Fall 2002

VLSI design of stability routing protocol for sensors in wireless mobile ad-hoc networks

Vishnu M. Mandava
New Jersey Institute of Technology

ABSTRACT

VLSI DESIGN OF STABILITY ROUTING PROTOCOL FOR SENSORS IN WIRELESS MOBILE AD-HOC NETWORKS

by
Vishnu M. Mandava

This thesis gives a detailed description of the Application specific integrated circuit (ASIC) design of Stability routing protocol for sensors in mobile ad-hoc networks. The Stability routing protocol is based on the signal strength and position components during data transmission while considering sensors in an ad-hoc network. A general ad-hoc network has unpredictable and variable mobility patterns therefore the signal strength criteria is adopted for routing. Signal strength criteria has been proved to be efficient for communication between the mobile nodes without any data loss. In this thesis an architecture for a processor implementing stability routing protocol for effective communication has been designed. The processor detects the alert signal from the sensor network and sends an emergency signal to all the other nodes in the network. Apart form sending the emergency signal the processor also sends the position and velocity components of its own node to all the other nodes in the network. The other functionality of the processor is whenever the processor receives data from another node it updates the information and sends that information to the destination node. A VHDL model for this architecture was developed, a selected set of specific conditions are evaluated through simulation. VHDL simulation validates the functionality of the architecture. This model was synthesized and the place and route was done using cadence tools.

VLSI DESIGN OF STABILITY ROUTING PROTOCOL FOR SENSORS IN WIRELESS MOBILE AD-HOC NETWORKS

by
Vishnu M. Mandava

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 2003

APPROVAL PAGE

VLSI DESIGN OF STABILITY ROUTING PROTOCOL FOR SENSORS IN WIRELESS MOBILE AD-HOC NETWORKS

Vishnu.M.Mandava

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

Dr. Lev Zakrevski, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

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

BIOGRAPHICAL SKETCH

Author: Vishnu M.Mandava

Degree: Master of Science

Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2003

- Bachelor of Engineering in Electronics and Communication,
  Sri Rama Krishna Engineering College, Coimbatore, India, 2000

Major: Electrical Engineering

To My Parents and the Almighty

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Dr. Durga Misra, who not only served as my thesis advisor, providing valuable resources, but also constantly gave me support, encouragement and reassurance. Special thanks given to Dr. Lev Zakrveski, Dr. Symeon Papavassiliou for actively participating in my committee.

I also wish to thank my fellow graduate students in Electrical Engineering for their support over the years and a special thanks to Kiran Gururaj for his assistance over the years.

TABLE OF CONTENTS

TABLE OF CONTENTS

(Continued)

Chapter **Page**

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Introduction to Ad-hoc Networks and Applications

An ad-hoc network is defined as the cooperative engagement of wireless mobile hosts forming a temporary network without any intervention of established infrastructure or centralized administration. In such environment, it may be necessary for each of the node to act as a router, they are free to move randomly and organize themselves arbitrarily, forward the packets to its destination. Due to the limited range of each mobile host's wireless transmission mobile users will want to communicate in situations in which no fixed wired infrastructure is available, either because it may not be economically or physically feasible to provide the necessary infrastructure or because the expediency of the situation does not permit its installation. The links of the network are dynamic and are based on the proximity of one node to another node. These links are likely to break and change as the nodes move about the network. Such a network may operate in standalone fashion, or may be connected to the large internet. An ad-hoc mobile networking is all about providing connectivity between mobile nodes, which have no supporting connections to the fixed networking infrastructure.

Since ad-hoc networks have no fixed infrastructure, therefore they can be deployed rapidly with relatively low cost. For this reason, the applications of ad-hoc networks extend from military u ses to commercial uses. H owever, a wireless ad-hoc network is practically vulnerable because of its dynamically changing topology, and lack of centralized monitoring.

The ad-hoc networks are used to support emergency responses to natural disasters, surveillance and information gathering in hostile territories and rescue operations where existing communication infrastructures are not available, rapid deployment of an unstructured mobile network, where each unit is capable of transmitting video information and sensor data, would be essential. The requirements may include some or all of the following a higher bandwidth (for transmitting video data), mobility, sufficient area coverage, communications beyond the line of sight, and low energy consumption.

Because of the temporal nature of the network links, because of the additional constraints implied by mobile nodes, such as limited bandwidth and power, conventional routing protocols are not appropriate for ad-hoc mobile networks. Therefore, new protocols are being developed to exploit the properties of such networks in more appropriate way, one such protocol is Stability Routing protocol [4, 7] (SRP). This protocol adapts quickly to routing changes when host movement is frequent and it also guarantee's that the data is received at the destination node.

1.2 Principles of this Thesis

In this thesis, architecture for a processor implementing the Stability Routing Protocol is designed. A Subnet of 8 nodes which can be scalable to a large network is considered, each node is considered to be in the transmittable distance of atleast one node in the network. It is considered that each node moves in a two dimensional fashion, each node gets its position and velocity components from the GPS. The actual coordinates from the GPS system can be converted into planar coordinates [8-10] using various conversion systems. The processor checks if there is a change in the position or velocity components,

the processor also checks for any alert signal from the sensor network. The alert signal from the sensor network is considered as highest priority. There are total of 8 inputs and two outputs of which 5 inputs are from the GPS unit and one from the sensor network and the other two from the transceiver.

1.3 Outline of the Thesis

The remainder of the thesis is organized as follows. Chapter 2 provides an overview of Stability Routing Algorithm, Application of the algorithm for communication between the sensors. Chapter 3 describes the architecture of the chip and its various components. Chapter 4 describes the implementation of the architecture in VHDL and different implementation issues, the test and the simulation results. Chapter 5 addresses the synthesis issues, problems encountered during the synthesis and modifications of the actual VHDL model to make it synthesizable. Chapter 6 describes the physical layout. Chapter 7 concludes this thesis by summarizing the work and discussing alternative implementations for reducing the gate count.

CHAPTER 2

STABILITY ROUTING ALGORITHM

2.1 Overview of Related Routing Algorithms

Some of the related routing algorithms for mobile ad-hoc networks are Cluster based routing protocol [1], Ad-hoc on-demand distance vector routing protocol [2], Dynamic source routing protocol [3], Signal stability routing protocol [4].

In the Cluster based routing protocol the nodes are divided into clusters, when a node comes up into a cluster it enters the undecided state and it sends a hello message, when a cluster-head receives this hello message it responds with a hello message immediately. When the undecided node gets this message it becomes a member of the cluster. Each node in the cluster maintains a neighbor table which contains the state of neighbor (cluster-head or member). When a source node has to send data to destination node it floods route request packet to cluster-heads if the destination is in this cluster then the cluster-head sends the packet to the destination if the destination is not in this cluster then the cluster-head sends the packet to all the other cluster-heads in the network the cluster-head that has the information of the route sends the route.

In the Ad-hoc on-demand distance vector protocol to find a path to the destination the source broadcasts a route request packet to its neighbor and the neighbor in turn broadcast the packet to their neighbors this is done till the packet reaches its destination or an intermediate node that has a recent route information about the destination.

In the Dynamic source routing each node maintains route cache and the cache contains the source route that it is aware of, when the source node wants to send a packet

to a destination it looks up the cache if it already contains the route to the destination. If the route is not existent in the cache then it initiates the route discovery by broadcasting route request packet, a route reply is generated when the intermediate node with current information about the destination receives the route request.

In the signal stability routing protocol the routes are based on the signal strength between the nodes where signal strength is the maximum time until which the node stays in the transmittable distance of any other node. This route selection criteria has the effect of choosing routes that have stronger connectivity, in this protocol each node maintains the signal stability table and the routing table which has the information of stability of each node and the routing information to all its destinations in the network, if a source wants to send a packet to a particular destination it looks at the routing table and sends the packet.

2.2 Why this Protocol

The sensors main aim is to establish effective communication between the various nodes of the network. In the case of emergency in a hazardous area communication between the various nodes becomes critical. So the information that has to be send to various nodes using the wireless media requires an algorithm which has the capability of transmitting the information without any loss. When compared to the other routing algorithms the data is not send in a particular path and the path to the destination is not known initially and is found only when a node wants to send any information where as in the stability routing algorithm each node has the routing table to all its destinations and the information is sent

in the route which has strongest connectivity which ensures that the information is sent to its destination with out any loss.

2.3 Overview of the Algorithm

The case of mobile Ad-Hoc network is considered where each node is moving in a two dimensional fashion where in each node is characterized by four parameters, two parameters represent the position of the node, characterized by '**X**' and '**Y**' ,the other two parameters represent the velocity with which the node is moving, characterized by '**U**' and '**V**'. Where **U** = dx/dt, **V** = dy/dt. Each node in the network should be able to communicate with one or the other nodes in the network. I communication is not possible the node may not be in the network. A node should be within the maximum possible Transmission Range of at least one node when it is considered as a part of a network. The maximum possible transmission range is 'D'.

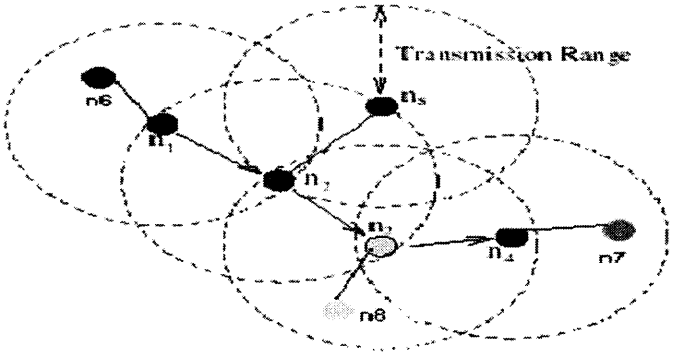

Figure 2.1 Subnet of '8' nodes.

We assume that the positions and velocities of all nodes are known. If necessary, this information has to be broadcasted. Flooding algorithm is used for this purpose. Each node had information about

1. Real Position, Velocity.
2. Position and Velocity broadcasted to the whole system in the past.

If there is a difference between '1' and '2' is greater than a given threshold broadcast is initialized and all the other nodes in the network are informed of this change.

Each node ***'A'*** calculates the stability of all links of the type ***(A, B)*** where **'B'** is another node in the network. The stability t ***(A, B)*** is determined as the expected time, where t ***(A, B)*** is defined as the maximum time until which node **'A'** stays within the transmittable distance of node **'B'.** We note that t ***(A, B)*** is equal to t ***(B, A)***. At time t = 0 **|AB| < D.** Each node in the network knows the stability of the other nodes in the network. Each node ***'A'*** constructs a routing table for all destinations ***X,*** using shortest path algorithm to find the path from ***'A'*** to ***'X'*** with maximal stability. Stability of the path is determined as the minimal stability of links.

2.4 Application

This thesis concentrates on the effective use of this algorithm in coordination with the sensor unit output. The sensor unit output is of prime importance and should be considered with highest priority. By the use of this algorithm it can be ensured that the sensor information can be sent to all nodes in the network with out any loss.

CHAPTER 3

ARCHITECTURAL DESIGN OF THE CHIP

3.1 Overview of the System

The proposed VLSI architectural design of the whole chip is discussed in this section Figure.3.1 gives an overview of the whole system, which can be considered as a unit in a single node of an Ad-hoc network.

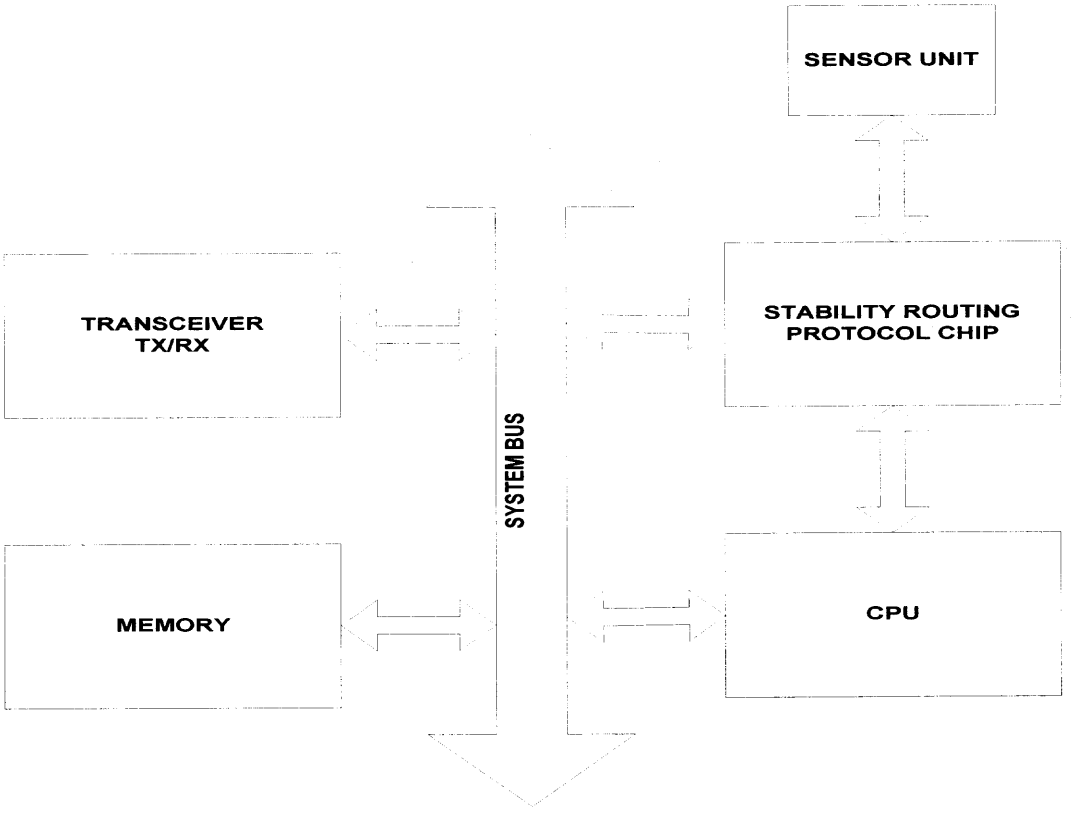

Figure 3.1 Overview of the system.

A transceiver is taken into consideration which is used to transmit and receive data from all the other nodes. The central processing unit (CPU) of the work station is linked

A transceiver is taken into consideration which is used to transmit and receive data from all the other nodes. The central processing unit (CPU) of the work station is linked to the stability routing module which decides the route the packet should take from source to destination. The Stability routing protocol chip module is responsible for the protocol, when it is implemented the routing table will be stored in the memory. Then CPU can utilize the routing information to receive and transmit information to all the other nodes in the network.

3.2 Architecture of the Chip

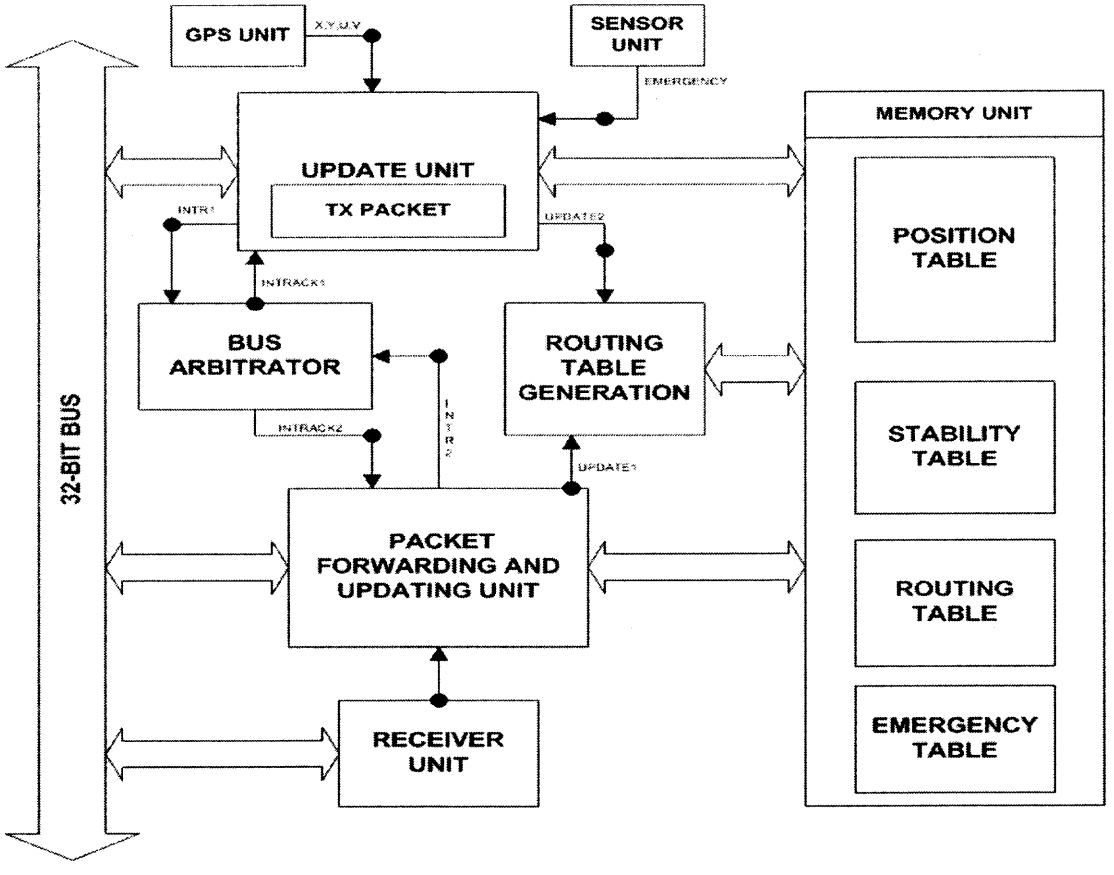

Figure 3.2 Architecture of stability routing chip.

The architecture of the stability routing chip [4] is divided into sub modules as Bus Arbitrator, Memory Unit (MU), Receiver Unit (RU), Update and Transmitter Unit (UTx), Packet Forwarding and Update Unit (PFU), Routing Table Generation Unit (RTU). Figure. 3.2 demonstrates the architecture of the Stability routing chip. The Bus arbitrator selects the one with the highest priority on the system bus based on the priority of the interrupt signal from the Update and Transmitter Unit (UTx) and the Packet Forwarding and Update Unit (PFU). The highest priority is given to the Update and Transmitter Unit (UTx) as it handles the data coming from the Geographical Positioning Unit (GPS), which will give the information regarding the position and velocity of the node in consideration and the Emergency notification data coming from the Sensor unit. This unit sends back an acknowledgement signal to the unit from which it gets the bus request signal. This is done when the request is accepted.

The Memory Unit (MU) has three Look up Tables (LUT). They are Position LUT, Stability LUT, Emergency LUT and the Routing table. The Position LUT has the information of the position parameters X, Y and the velocity parameters U, V of all the other nodes in the network and it is assumed to be constantly updated by the GPS for every 10 seconds.

The Stability LUT has the Stability time of each node with respect to every other node in the network. The Emergency LUT has the sensor information of each and every node in the network.

The Routing Table consists of most stable paths from a particular source node to every other node in the network (Most stable path = the communication path that lasts long without breakup).

The Receiver Unit (RU) checks the type of the packet received. If it is a Data packet then it is sent to the Packet Forwarding and Updating Unit. In the PFU unit the node that receives this packet comes into play. It compares the origination id to its own node id. Incase if its own node id it will discard the packet. If the origination id is different form its own node then the position data is updated in the Position LUT. If the packet is an Emergency packet then it is updated in the Emergency LUT and it is sent to the CPU for Processing.

The Update and Transmitter Unit (UTx) receives position and velocity coordinates from the GPS. These coordinates are compared with the existing coordinates, which are present in the Position LUT. The new coordinates are stored if a change in position and velocity components are detected. If the Update and Transmitter Unit (UTx) receives an emergency indication from the Sensor Unit a packet is formed and then it is transmitted to all the other nodes in the network and it is sent to the CPU for Processing.

When ever a data is updated in the LUD an update signal is sent to the Routing Table Generation Unit then this unit gets the stability information from the Stability LUT and constructs Routing table.

3.3 Design of Updation and Transmitter Module

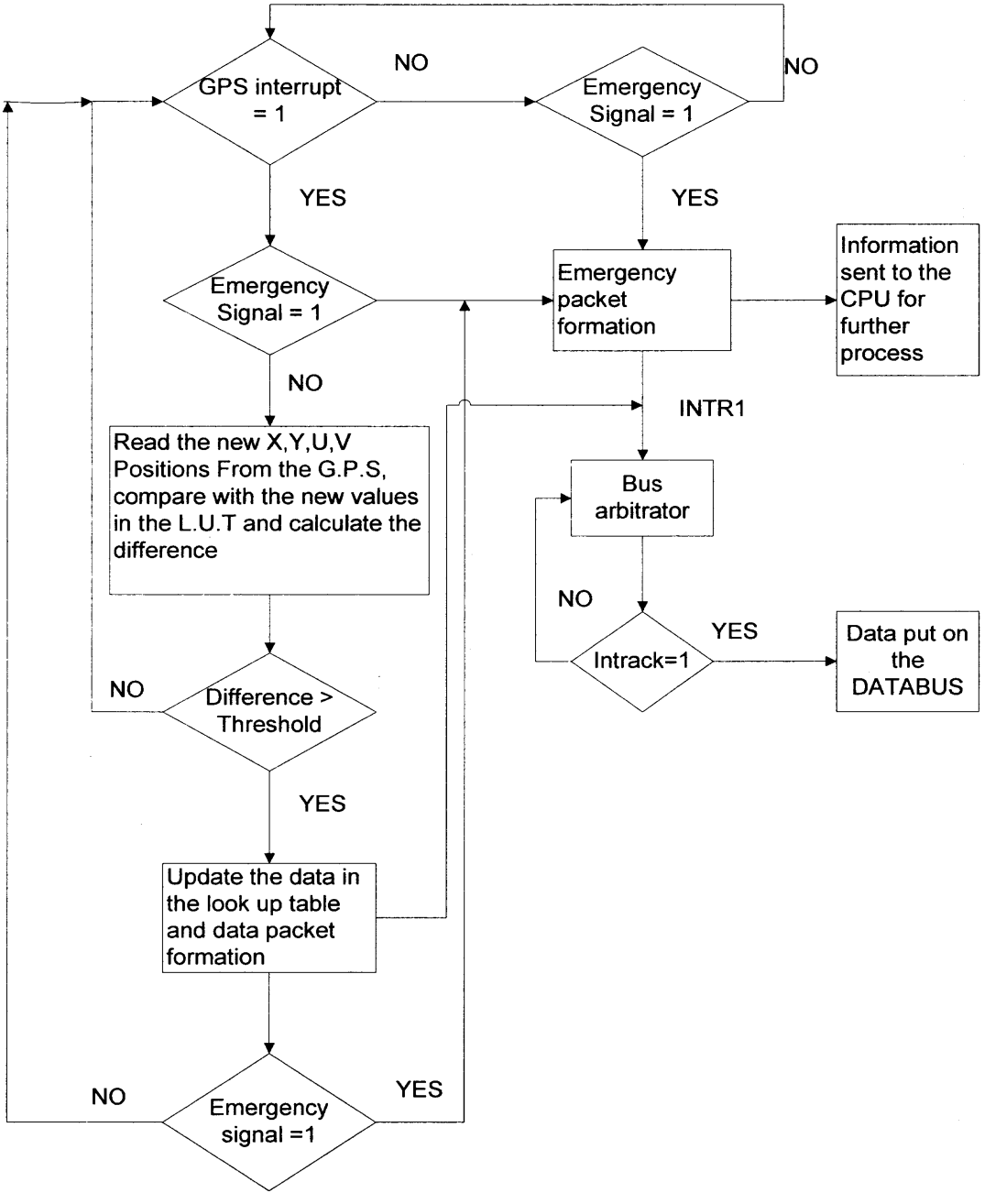

Figure 3.3 Design flow of updation and transmitter module.

Figure. 3.3 illustrates the design of the Updation and Transmitter Module. In this module an interrupt from the GPS Unit is checked which is an indication of change in Position parameters X, Y and the Velocity Parameters U, V. If there is any emergency signal from the Sensor Unit an Emergency Packet is formed. Figure 3.4 shows the Structure of the Emergency Packet.

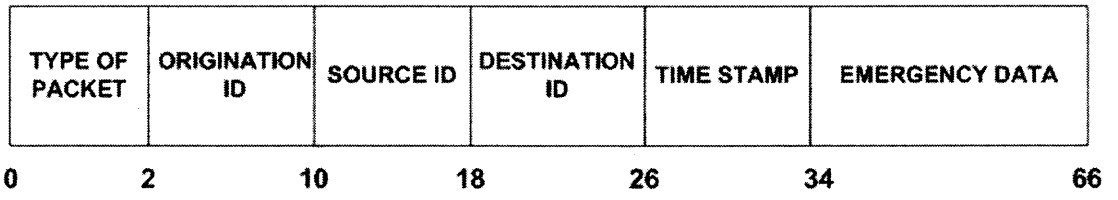

Figure 3.4 Structure of emergency packet.

Where the Type of packet is '01' since it is an emergency packet, Origination ID is same as Node ID, Source ID is same as Node ID , Destination ID = "11111111" , Time Stamp is initiated as '00000001', and the last field Emergency Data is the data that comes from the Sensor Unit. After the packet is formed the information from the Sensor Unit is passed to CPU for further Process. And an interrupt (INTR1) is sent to the Bus Arbitrator for the use of the Bus whenever an interrupt acknowledge (INTRACK1) is received then this packet is put on the Data bus. If there is no Emergency Signal from the Sensor unit then the parameters that are received from the G.P.S Unit are compared with those in the LUT and their difference is calculated if this difference is greater than the threshold values (X = 10 meters, Y = 10 meters, U = 5 m/sec, V = 5 m/sec) then this information is updated in the Position LUT and a Data Packet is formed. Figure 3.5 shows the Structure of the Data Packet.

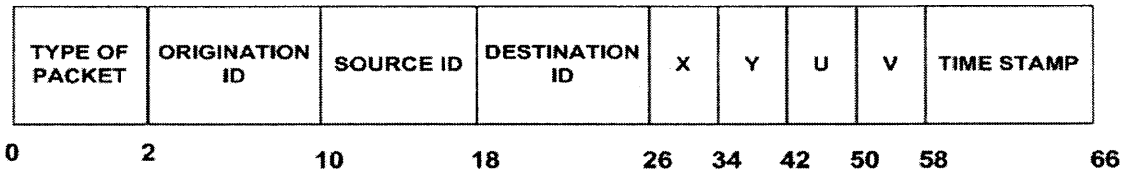

Figure 3.5 Structure of data packet.

Where the Type of packet is '00' since it is an Data packet, Origination ID is same as Node ID, Source ID is same as Node ID, Destination ID = "11111111", X = X Value from G.P.S Unit, Y= Y Value from G.P.S Unit, U = U Value from G.P.S Unit, V = V Value from G.P.S Unit and the Time Stamp is initiated as '00000001'. After the packet is formed then an interrupt (INTR1) is sent to the Bus Arbitrator for the use of the Bus whenever an interrupt acknowledge (INTRACK1) is received then this packet is put on the Data bus. This process is repeated mean while if there is any Emergency Signal detected from the output of the Sensor Unit then the Emergency Process is serviced.

3.4 Design of Receiver Module

Figure. 3.6 illustrates the design of Receiver Module. In this module it checks for the packet type and it forwards it to the Packet Forwarding and Update Unit. When ever a packet comes into this Receiver Unit its information is stored in the buffer and Origination ID is checked if it is same as the Node ID then that particular packet is discarded. If it is not same as the Node ID then the Time Stamp component in the Packet is compared, if the time stamp component in a packet is greater than 8 it is considered that all the nodes in the network have the information of that particular packet and the packet is discarded. It is considered that there are 8 nodes in a network and whenever a

node receives a packet from the other node it increments the time stamp component in the packet.

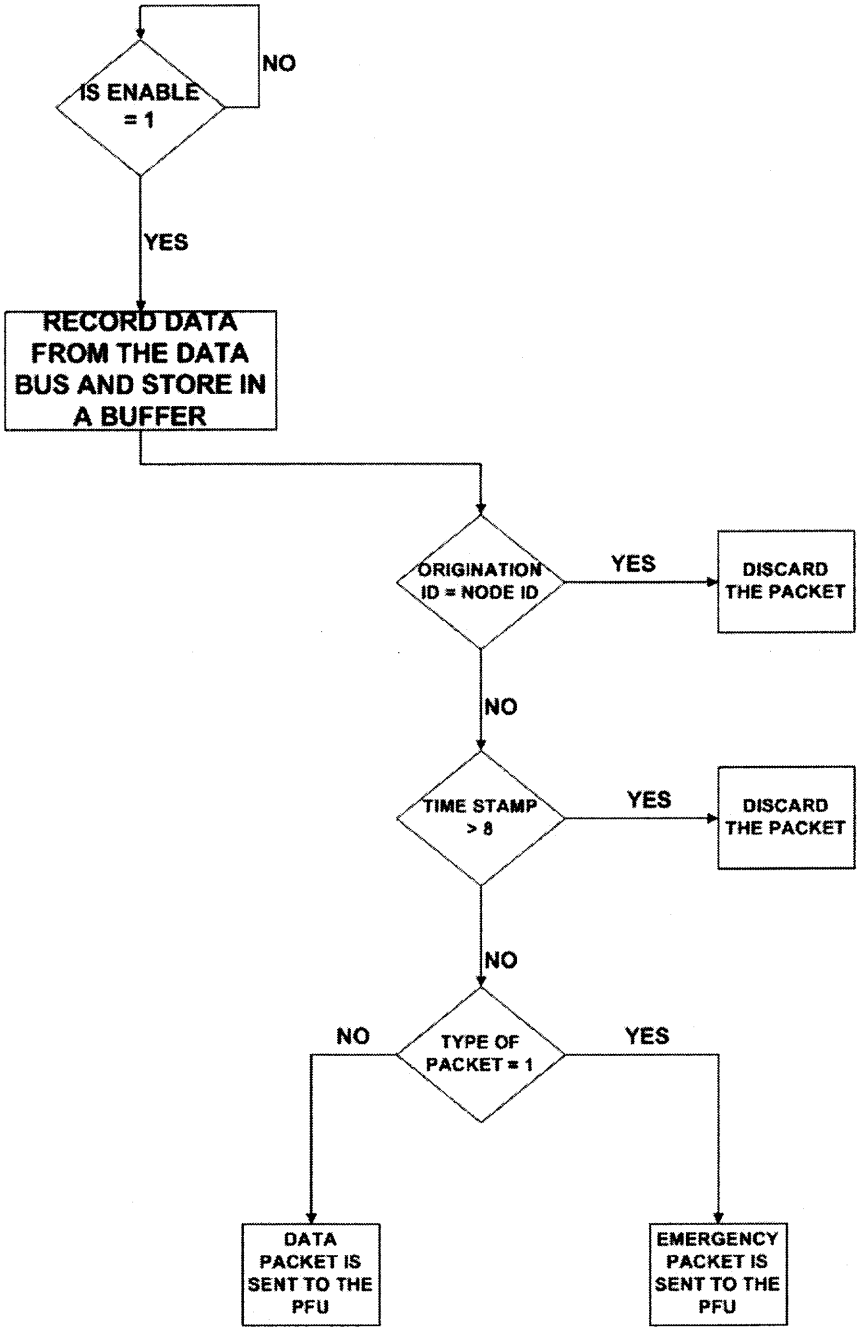

Figure 3.6 Design flow of receiver module.

If the Time Stamp is less than '8' then it is checked for the Type of Packet if the Type of the Packet is '01' then the packet is sent to the Packet Forwarding and Update Unit and is notified that it is an Emergency Packet. If the Type of the Packet is '00' then the packet is sent to the Packet Forwarding and Update Unit and is notified that it is a Data Packet. This process continues whenever it receives an enable indicating that information has come from the other node.

3.5 Design of Packet Forwarding and Updating Module

Figure. 3.7 illustrates the design of Packet Forwarding and Updating Module. In this module after it receives the packet from the receiver module it updates the data in the lookup table and it forwards the packet to other nodes in the network. When the packet comes from the Receiver module it checks whether it is a Data Packet or Emergency Packet if the packet is Emergency Packet then the Origination ID in the packet is taken and the data in updated in the look up table at that corresponding location. Then the information of the Source ID is updated with that of the Node ID and the Time Stamp is incremented once. Then this information is sent to the CPU for further Process, an interrupt (INTR2) is sent to the Bus Arbitrator for the use of the Bus whenever an interrupt acknowledge (INTRACK2) is received then this packet is put on the Data bus.

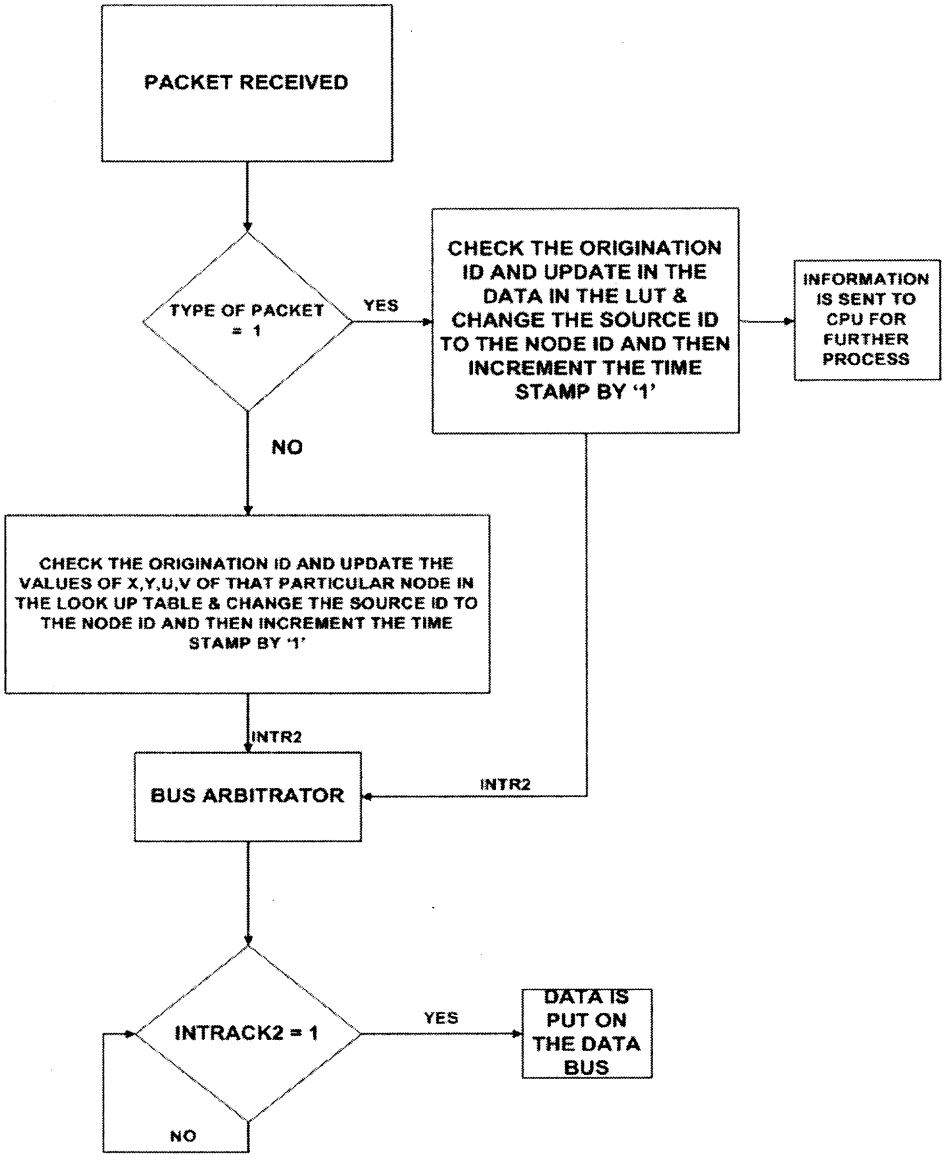

Figure 3.7 Design flow of packet forwarding and updating module.

If the packet is a Data Packet then the Origination ID in the packet is taken and the X, Y, U, V parameters are updated in the Position LUT at the corresponding location. Then the information of the Source ID is updated with that of the Node ID and the Time Stamp

is incremented once. An interrupt (INTR2) is sent to the Bus Arbitrator for the use of the Bus whenever an interrupt acknowledge (INTRACK2) is received then this packet is put on the Data bus. This process continues whenever the Packet Forwarding and Updating Unit gets a packet from the Receiver Unit.

3.6 Design of Routing Table Generation Module

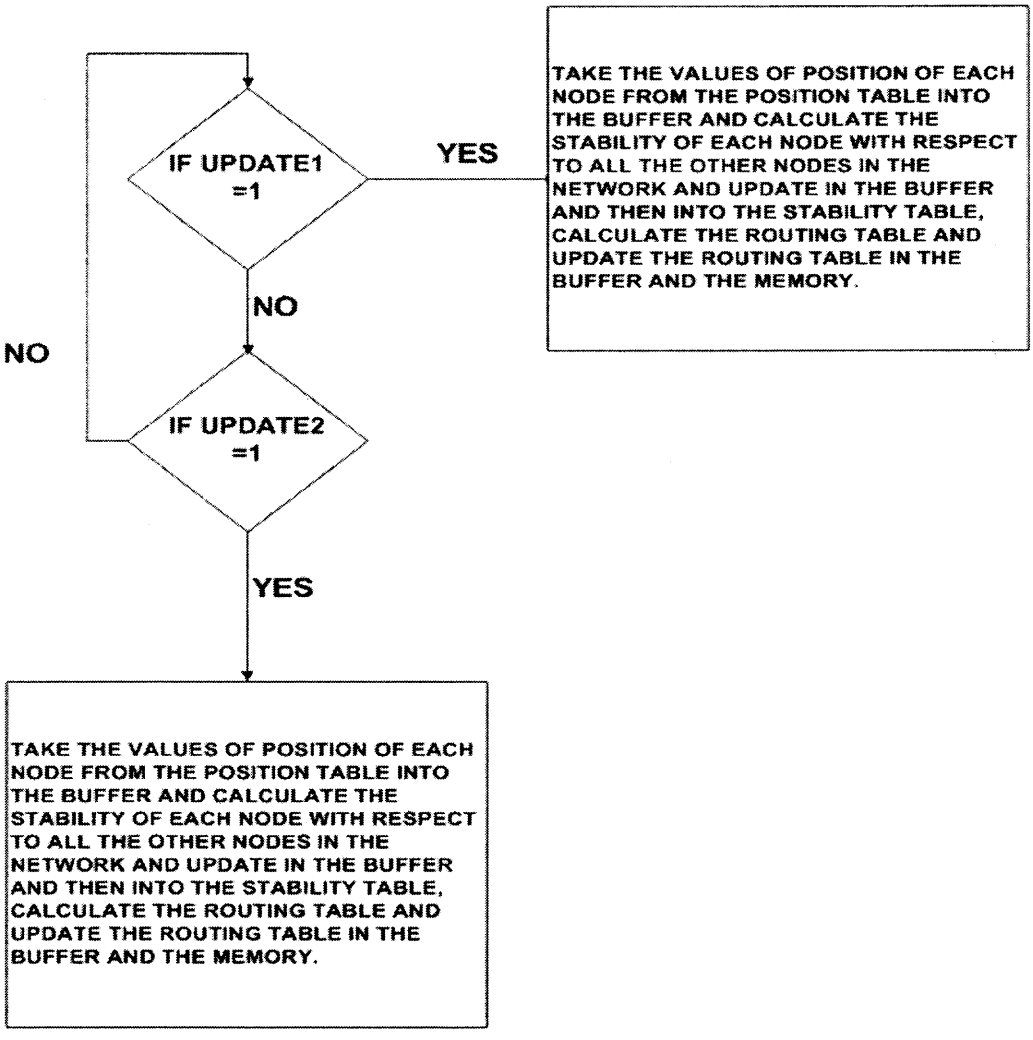

Figure 3.8 Design flow of routing table generation module.

Figure. 3.8 illustrates the design of Routing Table Generation Module. In this module after it receives the update1 signal from the Updation and Packet forwarding module the routing table module gets the position components (X, Y) and the velocity components (U, V) of each node from the position table and calculates the stability of one node to all the other nodes in the network.

3.6.1 Stability Calculation

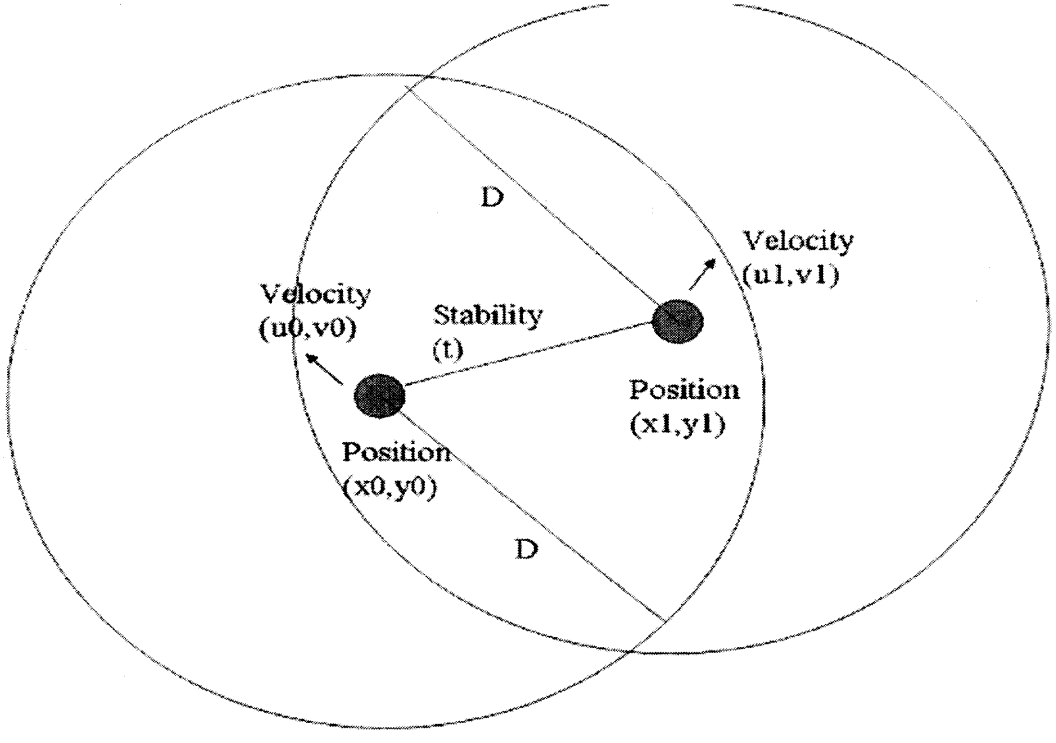

Figure 3.9 Stability calculations between two nodes in a network.

Figure. 3.9 illustrates the calculation of Stability between Two Nodes in the Network. Where the position components of node '0' are (x0, y0) and the velocity components of the node '0' are (u0, v0), the position components of node '1' are (x1, y1) and their velocity components are (u1, v1) and 'D' is the maximum possible transmission distance, Stability ('t') the time until which the node stays within the maximum transmission range with respect to the other node. The maximum possible transmission distance 'D' is given by the below expression where x0, y0, x1, y1, uo, v0, u1, v1 are position and velocity components. $[[(x0+u0*t)-(x1+u1*t)]^2+[(y0+vo*t)-(y1+v1*t)]^2=D^2$ (3.1)

The above expression can be transformed into a quadratic equation $a*t^2+b*t+c=0$ from which the Stability ('t') is calculated. Where $a=(u0-u1)^2+(v0-v1)^2$, $b=2^*[(x0-x1)+(y0-y1)]$ and $c=(x0-x1)^2+(y0-y1)^2+D^2$. The roots of this quadratic equation give the Stability ('t').

3.6.2 Routing Table Generation

After the stability table is updated in the buffer and in the global memory then the Routing Table is constructed from each node to all destinations using the shortest path algorithms. The Dijkstra`s algorithm[6] and the Bellman-Ford's algorithm[6] are some of the shortest path algorithms, the Dijkstra's algorithm, which is used to find the shortest path between the nodes is used, is transformed to find the path with highest stability. The Dijkstra's algorithm solves the problem of finding the shortest path from a given source node to all the other destinations in the network of nodes. Initially in a network nodes each node and its cost with respect to its adjacent node is known where cost is the label assigned between two nodes in the network. The label can be function of distance, band

label assigned between two nodes in the network. The label can be function of distance, band width, time, average traffic, stability, communication cost, mean average length, measured delay [6]. In the Figure. 3.10 to explain the working of Dijkstra`s algorithm the costs between the nodes is considered as a function of distance, the source node from which the shortest path is to be found is labeled as '0' and the costs of all the other nodes are labeled as '∞'. Then the node with the lowest cost is chosen and the nodes adjacent to the chosen node are relaxed where relaxation is a process of assigning cost to a node as a function of sum of previous costs, the predecessors for all the relaxed nodes are updated.

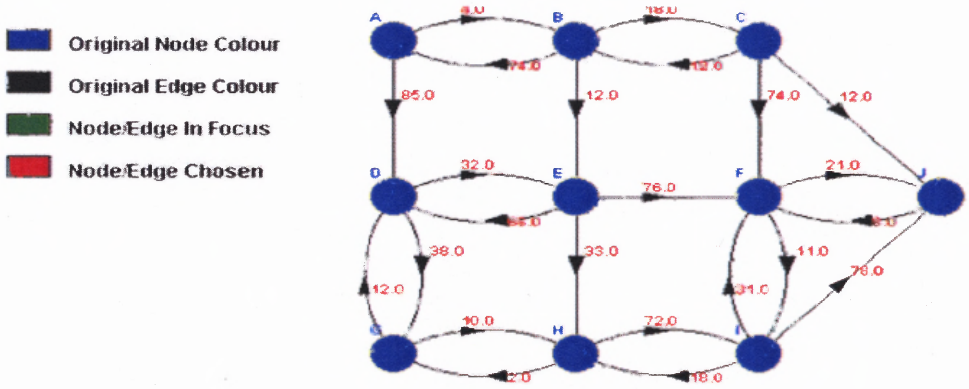

Figure 3.10 Network of nodes with their costs with respect to its adjacent nodes.

Figure. 3.10 illustrates a network with its nodes and their costs with respect to the other adjacent nodes where A,B,C,D,E,F,G,H,I,J are the nodes in network.

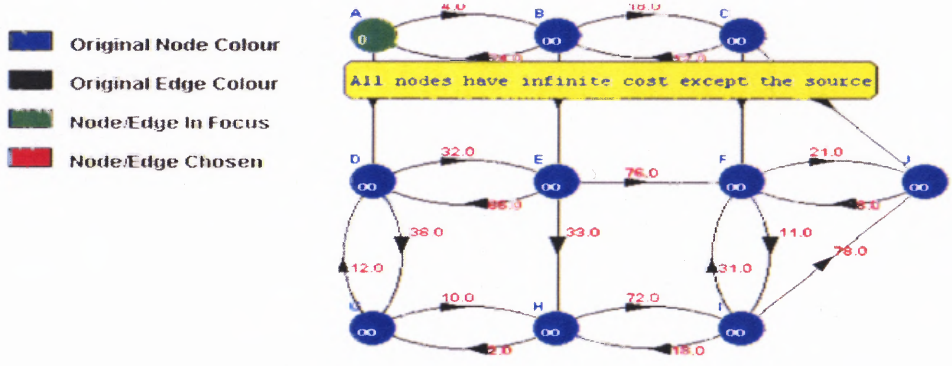

Figure 3.11 Network of nodes with source node and its adjacent nodes.

Figure. 3.11 illustrates a network of nodes where 'A' is the source node whose cost is labeled as zero and all the other adjacent nodes are labeled as infinity.

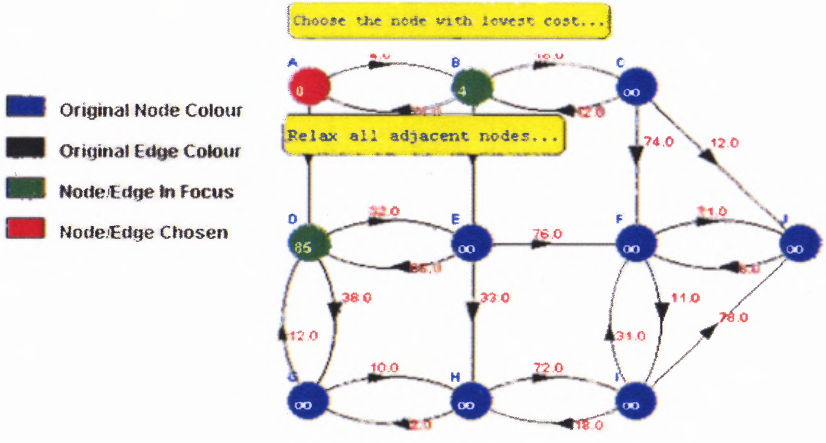

Figure 3.12 Network of nodes with the node having lowest cost.

Figure. 3.12 illustrates a network of nodes where 'A' is chosen, the node with lowest cost and all adjacent nodes to the source node are relaxed.

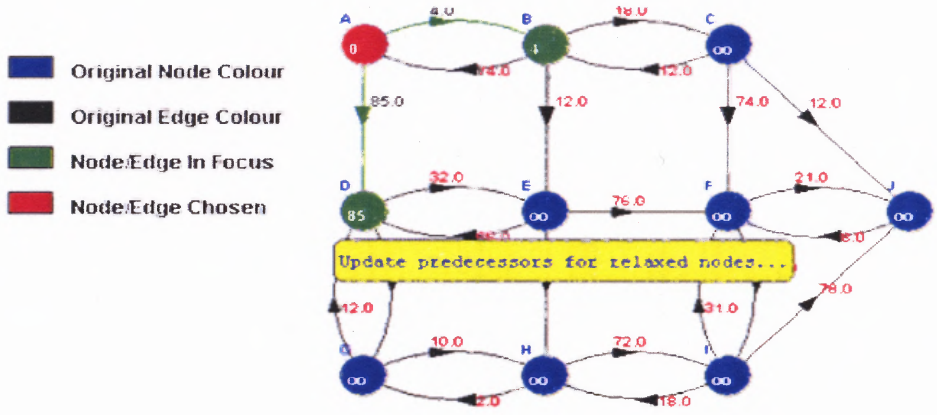

Figure 3.13 Network of nodes with the updated Predecessors.

Figure. 3.13 illustrates the updation of predecessors for all the relaxed nodes. This procedure is repeated until all the nodes in the network are chosen. This gives the shortest path from the source node to all the other nodes in the network.

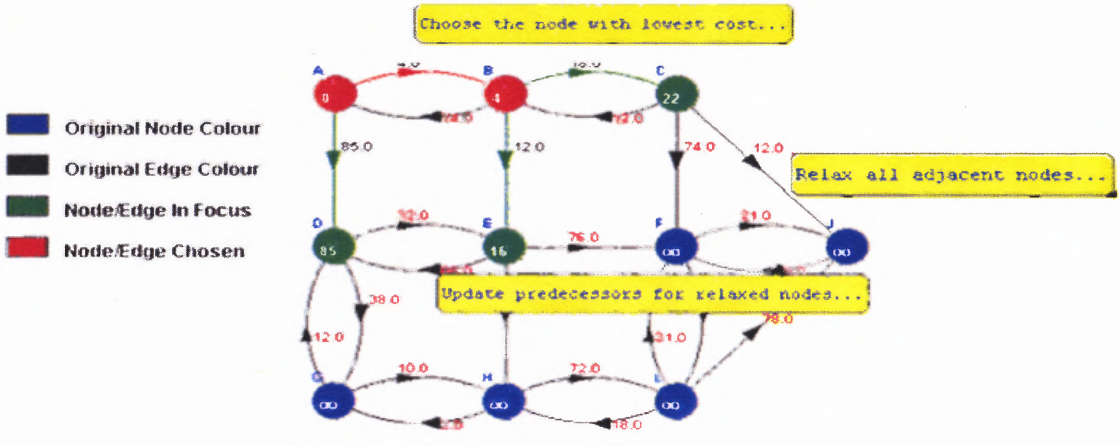

Figure 3.14 Network of nodes with the second node chosen.

Figure. 3.14 illustrate the repeated procedure where the second node 'B' is chosen.

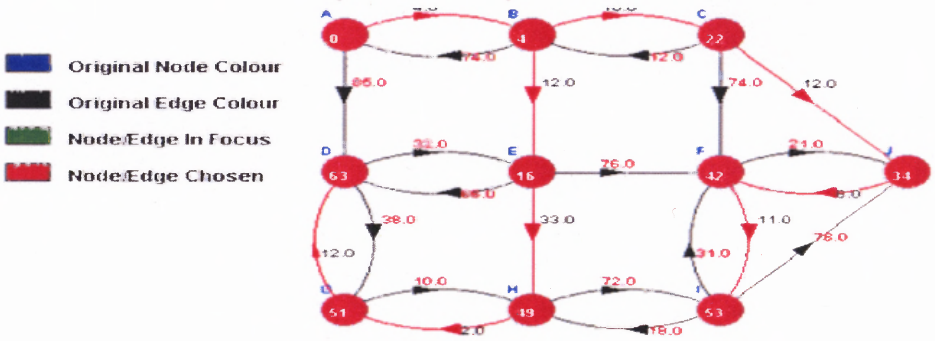

Figure 3.15 Network of nodes with all chosen nodes.

Figure. 3.15 shows a network where all its nodes are chosen and the shortest path from the source node to all nodes in the network. According to stability routing algorithm the routing table is constructed using the shortest path algorithm with maximum stability as criteria, the Dijkstra`s algorithm is transformed to find the shortest path with maximum stability.

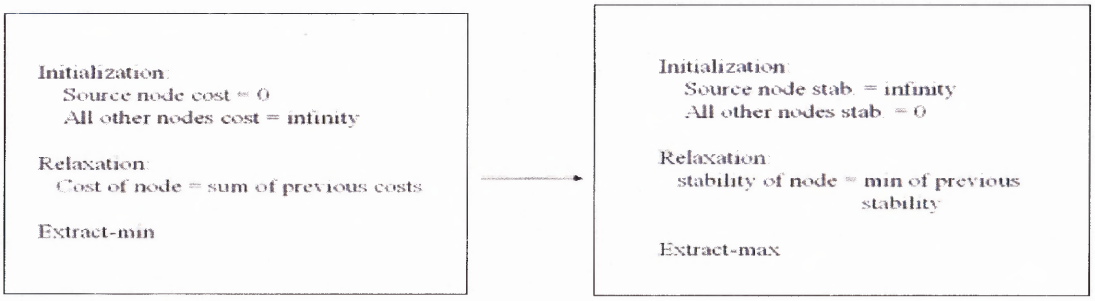

Figure 3.16 Transformation of Dijkstra's algorithm to find the most stable path.

Figure. 3.16 illustrates the transformation of the Dijkstra's algorithm to find the path with maximum stability where the source node is initially labeled as infinity and all the other nodes are zero, the relaxation as the minimum of the previous stabilities and the

node with the maximum stability is considered. The paths from each node to all the other nodes in the network are calculated depending on the transformed algorithm and are updated in Routing Table of the Memory unit.

The hardware realization of this particular algorithm is considered to be efficient and is faster compared to the software implementation. As this particular architecture is application specific further enhancements as regards to implementing the protocol is not necessary, while software implementation has an advantage of modifications for other applications at the expense of speed. Hence hardware realization is suitable for implementing this protocol.

CHAPTER 4

THE VHDL SIMULATION

The high level simulation for the individual blocks of the architecture as discussed in the previous chapter is simulated and tested in VHDL. This basic architectural module underlying the behavioral process is divided into 3 units Flooding Unit, Stability Unit and the Routing table Generation Unit, where the Flooding Unit is integrated unit of Update Transmitter Unit & Packet Forwarding Update Unit and Receiver Unit. Each unit output is fed as an input to the other unit. The flooding unit is invoked by 3 different inputs namely gps_intr, when the gps information is delivered to the node, broadcast_intr, when the information about the other nodes is delivered to the node and sensorsignal_ready, when the sensor information is delivered to the node. These three inputs can occur individually or simultaneously.

A Subnet of 8 nodes in an ad-hoc network which can be scaled to a large network is considered in which the nodes are randomly placed in an XY plane such that each node can communicate at least with one other node. Each node has a stability routing chip and its own GPS unit and sensor unit, each unit gets the position and velocity components from the GPS unit. A random node is considered and the functionality of the stability routing chip is checked for this particular node. The initial position and velocity components for this node are assumed as (16, 20) and (4, 4) respectively, the position and velocity components for the remaining '7' nodes in the network are assumed for simulation.

As can be seen from Figure. 4.1, gps_intr is '1' at 0ns, the other two inputs broadcast_intr is '0' and the sensorsignal_ready is '0' which means that there is only change in the position components X,Y or the velocity components U,V. The new position components are x_position is '00010000' (16), y_position is '00010100'(20) and the velocity components are u_velocity is '00000100'(4) and the v_velocity is '00000100'(4) are obtained from the gps unit at 0ns.This position and velocity components are stored in the gpsin_input_buff_1 and gpsin_input_buff_2 and then these position and velocity components are broadcasted to all the other nodes in the network. A packet 'broadcast_output' is formed it has various fields along with the x_position, y_position, u_velocity, v_velocity parameters, type_packet is '00' which indicates that is not an emergency packet, original_id is '00000001' which indicates that origination of the packet is node1 and source_id is '00000001' which indicates that source of the packet is node1 and desitation_id is '11111111' which indicates that packet has to be sent to all the other nodes in the network, timestamp is '00000000' whenever a new packet is formed the timestamp is made zero and whenever the packet is received by another node and the sent to the next node the timestamp is incremented by '1'. When the time stamp is greater than '8' it is considered that all the information is received by all the nodes in the network. At 10ns it can be seen that the packet is formed, the rfinput_outputbuffer_data(1) is

00000000	00000100	00000100	00010100	00010000	11111111	00000001	00000001	00
Time_stamp	V_velocity	U_velocity	Y_position	X_position	Destination_id	Source_id	Original_id	Type_ packet

this is the information that is to be sent to the other nodes in the network. The gps_intr is zero at 20nsec which indicates that the change in the position and velocity components are acknowledged.

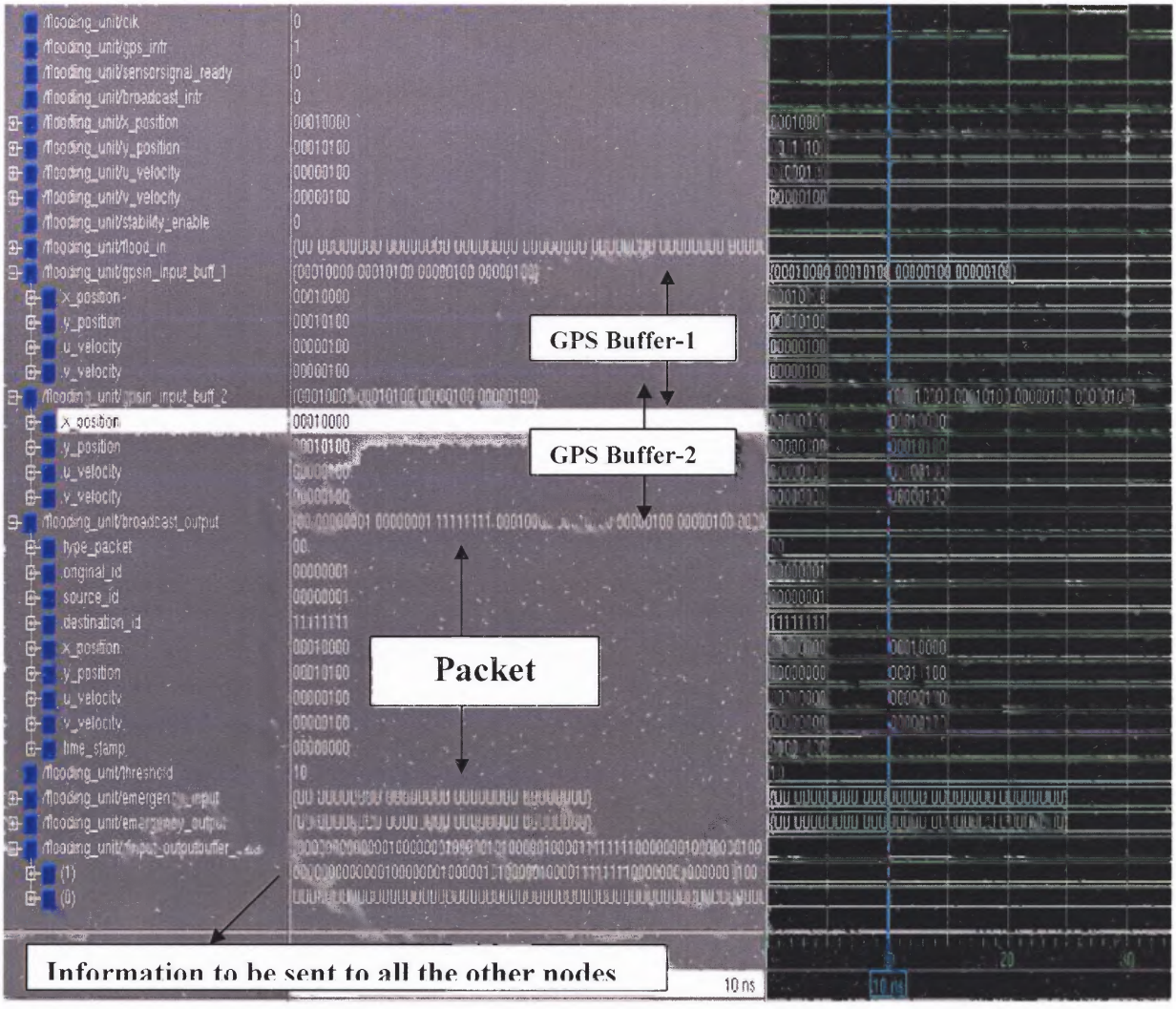

Figure 4.1 High-level timing simulations for the mentioned sequential tasks.

At 30ns gps_intr is '1 indicating that there is a change in the position or velocity components now the new positions that are obtained form the GPS are stored in the gpsin_input_buff_1 and the old positions are stored in gpsin_input_buff_2. It can be seen at 30ns in Figure. 4.2.

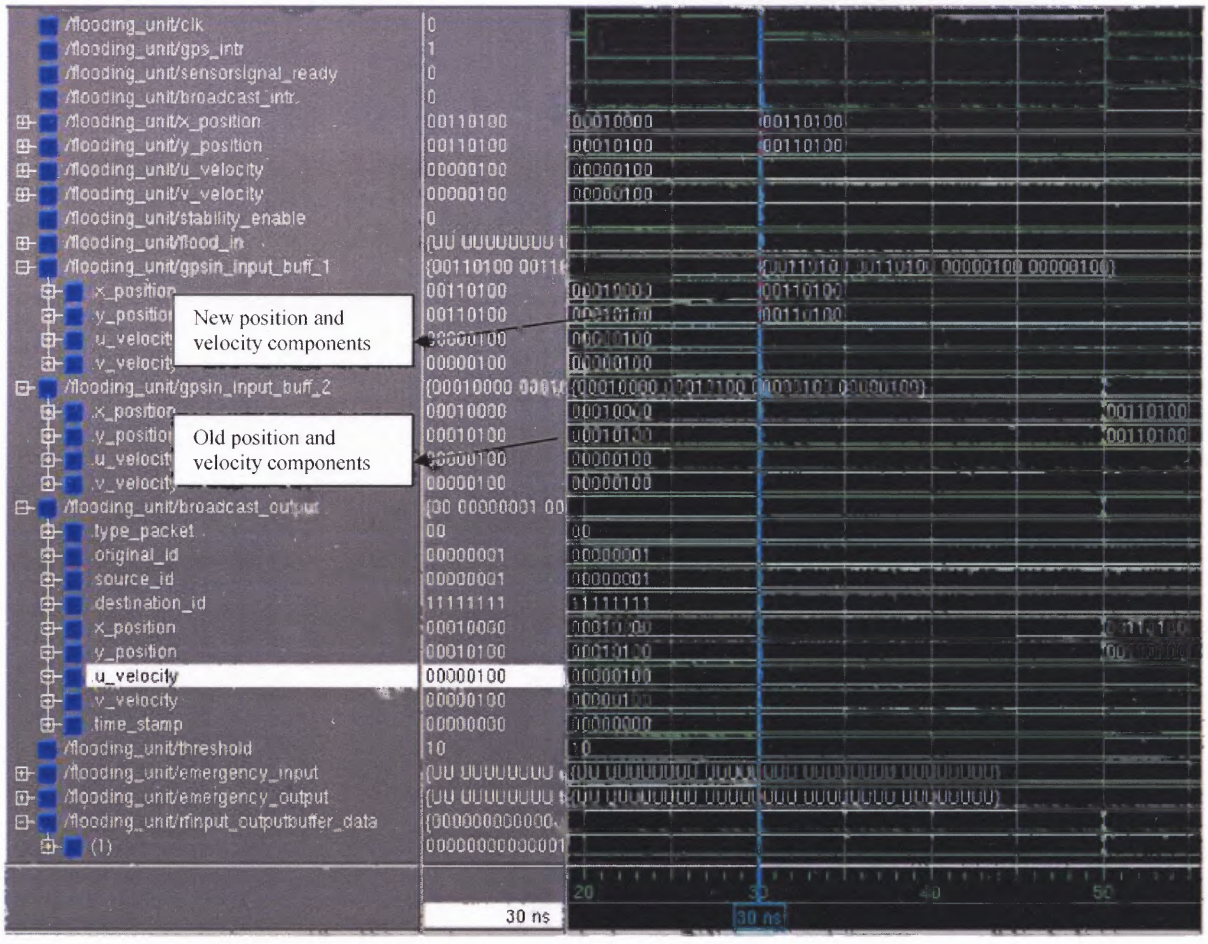

Figure 4.2 Timing simulation continued after figure. 4.1.

The position and velocity components in the both buffers are compared and if the change in any of these components is greater than threshold if the node has moved from the transmittable region of one particular node to another node it is considered that it has exceeded the threshold level. Then the new position parameters are updated in the position table and then broadcasted to all the other nodes in the network. As difference in the x_position and y_position components are greater than the threshold it can be seen in

the Figure. 4.3 At 50ns the broadcast_output packet is formed with the updated parameters and the rfinput_outputbuffer_data(1) is

00000000	00000100	00000100	00110100	00110100	11111111	00000001	00000001	00
Time_stamp	V_velocity	U_velocity	Y_position	X_position	Destination_id	Source_id	Original_id	Type_ packet

this is the information that is to be sent to all the other nodes in the network..

Figure 4.3 Timing simulation continued after figure. 4.2.

At 50ns the gps_intr is '0', sensor ready is '0' and the broadcast_intr is '1' which indicates that it is a broadcast packet, a packet containing the information of the other nodes, rfinput_outputbuffer_data(0) contains the received information from the other nodes.

00000110	00010010	00011110	00010010	00011100	11111111	00000100	00001000	00
Time_stamp	V_velocity	U_velocity	Y_position	X_position	Destination_id	Source_id	Original_id	Type_ packet

The first two bits are type_packet '00' which indicate that the packet is not an emergency packet and the last eight bits are time_stamp '00000110' '6' is checked if the time stamp is greater than '8' then the information is discarded. The next 8 bits next to the type_packet are the original_id '00001000' which indicates that the position and the velocity components are corresponding to node '8', the next 8 bits '00000100' source_id which indicates that the packet has come from node '4' and the next '8' bits '11111111' destination_id indicates that the information has to be sent to all the nodes of the network, the next 8 bits '00011100' is x_position component, the next 8 bits '00010010' is y_position component, the next 8 bits '00011110' is u_velocity component and the next '8' bits are v_velocity component of the node '8'. As the time_stamp is less than '8' and the original_id is not same as this nodeid this information should be updated in the position table and source_id is updated as '00000001' and the time_stamp is incremented by one '00000111'and broadcasted again, and the stability_enable is made '1' indicating the stability unit that the positions have been changed. The information that should be sent to other nodes in the network can be seen in rfinput_outputbuffer_data(1) at 80ns in Figure. 4.4.

00000111	00010010	00011110	00010010	00011100	11111111	00000001	00001000	00
Time_stamp	V_velocity	U_velocity	Y_position	X_position	Destination_id	Source_id	Original_id	Type_ packet

00000111	00010010	00011110	00010010	00011100	11111111	00000001	00001000	00
Time_stamp	V_velocity	U_velocity	Y_position	X_position	Destination_id	Source_id	Original_id	Type_ packet

it can be seen that the time stamp is incremented by '1' and the source_id is updated.

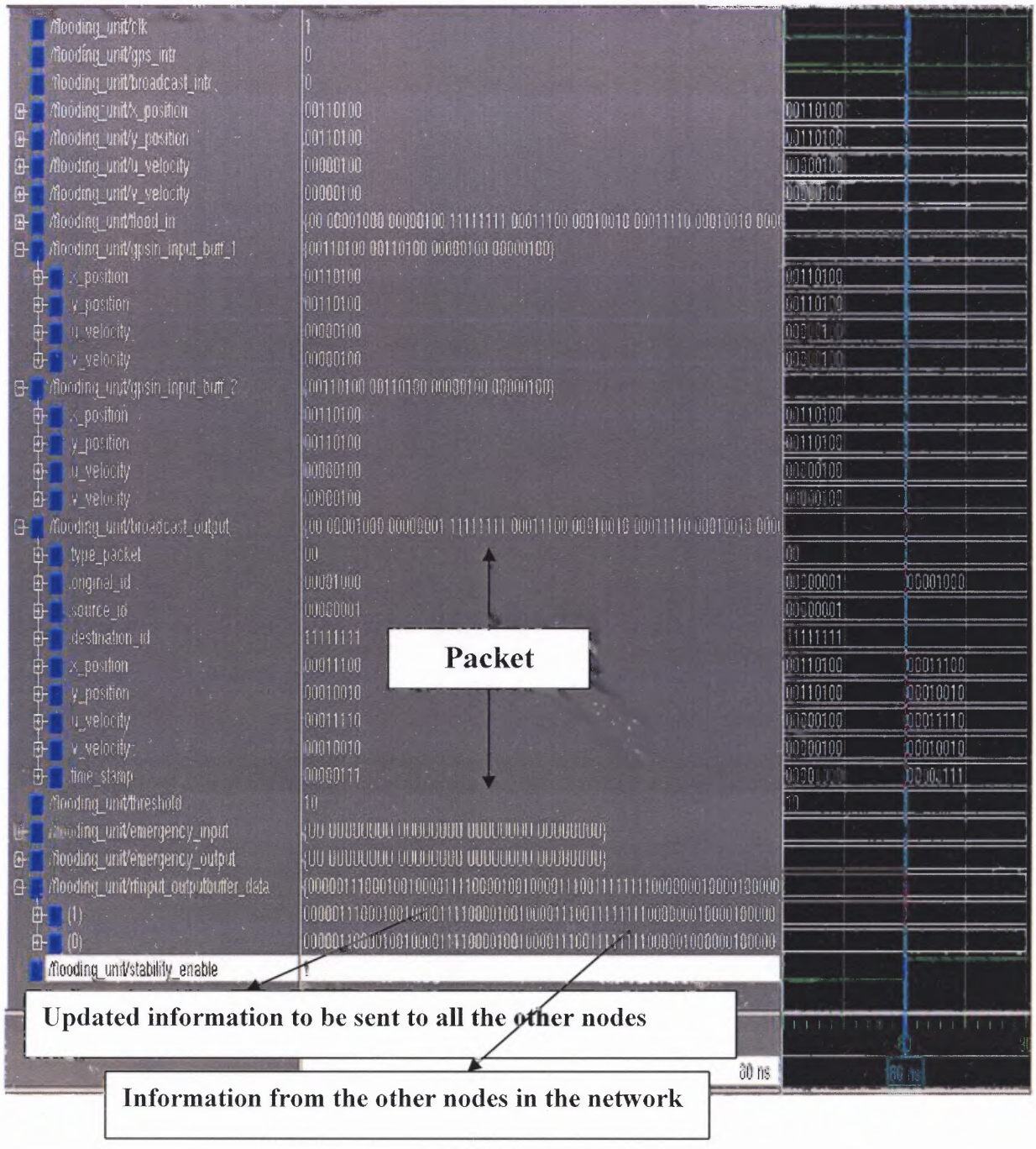

Figure 4.4 Timing simulation continued after figure. 4.3.

At 90ns when both the gps_intr and broadcast_intr are '1' which indicates a change in either the position and velocity components of this particular node and a broadcast packet which has the information of the other nodes have occurred simultaneously in this case the broadcast_intr has a higher priority than the gps_intr so the data from rfinput_outputbuffer_data(0) is processed first.

The rfinput_outputbuffer_data(0) has the received information of the other nodes,

00000000000000000000000000000000	00000100	11111111	00000110	00000111	01
Sensor information	Time_stamp	Destination_id	Source_id	Original_id	Type_ packet

the first two bits are type_packet '01' which indicate that the packet is an emergency packet and the next 8 bits are the original_id '00000111' which indicates that the emergency information corresponds to node '7', the next 8 bits '00000110' source_id which indicates that the packet has come from node '6' and the next '8' bits '11111111' destination_id indicates that the information has to be sent to all the nodes of the network, the next eight bits are time_stamp '00000100' '4' is checked if the time stamp is greater than '8' then the information is discarded, the last 32 bits are left free which can be used for sending the sensor information. As the time_stamp is less than '8' and the original_id is not same as this nodeid the source_id is updated as '00000001' and the time_stamp is incremented by one '00000101'and broadcasted again, The information to broadcasted can be seen in rfinput_outputbuffer_data(1) at 120ns in Figure. 4.5. Then after the broadcast_intr is processed the gps_intr is processed. The new positions are stored in the gpsin_input_buff_1 and the old positions are stored in gpsin_input_buff_2, The position and velocity components in the both buffers are compared and if the change in any of these components is greater than threshold then the new position parameters are updated

these components is greater than threshold then the new position parameters are updated in the position table and then broadcasted to all the other nodes in the network. As difference in the x_position and y_position components are greater than the threshold the broadcast_output packet is formed with the updated parameters and the rfinput_outputbuffer_data(1) has the information which can be seen at 130ns in Figure. 4.6. The stability_enable is made '1' indicating the stability unit that the positions have been changed.

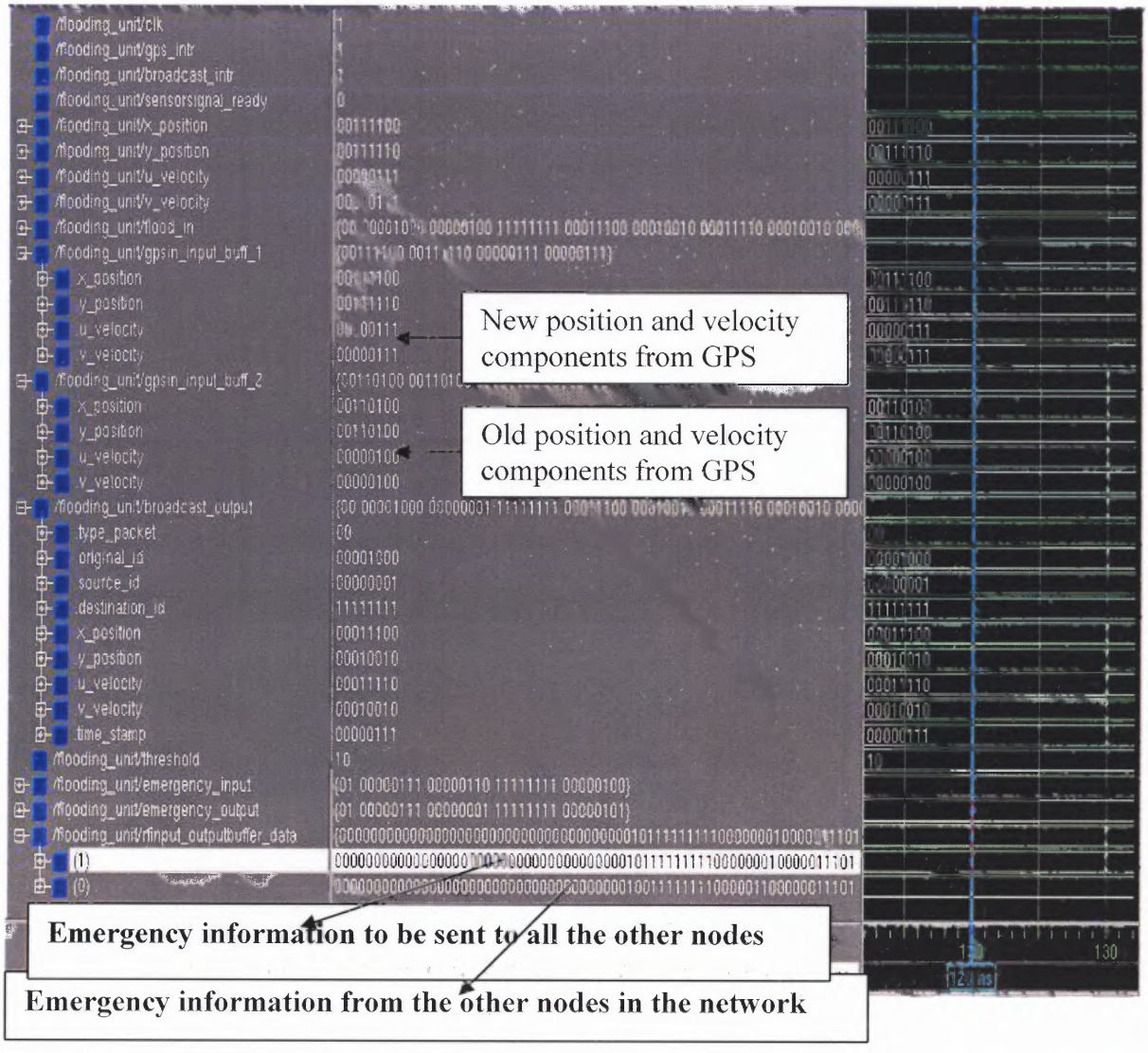

Figure 4.5 Timing simulation continued after figure. 4.4.

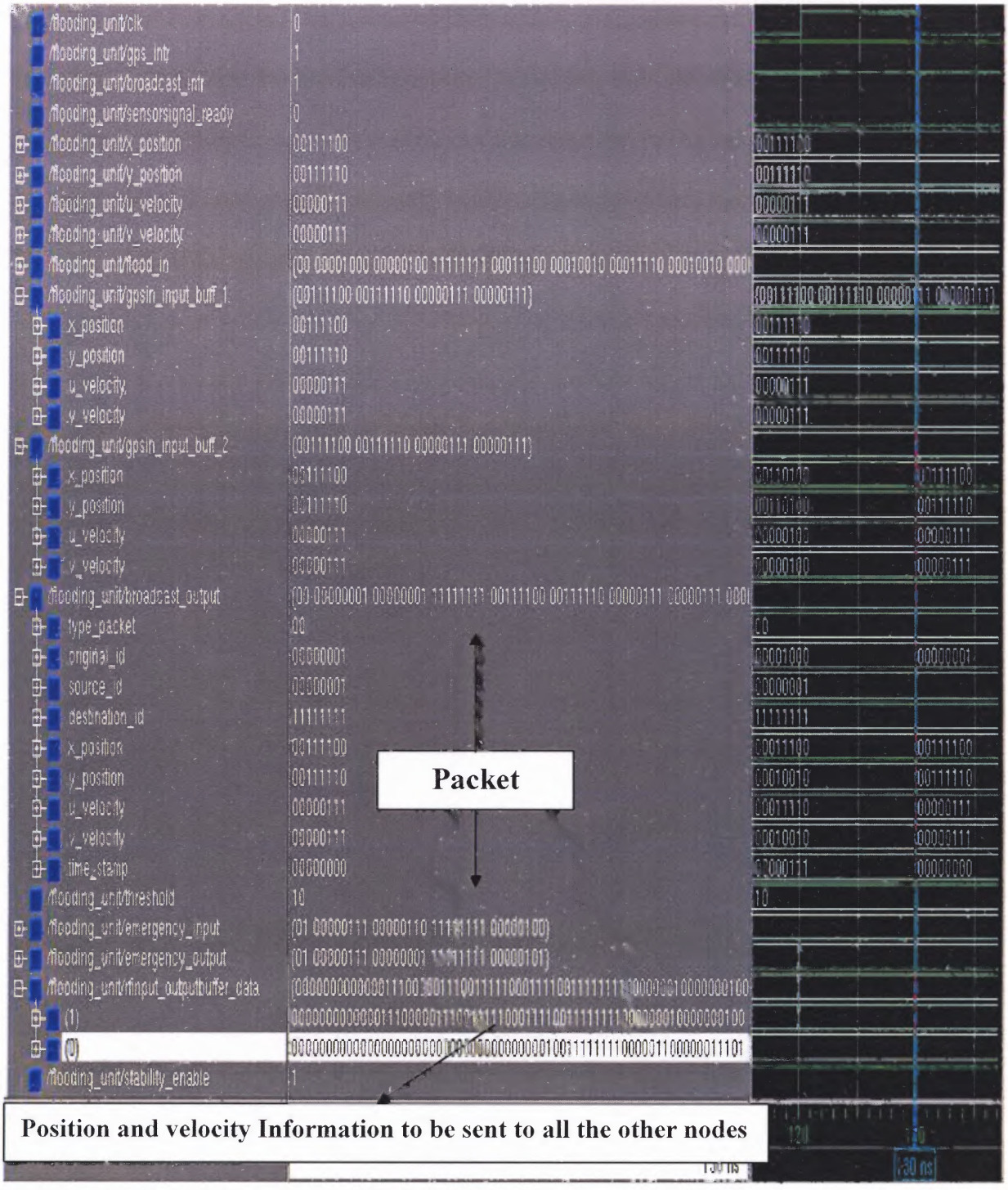

Figure 4.6 Timing simulation continued after figure. 4.5.

At 180ns when both the gps_intr and sensorsignal_ready goes high from a low which indicates a change in either the position and velocity components of this particular node and a emergency signal from the sensor of this node have occurred simultaneously in this case the sensorsignal_ready has a higher priority than the gps_intr. When the sensorsignal_ready occurs an emergency packet has to be formed where type_packet is '01'idicating an emergency packet, original_id is '00000001', source_id is '00000001' and desitination_id is '11111111' which indicates that the information has to be sent to all the nodes in the network, the remaining bits are left which can be used for sending some sensor information. The information that should be sent to all the nodes in the network can be seen in rfinput_outputbuffer_data(1)

00000000000000000000000000000000	00000000	11111111	00000001	00000001	01
Sensor information	Time_stamp	Destination_id	Source_id	Original_id	Type_ packet

at 180ns in Figure. 4.7.

Then after the sensorsignal_ready is processed the gps_intr is processed. The new positions are stored in the gpsin_input_buff_1 and the old positions are stored in gpsin_input_buff_2, The position and velocity components in the both buffers are compared and if the change in any of these components is greater than threshold then the new position parameters are updated in the position table and then broadcasted to all the other nodes in the network. As difference in the x_position and y_position components are greater than the threshold the broadcast_output packet is formed with the updated parameters and the rfinput_outputbuffer_data(1) has the information which can be seen at 210ns in Figure. 4.8. The stability_enable is made '1' indicating the stability unit that the positions have been changed. The gps_intr and sensorsignal_ready becomes '0' at 210ns

which indicates that the sensor information and the change in the position or velocity components are acknowledged.

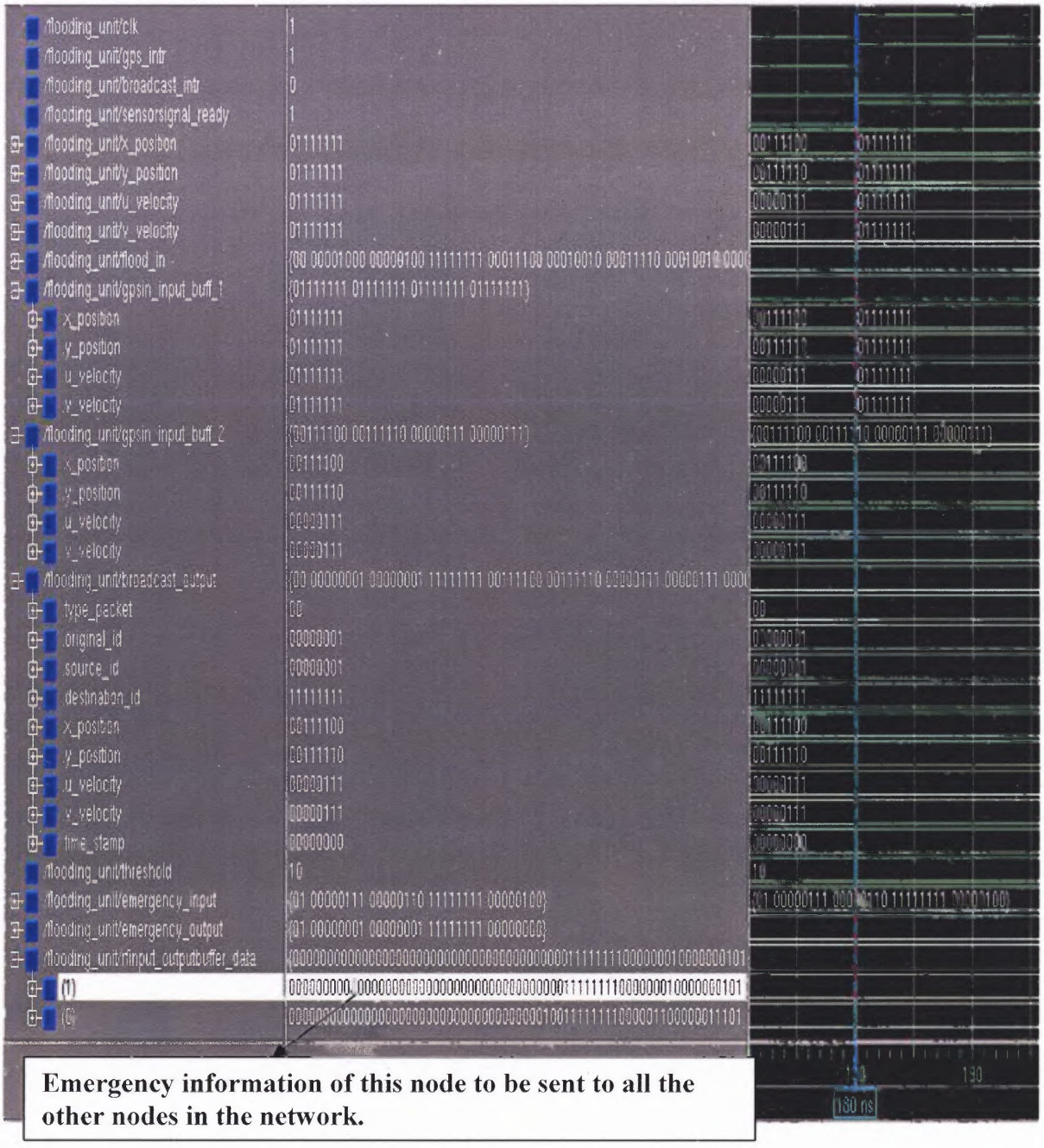

Figure 4.7 Timing simulation continued after figure. 4.6.

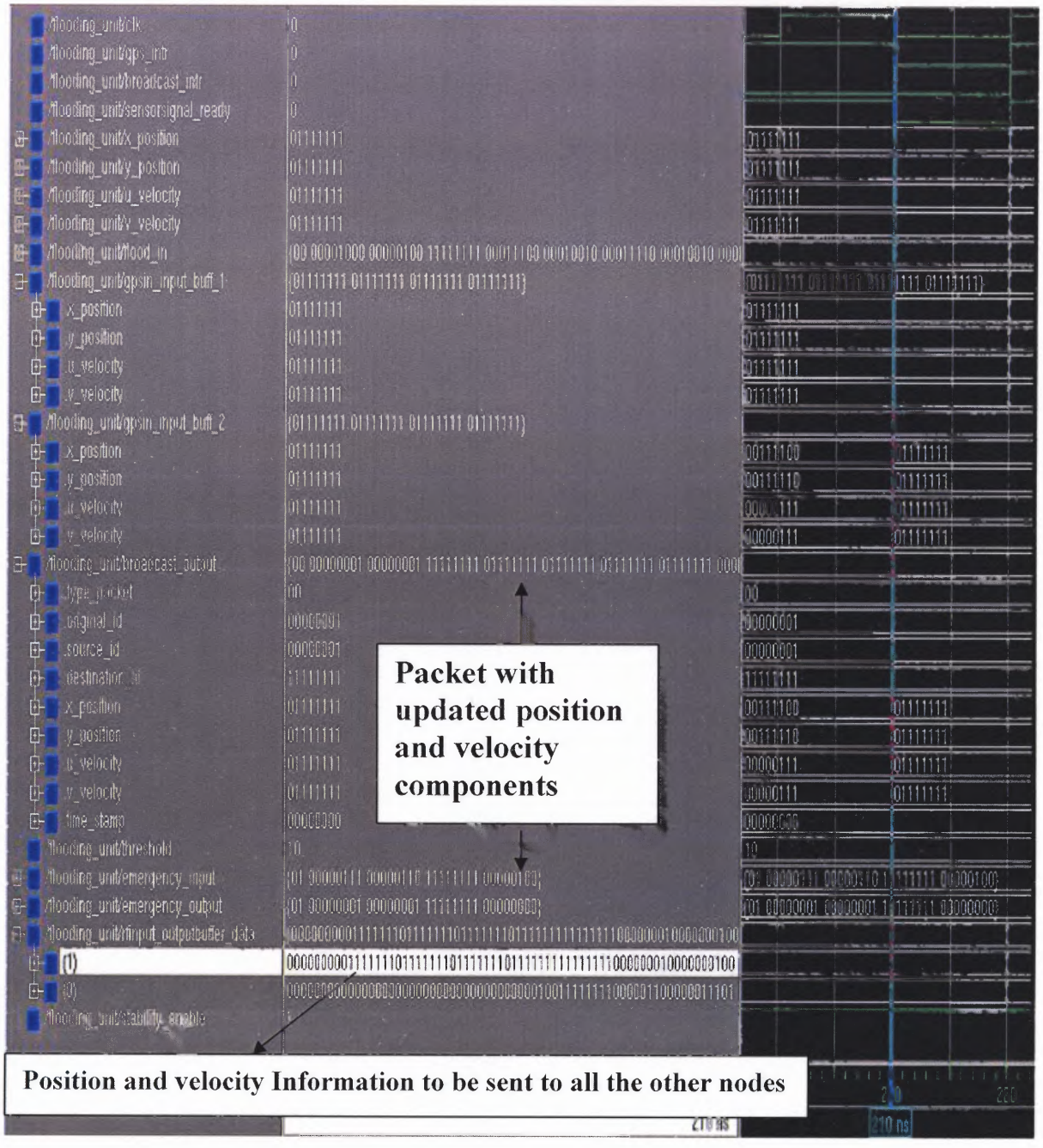

Figure 4.8 Timing simulation continued after figure. 4.7.

At 220ns when the gps_intr and sensorsignal_ready and broadcast_intr are '1' which indicates a change in either the position and velocity components of this particular node, emergency signal from the sensor of this node and a broadcast packet which has the

information of the other nodes have occurred simultaneously. sensorsignal_ready has a higher priority than broadcast_intr, broadcast_intr has a higher priority than the gps_intr. When the sensorsignal_ready occurs an emergency packet has to be formed where type_packet is '01'idicating an emergency packet, original_id is '00000001', source_id is '00000001' and desitination_id is '11111111' which indicates that the information has to be sent to all the nodes in the network, the remaining bits are left which can be used for sending some sensor information. The information that should be sent to all the nodes in the network can be seen in rfinput_outputbuffer_data(1)

00000000000000000000000000000000	00000000	11111111	00000001	00000001	01
Sensor information	Time_stamp	Destination_id	Source_id	Original_id	Type_ packet

at 220ns in Figure. 4.9.then after the sensorsignal_ready is processed the broadcast_intr is processed so the data from rfinput_outputbuffer_data(0) is processed.

The rfinput_outputbuffer_data(0) contains the received information from the other nodes.

00000001	00000001	00000001	00000001	00000100	11111111	00000110	00000111	00
Time_stamp	V_velocity	U_velocity	Y_position	X_position	Destination_id	Source_id	Original_id	Type_ packet

the first two bits are type_packet '00' which indicate that the packet is not an emergency packet and the last eight bits are time_stamp '00000001' '1' is checked if the time stamp is greater than '8' then the information is discarded. The next 8 bits next to the type_packet are the original_id '00000111' which indicates that the position and the velocity components are corresponding to node '7', the next 8 bits '00000110' source_id which indicates that the packet has come from node '6' and the next '8' bits '11111111' destination_id indicates that the information has to be sent to all the nodes of the network,

the next 8 bits '00000100' is x_position component, the next 8 bits '00000001' is y_position component, the next 8 bits '00000001' is u_velocity component and the next 8 bits '00000001' are v_velocity component of the node '8'. As the time_stamp is less than '8' and the original_id is not same as this nodeid this information should be updated in the position table and source_id is updated as '00000001' and the time_stamp is incremented by one '00000010'and broadcasted again, and the rfinput_outputbuffer_data(1)

00000010	00000001	00000001	00000001	00000100	11111111	00000001	00000111	00
Time_stamp	V_velocity	U_velocity	Y_position	X_position	Destination_id	Source_id	Original_id	Type_ packet

this is the information that should be sent to all other nodes in the network which can be seen at 240ns in Figure. 4.10. Then after the broadcast_intr is processed the gps_intr is processed. The new positions are stored in the gpsin_input_buff_1 and the old positions are stored in gpsin_input_buff_2, The position and velocity components in the both buffers are compared and if the change in any of these components is greater than threshold then the new position parameters are updated in the position table and then broadcasted to all the other nodes in the network. As difference in the x _position and y_position components are greater than the threshold '10' the broadcast_output packet is formed with the updated parameters and the rfinput_outputbuffer_data(1)

00000000	00000111	00000111	00111110	00111110	11111111	00000001	00000001	00
Time_stamp	V_velocity	U_velocity	Y_position	X_position	Destination_id	Source_id	Original_id	Type_ packet

this is the information that should be sent to all the other nodes in the network which can be seen at 250ns in Figure. 4.11. The stability_enable is made '1' indicating the stability unit that the positions have been changed.

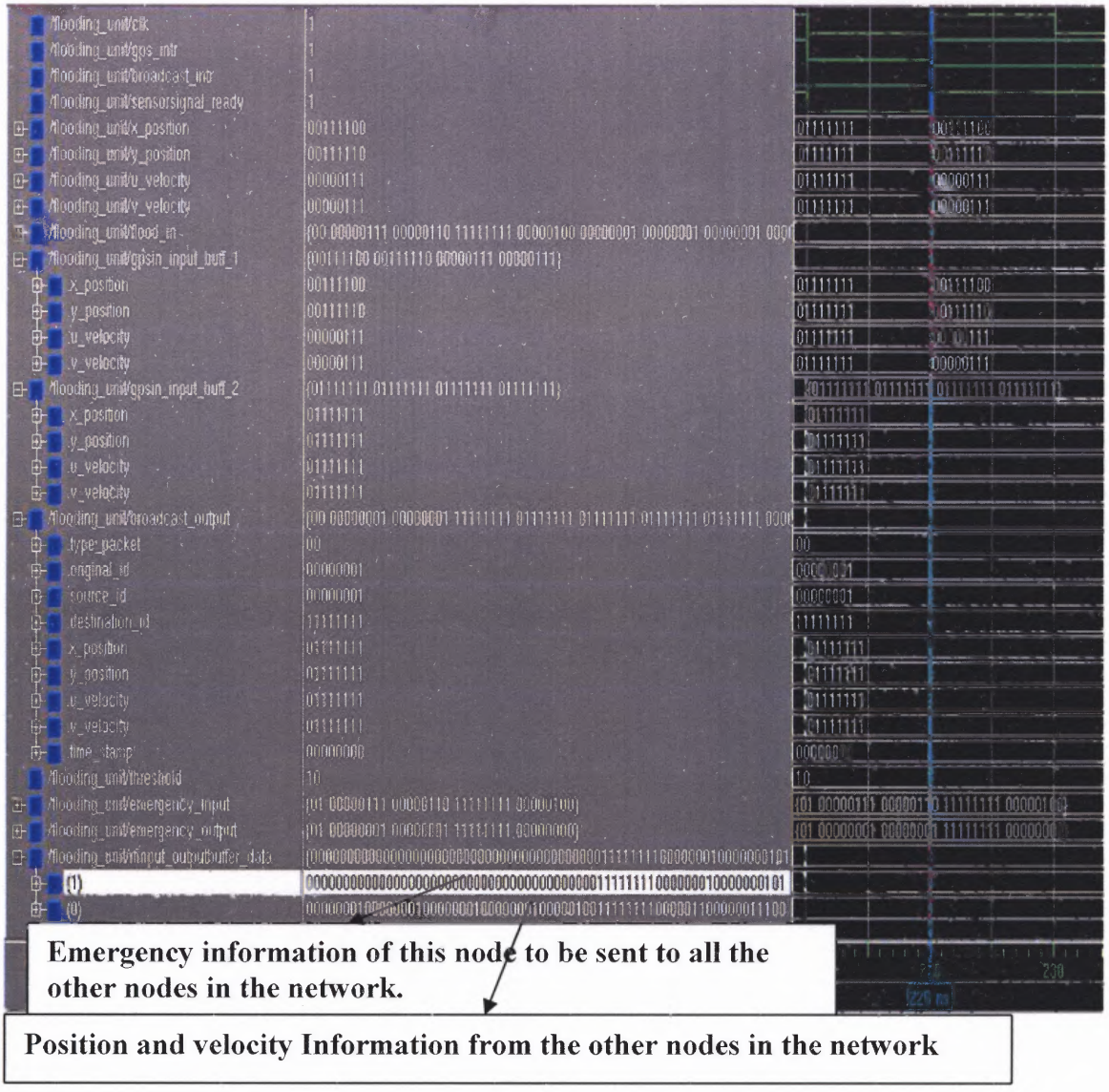

Figure 4.9 Timing simulation continued after figure. 4.8.

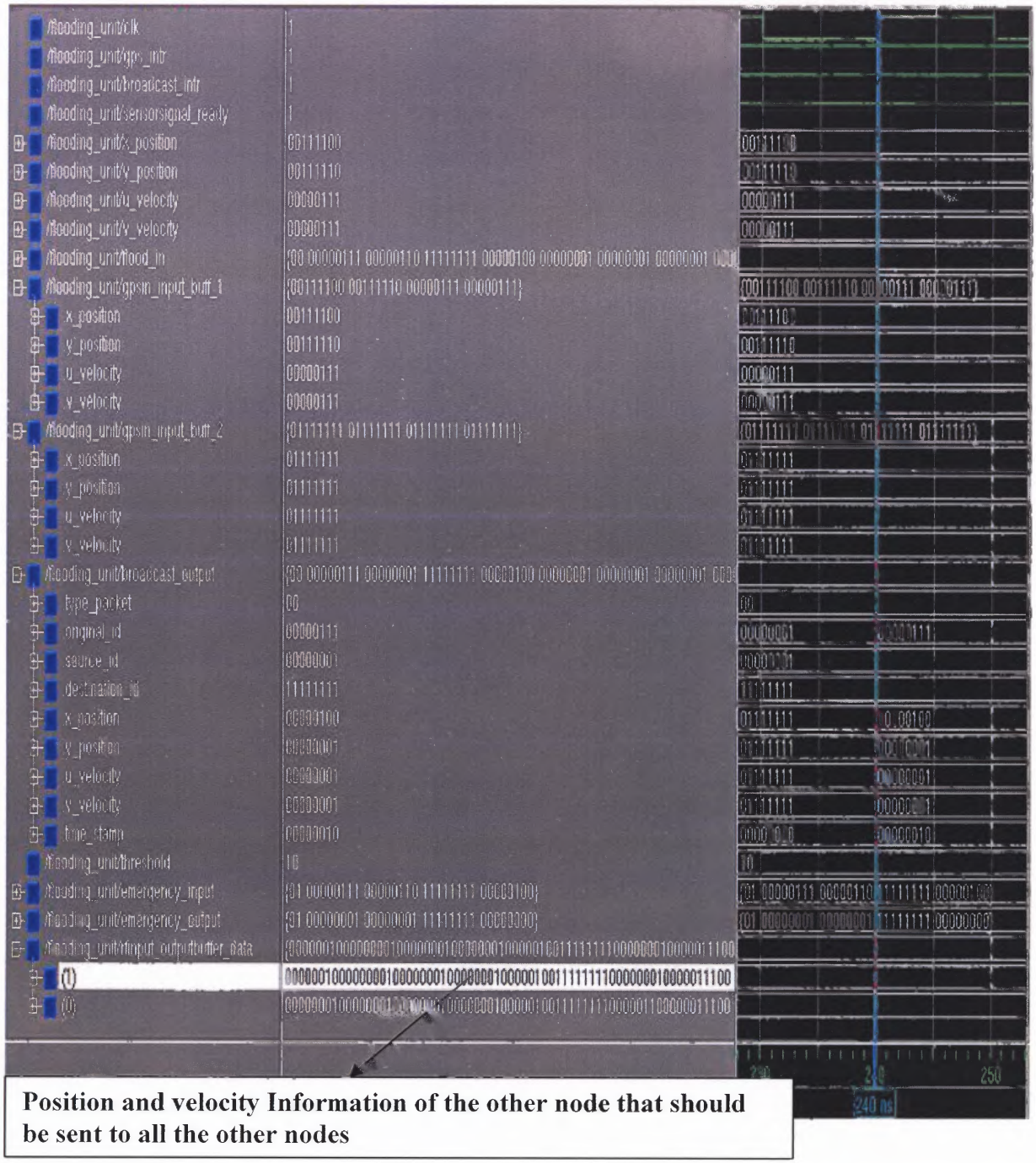

Figure 4.10 Timing simulation continued after figure. 4.9.

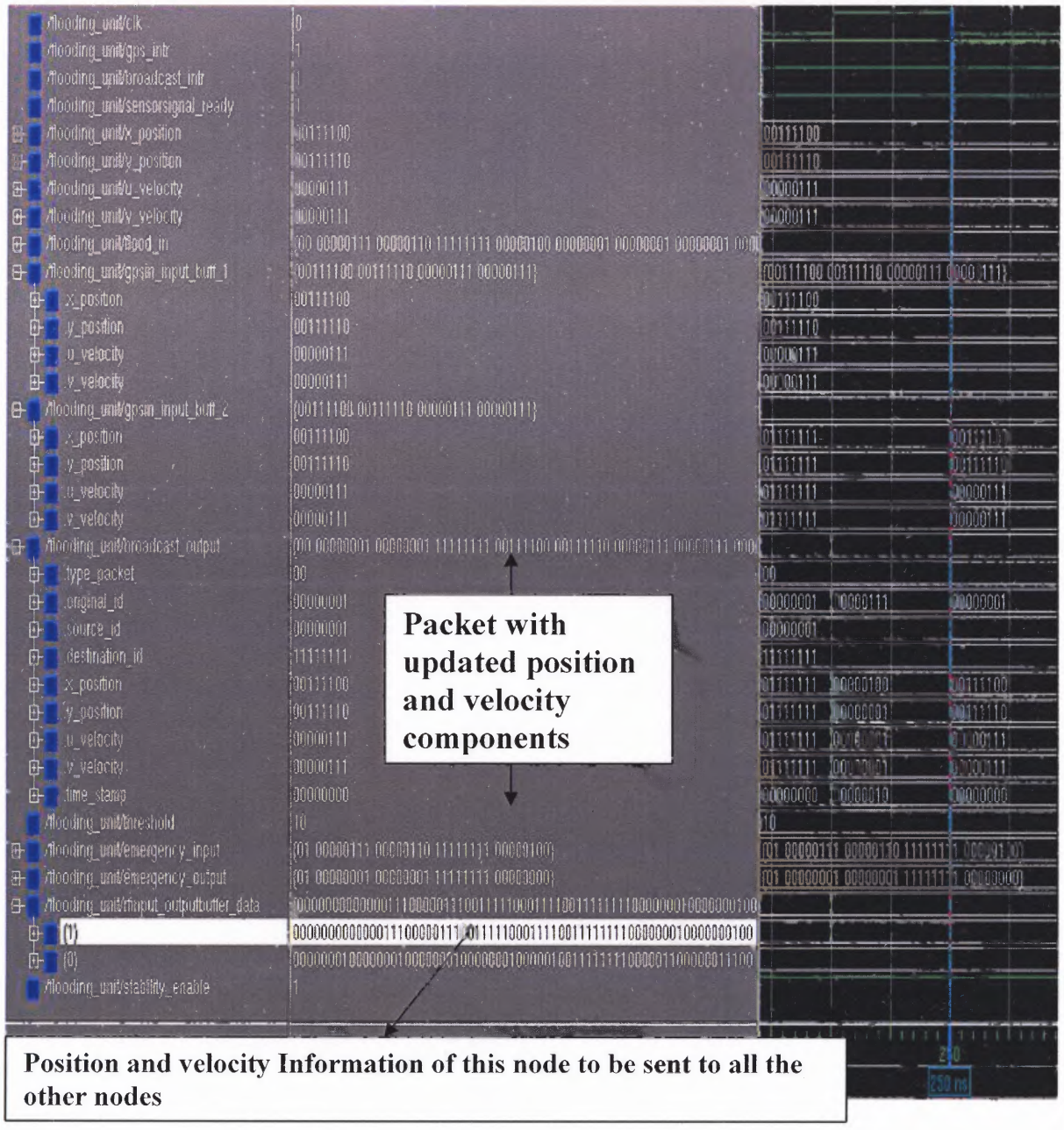

Figure 4.11 Timing simulation continued after figure. 4.10.

The stability unit is invoked by the stability_enable signal this is the output signal of the flooding unit and this is connected to data_update which is the input of stability unit. when the data_update is `1` which means that the position components and velocity

components are updated in the memory. The stability unit checks for this data_update when the data_update becomes '1' the position and velocity components of each node are read from the memory and stored in positiontable for calculation of stability, the stability of each node with respect to all the nodes in the network is calculated and stored in stabilitytable these values stored in the stabilitytable are written back into the memory.

As can be seen from the Figure. 4.12 the data_update becomes '1' at 80ns then read becomes '1' at 90ns indicating to read the position and velocity components from the memory. At 110ns it can be seen that the data is being read from that particular address 'addr' in the memory. In the Figure. 4.13 at 770ns read is '0' and the write is '1'which shows that the read operation is complete and the stability of each node with respect to all the other nodes in the network are calculated and ready to write back to memory. After the stability of all the nodes are written back into the memory and table_update is made '1' this can be seen at 2110ns in the Figure. 4.14.

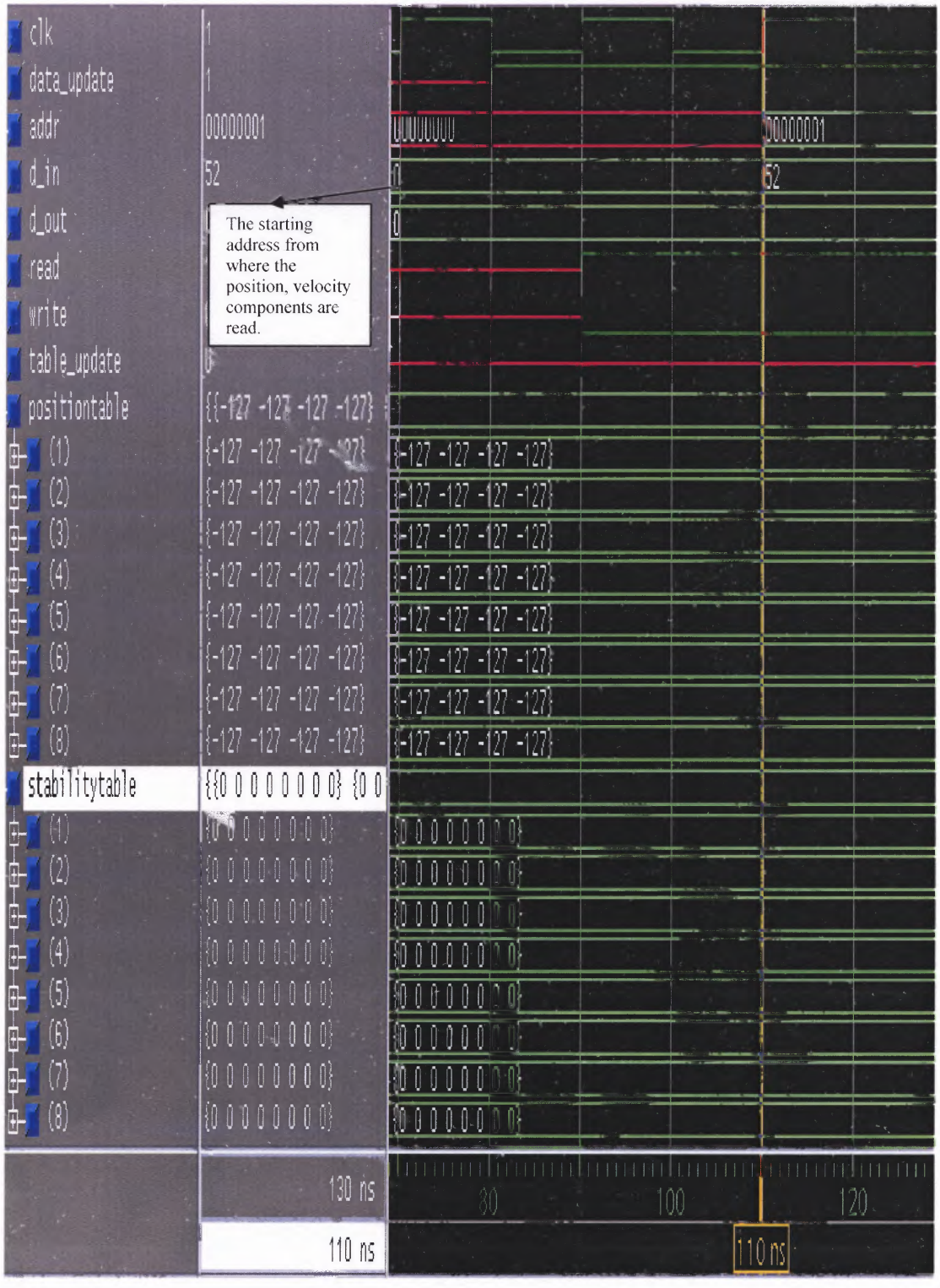

Figure 4.12 Timing simulation continued after figure. 4.11.

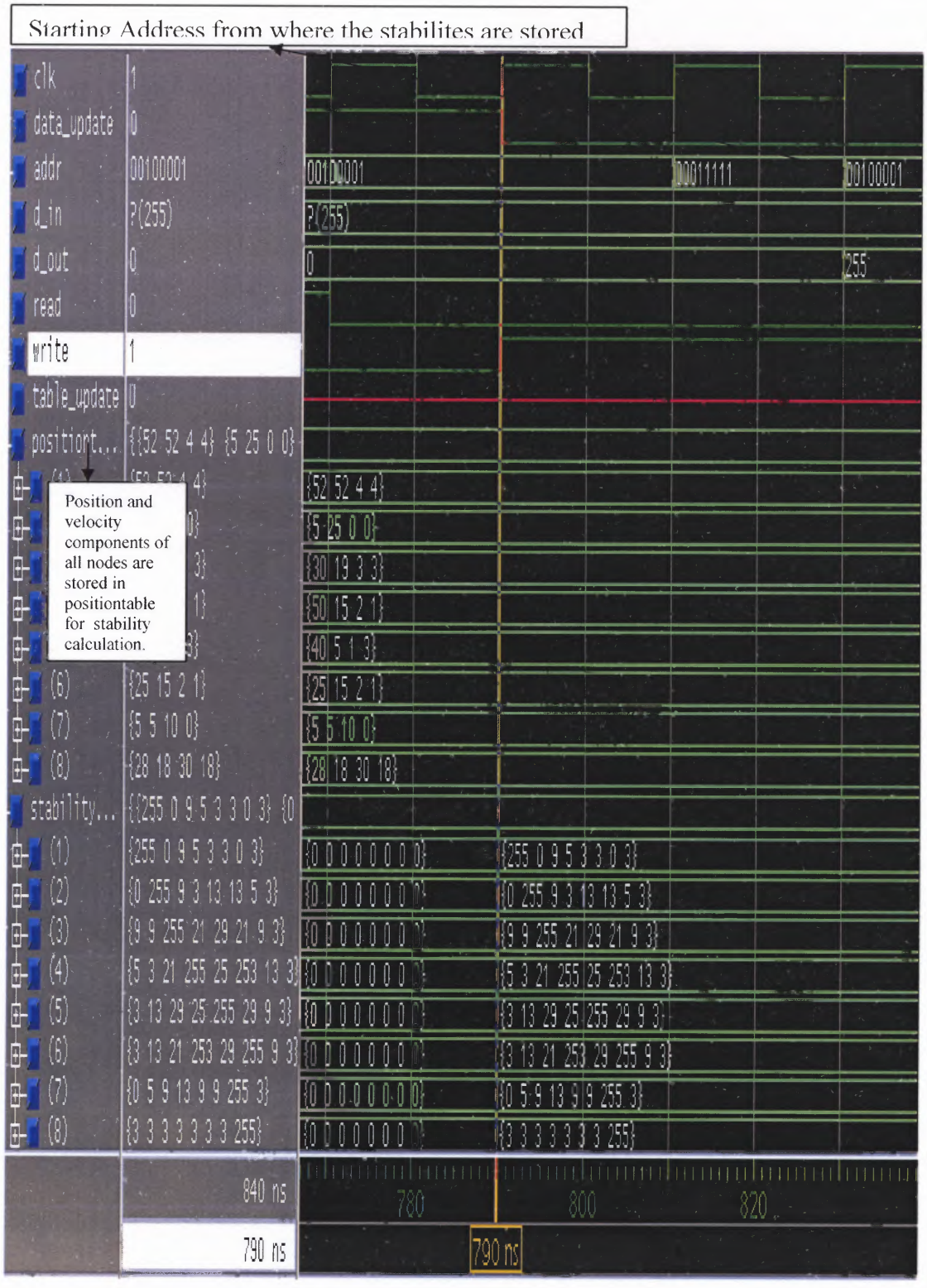

Figure 4.13 Timing simulation continued after figure. 4.12.

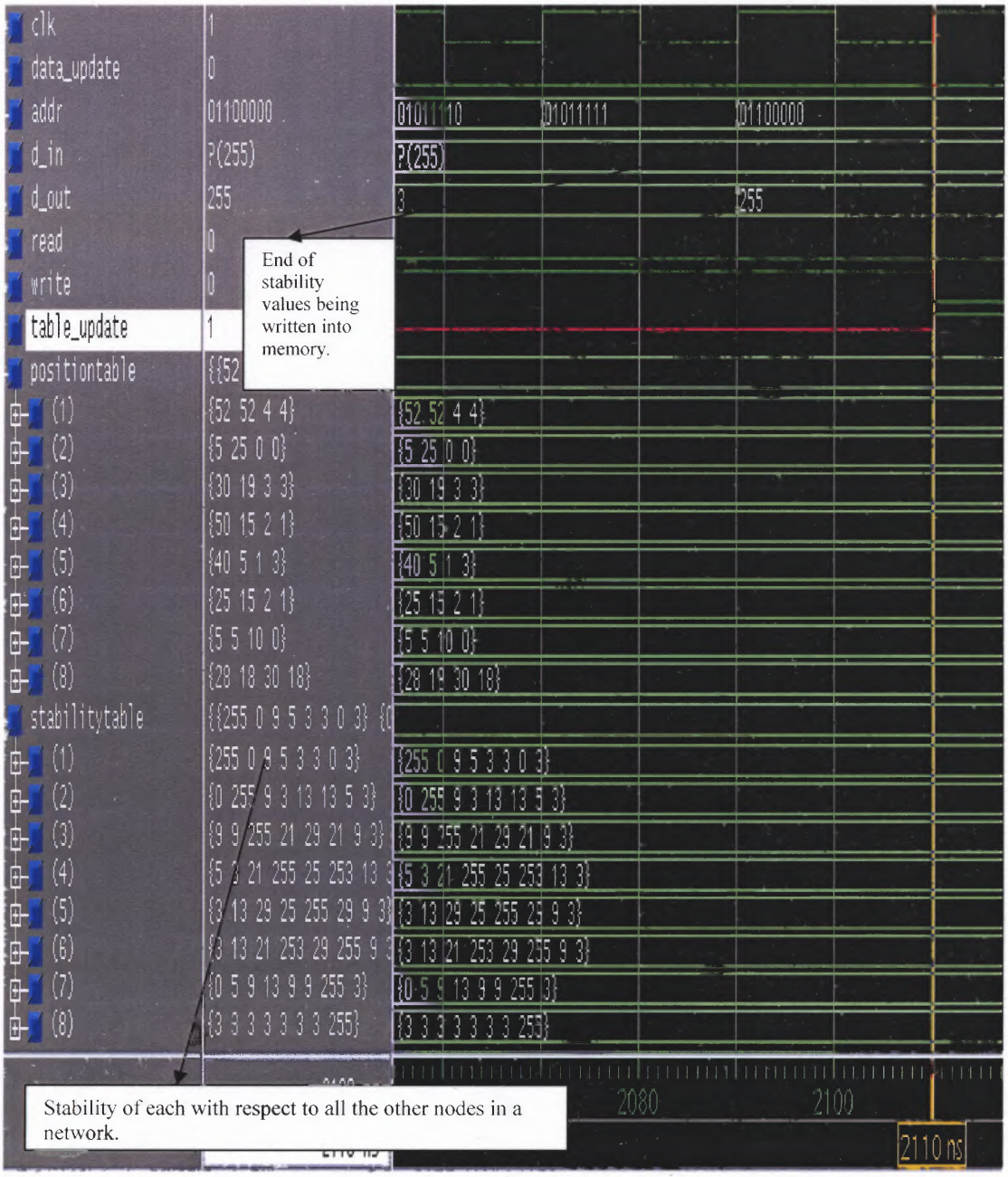

Figure 4.14 Timing simulation continued after figure. 4.13.

The routing table generation unit is invoked by the table_update signal this is the output of the stability unit this is connected to st_update which is the input of routing table generation unit. when the st_update is `1` which means that the stability of each node with respect to all the other nodes in the network are updated in the memory. The routing table generation unit checks for this st_update when the st_update becomes 'then the stability information is read from the memory into a buffer stabilitybuffer and the routing table is generated and stored in a buffer routingtable and is written back into the memory.

As can be seen from the Figure. 4.15 the st_update becomes '1' at 2110ns At 2130 the 'read' becomes '1' indicating to read the stability information of the nodes from the memory and the data is being read from that particular address 'addr' in the memory into the buffer stabilitybuffer. In the Figure. 4.16 at 3430ns 'read' is '0' indicating that the data from the memory is written into the stabilitybuffer for calculating the routing table and at 3470ns 'write' is '1' the routing table from the node '1'to all the other nodes in the network are calculated and ready to write back to memory. After the routing table form this node to all the other nodes in the network are written back into the memory the route_update is made this can be seen at 4770ns in the Figure. 4.17.

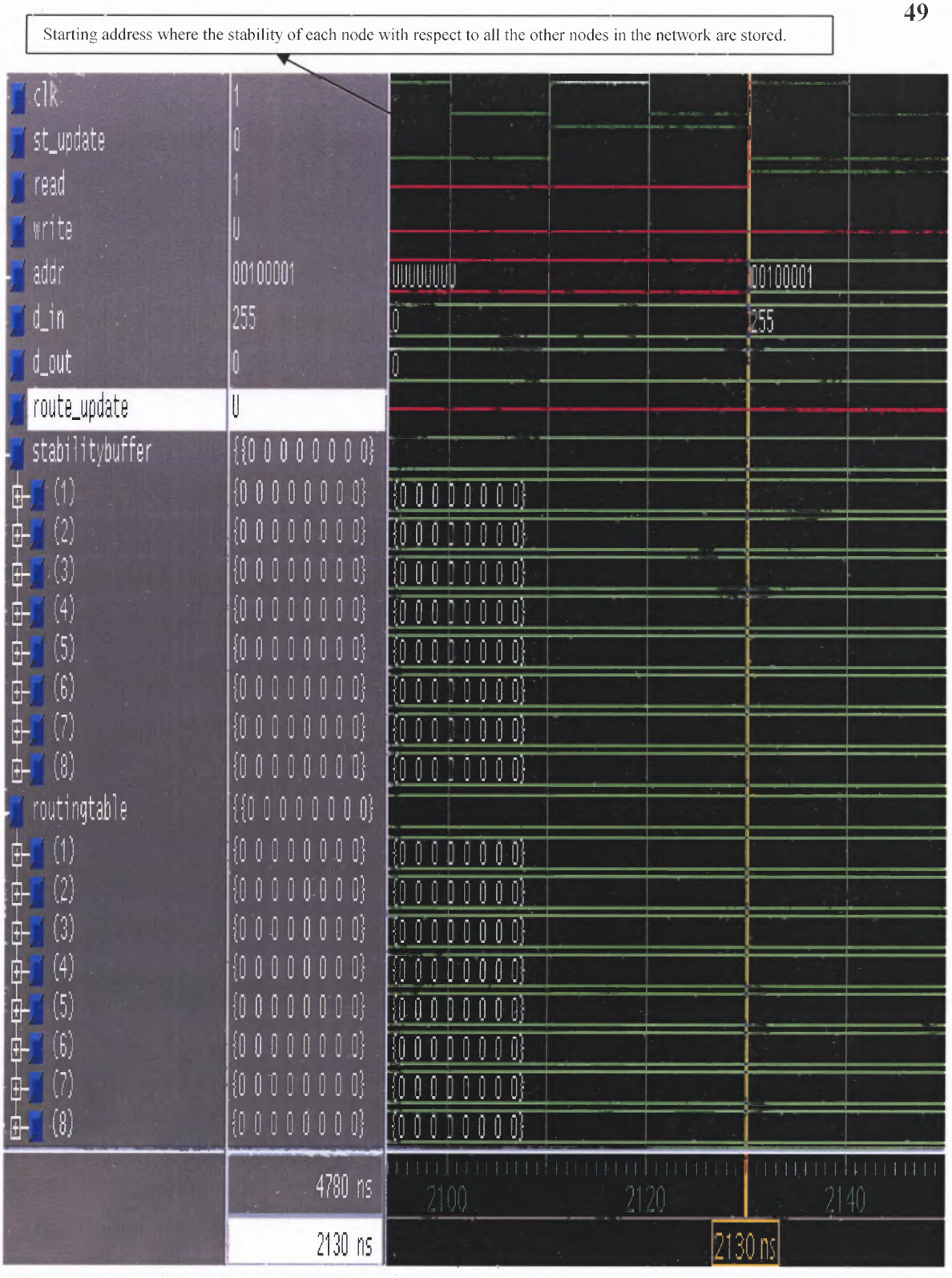

Figure 4.15 Timing simulation continued after figure. 4.14.

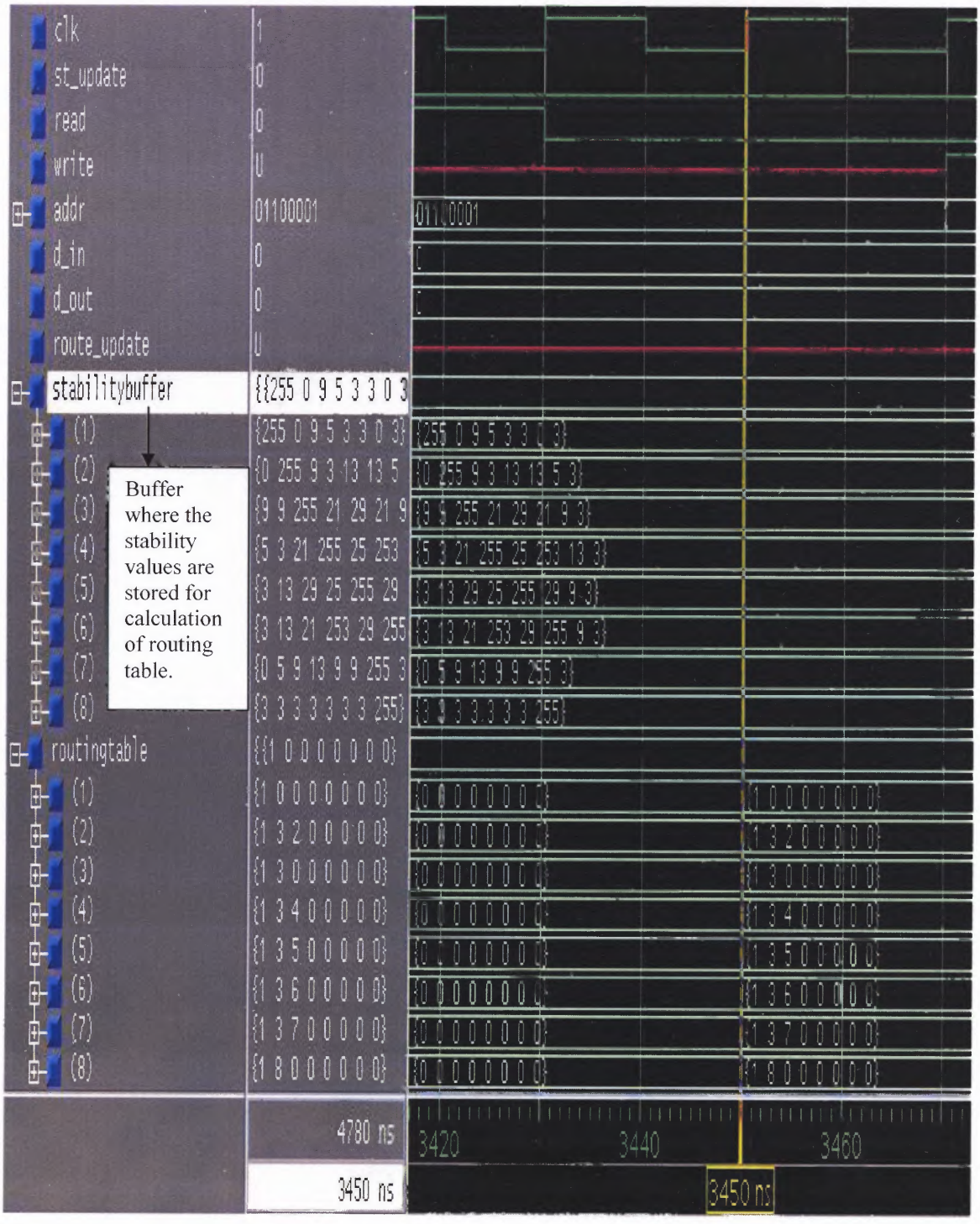

Figure 4.16 Timing simulation continued after figure. 4.15.

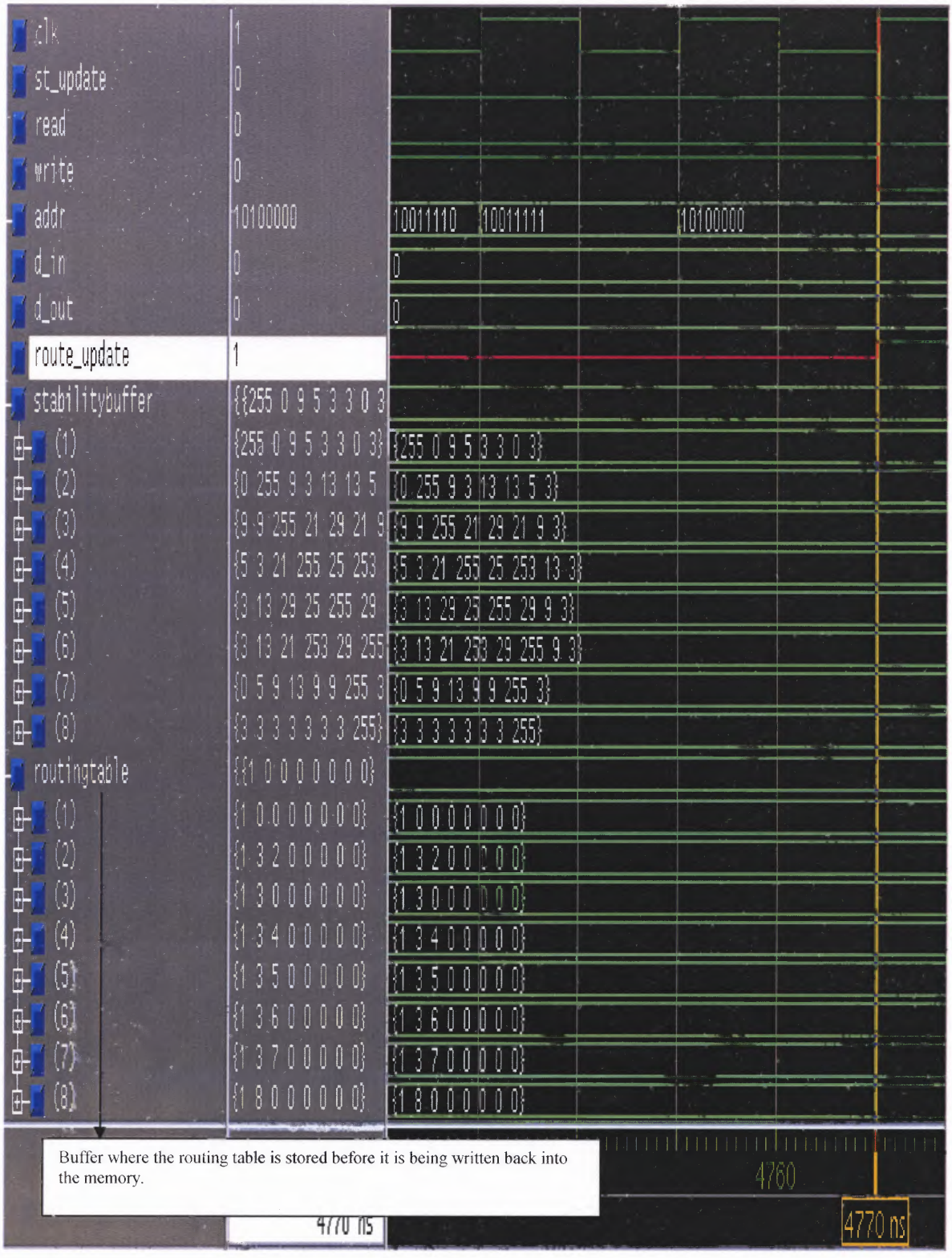

Figure 4.17 Timing simulation continued after figure. 4.16.

CHAPTER 5

SYNTHESIS OF THE MODEL

5.1 Synthesis Problems

The VHDL model is synthesized with Cadence Ambit Bulidgates. However, it was not synthesizable first up as the build gates didn't support multiple wait statements in a process and also the event scheduling on signals. Therefore the model had to be modified keeping these in mind. Only clock'event was synthesizable and this event could be used once in a process. Checking the condition for every rising or falling edge of a clock was a difficult task. The model was changed accordingly so that at the beginning of every process the rising edge of the clock is checked and whenever there is an event desired either the rising edge or the falling edge, a flag is made high and this flag was being checked periodically when there was a necessity for a condition to be checked at the rising or falling edge of the clock. Also mathematical functions like square root were not supported by the build gates, so whenever square root was necessary, the squares of the signal had to be taken and calculated for results. There was also problem due to insufficient memory, which resulted in disintegrating the blocks into smaller sizes. Power optimization is not supported by the Build gates. Only the blocks could be optimized for Area and Timing.

In Build Gates, first build the generic and then optimize the generic for Area and Time. When the generic is built, the verilog netlist is created which will have the cell structures of ATL and XATL formats. These formats are further disintegrated to the Nand, Nor And, Or, Mux, Latch, Flip Flop and other basic components during

Optimization. The problem was that during disintegration, some of the cells in the generic like ATL_TRI ,ATL_DC couldn't be disintegrated further down into one of the basic components. This created a problem in Silicon Ensemble as these ATL cells were not recognized. Therefore back tracking had to be done to see which process had created the ATL cells and the process had to be modified keeping the functionality in picture.

In the synthesis, the design was mapped to TSMC 0.35μ technology standard cells generated by CMC. A total of 1 million gates resulted after optimizing it with strict area and time constraints.

5.2 Synthesis of the Model with Cadence Ambit BuildGates

In this section, synthesis of the VHDL model with Cadence Ambit Buildgates is described. Following are the steps that were followed for synthesis.

1. Ambit BuildGates is started by entering a command called "cadence" at the console and choosing the 6^{th} option.

2. The 'File -> Open' menu is brought up and the timing library option is selected to read the 'timing.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 VHDL file is read by selecting the VHDL option in the File -> open window.

4. The VHDL model is then mapped to generic gates with 'Commands -> Build Generic …' and selecting the first 3 options in the build generic window.

5. 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
```

6. Optimization window is brought up by 'Command -> Optimize' menu. The 'Effort level' is set to high, 'Flatten mode' is set to off, 'Priority' is set to Area/Time and in 'Options' minimize area/Timing budget is selected accordingly. The optimization is shown in Figure. 5.1.

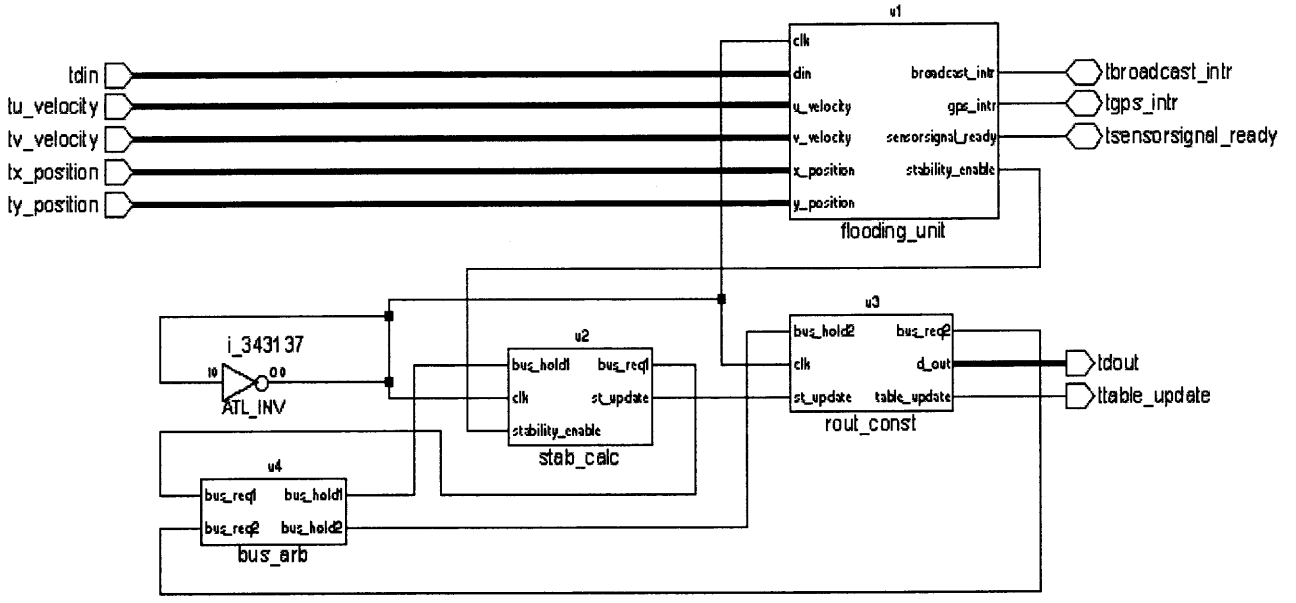

Figure 5.1 Synthesis for the processor implementing the stability routing protocol.

In the Figure. 5.1 it can be seen that synthesis for the processor implementing the Stability Routing Protocol, it has 4 units U1, U2, U3, U4 where U1 is the flooding unit which is a combination of Update and Transmitter Unit (UTx) and the Packet Forwarding and Update Unit (PFU), U2 is the stability calculation unit, U3 is the routing table generation unit and U4 is the bus arbitrator. It can be seen in unit U1 that all the inputs are connected to this unit, where gps_intr is the input from the GPS unit and x_position, y_position are the position components and u_velocity, v_velocity are the velocity components, sensorsignal_ready is the signal from the sensor network, broadcast_intr and d_in are the inputs from the transceiver and stability_enable is the signal which is the output of the unit U1 which is a indication to the stability calculation unit that there is a change in the position or velocity components and the new position and velocity components are updated and the stability unit can start the stability calculations. This stability_enable is connected to the input of the unit U2, the unit U2 starts after it receives this stability_enable signal. The output of unit U2 st_update is connected to input of U3 which is a indication to the routing table generation unit that the stability unit has calculated the stabilities with respect to all the other nodes in the network. This st_update is connected to the input of unit U3, the unit U3 starts after it receives this st_update signal the output of unit U3 is d_out which is the output that is sent to the transceiver.

7. The synthesis is complete and the design is saved as gate level verilog netlist 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.

CHAPTER 6

LAYOUTS

6.1 Place and Route with Silicon Ensemble

The placement and routing using Cadence Silicon Ensemble is discussed in this section. This performs the timing and power driven placement. The layouts of the individual blocks are put together using the top-level verilog module with the help of the same tool. Only regular placement and routing is done to save time and computer memory. The total die size without pads came upto 4.74 mm x 4.74 mm. This could not be imported into the Cadence IC tools for DRC and Extraction due to insufficient computer memory. In this tool the synthesized design is imported in block level verilog format .The standard cell library is imported into tool in LEF format which is the "cmosp35_4m.lef". The output to this tool is the layout which is stored in the form of LEF block and DEF form which is then imported to the Cadence Virtuso Layouts for Design Rule check and the Extraction. Following steps are followed to get the layouts using the Silicon Ensemble.

1) Silicon Ensemble is started by entering a command called "cadence" at the console and choosing the 4^{th} option.
2) The "cmosp35_4m.lef" file is imported by using the 'File → Import → LEF' menu. After this step a database is created for storing the created design and viewing it when needed.
3) The verilog netlist which has a '.v ' extension ,which was generated in the Ambit Build Gates is imported by using 'File → Import → Verilog' menu. The top module should be identified correctly. Also the power (VDD!), and ground

(VSS!) nets should be entered as show .Make sure the VDD and VSS are in upper case. The reference libraries and the compiled verilog output libraries are "cds_vbin" by default . Also a verilog module called "cells.v" is also imported along with the main verilog module, which has the information about the basic gates and their specifications, which are helpful for generation fo the layouts.

4) After the compiling is done successfully the floorplan has to be initialized by using 'Floor plan → Initialize' menu. The IO to core distances in the dialog are initialized to 40 microns. Space utilizations is kept for about 70-75 %. 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. Figure. 6.1 shows the layout without the power and ground stripes.
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.
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.

Figure 6.1 Layout of a block without the power rails.

8) The connection to the rails is done by using 'Route → Connect Ring' menu. Routing is done with 'Route -> Wroute' menu here global and the final routing is done together.

9) The design is saved as DEF, LEF BLOCK 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.

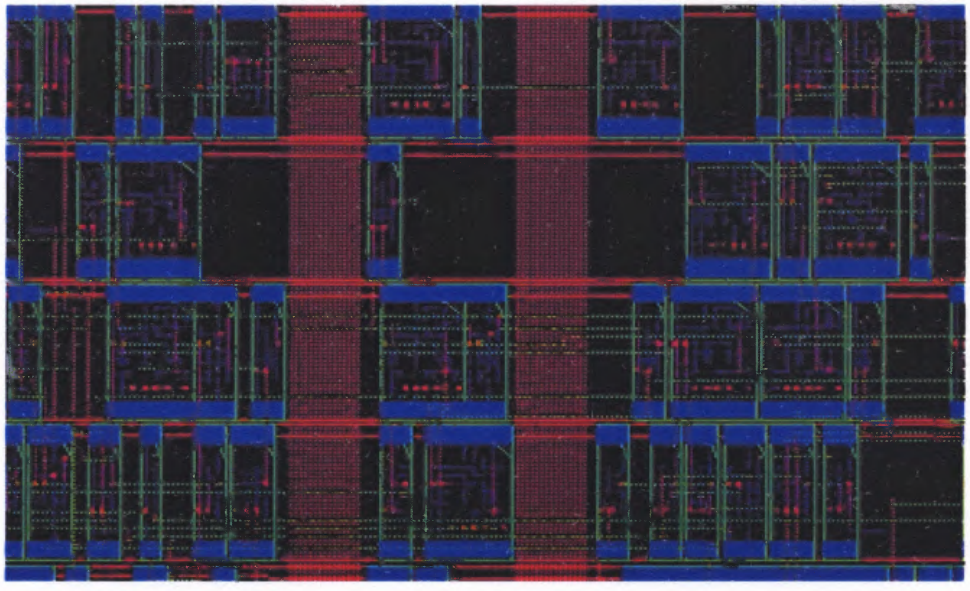

Figure 6.2 VDD, VSS stripes in between the layout.

Figure. 6.2 shows the power paths that pass over the core area in between the layout these power paths are used to reduce the voltage drop. This concludes the layout process through Silicon Ensemble tool. The DEF and the LEF block format is then exported to the Cadence IC tools for DRC and Extraction.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

This thesis gives the architectural details and the on–chip implementation procedure for the Stability Routing Protocol for Mobile Ad-Hoc Networks. The proposed approach provides a way for sending the sensor information to all the other nodes with out any data loss. A VHDL model for the proposed architecture was developed and the high level simulations confirm the performance of this architecture. The VHDL model is synthesized and implemented on silicon with some exceptions. The total number of gates produced was around 1 million. The numeric_std is not supported by the ambit buildgates synthesis tool so a own package was developed for the arithmetic operations of the type std_logic_vector, the function square root which is necessary for the stability calculation module is also not supported by std_logic_1164.all library so successive approximation method is used for calculating the square root which uses more for loops and repetitive procedures which resulted in large gate count of which most of the gates were due to the use of 'for' loops and also due to the lack of complex gates in the standard library. The finite state machine model is used in developing update and transmitted unit and the packet forwarding unit this resulted in large area and hence the gates. The gate level verilog simulation of the design could not be carried out and so was the DRC and extraction because of large gate count and insufficient computing power. It is therefore required to reduce the gate count.

7.2 Future work

The gate count could be considerably decreased by doing one or more of the following.

1. Use of Karnaugh–maps for breaking down the modules and getting it to a basic Boolean expressions and then performing the manual synthesis.
2. Use of shift operations for addition and multiplication of Std_logic_vectors, as there is no direct syntesis for operations on Std_logic_vector
3. Reducing the type conversion, like integer to std_logic_vector and back, can reduce the gate counts drastically.
4. Including libraries in the Ambit Buildgates which support numric_std and unsigned libraries for optimization during the synthesis.
5. Using the pipelined architecture for square root calculation increases its performance.
6. Using standard cell libraries which have complex gates, for optimization in area and timing. The library used here was very well optimized for only for timing and area.
7. The device level simulations can be carried out by using Star-Sim and Start-time which are high capacity simulators.

With implementation of the above steps the design could be more efficient.

REFERENCES

1. Mingliang Jiang, Jinyang Li, Y.C. Tay, "Cluster based routing protocol" August 1999 IETF Draft, 27 pages.

2. Charles E. Perkins, Elizabeth M. Royer, Samir R. Das, "Ad-hoc on-demand distance vector routing", October 99 IETF Draft, 33 pages.

3. David B. Johnson, Davis A. Maltz, "The Dynamic source routing protocol for mobile ad-hoc networks" October 1999 IETF Draft, 49 pages.

4. R. Dube, Cynthia D. Rais, Kunag-Yeh Wang, "Signal stability based adaptive routing for ad-Hoc mobile networks",IEEE Personal Communications, Feb. 1997, pp. 36-45.

5. Elizabeth M. Royer, Chai-Keong Toh, "A Review of current routing protocols for ad-hoc mobile wireless networks", IEEE Personal Communications, Vol. 6, No. 2, pp. 46-55, April 1999.

6. Andrew S. Tanenbaum, "Computer Networks", Third Edition, Prentice Hall, March 1996.

7. S.Xu, S.Papavassiliou and Amouris, "On the optimal multizone configuration for the position-guided sliding-window routing (PSR) protocol for mobile ad-hoc networks", IEEE Military Communications Conference, October 2000.

8. J. Ackeret and D.Specht, "Handbook for datum transformations and coordinate conversions and GEOTR software", IEEE Conference and Exhibition, 2000.

9. D.K.Olson "Converting earth-centered, earth-fixed coordinates to geodetic coordinates", IEEE Transactions on Aerospace and Electronic Systems, January 1996.

10. J.Zhu "Conversion of earth-centered, earth-fixed coordinates to geodetic coordinates", IEEE Transactions on Electronic Systems, July 1994.