ABSTRACT

HYPERBOLIC POSITION LOCATION ESTIMATOR WITH TDOAS FROM FOUR STATIONS

by
Sreeram Potluri

This thesis presents a detailed derivation of a set of equations needed to locate the three dimensional position of a mobile given the locations of four fixed stations (like a global positioning system (GPS) satellite or a base station in a cell) and the signal time of arrival (TOA) from the mobile to each station. From these derived equations, a synthesizable VHDL model was developed and simulated using IEEE numeric_std package. All the inputs and outputs were described by 32 bit vectors. From the simulations, it was observed that in the best case the mobile position was off by 1 meter and in the worst case the position was off by 36 meters. This model was synthesized with Cadence tools and the total number of gates produced was 2.7 million.

HYPERBOLIC POSITION LOCATION ESTIMATOR WITH TDOAS FROM FOUR STATIONS

by
Sreeram Potluri

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

January 2002

APPROVAL PAGE

HYPERBOLIC POSITION LOCATION ESTIMATOR WITH TDOAS FROM FOUR STATIONS

Sreeram Potluri

Dr. Durga Misra, Thesis Advisor
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. William Carr, Committee Member
Professor of Electrical and Computer Engineering, NJIT

Dr. Symeon Papavassiliou, Committee Member
Assistant Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Sreeram Potluri

Degree: Master of Science

Date: January 2002

Undergraduate and Graduate Education

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2001

- Bachelor of Engineering in Electrical & Electronics Engineering,
  Andhra University, Visakhapatnam, AP, INDIA

Major: Electrical Engineering

To the Almighty

ACKNOWLEDGEMENT

The author wishes to express his sincere gratitude to his advisor, Professor Durga Misra, for his guidance and moral support throughout this work.

Special thanks to Professors William Carr and Symeon Papavassiliou for serving as the members of the committee.

The author also thanks Department of Electrical Engineering, New Jersey Institute of Technology for the financial assistance during summer 2001.

The author appreciates the technical support from the Cadence Design Systems for their EDA tools and Computing Services Division of NJIT.

Finally, the author would like to thank his beloved parents for supporting him through out his course of study at NJIT.

TABLE OF CONTENTS

TABLE OF CONTENTS
(continued)

Chapter **Page**

LIST OF FIGURES

Figure **Page**

LIST OF FIGURES
(continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Mobile Radio Position Location

Several different position location (PL) technologies present themselves as candidates for a mobile radio PL system. However, radio frequency (RF) PL systems have dominated the field because they offer the advantages of relatively low cost, ease of integration and potentially high accuracy. Radio frequency PL techniques also work with the existing cellular/PCS infrastructure, eliminating the need for external network implementations. Furthermore, radio frequency systems may operate, to a limited extent, in cases where other PL methods completely fail, such as when the line-of-sight (LOS) to the source is not available.

Radio frequency PL systems attempt to locate a source by direct measurements on radio signals traveling between the transmitter and receiver. These RF PL systems use time, phase or frequency measurements to first estimate the direction or range information of the signal propagation path, then utilize estimators that provide PL solutions from the measured data. The most widely used RF PL technique for geolocation of mobile users is the hyperbolic position location technique. The hyperbolic PL technique, also known as the time difference of arrival (TDOA) PL technique, utilizes cross-correlation techniques to estimate the TDOA of a propagating signal received at two receivers. This delay measurement defines a hyperbola of constant range difference from the receivers, which are located at the foci. When multiple receiving stations are used, multiple hyperbolas are formed, and the intersection of the set of hyperbolas

provides the PL estimate of the source. The hyperbolic position location technique offers the advantages of not requiring additional hardware or software within the mobile unit, ability to resolve ambiguities in the PL estimate and minimizing the effect of noise within the mobile radio channel.

Many organizations are developing competing products to comply with the FCC's E-911 mandate, which requires U.S. cellular carriers to provide location information of phone calls, effective October 2001. The accuracy required is 100 meters or better. Many of these products will implement the above-mentioned time difference of arrival technique for locating a mobile with varying degrees of accuracy. Methods for calculating the TDOA and mobile position have been reviewed previously [1][2]. Some methods calculate the two dimensional position and others the three-dimensional position depending on the degree of simplicity desired.

1.2 Design Flow of Position Location Estimation Chip

In this thesis, a more detailed derivation of a set of equations needed to locate the three dimensional position of a mobile is presented. This detailed derivation will be the basis for implementing a positioning algorithm in VHDL and designing an ASIC. The design flow for implementing the algorithm as an ASIC is as shown below. Mostly Cadence tools were used, except ModelSim for HDL simulations and HSPICE for spice simulations.

The VHDL model for the algorithm is developed from derived equations and compiled and simulated using MTI ModelSim. ModelSim is chosen on account of its user-friendly interface.

The VHDL model is synthesized using Cadence Ambit Bulidgates with Physically Knowledgeable Synthesis (PKS) option. In this, the design is synthesized, mapped to the cells in the library and then the cells are actually placed to using Ultra placer for calculating the exact wire lengths for timing calculations. The outputs from this

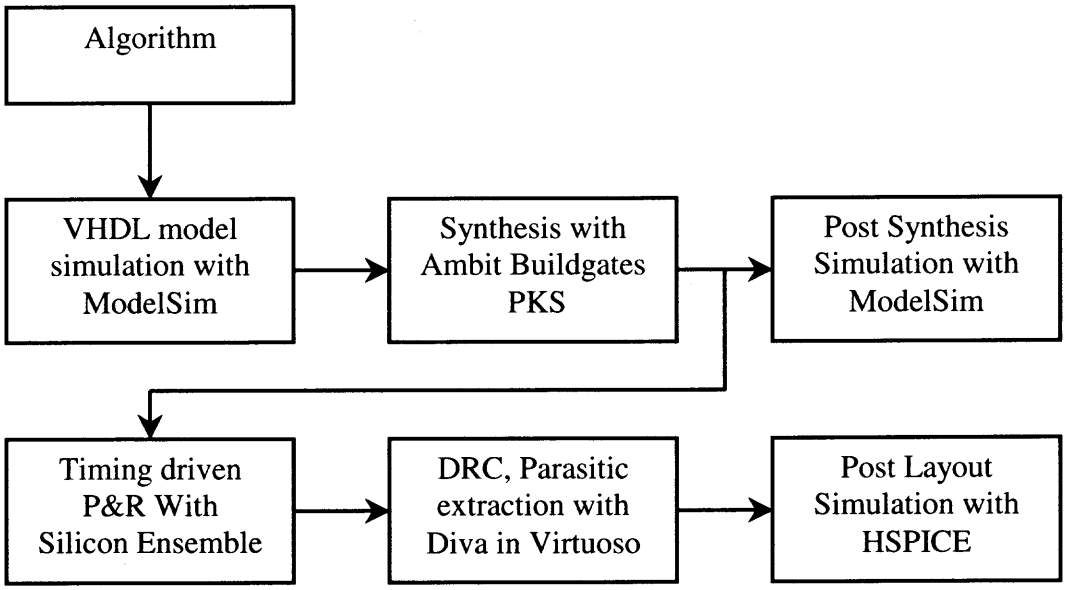

Figure 1.1 Position Location Estimation chip Design Flow.

tool are a Verilog file, which has gate level design information, GCF file which has timing information and DEF file which has placement information.

The placement and Routing of the standard cells is done using Cadence Silicon Ensemble. The design, placement and timing information is read from the above produced files and timing driven routing is done.

The DRC and parasitic extraction is done in Cadence Virtuoso environment using Assura Diva. The extracted spice file is simulated using HSPICE.

1.3 Outline of the Thesis

The remainder of the thesis is organized as follows. Chapter 2 provides an overview of various position location techniques such as direction finding PL, ranging PL, elliptical PL and hyperbolic PL. Finally a comparison between hyperbolic and direction finding techniques since they are most commonly used. Chapter 3 provides a detailed derivation of the algorithm and rearrangement of the equations for implementation in VHDL. Chapter 4 describes the VHDL model and different implementation issues. Then the test data and the simulation results are listed. Chapter 5 addresses the synthesis issues, problems encountered during the synthesis and modifications of the actual VHDL model to make it synthesizable. Then the simulations of the gate level verilog netlist after synthesis are listed. Chapter 6 describes the physical layout and device level simulation of the final layout. Chapter 6 concludes this thesis by summarizing the results of the work and discussing alternative implementations for reducing the gate count.

CHAPTER 2

POSITION LOCATION TECHNIQUES

2.1 Classification of PL Systems

Position location systems can be classified into two broad categories: direction finding (DF) and range-based PL systems [4]. Each of these systems can be classified as a satellite or terrestrial based system, indicating whether the base station is located on the surface of the earth or in orbit around the earth.

Direction finding systems estimate the position location of a source by measuring the direction of arrival (DOA), or angle of arrival (AOA), of the source's signal. The DOA measurement restricts the location of the source along a line in the estimated DOA. When multiple DOA measurements from multiple base stations are used in a triangulation configuration, the location estimate of the source is obtained at the intersection of these lines. Consequently, direction finding PL systems are also known as direction of arrival or angle of arrival PL systems.

Range-based PL systems can be categorized as a ranging, range sum, or range difference PL system [4]. The type of measurement used in each of these systems defines a unique geometry, or configuration, of the position location solution. Ranging PL systems locate the source by measuring the absolute distance between a source and the receiver. Range measurements are determined by estimating the time-of-arrival (TOA) of the signal propagating between the source and receiver. The TOA estimate defines a sphere of constant range around the receiver. The intersection of multiple spheres produced by multiple range measurements from multiple base stations pro- vides the

position location estimate of the user. Consequently, ranging systems are also known as TOA or spherical PL systems. Most practical ranging systems are unable to measure the range between the user and a base station directly, and as a result, measurement of the range and a bias term is commonly performed. This bias term can be calculated using an additional range measurement by an additional base station. Ranging systems of this type are often called pseudo-range systems.

Range sum PL systems measure the relative sum of ranges between the source and receiver respectively. These systems measure the time sum of arrival (TSOA) of the propagating signal between two base stations to produce a range sum measurement. The range sum estimate defines an ellipsoid around the receiver, and when multiple range sum measurements are obtained, the position location estimate of the user is at the intersection of the ellipsoids [4]. Consequently, range sum PL systems are also known as TSOA or elliptical PL systems.

Range difference PL systems measure the relative difference in ranges between the source and receiver respectively. These systems measure the time difference of arrival (TDOA) of the propagating signal between two base stations to produce a range difference measurement. The range difference measurement defines a hyperboloid of constant range difference with the base stations at the foci. When multiple range difference measurements are obtained, producing multiple hyperboloids, the position location estimate of the user is at the intersection of the hyperboloids [4]. Consequently, range difference PL systems are also known as TDOA or hyperbolic PL systems.

2.2 Direction Finding PL Systems

Direction finding (DF) systems utilize multi-array antenna and direction of arrival (DOA) techniques to estimate the direction of the signal of interest. The DOA measurement restricts the source location along a line in the estimated DOA. When multiple DOA measurements from multiple base stations are used in a triangulation configuration, the location estimate of the source is obtained at the intersection of these lines. Figure 2.1 illustrates the two dimensional (2-D) PL solution of DF systems. While only two DOA estimates are required to estimate the PL of a source, multiple DOA estimates are commonly used to improve the estimation accuracy.

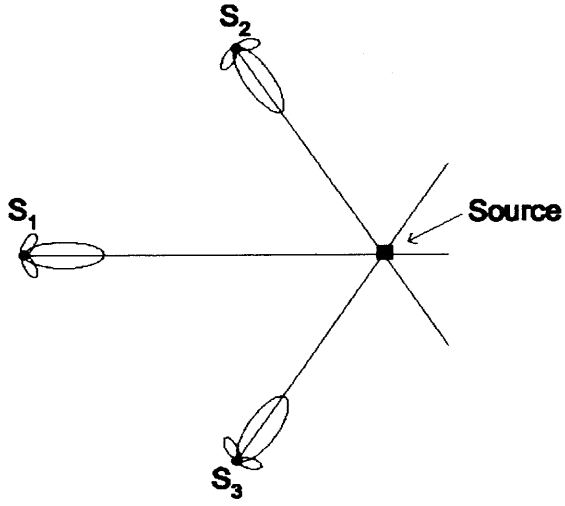

Figure 2.1 2D Direction Finding Position Location Solution.

Direction of arrival estimation is performed by signal parameter estimation algorithms which exploit the phase differences, or other signal characteristics, between closely spaced antenna elements of an antenna array and employ phase-alignment methods for beam/null steering. The spacing of antenna elements within the antenna

array is typically less than 1/2 wavelength of all received signals. This alignment is required to produce phase differences on the order of π radians or less to avoid ambiguities in the DOA estimate. The resolution of DOA estimators improves as the baseline distances between antenna elements increases. However, this improvement is at the expense of ambiguities. As a result, DOA estimation methods are often used with Short baselines to reduce or eliminate the ambiguities and long baselines to improve resolution.

Although, direction-finding methods can provide accurate DOA estimation given the appropriate conditions, they do suffer from elements encountered within the mobile radio channel. First, DOA estimation techniques estimate the direction of a source based on the strongest received signal, which is assumed to be the line-of-sight (LOS) signal. However, in shadowed environments such as encountered in urban areas, the surrounding environment may obstruct the true LOS signal path and only multipath components of the signal may exist. In this case, the DOA estimate will be the direction of the strongest multipath component, which leads to errors in the DOA estimate. Depending on the transmitter-receiver distance, these errors in the DOA estimate can lead to dramatic errors in the PL estimate. Even if the LOS signal is available, multipath has been shown to severely degrade the accuracy of DF methods. While angular accuracy's of several degrees are possible with these techniques, this generally does not provide an acceptable position location accuracy when using the triangulation configuration solution.

2.3 Ranging PL Systems

Ranging PL systems measure the absolute distance between a source and a set of base stations through the use of time-of-arrival (TOA) measurements. The TOA measurements are related to range estimates that define a sphere around the receiver. When measurements are made from receivers with known locations, the spheres described by the range measurements intersect at a unique point indicating the position location estimate of the source. Figure 2.2 illustrates the three dimensional (3-D) solution of the ranging PL system. If the spheres described by the range measurements intersect at more than one point, an ambiguous solution to the position location estimate results. Redundant range measurements, resulting in a multilateration ranging PL estimation, are commonly made to reduce or eliminate PL ambiguities.

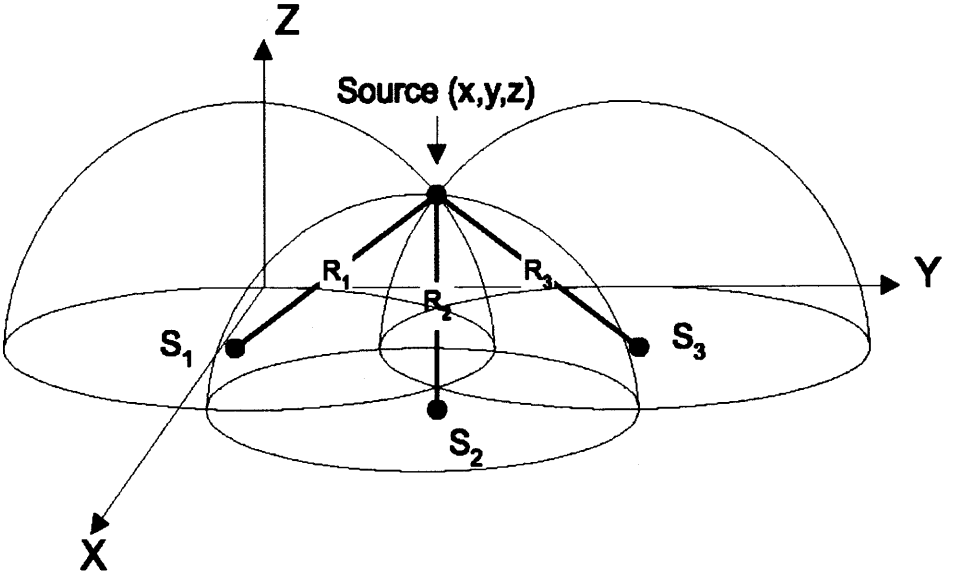

Figure 2.2 3D Ranging Position Location Solution.

To illustrate the ranging PL concept, consider a 3-D ranging PL system using N base stations. The time of arrival of a signal at each receiver is estimated and related to the range measurement by the relationship

$$R_i = cd_i \tag{2.1}$$

where R_i is the range measurement, c is the signal propagation speed and d_i is the TOA estimate at the i^{th} receiver. The mathematical relationships between range measurements at N base stations, the coordinates of the known base station locations, and the coordinates of the source are

$$R_i = \sqrt{(X_i - x)^2 + (Y_i - y)^2 + (Z_i - z)^2} \quad \text{for i} = 1,2,3...N \tag{2.2}$$

where (X_i, Y_i, Z_i) are the coordinates of the i^{th} base station, R_i is the i^{th} range estimate to the source and (x, y, z) is the location of the user. Above equation defines an N x 3 set of nonlinear equations whose solution is the location coordinates of the source. If the number of unknowns, or coordinates of the source to be solved, is equal the number of range measurements, the set of equations are consistent and a unique solution exists. However, if redundant measurements produce more range measurements than the number of unknowns, then the system is inconsistent and a unique solution may or may not exist. This generally requires an error criterion to be selected and iterative techniques to be employed to produce a solution. A least squares (LS) is commonly used to simultaneously solve these equations for both the position location and error coefficients.

Accurate time or phase measurements in ranging PL systems require strict clock synchronization between the source and base stations. This is accomplished through the use of stable clocks, such as the rubidium or cesium standard clocks used in GPS satellites, at both the source and base station. As such, ranging PL system may require

additional hardware implementation in a mobile unit, resulting in additional power, size and weight requirements. A disadvantage of the ranging PL technique is that accuracy is very dependent on system geometry. Highest accuracies are attained when all ranging spheres intersect at 90 degrees. Degradation in performance is experienced as the intersections deviate from this angle. For systems with fixed receivers and moving sources, such as cellular and PCS systems, the optimum situation will rarely be attained. Another disadvantage of this PL technique is that the errors in the TOA estimate common to all receivers are not treated before the position location estimate.

2.4 Elliptical PL Systems

Elliptical PL systems locate a source by the intersection of ellipsoids describing the range sum measurements between multiple receivers. Figure 2.3 illustrates the 2-D solution of an elliptical location system.

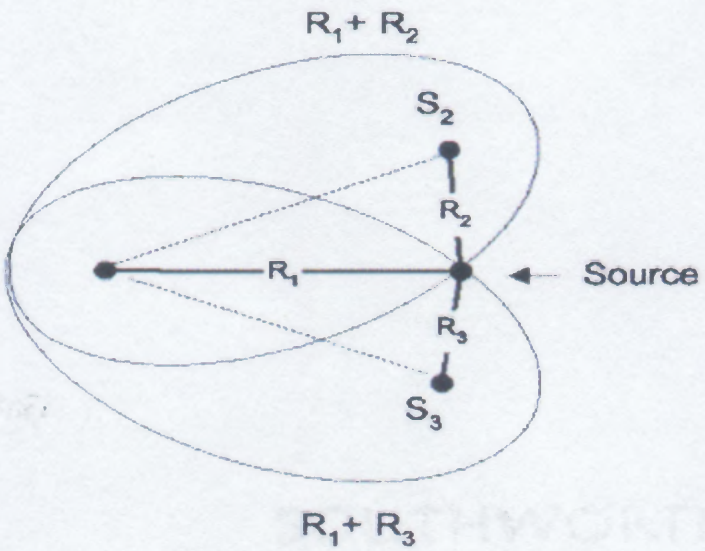

Figure 2.3 2D Elliptical Position Location Solution.

The range sum is determined from the sum of signal TOA's at multiple receivers. The relationship between range sum, $R_{i,j}$ and the TOA between receivers is given by

$$R_{i,j} = cd_{i,j} = Ri + R_j \tag{2.3}$$

where c is the signal propagation speed and $d_{i,j}$ is the sum of TOA at receiver i and j. The range sum measurement restricts the possible source locations to an ellipsoid. The ellipsoids that describe the range sum between receivers is given by

$$R_{i,j} = \sqrt{(X_i - x)^2 + (Y_i - y)^2 + (Z_i - z)^2} + \sqrt{(X_j - x)^2 + (Y_j - y)^2 + (Z_j - z)^2} \tag{2.4}$$

where (X_i, Y_i, Z_i) and(X_j, Y_j, Z_j) define the location of receiver i and j, and(x, y, z) is the position location estimate of the source. A source location can be uniquely determined by the intersection of three or more ellipsoids. Redundant range sum measurements can be made to improve the accuracy and resolve location solution ambiguities. This method offers the advantage of not requiring high precision clocks at the mobile. While there exist some systems that use this method, it appears that it offers no performance advantage over the spherical or hyperbolic configurations.

2.5 Hyperbolic PL Systems

Hyperbolic position location systems estimate the location of a source by the intersection of hyperboloids describing range difference measurements between three or more base stations. The range difference between two receivers is determined by measuring the difference in time of arrival of a signal between them. The relationship between range difference and the TDOA between receivers is given by

$$R_{i,j} = cd_{i,j} = R_i - R_j \tag{2.5}$$

where c is the signal propagation speed and di,j is the TDOA between receiver i and j. The TDOA estimate, in the absence of noise and interference, restricts the possible source locations to a hyperboloid of revolution with the receiver as the foci. Figure 2.4 illustrates a 2-D hyperbolic position location solution. In a 3-D system, the hyperboloids that describe the range difference, $R_{i,j}$ between receivers are given by

$$R_{i,j} = \sqrt{(X_i - x)^2 + (Y_i - y)^2 + (Z_i - z)^2} - \sqrt{(X_j - x)^2 + (Y_j - y)^2 + (Z_j - z)^2} \qquad (2.6)$$

where (X_i, Y_i, Z_i) and (X_j, Y_j, Z_j) define the location of receiver i and j respectively, $R_{i,j}$ is the range difference measurement between base station i and j, and (x, y, z) are the unknown coordinates of the source. If the number of unknowns, or coordinates of the source to be determined, is equal to the number of equations, or range difference measurements, then the system is consistent and a unique solution exist. However, if redundant range difference measurements are made, then the system may be inconsistent and a unique solution may or may not exist. In this situation, some error criteria must be selected for determining the optimum solution to the system of equations.

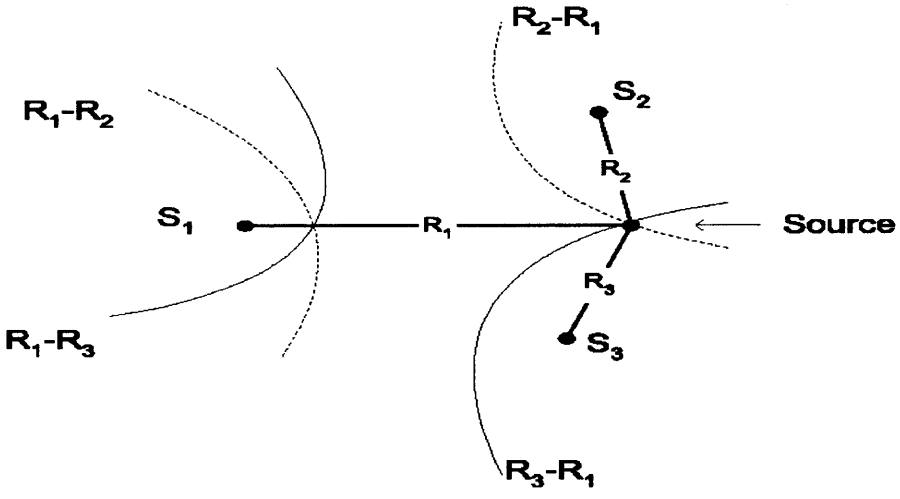

Figure 2.4 Two-Dimensional Hyperbolic Position Location Solution.

If the source and receivers are coplanar, two-dimensional (2-D) source location can be estimated from the intersection of two or more hyperboloids produced from three or more TDOA measurements, resulting in a hyperbolic trilateration solution. Three dimensional (3-D) source location estimation is produced by the intersection of three or more independently generated hyperboloids generated from four or more TDOA measurements, resulting in a hyperbolic multilateration solution. If the hyperbola determined from multiple receivers intersects at more than one point, then ambiguity in the estimated position exists. This location ambiguity may be resolved by using a priori information about the source location, bearing measurements at one or more of the stations, or redundant range difference measurements at additional base station to generate additional hyperbolas.

A major advantage of this TDOA method is that it does not require knowledge of the transmit time from the source, as do TOA methods. Consequently, strict clock synchronization between the source and receiver is not required. As a result, hyperbolic position location techniques do not require additional hardware or software implementation within the mobile unit. However, clock synchronization is required of all receivers used for the PL estimate. Furthermore, unlike TOA methods, the hyperbolic position location method is able to reduce or eliminate common errors experienced at all receivers due to the channel.

2.6 Hyperbolic versus DF PL Systems

The two most commonly used PL techniques are direction finding (DF) and hyperbolic methods [2]. While DF systems exploit the relative phase differences between closely

spaced antenna elements and employ phase-alignment methods for beam/null steering to estimate the direction of arrival (DOA) of the signal of interest, hyperbolic methods exploit the relative time differences of a signal arriving at different receivers.

The requirements on accuracy and spatial resolution capabilities of array-based DF methods become more stringent as the distance between the sources and the receiving platform increases, since this decreases the differences between DOA's of the sources at the platform. In contrast, the requirements for accuracy and temporal resolution capabilities of time difference of arrival (TDOA) based methods become less stringent as the relative distance between base stations and the source increases, since this increases the TDOA between them.

The need for high resolution arises primarily when closely spaced sources give rise to multiple received signals that cannot be separated by preprocessing methods before the PL estimate is made. For instance, when cross correlating TDOA's of multiple signals that are not separated by more than the widths of their cross-correlation peaks, the peak cross correlation of the signal of interest usually cannot be resolved with conventional TDOA-based methods. To minimize this problem, the distance between platforms is typically made as large as possible to minimize overlap of adjacent peaks. This presents a fundamental resolution-limit problem for TDOA estimation of two closely spaced sources. The best performing array-based DF methods attempt to resolve the resolution problem in locating multiple signal sources by simultaneously estimating multiple DOA's rather than estimating the DOA of each signal as is commonly done by conventional beam formers and TDOA-based techniques.

Although DF techniques offer greater spatial resolution and the ability to simultaneously locate a number of signals, their complexity is typically much higher than that of TDOA techniques. They tend to be more complex because of the need for measurement, storage and usage of large amounts of array calibration data and because of the computationally intensive algorithms. The tradeoff between these systems is the highly complexity high-resolution array-based DF methods and the simplicity of TDOA based methods that require widely separated base stations.

CHAPTER 3

THE ALGORITHM

3.1 Hyperbolic Equation Solving Algorithms

The TDOA obtained estimates are converted into range difference measurements and these measurements can be converted into nonlinear hyperbolic equations. As these equations are non-linear, solving them is not a trivial operation. Several algorithms have been proposed for this purpose having different complexities. First the mathematical model that is used by these algorithms is discussed, which is then followed by the algorithms that can be used for solving hyperbolic equations.

3.1.1 Mathematical Model for Hyperbolic TDOA Equations

A general model for the three dimensional (3-D) PL estimation of a source using M base stations is developed. Referring all TDOAs to the first base station, which is assumed to be the base station controlling the call and the first to receive the transmitted signal, let the index i = 2,…,M, unless otherwise specified, (x, y, z) be the source location and (X_i, Y_i, Z_i) be the known location of the i^{th} receiver. The squared range distance between the source and the i^{th} receiver is given as

$$R_i = \sqrt{(X_i - x)^2 + (Y_i - y)^2 + (Z_i - z)^2} \tag{3.1}$$

The range difference between base stations with respect to the base station where the signal arrives first, is

$$\begin{aligned} R_{i,1} &= cd_{i,1} = R_i - R_1 \\ &= \sqrt{(X_i - x)^2 + (Y_i - y)^2 + (Z_i - z)^2} - \sqrt{(X_1 - x)^2 + (Y_1 - y)^2 + (Z_1 - z)^2} \end{aligned} \tag{3.2}$$

where c is the speed of light, $R_{i,1}$ is the range difference distance between the first base station and the i^{th} base station, R_1 is the distance between the first base station and the source, and $d_{i,1}$ is the estimated TDOA between the first base station and the i^{th} base station. This defines the set of nonlinear hyperbolic equations whose solution gives the 3-D coordinates of the source. Solving the nonlinear equations of (3.2) is difficult. Consequently, linearizing this set of equations is commonly performed using the following algorithms.

3.1.2 Taylor-Series Method

The Taylor-series method linearizes the set of equations in (3.2) by Taylor-series expansion, then uses an iterative method to solve the system of linear equations. The iterative method begins with an initial guess and improves the estimate at each iteration by determining the local linear least-square solution. The Taylor-series can provide accurate results and is robust. It can also make use of redundant measurements to improve the PL solution. However, it requires a good initial guess and can be computationally intensive. For most situations, linearization of the nonlinear equations does not introduce undue errors in the position location estimate

.

3.1.3 Fang's Method

For arbitrarily placed base stations and a consistent system of equations in which the number of equations equals the number of unknown source coordinates to be solved, Fang [1] provides an exact solution to the equations of (3.2). However, his solution does not make use of redundant measurements made at additional receivers to improve

position location accuracy. Furthermore, his method experiences an ambiguity problem due to the inherent squaring operation. These ambiguities can be resolved using a priori information or through the use of symmetry properties. Unlike the algorithms mentioned previously, this method provides a closed form and exact solution and it is also computationally less intensive than the Taylor-series method.

3.1.4 Chan's Method

A non-iterative solution to the hyperbolic position estimation problem which is capable of achieving optimum performance for arbitrarily placed sensors was proposed by Chan [5]. The solution is in closed-form and valid for both distant and close sources. When TDOA estimation errors are small, this method is an approximation to the maximum likelihood estimator. It provides an explicit solution form that is not available in the Taylor-series method. It is also better than Fang's method as it can take advantage of redundant measurements like the Taylor-series method. However, it needs a priori information to resolve an ambiguity in its calculations like the Fang's method.

3.2 Derivation of the Hyperbolic Position Location Algorithm

The time difference of arrivals (TDOA) at a pair of stations locate the navigator on a hyperboloid of revolution and time difference of arrivals at three stations place the navigator on the curve of intersection of two such hyperboloids. To fix the position at a point on this curve of intersection (ellipse or hyperbola) a fourth satellite is required [1]. The rest of this chapter describes the derivation of the solution for the position fix using the fourth satellite.

The distance between a mobile and a station is determined indirectly by measuring the time it takes for a signal to reach the station from the mobile. Multiplying the TOA t by the signal velocity c gives us the distance R.

It is needed to solve for the three unknowns x, y and z (mobile position). Therefore, equation (3.1) is expanded to three equations when the specific locations of three satellites i, j and k are given. This requirement can be easily met since GPS satellites broadcast their exact locations.

$$ct_i = R_i = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} \tag{3.3}$$

$$ct_j = R_j = \sqrt{(x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2} \tag{3.4}$$

$$ct_k = R_k = \sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} \tag{3.5}$$

Unfortunately, solving the three equations for three unknowns will not lead to a simple and satisfactory solution because of the square root terms. The solution can be simplified by adding another satellite l for an additional equation. This requirement is easily met since four GPS satellites are guaranteed to be in the horizon of any location on earth. The four equations will be combined to form expressions for time difference of arrivals (TDOAs) R_{ij}, R_{ik}, R_{kj} and R_{kl}.

$$R_i - R_j = R_{ij} = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} - \sqrt{(x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2} \tag{3.6}$$

$$R_i - R_k = R_{ik} = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} - \sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} \tag{3.7}$$

$$R_k - R_j = R_{kj} = \sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} - \sqrt{(x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2} \tag{3.8}$$

$$R_k - R_l = R_{kl} = \sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} - \sqrt{(x_l - x)^2 + (y_l - y)^2 + (z_l - z)^2} \tag{3.9}$$

Moving one square root term to the other side gives us:

$$R_{ij} - \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} = -\sqrt{(x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2} \quad (3.10)$$

$$R_{ik} - \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} = -\sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} \quad (3.11)$$

$$R_{kj} - \sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} = -\sqrt{(x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2} \quad (3.12)$$

$$R_{kl} - \sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} = -\sqrt{(x_l - x)^2 + (y_l - y)^2 + (z_l - z)^2} \quad (3.13)$$

Squaring both sides produces the following set of equations:

$$R_{ij}^2 - 2R_{ij}\sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} + (x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 = (x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2 \quad (3.14)$$

$$R_{ik}^2 - 2R_{ik}\sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} + (x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 = (x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2 \quad (3.15)$$

$$R_{kj}^2 - 2R_{kj}\sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} + (x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2 = (x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2 \quad (3.16)$$

$$R_{kl}^2 - 2R_{kl}\sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} + (x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2 = (x_l - x)^2 + (y_l - y)^2 + (z_l - z)^2 \quad (3.17)$$

Expanding the squared terms to the left of the square root term produces:

$$R_{ij}^2 - 2R_{ij}\sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} + x_i^2 - 2x_i x + x^2 + y_i^2 - 2y_i y + y^2 + z_i^2 - 2z_i z + z^2 = x_j^2 - 2x_j x + x^2 + y_j^2 - 2y_j y + y^2 + z_j^2 - 2z_j z + z^2 \quad (3.18)$$

$$R_{ik}^2 - 2R_{ik}\sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} + x_i^2 - 2x_i x + x^2 + y_i^2 - 2y_i y + y^2 + z_i^2 - 2z_i z + z^2 = x_k^2 - 2x_k x + x^2 + y_k^2 - 2y_k y + y^2 + z_k^2 - 2z_k z + z^2 \quad (3.19)$$

$$R_{kj}^2 - 2R_{kj}\sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} + x_k^2 - 2x_k x + x^2 + y_k^2 - 2y_k y + y^2 + z_k^2 - 2z_k z + z^2 = x_j^2 - 2x_j x + x^2 + y_j^2 - 2y_j y + y^2 + z_j^2 - 2z_j z + z^2 \quad (3.20)$$

$$R_{kl}^2 - 2R_{kl}\sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} + x_k^2 - 2x_k x + x^2 + y_k^2 - 2y_k y + y^2 + z_k^2 - 2z_k z + z^2 = x_l^2 - 2x_l x + x^2 + y_l^2 - 2y_l y + y^2 + z_l^2 - 2z_l z + z^2 \quad (3.21)$$

Eliminating the x^2, y^2 and z^2 terms reduces the equation set to:

$$R_{ij}^2 - 2R_{ij}\sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} + x_i^2 - 2x_i x + y_i^2 - 2y_i y + z_i^2 - 2z_i z = x_j^2 - 2x_j x + y_j^2 - 2y_j y + z_j^2 - 2z_j z \quad (3.22)$$

$$R_{ik}^2 - 2R_{ik}\sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} + x_i^2 - 2x_i x + y_i^2 - 2y_i y + z_i^2 - 2z_i z = x_k^2 - 2x_k x + y_k^2 - 2y_k y + z_k^2 - 2z_k z \quad (3.23)$$

$$R_{kj}^2 - 2R_{kj}\sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} + x_k^2 - 2x_k x + y_k^2 - 2y_k y + z_k^2 - 2z_k z = x_j^2 - 2x_j x + y_j^2 - 2y_j y + z_j^2 - 2z_j z \quad (3.24)$$

$$R_{kl}^2 - 2R_{kl}\sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} + x_k^2 - 2x_k x + y_k^2 - 2y_k y + z_k^2 - 2z_k z = x_l^2 - 2x_l x + y_l^2 - 2y_l y + z_l^2 - 2z_l z \quad (3.25)$$

Shifting all but the square root term to the right and combining similar terms produces

$$\sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} = [R_{ij}^2 + x_i^2 - x_j^2 + y_i^2 - y_j^2 + z_i^2 - z_j^2 + 2x_j x - 2x_i x + 2y_j y - 2y_i y + 2z_j z - 2z_i z]/2R_{ij} \quad (3.26)$$

$$\sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} = [R_{ik}^2 + x_i^2 - x_k^2 + y_i^2 - y_k^2 + z_i^2 - z_k^2 + 2x_k x - 2x_i x + 2y_k y - 2y_i y + 2z_k z - 2z_i z]/2R_{ik} \quad (3.27)$$

$$\sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} = [R_{kj}^2 + x_k^2 - x_j^2 + y_k^2 - y_j^2 + z_k^2 - z_j^2 + 2x_j x - 2x_k x + 2y_j y - 2y_k y + 2z_j z - 2z_k z]/2R_{kj} \quad (3.28)$$

$$\sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} = [R_{kl}^2 + x_k^2 - x_l^2 + y_k^2 - y_l^2 + z_k^2 - z_l^2 + 2x_l x - 2x_k x + 2y_l y - 2y_k y + 2z_l z - 2z_k z]/2R_{kl} \quad (3.29)$$

The equation set can now be simplified by substituting x_{ji} for $x_j - x_i$, y_{ji} for $y_j - y_i$ and so on.

$$\sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} = [R_{ij}^2 + x_i^2 - x_j^2 + y_i^2 - y_j^2 + z_i^2 - z_j^2 + 2x_{ji} x + 2y_{ji} y + 2z_{ji} z]/2R_{ij} \quad (3.30)$$

$$\sqrt{(x_i-x)^2+(y_i-y)^2+(z_i-z)^2} = [R_{ik}^2+x_i^2-x_k^2+y_i^2-y_k^2+z_i^2-z_k^2+2x_{ki}x+2y_{ki}y+2z_{ki}z]/2R_{ik} \quad (3.31)$$

$$\sqrt{(x_k-x)^2+(y_k-y)^2+(z_k-z)^2} = [R_{kj}^2+x_k^2-x_j^2+y_k^2-y_j^2+z_k^2-z_j^2+2x_{jk}x+2y_{jk}y+2z_{jk}z]/2R_{kj} \quad (3.32)$$

$$\sqrt{(x_k-x)^2+(y_k-y)^2+(z_k-z)^2} = [R_{kl}^2+x_k^2-x_l^2+y_k^2-y_l^2+z_k^2-z_l^2+2x_{lk}x+2y_{lk}y+2z_{lk}z]/2R_{kl} \quad (3.33)$$

Equations (3.6), (3.7), (3.8) and (3.9) are now in a useful arrangement. Equations (3.30), (3.31), (3.32) and (3.33), when squared, are intersecting hyperboloids. By equating equations (3.30) and (3.31) to form equation (3.34), a plane equation in the form of $y=Ax+By+C$ can be derived by rearranging the terms as shown in equations (3.35) and (3.36).

$$[R_{ij}^2+x_i^2-x_j^2+y_i^2-y_j^2+z_i^2-z_j^2+2x_{ji}x+2y_{ji}y+2z_{ji}z]/2R_{ij} = [R_{ik}^2+x_i^2-x_k^2+y_i^2-y_k^2+z_i^2-z_k^2+2x_{ki}x+2y_{ki}y+2z_{ki}z]/2R_{ik} \quad (3.34)$$

$$R_{ik}[R_{ij}^2+x_i^2-x_j^2+y_i^2-y_j^2+z_i^2-z_j^2]/2-R_{ij}[R_{ik}^2+x_i^2-x_k^2+y_i^2 -y_k^2+z_i^2-z_k^2]/2 = R_{ij}[x_{ki}x+y_{ki}y+z_{ki}z]-R_{ik}[x_{ji}x+y_{ji}y+z_{ji}z] \quad (3.35)$$

$$x[R_{ij}x_{ki}-R_{ik}x_{ji}]+y[R_{ij}y_{ki}-R_{ik}x_{ji}]+z[R_{ij}z_{ki}-R_{ik}z_{ji}] = R_{ik}[R_{ij}^2+x_i^2 -x_j^2+y_i^2-y_j^2+z_i^2-z_j^2]/2-R_{ij}[R_{ik}^2+x_i^2-x_k^2+y_i^2-y_k^2+z_i^2-z_k^2]/2 \quad (3.36)$$

Equation (3.36) is now in the desired form of a plane equation as follows:

$$y = Ax+Bz+C \quad (3.37)$$

$$where \quad A=[\frac{R_{ik}x_{ji}-R_{ij}x_{ki}}{R_{ij}y_{ki}-R_{ik}y_{ji}}] \quad (3.38)$$

$$and \quad B=[\frac{R_{ik}z_{ji}-R_{ij}z_{ki}}{R_{ij}y_{ki}-R_{ik}y_{ji}}] \quad (3.39)$$

$$and \quad C = \frac{R_{ik}[R_{ij}^2 + x_i^2 - x_j^2 + y_i^2 - y_j^2 + z_i^2 - z_j^2] - R_{ij}[R_{ik}^2 + x_i^2 - x_k^2 + y_i^2 - y_k^2 + z_i^2 - z_k^2]}{2[R_{ij}y_{ki} - R_{ik}y_{ji}]} \quad (3.40)$$

Similarly, equating equations (3.32) and (3.33) produces a second plane equation $y=Dx+Ey+F$. The resulting set of equations are:

$$y = Dx + Ez + F \quad (3.41)$$

$$where \quad D = [\frac{R_{kl}x_{jk} - R_{kj}x_{lk}}{R_{kj}y_{lk} - R_{kl}y_{jk}}] \quad (3.42)$$

$$and \quad E = [\frac{R_{kl}z_{jk} - R_{kj}z_{lk}}{R_{kj}y_{lk} - R_{kl}y_{jk}}] \quad (3.43)$$

$$and \quad F = \frac{R_{kl}[R_{kj}^2 + x_k^2 - x_j^2 + y_k^2 - y_j^2 + z_k^2 - z_j^2] - R_{kj}[R_{kl}^2 + x_k^2 - x_l^2 + y_k^2 - y_l^2 + z_k^2 - z_l^2]}{2[R_{kj}y_{lk} - R_{kl}y_{jk}]} \quad (3.44)$$

Equating the plane equations (3.37) and (3.41) produces a linear equation for x in terms of z.

$$Ax + Bx + C = Dx + Ez + F \quad (3.45)$$

$$x = Gz + H \quad (3.46)$$

$$where \quad G = \frac{E - B}{A - D} \quad (3.47)$$

$$and \quad H = \frac{F - C}{A - D} \quad (3.48)$$

Substituting equation (3.46) back into equation (3.37) produces a linear equation for y in terms of z.

$$y = A(Gz + H) + Bz + C \quad (3.49)$$

$$y = Iz + J \quad (3.50)$$

$$where \quad I = AG + B \quad (3.51)$$

$$and \quad J = AH + C \tag{3.52}$$

Equations (3.46) and (3.50) are now substituted back into equation (3.31) .

$$\begin{aligned} &2R_{ik}\sqrt{(x_i-(Gz+H))^2+(y_i-(Iz+J))^2+(z_i-z)^2} = \\ &[R_{ik}^2+x_i^2-x_k^2+y_i^2-y_k^2+z_i^2-z_k^2+2x_{ki}(Gz+H)+2y_{ki}(Iz+J)+2z_{ki}z] \end{aligned} \tag{3.53}$$

$$\begin{aligned} &2R_{ik}\sqrt{(G^2z^2-2Gz(x_i-H)+(x_i-H)^2)+(I^2z^2-2Iz(y_i-J)+(y_i-J)^2)+(z^2-2z_iz+z_i^2)} \\ &=Lz+K \end{aligned} \tag{3.54}$$

$$where \quad K = R_{ik}^2 + x_i^2 - x_k^2 + y_i^2 - y_k^2 + z_i^2 - z_k^2 + 2x_{ki}H + 2y_{ki}J \tag{3.55}$$

$$and \quad L = 2[x_{ki}G + y_{ki}I + 2z_{ki}] \tag{3.56}$$

Squaring on both sides

$$\begin{aligned} &4R_{ik}^2[G^2z^2+I^2z^2+z^2-2Gz(x_i-H)-2Iz(y_i-J)- \\ &2z_iz+(x_i-H)^2+(y_i-J)^2+z_i^2]=L^2z^2+2KLz+K^2 \end{aligned} \tag{3.57}$$

$$\begin{aligned} &4R_{ik}^2[G^2+I^2+1]z^2-8R_{ik}^2[G(x_i-H)+I(y_i-J)+z_i]z+ \\ &4R_{ik}^2[(x_i-H)^2+(y_i-J)^2+z_i^2]=L^2z^2+2KLz+K^2 \end{aligned} \tag{3.58}$$

To obtain z, equation (3.58) is rearranged into a binomial equation.

$$Mz^2 - Nz + O = 0 \tag{3.59}$$

$$where \quad M = 4R_{ik}^2[G^2 + I^2 + 1] - L^2 \tag{3.60}$$

$$and \quad N = 8R_{ik}^2[G(x_i - H) + I(y_i - J) + z_i] + 2LK \tag{3.61}$$

$$and \quad O = 4R_{ik}^2[(x_i - H)^2 + (y_i - J)^2 + z_i^2] - K^2 \tag{3.62}$$

The solution for z is:

$$z = \frac{N}{2M} \pm \sqrt{\left(\frac{N}{2M}\right)^2 - \frac{O}{M}} \tag{3.63}$$

The z coordinate can be put back into the linear equations (3.46) and (3.50) to solve for the coordinates x and y.

CHAPTER 4

THE VHDL MODEL

4.1 Description of the VHDL Model

The VHDL model was constructed using the above equations for x, y and z. IEEE numeric_std package was used to realize all the arithmetic operators. The input signals of the model are the *x, y, z* positions of four GPS satellites *i, j, k, l* in meters, and the signal TOAs from the individual satellites to the mobile in nanoseconds. The input signal assignments are *xi, yi, zi, ti, xj, yj, zj, tj, xk, yk, zk, tk, xl, yl, zl and tl.*

GPS satellite altitudes are approximately 10,900 nautical miles (20,186,800 meters). Therefore, the TOA range is roughly 6,700,000 to 7,600,000 ns. This means the input signals can be adequately described by a 32-bit vector. In order to perform signed arithmetic operations, the input signal assignments are of type SIGNED. The binary representation for negative numbers is 2's complement.

The TDOAs are converted to range differences by multiplying with binary representation of 100,000, and then dividing the result by the binary representation of 333,564 ns/m. This is equivalent to division by speed of light.

Since all signal and variable assignments are vectors representing integers, a method for maintaining adequate precision in divide and square root operations is needed. This will be achieved by multiplying the numerator by the binary representation of 1.0×10^{10} in divide operations. This method is preferred to using decimal point notation to decrease the complexity of the model. However, the length of the vectors

increases for successive multiplication operations, leading to a 200-bit vector for the interim value *O*.

The numeric_std package does not contain an overloaded square root operator. Therefore, bisection algorithm is used to compute the integer square root of a positive integer represented by a 64-bit vector. 64 bits is deemed adequate since the position z and the square root term cannot be larger than 32 bits by definition.

The square root operation gives two values for *z*, so the output signals *z1, z2, x1, x2, y1, y2* are for two possible mobile positions. The *z* value representing the mobile position can be determined by using a fifth satellite, or checking if the value is in the horizon of the four satellites relative to earth. The VHDL model is listed in Appendix 1.

4.2 Simulation of VHDL Model

The VHDL model developed was compiled and simulated with MTI ModelSim. VHDL test benches representing two real life situations were used to simulate the model. For ease of understanding the outputs were converted to base 10 in the test benches. The results from the simulations were considerably accurate. In the best case, the x coordinate was off by 1 meter and in the worst case the y coordinate was off by 36 meters.

Figure 4.1 shows one real life situation. It shows the positions of the four satellites and the mobile and the TOAs to the satellites.

The VHDL Model is simulated with the data in Figure 4.1 and the output results are shown in Figure 4.2. In this case, x1, y1, z1 is the solution, since the other one is not in the horizon of the satellites. It can be seen that the x coordinate is off by one meter.

Figure 4.3 shows another real life situation. It also shows the positions of the four satellites and the mobile and the TOAs to the satellites.

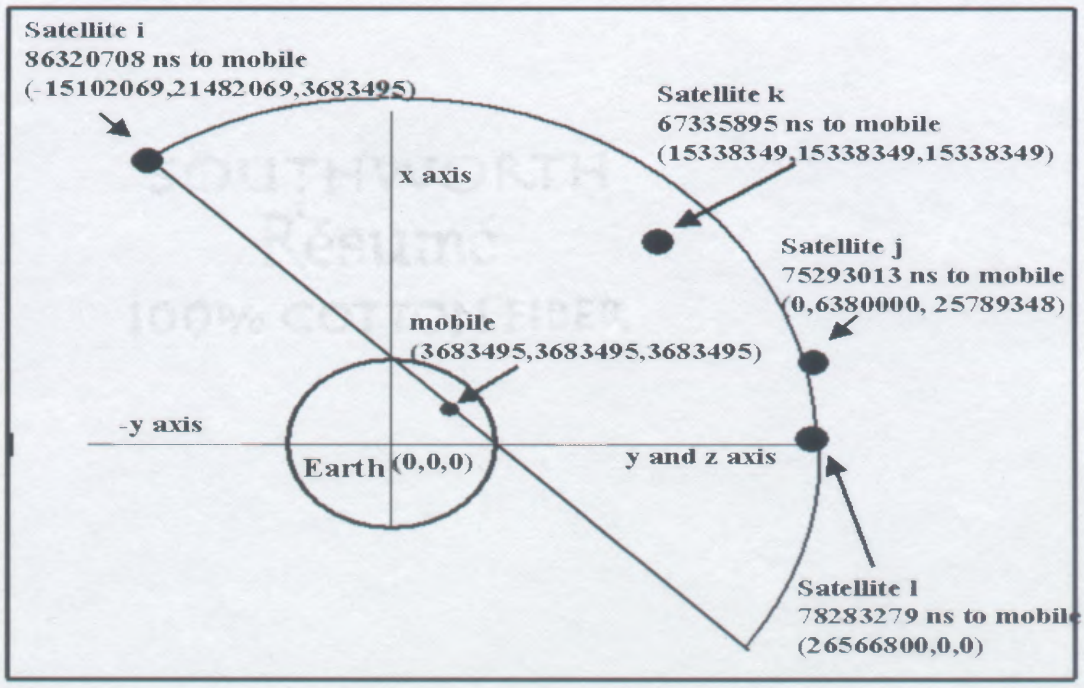

Figure 4.1 Real life situation 1.

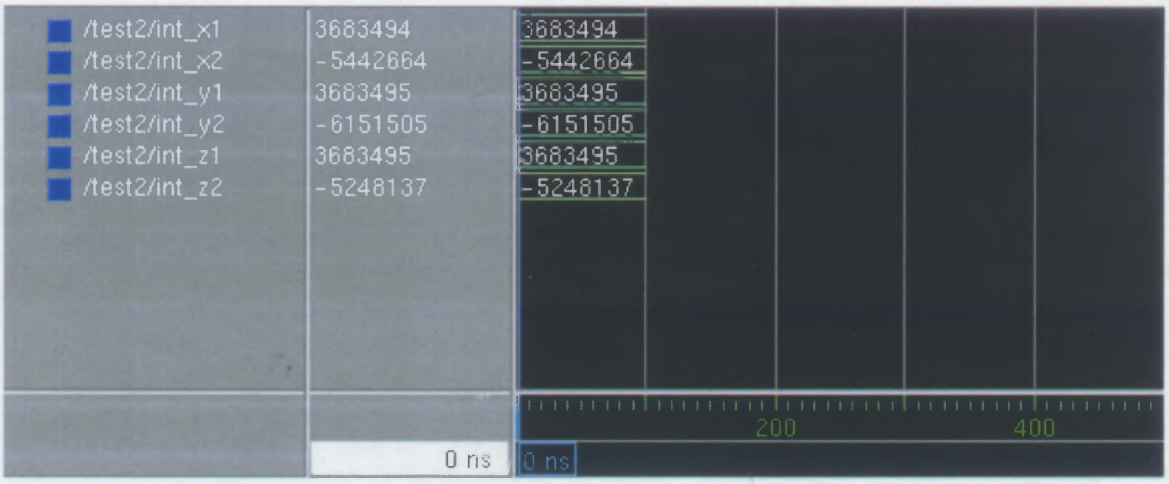

Figure 4.2 Simulation results real life situation 1.

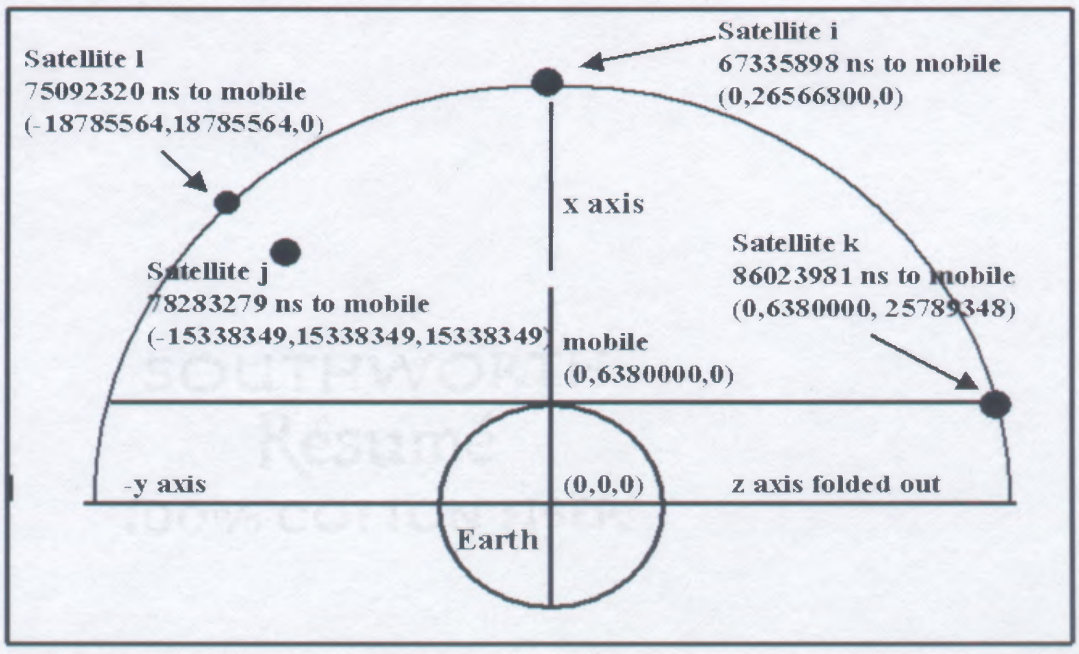

Figure 4.3 Real life situation 2.

Figure 4.4 Simulation results of real life situation 2.

The VHDL model is again simulated with the data in Figure 4.3. Figure 4.4 shows the simulation results. It can be seen that x coordinate is off by 1 meter and y coordinate is off by 36 meters.

CHAPTER 5

SYNTHESIS OF THE VHDL MODEL

5.1 Synthesis Issues

The VHDL model is synthesized with Cadence Ambit Bulidgates. However, it was not synthesizable as it is. There were no build in implementations for the division operation in the tool. So data sheets of various synthesis tools were studied and it was observed that none of them has built in implementation for division operation.

To solve this problem, a division algorithm should be modeled in VHDL as a function and called in the architecture. For this purpose different division algorithms were studied carefully. They are listed below.

1. Restoring division algorithm
2. Non restoring division algorithm
3. Multiplicative division algorithm
4. Division by Newton Raphson Iteration algorithm
5. SRT division algorithm

In the above list, 1, 2 and 5 could easily modeled in VHDL and synthasizable. 1, 2 gave the minimum area while 5 gave minimum delay. In this particular case, minimum area was preferred and hence 1, 2 were selected. Again, in these 1 could be modeled with minimum iterations. Finally, Non-restoring division algorithm was selected and modeled in VHDL. Now the whole VHDL model was synthesizable.

There was a problem due to insufficient memory during synthesis. To over come this the whole model was split into 27 blocks and simulated with ModelSim as described

in chapter 6. For simulation purposes, a top-level model is also developed in which all the 27 blocks are instantiated as components. The outputs obtained in the simulation of this top-level model were exactly same as those of the whole model. The VHDL models of all the 27 blocks and top-level model are listed in Appendix 2.

In the synthesis, the design was mapped to TSMC 0.35μ technology standard cells generated by CMC. A total of 2.6 million gates resulted after optimizing it with strict area and time constraints. The design was optimized with area as priority. The optimization was done with Physically Knowledgeable Synthesis option where the cells are physically placed for calculating the wire lengths and hence the parasitic capacitances.

5.2 Synthesis with Cadence Ambit Buildgates

In this section, synthesis of the VHDL model with Cadence Ambit Buildgates is described. The flow diagram is as below.

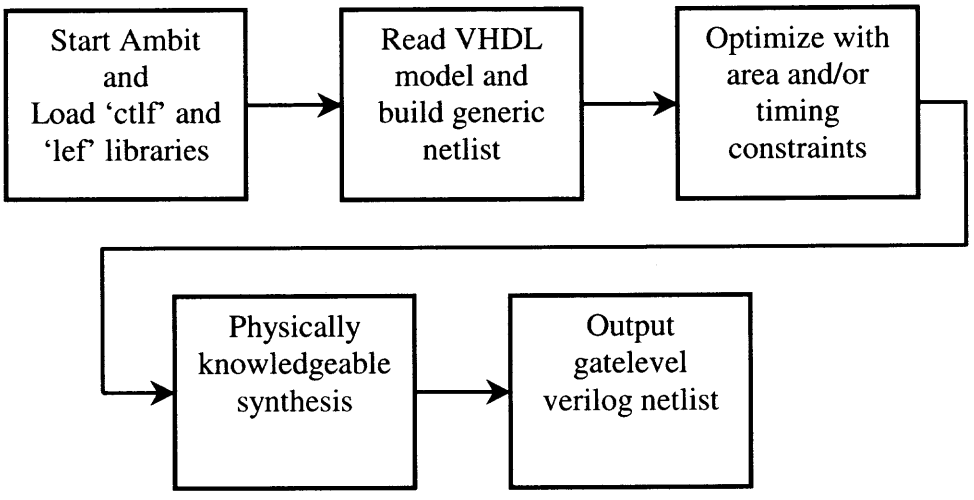

Figure 5.1 Cadence AmbitBuildGates synthesis flow.

1. Buildgates is started with *ac_shell –gui –pks &* making sure the executables are in the path.
2. The 'open' window is brought up with 'File -> Open' menu. In that, timing library option is selected to read the 'ctlf' file provided by standard cell library vendor. This file has capacitance, timing and functionality of the cells and wire load models for calculating the delay due to routing parasitics.
3. The 'lef' file is read by typing 'read_lef path/filename.lef' in the command window. This file contains the technology information of the technology used to develop the standard cells and also their abstract views. This is need only for PKS.
4. The VHDL file is read with 'open' window by selecting the VHDL option.
5. The VHDL model is mapped to generic gates with 'Commands -> Build Generic …' and selecting the options in the build generic window. The first and second options tell the tools to group the processes under subsection called process, third option tells the tools to group the subprograms under one section with the name of the subprogram.
6. The constraints are set by typing the following commands in the command window. The clock is necessary for timing optimization. The second command tells the tool that the input arrival time is 0; third one tells the tools that data required time is 10 ns. These two commands are the constraints to optimizer. The fourth command tells the tools to use the wire load model enclosed.

```
set_clock clock -period 2.0 -waveform {0.0 1.0}
set_input_delay 0.0 -clock clock [find -input *]
set_data_required_time 10 -clock clock [find -output *]
set_wire_load_mode enclosed
```

7. Optimization window is brought up with 'Command -> Optimize ...' menu. The 'Effort level' is set to high, 'Flatten mode' is set to off, 'Priority' is set to Area and in 'Options' minimize area is selected.
8. After step 7, a gate level design is produced in which optimization is done with the wire load models in the library. So the wire lengths calculated are approximate and the timing analysis done by the tools to added buffers is not accurate. For accurate results PKS is done by selecting the PKS option in the optimize window. In this, the cells are actually placed and the wire lengths are calculated. The placement is shown in figure 5.8

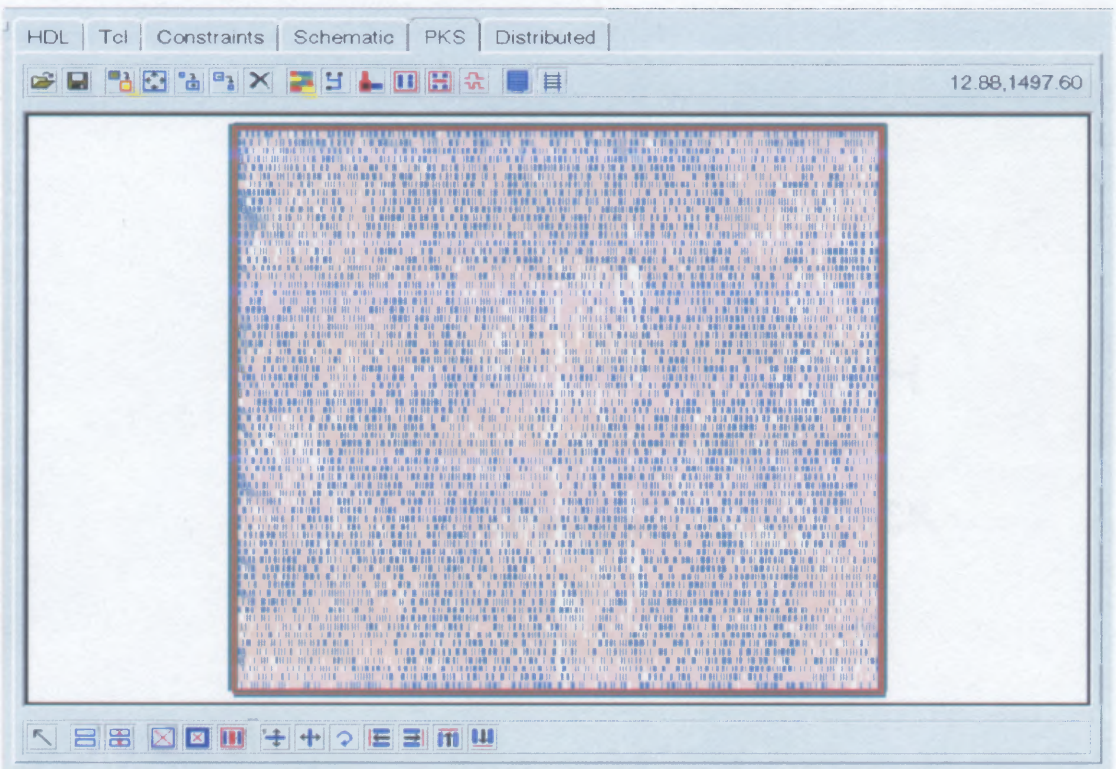

Figure 5.2 PKS window showing placement.

9. The synthesis is now complete and the design is saved as gate level verilog netlist and DEF format which has the placement information generated in PKS using

'save' window. If timing driven placement and routing is to be done a GCF file should also be produced which has the timing constraints and the path of 'ctlf' file.

5.3 Post Synthesis Simulation

In the synthesis process, gate level models of the design were produced and were simulated with MTI ModelSim for functionality as well as timing as described in chapter 6. For simulation purposes, again a top-level verilog module is developed in which the modules of the 27 blocks are instantiated. It is listed in Appendix 3. Again there was a problem of memory during the simulation of the top-level module. Therefore, the blocks were simulated individually in a pipelined way (output of one block is input to the next one with the produced delay). The final results are as below.

Figures 5.3 and 5.4 show the solutions for the data in Figure 4.1. The results were exactly same as those obtained before synthesis. It can be observed that the worst delay is 0.82μ sec.

Figures 5.5 and 5.6 show the solutions for the data in Figure 4.3. Again in this case also results were exactly same as those obtained before synthesis. It can be observed that the worst-case delay is 0.88μ seconds.

The delays observed here are just due to the gate delays. So in the worst case this delay will not be more than 1μ second. However, in the actual layout there will be a lot of parasitic capacitances due to routing metal layers. Due to size of the design, the actual delay may drastically go up because of these parasitic capacitances.

Figure 5.3 Post Synthesis Simulation Solution set 1 for real life situation 1.

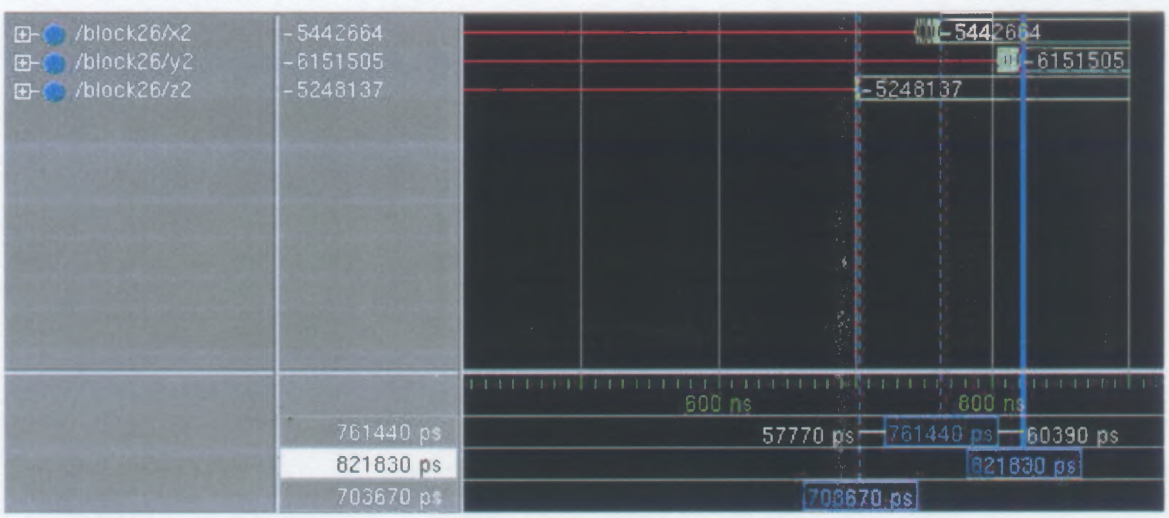

Figure 5.4 Post Synthesis Simulation Solution set 2 for real life situation 1.

Figure 5.5 Post Synthesis Simulation Solution set 1 for real life situation 2.

Figure 5.6 Post Synthesis Simulation Solution set 2 for real life situation 2.

CHAPTER 6

LAYOUT (P&R OF STANDARD CELLS)

The placement and routing of the individual blocks is done with Cadence Silicon Ensemble tool. This tool is capable of timing and power driven placement. Since only the functionality of the algorithm is needed on silicon, regular placement and routing is done to save time and computer memory. The layouts of the individual blocks are put together using the top-level verilog module in Appendix3 with the help of the same tool. The total die size without pads and parallel to serial conversion logic is 28mm X 28 mm. This could not be imported into the Cadence IC tools for DRC and Extraction due to insufficient computer memory.

6.1 P&R with Silicon Ensemble

In this section, placement and routing using Cadence Silicon Ensemble is described. The synthesized design is imported into the tool in gate or block level verilog format or DEF format. The standard cell library is imported into tool in LEF format. The output from the tool is LEF BLOCK, which is used at the higher level of hierarchy and DEF format for importing into Cadence IC tools for DRC and extraction. The flow with SE is shown in figure 6.1.

1. Silicon Ensemble is started with *seultra –m=96 &* making sure the executable is in the path. Here option m is the memory to be allocated to the tool.
2. The LEF file import dialog is brought up with 'File -> Import -> LEF' menu. After this step a database is created for storing the design and opening it later.

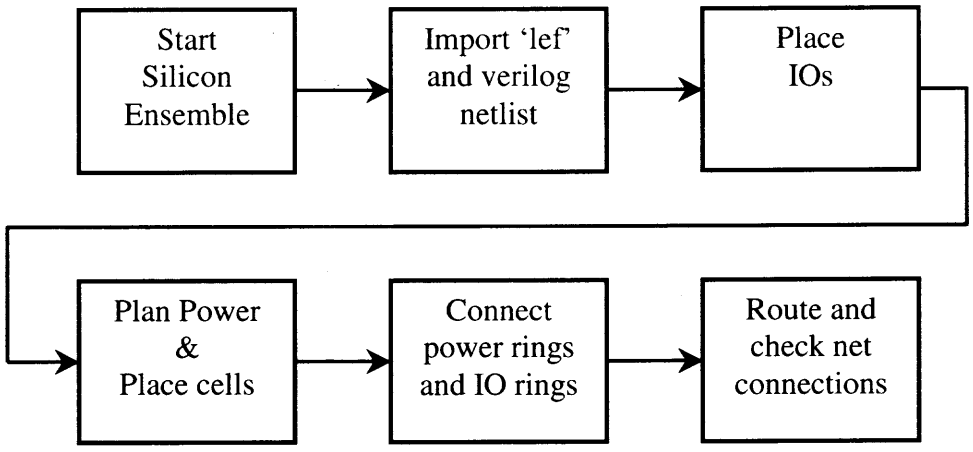

Figure 6.1 Placement & Routing flow with Silicon Ensemble.

3. The verilog netlist is imported with 'File -> Import -> Verilog' menu. Care should be taken that top module, power, ground nets are entered correctly. The reference libraries and the compiled verilog output libraries can default ones. Usually they will be loaded from the cds.lib file.
4. Floorplan dialog is brought up with 'Floorplan -> Initialize...' menu. The IO to core distances in the dialog are changed to 40 microns. This creates the rows for standard cell placement and 40 micron empty space around them. This space is used for VDD and VSS rings and IO connectivity. The floor plan is shown in figure 6.2.
5. IOs are placed with 'Place -> IO..' menu. In the displayed dialog, IO constraint file option is selected and the name of the file is entered. This is a DEF file, which has IO pads placement information. This is developed manually as per our requirements.

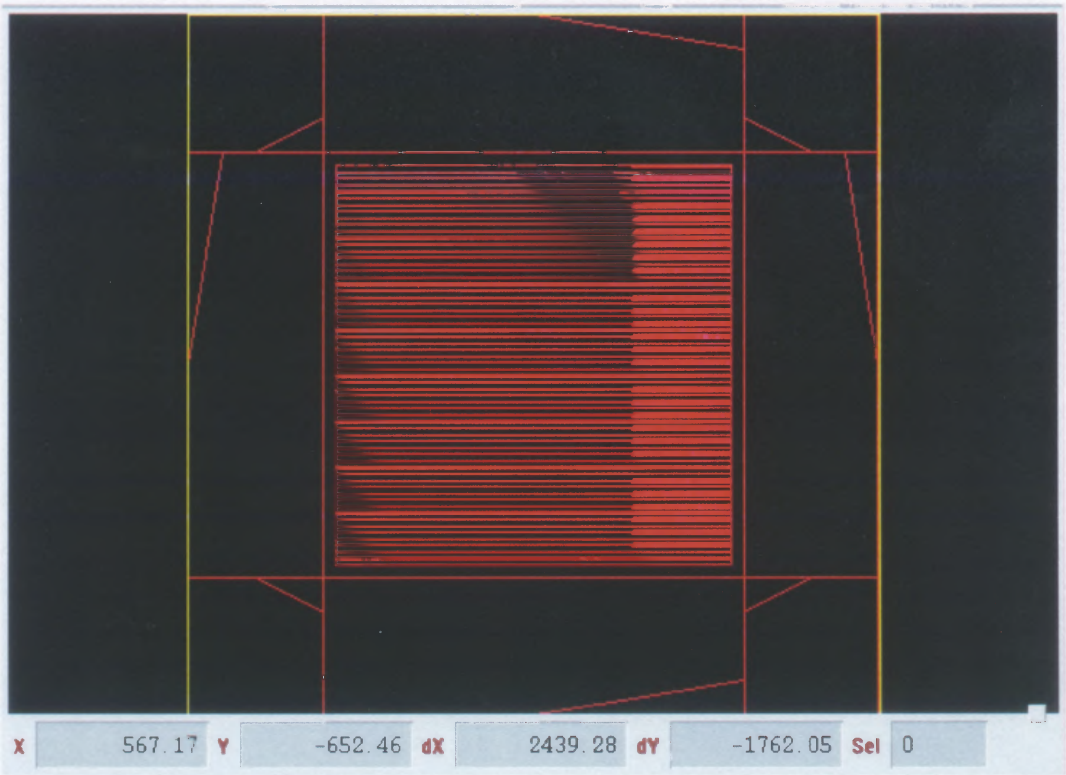

Figure 6.2 Floor plan in SE.

6. Power plan dialog is brought up with 'Route -> Plan Power' menu. VDD and VSS rings and stripes are added. Rings are the power paths that surround the core area and stripes are the power paths that pass over the core area.
7. Standard cells are placed with 'Place -> Cells' menu with 'Generate Congestion Map' option. The layout after placement of IO and cells is shown in figure 6.3.
8. All the VDD and VSS pins are connected to the rings and rings are connected to pads with 'Route -> Connect Ring' menu. Figure 6.4 shows how VDD and VSS rails of cells in the rows are connected. Figure 6.5 shows how these rails are connected to VDD VSS rings around the core area. Figure 6.6 shows how the 'VDDCORE', 'VDDRING', 'SUBESD', 'SUBCORE' and 'SUBRING' rings

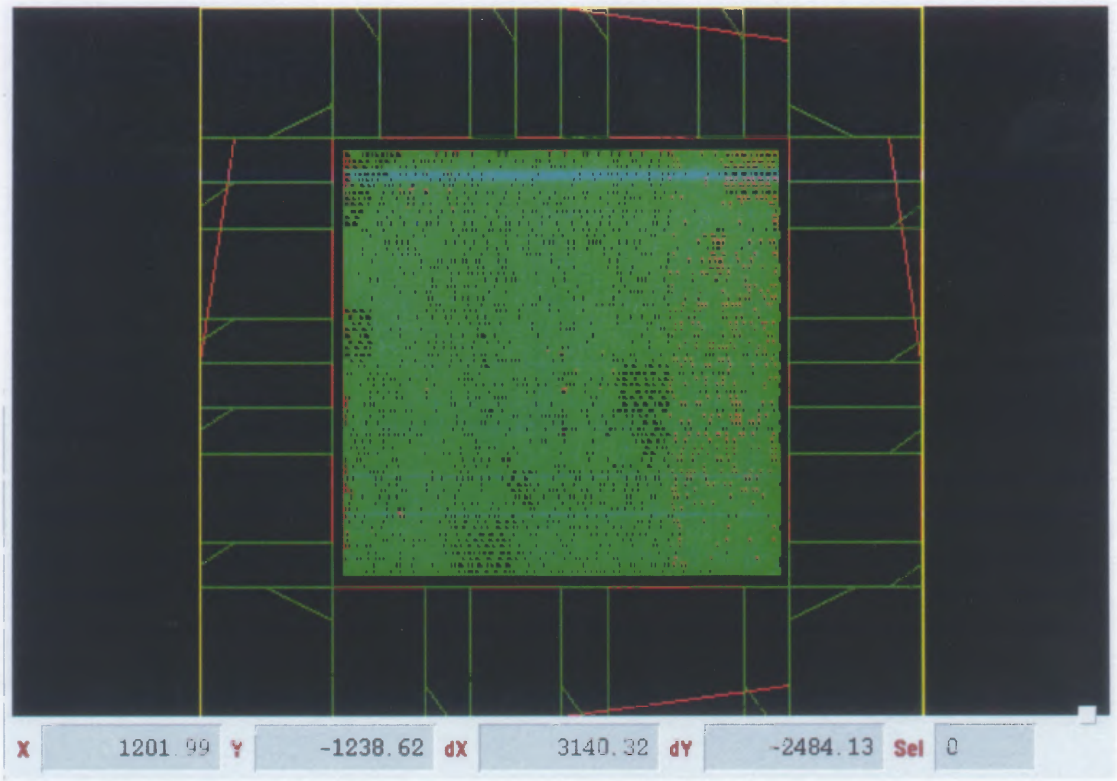

Figure 6.3 Layout after placement of IO pads and cells.

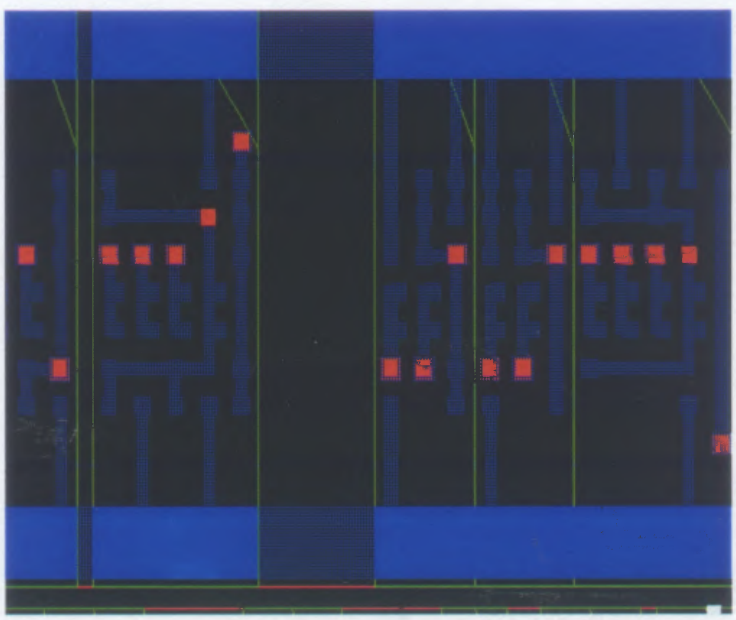

Figure 6.4 VDD and GND rail connectivity.

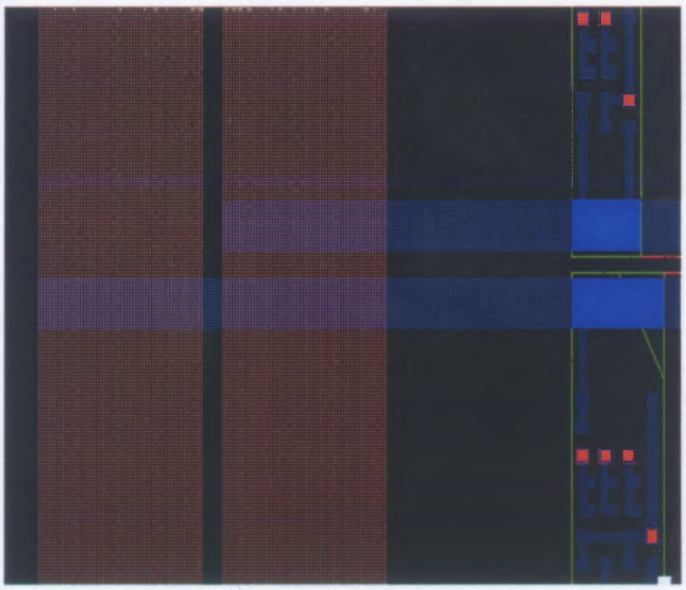

Figure 6.5 Power rails to power rings connection.

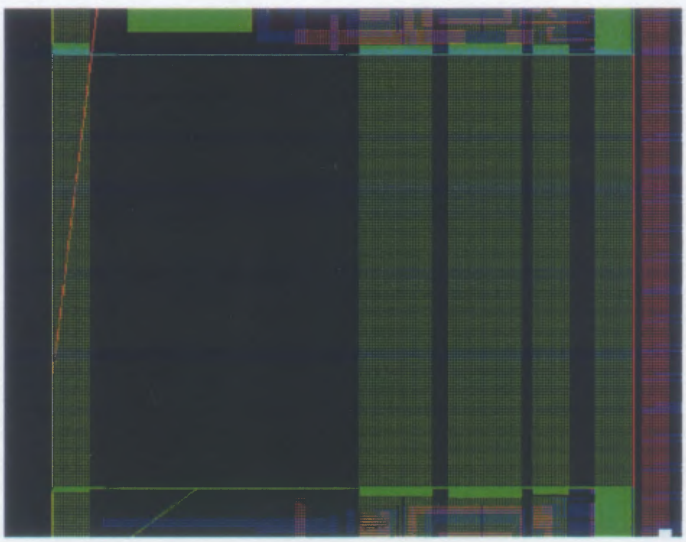

Figure 6.6 Pad rings connection.

between the pads are connected. Figure 6.7 shows how the VDD ring is connected to VDD pad with 20 micron connect.

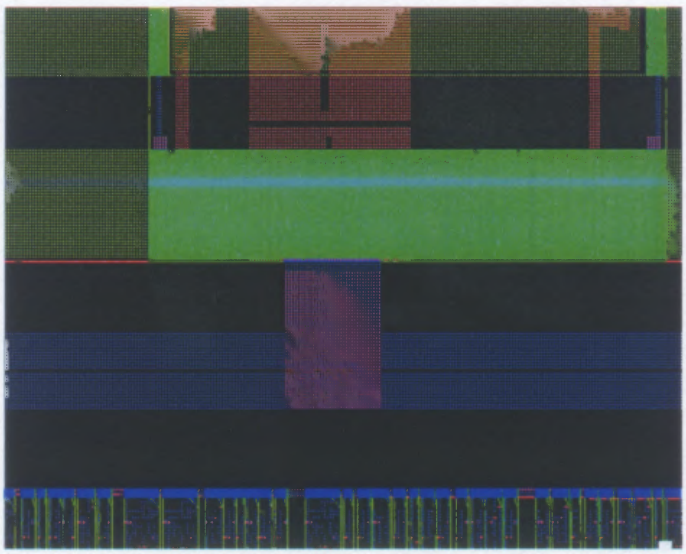

Figure 6.7 Pad to VDD ring connection.

9. Routing is done with 'Route -> Wroute' menu as shown in fig 6.11. With this command Wrap Router is invoked and global as well as final routing is done in one shot. Some of the connections to the cells are shown in Fig 6.12 and to a pad is shown in Fig 6.13.
10. The design is saved in LEF BLOCK and DEF formats with 'File -> Export' menu. Also the database is saved with 'File -> Save' menu. This is useful for viewing or modifying the layout at any time. Then Exit the tool with 'File -> Exit' menu.

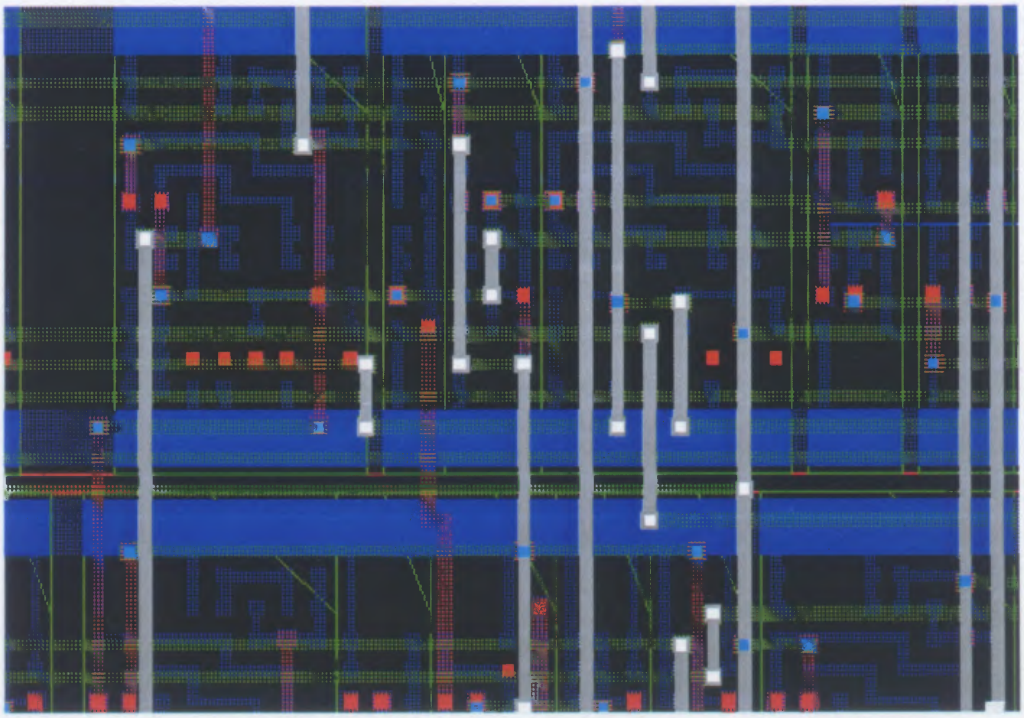

Figure 6.8 A view of cell connections.

6.2 DRC and LPE with Cadence IC

Cadence IC tools are used for DRC and Extraction of parasitics. Also the GDSII for tape out can be produced only from this tool set. A step-by-step procedure for working with the imported design from Silicon Ensemble is described here. The flow diagram is as below.

1. ICFB is invoked with '*icfb &*' command .
2. In the IC environment, a library is created for the design with 'File -> New -> Library' menu in the CIW window.
3. The layout developed in the SE is imported into IC in DEF format with 'File -> Import -> DEF' menu. In the dialog, the library name is the name of the library created and view name is autoLayout.

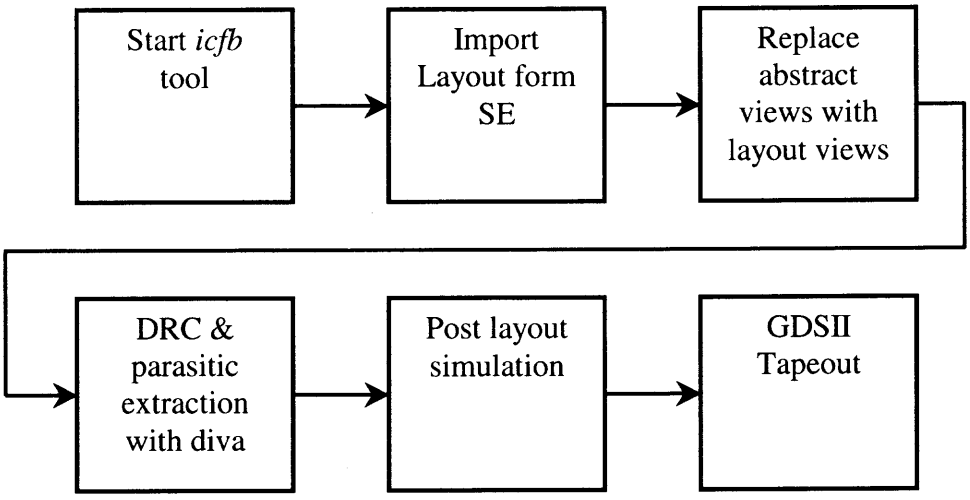

Figure 6.9 Cadence IC design Flow.

4. Imported design will use the 'abstract' view for all cells. All 'abstract' views are to be changed to 'layout' views. First, 'autoLayout' view is opened, then with 'Tools – >Layout' menu layout tool is invoked.
5. Using 'Edit –> Search' command search menu is brought up. All instances with view name = 'abstract' are searched and replaced with view name = 'layout'. The modified design is saved as 'layout' view.
6. With 'Verify -> DRC...' menu DRC dialog is brought up and in the library box, name is changed to the name of the technology library and "OK" button is clicked. This will check for DRC and report the errors with flashing rectangles. These are to be corrected manually. Usually if the LEF file imported into SE is perfect in technology point of view then there will be no DRC errors.

7. With 'Verify -> Extract...' menu Extract dialog is brought up. In the technology box, the default one is changed to the technology used and click "OK" button. This produces an Extracted view.
8. Now the Extracted View is opened in Virtuoso and Simulation environment is started with 'Tools -> Simulation -> other' menu.
9. The tool is changed to HSPICE with 'Simulation -> Initialize...' menu.
10. The netlist is generated with 'Simulation -> Netlist/Simulate...' menu and deselecting Simulate option.

6.3 Post Layout Simulations

The Extracted view of the design is used to generate the HSPICE netlist in the Vistuoso layout tool. This netlist is then simulated with the test inputs using HSPICE in the following order.

1. Standard cells
2. Individual blocks
3. Whole chip

The standard cell characterizations were already done by the vendor and hence it was not done again. For the Individual blocks, due to the limit in transistor count in HSPICE the simulation could be carried out. These simulations could be performed if there were tools like Star-Sim and Star-time which could handle about 50 million transistors and 10000 times faster than HSPICE.

CHAPTER 7

CONCLUSIONS

This thesis provides a position location estimation algorithm, which can be used for position fixes in GPS as well as cellular services. The implementation in VHDL was a straightforward approach in which the equations were realized as concurrent statements. The VHDL model could be synthesized and implemented on silicon. The total number of gates produced in synthesis was 2.7 million, which is very huge. Most of the gates were due to the multiplication and division operations. Despite using well-optimized algorithms for these operations huge gate count was unavoidable. The parallel implementation approach in the multiplication algorithm and use of 'for' loops in the division and square root operations, added to the gate count. The standard cell library was also one of the reasons for the huge gate count. The library has only the basic gates and lacked complex gates. If complex gates were there, the gate count in the design, the silicon area and obviously the device count would have considerably decreased. Another reason for huge gate count was, the numerators in the division operations were multiplied by 10000 to preserve the accuracy. The gate count could be considerably decreased by:

1. Using serial multipliers for multiplication operations.
2. Using state machine or feedback architecture in the division and square root algorithms
3. Using standard cell libraries which have complex gates, well optimized for area and timing. (The library used here was very well optimized for only for timing, but not area.)

4. Using operator merging during the synthesis. For this the whole VHDL model should be synthesized with out breaking it into blocks, which requires a large memory and time.

The gate level verilog simulation of the whole design and the device level simulations could not be carried out because of insufficient computing power and lack of high capacity device level simulators respectively. The device level simulations could have been carried out with the existing computing power if the high capacity simulators like Star-Sim and Star-Time were there. This would have given us the confidence that design would definitely work on silicon. To make sure that all the connections are made, connectivity checking is done in Silicon Ensemble.

APPENDIX A

VHDL CODE

This appendix is the listing of the VHDL model for the Hyperbolic Position Location Algorithm and the test benches.

VHDL Model for the Hyperbolic Position Location Algorithm

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity hyperbolic is
port(xi,xj,xk,xl,yi,yj,yk,yl,zi,zj,zk,zl,ti,tk,tj,tl: in SIGNED(31
downto 0);
     x1,x2,y1,y2,z1,z2: out SIGNED(31 downto 0));
end hyperbolic;

architecture behave of hyperbolic is
signal o: SIGNED(199 downto 0);
signal n: SIGNED(195 downto 0);
signal m: SIGNED(191 downto 0);
signal c,f,l: SIGNED(95 downto 0);
signal s12,s16,s21,s22,s24: SIGNED(131 downto 0);
signal s9,s11,s13,s15,s23,k: SIGNED(99 downto 0);
signal s1,s2,s3,s4,s5,s6,s7,s8,s10,s14,s17,s18,s19,s20,s26,s27,s29,s30,
 a,b,d,e,g,h,i,j,yi2,yj2,yk2,yl2,xi2,xj2,xk2,xl2,zi2,zj2,zk2,zl2,
 rij2,rik2,rkl2,rkj2: SIGNED(63 downto 0);
signal rij,rik,rkj,rkl,xji,xki,xjk,xlk,yji,yki,yjk,ylk,zji,zki,zjk,zlk,
 light,thou,root,s28,s31,s32,s33,s34,s35,s36: SIGNED(31 downto 0);
signal one_e_10: SIGNED(35 downto
0):="001001010100000010111110010000000000";
begin
light<=to_signed(333564,32); thou<=to_signed(100000,32);
s1<=abs(thou*(ti-tj));
s2<=abs(thou*(ti-tk));
s3<=abs(thou*(tk-tl));
s4<=abs(thou*(tk-tj));
s5<=s1/light;
s6<=s2/light;
s7<=s3/light;
s8<=s4/light;
rij<=resize(s5,32);
rik<=resize(s6,32);
rkl<=resize(s7,32);
rkj<=resize(s8,32);
rij2<=rij*rij;
rik2<=rik*rik;
```

```
rkl2<=rkl*rkl;
rkj2<=rkj*rkj;

xji<=xj-xi;
yji<=yj-yi;
zji<=zj-zi;
yi2<=yi*yi;
xi2<=xi*xi;
zi2<=zi*zi;
xki<=xk-xi;
yki<=yk-yi;
zki<=zk-zi;
yj2<=yj*yj;
xj2<=xj*xj;
zj2<=zj*zj;
xlk<=xl-xk;
ylk<=yl-yk;
zlk<=zl-zk;
yk2<=yk*yk;
xk2<=xk*xk;
zk2<=zk*zk;
xjk<=xj-xk;
yjk<=yj-yk;
zjk<=zj-zk;
yl2<=yl*yl;
xl2<=xl*xl;
zl2<=zl*zl;

s9 <=one_e_10*(rik*xji-rij*xki);
s10<=rij*yki-rik*yji;
s11<=one_e_10*(rik*zji-rij*zki);
s13<=one_e_10*(rkl*xjk-rkj*xlk);
s14<=rkj*ylk-rkl*yjk;
s15<=one_e_10*(rkl*zjk-rkj*zlk);

s17<=rij2+xi2-xj2+yi2-yj2+zi2-zj2;
s18<=rik2+xi2-xk2+yi2-yk2+zi2-zk2;
s19<=rkj2+xk2-xj2+yk2-yj2+zk2-zj2;
s20<=rkl2+xk2-xl2+yk2-yl2+zk2-zl2;
s12<=one_e_10*(rik*s17-rij*s18);
s16<=one_e_10*(rkl*s19-rkj*s20);
s21<=SHIFT_RIGHT(s12,1);
s22<=SHIFT_RIGHT(s16,1);

a<=resize(s9/s10,64);
b<=resize(s11/s10,64);
c<=resize(s21/s10,96);
d<=resize(s13/s14,64);
e<=resize(s15/s14,64);
f<=resize(s22/s14,96);

s23<=one_e_10*(e-b);
s24<=one_e_10*(f-c);

g<=resize(s23/(a-d),64);
h<=resize(s24/(a-d),64);
i<=resize(((a*g)/one_e_10)+b,64);
```

```
j<=resize(((a*h)/one_e_10)+c,64);
k<=s18*one_e_10+SHIFT_LEFT(j*yki,1)+SHIFT_LEFT(h*xki,1);
l<=SHIFT_LEFT(g*xki+i*yki+zki*one_e_10,1);
m<=SHIFT_LEFT(rik2*(g*g+i*i+one_e_10*one_e_10),2)-l*l;

s26<=resize(one_e_10*xi-h,64);
s27<=resize(one_e_10*yi-j,64);

n<=SHIFT_LEFT(rik2*(g*s26+i*s27+zi*one_e_10*one_e_10),3)+SHIFT_LEFT(l*k
    ,1);
o<=SHIFT_LEFT(rik2*(s26*s26+s27*s27+zi*zi*one_e_10*one_e_10),2)-k*k;

s28<=resize(SHIFT_RIGHT(n/m,1),32);
s29<=resize(o/m,64);
s30<=s28*s28-S29;

squareroot:process (s30)
      variable q,r,s,t:signed(63 downto 0);
      begin
      q:=s30;
      s:=to_signed(0,64);
      r:=to_signed(0,64);
      t:="0100000000000000000000000000000000000000000000000000000000000000
          000";
      for j in 1 to 32 loop
      s:=r+t;
      if s<=q then
            q:=q-s;
            r:=s+t;
      end if;
      r:=shift_right(r,1);
      t:=shift_right(t,2);
      end loop;
root<=resize(r,32);
end process squareroot;

s31<=S28+root;
s32<=resize((g*s31+h)/one_e_10,32);
s33<=resize((a*s32+b*s31+c)/one_e_10,32);
s34<=s28-root;
s35<=resize((g*s34+h)/one_e_10,32);
s36<=resize((a*s35+b*s34+c)/one_e_10,32);
z1<=s31;
x1<=s32;
y1<=s33;
z2<=s34;
x2<=s35;
y2<=S36;
end behave;
```

Test Benches

```
--TEST CASE 1

library ieee;
use ieee.std_logic_1164.all;
```

```
use ieee.numeric_std.all;

entity test1 is
end test1;

architecture stimulus of test1 is

component hyperbolic
port(xi,xj,xk,xl,yi,yj,yk,yl,zi,zj,zk,zl,ti,tk,tj,tl:in SIGNED(31
downto 0);
     x1,x2,y1,y2,z1,z2: out SIGNED(31 downto 0));
end component;

signal ti:SIGNED(31 downto 0):=to_signed(67335898,32);
signal tj:SIGNED(31 downto 0):=to_signed(78283279,32);
signal tk:SIGNED(31 downto 0):=to_signed(86023981,32);
signal tl:SIGNED(31 downto 0):=to_signed(75092320,32);

signal xi:SIGNED(31 downto 0):=to_signed(0,32);
signal xj:SIGNED(31 downto 0):="11111111000101011111010010010011";-- (-
15338349)
signal xk:SIGNED(31 downto 0):=to_signed(0,32);
signal xl:SIGNED(31 downto 0):="11111101111000010101101011100100";-- (-
18785564)

signal yi:SIGNED(31 downto 0):=to_signed(26566800,32);
signal yj:SIGNED(31 downto 0):=to_signed(15338349,32);
signal yk:SIGNED(31 downto 0):=to_signed(6380000,32);
signal yl:SIGNED(31 downto 0):=to_signed(18785564,32);

signal zi:SIGNED(31 downto 0):=to_signed(0,32);
signal zj:SIGNED(31 downto 0):=to_signed(15338349,32);
signal zk:SIGNED(31 downto 0):=to_signed(25789348,32);
signal zl:SIGNED(31 downto 0):=to_signed(0,32);

signal x1,x2,y1,y2,z1,z2:SIGNED(31 downto 0):=to_signed(0,32);

signal int_x1,int_x2,int_y1,int_y2,int_z1,int_z2:integer:=0;

begin

s1: hyperbolic port map (xi => xi, xj => xj, xk => xk, xl=>xl,
                  yi => yi, yj => yj, yk => yk, yl=>yl,
                  zi => zi, zj => zj, zk => zk, zl=>zl,
                  ti => ti, tj => tj, tk => tk, tl=>tl,
                  z1 => z1, z2 => z2, x1 => x1,
                        x2 => x2, y1 => y1, y2 => y2);

int_x1<=to_integer(x1); int_x2<=to_integer(x2);
int_y1<=to_integer(y1); int_y2<=to_integer(y2);
int_z1<=to_integer(z1); int_z2<=to_integer(z2);
end stimulus;


--TEST CASE 2
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity test2 is
end test2;

architecture stimulus of test2 is

component hyperbolic
port(xi,xj,xk,xl,yi,yj,yk,yl,zi,zj,zk,zl,ti,tk,tj,tl: in SIGNED(31
downto 0);
     x1,x2,y1,y2,z1,z2: out SIGNED(31 downto 0));
end component;

signal ti:SIGNED(31 downto 0):=to_signed(86320708,32);
signal tj:SIGNED(31 downto 0):=to_signed(75293013,32);
signal tk:SIGNED(31 downto 0):=to_signed(67335895,32);
signal tl:SIGNED(31 downto 0):=to_signed(78283279,32);

signal xi:SIGNED(31 downto 0):="11111111000110011000111110001011";-- (-
15102069)
signal xj:SIGNED(31 downto 0):=to_signed(0,32);
signal xk:SIGNED(31 downto 0):=to_signed(15338349,32);
signal xl:SIGNED(31 downto 0):=to_signed(26566800,32);

signal yi:SIGNED(31 downto 0):=to_signed(21482069,32);
signal yj:SIGNED(31 downto 0):=to_signed(6380000,32);
signal yk:SIGNED(31 downto 0):=to_signed(15338349,32);
signal yl:SIGNED(31 downto 0):=to_signed(0,32);

signal zi:SIGNED(31 downto 0):=to_signed(3683495,32);
signal zj:SIGNED(31 downto 0):=to_signed(25789348,32);
signal zk:SIGNED(31 downto 0):=to_signed(15338349,32);
signal zl:SIGNED(31 downto 0):=to_signed(0,32);

signal x1,x2,y1,y2,z1,z2:SIGNED(31 downto 0):=to_signed(0,32);

signal int_x1,int_x2,int_y1,int_y2,int_z1,int_z2:integer:=0;

begin

s1: hyperbolic port map (xi => xi, xj => xj, xk => xk, xl=>xl,
                  yi => yi, yj => yj, yk => yk, yl=>yl,
                  zi => zi, zj => zj, zk => zk, zl=>zl,
                  ti => ti, tj => tj, tk => tk, tl=>tl,
                  z1 => z1, z2 => z2, x1 => x1,
                        x2 => x2, y1 => y1, y2 => y2);

int_x1<=to_integer(x1); int_x2<=to_integer(x2);
int_y1<=to_integer(y1); int_y2<=to_integer(y2);
int_z1<=to_integer(z1); int_z2<=to_integer(z2);
end stimulus;
```

APPENDIX B

BLOCK LEVEL VHDL CODE

This appendix is the listing of the hierarchical VHDL model for the Hyperbolic Position Location Algorithm.

VHDL Models of Different Blocks

```
---Block1---
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block1 IS
  PORT(ti, tj, tk, tl : IN SIGNED(31 DOWNTO 0);
       s1, s2, s3, s4 : OUT SIGNED(63 DOWNTO 0));
END ENTITY block1;
ARCHITECTURE behave OF block1 IS
  CONSTANT thou : SIGNED(31 DOWNTO 0) :=
"00000000000000011000011010100000";
BEGIN
  s1<=abs(thou*(ti-tj));
  s2<=abs(thou*(ti-tk));
  s3<=abs(thou*(tk-tl));
  s4<=abs(thou*(tk-tj));
END ARCHITECTURE behave;

---Block2---
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
ENTITY block2 IS
  PORT(s1,s2,s3,s4 : IN SIGNED(63 DOWNTO 0);
       rij,rik,rkl,rkj : OUT SIGNED(31 DOWNTO 0);
       rij2,rik2,rkl2,rkj2 : OUT SIGNED(63 DOWNTO 0));
END ENTITY block2;
ARCHITECTURE behave OF block2 IS
  CONSTANT light : SIGNED(31 DOWNTO 0) :=
"00000000000001010001011011111100";
  SIGNAL rij1,rik1,rkl1,rkj1 : SIGNED(31 DOWNTO 0);
  FUNCTION DIV (dd, dv : SIGNED) RETURN SIGNED IS
    VARIABLE rm : SIGNED(dd'RANGE);
    VARIABLE dv1 : SIGNED(dd'RANGE);
    VARIABLE dv_neg1 : SIGNED(dd'RANGE);
    VARIABLE trl_zeros : SIGNED((dd'LENGTH - dv'LENGTH - 1) DOWNTO 0);
    VARIABLE q1 : SIGNED((dd'LENGTH - dv'LENGTH)  DOWNTO 0);
    VARIABLE one : SIGNED((dd'LENGTH-1) DOWNTO 0);
    VARIABLE corr : UNSIGNED (2 DOWNTO 0);
```

```
    VARIABLE add_sub : SIGNED(dd'RANGE);
  BEGIN
    trl_zeros := (OTHERS => '0');
    one := (OTHERS => '0');
    one(0) := one(0) OR '1';
    q1 := (OTHERS => '0');
    rm := dd;
    dv1 := dv & trl_zeros;
    dv_neg1 := (NOT dv1) + one;
    iteration : FOR i IN 0 TO (dd'LENGTH - dv'LENGTH - 1) LOOP
      rm := SHIFT_LEFT(rm, 1);
      q1 := SHIFT_LEFT(q1, 1);
      q1(0) := NOT (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1));
      add_sub := (OTHERS => (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1)) );
      rm := rm +( (dv1 AND add_sub) OR (dv_neg1 AND (NOT add_sub)) );
    END LOOP iteration;
    q1 := SHIFT_LEFT(q1, 1);
    q1(dd'LENGTH - dv'LENGTH) := q1(dd'LENGTH - dv'LENGTH) XOR '1';
    corr := dd(dd'LENGTH - 1) & dv(dv'LENGTH - 1) & rm(dd'LENGTH - 1);
    IF corr = "000" OR corr = "111"  THEN
      q1(0) := '1';
    ELSIF corr = "001" OR corr = "110" THEN
      q1(0) := '0';
    ELSIF corr = "010" OR corr = "101" THEN
      q1(0) := '0';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    ELSIF corr = "011" OR corr = "100" THEN
      q1(0) := '1';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    END IF;
    RETURN q1;
  END FUNCTION DIV;
BEGIN
  rij1<=resize(DIV(s1,light), 32);rij<=rij1;
  rik1<=resize(DIV(s2,light), 32);rik<=rik1;
  rkl1<=resize(DIV(s3,light), 32);rkl<=rkl1;
  rkj1<=resize(DIV(s4,light), 32);rkj<=rkj1;
  rij2<=rij1*rij1;
  rik2<=rik1*rik1;
  rkl2<=rkl1*rkl1;
  rkj2<=rkj1*rkj1;
END ARCHITECTURE behave;

---Block3---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block3 IS
  PORT(xi,xj,xk,yi,yj,yk,zi,zj,zk : IN SIGNED(31 DOWNTO 0);
       xji,yji,zji,xki,yki,zki : OUT SIGNED(31 DOWNTO 0);
       xi2,yi2,zi2,xj2,yj2,zj2 : OUT SIGNED(63 DOWNTO 0));
END ENTITY block3;

ARCHITECTURE behave OF block3 IS
BEGIN
```

```
  xji<=xj-xi;
  yji<=yj-yi;
  zji<=zj-zi;
  yi2<=yi*yi;
  xi2<=xi*xi;
  zi2<=zi*zi;
  xki<=xk-xi;
  yki<=yk-yi;
  zki<=zk-zi;
  yj2<=yj*yj;
  xj2<=xj*xj;
  zj2<=zj*zj;
END ARCHITECTURE behave;

---Block4---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block4 IS
  PORT(xj,yj,zj,xk,yk,zk,xl,yl,zl : IN SIGNED(31 DOWNTO 0);
       xlk,ylk,zlk,xjk,yjk,zjk : OUT SIGNED(31 DOWNTO 0);
       xk2,yk2,zk2,xl2,yl2,zl2 : OUT SIGNED(63 DOWNTO 0));
END ENTITY block4;

ARCHITECTURE behave OF block4 IS
BEGIN
  xlk<=xl-xk;
  ylk<=yl-yk;
  zlk<=zl-zk;
  yk2<=yk*yk;
  xk2<=xk*xk;
  zk2<=zk*zk;
  xjk<=xj-xk;
  yjk<=yj-yk;
  zjk<=zj-zk;
  yl2<=yl*yl;
  xl2<=xl*xl;
  zl2<=zl*zl;
END ARCHITECTURE behave;

---Block5---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block5 IS
  PORT(rij,rik,xji,xki,yji,yki : IN SIGNED(31 DOWNTO 0);
       s9 : OUT SIGNED(99 DOWNTO 0);
       s10 : OUT SIGNED(63 DOWNTO 0));
END ENTITY block5;

ARCHITECTURE behave OF block5 IS
  CONSTANT one_e_10: SIGNED(35 downto 0) :=
"001001010100000010111110010000000000";
```

```
BEGIN
  s9 <=one_e_10*(rik*xji-rij*xki);
  s10<=rij*yki-rik*yji;
END ARCHITECTURE behave;

---Block6---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block6 IS
  PORT(rij,rik,rkl,rkj,xjk,xlk,zji,zki : IN SIGNED(31 DOWNTO 0);
       s11,s13 : OUT SIGNED(99 DOWNTO 0));
END ENTITY block6;

ARCHITECTURE behave OF block6 IS
  CONSTANT one_e_10: SIGNED(35 downto 0) :=
"001001010100000010111110010000000000";
BEGIN
  s11<=one_e_10*(rik*zji-rij*zki);
  s13<=one_e_10*(rkl*xjk-rkj*xlk);
END ARCHITECTURE behave;

---Block7---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block7 IS
  PORT(rkj,rkl,yjk,ylk,zjk,zlk : IN SIGNED(31 DOWNTO 0);
       s14 : OUT SIGNED(63 DOWNTO 0);
       s15 : OUT SIGNEd(99 DOWNTO 0));
END ENTITY block7;

ARCHITECTURE behave OF block7 IS
  CONSTANT one_e_10: SIGNED(35 downto 0) :=
"001001010100000010111110010000000000";
BEGIN
  s14<=rkj*ylk-rkl*yjk;
  s15<=one_e_10*(rkl*zjk-rkj*zlk);
END ARCHITECTURE behave;

---Block8---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block8 IS

PORT(yi2,yj2,yk2,yl2,xi2,xj2,xk2,xl2,zi2,zj2,zk2,zl2,rij2,rik2,rkl2,rkj
2: IN SIGNED(63 downto 0);
       s17,s18,s19,s20 : OUT SIGNED(63 DOWNTO 0));
END ENTITY block8;
```

```
ARCHITECTURE behave OF block8 IS
BEGIN
  s17<=rij2+xi2-xj2+yi2-yj2+zi2-zj2;
  s18<=rik2+xi2-xk2+yi2-yk2+zi2-zk2;
  s19<=rkj2+xk2-xj2+yk2-yj2+zk2-zj2;
  s20<=rkl2+xk2-xl2+yk2-yl2+zk2-zl2;
END ARCHITECTURE behave;

---Block9---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block9 IS
  PORT(rij,rik,rkj,rkl : IN SIGNED(31 DOWNTO 0);
       s17,s18,s19,s20 : IN SIGNED(63 DOWNTO 0);
       s21,s22 : OUT SIGNED(131 DOWNTO 0));
END ENTITY block9;

ARCHITECTURE behave OF block9 IS
  CONSTANT one_e_10: SIGNED(35 downto 0) :=
"001001010100000010111110010000000000";
  SIGNAL s12,s16 : SIGNED(131 DOWNTO 0);
BEGIN
  s12<=one_e_10*(rik*s17-rij*s18);
  s16<=one_e_10*(rkl*s19-rkj*s20);
  s21<=SHIFT_RIGHT(s12,1);
  s22<=SHIFT_RIGHT(s16,1);
END ARCHITECTURE behave;

---Block10---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block10 IS
  PORT(s9,s11 : IN SIGNED(99 DOWNTO 0);
       s10 : IN SIGNED(63 DOWNTO 0);
       a,b : OUT SIGNED(63 DOWNTO 0));
END ENTITY block10;

ARCHITECTURE behave OF block10 IS
  FUNCTION DIV (dd, dv : SIGNED) RETURN SIGNED IS
    VARIABLE rm : SIGNED(dd'RANGE);
    VARIABLE dv1 : SIGNED(dd'RANGE);
    VARIABLE dv_neg1 : SIGNED(dd'RANGE);
    VARIABLE trl_zeros : SIGNED((dd'LENGTH - dv'LENGTH - 1) DOWNTO 0);
    VARIABLE q1 : SIGNED((dd'LENGTH - dv'LENGTH)  DOWNTO 0);
    VARIABLE one : SIGNED((dd'LENGTH-1) DOWNTO 0);
    VARIABLE corr : UNSIGNED (2 DOWNTO 0);
    VARIABLE add_sub : SIGNED(dd'RANGE);
  BEGIN
    trl_zeros := (OTHERS => '0');
    one := (OTHERS => '0');
    one(0) := one(0) OR '1';
```

```
    q1 := (OTHERS => '0');
    rm := dd;
    dv1 := dv & trl_zeros;
    dv_neg1 := (NOT dv1) + one;
    iteration : FOR i IN 0 TO (dd'LENGTH - dv'LENGTH - 1) LOOP
      rm := SHIFT_LEFT(rm, 1);
      q1 := SHIFT_LEFT(q1, 1);
      q1(0) := NOT (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1));
      add_sub := (OTHERS => (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1)) );
      rm := rm +( (dv1 AND add_sub) OR (dv_neg1 AND (NOT add_sub)) );
    END LOOP iteration;
    q1 := SHIFT_LEFT(q1, 1);
    q1(dd'LENGTH - dv'LENGTH) := q1(dd'LENGTH - dv'LENGTH) XOR '1';
    corr := dd(dd'LENGTH - 1) & dv(dv'LENGTH - 1) & rm(dd'LENGTH - 1);
    IF corr = "000" OR corr = "111"  THEN
      q1(0) := '1';
    ELSIF corr = "001" OR corr = "110" THEN
      q1(0) := '0';
    ELSIF corr = "010" OR corr = "101" THEN
      q1(0) := '0';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    ELSIF corr = "011" OR corr = "100" THEN
      q1(0) := '1';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    END IF;
    RETURN q1;
  END FUNCTION DIV;
BEGIN
  a<=resize(DIV(resize(s9, 128),s10),64);
  b<=resize(DIV(resize(s11, 128),s10),64);
END ARCHITECTURE behave;

---Block10---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block10 IS
  PORT(s9,s11 : IN SIGNED(99 DOWNTO 0);
       s10 : IN SIGNED(63 DOWNTO 0);
       a,b : OUT SIGNED(63 DOWNTO 0));
END ENTITY block10;

ARCHITECTURE behave OF block10 IS
  FUNCTION DIV (dd, dv : SIGNED) RETURN SIGNED IS
    VARIABLE rm : SIGNED(dd'RANGE);
    VARIABLE dv1 : SIGNED(dd'RANGE);
    VARIABLE dv_neg1 : SIGNED(dd'RANGE);
    VARIABLE trl_zeros : SIGNED((dd'LENGTH - dv'LENGTH - 1) DOWNTO 0);
    VARIABLE q1 : SIGNED((dd'LENGTH - dv'LENGTH)  DOWNTO 0);
    VARIABLE one : SIGNED((dd'LENGTH-1) DOWNTO 0);
    VARIABLE corr : UNSIGNED (2 DOWNTO 0);
    VARIABLE add_sub : SIGNED(dd'RANGE);
  BEGIN
    trl_zeros := (OTHERS => '0');
    one := (OTHERS => '0');
```

```
    one(0) := one(0) OR '1';
    q1 := (OTHERS => '0');
    rm := dd;
    dv1 := dv & trl_zeros;
    dv_neg1 := (NOT dv1) + one;
    iteration : FOR i IN 0 TO (dd'LENGTH - dv'LENGTH - 1) LOOP
      rm := SHIFT_LEFT(rm, 1);
      q1 := SHIFT_LEFT(q1, 1);
      q1(0) := NOT (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1));
      add_sub := (OTHERS => (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1)) );
      rm := rm +( (dv1 AND add_sub) OR (dv_neg1 AND (NOT add_sub)) );
    END LOOP iteration;
    q1 := SHIFT_LEFT(q1, 1);
    q1(dd'LENGTH - dv'LENGTH) := q1(dd'LENGTH - dv'LENGTH) XOR '1';
    corr := dd(dd'LENGTH - 1) & dv(dv'LENGTH - 1) & rm(dd'LENGTH - 1);
    IF corr = "000" OR corr = "111"  THEN
      q1(0) := '1';
    ELSIF corr = "001" OR corr = "110" THEN
      q1(0) := '0';
    ELSIF corr = "010" OR corr = "101" THEN
      q1(0) := '0';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    ELSIF corr = "011" OR corr = "100" THEN
      q1(0) := '1';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    END IF;
    RETURN q1;
  END FUNCTION DIV;
BEGIN
  a<=resize(DIV(resize(s9, 128),s10),64);
  b<=resize(DIV(resize(s11, 128),s10),64);
END ARCHITECTURE behave;


---Block12---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block12 IS
  PORT(s22 : IN SIGNED(131 DOWNTO 0);
       s15 : IN SIGNED(99 DOWNTO 0);
       s14 : IN SIGNED(63 DOWNTO 0);
       e : OUT SIGNED(63 DOWNTO 0);
       f : OUT SIGNED(95 DOWNTO 0));
END ENTITY block12;

ARCHITECTURE behave OF block12 IS
  FUNCTION DIV (dd, dv : SIGNED) RETURN SIGNED IS
    VARIABLE rm : SIGNED(dd'RANGE);
    VARIABLE dv1 : SIGNED(dd'RANGE);
    VARIABLE dv_neg1 : SIGNED(dd'RANGE);
    VARIABLE trl_zeros : SIGNED((dd'LENGTH - dv'LENGTH - 1) DOWNTO 0);
    VARIABLE q1 : SIGNED((dd'LENGTH - dv'LENGTH)  DOWNTO 0);
    VARIABLE one : SIGNED((dd'LENGTH-1) DOWNTO 0);
    VARIABLE corr : UNSIGNED (2 DOWNTO 0);
```

```
    VARIABLE add_sub : SIGNED(dd'RANGE);
  BEGIN
    trl_zeros := (OTHERS => '0');
    one := (OTHERS => '0');
    one(0) := one(0) OR '1';
    q1 := (OTHERS => '0');
    rm := dd;
    dv1 := dv & trl_zeros;
    dv_neg1 := (NOT dv1) + one;
    iteration : FOR i IN 0 TO (dd'LENGTH - dv'LENGTH - 1) LOOP
      rm := SHIFT_LEFT(rm, 1);
      q1 := SHIFT_LEFT(q1, 1);
      q1(0) := NOT (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1));
      add_sub := (OTHERS => (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1)) );
      rm := rm +( (dv1 AND add_sub) OR (dv_neg1 AND (NOT add_sub)) );
    END LOOP iteration;
    q1 := SHIFT_LEFT(q1, 1);
    q1(dd'LENGTH - dv'LENGTH) := q1(dd'LENGTH - dv'LENGTH) XOR '1';
    corr := dd(dd'LENGTH - 1) & dv(dv'LENGTH - 1) & rm(dd'LENGTH - 1);
    IF corr = "000" OR corr = "111"  THEN
      q1(0) := '1';
    ELSIF corr = "001" OR corr = "110" THEN
      q1(0) := '0';
    ELSIF corr = "010" OR corr = "101" THEN
      q1(0) := '0';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    ELSIF corr = "011" OR corr = "100" THEN
      q1(0) := '1';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    END IF;
    RETURN q1;
  END FUNCTION DIV;
BEGIN
  e<=resize(DIV(resize(s15, 128),s14),64);
  f<=resize(DIV(s22,s14),96);
END ARCHITECTURE behave;


---Block13---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block13 IS
  PORT(b,e : IN SIGNED(63 DOWNTO 0);
       c,f : IN SIGNED(95 DOWNTO 0);
       s23 : OUT SIGNED(99 DOWNTO 0);
       s24 : OUT SIGNED(131 DOWNTO 0));
END ENTITY block13;

ARCHITECTURE behave OF block13 IS
  CONSTANT one_e_10: SIGNED(35 downto 0) :=
"001001010100000010111110010000000000";
BEGIN
  s23<=one_e_10*(e-b);
  s24<=one_e_10*(f-c);
```

```
END ARCHITECTURE behave;


---Block14---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block14 IS
  PORT(s23 : IN SIGNED(99 DOWNTO 0);
       s24 : IN SIGNED(131 DOWNTO 0);
       a,d : IN SIGNED(63 DOWNTO 0);
       g,h : OUT SIGNED(63 DOWNTO 0));
END ENTITY block14;

ARCHITECTURE behave OF block14 IS
  FUNCTION DIV (dd, dv : SIGNED) RETURN SIGNED IS
    VARIABLE rm : SIGNED(dd'RANGE);
    VARIABLE dv1 : SIGNED(dd'RANGE);
    VARIABLE dv_neg1 : SIGNED(dd'RANGE);
    VARIABLE trl_zeros : SIGNED((dd'LENGTH - dv'LENGTH - 1) DOWNTO 0);
    VARIABLE q1 : SIGNED((dd'LENGTH - dv'LENGTH)  DOWNTO 0);
    VARIABLE one : SIGNED((dd'LENGTH-1) DOWNTO 0);
    VARIABLE corr : UNSIGNED (2 DOWNTO 0);
    VARIABLE add_sub : SIGNED(dd'RANGE);
  BEGIN
    trl_zeros := (OTHERS => '0');
    one := (OTHERS => '0');
    one(0) := one(0) OR '1';
    q1 := (OTHERS => '0');
    rm := dd;
    dv1 := dv & trl_zeros;
    dv_neg1 := (NOT dv1) + one;
    iteration : FOR i IN 0 TO (dd'LENGTH - dv'LENGTH - 1) LOOP
      rm := SHIFT_LEFT(rm, 1);
      q1 := SHIFT_LEFT(q1, 1);
      q1(0) := NOT (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1));
      add_sub := (OTHERS => (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1)) );
      rm := rm +( (dv1 AND add_sub) OR (dv_neg1 AND (NOT add_sub)) );
    END LOOP iteration;
    q1 := SHIFT_LEFT(q1, 1);
    q1(dd'LENGTH - dv'LENGTH) := q1(dd'LENGTH - dv'LENGTH) XOR '1';
    corr := dd(dd'LENGTH - 1) & dv(dv'LENGTH - 1) & rm(dd'LENGTH - 1);
    IF corr = "000" OR corr = "111"  THEN
      q1(0) := '1';
    ELSIF corr = "001" OR corr = "110" THEN
      q1(0) := '0';
    ELSIF corr = "010" OR corr = "101" THEN
      q1(0) := '0';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    ELSIF corr = "011" OR corr = "100" THEN
      q1(0) := '1';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    END IF;
    RETURN q1;
  END FUNCTION DIV;
```

```
BEGIN
  g<=resize(DIV(resize(s23, 128),(a-d)),64);
  h<=resize(DIV(s24,(a-d)),64);
END ARCHITECTURE behave;


---Block15---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block15 IS
  PORT(a,b,g,h : IN SIGNED(63 DOWNTO 0);
       c : IN SIGNED(95 DOWNTO 0);
       i,j : OUT SIGNED(63 DOWNTO 0));
END ENTITY block15;

ARCHITECTURE behave OF block15 IS
  CONSTANT one_e_10: SIGNED(35 downto 0) :=
"001001010100000010111110010000000000";
  FUNCTION DIV (dd, dv : SIGNED) RETURN SIGNED IS
    VARIABLE rm : SIGNED(dd'RANGE);
    VARIABLE dv1 : SIGNED(dd'RANGE);
    VARIABLE dv_neg1 : SIGNED(dd'RANGE);
    VARIABLE trl_zeros : SIGNED((dd'LENGTH - dv'LENGTH - 1) DOWNTO 0);
    VARIABLE q1 : SIGNED((dd'LENGTH - dv'LENGTH)  DOWNTO 0);
    VARIABLE one : SIGNED((dd'LENGTH-1) DOWNTO 0);
    VARIABLE corr : UNSIGNED (2 DOWNTO 0);
    VARIABLE add_sub : SIGNED(dd'RANGE);
  BEGIN
    trl_zeros := (OTHERS => '0');
    one := (OTHERS => '0');
    one(0) := one(0) OR '1';
    q1 := (OTHERS => '0');
    rm := dd;
    dv1 := dv & trl_zeros;
    dv_neg1 := (NOT dv1) + one;
    iteration : FOR i IN 0 TO (dd'LENGTH - dv'LENGTH - 1) LOOP
      rm := SHIFT_LEFT(rm, 1);
      q1 := SHIFT_LEFT(q1, 1);
      q1(0) := NOT (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1));
      add_sub := (OTHERS => (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1)) );
      rm := rm +( (dv1 AND add_sub) OR (dv_neg1 AND (NOT add_sub)) );
    END LOOP iteration;
    q1 := SHIFT_LEFT(q1, 1);
    q1(dd'LENGTH - dv'LENGTH) := q1(dd'LENGTH - dv'LENGTH) XOR '1';
    corr := dd(dd'LENGTH - 1) & dv(dv'LENGTH - 1) & rm(dd'LENGTH - 1);
    IF corr = "000" OR corr = "111"  THEN
      q1(0) := '1';
    ELSIF corr = "001" OR corr = "110" THEN
      q1(0) := '0';
    ELSIF corr = "010" OR corr = "101" THEN
      q1(0) := '0';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    ELSIF corr = "011" OR corr = "100" THEN
      q1(0) := '1';
```

```
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    END IF;
    RETURN q1;
  END FUNCTION DIV;
BEGIN
  i<=resize(DIV((a*g),one_e_10)+b,64);
  j<=resize(DIV((a*h),one_e_10)+c,64);
END ARCHITECTURE behave;


---Block16---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block16 IS
  PORT(s18,h,j : IN SIGNED(63 dOWNTO 0);
       xki,yki : IN SIGNED(31 DOWNTO 0);
       k : OUT SIGNED(99 DOWNTO 0));
END ENTITY block16;

ARCHITECTURE behave OF block16 IS
  CONSTANT one_e_10: SIGNED(35 downto 0) :=
"001001010100000010111110010000000000";
BEGIN
  k<=s18*one_e_10+SHIFT_LEFT j*yki,1)+SHIFT_LEFT(h*xki,1);
END ARCHITECTURE behave;


---Block17---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block17 IS
  PORT(g,i : IN SIGNED(63 dOWNTO 0);
       xki, yki,zki : IN SIGNED(31 DOWNTO 0);
       l : OUT SIGNEd(95 DOWNTO 0));
END ENTITY block17;

ARCHITECTURE behave OF block17 IS
  CONSTANT one_e_10: SIGNED(35 downto 0) :=
"001001010100000010111110010000000000";
BEGIN
  l<=SHIFT_LEFT(g*xki+i*yki+zki*one_e_10,1);
END ARCHITECTURE behave;


---Block18---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block18 IS
```

```
  PORT(i,g,rik2 : IN SIGNED(63 DOWNTO 0);
       l : IN SIGNED(95 DOWNTO 0);
       m : OUT SIGNED(191 DOWNTO 0));
END ENTITY block18;

ARCHITECTURE behave OF block18 IS
  CONSTANT one_e_10: SIGNED(35 downto 0) :=
"001001010100000010111110010000000000";
BEGIN
  m<=SHIFT_LEFT(rik2*(g*g+i*i+one_e_10*one_e_10),2)-l*l;
END ARCHITECTURE behave;


---Block19---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block19 IS
  PORT(h,j : IN SIGNED(63 DOWNTO 0);
       xi,yi : IN SIGNED(31 DOWNTO 0);
       s26,s27 : OUT SIGNED(63 DOWNTO 0));
END ENTITY block19;

ARCHITECTURE behave OF block19 IS
  CONSTANT one_e_10: SIGNED(35 downto 0) :=
"001001010100000010111110010000000000";
BEGIN
  s26<=resize(one_e_10*xi-h,64);
  s27<=resize(one_e_10*yi-j,64);
END ARCHITECTURE behave;


---Block20---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block20 IS
  PORT(g,i,rik2,s26,s27 : IN SIGNED(63 DOWNTO 0);
       zi : IN SIGNED(31 DOWNTO 0);
       k : IN SIGNED(99 DOWNTO 0);
       l : IN SIGNED(95 DOWNTO 0);
       n : OUT SIGNED(195 DOWNTO 0));
END ENTITY block20;

ARCHITECTURE behave OF block20 IS
  CONSTANT one_e_10: SIGNED(35 downto 0) :=
"001001010100000010111110010000000000";
BEGIN

n<=SHIFT_LEFT(rik2*(g*s26+i*s27+zi*one_e_10*one_e_10),3)+SHIFT_LEFT(l*k
,1);
END ARCHITECTURE behave;
```

```
---Block21---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block21 IS
  PORT(rik2,s26,s27 : IN SIGNED(63 DOWNTO 0);
       zi : IN SIGNED(31 DOWNTO 0);
       k : IN SIGNED(99 DOWNTO 0);
       o : OUT SIGNED(199 DOWNTO 0));
END ENTITY block21;

ARCHITECTURE behave OF block21 IS
  CONSTANT one_e_10: SIGNED(35 downto 0) :=
"001001010100000010111110010000000000";
BEGIN
  o<=SHIFT_LEFT(rik2*(s26*s26+s27*s27+zi*zi*one_e_10*one_e_10),2)-k*k;
END ARCHITECTURE behave;


---Block21p5---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block21p5 IS
  PORT(n : IN SIGNED(195 DOWNTO 0);
       m : IN SIGNED(191 DOWNTO 0);
       o: IN SIGNED(199 downto 0);
       n_redu : OUT SIGNED(87 DOWNTO 0);
       m_redu : OUT SIGNED(83 DOWNTO 0);
       o_redu : OUT SIGNED(90 DOWNTO 0));
END ENTITY block21p5;

ARCHITECTURE behave OF block21p5 IS
  CONSTANT one_e_30: SIGNED(107 downto 0) :=
"000000001100100111111001011001001111001101000001000110011110100111011011
10101001000000000000000000000000000000";
  FUNCTION DIV (dd, dv : SIGNED) RETURN SIGNED IS
    VARIABLE rm : SIGNED(dd'RANGE);
    VARIABLE dv1 : SIGNED(dd'RANGE);
    VARIABLE dv_neg1 : SIGNED(dd'RANGE);
    VARIABLE trl_zeros : SIGNED((dd'LENGTH - dv'LENGTH - 1) DOWNTO 0);
    VARIABLE q1 : SIGNED((dd'LENGTH - dv'LENGTH)  DOWNTO 0);
    VARIABLE one : SIGNED((dd'LENGTH-1) DOWNTO 0);
    VARIABLE corr : UNSIGNED (2 DOWNTO 0);
    VARIABLE add_sub : SIGNED(dd'RANGE);
  BEGIN
    trl_zeros := (OTHERS => '0');
    one := (OTHERS => '0');
    one(0) := one(0) OR '1';
    q1 := (OTHERS => '0');
    rm := dd;
    dv1 := dv & trl_zeros;
    dv_neg1 := (NOT dv1) + one;
```

```
    iteration : FOR i IN 0 TO (dd'LENGTH - dv'LENGTH - 1) LOOP
      rm := SHIFT_LEFT(rm, 1);
      q1 := SHIFT_LEFT(q1, 1);
      q1(0) := NOT (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1));
      add_sub := (OTHERS => (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1)) );
      rm := rm +( (dv1 AND add_sub) OR (dv_neg1 AND (NOT add_sub)) );
    END LOOP iteration;
    q1 := SHIFT_LEFT(q1, 1);
    q1(dd'LENGTH - dv'LENGTH) := q1(dd'LENGTH - dv'LENGTH) XOR '1';
    corr := dd(dd'LENGTH - 1) & dv(dv'LENGTH - 1) & rm(dd'LENGTH - 1);
    IF corr = "000" OR corr = "111"  THEN
      q1(0) := '1';
    ELSIF corr = "001" OR corr = "110" THEN
      q1(0) := '0';
    ELSIF corr = "010" OR corr = "101" THEN
      q1(0) := '0';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    ELSIF corr = "011" OR corr = "100" THEN
      q1(0) := '1';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    END IF;
    RETURN q1;
  END FUNCTION DIV;
BEGIN
  n_redu <= resize(DIV(n, one_e_30), 88);
  m_redu <= resize(DIV(m, one_e_30), 84);
  o_redu <= resize(DIV(o, one_e_30), 91);
END ARCHITECTURE behave;


---Block22---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block22 IS
  PORT(n_redu : IN SIGNED(87 DOWNTO 0);
       m_redu : IN SIGNED(83 DOWNTO 0);
       s28 : OUT SIGNED(31 DOWNTO 0));
END ENTITY block22;

ARCHITECTURE behave OF block22 IS

  FUNCTION DIV (dd, dv : SIGNED) RETURN SIGNED IS
    VARIABLE rm : SIGNED(dd'RANGE);
    VARIABLE dv1 : SIGNED(dd'RANGE);
    VARIABLE dv_neg1 : SIGNED(dd'RANGE);
    VARIABLE trl_zeros : SIGNED((dd'LENGTH - dv'LENGTH - 1) DOWNTO 0);
    VARIABLE q1 : SIGNED((dd'LENGTH - dv'LENGTH)  DOWNTO 0);
    VARIABLE one : SIGNED((dd'LENGTH-1) DOWNTO 0);
    VARIABLE corr : UNSIGNED (2 DOWNTO 0);
    VARIABLE add_sub : SIGNED(dd'RANGE);
  BEGIN
    trl_zeros := (OTHERS => '0');
    one := (OTHERS => '0');
    one(0) := one(0) OR '1';
```

```
    q1 := (OTHERS => '0');
    rm := dd;
    dv1 := dv & trl_zeros;
    dv_neg1 := (NOT dv1) + one;
    iteration : FOR i IN 0 TO (dd'LENGTH - dv'LENGTH - 1) LOOP
      rm := SHIFT_LEFT(rm, 1);
      q1 := SHIFT_LEFT(q1, 1);
      q1(0) := NOT (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1));
      add_sub := (OTHERS => (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1)) );
      rm := rm +( (dv1 AND add_sub) OR (dv_neg1 AND (NOT add_sub)) );
    END LOOP iteration;
    q1 := SHIFT_LEFT(q1, 1);
    q1(dd'LENGTH - dv'LENGTH) := q1(dd'LENGTH - dv'LENGTH) XOR '1';
    corr := dd(dd'LENGTH - 1) & dv(dv'LENGTH - 1) & rm(dd'LENGTH - 1);
    IF corr = "000" OR corr = "111"  THEN
      q1(0) := '1';
    ELSIF corr = "001" OR corr = "110" THEN
      q1(0) := '0';
    ELSIF corr = "010" OR corr = "101" THEN
      q1(0) := '0';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    ELSIF corr = "011" OR corr = "100" THEN
      q1(0) := '1';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    END IF;
    RETURN q1;
  END FUNCTION DIV;
BEGIN
  s28<=resize(SHIFT_RIGHT(DIV(resize(n_redu, 168),m_redu),1),32);
END ARCHITECTURE behave;


---Block23---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block23 IS
  PORT(o_redu : IN SIGNED(90 DOWNTO 0);
       m_redu : IN SIGNED(83 DOWNTO 0);
       s29 : OUT SIGNED(63 DOWNTO 0));
END ENTITY block23;

ARCHITECTURE behave OF block23 IS
  FUNCTION DIV (dd, dv : SIGNED) RETURN SIGNED IS
    VARIABLE rm : SIGNED(dd'RANGE);
    VARIABLE dv1 : SIGNED(dd'RANGE);
    VARIABLE dv_neg1 : SIGNED(dd'RANGE);
    VARIABLE trl_zeros : SIGNED((dd'LENGTH - dv'LENGTH - 1) DOWNTO 0);
    VARIABLE q1 : SIGNED((dd'LENGTH - dv'LENGTH)  DOWNTO 0);
    VARIABLE one : SIGNED((dd'LENGTH-1) DOWNTO 0);
    VARIABLE corr : UNSIGNED (2 DOWNTO 0);
    VARIABLE add_sub : SIGNED(dd'RANGE);
  BEGIN
    trl_zeros := (OTHERS => '0');
    one := (OTHERS => '0');
```

```
    one(0) := one(0) OR '1';
    q1 := (OTHERS => '0');
    rm := dd;
    dv1 := dv & trl_zeros;
    dv_neg1 := (NOT dv1) + one;
    iteration : FOR i IN 0 TO (dd'LENGTH - dv'LENGTH - 1) LOOP
      rm := SHIFT_LEFT(rm, 1);
      q1 := SHIFT_LEFT(q1, 1);
      q1(0) := NOT (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1));
      add_sub := (OTHERS => (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1)) );
      rm := rm +( (dv1 AND add_sub) OR (dv_neg1 AND (NOT add_sub)) );
    END LOOP iteration;
    q1 := SHIFT_LEFT(q1, 1);
    q1(dd'LENGTH - dv'LENGTH) := q1(dd'LENGTH - dv'LENGTH) XOR '1';
    corr := dd(dd'LENGTH - 1) & dv(dv'LENGTH - 1) & rm(dd'LENGTH - 1);
    IF corr = "000" OR corr = "111"  THEN
      q1(0) := '1';
    ELSIF corr = "001" OR corr = "110" THEN
      q1(0) := '0';
    ELSIF corr = "010" OR corr = "101" THEN
      q1(0) := '0';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    ELSIF corr = "011" OR corr = "100" THEN
      q1(0) := '1';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    END IF;
    RETURN q1;
  END FUNCTION DIV;
BEGIN
  s29<=resize(DIV(resize(o_redu, 168),m_redu),64);
END ARCHITECTURE behave;


---Block24---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block24 IS
  PORT(s29 : IN SIGNED(63 DOWNTO 0);
       s28 : IN SIGNED(31 DOWNTO 0);
       root: OUT SIGNED(31 DOWNTO 0));
END ENTITY block24;

ARCHITECTURE behave OF block24 IS
  SIGNAL s30 : SIGNED(63 DOWNTO 0);
BEGIN
  s30<=s28*s28-S29;
  squareroot:process (s30)
      variable q,r,s,t:signed(63 downto 0);
      begin
      q:=s30;
      s:=to_signed(0,64);
      r:=to_signed(0,64);
      t:="0100000000000000000000000000000000000000000000000000000000000000
000";
```

```
      for j in 1 to 32 loop
      s:=r+t;
      if s<=q then
            q:=q-s;
            r:=s+t;
      end if;
      r:=shift_right(r,1);
      t:=shift_right(t,2);
      end loop;
  root<=resize(r,32);
  end process squareroot;
END ARCHITECTURE behave;


---Block25---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block25 IS
  PORT(a,b,g,h : IN SIGNED(63 DOWNTO 0);
       c : IN SIGNED(95 DOWNTO 0);
       s28, root : IN SIGNED(31 DOWNTO 0);
       x1,y1,z1 : OUT SIGNED(31 DOWNTO 0));
END ENTITY block25;

ARCHITECTURE behave OF block25 IS
  SIGNAL s31,s32,s33 : SIGNED(31 DOWNTO 0);
  CONSTANT one_e_10: SIGNED(35 downto 0) :=
"001001010100000010111110010000000000";
  FUNCTION DIV (dd, dv : SIGNED) RETURN SIGNED IS
    VARIABLE rm : SIGNED(dd'RANGE);
    VARIABLE dv1 : SIGNED(dd'RANGE);
    VARIABLE dv_neg1 : SIGNED(dd'RANGE);
    VARIABLE trl_zeros : SIGNED((dd'LENGTH - dv'LENGTH - 1) DOWNTO 0);
    VARIABLE q1 : SIGNED((dd'LENGTH - dv'LENGTH)  DOWNTO 0);
    VARIABLE one : SIGNED((dd'LENGTH-1) DOWNTO 0);
    VARIABLE corr : UNSIGNED (2 DOWNTO 0);
    VARIABLE add_sub : SIGNED(dd'RANGE);
  BEGIN
    trl_zeros := (OTHERS => '0');
    one := (OTHERS => '0');
    one(0) := one(0) OR '1';
    q1 := (OTHERS => '0');
    rm := dd;
    dv1 := dv & trl_zeros;
    dv_neg1 := (NOT dv1) + one;
    iteration : FOR i IN 0 TO (dd'LENGTH - dv'LENGTH - 1) LOOP
      rm := SHIFT_LEFT(rm, 1);
      q1 := SHIFT_LEFT(q1, 1);
      q1(0) := NOT (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1));
      add_sub := (OTHERS => (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1)) );
      rm := rm +( (dv1 AND add_sub) OR (dv_neg1 AND (NOT add_sub)) );
    END LOOP iteration;
    q1 := SHIFT_LEFT(q1, 1);
    q1(dd'LENGTH - dv'LENGTH) := q1(dd'LENGTH - dv'LENGTH) XOR '1';
```

```
    corr := dd(dd'LENGTH - 1) & dv(dv'LENGTH - 1) & rm(dd'LENGTH - 1);
    IF corr = "000" OR corr = "111"  THEN
      q1(0) := '1';
    ELSIF corr = "001" OR corr = "110" THEN
      q1(0) := '0';
    ELSIF corr = "010" OR corr = "101" THEN
      q1(0) := '0';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    ELSIF corr = "011" OR corr = "100" THEN
      q1(0) := '1';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    END IF;
    RETURN q1;
  END FUNCTION DIV;
BEGIN
  s31<=S28+root;
  s32<=resize(DIV((g*s31+h),one_e_10),32);
  s33<=resize(DIV((a*s32+b*s31+c),one_e_10),32);
  z1<=s31;
  x1<=s32;
  y1<=s33;
END ARCHITECTURE behave;


---Block26---

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY block26 IS
  PORT(a,b,g,h : IN SIGNED(63 DOWNTO 0);
       c : IN SIGNED(95 DOWNTO 0);
       s28, root : IN SIGNED(31 DOWNTO 0);
       x2,y2,z2 : OUT SIGNED(31 DOWNTO 0));
END ENTITY block26;

ARCHITECTURE behave OF block26 IS
  SIGNAL s34,s35,s36 : SIGNED(31 DOWNTO 0);
  CONSTANT one_e_10: SIGNED(35 downto 0) :=
         "001001010100000010111110010000000000";
  FUNCTION DIV (dd, dv : SIGNED) RETURN SIGNED IS
    VARIABLE rm : SIGNED(dd'RANGE);
    VARIABLE dv1 : SIGNED(dd'RANGE);
    VARIABLE dv_neg1 : SIGNED(dd'RANGE);
    VARIABLE trl_zeros : SIGNED((dd'LENGTH - dv'LENGTH - 1) DOWNTO 0);
    VARIABLE q1 : SIGNED((dd'LENGTH - dv'LENGTH)  DOWNTO 0);
    VARIABLE one : SIGNED((dd'LENGTH-1) DOWNTO 0);
    VARIABLE corr : UNSIGNED (2 DOWNTO 0);
    VARIABLE add_sub : SIGNED(dd'RANGE);
  BEGIN
    trl_zeros := (OTHERS => '0');
    one := (OTHERS => '0');
    one(0) := one(0) OR '1';
    q1 := (OTHERS => '0');
    rm := dd;
    dv1 := dv & trl_zeros;
```

```
    dv_neg1 := (NOT dv1) + one;
    iteration : FOR i IN 0 TO (dd'LENGTH - dv'LENGTH - 1) LOOP
      rm := SHIFT_LEFT(rm, 1);
      q1 := SHIFT_LEFT(q1, 1);
      q1(0) := NOT (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1));
      add_sub := (OTHERS => (rm(dd'LENGTH -1) XOR dv1(dv1'LENGTH-1)) );
      rm := rm +( (dv1 AND add_sub) OR (dv_neg1 AND (NOT add_sub)) );
    END LOOP iteration;
    q1 := SHIFT_LEFT(q1, 1);
    q1(dd'LENGTH - dv'LENGTH) := q1(dd'LENGTH - dv'LENGTH) XOR '1';
    corr := dd(dd'LENGTH - 1) & dv(dv'LENGTH - 1) & rm(dd'LENGTH - 1);
    IF corr = "000" OR corr = "111"  THEN
      q1(0) := '1';
    ELSIF corr = "001" OR corr = "110" THEN
      q1(0) := '0';
    ELSIF corr = "010" OR corr = "101" THEN
      q1(0) := '0';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    ELSIF corr = "011" OR corr = "100" THEN
      q1(0) := '1';
      q1 := q1 + one((q1'LENGTH-1) DOWNTO 0);
    END IF;
    RETURN q1;
  END FUNCTION DIV;
BEGIN
  s34<=s28-root;
  s35<=resize(DIV((g*s34+h),one_e_10),32);
  s36<=resize(DIV((a*s35+b*s34+c),one_e_10),32);
  z2<=s34;
  x2<=s35;
  y2<=S36;
END ARCHITECTURE behave;
```

Top Level VHDL Model

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY hyperbolic IS
  PORT(xi,xj,xk,xl,yi,yj,yk,yl,zi,zj,zk,zl,ti,tk,tj,tl: in SIGNED(31
downto 0);
       x1,x2,y1,y2,z1,z2: out SIGNED(31 downto 0));
END ENTITY hyperbolic;

ARCHITECTURE behave OF hyperbolic IS
  signal o: SIGNED(199 downto 0);
  signal o_redu : SIGNED(90 DOWNTO 0);
  signal n: SIGNED(195 downto 0);
  signal n_redu :SIGNED(87 DOWNTO 0);
  signal m: SIGNED(191 downto 0);
  signal m_redu :SIGNED(83 DOWNTO 0);
  signal c,f,l: SIGNED(95 downto 0);
  signal s12,s16,s21,s22,s24: SIGNED(131 downto 0);
  signal s9,s11,s13,s15,s23,k: SIGNED(99 downto 0);
```

```
  signal
s1,s2,s3,s4,s5,s6,s7,s8,s10,s14,s17,s18,s19,s20,s26,s27,s29,s30,
   a,b,d,e,g,h,i,j,yi2,yj2,yk2,yl2,xi2,xj2,xk2,xl2,zi2,zj2,zk2,zl2,
   rij2,rik2,rkl2,rkj2: SIGNED(63 downto 0);
  signal
rij,rik,rkj,rkl,xji,xki,xjk,xlk,yji,yki,yjk,ylk,zji,zki,zjk,zlk,
   root,s28: SIGNED(31 downto 0);
  COMPONENT block1 IS
    PORT(ti, tj, tk, tl : IN SIGNED(31 DOWNTO 0);
         s1, s2, s3, s4 : OUT SIGNED(63 DOWNTO 0));
  END COMPONENT block1;
  COMPONENT block2 IS
  PORT(s1,s2,s3,s4 : IN SIGNED(63 DOWNTO 0);
       rij,rik,rkl,rkj : OUT SIGNED(31 DOWNTO 0);
       rij2,rik2,rkl2,rkj2 : OUT SIGNED(63 DOWNTO 0));
  END COMPONENT block2;
  COMPONENT block3 IS
    PORT(xi,xj,xk,yi,yj,yk,zi,zj,zk : IN SIGNED(31 DOWNTO 0);
         xji,yji,zji,xki,yki,zki : OUT SIGNED(31 DOWNTO 0);
         xi2,yi2,zi2,xj2,yj2,zj2 : OUT SIGNED(63 DOWNTO 0));
  END COMPONENT block3;
  COMPONENT block4 IS
    PORT(xj,yj,zj,xk,yk,zk,xl,yl,zl : IN SIGNED(31 DOWNTO 0);
         xlk,ylk,zlk,xjk,yjk,zjk : OUT SIGNED(31 DOWNTO 0);
         xk2,yk2,zk2,xl2,yl2,zl2 : OUT SIGNED(63 DOWNTO 0));
  END COMPONENT block4;
  COMPONENT block5 IS
    PORT(rij,rik,xji,xki,yji,yki : IN SIGNED(31 DOWNTO 0);
         s9 : OUT SIGNED(99 DOWNTO 0);
         s10 : OUT SIGNED(63 DOWNTO 0));
  END COMPONENT block5;
  COMPONENT block6 IS
    PORT(rij,rik,rkl,rkj,xjk,xlk,zji,zki : IN SIGNED(31 DOWNTO 0);
         s11,s13 : OUT SIGNED(99 DOWNTO 0));
  END COMPONENT block6;
  COMPONENT block7 IS
    PORT(rkj,rkl,yjk,ylk,zjk,zlk : IN SIGNED(31 DOWNTO 0);
         s14 : OUT SIGNED(63 DOWNTO 0);
         s15 : OUT SIGNEd(99 DOWNTO 0));
  END COMPONENT block7;
  COMPONENT block8 IS

PORT(yi2,yj2,yk2,yl2,xi2,xj2,xk2,xl2,zi2,zj2,zk2,zl2,rij2,rik2,rkl2,rkj
2: IN SIGNED(63 downto 0);
         s17,s18,s19,s20 : OUT SIGNED(63 DOWNTO 0));
  END COMPONENT block8;
  COMPONENT block9 IS
    PORT(rij,rik,rkj,rkl : IN SIGNED(31 DOWNTO 0);
         s17,s18,s19,s20 : IN SIGNED(63 DOWNTO 0);
         s21,s22 : OUT SIGNED(131 DOWNTO 0));
  END COMPONENT block9;
  COMPONENT block10 IS
    PORT(s9,s11 : IN SIGNED(99 DOWNTO 0);
         s10 : IN SIGNED(63 DOWNTO 0);
         a,b : OUT SIGNED(63 DOWNTO 0));
  END COMPONENT block10;
  COMPONENT block11 IS
```

```
  PORT(s21 : IN SIGNED(131 DOWNTO 0);
       s13 : IN SIGNED(99 DOWNTO 0);
       s10,s14 : IN SIGNED(63 DOWNTO 0);
       c : OUT SIGNED(95 DOWNTO 0);
       d : OUT SIGNED(63 DOWNTO 0));
END COMPONENT block11;
COMPONENT block12 IS
  PORT(s22 : IN SIGNED(131 DOWNTO 0);
       s15 : IN SIGNED(99 DOWNTO 0);
       s14 : IN SIGNED(63 DOWNTO 0);
       e : OUT SIGNED(63 DOWNTO 0);
       f : OUT SIGNED(95 DOWNTO 0));
END COMPONENT block12;
COMPONENT block13 IS
  PORT(b,e : IN SIGNED(63 DOWNTO 0);
       c,f : IN SIGNED(95 DOWNTO 0);
       s23 : OUT SIGNED(99 DOWNTO 0);
       s24 : OUT SIGNED(131 DOWNTO 0));
END COMPONENT block13;
COMPONENT block14 IS
  PORT(s23 : IN SIGNED(99 DOWNTO 0);
       s24 : IN SIGNED(131 DOWNTO 0);
       a,d : IN SIGNED(63 DOWNTO 0);
       g,h : OUT SIGNED(63 DOWNTO 0));
END COMPONENT block14;
COMPONENT block15 IS
  PORT(a,b,g,h : IN SIGNED(63 DOWNTO 0);
       c : IN SIGNED(95 DOWNTO 0);
       i,j : OUT SIGNED(63 DOWNTO 0));
END COMPONENT block15;
COMPONENT block16 IS
  PORT(s18,h,j : IN SIGNED(63 dOWNTO 0);
       xki,yki : IN SIGNED(31 DOWNTO 0);
       k : OUT SIGNED(99 DOWNTO 0));
END COMPONENT block16;
COMPONENT block17 IS
  PORT(g,i : IN SIGNED(63 dOWNTO 0);
       xki, yki,zki : IN SIGNED(31 DOWNTO 0);
       l : OUT SIGNEd(95 DOWNTO 0));
END COMPONENT block17;
COMPONENT block18 IS
  PORT(i,g,rik2 : IN SIGNED(63 DOWNTO 0);
       l : IN SIGNED(95 DOWNTO 0);
       m : OUT SIGNED(191 DOWNTO 0));
END COMPONENT block18;
COMPONENT block19 IS
  PORT(h,j : IN SIGNED(63 DOWNTO 0);
       xi,yi : IN SIGNED(31 DOWNTO 0);
       s26,s27 : OUT SIGNED(63 DOWNTO 0));
END COMPONENT block19;
COMPONENT block20 IS
 PORT(g,i,rik2,s26,s27 : IN SIGNED(63 DOWNTO 0);
     zi : IN SIGNED(31 DOWNTO 0);
     k : IN SIGNED(99 DOWNTO 0);
     l : IN SIGNED(95 DOWNTO 0);
     n : OUT SIGNED(195 DOWNTO 0));
END COMPONENT block20;
```

```
  COMPONENT block21 IS
    PORT(rik2,s26,s27 : IN SIGNED(63 DOWNTO 0);
         zi : IN SIGNED(31 DOWNTO 0);
         k : IN SIGNED(99 DOWNTO 0);
         o : OUT SIGNED(199 DOWNTO 0));
  END COMPONENT block21;
  COMPONENT block21p5 IS
    PORT(n : IN SIGNED(195 DOWNTO 0);
         m : IN SIGNED(191 DOWNTO 0);
         o: IN SIGNED(199 downto 0);
         n_redu : OUT SIGNED(87 DOWNTO 0);
         m_redu : OUT SIGNED(83 DOWNTO 0);
         o_redu : OUT SIGNED(90 DOWNTO 0));
  END COMPONENT block21p5;
  COMPONENT block22 IS
    PORT(n_redu : IN SIGNED(87 DOWNTO 0);
         m_redu : IN SIGNED(83 DOWNTO 0);
         s28 : OUT SIGNED(31 DOWNTO 0));
  END COMPONENT block22;
  COMPONENT block23 IS
    PORT(o_redu : IN SIGNED(90 DOWNTO 0);
         m_redu : IN SIGNED(83 DOWNTO 0);
         s29 : OUT SIGNED(63 DOWNTO 0));
  END COMPONENT block23;
  COMPONENT block24 IS
    PORT(s29 : IN SIGNED(63 DOWNTO 0);
         s28 : IN SIGNED(31 DOWNTO 0);
         root: OUT SIGNED(31 DOWNTO 0));
  END COMPONENT block24;
  COMPONENT block25 IS
    PORT(a,b,g,h : IN SIGNED(63 DOWNTO 0);
         c : IN SIGNED(95 DOWNTO 0);
         s28, root : IN SIGNED(31 DOWNTO 0);
         x1,y1,z1 : OUT SIGNED(31 DOWNTO 0));
  END COMPONENT block25;
  COMPONENT block26 IS
    PORT(a,b,g,h : IN SIGNED(63 DOWNTO 0);
         c : IN SIGNED(95 DOWNTO 0);
         s28, root : IN SIGNED(31 DOWNTO 0);
         x2,y2,z2 : OUT SIGNED(31 DOWNTO 0));
  END COMPONENT block26;
BEGIN
  blk1 : COMPONENT block1 PORT MAP(ti, tj, tk, tl,s1, s2, s3, s4);
  blk2 : COMPONENT block2 PORT
         MAP(s1,s2,s3,s4,rij,rik,rkl,rkj,rij2,rik2,rkl2,rkj2);
  blk3 : COMPONENT block3 PORT MAP(xi,xj,xk,yi,yj,yk,zi,zj,zk,
         xji,yji,zji,xki,yki,zki,xi2,yi2,zi2,xj2,yj2,zj2);
  blk4 : COMPONENT block4 PORT MAP(xj,yj,zj,xk,yk,zk,xl,yl,zl,
         xlk,ylk,zlk,xjk,yjk,zjk,xk2,yk2,zk2,xl2,yl2,zl2);
  blk5 : COMPONENT block5 PORT MAP(rij,rik,xji,xki,yji,yki,s9,s10);
  blk6 : COMPONENT block6 PORT
         MAP(rij,rik,rkl,rkj,xjk,xlk,zji,zki,s11,s13);
  blk7 : COMPONENT block7 PORT MAP(rkj,rkl,yjk,ylk,zjk,zlk,s14,s15);
  blk8 : COMPONENT block8 PORT MAP(yi2,yj2,yk2,yl2,xi2,xj2,xk2,xl2,
          zi2,zj2,zk2,zl2,rij2,rik2,rkl2,rkj2,s17,s18,s19,s20);
  blk9 : COMPONENT block9 PORT
         MAP(rij,rik,rkj,rkl,s17,s18,s19,s20,s21,s22);
```

```
  blk10 : COMPONENT block10 PORT MAP(s9,s11,s10,a,b);
  blk11 : COMPONENT block11 PORT MAP(s21,s13,s10,s14,c,d);
  blk12 : COMPONENT block12 PORT MAP(s22,s15,s14,e,f);
  blk13 : COMPONENT block13 PORT MAP(b,e,c,f,s23,s24);
  blk14 : COMPONENT block14 PORT MAP(s23,s24,a,d,g,h);
  blk15 : COMPONENT block15 PORT MAP(a,b,g,h,c,i,j);
  blk16 : COMPONENT block16 PORT MAP(s18,h,j,xki,yki,k);
  blk17 : COMPONENT block17 PORT MAP(g,i,xki,yki,zki,l);
  blk18 : COMPONENT block18 PORT MAP(i,g,rik2,l,m);
  blk19 : COMPONENT block19 PORT MAP(h,j,xi,yi,s26,s27);
  blk20 : COMPONENT block20 PORT MAP(g,i,rik2,s26,s27,zi,k,l,n);
  blk21 : COMPONENT block21 PORT MAP(rik2,s26,s27,zi,k,o);
  blk21p5 : COMPONENT block21p5 PORT MAP(n,m,o,n_redu,m_redu,o_redu);
  blk22 : COMPONENT block22 PORT MAP(n_redu,m_redu,s28);
  blk23 : COMPONENT block23 PORT MAP(o_redu,m_redu,s29);
  blk24 : COMPONENT block24 PORT MAP(s29,s28,root);
  blk25 : COMPONENT block25 PORT MAP(a,b,g,h,c,s28, root,x1,y1,z1);
  blk26 : COMPONENT block26 PORT MAP(a,b,g,h,c,s28, root,x2,y2,z2);
END ARCHITECTURE behave;
```

APPENDIX C

TOP LEVEL VERILOG MODULE

This appendix is the listing of top level Verilog model used for the gate level simulation of the synthesized blocks.

Top Level Gate Level Verilog Module

```
module
hyperbolic_gate(xi,xj,xk,xl,yi,yj,yk,yl,zi,zj,zk,zl,ti,tk,tj,tl,x1,x2,y
1,y2,z1,z2);
    input [31:0] xi,xj,xk,xl,yi,yj,yk,yl,zi,zj,zk,zl,ti,tk,tj,tl;
    output [31:0] x1,x2,y1,y2,z1,z2;

    block1 blk1(ti, tj, tk, tl,s1, s2, s3, s4);
    block2 blk2(s1,s2,s3,s4,rij,rik,rkl,rkj,rij2,rik2,rkl2,rkj2);
    block3 blk3(xi,xj,xk,yi,yj,yk,zi,zj,zk,
                xji,yji,zji,xki,yki,zki,xi2,yi2,zi2,xj2,yj2,zj2);
    block4 blk4(xj,yj,zj,xk,yk,zk,xl,yl,zl,
                xlk,ylk,zlk,xjk,yjk,zjk,xk2,yk2,zk2,xl2,yl2,zl2);
    block5 blk5(rij,rik,xji,xki,yji,yki,s9,s10);
    block6 blk6(rij,rik,rkl,rkj,xjk,xlk,zji,zki,s11,s13);
    block7 blk7(rkj,rkl,yjk,ylk,zjk,zlk,s14,s15);
    block8 blk8(yi2,yj2,yk2,yl2,xi2,xj2,xk2,xl2,
                zi2,zj2,zk2,zl2,rij2,rik2,rkl2,rkj2,s17,s18,s19,s20);
    block9 blk9(rij,rik,rkj,rkl,s17,s18,s19,s20,s21,s22);
    block10 blk10(s9,s11,s10,a,b);
    block11 blk11(s21,s13,s10,s14,c,d);
    block12 blk12(s22,s15,s14,e,f);
    block13 blk13(b,e,c,f,s23,s24);
    block14 blk14(s23,s24,a,d,g,h);
    block15 blk15(a,b,g,h,c,i,j);
    block16 blk16(s18,h,j,xki,yki,k);
    block17 blk17(g,i,xki,yki,zki,l);
    block18 blk18(i,g,rik2,l,m);
    block19 blk19(h,j,xi,yi,s26,s27);
    block20 blk20(g,i,rik2,s26,s27,zi,k,l,n);
    block21 blk21(rik2,s26,s27,zi,k,o);
    block21p5 blk21p5(n,m,o,n_redu,m_redu,o_redu);
    block22 blk22(n_redu,m_redu,s28);
    block23 blk23(o_redu,m_redu,s29);
    block24 blk24(s29,s28,root);
    block25 blk25(a,b,g,h,c,s28, root,x1,y1,z1);
    block26 blk26(a,b,g,h,c,s28, root,x2,y2,z2);

endmodule
```

REFERENCES

1. B.T. Fang, Simple solutions for a hyperbolic and related position fixes, *IEEE Trans. on Aerosp. and Elect. Systems*, vol. 26, no. 5, pp. 748-753, Sept 1990.

2. K.J. Krizman, T.E. Biedka, and T.S. Rappaport, Wireless position location: Fundamentals, implementation strategies, and sources of error, invited paper, *Proceedings of Virginia Tech's Seventh Symposium on Wireless Personal Communications*, Blacksburg, VA, June 11-13, 1997.

3. T. S. Rappaport, J. H. Reed, B. D. Woerner, Position Location using Wireless Communications on Highways of the Future, invited paper, *IEEE Communications Magazine*, Vol. 34, No. 10, October 1996, pp. 33-41.

4. J. C. Liberti, Jr., & T. S. Rappaport, *Smart Antennas for Wireless Communications: IS-95 & Third Generation CDMA Applications*, Prentice Hall, 1999.

5. Y. T. Chan and K. C. Ho, Simple and Efficient Estimator for Hyperbolic Location, *IEEE Transactions on Signal Processing*, vol. 42, no. 8, pp. 1905-1915, August 1994.

6. *Cadence OpenBook Documentation*, Cadence Design Systems, 2000.

7. *HSPICE User Manual*, Avanti Corp, 2001.