculation. Thus, the fix-point arithmetic is employed to evaluate the cost function. The function evaluations for each

layer could be organized in a parallel fashion. This idea gives much better performance than Von Neumann micro-processors (Bording, 1996).

1.5. Thesis Outline

In this thesis, there are seven chapters. Chapter 1 is the introduction. Chapter 2 provides the principle of GA. Specifically, the theory of GA, the design components of GA, the workflow of GA and the exploration and exploitation search strategies of GA are discussed.

Chapter 3 proposes two layer based ray tracing methods, the MTTM and the MESLM. Both methods are ray bending methods and implemented by GA. The fitness functions developments and the functions sensitive analysis of both methods are discussed in detail. The GA components implementations of both methods are provided.

Chapter 4 provides the implementations of seismic modeling and Pre-stack Krichhoff migration. The message passing interface (MPI) parallel technique design for both applications is discussed. Two 2D and two 3D layer based synthetic velocity models are introduced in the experiments. From the results of Pre-stack Kirchhoff migration, the proposed methods are tested.

Chapter 5 compares the proposed MTTM with MESLM from the calculation efficiency, the simulated ray paths and results quality, the software parallelizability, as well as the hardware implementations. Moreover, the reasons of the differences are analyzed.

Chapter 6 provides the hardware algorithm and the implementation of the hardware engine, based on the proposed MTTM. In this chapter, the top level design, the hardware implementations of the components of the designed GA and a high efficient sorting

engine with temporal and spatial complexity of O(N) are provided. The hardware performances are investigated by comparing the synthesized engine with the IBM workstation and the function of the implemented engine is validated by the post-route timing simulations.

Chapter 7 draws a conclusion for this thesis and provides the directions for some future work.

Chapter 2

Genetic Algorithm

As one of the effective and powerful evolutionary algorithms, GA was originally proposed by John Holland (Holland, 1975) to represent a fairly abstract model of biological evolution. It has been successfully applied to various scientific and engineering problems, especially, to the very complex function optimizations (Goldberg, 1989). Based on the stochastic search, GA provides the approximate solutions, which could be further optimized by a mathematical optimizing method, e.g. Newton method.

There are several advantages (Davis, 1991), which allow GA to be the efficient function optimization method. Firstly, the function to be optimized in GA is operated as a black box. The algorithm only needs to be informed the types of the inputs and outputs of the black box. The mathematical properties of the function are unnecessarily considered. In theory, the same platform of GA could be applied for any function optimization. Secondly, unlike other stochastic search algorithms (Monte Carlo and simulated annealing), GA is based on the suitability of the previous solutions to generate the next

solutions. Thus, the random solutions are pushed to the direction, which optimizes the function. Moreover, because more than one previous solution are selected, the local optima problems are greatly avoided.

As one can notice that in order to design a GA to optimize a realistic problem, the pre-step has to abstract the problem into a mathematical function representation. The inputs of the function must be sensitive to the outputs of the function.

2.1. The Components of GA

An efficient GA design usually contains five components: population, selection replacement, individual evolution, fitness evaluation and termination. Each component is described below.

2.1.1. Population

Because of GA coming from the biological theory, the names of the components of the GA are close to biology. A set of solutions in GA is called a population and a single solution in the population is called an individual. The solution space, which includes all the potential solutions, is called the search space. All the individuals in the population are initialized in the search space and will be evolved in the search space.

An individual in the population could be represented by several formats. Generally, it is represented by a binary string, which is called the genotype of the individual. The value of the binary string, a real or integer value is called the phenotype. It could also be represented by a real number vector, which is the natural representation for the numerical

optimization (Koza, 1992). In the real number representation, the genotype and the phenotype are assumed to be the same for an individual.

For example, using GA to minimize the function: $y = x^2 + 10$, as shown in Fig. 2.1, the search space of this function should be designated as the entire real value range. The population might be defined as including ten individuals (represented by green points in Fig. 2.1). The number of individuals in the population is called population size.

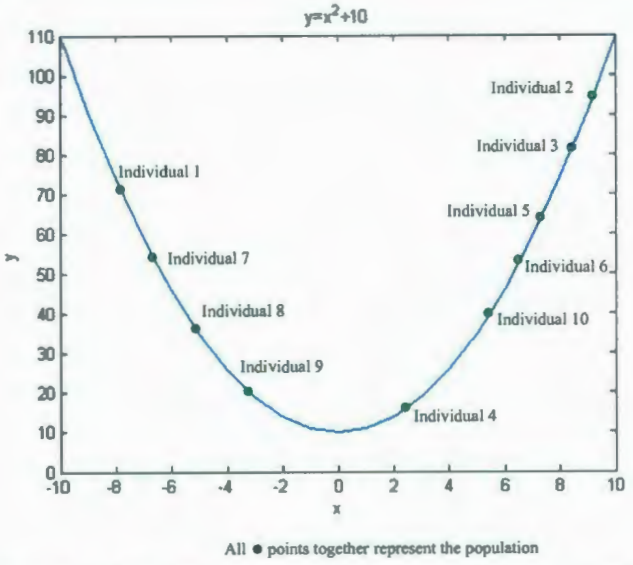

Fig. 2.1. The illustration for the population of GA.

The population size will affect the performance of GA. With the same setup of other components, the larger population size, the more accurate solutions are produced, but the computational complexity is increased as well. With a larger population size, more individuals are included in the search space thereby increasing the population density, so that the local optima are easily avoided.

2.1.2. Selection and Replacement

In GA, the selection occurs in two places. First, GA selects the individuals in the population as parents to evolve and produce the offspring. Second, GA selects the suitable individuals to replace the worse individuals and generates a new population. The selection strategies work on the entire individual formats, i.e. they are representation independent. The different selection strategies have to be adopted for different problems. For the same optimizing problem, the different strategies perform differently.

2.1.3. Individual Evolution

Basing on the selected parents, the offspring are produced at the individual evolution step. The evolution methods, called genetic operators, are different with respect to the different individual representations. Generally, there are three main categories: crossover, mutation and copy.

Crossover operator means that GA combines two or more individuals to generate at least one offspring. In the binary representation, the crossover occurs with exchanging the bits in the same position of two or more parents. In the real number vector representation, besides operating the same way as that on the binary strings, the arithmetic crossover (Michalewicz, 1996) is also applied to generate the offspring, which are the linear combination of their parents. The purpose of using crossover operator is to reserve the suitable genes in the parents to the offspring. That similarly happens in nature as inheritance.

Mutation generates one or more offspring from only one parent. In the binary representation, the offspring are produced by flipping the bits of a binary individual, with

some probabilities. For the real number vectors, other random vectors are generated and the offspring are produced by adding the new random vectors to their parent. This genetic operator brings the extra genes into the original population and is called aberrance in nature.

Copy operator works as its name, producing the identical offspring from the parent. It is designed to keep suitable individuals in the population for future.

2.1.4. Fitness Evaluation

The fitness evaluation is applied to measure the suitability of an individual. It substitutes the individuals into the fitness function, which is supposed to be optimized, and collects the results of the function as the fitness. For the numerical optimization, the fitness would be the sum error, mean squared error, root mean squared error etc. A healthy fitness function has one and only one fitness for an individual.

2.1.5. Termination Criteria

After evaluating the fitness of the parents and their offspring, the GA checks the termination criteria to decide whether the optimization is successful. The criteria would be either a tolerable time period or a threshold for the optimum solution. If the program reaches the criteria, the algorithm is terminated and the optimum solution is outputted. Otherwise the old population is replaced by the suitable individuals to generate the new population.

2.2. The Workflow of GA

As shown in Fig. 2.2, GA is an iterative algorithm. In an iteration, called a generation, parents selection strategy chooses the individuals as the parents and genetic operators generate evolutions in the selected individuals. Based on the fitness evaluation criteria, the survivor selection chooses the remaining survivors and replacement produces the new population. With the number of generations increasing, GA pushes the random individuals in the population to the directions that optimize the function.

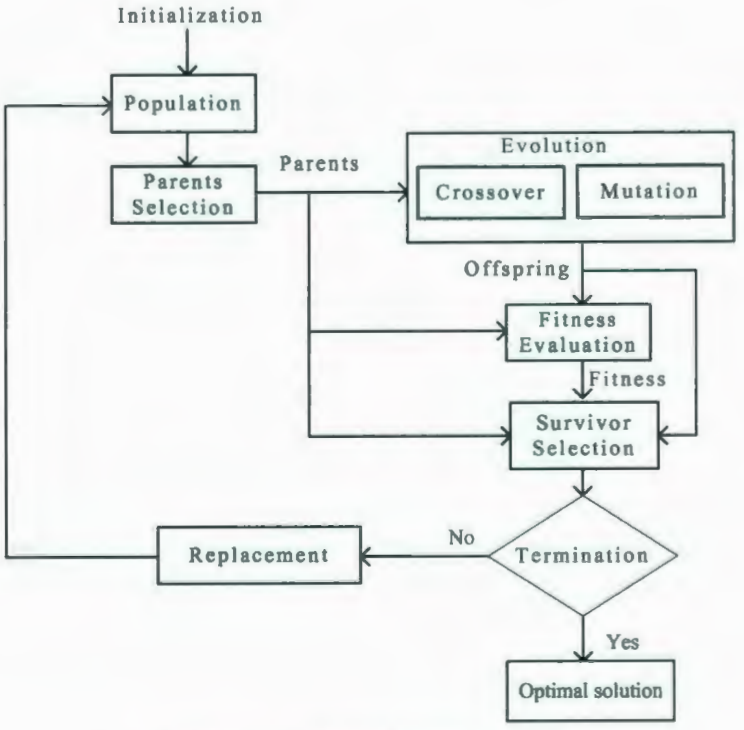

Fig. 2.2. The workflow of GA.

2.3. The Search Strategies in GA: Exploration and Exploitation

As a successful search algorithm, GA obtains its power from two sources: exploration and exploitation. The exploration strategy brings the new genes into the population and

exploitation strategy combines the existing genes in the population to produce the new individuals (Eiben and Schippers, 1998).

If the search space is large e.g. real number space, it is impossible to search all the inclusive cases. GA is based on a set of explored values (initial population) to exploit the potential solution within the population. If the solution genes are in the population, GA may easily and quickly pick up the solution. Otherwise, GA uses the potential solutions to explore more related values in the search space, and lead the random search to the solution direction. In general, the exploration strategies in GA include: initialization and mutation. The exploitation strategies include: selection and crossover.

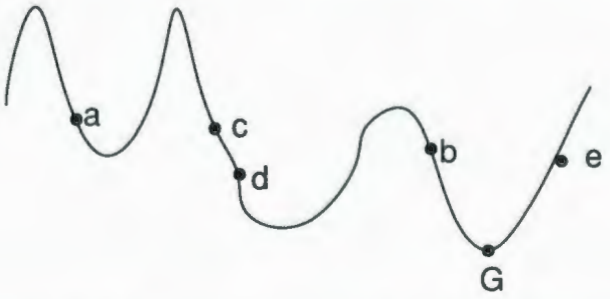

Fig. 2.3. The illustration of the exploration and exploitation search strategies.

An example, which shows how the exploration and exploitation work, is illustrated in Fig. 2.3. The curve represents the function that GA tries to minimize. The point 'G' is the global minimum solution. The points 'a', 'b', 'c', 'd' and 'e' are the explored initial individuals in the population. GA exploits the existing population and selects points 'd', 'b' and 'e' as the potential solutions. Then GA may continue exploiting the potential solutions using arithmetic crossover, which combines the genes in the selected solutions to produce the other solutions, or explore the potential solutions using mutation, which mutates the genes in the selected solutions. Either way will lead the GA to the solution direction.

2.4. Summary

This chapter introduces the concepts and the components of GA. The search strategies that GA used are also discussed. The exploration and exploitation search in GA have to be balanced (Eiben and Schippers, 1998) for a certain problem. Otherwise, the performance would be greatly affected. In general situation, if the computational resources are enough, the more explorations at the initialization step and more exploitations at evolution step would produce a better performance.

A healthy GA has to include five components. They have the different implementations for the different optimization problems. The fitness evaluation component is strategy independent, i.e. it is black box when the GA is operated. However, for the same optimization problem, developing an efficient fitness function is very important. The following chapters illustrate two different fitness functions for the same ray tracing problems, and compare the results.

Chapter 3

Methodology

Seismic ray bending methods are accurate ray tracing methods, by which the difficulties of shooting methods, e.g. rugose-salt-sediment interfaces and steeply dipping interfaces problems, are overcome. In order to design an efficient bending method, the subsurface velocity model must to be defined. In our research, the velocity model is designated as a layer-based model, in which the constant velocity is maintained between each layer. Each layer is described by a single value continuous function of x-offset and y-offset (as shown in Fig. 3.1). The ray path only bends at the layer interfaces and at the bottom reflector. Therefore, to simulate a ray path in the defined velocity model, one effective method is to construct the ray path by searching the bending points on the layers (reflectors). Based on Fermat's Principle and Snell's Law, two ray bending methods using GA are proposed in this chapter.

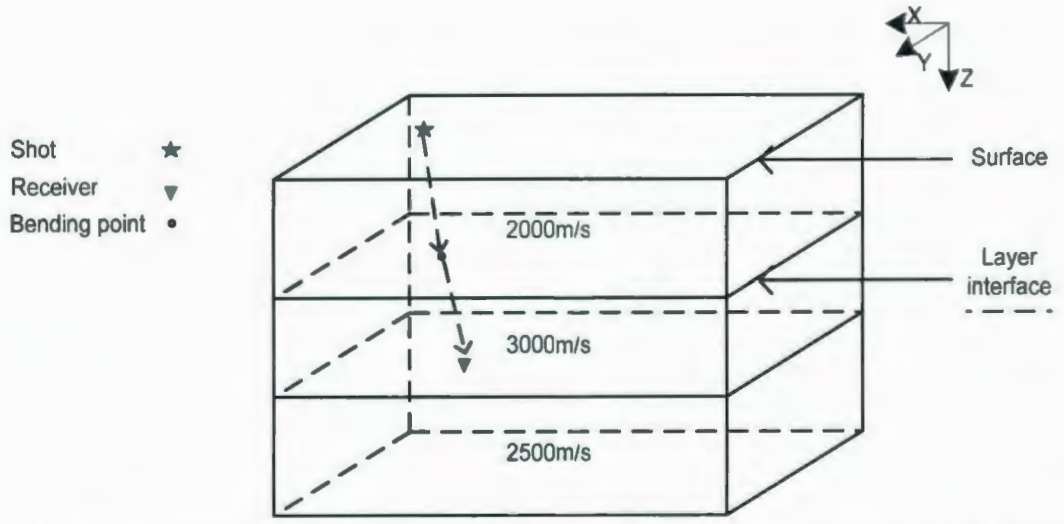

Fig. 3.1. The illustration of a ray path from a shot to a receiver in the assumed subsurface velocity model.

3.1. The MTTM

Based on Fermat's Principle, the first reflected arrival wave is recorded by the receiver, which is triggered from the shot along the ray path with the minimum travel time. The energy of the wave is also transmitted along this path and reflected on the velocity interface (Moser, 1991). From this theory, a MTTM can be developed using GA.

3.1.1. Fitness Function Description

In order to design a GA, the fitness function has to be derived initially. This fitness function works as a black box in GA and the rest of GA components associate with it but do not depend on it. The fitness function in MTTM is designed to calculate the travel time between the shot point and the receiver point.

For a given velocity model, the travel time of a ray segment (the ray between two bending points) is calculated by the distance between two points divided by the velocity, as shown in Fig. 3.2. The distance is computed by Pythagorean principle and the travel

time between the shot point and the receiver point are sum of the travel time of each ray segment.

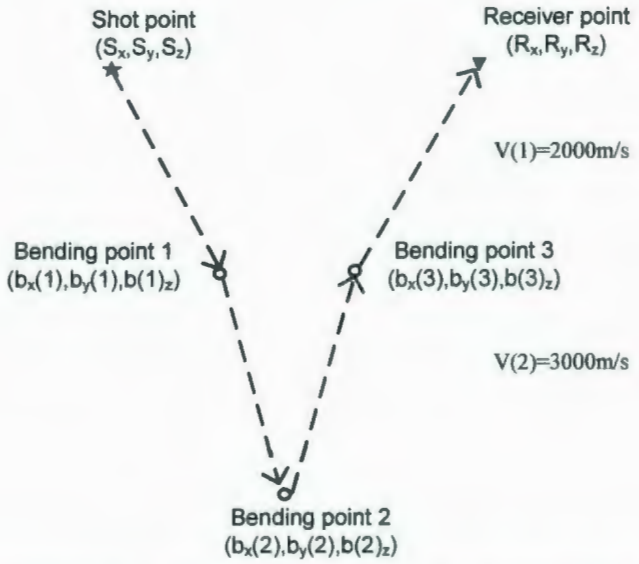

Fig. 3.2. The illustration of travel time calculation between two bending points.

The travel time function in 3D velocity model is given as equation 3.1,

$$T = \frac{\|S - b(1)\|}{V(1)} + \frac{\|R - b(l)\|}{V(N)} + \sum_{i=2}^{l-1} \frac{\|b(i) - b(i-1)\|}{V(j)}, j = 1, \ldots, N \qquad (3.1)$$

where T is the travel time, S is the shot point, R is the receiver point, $b(i)$ is the bending point i, notation $\| \quad \|$ is the distance between two points, l is the number of the bending points between the shot and receiver points, which is calculated by counting the number of wave reflections and $V(j)$ is the p-wave velocity of jth layer.

In equation 3.1, the coordinates of the shot and receiver points (S_x, S_y, S_z, R_x, R_y, and R_z) as well as the velocities ($V(j), j=1, \ldots, N$) are already defined. As the result of the function, the travel time is only sensitive to the bending points set and it is designed to be minimized. Thus, the equation 3.1 is reasonably designed to be the fitness evaluation

component in the GA. The input of this component is the bending points set and the output is the travel time.

3.1.2. Genetic Algorithm Design

Based on the fitness function, the rest of GA components can be developed.

3.1.2.1. Population

The individuals in the population are represented by the natural representation of the numerical optimization, real number vectors, in which the genotypes are the same as the phenotypes. Each individual contains three vectors, and each vector separately stores the x, y, and z coordinates of the bending points. The lengths of the vectors are equal to the number of bending points between the shot point and the receiver point.

The search space is defined as real value within the range of the layers descriptions of the velocity model. In the initial population, the individuals are assigned as the random points within all the possible bending points on all the layers. The population size depends on the available computational resources. However, with the crossover operation procedure, at least two individuals are required in the population. In our software implementations, after several experiments, the population size was defined as 40 individuals, which is enough for our optimization problems.

3.1.2.2. Selection Strategies and Replacement

The deterministic selection strategies are adopted in our GA design. In the parents selection, all individuals in the population are selected as the parents. The selection, mutation and crossover are executed together on the old population to produce a larger temporary population (Eshelman, 1990), which contains the parents individual and the

offspring individual. Each individual, including the parent and offspring, is evaluated using the fitness function. In the replacement selection, the same number as the population size of elitism individuals are chosen. The old population is replaced by the chosen individuals as the new population in the next iteration.

3.1.2.3. Individual Evolution

In the process of individual evaluation, the uniform mutation and the arithmetic crossover are performed together to evolve the parents and to generate offspring. With each pair of parents, there is an 80% chance of being mated using the uniform mutation and a 20% chance of being mated using the arithmetic crossover.

For an parent individual, which contains the coordinates of a bending points set, the uniform mutation will generate two vectors and will add these new vectors to the x, y coordinates vectors of the parent individual. The new z coordinates vector is calculated based on the established layer function previously derived. Each new individual is produced as an offspring.

Fig. 3.3. An Example of uniform mutation.

The contents of these two new vectors are based on uniform random real values within a certain range, called step size. The step size in our design will evolve as the generation

number increases (Iorio and Li, 2002). It is initialized as ten percent of the x offset and y offset range and decreases with the programming run. Using this genetic operator, the algorithm searches the points around a parent point and the precision of the search is enhancing with iteration. An example of uniform mutation for x coordinates of the parent is illustrated in Fig. 3.3.

In the arithmetic crossover, two new vectors (Ve_1 and Ve_2) are generated. The contents of the vectors are real numbers from range 0 to 1 with uniform probability distribution. The offspring are produced by:

$$Offspring_x(i) = P_{1x}(i) \bullet Ve_1(i) + P_{2x}(i) \bullet (1 - Ve_1(i)),\ i = 1,....,l$$
and, (3.2)
$$Offspring_y(i) = P_{1y}(i) \bullet Ve_2(i) + P_{2y}(i) \bullet (1 - Ve_2(i)),\ i = 1,....,l$$

where $P_{1x}(i)$, $P_{2x}(i)$, $P_{1y}(i)$, and $P_{2y}(i)$ represent x, y coordinates of the parents, $Ve_1(i)$ and $Ve_2(i)$ are the new vectors, i$\in$ $\{1,\ldots l\}$ is the index of vector value, l is the number of the bending points between the shot and receiver points. We use the arithmetic crossover to search the points between parents.

3.1.2.4. Termination Criteria

In the MTTM, no threshold would be decided for the program termination. Therefore, the termination criterion is set to a tolerated time period. In our experiments, we set the termination criterion as either 100 GA generations or when all the individuals in the population converge to the same value.

The MTTM is accomplished based on the above GA setups. The method simulates the ray propagation and is used to calculate the travel time between two given points in the assumed earth model. Using this method, seismic modeling and Pre-stack Kirchhoff

migration are implemented. The above setting is reasonably accurate for the following modeling and migration problems. Moreover, as one can notice that by neglecting all the operations on the y coordinates, the algorithm can also be transformed to the 2D seismic ray tracing method.

3.2. The MESLM

When a ray bends at a velocity interface, the reflected ray path can be determined based on Snell's Law. From this theory, the shooting ray tracing method is designed to simulate the ray propagation between the shot point and the receiver point in the subsurface model. However, an error may be caused by Snell's Law at receiver point. In order to produce an accurate result and minimize the error, the MESLM is proposed to distribute the error at the receiver along all the bending points on the layers and these errors can be managed using GA.

3.2.1. Fitness Function Description

The fitness function in the MESLM is designed to calculate the error caused by Snell's Law at each bending point. As shown in Fig. 3.4, between the shot point S and the receiver point R, two bending points (A and B) are randomly initialized on the layers. The ray paths SA, AB and BR are thereby random ray paths. Based on Snell's Law and the random incidence ray paths and the reflected ray paths AB' and BR' are determined. The errors of Snell's Law are defined as the difference between the random ray paths and reflected ray paths. Therefore, the input of the fitness function is a bending points coordinates set and the output is the sum of the errors of Snell's Law. If the sum of errors equal to zero, the Snell's paths are found.

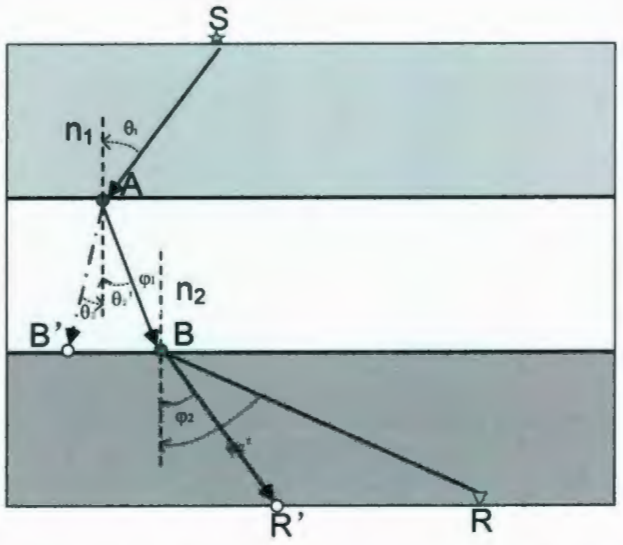

Fig. 3.4. The illustration of the errors caused by Snell's Law.

The error may be represented by the different forms, such as the angle error or the distance error. In the following sections, the derivations of the angle fitness function and the distance fitness function are provided. Moreover, based on the computational complexity, one of the fitness functions is selected to apply in the MESLM.

3.2.1.1. The Angle Fitness Function Derivation

As shown in Fig. 3.5, the points $A(x_1, z_1)$, $B(x_2, z_2)$ and $C(x_3, z_3)$ are defined random bending points. The layer l is described as $z=f(x)$, a single value continuous function of x-offset. Based on the incidence angle θ_1, velocity V_1 and V_2, the reflection angle θ_2 is calculated. Moreover, using trigonometric functions, β, the angle between path BC and normal, is also calculated. Thus, the error is represented by the difference between θ_2 and β. The following equations provide the detail of the derivation of the error function for a single bending point.

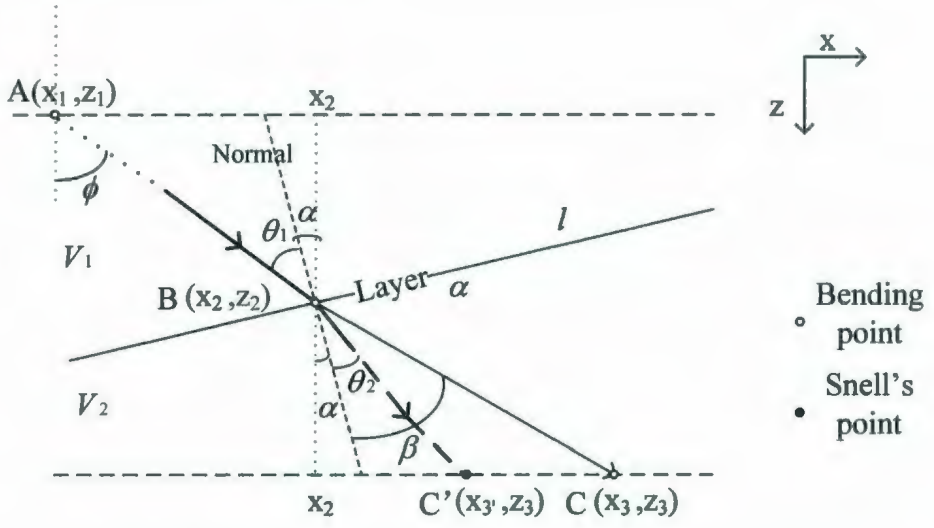

Fig. 3.5. The illustration of the derivation of the angle fitness function.

$$\theta_2 = \arcsin(\frac{\sin(\theta_1)\bullet V_2}{V_1}),\quad \theta_1 = \phi + \alpha,\quad \phi = \arctan(\frac{(x_2 - x_1)}{(z_2 - z_1)}),\quad \alpha = \arctan(f'(x))$$

$$\beta = \arctan(\frac{(x_3 - x_2)}{(z_3 - z_2)}) - \alpha$$

$$Error = abs(\theta_2 - \beta)$$

$$Error = abs\left(\arcsin\left(\frac{\sin\left(\arctan\left(\frac{(x_2 - x_1)}{(z_2 - z_1)}\right) + \arctan\left(f'(x_2)\right)\right)\bullet V_2}{V_1}\right) - \arctan\left(\frac{(x_3 - x_2)}{(z_3 - z_2)}\right) + \partial\right) \qquad (3.3)$$

By summing up the error of each bending point, a function is derived. The error has the similar gradient as the difference between the random bending point and the Snell's point, and therefore, when the difference is small, the error is also a small value. Thus, this function is adoptable as a fitness evaluation function that is used in GA for minimization.

However, many trigonometric functions are applied in the fitness function. They cause the high degree computational complexity when the fitnesses are evaluated in GA.

Therefore, other fitness functions are investigated, which are not so computational intensive.

3.2.1.2. The Distance Fitness Function Derivation

As shown in Fig. 3.4, based on the emitted ray, the Snell's points, B' and R' can be calculated. The distances between the Snell's points and the random points are sensitive to the random ray path, and also if the sum of the distances is zero, the random ray path is a Snell's ray path. Thus, a fitness function could be also represented by the distance function.

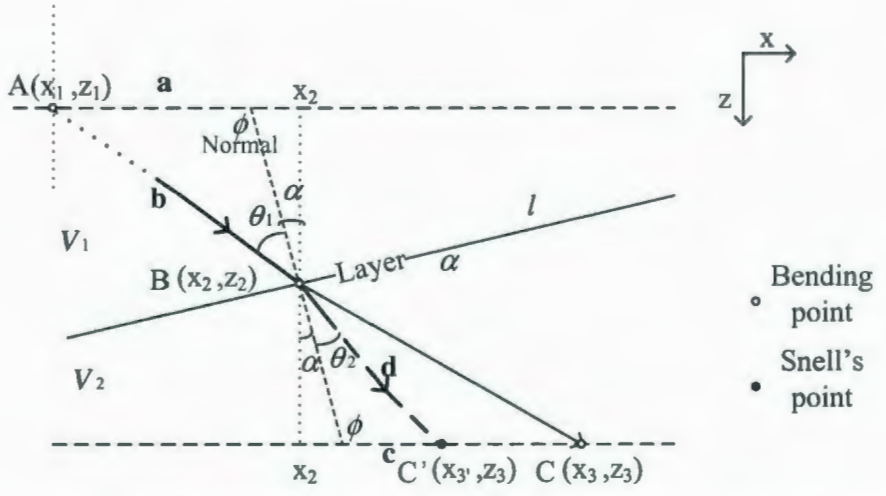

Fig. 3.6. The illustration of the derivation of the distance fitness function.

As shown in Fig. 3.6, the distance between the first random point $A(x_1, z_1)$ and the normal intersected point with layer $z=z_1$ is represented by **a**; the distance between the Snell's point $C'(x_{3'}, z_3)$ and the normal intersected point with layer $z=z_3$ is represented by **c**; the distance between the point A and the second random point $B(x_2, z_2)$ and the distance between point B and point C' are represented by **b** and **d** respectively. Using trigonometric function and Pythagorean principle, **a**, **b**, **c** and **d** are calculated by the following equations respectively.

$$
\begin{aligned}
a &= (x_2 - x_1) + Tan(\alpha) \bullet (z_2 - z_1) \\
b &= \sqrt{(x_2 - x_1)^2 + (z_2 - z_1)^2} \\
c &= (x_{3'} - x_2) + Tan(\alpha) \bullet (z_3 - z_2) \\
d &= \sqrt{(x_2 - x_{3'})^2 + (z_2 - z_3)^2}
\end{aligned}
\tag{3.4}
$$

Based on Sine Law:

$$
\begin{aligned}
\frac{a}{\sin\theta_1} = \frac{b}{\sin\phi} &\Rightarrow \sin\theta_1 = \frac{a}{b} \bullet \sin\phi \\
\frac{c}{\sin\theta_2} = \frac{d}{\sin\phi} &\Rightarrow \sin\theta_2 = \frac{c}{d} \bullet \sin\phi
\end{aligned}
\tag{3.5}
$$

Then using Snell's Law we get:

$$
\frac{a}{b} \bullet V_2 = \frac{c}{d} \bullet V_1 \tag{3.6}
$$

Substituting equation (3.4) into (3.6), we get:

$$
\frac{(x_2 - x_1) + f(x_2)' \bullet (z_2 - z_1)}{V_1 \bullet \sqrt{(x_2 - x_1)^2 + (z_2 - z_1)^2}} + \frac{(x_2 - x_{3'}) + f(x_2)' \bullet (z_2 - z_3)}{V_2 \bullet \sqrt{(x_2 - x_{3'})^2 + (z_2 - z_3)^2}} = 0 \tag{3.7}
$$

Instead of calculating the Snell's point C'($x_{3'}$, $z_{3'}$) and then calculating the distance between C' and C, the error function (equation 3.8) is produced by directly substituting the coordinates of third random point C(x_3, z_3) into equation (3.7).

$$
Error = abs\left(\frac{(x_2 - x_1) + f(x_2)' \bullet (z_2 - z_1)}{V_1 \bullet \sqrt{(x_2 - x_1)^2 + (z_2 - z_1)^2}} + \frac{(x_2 - x_3) + f(x_2)' \bullet (z_2 - z_3)}{V_2 \bullet \sqrt{(x_2 - x_3)^2 + (z_2 - z_3)^2}} \right) \tag{3.8}
$$

$$Total\_Error = abs\left(\frac{(x_1 - Sx) + f_1(x_1)'\bullet(z_1 - Sz)}{V_1\bullet\sqrt{(x_1 - Sx)^2 + (z_1 - Sz)^2}} + \frac{(x_1 - x_2) + f_1(x_1)'\bullet(z_1 - z_2)}{V_2\bullet\sqrt{(x_1 - x_2)^2 + (z_1 - z_2)^2}}\right) +$$
$$abs\left(\frac{(x_1 - Rx) + f_1(x_1)'\bullet(z_1 - Rz)}{V_1\bullet\sqrt{(x_1 - Rx)^2 + (z_1 - Rz)^2}} + \frac{(x_1 - x_{1-1}) + f_1(x_1)'\bullet(z_1 - z_{1-1})}{V_2\bullet\sqrt{(x_1 - x_{1-1})^2 + (z_1 - z_{1-1})^2}}\right) + \quad (3.9)$$
$$\sum_{i=2}^{I-1} abs\left(\frac{(x_i - x_{i-1}) + f_i(x_i)'\bullet(z_i - z_{i-1})}{V_{i-1}\bullet\sqrt{(x_i - x_{i-1})^2 + (z_i - z_{i-1})^2}} + \frac{(x_i - x_{i+1}) + f_i(x_i)'\bullet(z_i - z_{i+1})}{V_i\bullet\sqrt{(x_i - x_{i+1})^2 + (z_i - z_{i+1})^2}}\right)$$

It can be proved that the error function (equation 3.8) is a single value function and sensitive to the input random points set. An example of an error function plot is shown in Fig. 3.7. In this example the incidence ray is fixed, and the Snell's point is calculated at coordinates (x_3'=50, z_3'=190). The layer z_3 is described as $z_3 = 10\bullet sin(0.1\bullet x) + 200, x \in \{1.....100\}$. The error decreases as the random point approaches the Snell's point. Therefore, this error function is a suitable fitness function for GA minimization. Summing up all of the errors for the entire bending points set, the distance errors function is produced in equation 3.9. By comparing to the angle errors function (equation 3.3), the computational performance is greatly increased.

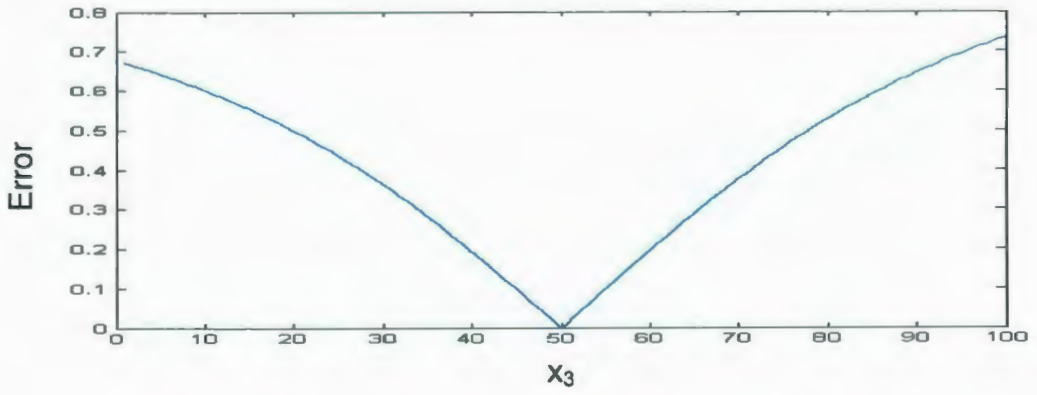

Fig. 3.7. The illustration of fitness function curves versus distance.

3.2.2. The Genetic Algorithm Design

The designs of the GA components for the MESLM are similar to the MTTM. However, because the developed fitness functions are different, in order to increase the runtime performance, the termination criteria are modified for the MESLM.

In the MESLM, the suitability of an individual could be decided before finishing the evaluation. Because, if the error of one bending point is a large value, for instance, the distance between B and B' is too big, this individual ray path is absolutely not a Snell's path. This method of ray rejection greatly improves the program runtime. The termination criterion is also clear in the MESLM, in which the total errors are minimized to zero. Therefore, it may terminate the search with a successful ray path before the number of generations reaches the iteration limit.

3.3. Summary

The MTTM and the MESLM, two layers based seismic ray bending methods, have been proposed in this chapter. The fitness functions derivations and the GA components designs of both methods are discussed in detail. Using either of methods, the limitations of other seismic ray tracing methods are overcome. In the MESLM, two fitness functions are derived. The angle fitness function may be further employed by another optimization algorithm, but, based on the computational complexity, the distance fitness function is applied in our research.

The MTTM has the lower computational complexity, which is very important for the scientific computing. Therefore, two applications, seismic modeling and Pre-stack

Kirchhoff migration, are implemented based on this method. Furthermore, the MESLM is validated in the following chapter.

Chapter 4

Applications and Results Verification

There are many applications of seismic ray tracing in the seismic exploration and the most of them are the inversion problems. In the practical seismic exploration, the seismic survey produces the wave forms and travel times information, which are recorded by the geophones. In order to understand the subsurface structure, the seismic ray tracing methods are the most frequently employed to simulate the ray propagation in the subsurface and calculate the travel time.

As the most important applications of seismic ray tracing, seismic modeling and migration are essential stages in the seismic exploration (Dong *et al*, 1991). They are effective methods for the seismic data interpretation and the subsurface image generation. In both of seismic modeling and migration, the seismic ray tracing methods are applied to calculate the ray travel time between the shot point and the receiver point. The synthetic seismograms are generated in the modeling process and the subsurface image is reconstructed in the migration (Carcione *et al.*, 2002).

4.1. Seismic Modeling

4.1.1. Seismic Modeling Theory

From the seismic survey, the seismograms are generated. The information on the seismograms contains the travel times of all shot-receiver pairs and the received wave energies. Based on this information, the seismic data are processed by the geophysicists and the subsurface structure is interpreted. For the research in the lab, in order to better understand the seismic data processing, the synthetic subsurface velocity models are usually employed. Seismic modeling is a process that simulates the seismic ray propagation in the synthetic velocity models and produces the synthetic seismograms. The synthetic seismograms could be applied to match to the real seismograms (Bording and Lines, 1997), which is a criterion to verify the correctness of the seismic data processing.

As the same as the real seismic survey, in the seismic modeling, usually, the receivers are laid out on the surface of the velocity model and sample the seismic waves with a certain interval also monitor the shot activities. The shot is moved from one position to others on the surface of the model and explodes to generate the seismic waves. When the shot explodes the seismic waves at one position, the waves propagate through the velocity model and reflect at the velocity interfaces. After a time period, when the waves arrive at receiver, they are recorded and recorded time traces build up the expected synthetic seismogram.

The objectives of this chapter in our research are to implement the applications of the seismic ray tracing methods, seismic modeling and Pre-stack Kirchhoff migration, and to

verify the proposed methods. Thus, the modeling processes are simplified and the several assumptions are considered.

4.1.2. Seismic Modeling Methodology

In our research, the p-wave, a type of elastic waves, is the considered seismic wave. The synthetic velocity models are represented by the layer based models. The process of the shot movements from one position to others on the surface is considered as many shot points lying out at the different positions. To simplify the modeling implementations, the energy attenuation of the ray propagation through the bending process is not considered and the multiple reflections phenomena are neglected. Therefore, the frequency of wave forms that receiver recorded are the same wave forms as the shot generated, and the differences in time of waves arriving at the receiver are caused by the reflections at the different velocity interfaces (layers). In order to mathematically represent the receiver sampling process, the numbers of the samples are calculated by the travel time divided by sample interval and the sampled wave forms are generated by convolving the reflection coefficients with the source function.

The travel time from the shot point to the receiver point is produced by employing the proposed MTTM. The inputs of our method are the shot and receiver positions as well as the velocity model, the outputs are the minimum travel time and a bending points set, which constructs a ray path. An example of the ray path, which is produced by the MTTM, in the 2D polynomial layers velocity model is illustrated in Fig. 4.1, which shows that the produced ray path is close to the Snell's path.

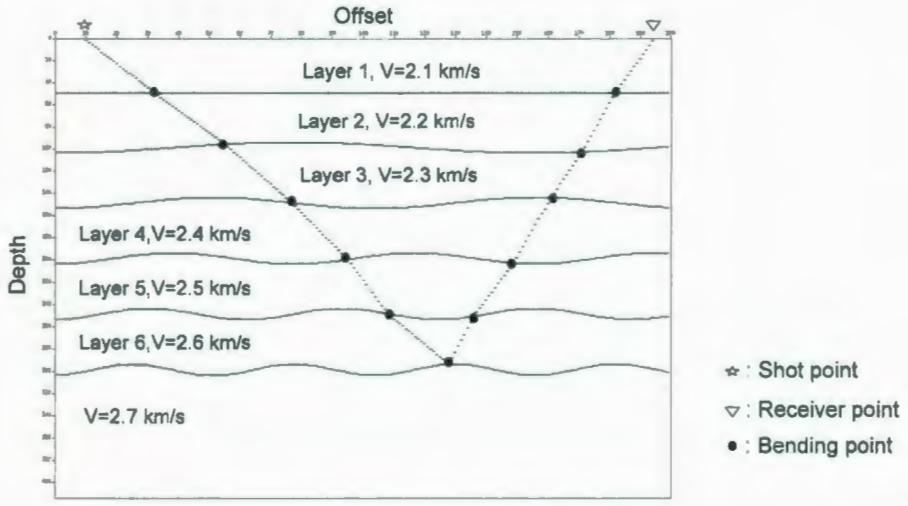

Fig. 4.1. An illustration of a ray path with the minimum travel time of one shot- receiver pair in the assumed velocity model.

From the calculated travel time, the number of sample is generated by:

$$Ns(i) = \frac{mtime(i)}{dt} - 0.5, \; i = 1,....,N \tag{4.1}$$

where $Ns(i)$ is the number of samples, $mtime(i)$ is the minimum travel time, dt is the sampling interval, N is the number of layers in the model, and 0.5 is used to produce the closest integer value for the number of samples.

Without considering the energy attenuation, the reflection coefficient of each reflector, which determines the amplitude of the reflected wave, is calculated by:

$$R(i) = \frac{\rho(i+1)\bullet V(i+1) - \rho(i)\bullet V(i)}{\rho(i+1)\bullet V(i+1) + \rho(i)\bullet V(i)}, \; i = 1,....,N \tag{4.2}$$

where $R(i)$ is the reflection coefficient of i^{th} layer, $V(i)$ is the velocity of i^{th} layer, $\rho(i)$ is the density of i^{th} layer and it is assumed to be constant value, and N is the number of layers in the model. The reflectivity series, an unfinished synthetic seismogram, which contains the peaks in the sampled position, is generated by:

$$reflectivity\_series(j)=\begin{cases} R(i), & j == Ns(i) \\ 0, & Others \end{cases}, j=1,....,Ns(N),\ i=1,....,N \qquad (4.3)$$

The integrated synthetic seismogram is accomplished by convolving the reflectivity series with the source function. The flow chart of modeling process is shown in Fig. 4.2.

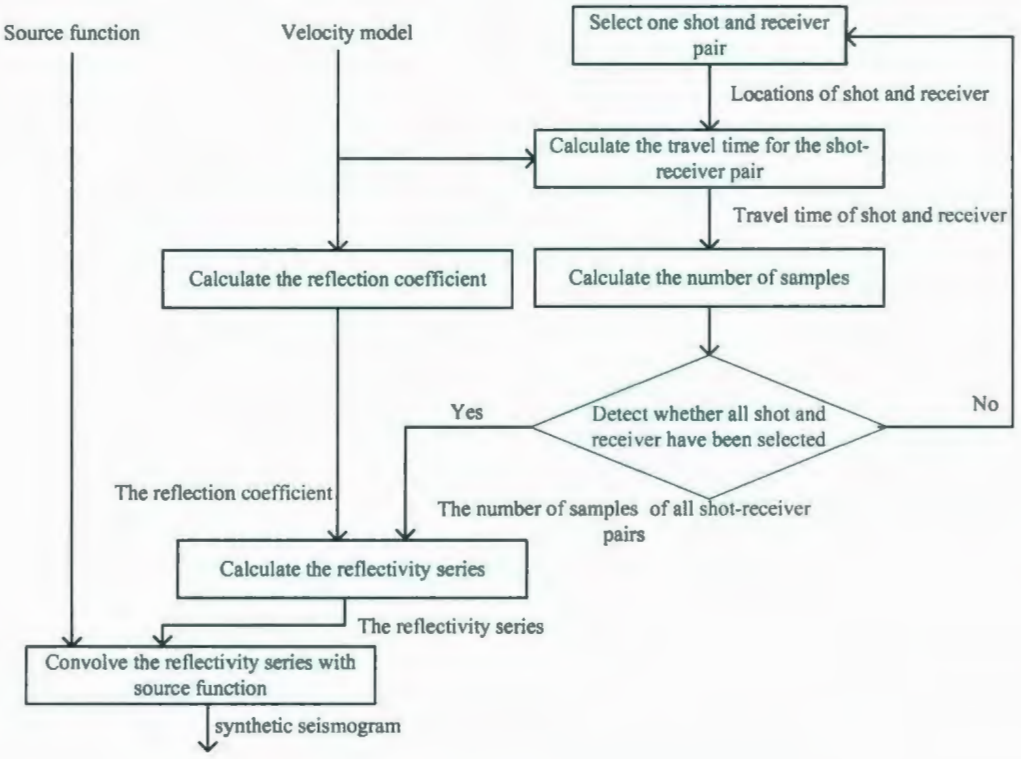

Fig. 4.2. The flow chart of modeling process.

4.1.3. Seismic Modeling Experiments and Results

As shown in Fig. 4.3, two 2D velocity models and two 3D velocity models are introduced in the experiment. The parameters of 2D and 3D models are listed in Table 4.1 and Table 4.2 respectively. In each model, the p-wave velocities are distinguished by the different layers and set to constant value in the same layer.

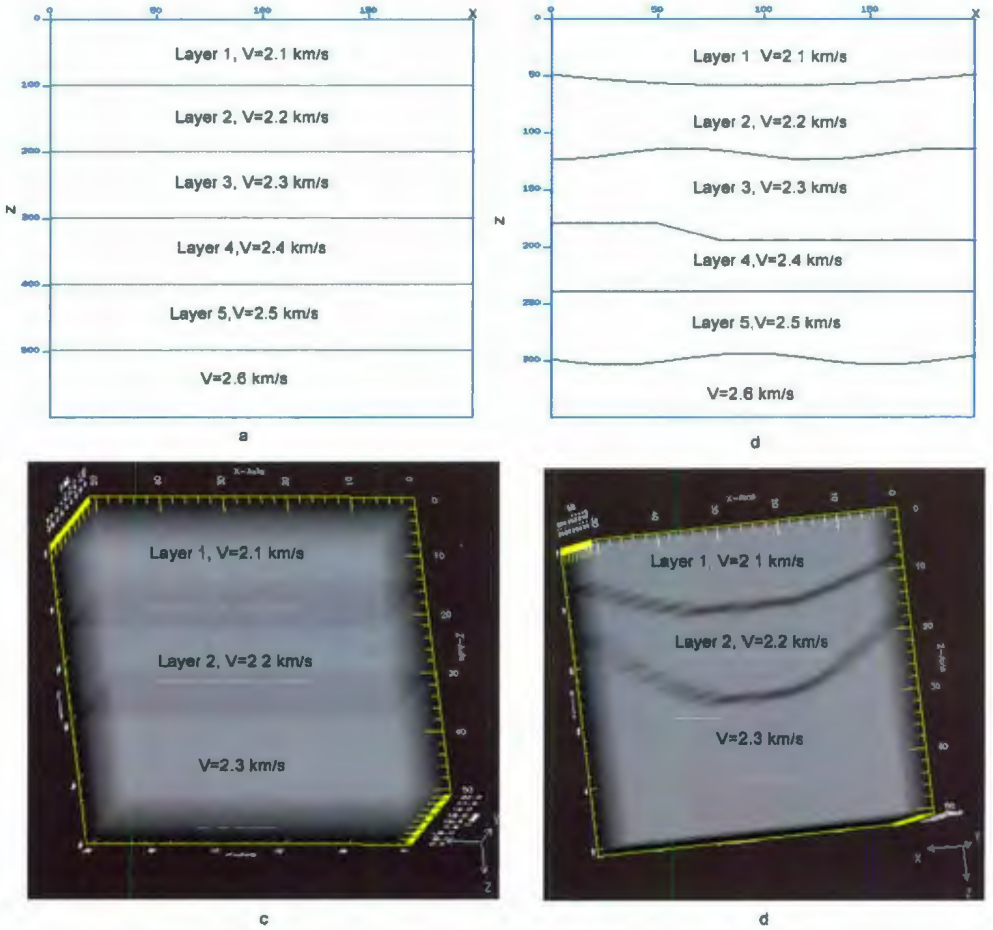

Fig. 4.3. The structures of adopted velocity models.

Table 4.1. The parameters of 2D experimental models.

Model	Layer	Velocity (Km/s)	Depth (m)	x-offset (m)
a	1	2.1	600	200
	2	2.2		
	3	2.3		
	4	2.4		
	5	2.5		
b	1	2.1	400	200
	2	2.2		
	3	2.3		
	4	2.4		
	5	2.5		

Table 4.2. The parameters of 3D experimental models.

Model	Layer	Velocity (Km/s)	Depth (m)	x-offset (m)	y-offset (m)
c	1	2.1	50	50	20
	2	2.2			
d	1	2.1	50	50	20
	2	2.2			

In the 2D models, 20 shots and 50 receivers are distributed on the flat 1D surface, with shot spacing at 10 meters and receiver spacing at 4 meters. In 3D models, 40 shots and 40 receivers are laid on the 2D flat surface, with shot and receiver spacing at 5 meters in x-coordinates and 5 meters in y-coordinates (shown in Fig. 4.4).

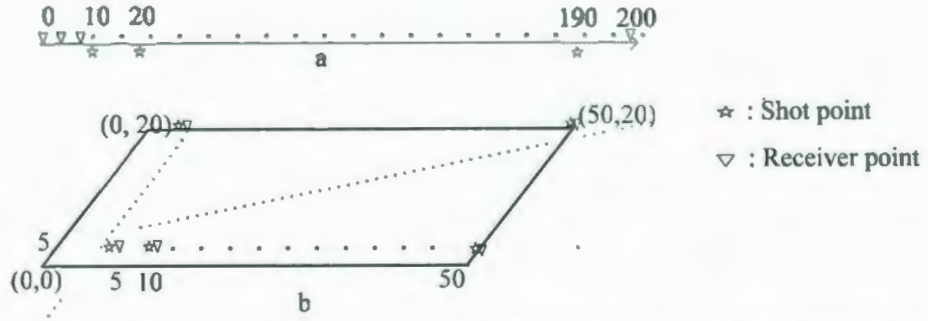

Fig. 4.4. The lay out of the shots and the receivers for 2D and 3D models.

The 50Hz Gaussian pulse is applied as the source function. Based on the Nyquist-Shannon sampling theorem, in order to better reconstruct the analog wave signal, the sampling interval is set to 0.01 second (1/ (2•50)). The source function and its amplitude spectrum are shown in Fig. 4.5.

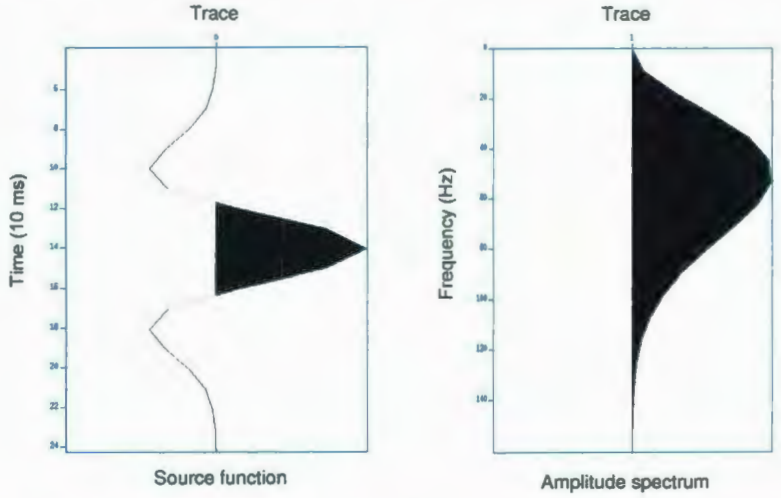

Fig. 4.5. The source function and its amplitude spectrum.

The synthetic seismograms of adopted velocity models are produced by the implemented seismic modeling. The Fig. 4.6 shows two examples of traces of receivers in the 2D model b and the 3D model d. The records from a certain shot to all of the receivers are illustrated. From the initial interpretation of the synthetic seismograms, the proposed the ray tracing methods are primarily verified. All the seismograms are employed into Pre-stack Kirchhoff migration, which further validate the proposed methods.

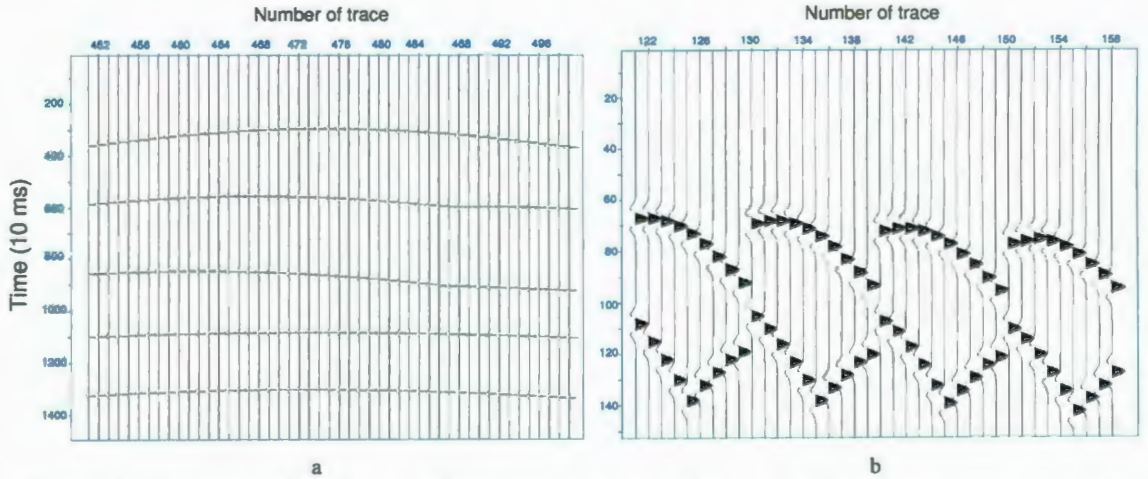

Fig. 4.6. The example of a set of receiver traces in the synthetic seismogram. .a is the traces of 2D model b, b is the traces of 3D model d.

4.2. Pre-Stack Kirchhoff Migration

4.2.1. Pre-Stack Kirchhoff Migration Theory

The seismogram provides the travel time information and the energies of waves reflected. Based on the known information, the migration process reconstructs a structure of the subsurface model (Bleistein, 1999). This structure is used to represent the model, which produces the seismogram.

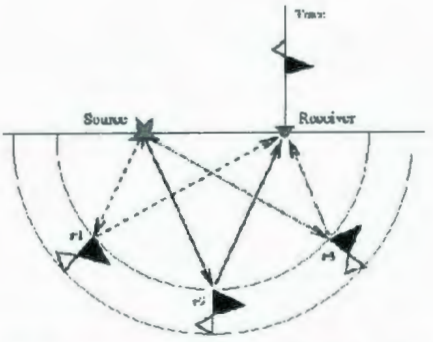

Fig. 4.7. The illustration of Pre-stack Kirchhoff migration theory.

In the seismogram, for any receiver record which is represented by a trace shown in Fig. 4.7, if there is a value at time t, then the model should have an elliptical mirror (layer) focused at the locations of the shot point and the receiver point (Bleistein, 1999). This is because only this model produces the required data, namely, no received signal except when the shot point and the receiver point are in the focus of the semi-ellipse. This observation plus the superposition principle suggests an algorithm (Buske, 1999) for making earth images, which is the basic idea of Pre-stack Kirchhoff migration. An example is shown in Fig. 4.8.

This example illustrates Pre-stack Kirchhoff migration results from the records of five receivers, as one can notice that the bottom of each semi-ellipse lies along a line that could be the line of reflector for observed plane waves. Thus, in Pre-stack Kirchhoff

migration, instead considering the p-wave is reflected at a point in the model, each being interpreted as coming from a semi-ellipse layer. Adding the layers yields a flat reflector.

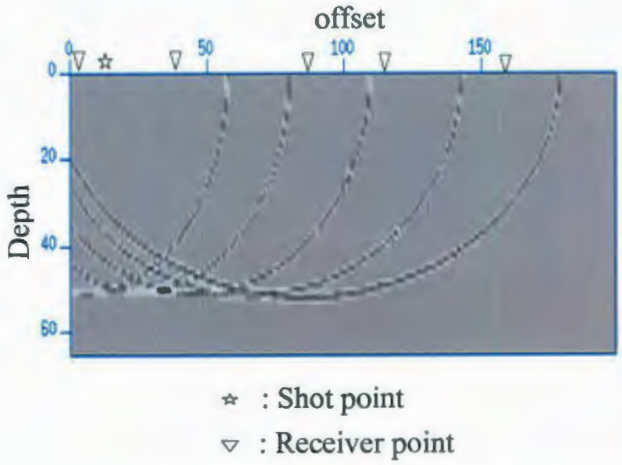

Fig. 4.8. An example of Pre-stack Kirchhoff migration result.

Typically, the Pre-stack Kirchhoff migration is applied to the time migration and depth migration. If the velocity model is unknown, the time migration is employed to reconstruct the shapes of the layers of the model. The simplest approach to implement time migration is that draw the semi-ellipses in an empty image, the focuses of the semi-ellipses are on the position of the shot point and the receiver point, the radius are the records in one time trace. Summing up all the images for all the traces, the model image is produced. The reconstructed model image only contains the primary information of the shape of the layers, but not others. Therefore, the Pre-stack Kirchhoff time migration is usually used to preprocess the seismic data.

Pre-stack Kirchhoff migration is employed as the depth migration algorithm when the velocity model has been presumed. In the depth migration results, the information on both of the locations and shapes of the velocity layers are obtained. In our research, the adopted four models in the modeling process are applied in the Pre-stack Kirchhoff depth

migration and, based on the produced synthetic seismograms, the structure of the models are reconstructed.

4.2.2. Pre-stack Kirchhoff Migration Methodology

In our Pre-stack Kirchhoff depth migration implementation, two steps are applied. The first step, which generates the time tables for all the shot points and the receiver points, operates the known velociy model. The second step handles the matching from the seismogram into the structure image.

In the first step, because the ray reflected points are unknown, every grid point in the velocity model is considered as a reflected point. The time table for a shot-receiver pair is built up by calculating the travel times of the ray propagation from the shot point to the receiver point, which reflected at each grid point. The time table calculation also can be separated into two segments: shot table calculation and receiver table calculation. In the shot table calculation, the travel times from the shot point to each grid point are produced by proposed ray tracing method, in which the shot point is processed as the first given point and each grid point is substituted as the second given point. Similarly, the receiver table is produced. Adding the shot table with the receiver table, a time table for this shot-receiver pair is generated. An example of the time table contour is illustrated in Fig. 4.9.

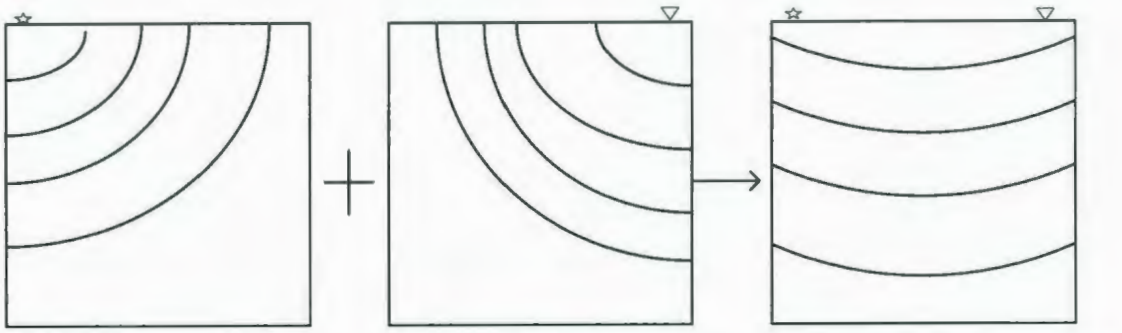

Fig. 4.9. An illustration of a travel time table calculation process.

The time tables of all the shot receiver pairs are accomplished in the first step. In the second step, in order to match the seismogram with the time table, the time traces are interpreted from the number of samples into the time. The grid points, which produce the semi-ellipse layers, are searched by matching the time table with the time traces. The model image is reconstructed by setting the trace value in the matched grid points coordinates of the image table. The pseudo code for the second step of Pre-stack Kirchhoff migration is given in the Fig. 4.10.

```
Do si=1 to the_number_of_shot_points
   Do ri=1 to the_number_of_receiver_points
      Do xi=1 to x_offset
         Do y=1 to depth
            Do t=1 to number_of_sample_in_the_trace
               If ( trace(si, ri, t) not equal to 0 ) then
                  Time=t*dt+0.5*dt      \\ interpret the number of sample into time
               Endif
               If ( (Timetable(si, ri, xi, zi) – Time) equal to 0) then   \\search the matched grid points
                  Image (xi, zi)=Image(xi, zi)+trace(si, ri, t)
               Endif
            End do
         End do
      End do
   End do
End do
```

Fig. 4.10. The pseudo code for Pre-stack Kichhoff migration matching process.

4.2.3. Pre-stack Kirchhoff Migration Experiments and Results

The adopted four models in the modeling process are employed as the experimental models in Pre-stack Kirchhoff migration. The number of shot points, receiver points, the lay out of the shot, receiver points and the sample interval *dt* are the same setups as the modeling process. The MTTM is applied to calculate the time table. The results of the Pre-stack Kichhoff migration are shown in Fig. 4.11.

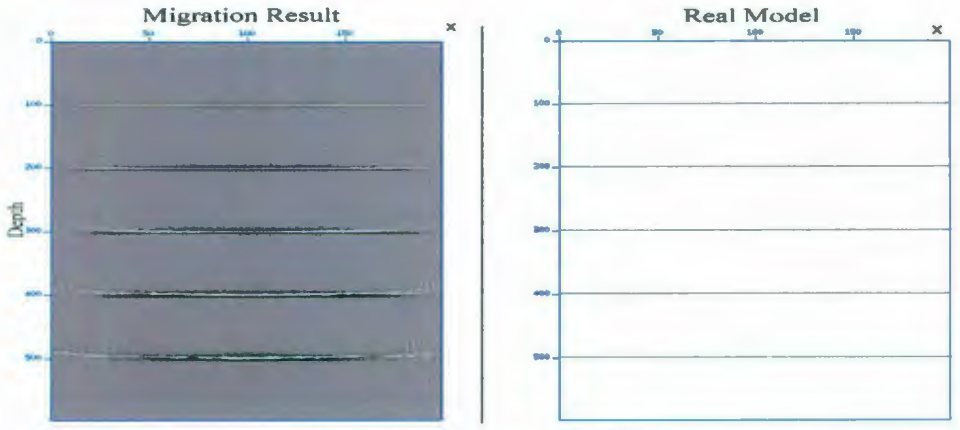

Fig. 4.11a. The result of the Pre-stack Kirchhoff Migration for Model a.

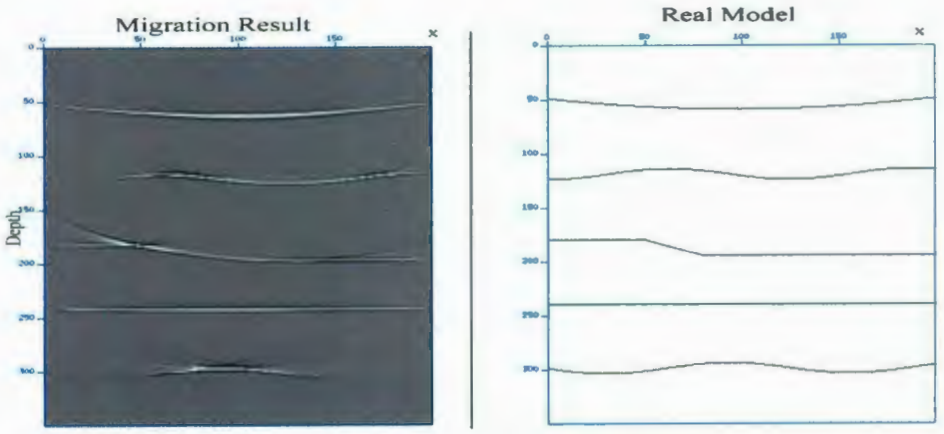

Fig. 4.11b. The result of the Pre-stack Kirchhoff Migration for Model b.

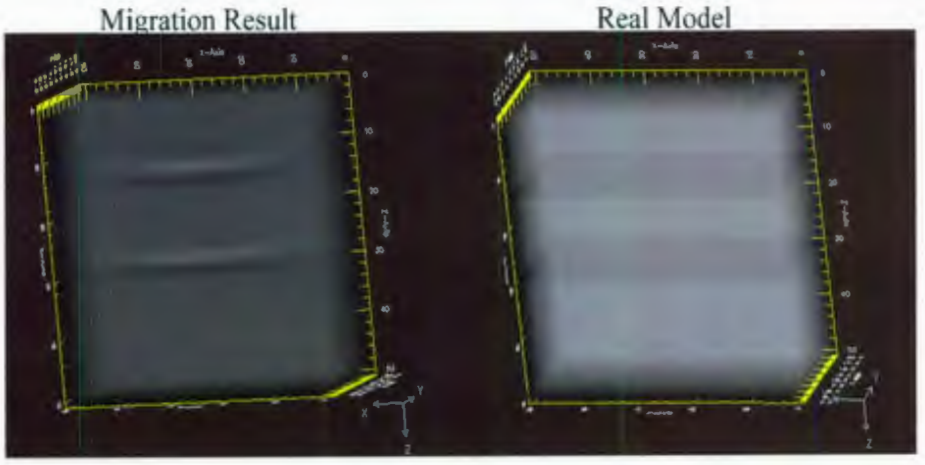

Fig. 4.11c. The result of the Pre-stack Kirchhoff Migration for Model c.

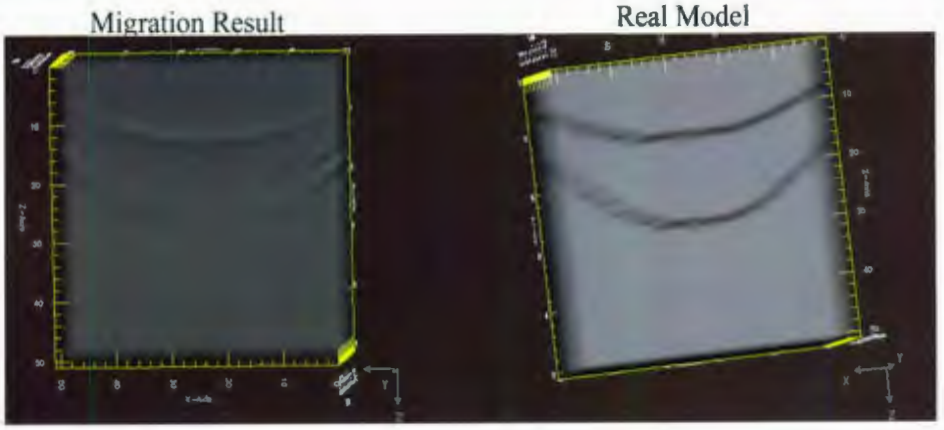

Fig. 4.11d. The result of the Pre-stack Kirchhoff Migration for Model d.

As shown in the results, based on the MTTM, our Pre-stack Kirchhoff migration fully recovers the synthetic models for both of 2D and 3D flat layer, 2D and 3D polynomial layer. All of the interfaces are visible, and approximate the original shapes and original depths.

To further testify the practicability of our Pre-stack Kirchhoff migration process and verify the proposed the MESLM, another model with more complex structures, which is closer to real subsurface model, is employed. The travel time calculation in the modeling process is implemented by the MESLM. In the Kirchhoff migration process, the MTTM is utilized. 50 shot points and 100 receiver points are lay out at the surface, with shot spacing at 20 meters and receiver spacing at 10 meters. The source function and the sample interval are unchanged. The structure of the complex model and the Pre-stack Kichhoff migration result are shown in Fig. 4.12.

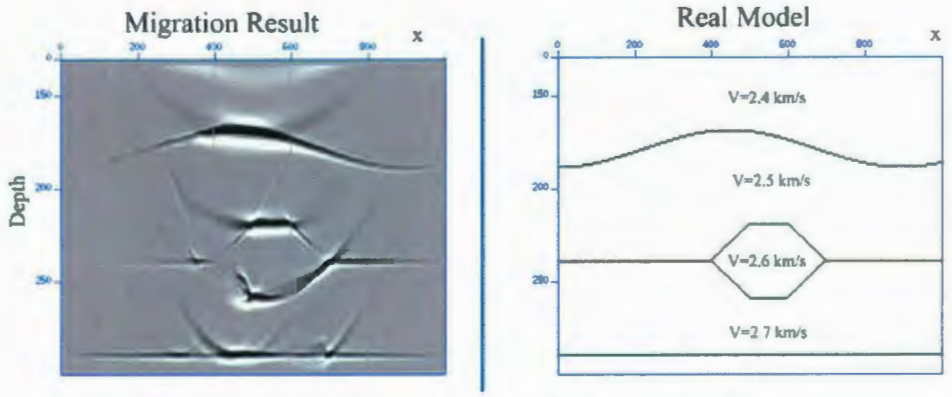

Fig. 4.12. The complex model structure and the Kirchhoff migration result.

As shown, our migration does a good job regenerating the complex subsurface structure. The box structure in the bottom of the model has even been clearly reconstructed. Therefore, we are fairly confident our algorithms are effective ray tracing methods, which are worth being implemented as a computer hardware chip to increase the calculation efficiency.

4.3. Software Parallel Design

The MTTM and the MESLM are ray bending methods implemented by the GAs. In order to detect a proper ray path, which is transmitted from a shot point, reflected on the layers of velocity model and received at a receiver point with the minimum travel time, lots of other candidate ray paths are calculated by the GA. In the processes of proposed seismic modeling and Pre-stack Kirchhoff migration, all the ray paths from all the shot points to all the receiver points have to be determined to provide travel times information, with very high computational complexities. Those applications are very time consuming processes, so, in order to increase the computing performance, the parallel programming technique, MPI, are applied into the modeling and migration processes.

MPI is a specification for an application programming interface (API), which is applied in the computer clusters and supercomputers. In the specified resources, each processor has a private memory space and individually finishes the assigned workloads. The processors communicate with each others when the workloads are finished by passing the messages between each independent memory space (Gropp *et al.*, 1994). The processors communications occupy extra processing time, which may be longer than the data processing time. Thus, the trade-off between the distributed workloads and the communications has to be considered in a mature MPI parallel design (Bording, 1996).

Seismic modeling and Pre-stack Kirchhoff migration programs are parallely implemented with MPI specification in the Atlantic Computational Excellence Network (ACEnet), Placentia cluster, which has total 464 cores and 4GB RAM per core. In order to reduce the frequency of the communications between each processor, the modeling process is decomposed by the shot points (Phadke *et al.*, 2000). In this problem decomposition

strategy, each available processor is assigned the same workload for the calculation of the travel times from one or many shots points to all of the receivers points. The workflow of modeling process is shown in Fig. 4.13. Similarly, because of the methodology of the Pre-stack Kirchhoff migration, separated calculation for the time tables of shots and receivers, the migration process is decomposed into the shot and receiver points. Employing this strategy, the parallel programming provides a nearly linear speedup.

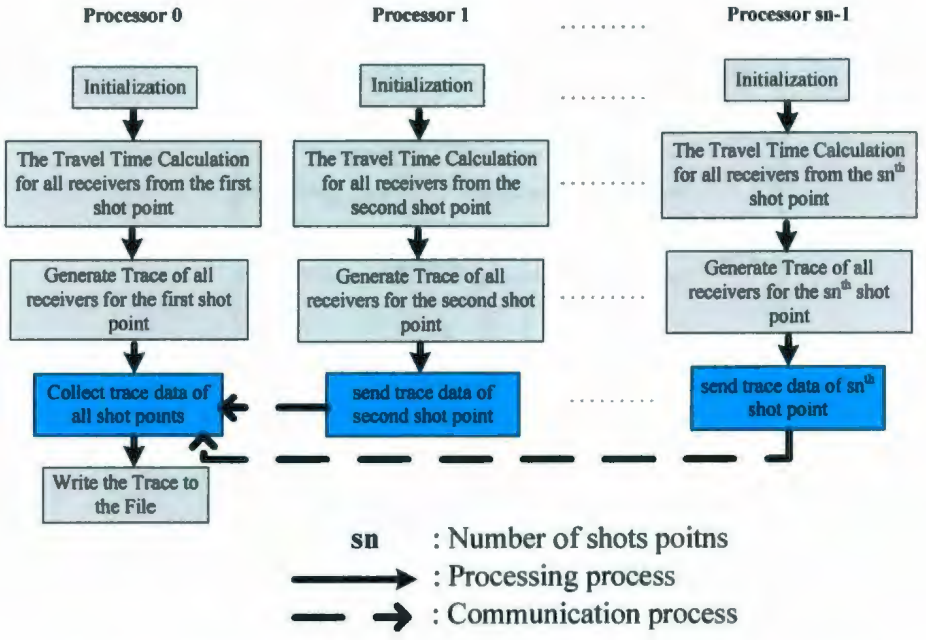

Fig. 4.13. The parallel program flow chart for modeling process.

4.4. Summary

Seismic modeling and Pre-stack Kirchhoff migration are provided in this chapter. The theories of seismic modeling and Pre-stack Kirchhoff migration, as well as the detail implementations of both applications are described. As shown in the experimental results, the modeling process is proven to effectively simulate the ray propagation in the introduced 2D/3D flat and polynomial layers velocity models, and the migration process is validated to fully recover the structures of the real models. In order to improve the

computational performance, the MPI parallel programming technique is employed in the implementations. The adopted problem decompositions strategy provides a nearly linear speedup.

The proposed MTTM and MESLM are validated in the applications. The different performances of two methods are noticed during the implementations, which are compared in the following chapter. As one can notice that the proposed methods well calculated the travel time, but have high computational complexity. Therefore, the purpose built, specific digital hardware implementation of the proposed methods are discussed and accomplished in the following chapters.

Chapter 5

The Comparisons of the Proposed Methods

The Proposed MTTM and MESLM are implemented in seismic modeling and Pre-stack Kirchhoff migration. Using the GA designs, both methods effectively simulate the ray transmission in the synthetic velocity models and calculate the travel time between two given points (shot point and receiver point). Because of the different principles of two methods, during the implementations, the performances are different.

5.1. Calculation Efficiency

As introduced in the chapter 3, for a ray path, the MTTM has to calculate the travel time for each ray segment in the ray path, and then decides this path is suitable or not. However, the MESLM may make a decision by partially calculating the ray path. Therefore, if the number of iterations is same, the MESLM is more efficient to find the ray path.

There is a clearly termination criterion in the MESLM: the total errors equal to zero. Therefore, it may terminate the search with a successful ray path before the number of generations reaches the iteration limit.

From the above two aspects, the MESLM should have higher efficiency to find the ray paths. However, the fitness function in this method, given in the equation 3.9, is more than double complexity of the fitness function in the MTTM, given in the equation 3.1. Thus, only if the velocity models are simple enough, such as the introduced models in the modeling and migration processes, in which less number of generations is required to find the ray path, the MESLM has the higher efficiency.

5.2. Simulated Ray Path and Results Quality

Both methods are verified to successfully calculate the travel time between the shot point and the receiver point. However, the simulated ray paths may be different, which because the same travel time might be produced by many ray paths. The correct ray propagation path would be found in the MESLM. Two examples ray paths, which have the same travel time in the velocity model, are illustrated in Fig. 5.1.

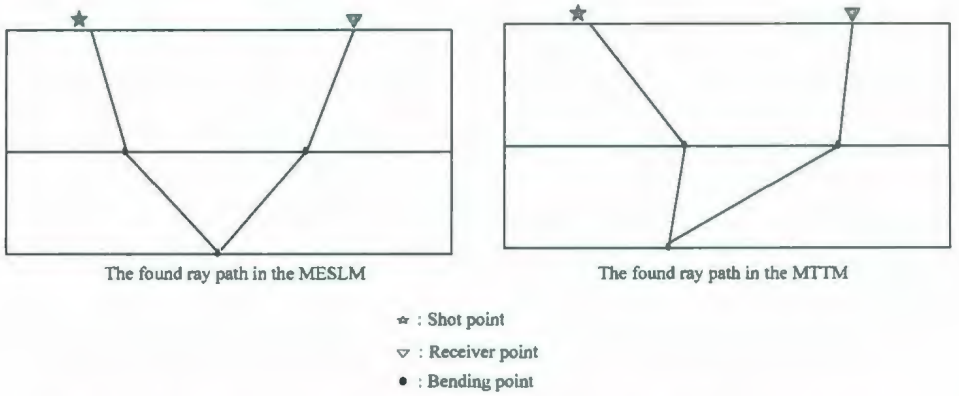

Fig. 5.1. Two examples of found ray paths in the different methods.

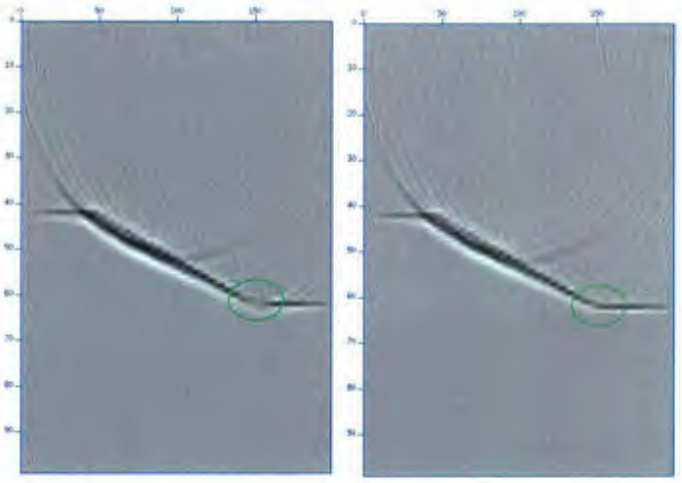

Fig. 5.2. The migration results of two methods. Left is the result of the MESLM. Right is the result of the MTTM.

From the migration results, another difference is noticed. As shown in Fig. 5.2, the qualities of the results produced by the two methods are closely same, except the range in the green circle. It can be explained that the MTTM produces many ray paths, which are reflected in that range, but not all of the ray paths satisfy Snell's Law.

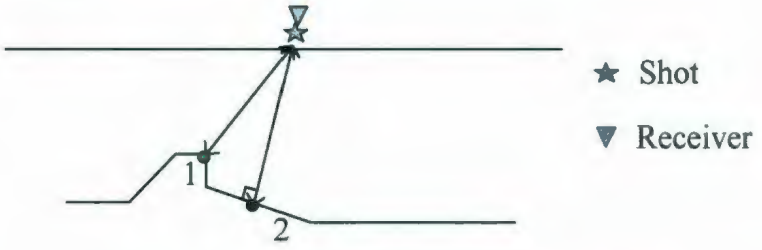

Fig. 5.3. An example of the correct results.

For the real data, the MESLM is inferred to produce the more correct results, because if the wave could be reflected by two points in the model, shown in Fig. 5.3, one is the minimum travel time point (point 1), other is the Snell's Law point (point 2). The most energy is reflected though the second point. The reflected diffraction wave from the first point becomes the noise (Milligan *et al.*, 1995).

5.3. Software Parallelizability

The MESLM has the higher parallelizability. For one shot-receiver pair, if the problem is decomposed by the different sets of individuals (e.g. ten individuals in each set) and worker-collector designs could be applied, as shown in Fig. 5.4. Each worker processor has the same workloads in the MTTM. Therefore, the collector processor remains in idle state before all the worker processors finish the calculations. However, the worker processors could reject the ray paths directly in the MESLM. The workloads of each worker processor become difference. Thus, in this method, the collector processor might receive the results from a worker processor and write those errors into memory, before others finished. The communication time is greatly reduced.

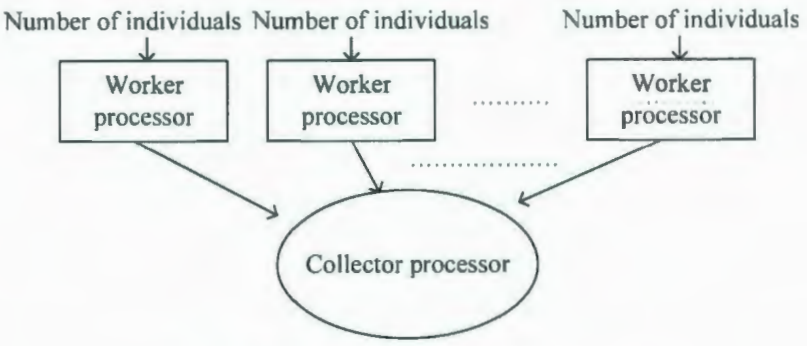

Fig. 5.4. The worker-collector design.

In the parallel modeling and migration programming, the MESLM has more advantages. Because of another alternate termination criterion for a GA run (the total errors equal to zero), not every shot-receiver pair need wait the tolerance time period to find the ray path. Therefore, if the modeling and migration parallel design, shown in Fig. 4.12, would be changed to a worker-collector design, some worker processors may share their workloads to others, in which the assigned workloads have been already finished.

5.4. Hardware Implementations

Because of the similar GA platforms are employed in both methods, the top level designs of the hardware algorithms of both methods are same, shown in Fig. 5.5. In this design, several individuals are first initialized in the Population registers block, and then they are passed into the Mutation&Crossover unit to produce new individuals as the offspring. After that, all individuals in both Population registers block and Offspring registers block are evaluated through the Fitness Evaluation units. Based on the fitness sorting results, the suitable individuals move to the Population register block and start a new iteration.

The main difference of two proposed methods is the Fitness Evaluation units are implemented based on equation 3.1 and equation 3.9. As one can noticed the equation 3.9 not only has to calculate the first derivatives of the layers function for each bending point, but also has more than double operations of equation 3.1. Moreover, the Fitness Evaluation units of the MESLM have to design the partial calculation and rejection for the ray paths. Therefore, the MESLM has more complexity for hardware implementations and requires more resources.

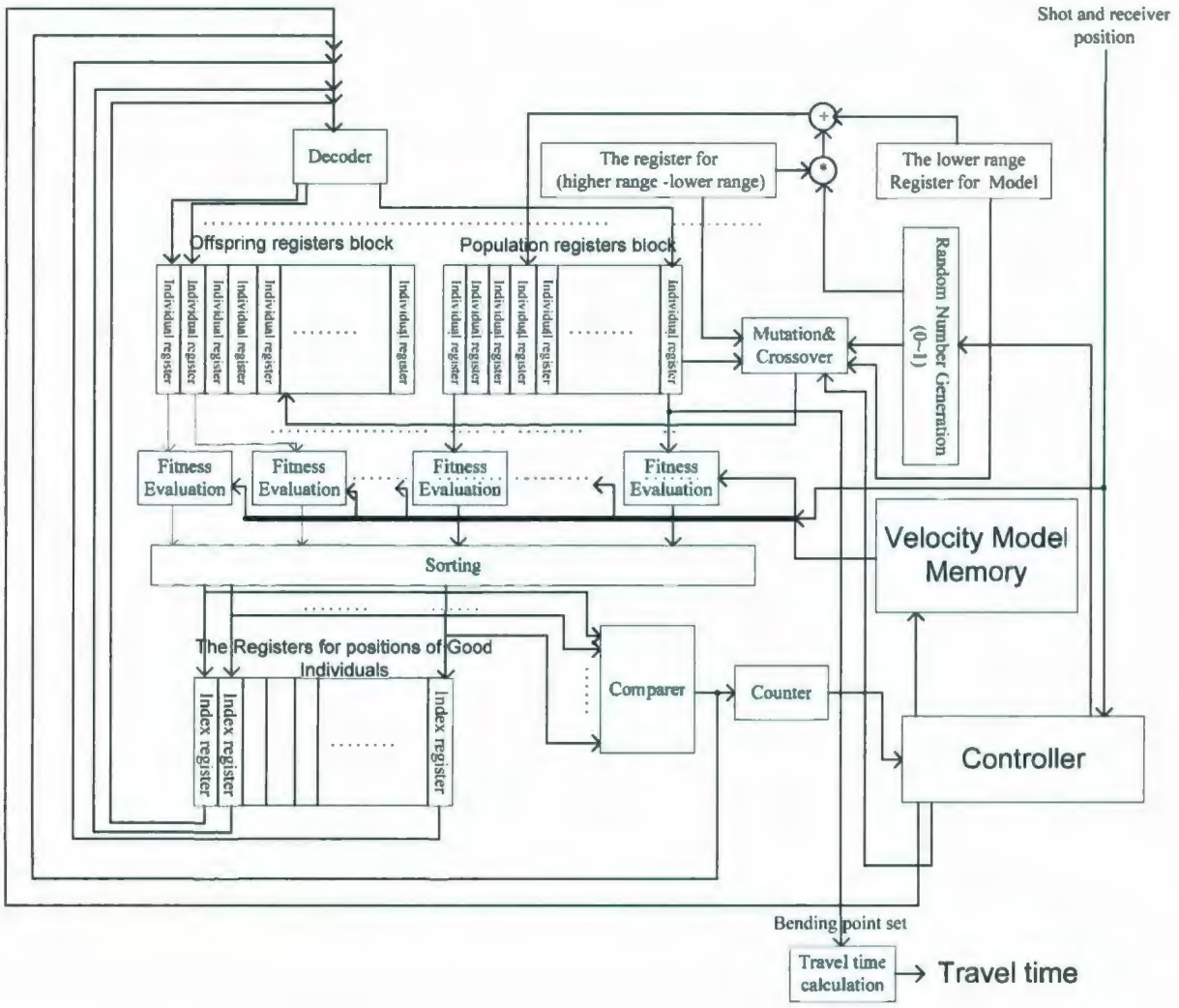

Fig. 5.5. The top level hardware algorithm design.

In order to validate our designed hardware algorithm, all the modules and blocks, shown in Fig. 5.5 are implemented in C++ environment, based on the lower computational and the lower implementation complexity method, the MTTM. The migration result is shown in Fig. 5.6, which demonstrates the designed hardware algorithm is a working algorithm.

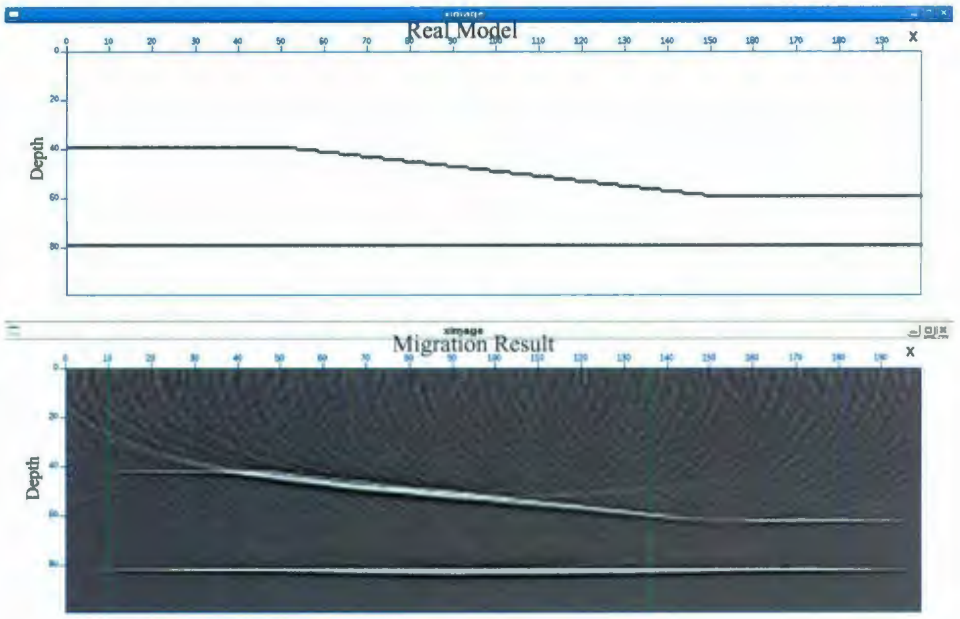

Fig. 5.6. The hardware algorithm verification in C++.

5.5. Summary

In this chapter, the MTTM and the MESLM are compared from the calculation efficiency, the simulated ray paths and results quality, the software parallelizability, as well as the hardware implementations. The comparison results show that the MESLM is more effective ray tracing method. It produces the more accurate results and is easily developed as a high efficiency parallel program. However, it also has the very high computational complexity. During the seismic modeling and migration experiments, if the velocity model are very complex, in which the Snell's paths are very hard to detect, the MESLM has the time consuming nearly three times more than the MTTM. Therefore, it is suitable to be implemented in the case of the high resolution results are required and the high performance clusters or supercomputers are available.

Taking a consideration of the comparison results from the hardware implementations, the MTTM is applied in the hardware design. Based on the validated top level design, which

is shown in Fig. 5.5, the hardware implementation, the synthesis results and the post-route timing simulation are provided in the next chapter.

Chapter 6

Hardware Implementation of the MTTM

From the implementations and the performances of seismic modeling and Pre-stack Kirchhoff migration, the travel time calculation for a ray path is noticed as the most computational intensive part. In order to accelerate our ray tracing algorithm, a purpose built, specific digital hardware implementation is described in this chapter. This hardware implementation adopts the fixed-point arithmetic, which runs faster and uses less chip space than the corresponding floating-point calculation, and applies the parallel architecture design for the fitness evaluation function as well as the mutation operation, which produces much better performance than Von Neumann microprocessors (Bording, 1996).

The implemented hardware design is supposed to work as an integrated card, which can be plugged into the PCI socket. A field-programmable gate array (FPGA) board, which is a semiconductor device containing programmable logic components, LATTICE ECP-DSP 20E 4F484C, is employed in our research. The interface between the FPGA and

computer PCI socket has been developed by other colleagues in the Music-π group. The 48-bits fixed-points arithmetic (He, Lu and Bording, 2008) is employed in the implementation.

6.1. Top Level Design

Because of the resource limitation of LATTICE ECP-DSP 20E 4F484C, the original design, shown in Fig. 5.5 has been modified. The velocity model memory is redesigned to the register buffer, which greatly reduce the access time. The top level design of the hardware implementation is shown in Fig. 6.1.

In Fig. 6.1, as the name suggests, the Population register block is designed to contain the parent individuals. The Offspring register block contains the offspring individuals. The Mutation&Crossover unit mates the parents to produce the offspring. The Random Number Generation unit produces the uniform random numbers from 0 to 1, which are used in the initialization and Mutation&Crossover unit. The MUX and F_counter work together to sequentially select the individuals and pass the selected individuals to the Fitness Evaluation unit to calculate the fitness. The Sorting engine is implemented to sort the fitness. The suitable individual indexes are stored in the Index Register Block. The number of generations is counted by the G_counter unit. if the number of generations does not reach the limited generations, based on the contents of the Index Register Block, the new population are produced by the Decoder. After the hardware implementation finishes the generations, the calculated travel time is stored in the 48 bit register.

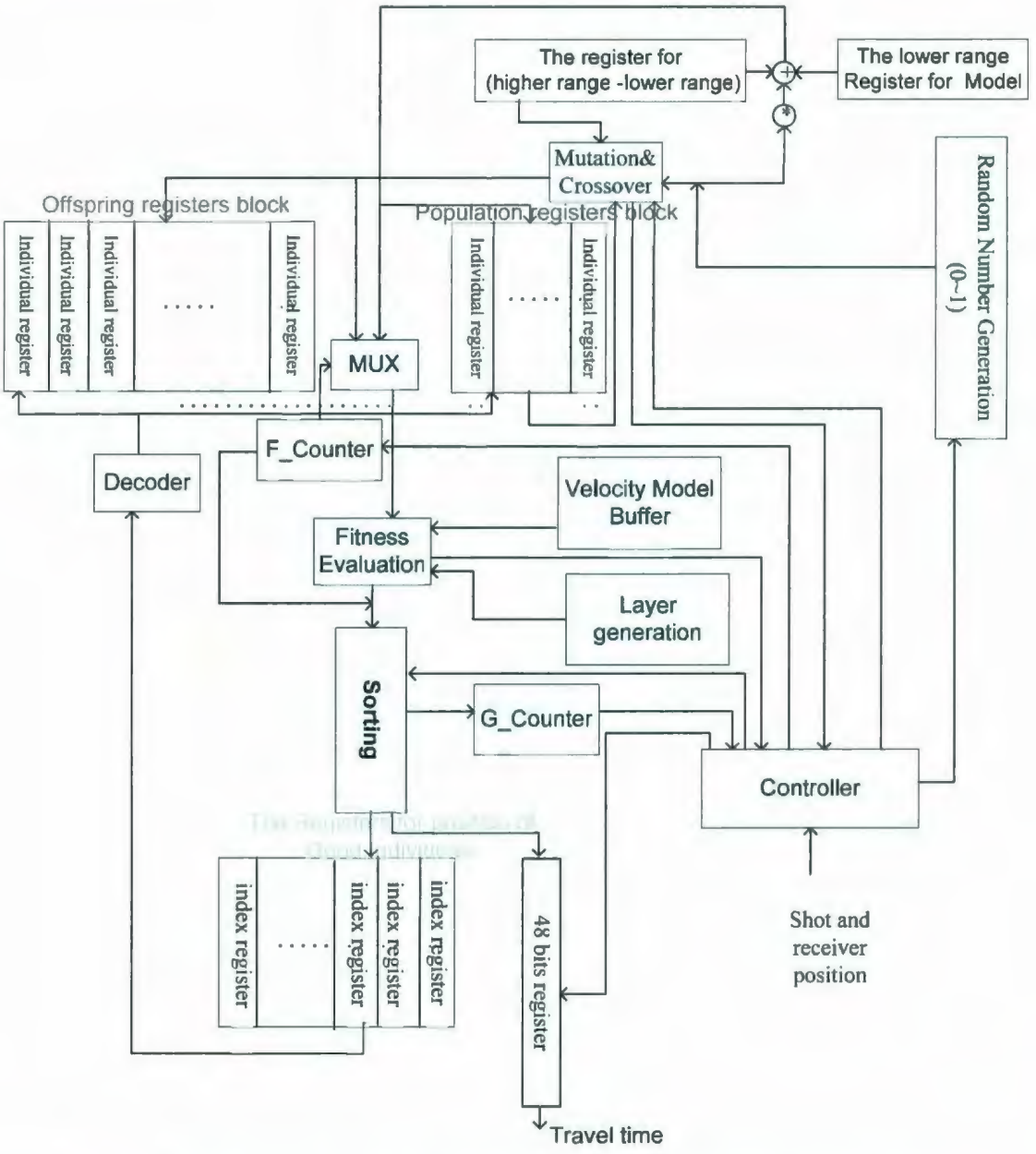

Fig. 6.1. The top level design of the minimum travel time ray tracing engine.

In a generation, all parent individuals are loaded into the Mutation&Crossover unit at the same time. Another several individuals are produced as the offspring, which are loaded into the Offspring register block. Each individual in the Population register block and the Offspring register block is sequentially evaluated by the Fitness Evaluation unit. The calculated fitness is attached a binary number index coming from the F_counter, which

indicates the location of the individuals. All fitnesses are sorted in the designed Sorting engine and the contents in the Index registers are produced by the locations of the minimum fitnesses. If the output of G_counter is less than the accepted number of generations, the sorted indexes are decoded into the locations the individuals and Population register block loads the corresponding individuals. The finite state machine (FSM) of the top level controller is shown in Fig. 6.2.

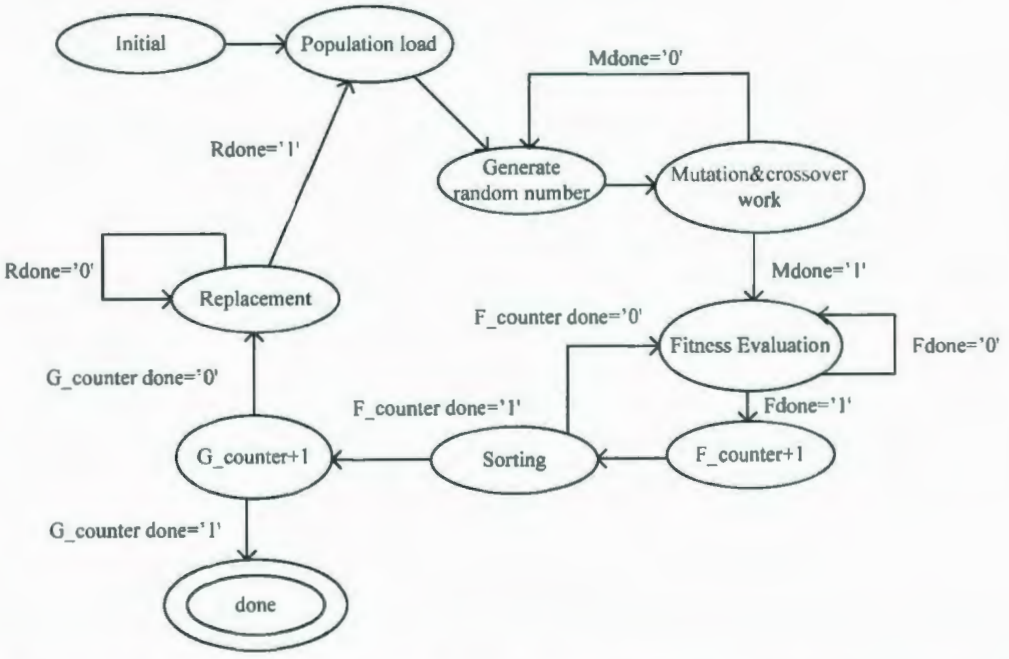

Fig. 6.2. The FSM of top level controller for the minimum travel time ray tracing engine.

6.2. The Fitness Evaluation Unit

An individual, representing a ray path, consists of a set of x coordinates of the bending points, which is stored in a register in the Population or the Offspring register block. The Fitness Evaluation unit calculates travel time for each ray segment between two bending points in a ray path. The travel time for the entire ray path is produced by summing up the travel time of each ray segment.

The design diagram is shown in Fig. 6.3, which is automatically produced by the synthesis tool "Synplify". For the whole ray path calculation, the x coordinates are selected by the multiplexer from an individual register and the y coordinates are produced by the LayerGenerator unit, which is implemented based on the layer description function. In order to avoid the slow division operation, the reciprocal velocities, are stored in the velocity buffer. TTu is the travel time calculation unit for one ray segment. The calculated travel time of each ray segment is accumulated in the register48. When the travel time for the entire ray path is computed, the Fitness Evaluation unit exports the travel time and a "done" signal.

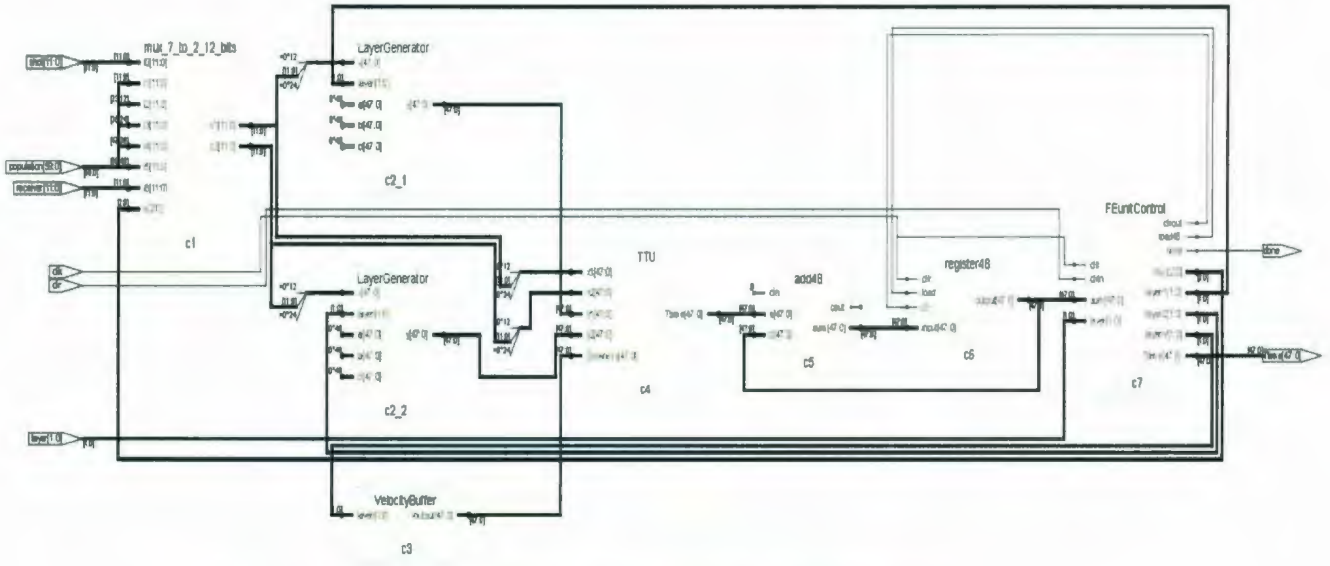

Fig. 6.3. The design diagram for the Fitness Evaluation unit.

For one ray segment, the travel time calculation unit is designed based on the given equation 3.1 in chapter 3. The designed block diagram is shown in Fig. 6.4. The input ports of the travel time calculation unit are the x, y coordinates for the first point and x, y coordinates for the second point, as well as the slowness between two points. The square-root unit is implemented by a binary shifting square-root algorithm (S. Majerski, 1985).

The fixed-point parallel multipliers and carry-saving adders (Cooper, 1988, Elguibaly *et al.*, 2000) are implemented in this unit. The combinational logic circuit and the functional parallel design make the travel time calculation have high efficiency.

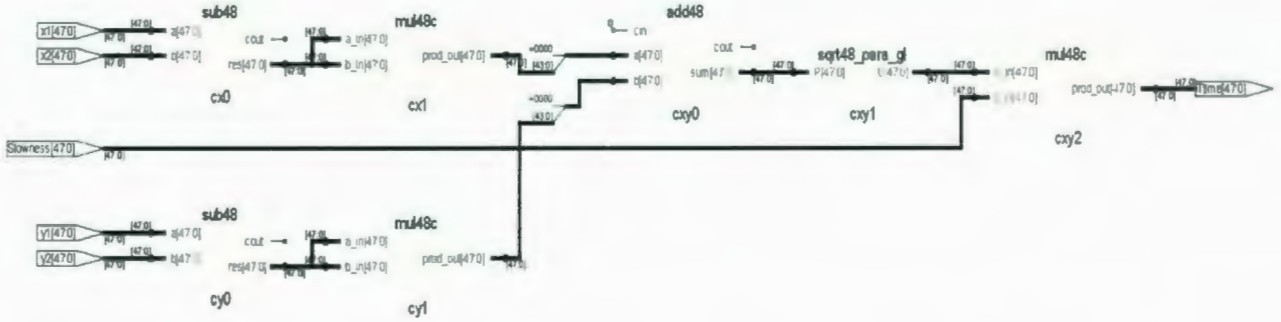

Fig. 6.4. The design diagram of the travel time calculation unit for a single ray segment.

6.3. The Sorting Engine

Sorting, which processes the unordered data sequence to ordered, is the most frequently used operation in an algorithm design. The best sorting algorithm in the computer software has the efficiency O(N*log*N), in which N is the number of items to sort. However, it is possible to design a hardware sorting algorithm with high time efficiency O(N) (Colavita *et al.*, 2003).

In our hardware implementation, the sorting engine is designed based on the insertion sort and has the time efficiency and area complexity at O(N). In the insertion sort, the data are sequential processed. An example of insertion sort is shown in Fig. 6.5, the sorted data are stored in the list. When a new data arrive, it is compared with the data in the list. After the comparison, the location of the new data is decided and the new data is inserted into the list.

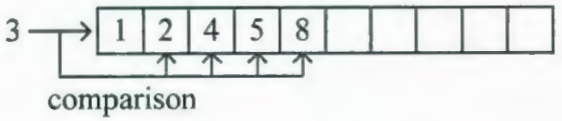

Fig. 6.5. An example of the insertion sort.

Instead of sequentially comparing the new arrived data with the existed data, a hardware algorithm (Lu and Bording, 2006) is proposed that operates the comparisons at the same time and detects the location by the returned values of the comparisons. If the sorting process is separated into three steps, the proposed algorithm is easy to understand and implement. As shown in Fig. 6.5, when the number 3, as the new data, arrives, the first step of the algorithm is compare 3 with 1, 2, 4, 5 and 8 at the same time. The returned value of the comparisons are "greater", "greater", "less", "less", "less". In the second step, the location of number 3 is decided as following the last "greater" result. As one can notice that in order to insert number 3 into the list, the number 4, 5, 8 have to shift to right and the returned comparisons values of those three numbers are the same. Therefore, in the third step, the "greater" numbers stay in the location and the "less" numbers are moved to the right and also the number 3 is inserted.

The hardware design diagrams are shown in Fig. 6.6.

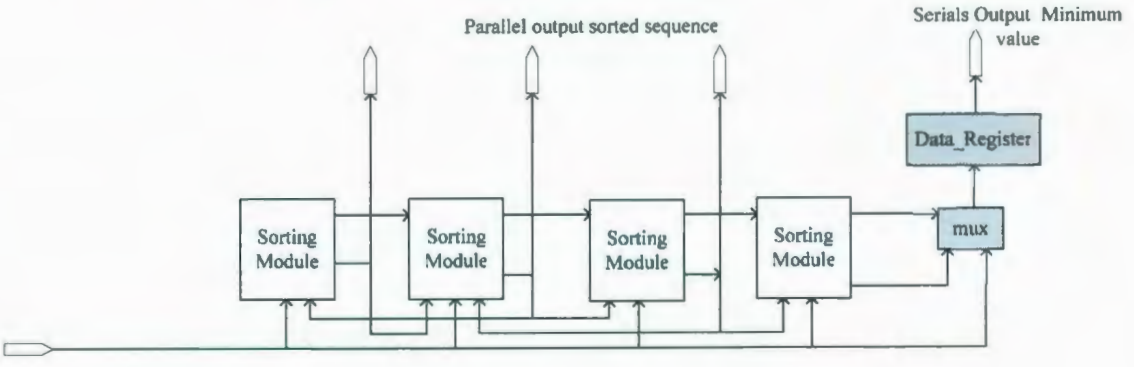

Fig. 6.6a. The design diagram of the Sorting engine.

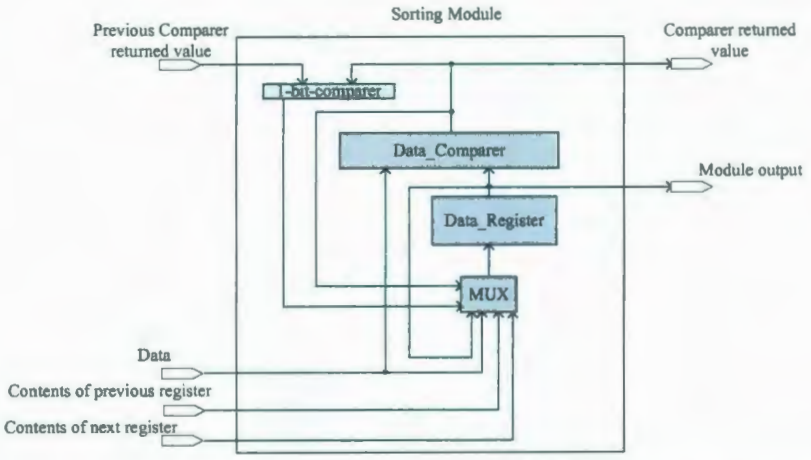

Fig. 6.6b. The design diagram of Sorting Module in the Sorting engine.

In a Sorting Module, the Data_Comparer is designed to compare the new data with the data in the Data_Register. Based on the Data_Comparer result, and the result from the previous Sorting Module Data_Comparer, the 1-bit-comparer returns equal or not. After the comparisons, the four to one MUX decides which data are load into the register.

The flux diagram of the Sorting Module is illustrated in Fig. 6.7.

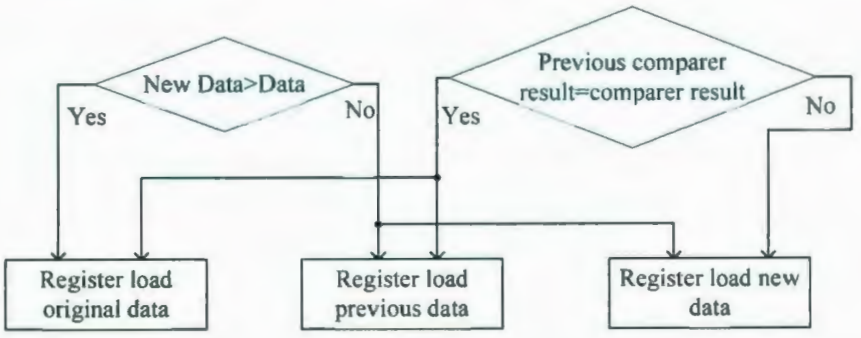

Fig. 6.7. The flux diagram of the Sorting Module.

6.4. The Random Number Generation Unit and the Mutation&Crossover Unit

The Random Number Generation unit (RNGu) and the Mutation&Crossover unit work together to produce the offspring. The RNGus are implemented by Galois Linear Feedback Shift Register (GLFSR) (Pardhan and Chatterjee, 1999), which generates the uniform random numbers. The generated random numbers are employed to perform the probability selection and the uniform mutation. When the parent individual arrives, a random number is produced to decide which operation should be performed on the individuals. An example design is illustrated in Fig. 6.8.

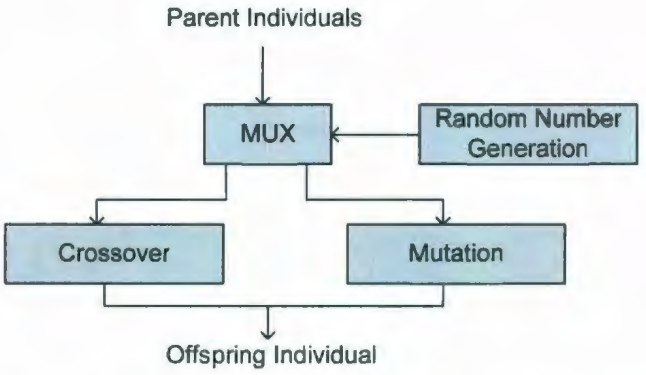

Fig. 6.8. The design of random number selection.

The crossover unit is implemented in hardware by the two points crossover, in which the bits between two points in the parent individuals are exchanged. In order to perform the uniform mutation, another two RNGus are required. An example design is shown in Fig. 6.9. In this design an individual contains five bending points. Each bending point is represented by twelve binary bits. The rng12 unit produces the uniform random numbers which are operated on the parent to produce the offspring. The rng4 unit is designed to select the subtraction or the addition operations. The produced offspring bending points

are loaded in the register12s. In this design, the mutation operations for the different bending points operate in parallel. The operation efficiency is greatly increased.

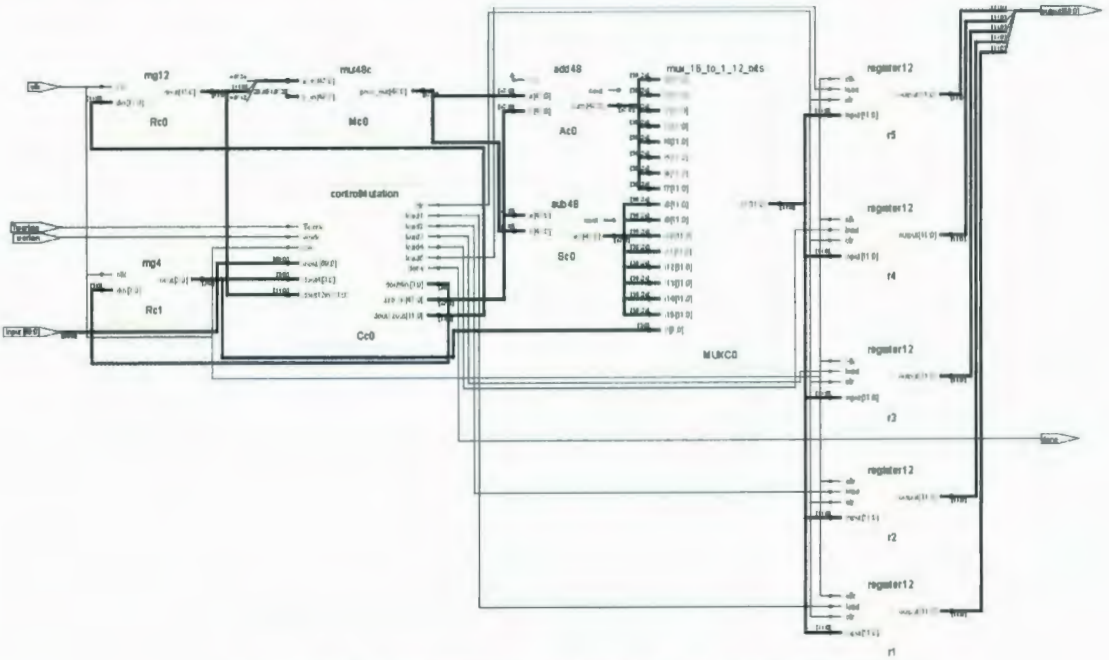

Fig. 6.9. An example design for the Mutation unit.

6.5. Hardware Performance

In order to validate our hardware design, a hardware engine is implemented in VHDL and synthesized for LATTICE ECP-DSP 20E 4F484C by employing the "Synplify for Lattice 9.4L" synthesis tools. The designed engine is expected to simulate the ray propagation in a three flat layers velocity model and calculate the travel time.

The Population register block consist of four registers to contain four individuals and the Offspring register block is implemented to contain twelve individuals. Each bending points set in an individual has five x-coordinates of bending points and each x-coordinate of bending point is represented by twelve binary bits. The y-coordinates of a bending point are calculated in the LayerGeneration unit in the Fitness Evaluation unit. In the initialization step, the number of generation is fixed to 100 and the velocity of each layer

is fixed to a constant value represented by 48 binary bits. The input ports and output ports of the designed engine is specified in the Table 6.1 and illustrated in Fig. 6.10.

Table 6.1. The ports specification table.

Port Name	Port attribute	Port width (bits)	Description
Tworken	Input	1	The global reset signal
clk	Input	1	The global clock signal
shot	Input	12	The position of the shot point
Re	Input	12	The position of the receiver point
layer	Input	2	The number of layers in the assumed velocity model
individual1	Output	60	The bending points set which optimized the function
Ttime	Output	48	Minimum travel time

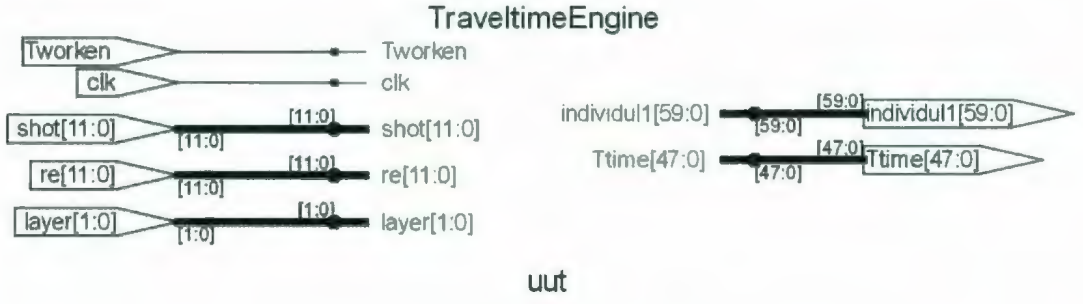

Fig. 6.10. The input output ports illustration.

The synthesis area performances are shown in the Fig. 6.11, from which the designed engine is proven to fit the employed device. As one can notice no external RAM is required for the engine, thus, the memoryless design provides the high operational performance.

```
Design Summary
   Number of registers:    2423
      PFU registers:    2375
      PIO registers:    48
   Number of SLICEs:          6237 out of  9856 (63%)
      SLICEs(logic/ROM):      6237 out of  7392 (84%)
      SLICEs(logic/ROM/RAM):     0 out of  2464 (0%)
         As RAM:            0
         As Logic/ROM:      0
   Number of logic LUT4s:     9423
   Number of distributed RAM:   0 (0 LUT4s)
   Number of ripple logic:    447 (894 LUT4s)
   Number of shift registers:   0
   Total number of LUT4s:     10317
   Number of external PIOs: 136 out of 360 (38%)
   Number of PIO IDDR/ODDR:     0
   Number of PIO FIXEDDELAY:    0
   Number of 3-state buffers:   0
   Number of DQSDLLs:  0 out of 2 (0%)
   Number of PLLs:  0 out of 4 (0%)
   Number of block RAMs:  0 out of 46 (0%)
   Number of GSRs:  1 out of 1 (100%)
   JTAG used :      No
   Readback used :  No
   Oscillator used :  No
   Startup used :   No
   Notes:-
      1. Total number of LUT4s = (Number of logic LUT4s) + 2*(Number of
     distributed RAMs) + 2*(Number of ripple logic)
      2. Number of logic LUT4s does not include count of distributed RAM and
     ripple logic.
```

Fig. 6.11. The synthesis area report.

The synthesis timing report is shown in Fig. 6.12. The maximum frequency of the designed engine is about 11 MHz, which means the travel time calculation for a ray segment is about 90 ns.

```
Report Summary
--------------
------------------------------------------------------------------------------
Preference                              |  Constraint|       Actual|Levels
------------------------------------------------------------------------------
                                        |            |             |
FREQUENCY PORT "clk" 12.000000 MHz ;    |  12.000 MHz|  11.114 MHz| 289 *
                                        |            |             |
CLOCK_TO_OUT ALLPORTS 5.000000 ns       |            |             |
CLKPORT "clk" ;                         |    5.000 ns|    4.310 ns|   2
                                        |            |             |
------------------------------------------------------------------------------
```

Fig. 6.12. The synthesis timing report.

The operational performance of the RNGu, Mutation&Crossover unit, Sorting engine and Replacement module are velocity model independent. In another words, for any cases of velocity models, in one generation, the 35 clock cycles are required by the RNGu and the

Mutation&Crossover unit, 16 clock cycles are required by the Sorting engine and 3 clock cycles are required for the replacement. The performance of the Fitness Evaluation unit is depended on the number of velocity layers in the model. For each ray segment, the travel time calculation requires one clock cycle and the loading process needs another clock cycle. Moreover, two clock cycles are required by the initialization states and done states. Therefore, for a ray path, which propagated through a three layers model (six ray segments), 14 clock cycles are required.

A generation of the serial process requires 150 clock cycles, 214 clock cycles and 278 clock cycles for one layer model, two layers model and three layers model, respectively. For three layers model, in a second, the hardware engine operates 11 MHz/278=39568 generations, which is a comparable performance with IBM MT-M 6218 workstation that has Intel® Pentium® 4 CPU 3.00GHz. The time consumption for the travel time calculation of three layers model, using the MTTM, in IBM workstation is roughly produced by the Linux system and illustrated in the Fig. 6.13. For 39568 generations, the IBM workstation requires about 0.5 second to process.

```
[shiliang@augusta timingtest]$ time ./main
 Shot ponit at (50,0), Receiver point at (80,0)

 Number of Generation:       39567

 Travel time:            5.512053

 x-coordinates of Bending point set
   29.30951       39.06391       50.51765       61.08103       71.56972

 Travel Time Calculation Done

real    0m0.521s
user    0m0.514s
sys     0m0.006s
```

Fig. 6.13. The time consumption illustration for the IBM MT-M workstation.

Based on the operational performance of each module of the hardware engine, if a larger enough FPGA device is available and original parallel fitness evaluation design (shown in Fig. 5.5) was implemented, a generation would only require 35+16+3+((N•2)+2) clock cycles, where N is the number of layers handled in the hardware engine. Moreover, if assume the 11MHz maximum frequency is unchanged, the inferred performances of the parallel hardware engine versus the 3GHz IBM workstation is illustrated in Fig. 6.14, which shows, with the number of layers increasing, the 11MHz parallel hardware engine produces a much better performance than the 3GHz IBM workstation. The direct comparison of these clock cycles requires an understanding of the instruction complexity. The PC instructions are rather basic while the hardware engine instructions are quite complex.

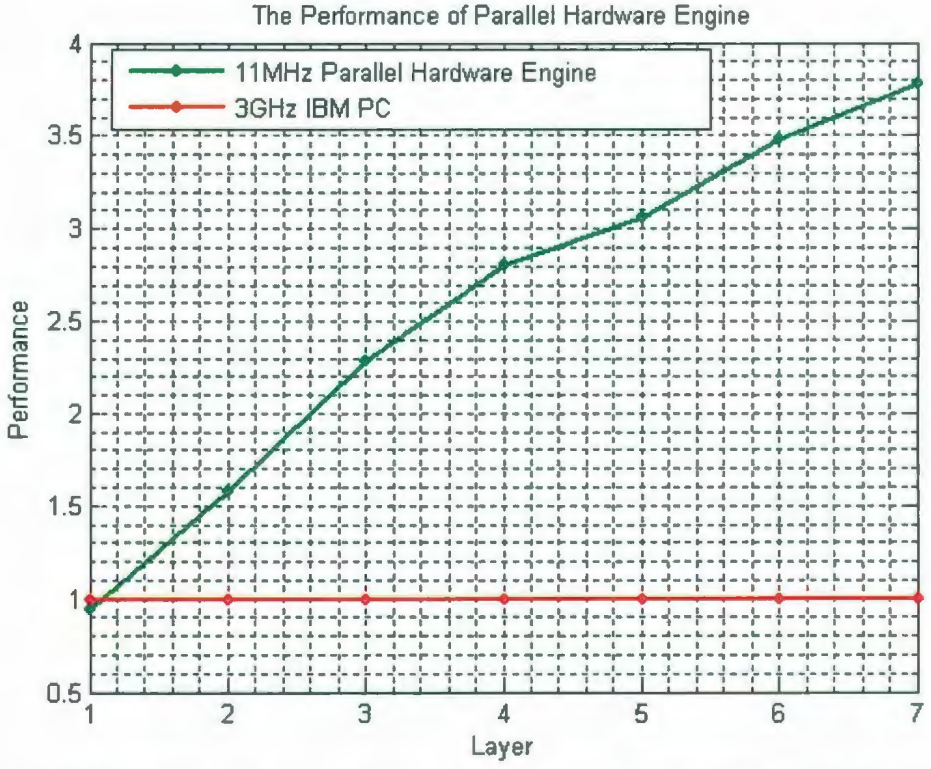

Fig. 6.14. The inferred performances of the 11MHz parallel hardware engine versus the 3GHz IBM workstation.

Furthermore, as one can notice the maximum frequency of our designed engine is much lower than the IBM workstation. However, the semiconductor techniques are improving every day. If the frequency of designed engine would be increased, the inferred performances of the series and parallel hardware engines versus the 3GHz IBM workstation are illustrated in Fig. 6.15. As shown in this figure, if the maximum frequencies of the designed engine would be increased to 200Hz, the travel time calculations in the hardware engine would be forty times faster than the 3GHz IBM workstation.

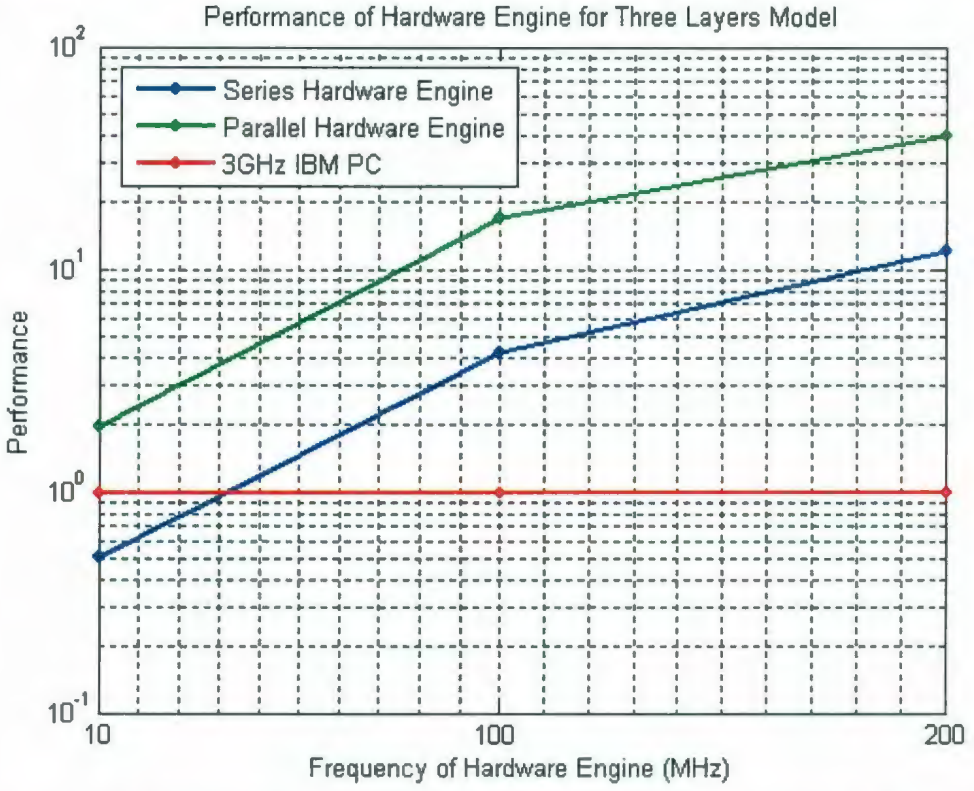

Fig. 6.15. The inferred performances of the series and parallel hardware engines versus the IBM workstation with the frequency of hardware engines increasing.

The post-route timing simulation (emulation) is finished in Modelsim. A 2D flat three layer model are adopted in the emulation and one shot point and one receiver point are

located on the flat surface. The emulated results for the travel time of each generation are shown in Fig. 6.16.

As one can notice that, the travel time is decreasing with the number of generation increasing, which is the expected performance. For the different layers, the required number of generations are different that are decided by check which generation has already produced the final travel time. Only the travel times that produced within the required number of generation are shown in Fig. 6.16.

For the ray path reflected from the first layer in the model, only five generations are required. The error between the produced travel time at fifth generation and the travel time calculated by hand is $6.7 \cdot 10^{-6}$. For the ray path reflected from the second layer, thirty eight generations are required. The error between the produced travel time and the hand calculated travel time is $3.39 \cdot 10^{-5}$. For the ray path reflected from the last layer, sixty five generations are needed and the error is $7.6 \cdot 10^{-4}$. The produced errors are increasing with the number of ray segments increasing, which is because the errors are accumulative from the previous ray segments.

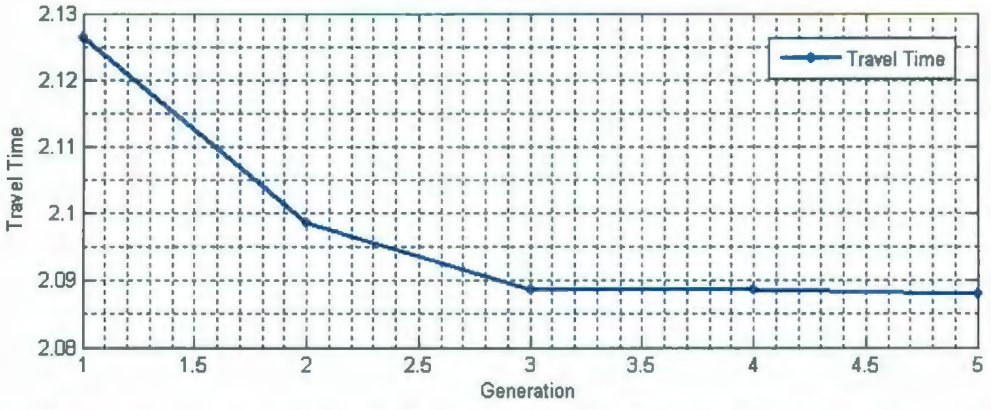

Fig. 6.16a. The emulation result for the ray path reflected from the first layer of the model.

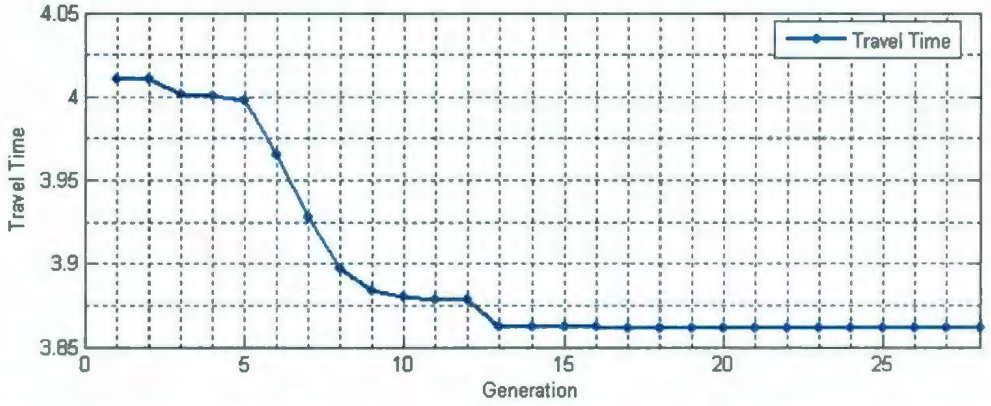

Fig. 6.16b. The emulation result for the ray path reflected from the second layer of the model.

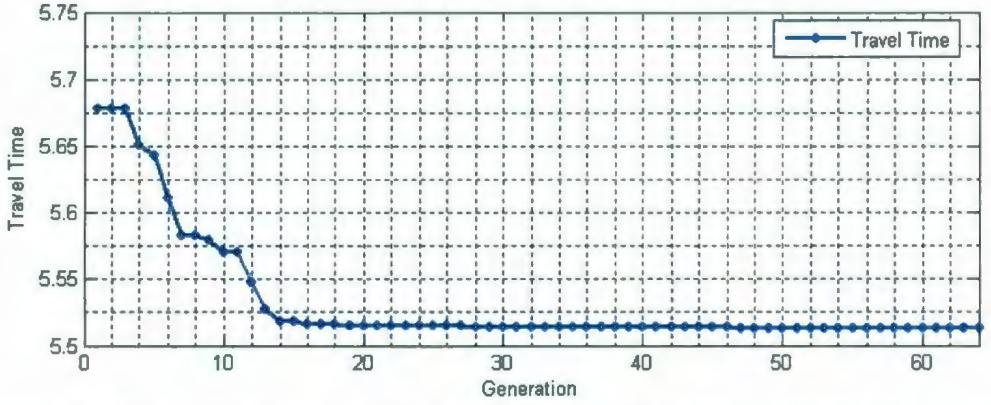

Fig. 6.16c. The emulation result for the ray path reflected from the third layer of the model.

6.6. Summary

In this chapter, based on the MTTM, a hardware algorithm is proposed and a hardware engine is implemented in the low-cost FPGA device, LATTICE ECP-DSP 20E 4F484C. The post-route timing simulation results of the designed hardware engine show that the implemented hardware engine finished the expected function and produced the acceptable travel times. Because of the limitation of the configurable slices resources for the desired FPGA device, many designed modules are serially implemented. The counter unit and the multiplexer unit work together to control the series data flow, which greatly

reduces the complexity of the FSM of the controller and makes the design highly adaptable.

The performance of the designed hardware engine is actually reduced by the series modules and the low-cost FPGA device. However, the fixed-point arithmetic and functional parallel design reduce the cost of the travel time calculation to one clock cycle, the high efficiency sorting engine decreases the complexity of sorting operation to O(N), the memoryless design causes almost no time penalty to access the velocity model. With all these advantages, the designs provide an implemented hardware engine with a comparable performance to the 3GHz IBM workstation. If a conservative design scale up is used then a parallel layer system would be forty times faster.

Chapter 7

Conclusions and Future Work

7.1. Conclusions

In this thesis, two layer based seismic ray bending methods are proposed. In each method, based on Fermat's Principle and Snell's Law, the ray paths are constructed by using the genetic algorithm to search the bending points on the layers of the velocity model. The limitations of ray shooting methods are overcome and the accurate travel times are calculated by the two proposed methods.

In order to validate our ray tracing methods, seismic modeling and Pre-stack Kirchhoff migration are implemented. The results of Pre-stack Kirchhoff migration show that the migration fully recovers the real models for both of 2D and 3D flat layers, 2D and 3D polynomial layers. All of the interfaces are visible, and approximate the original shapes and original depths. Thus, our ray tracing methods are proven to be effective ray tracing methods. Moreover, to improve the computational performance, the MPI parallel

programming technique is employed in the implementations and provides the implementations with a nearly linear speedup.

During the implementation of seismic modeling and Pre-stack Kirchhoff migration, the operational and computational performances of two proposed seismic ray tracing methods are noticed to be different. From the comparison results, the MESLM is proven to be a more effective ray tracing method. It produces the more accurate results and is easily developed as a high efficiency parallel program. However, it suffers from the very high computational complexity. Therefore, it is suitable to be implemented in the case of the high resolution results are required and the high performance clusters or supercomputers are available.

For the hardware implementations, the MTTM is the considered method to be implemented into the hardware engine. Because of the limitation of the configurable slices resources of the FPGA device, many implementations of the designed modules are modified from the original parallel design to the series design. However, the fixed-point arithmetic, the functional parallel design, the high efficiency sorting engine and the memoryless design for the velocity model work together to provide the implemented hardware engine with a comparable performance to the IBM workstation. The post-route timing simulation results of the designed hardware engine further proves that the implemented hardware engine finished the expected function and produced the acceptable travel time.

7.2. Future Work

From several aspects of our research, the future works, which should be studied, are:

- The MESLM is high efficient method, but it suffers from the high computational complexity. As one of the future study, continuing investigating the Snell's Law scheme and developing another similar but low cost fitness function are necessary.

- The results of Pre-stack Kirchhoff migration proved the proposed ray tracing methods are feasible for the introduced models. Thus, another future work would be to extend our methods for more complex 2D and 3D earth media.

- As one can notice, the proposed hardware algorithm could be extended to a fully parallel algorithm, which means one generation is finished in one clock cycle. However, that requires large amount of configurable slices resources and could not fit in any FPGA device even for a high end device. The future direction from this aspect is to develop the interface between FPGA devices and utilize multiple devices to solve the linear problem. Also, the pipelining technique hardware design could be considered in the future work.

References

Alaei, B. and Pajchel, J., "Single Arrival Kirchhoff Prestack Depth Migration of Complex Faulted Folds from the Zagros Mountains, Iran," *CSEG.*, pp. 42-46, Jan. 2006.

Bleistein, N., "Hagedoorn Told Us How to Do Kirchhoff Migration and Inversion," *The Leading Edge*, pp. 918-927, Aug. 1999.

Bording, R. P., "High Performance Computation (Teraflops) for Seismic Processing," *Journal of Canadian Society of Exploration, Geophysics*, pp. 121-129, Dec. 1996.

Bording, R. P., and Lines, L. R., "Seismic Modeling and Imaging with the Complete Wave Equation," *SEG publication*, Tulsa, 1997.

Buske, S., "3-D Pre-stack Kirchhoff Migration of the ISO89-3D Data Set," *pure and applied Geophysics*, vol. 156, issue 1-2, pp.157-171, 1999.

Carcione, J. M., Herman, G. C. , and ten Kroode, A.P.E., "Y2K Review Article Seismic Modeling," *Geophysics*, vol. 67, no. 4, pp.1304-1325, Jul. 2002.

Cerveny, V., *Seismic Ray Theory*, Cambridge University Press, 2001.

Glassner, A. (Ed.), *An Introduction to Ray Tracing*, Academic Press, 1989.

Colavita, A.A., Cicuttin, A., Fratnik, F. and Capello, G., "SORTCHIP: A VLSI Implementation of a Hardware Algorithm for Continuous Data Sorting," *IEEE Journal of solid-state circuits,* vol. 38, no. 6, pp. 1076-1079, Jun. 2003.

Cooper, A.R., "Parallel Architecture Modified Booth Multiplier," *IEEE Proc.-G*, vol. 135, pp. 125-128, 1988.

Davis, L.(ed.), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

Docherty, P., "A Brief Comparison of Some Kirchhoff Integral Formulas for Migration and Inversion," *Geophysics*, vol. 56, no. 8, pp. 1164-1169, Aug. 1991.

Dong, W., Emanuel, M. J., Bording, R. P., and Bleistein, N., "A Computer Implementation of 2.5D Common Shot Inversion," *Geophysics*, vol. 56, no. 9, pp. 1384-1394, 1991.

Eiben, A.E. and Schippers, C.A., "On Evolutionary Exploration and Exploitation," *Fundamenta Informaticae*, vol. 35, issue 1-4, pp. 35-50, 1998.

Elguibaly, F., "A Fast Parallel Multiplier-Accumulator Using the Modified Booth Algorithm," *IEEE Transactions on circuits and system-II*, vol. 47, no. 9, pp. 902-908, Sept. 2000.

Eshelman, L. J., "The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination," *Rawlins, G. J.E., Proceedings of Foundations of Genetic Algorithms Workshop -1*, San Mateo, CA, Morgan Kauffman, 1990.

Goldberg, D. E., *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison-wesley, Reading Mass, 1989.

Gropp, W., Lusk, E., and Skjellum, A., *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, MIT Press in Scientific, 1994.

Haslinger, F. and Kissling, E., "Investigating Effects of 3-D Ray Tracing Methods in Local Earth Quack Tomography," *Physics of the Earth and Planetary Interiors*, vol. 123, issue 2-4, pp. 103-114, 2001.

He, L., Lu, S., Bording, R. P., "A Computational Engine for Petroleum Applications Using Genetic Algorithms," *CCECE*, pp. 245-250, May 2008.

Holland, J. H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

Iorio, A. and Li, X., "Parameter Control within a Co-operative Co-evolutionary Genetic Algorithm," *Proceedings of the 7th international conference on Parallel Problem Solving from Nature*, pp. 247-256, Sept. 2002.

Keller, H. B. and Perozzi, D. J., "Fast Seismic Ray Tracing," *SIAM*, vol. 43, no. 4, pp. 981-992, Aug. 1983.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P., "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

Koza, J. R., *Genetic Programming: On Programming Computers by means of Natural Selection*, MIT Press, Cambridge, MA., 1992.

Lu, S. and Bording, R. P., "An Evolutionary Approach to 2D and 3D Kirchhoff Seismic Modeling and Migration," *IEEE, NECEC*, Nov. 2007.

Lu, S., and Bording, R. P., "Sorting Algorithm&Hardware Implementation," *Aldrich Interdisciplinary Student Conference*, Memorial University of Newfoundland, 2006.

Majerski, S., "Square-Rooting Algorithms for High-Speed Digital Circuits," *IEEE Transactions on Computers*, vol. C-34, no. 8, pp. 724-733, Aug. 1985.

Metropolis, N., and Ulam, S., "The Monte Carlo Method," *Journal of the American Statistical Association*, vol. 44, no. 247, pp. 335–341, 1949.

Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Third Edition, Springer, 1996.

Milligan, P., Rector, J. and Bainer, R., "Seismic Reflection Imaging at a Shallow Site," *65th Society of Exploration Geophysicists Annual Convention*, Oct., 1995.

Moser, T. J., "Shortest Path Calculation of Seismic Rays," *Geophysics*, vol. 56, no. 1, pp. 59-61, Jan. 1991.

Officer, C.B., *Introduction to Theoretical Geophysics*, Springer-Verlag, New-York, 1974.

Pardhan, D.K. and Chatterjee, M., "GLFSR-A New Test Pattern Generator for Built-in-Self-Test," *IEEE Transactions on CAD*, vol. 18, no. 2, pp. 238-247, Feb. 1999.

Peter, S. and Keith, M., *Realistic Ray Tracing, 2nd edition*, A.K.Peters, 2001.

Phadke, S., Bhardwaj, D. and Yerneni, S., "3D Seismic Modeling in a Message Passing Environment," *Society of Petroleum Geophysicists third conference-2000,* pp. 168-172, Jan. 2000.

Sadeghi, H., Suzuki, S., Takenaka, H., "A Two-Point, Three-Dimensional Seismic Ray Tracing Using Genetic Algorithms," *Physics of the Earth and Planetary Interiors,* vol. 113, pp. 355-365, 1999.

Wesson, R. L., "Travel-Time Inversion for Laterally Inhomogeneous Crustal Velocity Models," *Bull. Seism. Soc. Am.* 61, pp. 729-746, 1971.

Zhao, A., Zhang, Z. and Teng, J., "Minimum Travel Time Tree Algorithm for Seismic Ray Tracing: Improvement in Efficiency," *J. Geophys. Eng. 1*, pp. 245-251, 2004.