COMMUNICATION MECHANISM FOR HETEROGENEOUS COMPUTER SYSTEM

by
Lau Wing Chiu

Master of Philosophy

The Hong Kong Polytechnic University

May 1999

To My Parents and Elaine

Abstract of thesis entitled

'Communication Mechanism Heterogeneous Computer System'

submitted by Lau Wing Chiu

for the degree of Master of Philosophy

at The Hong Kong Polytechnic University in May 1999

Heterogeneous computing system is suitable for image processing because different kinds of processor could be used to handle image processing problems in different levels. Image data can be delivered to the low level processors for number crunching operations. Then the processed data can be delivered to other processing layer to perform recognition and other high level processing tasks. Apparently, the performance of this system is highly depended on its communication mechanism.

The heterogeneous system consists of two layers. The first layer consists of a DSP network for low-level processing. While the second layer is a network of workstations, or personal computers. This maps well onto a simplified image processing paradigm with the DSP layer for low-level processing and the network of workstations for non-image based processing. Based on this paradigm, the system resembles a two stage multi-layers architecture. In the pre-processing phase, image data is delivered to the low level processors and operations are handled by using SPMD paradigm. The processed data is delivered to the high level processors where feature extraction and recognition tasks are performed.

In order to connect the two layers together, an efficient mechanism must be applied. From our studies, the QuickRing technology which can sustain a high data throughput rate of 160 Mbytes/s, is suitable for our proposed system. We have developed a model for the QuickRing controller chip and studied the performance of the communication network of the proposed heterogeneous image processing system by simulation. The performance of QuickRing in different data transmission modes has been studied and this produces information regarding the feasibility for

achieving real-time processing under different transmission modes.

To realise the system based on the QuickRing communication controller, hardware components, such as a bridge hop, a DSP Layer Controller have been designed. In addition, a new communication protocol, based on the low level protocol of the QuickRing Network, is developed.

In our proposed system, the channel connecting the two layers could become a communication bottleneck and its performance is studied using a software simulator. We found that the communication channel will not affect the system performance.

ACKNOWLEDGMENTS

I am indebted to my supervisor, Dr. Y. F. Fung. I would like to thank him for his elaborate supervision in the two year graduate study and another year of extension period. He made many valuable suggestions during period of research and writing of this thesis. Without his guidance and support, this thesis could not be completed.

I wish to express my deepest appreciation to Mr. W. L. Chan. He is the first lecturer who led me to be engaged into the interesting research area of image processing in my final year project of my undergraduate study. Without this experience, I will not start my research study.

Sincere thanks are extended to my appreciation to my co-supervisor, Dr. A. B. Rad. He is very concern about our studies and helps us to overcome difficulties.

I would like to thank my colleague, Mr. M. F. Fikret and Mr. T. M. Heung. They always give me very useful opinion in my study.

Financial support by the way of studentship under Hong Kong Polytechnic Research Grant No 350/513 for the two-year graduate study is also acknowledged.

Also, I would like to thank to my colleague and boss of my present job. In the extension period, they give me support and arrangement to allow me to finish the thesis.

Finally, I would like to thank my best friend, Miss Elaine Kwok, who always give me encouragement and support that help me to face different difficulties. Her attitude in work and studies also affect me deeply.

Lau Wing Chiu

TABLE OF CONTENTS

TABLE OF ACRONYMS

DSP	Digital Signal Processor
DLC	Digital Signal Processors Layer Controller
MC	Master Controller
HP	High Level Layer Personal Computer
IUA	Image Understanding Architecture
WPM	Warwick Pyramid Machine
SIMD	Single Instruction Multiple Data
SPMD	Single Program Multiple Data
PE	Processing Element
MIMD	Multiple Instructions Multiple Data
DAP	Distributed Array Processor
ASP	Associative String Processors
FDDI	Fiber Distributed Data Interface
DQDB	Distributed Queue Dual Bus
ATM	Asynchronous Transfer Mode
FIFO	First In First Out Buffer
CCN	Control and Communication Network
ACD	Address Conversion Decoder
PAL	Programmable Array Logic
RM	Resource Manager
RA	Resource Agent
TM	Task Manager

TA	Task Agent
AEBB	Available Empty Buffer Blocks
CI	Communication Interface
TID	Task Identity number
PID	Processor Identity number
NAB	Number of Available Buffers
QC	QuickRing Controller

LIST OF FIGURES

LIST OF TABLES

Chapter 1

1. INTRODUCTION

1.1 OVERVIEW

The objective of this thesis is to design an effective communication mechanism for a multi-layer heterogeneous image processing system. Heterogeneous image processing system is mainly designed to tackle integrated image processing problems, for example autonomous vehicle guidance [1]. Image processing operations can be broadly categorised into three different classes, or levels, namely the low-level, intermediate-level and high-level. An integrated image processing problem refers to such a problem that includes all types of processing levels. Since processing algorithms in different levels have different computing requirements [2], therefore, a heterogeneous system, embodying different types of processing elements to handle different processing tasks, provides an effective solution for the problem.

In addition to the computing requirements, the quantity of data involved in an integrated vision problem is enormous and therefore, computing systems which can reduce the processing time is also sought. Parallel computing system is a feasible solution for such problem and there are many dedicated parallel machines designed for image processing, examples include the CLIP7A system [3], the MPP [4] and DAP [5]. Considering both requirements, researchers are developing multi-layer architectures for solving integrated vision problems. In a multi-layer system, each layer is dedicated to one type of image processing operations. In order to minimise the processing time, each layer will include many processing elements so that parallel computing can be achieved. The IUA [6] and WPM [7] are examples of multi-layer system. Another advantage of a multi-layer system is for the processing of a continuous stream of images, usually coming from a video camera. In such case, different processing layers will be working on different image frames in a pipelining mechanism.

In the above paragraphs, we have introduced the need for a multi-layer architecture and we have mentioned that different computing requirements are needed in different processing levels. We will now briefly discuss the different processing levels and try to determine their computing requirements.

Low level image processing usually refers to image to image transform. Such operations can be applied to enhance image features, such as edges. Since the operation is usually applied to every pixel in the image so SIMD type control mechanism is suitable. In addition, a one processing element (PE) per pixel

arrangement can maximise the parallelism. There are number of massively parallel, including CLIP4 [8] and DAP [5], using single bit processor, SIMD systems designed for low level image processing tasks.

High level image processing algorithms [2] process symbolic data, or symbols, extracted from the input image to perform tasks such as recognition and interpretation. Processing tasks at this level can be classified as knowledge processing and symbolic processing. Model-based object recognition is one example. With a set of object models given, the task finds instances of these objects in a given scene. Most model-based recognition operations are based on hypothesising matches between scene features and model features, predicting new matches, and verifying or changing the hypotheses through a search process. Due to the process requirements of this level, researchers usually apply high-end workstation which supports programming languages such as Prolog or Lisp to carry out the processing. Multiple Instruction Multiple Data (MIMD) control mechanism is suitable for co-ordinating operations in this level. Moreover, the degree of parallelism demanded in this level is relatively low, comparing to those in the low-level, because the quantity of symbols being processed is small.

There are different definitions for intermediate level processing [2] [9] and it is not always easy to identify intermediate processing tasks. In our studies, we define intermediate level image processing algorithms as a process to convert the format of information from an image to symbols. According to this definition, the input to an intermediate-level processing is still an image. Hough transform [10] can be regarded as an example of intermediate processing based on our definition. In Hough transform, an image is transform into a 2-D parameter space which represents function of straight lines. The computing requirements for intermediate processing are not well defined. In many cases [11] [12], researchers are using multi-bit processor and MIMD control mechanism in this layer.

In the above paragraphs, the computing and control requirements for different processing levels have been introduced. A three-layer system with a massively parallel SIMD array for low-level processing and two MIMD arrays for both intermediate and high-level processing should map well to the problem. There are several systems, such as the IUA and WPM, built based on this concept, and a brief description of them is presented in Chapter 2.

In our system, we only provide two processing layers, one for low level and the other one for high level. As described above, the major function of the intermediate level processing is to convert information conveyed by an image into symbolic data. This processing can be implemented by either the low level or the high level processing layer as well. Reducing the processing layer to two makes our system easier to implement. In addition, the software model is also simplified. Now all image based algorithms will be implemented in the low-level layer and

other algorithms run on the high-level layer.

Another feature of the our proposed system is the application of linear array as our processing layers. The linear architecture is easier to construct [13], comparing to the traditional 2-D array and other advantages of linear array can be found in [14]. Many linear systems [15] [16], have been built using powerful DSP processor, such as C40 and Transputer [17] for low level processing. Although the number of PE is small, comparing to a massively parallel system, due to the use of powerful processor, a linear array can also handle low level processing tasks effectively. Therefore, a linear array of DSP will be applied for low level processing in our system.

For high level processing, a network of PCs will be applied. Due to the advance in VLSI technology, many powerful microprocessors have been produced, examples include Pentium II [18] and Alpha [19]. A network of PCs can deliver significant amount of processing power and has applied to solve various computational intensive problems [20] [21] [22]. Moreover, the PC can also support AI programming languages and therefore, suitable for our application.

We have outlined the system architecture of the two-layer system. Obviously, a major design factor affecting the performance of the system is the communication mechanism. The communication channels linking the layers may become a bottle-neck. Therefore, a high bandwidth communication mechanism must be provided and in addition, a proper control method must be designed. This thesis is concentrated on these aspects of the system.

1.2 ORGANIZATION OF THESIS

In Chapter 2, a survey of multi-layer image processing system is given. The requirements for the communication network of the proposed system are defined. A survey of different communication protocols is presented. Different aspects of a communication protocol including cost, bandwidth, collision rate, topologies are compared. Eventually, a network using the QuickRing [23] communication controller is selected.

In Chapter 3, technical details of QuickRing are presented. A QuickRing network is constructed by connecting several QuickRing Controllers (QR1001) as a ring. The data transfer rate of QuickRing is 160 Mbytes per second. Data communication between nodes in a QuickRing network can be achieved either by Directed mode or Multicast mode. The simulation and performance analysis of different transmission modes are carried out to determine a suitable mechanism for our system. In addition, the problem involved when applying Multicast mode

in to our system is identified.

In Chapter 4, the network topology and hardware design of the DSP Layer Controller (DLC) and the bridge hop are given. The design of DLC is targeted to solve the problem mentioned in Chapter 3. The functions of the designed hardware in the system and how they handle the problem discussed in Chapter 3 are present.

In Chapter 5, the proposed system control mechanism is presented. The advantages of the control mechanism are discussed. The reliability of Multicast data transfer mode can be ensured by the proposed mechanism. Combined with the hardware components that are mentioned in Chapter 4, the proposed mechanism can solve the problem induced by the multicast mode of QuickRing.

In Chapter 6, simulations of the hardware components and the communication mechanism are discussed. Simulation is given to show the operation steps of the proposed communication mechanism. Another software simulation is presented to verify the performance of the hardware components.

Finally in Chapter 7, the studies carried out in this project are summarised. Then, concluding remarks and directions for future research are presented.

Chapter 2

2. SURVEY AND BACKGROUND

2.1 Introduction

In Chapter 1, we have highlighted the need for a multi-layer image processing system and introduced our proposed two-layer system. As mentioned in Chapter 1, the communication network for the system is an important element that must be designed carefully. In this Chapter, we will first present the system architecture of several heterogeneous systems and we will concentrate on the inter-layer communication mechanisms employed by those systems. Because we are using a network of PCs for high level processing, a survey on different network technologies is presented. Through this survey, we can select a suitable technology for the communication network of our proposed system.

2.2 Heterogeneous Processing System

Heterogeneous systems are dedicated for solving integrated image processing problems. Usually, such a system consists of three processing layers, with a SIMD array for low level processing and two MIMD arrays for intermediate and high level processing. The Warwick Pyramid Machine (WPM) and the Image Understanding Architecture (IUA) are two examples. Due to the reduction in information, the number of processing elements decreases in the upper layers. Therefore, the system is in a form similar to a pyramid. In addition, we will also introduce the Hybrid system which consists of two processing stages.

2.2.1 Warwick Pyramid Machine (WPM)

The Warwick Pyramid Machine[7] [24](WPM) consists of four layers and its organisation is shown in Figure 2.1. The low-level of WPM uses 16x16 DAP [25] processors to form a 128x128 PEs array. The DAP is a bit serial processor with a 1-bit ALU and four 1-bit registers. The intermediate layer is composed of a microcontroller (AMD 29116) and a sequencer (AMD 29331). A network of INMOS transputers is used for symbolic processing at the high level. This MIMD layer contains 8x8 32-bit processors with 256 Kbytes local memory. Each transputer is connected to its nearest four neighbours. The host computer at the fourth layer is used as a user interface and mass storage. The data exchange between SIMD low-level, intermediate level and transputer network is provided with dual ported RAMs. It is interesting to point out that the function of the intermediate layer is to control the bottom layer, forming a M-SIMD array as

shown in Figure 2.2.

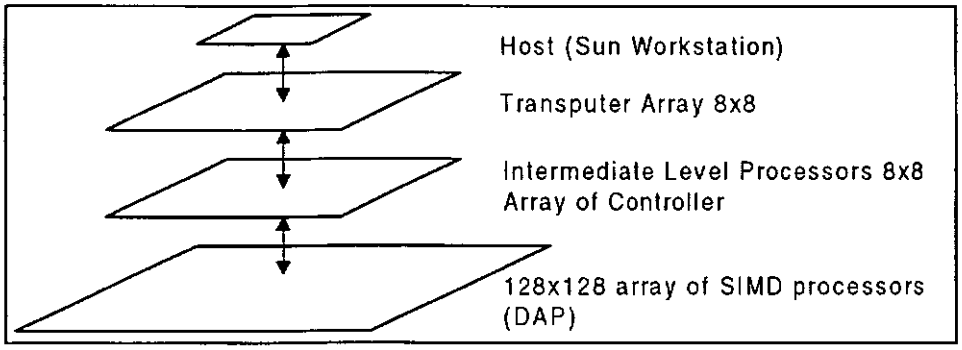


Figure 2.1 Warwick Pyramid Machine

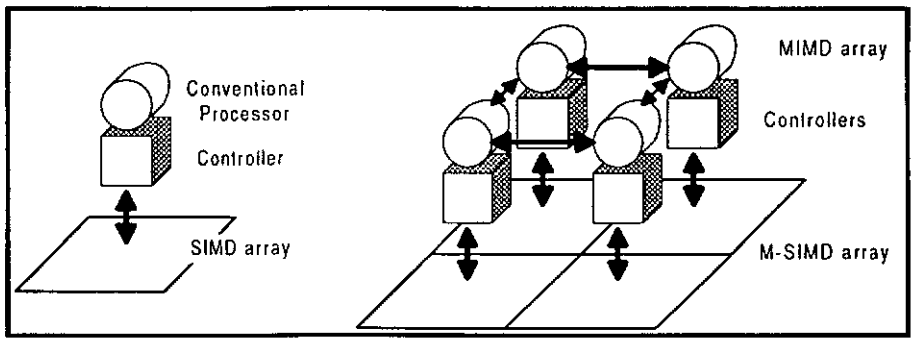


Figure 2.2 (a) Conventional SIMD architectures (b) the M-SIMD architecture

The advantage of this arrangement is to improve the communication bandwidth between the two layers. It is because each processor in the SIMD layer has its own path connected to the processor in the middle layer.

2.2.2 Image Understanding Architecture (IUA)

Different from the WPM, the IUA [6] [26] system embodies three levels of parallelism, see Figure 2.3. It is a hierarchical and heterogeneous architecture developed at University of Massachusetts. The lowest level is a SIMD 1Kx1K mesh-connected array of bit-serial processing elements to process pixel data of an image. The intermediate level, in the first generation IUA system, is an array of 64x64 mesh connected 16-bit DSP (TI320C25) chips with 256K of local memory. The DSP is capable of handling numeric computations and symbolic operations. Besides, grouping and matching operations, which are based on stored models are handled by the middle layer. A large shared memory is placed between the low level layer and the intermediate level layer. The DSPs in the intermediate level are

directly controlled by processors in high level layer. The high level layer is formed by an array of powerful 8x8 general-purpose microprocessors. They are responsible to process the high-level symbolic operations and generate the overall IUA processing plan. Also the high level layer and the intermediate level layer are connected via the shared memory between them. The high level layer treats the lower levels as an intelligent global shared memory.

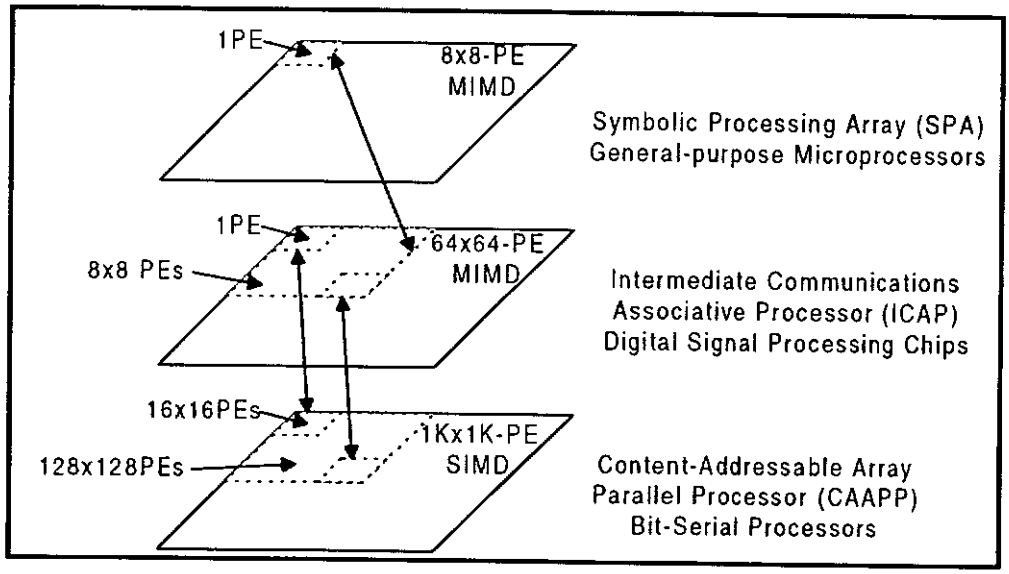


Figure 2.3 Image Understanding Architecture

2.2.3 ASTRA

ASTRA is another multilevel heterogeneous system developed at Brunel University [27]. The system architecture incorporates three levels of control and processing hierarchy. The lowest layer of the system is built with Associative String Processors (ASP) [28] and the next higher level is a low-level ASP controller. Design of the low-level controller is rather complex. It consists of a micro programming unit, made of an AMD 2931 sequencer and a micro instruction buffer. The scratch pad memory, scalar data buffer, and performance monitor are some of the components of this layer. Intermediate-level ASP controller incorporates 68020 processors. A 512 Kbytes shared memory block is used for data exchange between high and intermediate level controller. The high level controller of the system is a Sun 3/260 workstation. All system units are connected via VME bus. The high and intermediate-level controllers are programmed with modular-2 and the Low-level ASP Module (LAM) is programmed with LAM programming language. Three operating systems are used to co-ordinate ASTRA system. The ASTRA system is a test bed for recently developed TRAX-1 machine [29].

2.2.4 *HYBRID Computer Architecture for Machine Vision*

Figure 2.4 depicts the HYBRID system [30] and Figure 2.5 shows how different vision tasks are mapped onto this architecture. Low-level vision operations are performed by pipelined special function processors and high-level operations by a MIMD-type multiprocessor system. A signal-to-symbol processing module will be used for intermediate-level operations. A Sun-3/160 computer is used to emulate the operation of this module. The Sun-3 also controls the operation of the system, providing a user-friendly environment for software development.

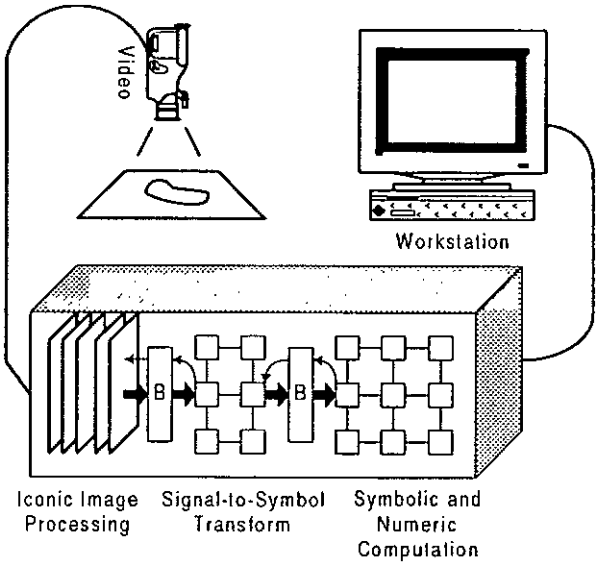


Figure 2.4 Block Diagram of HYBRID computer

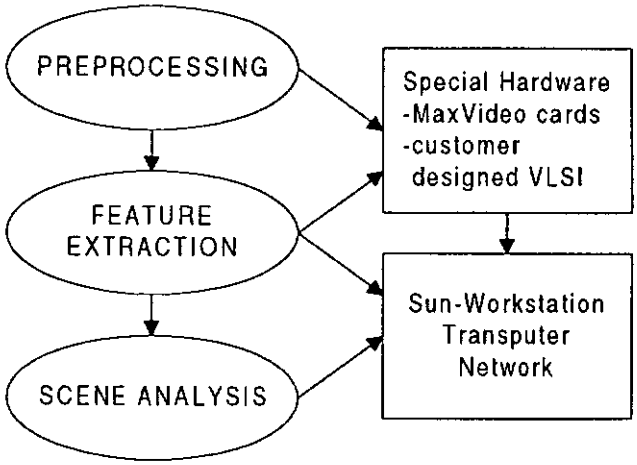


Figure 2.5 Mapping of vision algorithms on hardware

The system has two different modes of pipelined operation. In mode 1, all images generated are buffered in the low-level stage of the pipeline and delivered to the higher-level stages as required. This is needed in many visual inspection applications. In mode 2, the pipelined low-level vision system provides pre-processed data continuously at video rate, while the higher-level stage takes the most recent input data for analysis. The system can dynamically switch to an appropriate mode of operation according to the decisions made by the application program.

The low-level vision system is realised by Max Video machine vision cards and compatible custom-designed modules. The VME bus is used for command and control functions and the MAXbus is for image transfer. The system is based

on a pipelined architecture. Each processing unit in the pipeline reads data from the previous unit and sends its output to the next. The speed of synchronous data transfer through the MAXbus is 10 Mbytes/s. Commercial modules are available for image digitisation, geometric transformations, convolution, morphological operations, arithmetic and logical operations, histogramming and connected components analysis. The high-level vision system is realised by means of transputer network based on a distributed memory model of parallel computation. A transputer-based multiprocessor system can be easily reconfigured to various network topologies, such as ring, mesh, tree, cube, using the programmable interconnection network.

The systems presented in the above paragraphs have a very complicated structure. In all cases, the communication between processing layers is achieved via blocks of shared memory. The use of shared memory induces two problems. First, it is the control. As the memory block is shared between many processing elements from different layers, the control of memory access must be design carefully, to avoid contention and data being over-written. Second, the links with shared memory block must be hard-wired, therefore, the system is less flexible.

In our design, we will employ a network of PCs to execute high-level processing operations so the PCs will be supported by existing networking technology, such as Ethernet [31]. On the other hand, it is also desired to employ the same networking technology as the communication mechanism between the two layers and a network interface for the DSP layer must be provided. This will make our design coherent and easier to build. We will base on two factors, namely, the transmission rate of the mechanism and the feasibility of developing the appropriate interface circuit, for selecting the proper networking mechanism.

2.3 MODERN LAN TECHNOLOGY

With the advance in technology, such as fiber-optics, WDM[32], the data transmission rate of high speed network can now reach several Gbits/s. These kinds of networks meet the bandwidth requirement for our heterogeneous image processing system. In the following paragraphs, different technologies and standards applied in high-speed networks are surveyed. Many of them have similar factors such as low error rate, abundant bandwidth, long propagation delay comparing to the transmission delay, limited processing and data buffering.

In terms of network topology, high speed networks can be classified into different categories. The Ring network and the bus network are two widely accepted categories, and examples are presented in Table 2.1.

Table 2-1 Example of Different LAN Topology

Topology Categories	Sub- Categories	Examples
Ring	Single ring:	IEEE802.5 (token ring), Multiple token ring, and QuickRing
	Dual ring:	FDDI
Bus	Single bus:	Fast Ethernet
	Dual bus:	Distributed Queue Dual Bus (DQDB)

2.3.1 Token Ring and Multiple Token ring

The network topology of a Token Ring network [33] is shown in Figure 2.6. Token Ring network employs the Media Access Control which is known as the IEEE 802.5 standard. In a token ring system, a free token circulates in the idle communication channel. A station with messages to transmit must first capture the free token before it can do so. The token is released by the station when transmission is completed. To determine the maximum token-holding time for a station, a table of time-out values is available and a token number is assigned in the frame headers. The token number is used to index the table of time-out values. Then the values determine the maximum token holding time for the station.

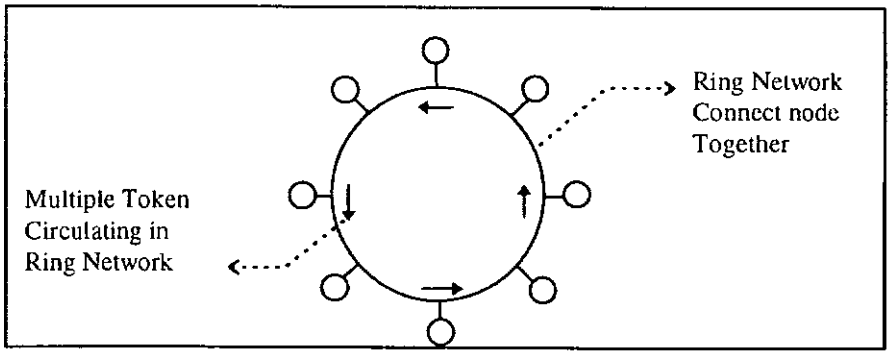


Figure 2.6 Network Topology of Token Ring

However, this arrangement allows only one node to transmit data in one token rotation cycle. With Multiple Token protocol [34], stations are grouped into different logical rings. Each logical ring has its own token passing in the ring. Ideally, the bandwidth of the network is multiplied by the number of logical rings in the network.

2.3.2 FAST-Ethernet

Fast-Ethernet [35], as shown in Figure 2.7, is based on the Ethernet protocol and can achieve a 100Mbits/s through put. It employs the bus topology and the contention access protocols - Carrier sense multiple access with collision detection (CSMA/CD)[31]. Under this scheme, nodes contend for access to the communications channels by an algorithm which may lead to concurrent transmissions among different nodes and cause collisions. Collisions require re-transmissions which waste communication bandwidth. Various schemes are devised to avoid collision, such as channel-sensing algorithms like CSMA and CSMA/CD.

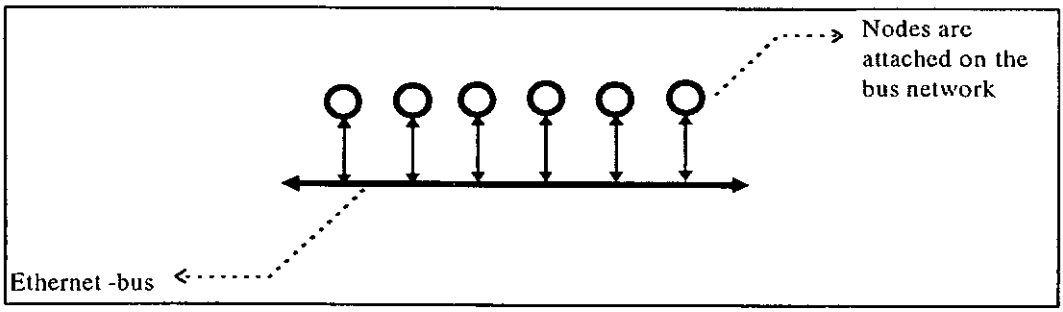


Figure 2.7 Network Topology of Fast Ethernet

CSMA/CD performs poorly in a high-speed transmission environment because of the long propagation delay for channel occupancy for information to be fed back to network users and the long delay for the collided packets to propagate out of the network. Channel-sensing protocols are not practical in a large propagation delay environment.

2.3.3 Distributed Queue Dual Bus (DQDB)

DQDB[37] consists of two unidirectional buses (one forward flowing and the other reverse flowing) and a set of stations with the two end stations on the buses serving as the network control units. Each station can send messages to another station via either of these two unidirectional buses. These two logical buses may be physically configured as a bus or a loop. As shown in Figure 2.8, the loop configuration even offers an excellent self-healing capability. If a link fails,

the DQDB network can recover by connecting the two end stations, breaking the connection between the two stations adjacent to the failing network. Note that the network does not lose any capacity due to this reconfiguration and it still preserves the dual bus topology.

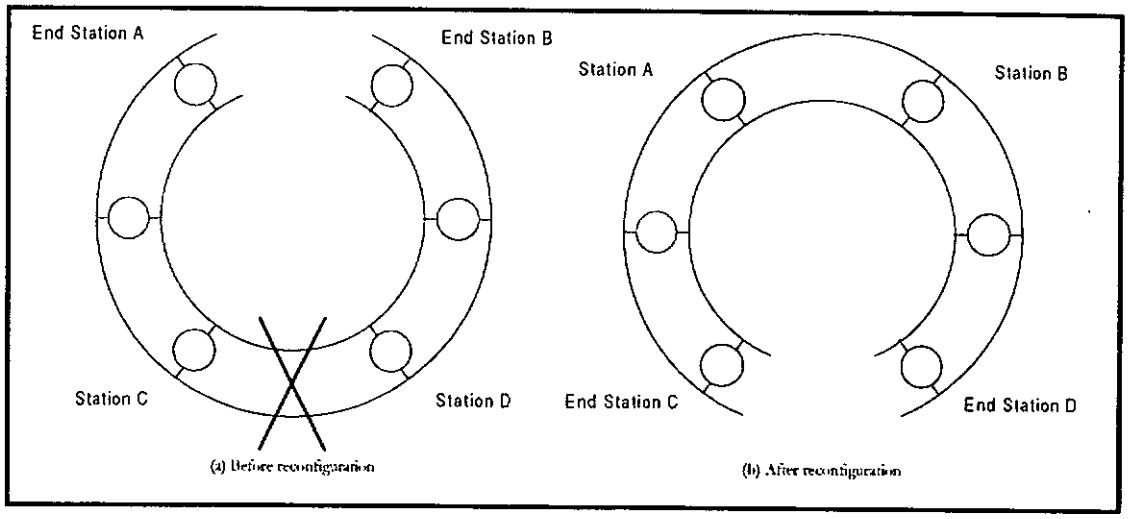


Figure 2.8 Network Topology of DQDB loop

Another special design for a DQDB network to enhance its reliability is to use optical fiber cable. DQDB employs distributed queuing protocol for MAC control and with 53-byte data frame. It offers 150Mbits/s of data transfer rate on each logical bus. So that in total, it can achieve a 300Mbits/s data transfer rate.

2.3.4 FDDI

The fiber distributed data interface (FDDI) [37] is evolved from the IEEE 802.5 token ring protocol. It has been proposed as the American National Standard for fiber optic networks. The basic topology of FDDI consists of two opposite direction fiber optic rings, as shown in Fig. 2.9. Under normal circumstances, only one is in operation while the other ring is used for redundancy purpose to enable network reconfiguration in case of link or node failure.

There are two types of frames: a fixed-length token frame and a variable-length information frame. The token frame contains the token. Only the node in possession of the token is allowed to transmit data. The transmitting node is required to empty its information frame when the frame has travelled around the ring. This also serves as an acknowledgement mechanism. After the transmitting node has finished sending its message, it will release the token to the next node downstream.

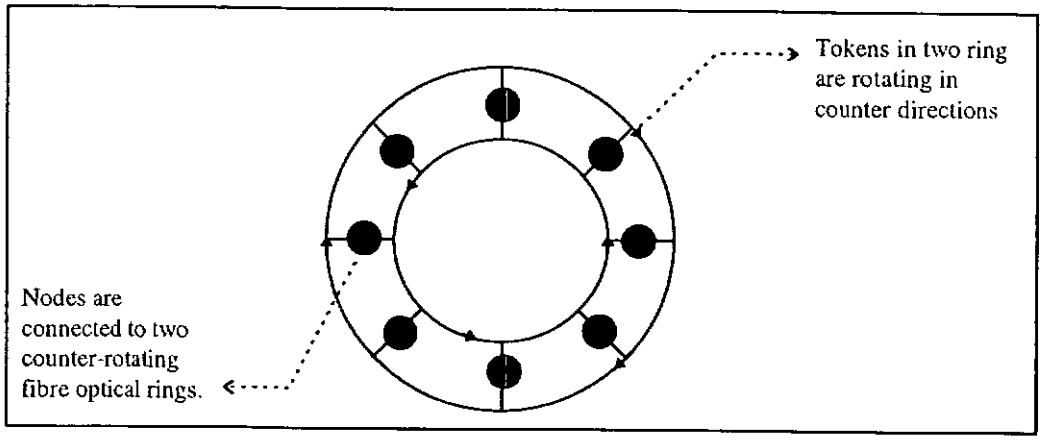


Figure 2.9 Network Topology of FDDI

Data, which is grouped into variable length information frames, is transmitted at 100Mbits/s. The length of a frame depends on the time limit allowed to send messages at the time of transmission. The 4B/5B (5 bits of code for every 4 bits of information) coding scheme is used, so that the actual transmission rate is 125Mbits/s.

FDDI provides a Timed Target Rotation (TTR) protocol for medium access control. The Target Token Rotation Time (TTRT) is decided by a bidding process during initialisation. Two types of services can be supported by this TTR protocol.

1. Synchronous service: reserved bandwidth and maximum token inter-arrival time for users with real-time constraints and predictable bandwidth.

2. Asynchronous service: rest of the bandwidth for users with no real-time requirement and users with bursty traffic.

2.3.5 Asynchronous transfer mode (ATM)

Among a number of candidates for broadband integrated services networks, an architecture based on the asynchronous transfer mode [38] is considered as the target protocol because of its flexibility and simplicity. The key techniques of ATM include fixed-sized cell, asynchronous transfer mode, statistical multiplexing, and lightweight routing and traffic control protocols. ATM offers 154Mbits/s of data transfer rate. ATM is basically a hybrid of packet-switched and circuit-switched network technologies. It is similar to packet switching in that the flow of information is divided into streams of fixed-size cells which are multiplexed and switched in the network. Routing, for cells in the same connection, however, uses the same path.

The advantage of ATM cell-based routing and traffic control is the reduction

of the protocol processing complexity. ATM uses the virtual path concept to simplify the routing protocol and associated network architecture and to reduce the size of routing tables in the transit nodes. The virtual path is defined as a set of multiplexed circuits which terminate at common end nodes, such as switch nodes, network gateways, etc.

Routing using the virtual path concept operates in this way. The network has a set of predetermined virtual paths which have their own routes and capacities. These routes and capacities, however, are changed adaptively to accommodate future traffic demand and to resolve network faults.

The ATM protocol may be described by a layered model. The physical-medium-dependent layer deals with the physical cell transmission and reception. The ATM layer describes the multiplexing and switching functions. The ATM adaptation layer provides the conversion and restoration between the source information bit stream and the ATM cell stream. The high-level service layer supports other services, such as network terminal signalling, information source coding, etc.

The ATM protocol transports information using fixed-size cells. A cell consists of a 48-byte information field and a five-byte header field. User payload is carried on the information field. The header field is used to provide control information, such as the identification of the cell, the properties of the user payload the priority and the length of user payload on the information field, error monitor on the header field, routing and switching information, etc.

In ATM networks, cells are transferred asynchronously to eliminate any synchronisation procedure at the frame, or the cell, levels. Asynchronous transfer also allows the network to accommodate assorted transmission rates. ATM networks shows high efficiency in broadband network access because it uses statistical multiplexing dynamically to distribute the network resources among a wide spectrum of users. Statistical multiplexing has been proved to be more efficient than other multiplexing methods, in that merging traffic reduces burstiness. With this multiplexing scheme, a number of cells should be queued in a common queue for network service. Proper queuing may enhance the performance.

2.3.6 QuickRing

QuickRing [23] is a high-speed communication network designed by Apple Computer, Inc., for applications that require high bandwidth at low cost. The integrated circuit interface for QuickRing, called the QuickRing controller, is produced by National Semiconductor Corporation.

QuickRing provides point-to-point data transmission of up to 160Mbytes/s and allows multiple data streams to flow simultaneously on different segments of a ring. With simultaneous transmissions, the maximum aggregate bandwidth is 1.6GBytes/s. QuickRing differs from a token ring because a controller can begin to transmit as soon as there is space available on the ring. Thus, the aggregate bandwidth is higher than what can be obtained with a token ring.

QuickRing provides the high bandwidth required for multimedia applications, including uncompressed high-definition television signals. QuickRing can also serve as the basis for highly reliable systems that will continue to operate even if some processors fail.

A QuickRing network is formed by connecting a sequence of QuickRing controllers as a ring, as shown in Figure 2.10. A controller is an integrated circuit that acts as an interface to a node or as a bridge between two rings. A bridge is constructed by connecting two controllers back-to-back. The addressing mechanism of QuickRing allows a maximum of 16 controllers per ring, but larger networks can be constructed from multiple rings interconnected by bridges between pairs of rings.

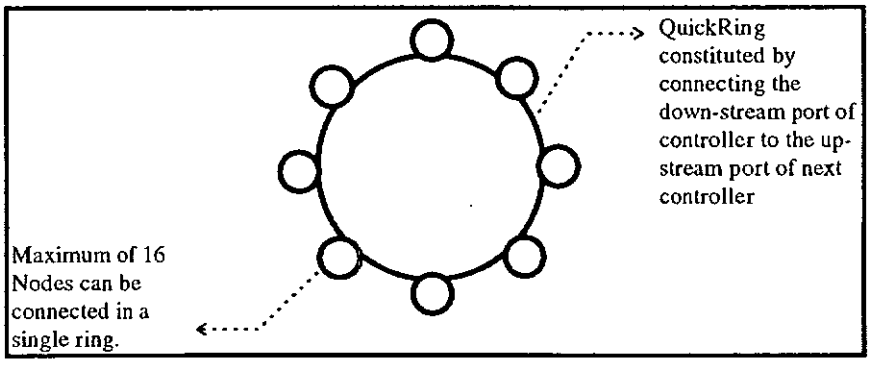


Figure 2.10 Network Topology of QuickRing

2.4 Conclusion

In the above paragraphs, we have introduced several networking mechanisms which could be applied to the high-level processing layer of our system. The important features of those mechanisms are summarised in Table 2.2.

Table 2-2 Summary of the Important Features of Modern LAN technology

Technology	Data Transfer Rate	Advantage	Disadvantage
Token-Ring & Multiple Token-Ring	16Mbits/s for Token-Ring	Low Collision rate even in high utilisation of network bandwidth	Low data transfer rate
FAST-Ethernet	100Mbits/s	Simple, Easy to build up a network	High collision rate, retransmission cause waste of network bandwidth
DQDB	150Mbits/s	Since station is tapped on the optical link, it is easy to isolate the faulty station by bypassing it	Cannot fully utilise the potential bandwidth of Fiber-Optics
FDDI	100Mbits/s	with dual ring connection, network is reconfigure if network node or link fail	Cannot fully utilise the potential bandwidth of Fiber-Optics
ATM	154Mbits/s	with virtual-circuit and virtual-path routing, protocol complexity is reduced	The cost is too high
QuickRing	160Mbytes/s	Achieve highest bandwidth low collision rate	Scale of the network is limited

Referring to Table 2.2, QuickRing provides the highest transfer rate, 160 Mbytes/s, among other high speed network. Moreover, it also supports low collision rate and low data loss rate. As mention in Section 2.2.3, we must also consider the feasibility of applying the selected technology to our system. In the case of QuickRing, the network is realised by using the QuickRing controller which is a flexible device. As described in Section 2.3.6, the controller acts as an interface and converts data between the host and the QuickRing network. From our initial studies, the QuickRing controller supports standard digital signal level from its host and therefore, it can be applied in our design. The drawback of QuickRing is the limited number of hosts that can be supported in the ring network. However, the number of nodes in our system will not exceed the capacity of the network. Considering both criteria, QuickRing is selected for the communication network of our heterogeneous image processing system. In the next Chapter, details of QuickRing are presented and this includes its different data transmission modes, as well as its performance.

Chapter 3

3. QUICKRING TECHNOLOGY

3.1 Introduction

In the last Chapter, a survey on different networking technologies has been presented. Since QuickRing offers the highest data transfer rate (160Mbytes/s), it has been chosen to implement the communication network of our image processing system. As an example, to transmit 30 frames of image, with 512 x 512 pixels and 256 gray scale in each image, the bandwidth requirement is only 7.5 Mbytes/s. The data transfer rate of QuickRing is able to handle the communication requirements demanded by real-time image processing operations. In addition, QuickRing also allows several source controllers to send data to different destination controllers without data collision problems so that efficiency of the communication will not be downgraded by re-transmission. These features make QuickRing outstanding.

In this Chapter, architectural details of the QuickRing controller will be presented. In addition, the data format and the two data transfer modes of QuickRing will be discussed. The performance of a QuickRing network under different transfer modes is studied through simulation. Based on the simulation results, we have identified the most efficient mechanism for our system. Finally, both advantages and disadvantages of QuickRing will be discussed.

3.2 The QuickRing Communication Controller

In Chapter 2, we have compared the QuickRing network with other networking technologies and concluded that the QuickRing network provides the highest communication bandwidth. Therefore, QuickRing is chosen for the development of the two-layer heterogeneous system. In order to design the interface circuit and communication mechanism between the two processing layers and the control components, an in depth studies of the QuickRing controller is necessary.

The block diagram of the QuickRing controller is depicted in Figure 3.1. The controller consists of two interfaces: Client interface and Ring interface. The Ring interface, as its name implies, is used for ring-to-ring communication and it includes an upstream port (Up Port) for receiving data coming from another controller and a downstream port (Down Port) for transmitting data to the next controller in the ring configuration. The Client interface is used for interfacing with external devices, in our application this includes DSP processors and PCs. In addition, the client interface can also be used to connect two QuickRing

controllers for forming a bridge hop, details are given in Section 3.2.2. The client interface consists of a receive port (Rx Port) and a transmit port (Tx Port). As shown in Figure 3.1, data coming from the client is received by the Tx port and is sent via the Down port. On the other hand, data received from the Up Port is delivered to the client through the Rx port.

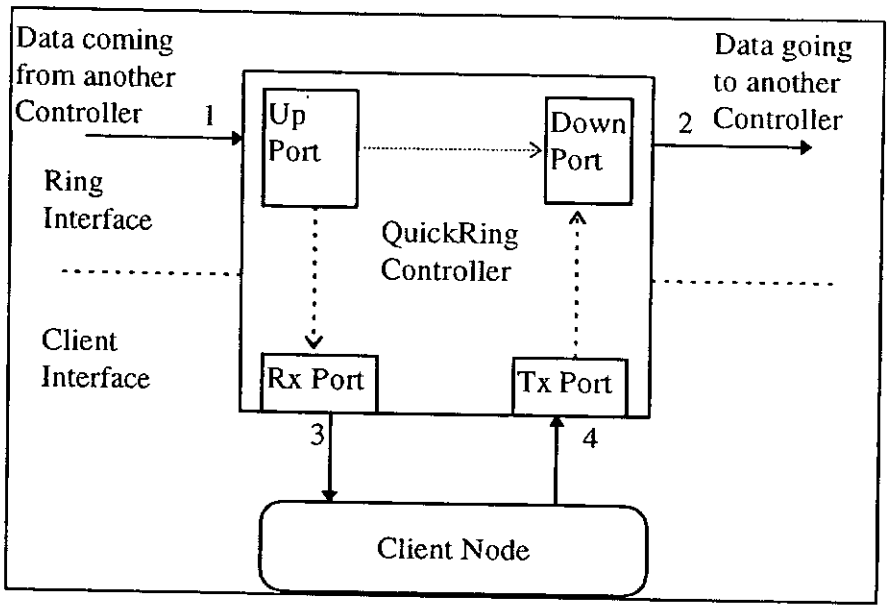


Figure 3.1 The QuickRing Controller

The client sends or receives data in a 32-bit format and is called symbol. A symbol is either a header symbol or a data symbol. Under different communication mode, the header symbol has different formats and details are given in Section 3.2.3.2 and Section 3.2.3.5. Before being transmitted down the ring network, symbols are processed by the QuickRing controller to convert the symbols into the QuickRing format, called packet.

The block diagram of the transmit port is shown in Figure 3.2. The transmit block of QuickRing is formed by: Tx Port, Tx Resynchronizer, Tx Router, and 3 independent FIFOs. All of these blocks form the transmit pipeline and their functions are:

1. The Tx Port is the first stage into the transmit pipeline. The Tx Port is a 4-deep pipeline.
2. The Tx Resynchronizer is a 32-deep asynchronous FIFO in the path between Tx Port and the Tx Router.
3. Tx Router directs the streams to the appropriate FIFO.
4. FIFOs X and Y are meant for handling one independent high bandwidth

stream each, and the LB (Low Bandwidth) FIFO is meant for low bandwidth transmissions.

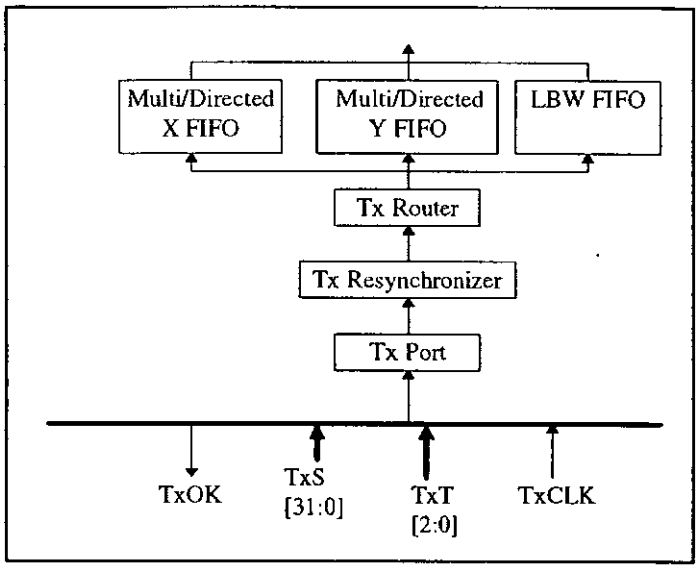


Figure 3.2 Block Diagram of Transmit Port

**The term Low Bandwidth refers to message with only a header symbol and one data symbol. This is mainly for transferring control signal.

When the client starts a transmission, it writes a header symbol followed by a stream of data symbols. The QuickRing Controller (QR1001) receives these symbols through the transmit port and directs them to either the X, Y or LB FIFO. Table 3.1 and Table 3.2 illustrate the format for a header symbol under different communication modes, and Table 3.3 is the definition for terms used in the header. Any header symbol with the CONN field set to 1 is always routed to the LB FIFO, as is every data symbol following such a header symbol. Any other header symbol with the CONN field equal to 0 and all data symbols following

Table 3-1 Fields in Transmit Port Header Symbol in Multicast mode

Field-name	Acc	Conn	Srce	Group Field	Multicast Field
bit-number	31:30	29:28	27:24	23:16	15: 0

Table 3-3 Fields in Transmit Port Header Symbol in Directed mode

Field-name	Acc	Conn	Srce	Trgt	HOP1	HOP2	HOP3	HOP4	HCNT
bit-number	31:30	29:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0

such a header symbol are routed to either the X or Y FIFO.

Table 3-2 Tx and Rx Port Symbol Field Definitions

Acc	The access code indicates the type of access symbol. Voucher, ticket, null and abort. This field is valid only in the ring ports, upstream and downstream. At the client ports ACCess field should be [00].
Conn	The connection code provides two types of transmission, normal and low bandwidth, Low-bandwidth streams are transmitted with higher priority.
Srce	The source field contains the node ID of the source of the associated payload.
Trgt	In Directed mode, the target field contain the node ID of the target of the associated payload.
HOP1 HOP2 HOP3 HOP4	In Directed mode, they distinguish between unique streams whose source-to-target, routes are identical. In a multiple-ring topology, they supplement source and target ID fields to route streams as they hop from ring to ring
HCNT	In Directed mode, it enables a bridging node to be address as a node. It will be decrement when the packet passes through a bridge hop. If it's value equal Fh when received at a bridge then the stream will be absorbed and not passed on to the bridged ring.
Group Field	In Multicast mode, it is available to designate multicast targets in rings beyond bridge hops. The bridge has the intelligence to look up the group in a table and set the appropriate multicast bits for the next ring.
Multicast Field	It has 16-bits, one for each node in the local ring. If the bit corresponding to the node address is set and there is Target FIFO space available, then that node will absorb the multicast packet, clear the bit and pass it downstream.

Figure 3.3 depicts the structure of the receive block. Functions of the block are to convert data from the ring format into symbols. The function for different components inside the block is given below.

1. The Header Stripper removes all header symbols except those identifying the beginning of a stream.
2. The Target FIFO reserves space for 3 normal packets and 6 LB packets.
3. The Rx Resynchronizer is an 8-deep FIFO in the path between Target FIFO and the Rx Port.
4. The Rx Port is the last stage between a data stream and the client interface.

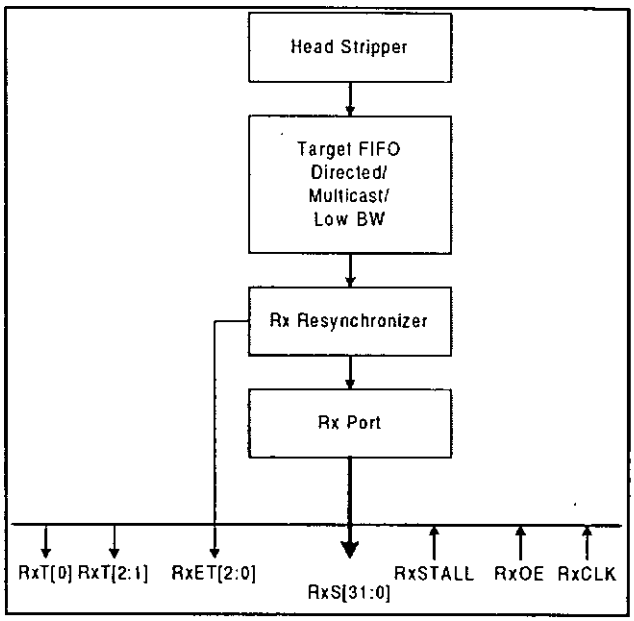


Figure 3.3 Block Diagram of Receive Block

3.2.1 The Ring Interface

The Ring interface performs three major functions. First, it receives data coming from other controller via the ring network from the Upstream port. Second, it transmits client's data. The last function is to forward data received from the Upstream Port to the next controller.

As shown in Figure 3.4, the Upstream Router/Multicast Handler, LB Target Handler, Directed Target Handler, and the Ring/Multicast FIFO constitute the forwarding path. The LB Target Handler processes LB vouchers targeted to this node into tickets. Voucher and ticket are protocols used in the reservation based communication mechanism, details can be found in Section 3.2.3.1. These tickets are forwarded to the source node through the downstream port. The Target handler processes vouchers targeted to this node and generates tickets for the vouchers. These tickets are forward to the source node through the downstream port. The Ring/Multicast FIFO stores incoming data from the upstream port that is intended to be forwarded to other nodes on the ring, when the downstream pipe is allocated to launch local data generated by this node.

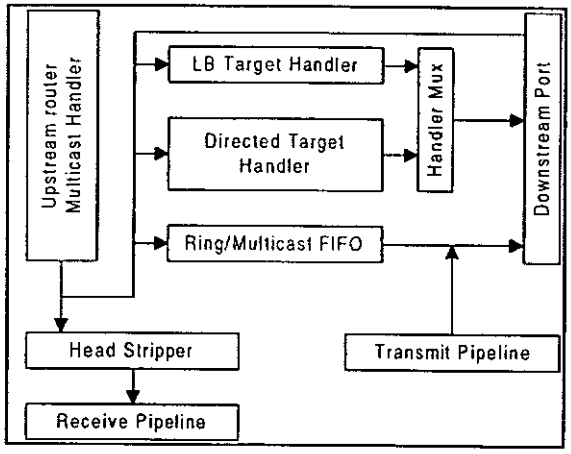


Figure 3.4 Block Diagram of Ring Interface

Since the ring path is a 6-bits wide channel, symbol is packed as 7 sub-symbols with 6 bits wide. 7 sub-symbols are transferred on every clock cycle. Table 3-3 illustrates the format for a symbol applied in the ring port. Symbol consists of an addition of 2-bits for the type field, 1-bit of frame field and 7-bits of EDC field. The type field indicates whether the symbol is a header or a payload symbol. The value of type field is input from and output to the client through type port and is added to the symbol of ring interface. Frame indicates the symbol is data or frame. The EDC field is provided for error detection purpose. Values of other fields will be put into the data field of the symbol according to their bit number. For example, ACC field in Table 3-1 and Table 3.2, has bit number 31:30, will be put to data bit 31 and 30.

Table 3-3 Symbol format in Ring Transmission

	Sub-symbol 1	Sub-symbol 2	Sub-symbol 3	Sub-symbol 4	Sub-symbol 5	Sub-symbol 6	Sub-symbol 7
Ring data channel 1	Type bit 1	Data bit 28	Data bit 22	Data bit 16	Data bit 10	Data bit 4	EDC6
Ring data channel 2	Type bit 0	Data bit 27	Data bit 21	Data bit 15	Data bit 9	Data bit 3	EDC5
Ring data channel 3	Frame bit 0	Data bit 26	Data bit 20	Data bit 14	Data bit 8	Data bit 2	EDC4
Ring data channel 4	Data bit 31	Data bit 25	Data bit 19	Data bit 13	Data bit 7	Data bit 1	EDC3
Ring data channel 5	Data bit 30	Data bit 24	Data bit 18	Data bit 12	Data bit 6	Data bit 0	EDC2
Ring data channel 6	Data bit 29	Data bit 23	Data bit 17	Data bit 11	Data bit 5	EDC7	EDC1

On the ring path, upstream and downstream ports, type and symbol fields organise data transmissions. Data on the ring flows in bounded streams called packets. Before data enters the ring, packets are formed by each controller internally. As shown in Figure 3.5, packets have one header and one or more payload symbols. Since multiple independent packets can be found inside one controller and multiplexed at the downstream port with a type field accompanying each symbol. Inside a controller, packets can be found that may originate from any other node on the system. The type field marks each symbol as a header, payload, tail, or access.

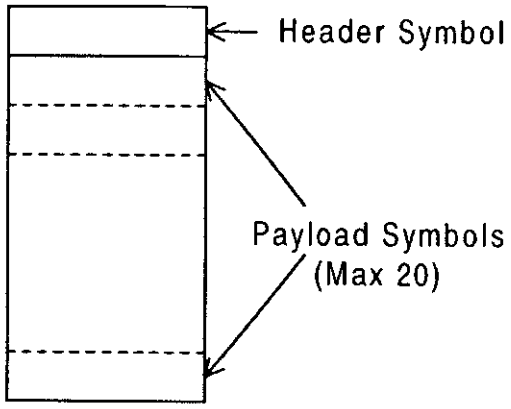


Figure 3.5 Formation of Packet

At the transmit and receive ports the length of a data stream that is uninterrupted by a header is unbounded. On the ring, upstream and downstream ports, data is bounded; there is an upper bound that gives the concept of a packet. There are two types of packets: normal and low bandwidth (LB). There is a ring protocol defining the symbol sequence. For normal packets, the maximum number is fixed at 20 symbols. The largest packet is 21 symbols in all; however, packets may be less than 21 symbols. The LB packet consists of a Header symbol and one data/Tail symbol. Since priority is given to LB packet, it is usually used to transmit user defined control message. QuickRing internally assembles packets from the data that the client writes into the transmit port. A header symbol precedes the packet and the last payload symbol of a packet is specially marked as the tail of that packet.

QuickRing organises data with a combination of access symbols and packet symbols. Access symbols are vouchers, tickets, nulls and aborts. Packet symbols are those that form packets such as heads and payloads. These symbols can also be grouped into routing symbols and payload symbols. Routing symbols hold source and target addresses such as voucher and ticket for access symbols, and headers for packet symbols. Payload symbols are data or frame; they hold the information that clients are trying to transmit. At the ring ports, a type field (01) represents a data symbol and a type field of (10) stands for a frame symbol. A data or frame

symbol at the ring ports is distinguished by an additional frame bit. The payload symbol, frame or data, at the end of a packet is called a tail. It is encoded such that the type field or frame symbol at the ring ports is distinguished by an additional frame bit.

3.2.2 Network architecture

The QuickRing controller forms the link to other nodes with a point-to-point architecture. The ring ports, are 6 bits wide plus a clock and it is implemented using Low Voltage Differential Signaling [23]drivers and receivers. The ring, formed by attaching Up ports and Down ports of adjacent QuickRing controllers, carries one 42 bits symbol in every clock cycle. To transmit 42 bits in one clock cycle, QuickRing divides the 42 bits symbol into 7 sub-symbols with 6 bits wide. The controller then multiplexes the sub-symbols onto the 6 LVDS channels on the downstream port.

The addressing mechanism of QuickRing allows a maximum of 16 controllers per ring, but larger networks can be constructed from multiple rings interconnected by bridge hops between pairs of rings as depicted in Figure 3.6. Figure 3.7 and Figure 3.8 illustrate the structure of a bridge hop under different communication modes. As shown in Figure 3.7, a bridge hop is constructed by connecting two controllers back-to-back. However, the structure of the bridge hop has to be modified in Multicast mode. As depicted in Figure 3.8, an intelligence address conversion mechanism has to be inserted between the two controllers to look up the group in a table and set the appropriate multicast bits for the next ring. Details of address conversion mechanism are presented in Section 4.2.5.

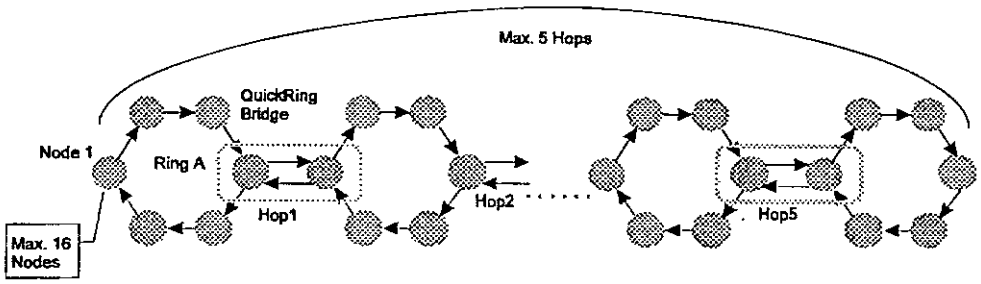


Figure 3.6 Multiple rings interconnect through bridge hop

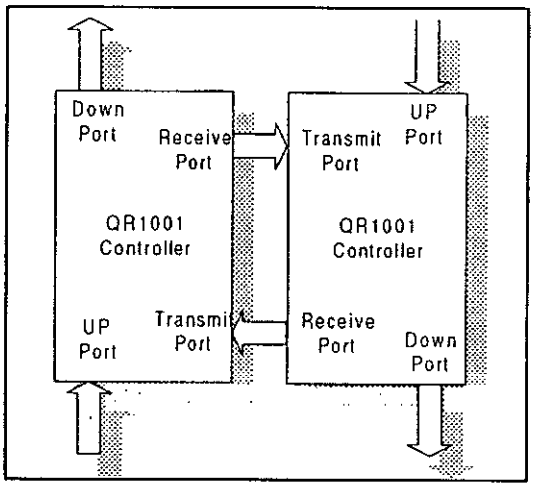


Figure 3.7 Bridge Hop of Directed mode

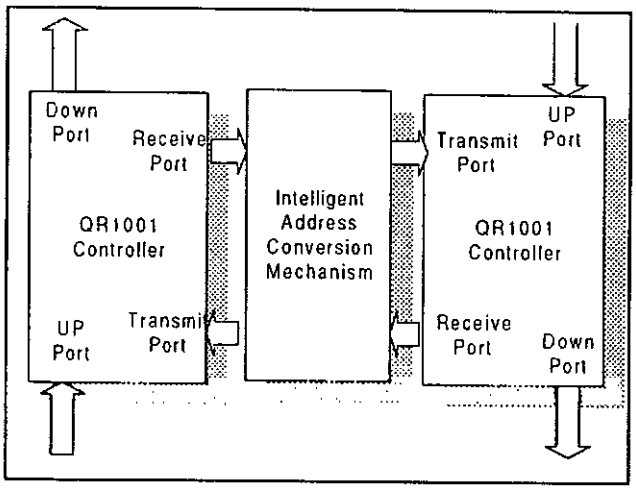


Figure 3.8 Bridge Hop for Multicast mode

3.2.3 Data Transfer modes

The QuickRing controller supports two data transfer modes. They are the Directed mode and the Multicast mode. They have different performance in terms of reliability and bandwidth utilisation. It is important that both mechanisms must be studied, in order to determine the most appropriate mechanism for our system. Before, we describe and analyse the different modes, symbols which are applied in our studies are listed in Table 3-4.

In our studies, we considered a system that consists of two layers. The system includes two QuickRing networks, see Figure 3.9. There are networks simulating the low-level processing layer and the high-level processing layer. The last network consists of a master controller, a bridge hop and a low-level controller. The bridge hop is constructed by connecting two controllers back-to-back. It is connected to the second QuickRing which connects *N* nodes of high level layer processors. All symbols transferred between low level layer controller and high level layer processors have to pass through the bridge hop.

Table 3-4 Parameter Definition

N:	Number of High level Processors in the ring
n:	Position of destination processor within the ring
S:	Number of data Symbols being transmitted
H:	Number of extra Header Symbols
N_{packet}:	Number of packets being transmitted
$Size_{frame}$:	Size of Image (bytes) = 262144
$Size_{packet}$:	Size of packet (bytes) = 80
N_{ps}:	Number of data Symbols in one packet = 20
N_{total}:	Total number of Symbols
$T1$:	The time (clock) consumed on transmitting the first three packets in Directed mode
$T2$:	The time (clock) consumed on transmitting every following three packets in Directed mode
$Ttotal$:	The time consumed on transmitting one image frame
Td_{25}:	The time consumed on transmitting 25 image frames in Directed Mode
Tm_{25}:	Time consumed on transmitting 25 image frames in Multicast Mode
$Tdelay$:	The propagation delay

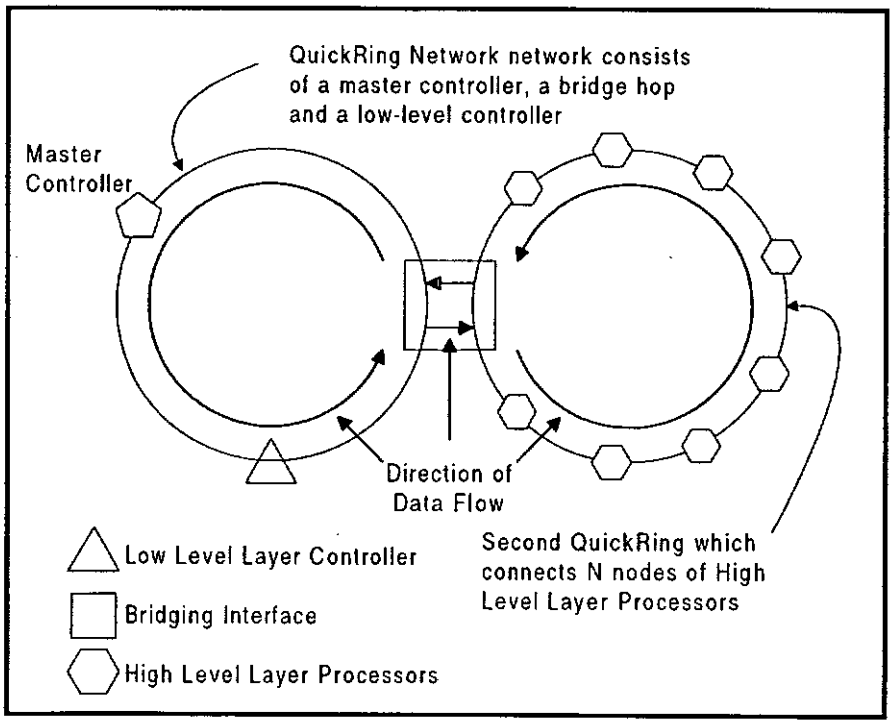


Figure 3.9 The Network Configuration for Performance Studies

In our studies, we considered the computing requirements demanded in real-time processing applications. In real-time applications, 25 image frames have to be processed per second and each image frame consists of 512x512 pixels, with 8-bit data per pixel. An image frame, after processed by the low-level layer, will be divided into packets before transmitting to the high level layer. In each packet, data are divided into 20 data-symbols and 1 extra header-symbol for addressing. Each symbol contains 4 bytes of information. One symbol can be transmitted within one clock cycle, that is 25 ns for QuickRing operating at 40MHz.

The total number of packets required to transfer an image frame, N_{total}, can be derived by the following equations:

$$\text{Number of Header Symbols, } H = N_{packet} = \frac{Size_{frame}}{Size_{packet}} \quad (1)$$

N_{packet} is an integer and $\frac{Size_{frame}}{Size_{packet}} = 3276.8$

$$\therefore N_{packet} = 3277$$

$$S = N_{packet} \times N_{ps} = 65540$$

$$N_{total} = S + H = 68817 \quad (2)$$

In order to transmit data from the low-level layer to the high-level layer, there are two kinds of mechanisms that can be applied, namely Directed mode and Multicast mode.

3.2.3.1 Directed mode

The Directed mode provides a point-to-point buffer reservation mechanism. When the communication medium is very reliable, the major cause of packet loss is buffer overflow. To avoid buffer overflow, QuickRing provides a buffer reservation scheme. Before transmitting a packet, the source controller must reserve a buffer in the target controller. This is achieved by first transmitting a voucher to the target controller. Voucher is only one symbol long. If buffer is available in the target controller, it will reserve the buffer for the packet and return a ticket, which is also one symbol long, to the source controller. By receiving the ticket, the source controller will transmit the packet. On the other hand, when buffer is not available then the voucher is returned to the source controller. If the source controller receives its own voucher, it may re-try, after a hundred clock cycles. The receiver can only mark the buffer as free after receiving the tail symbol of the packet.

The receiver interface has three buffers, each 20 symbols long, for storing packets, received upstream, that are addressed to the local client. Thus, for each controller, at most three tickets can be issued at any time. If a controller has already issued three tickets and has not yet passed the corresponding packets to the client node, it returns an arriving voucher. The transmitter interface has two 20 symbols long buffers which are used to store packets that are waiting for a ticket to be launched.

Communication on the QuickRing network is achieved through a stream of data symbols transmitted from source controllers to destination controllers, see Table 3-3. A symbol consists of 42 bits and 32 bits are used to carry user defined information. In addition, 7 bits are reserved for error control and 3 bits for data type definition. The type information is used to define the packet as data symbol, header symbol or control symbol such as voucher and ticket.

The data symbols are grouped together to form packets. Each packet consists of 1 to 20 data symbols and a header symbol. The header symbol has 10 fields, see Table 3-5. The Type field defines the symbol as being a header symbol; the next two fields are ACC and CONN fields. The following fields, Source, Target, Hop1, Hop2, Hop3, Hop4, HCNT, are identifier fields that provide the address of the destination. Since each field has 4 bits, only 16 destinations can be represented in one ring. The last data symbol of a packet is marked in the Type field as a tail symbol.

Table 3-5 Header Symbol of Directed Mode in Ring Transmission

	Sub-symbol 1	Sub-symbol 2	Sub-symbol 3	Sub-symbol 4	Sub-symbol 5	Sub-symbol 6	Sub-symbol 7
Ring data channel 1	Type bit 1	Conn	Trgt	HOP1	HOP3	HOP4	EDC6
Ring data channel 2	Type bit 0	Srce	Trgt	HOP2	HOP3	HCNT	EDC5
Ring data channel 3	Frame bit 0	Srce	Trgt	HOP2	HOP3	HCNT	EDC4
Ring data channel 4	Acc	Srce	HOP1	HOP2	HOP4	HCNT	EDC3
Ring data channel 5	Acc	Srce	HOP1	HOP2	HOP4	HCNT	EDC2
Ring data channel 6	Conn	Trgt	HOP1	HOP3	HOP4	EDC7	EDC1

3.2.3.2 Address and Routing mechanism

In Directed mode, routing is very simple on a single QuickRing. To transmit a packet, the source controller places the 4 bits identifier of the destination

controller in the Target field of the header symbol and its own 4 bits address in the Source field. The remaining fields are not used for communication on a single ring unless different data streams with the same source and target controllers, must be distinguished.

To transmit a packet to a controller on a different ring, the source controller includes in the Target and Hop fields of the routing symbol the addresses of all bridges to be crossed. As depicted in Figure 3.10, the QuickRing controller automatically shifts the position of the routing fields as the stream header is delivered at the Rx Port. Since the bridge hop of Directed mode is built by back to back connection of two QuickRing Controllers, the header symbol is modified as it crosses a bridge in such a way that, at each step along the path, the Target field contains the address of the destination on the ring for that step. As a routing symbol crosses a bridge, its fields are shifted into the Target field and the bridge identifier on the next ring is placed in the Source field. When a packet reached its final destination, the return address of the source is available to the destination.

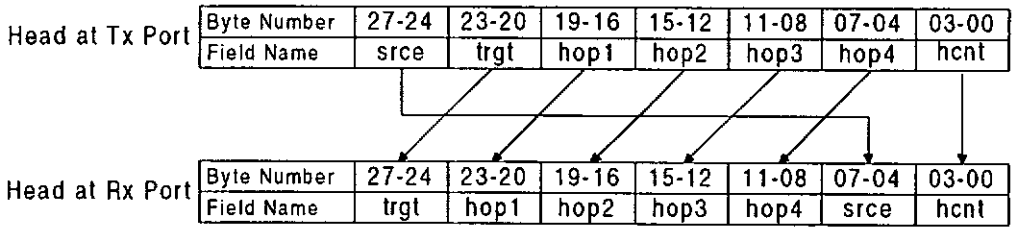


Figure 3.10 Shifting of Address Fields when header pass through Rx Port

3.2.3.3 Performance Model

In the previous paragraphs, we have discussed the mechanism applied in the Directed mode of communication. In this section, we will present a timing model for the system under the Directed mode. In Directed transmission mode, a voucher is generated and sent to the destination node when a packet reaches the data FIFO of the QuickRing controller from the client (*Tgv*) (refer to Table 3-6). If buffer is available, the destination node will reserve it for the packet and return a ticket (*Tgt*) to the source. The source node then transmits the packet to the destination. The data FIFO inside the controller can contain 2 packets and 2 symbols. For this reason, the first, second and third voucher will be generated simultaneously. After the arrival of the tickets, packets in the outgoing FIFO are transmitted. The controller will receive packets from the client and vouchers will be generated again. As a result, the clock cycles required to transmit every 3 packets are constant, with the exception for the first 3 packets due to the initial setup time.

Table 3-6 Typical Ring Latencies [23]

Description	Symbol	Clock Cycles
Time between the controller receive the data from DSP and the voucher be generated,	*Tgv*	13
Time between the controller receive the reply ticket and launch the packets, which is longer than time for the data transverse from the FIFO to the ring port	*Tgp*	6
Time for ticket or voucher to bypass a node,	*Tvb*	4
Time between the destination node receive the voucher and launch the reply ticket,	*Tgt*	2
Time for data symbol to bypass a node,	*Tpb*	5
Time between the destination node receive the data and transmit them to client,	*Tpt*	13
Time delay between the QuickRing controller receive the data from client and transmit the packet to ring port	*Tp*	19
Time for any symbol to pass through a bridge hop, < Tgv + Tgp + Tpt,	*Tph*	32

The clock cycles consumed on transmitting the first three packets is

$$T1 = 142+3N+5n \tag{3}$$

The clock cycles required to transmit every 3 packets, except the first three packets, is T2=84

Total clock cycles required for 25 image frames

$$Td_{25} = 25\times\left[T1+\left(T2\times\frac{H}{3}\right)\right] \tag{4}$$

$$< 2297450+75N+125n$$

Total time required to transfer 25 image frames is,

$$Td_{25} = 5.74\times10^{-2} + 1.88\times10^{-6}N + 3.13\times10^{-6}n, \tag{5}$$

This includes the setup times, delays, delays related to the total number of high level processors and delays caused by the position of destination processor in

the network. Figure 3.11 illustrates the results derived from Equation (5).

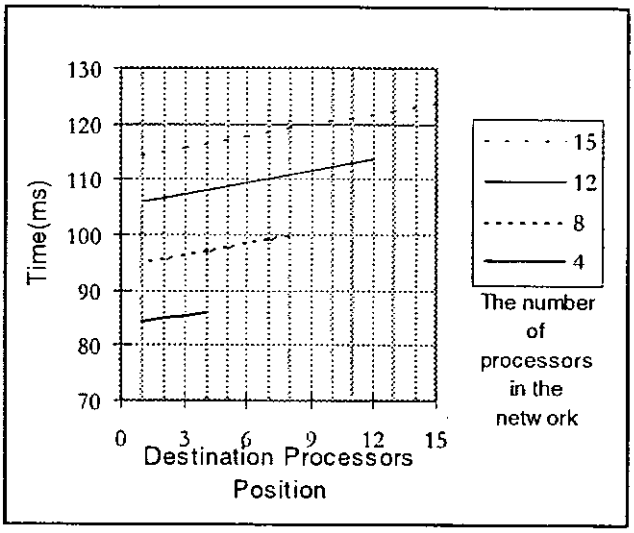


Figure 3.11 Calculated Data Transmission Time in Direct Mode

3.2.3.4 Multicast Mode

The Multicast mode is used to send non-reservation based streams to a subset of nodes on the system. If all nodes are in the subset, then data is broadcast. Non-reservation means that the voucher and ticket FIFO reservation protocol is turned off. This reduces ring latency but makes transfer unreliable.

In multicast mode, no buffer reservation procedure is performed during data transmission. Therefore, the time spent on transmitting voucher and waiting for the ticket is eliminated. This may cause problem when many data sources intended to send data to the same destination. Data of the next coming packet may not be received by the destination node.

3.2.3.5 Data format and Address Mechanism

In Multicast mode, header symbol of a packet is used to define the delivery pattern. As shown in Table 3-1, there is a group field and a multicast field. The Multicast field has 16 bits, one for each node in the local ring. If the bit corresponding to the node address is set and Target FIFO space is available, then that node will absorb the multicast packet, clear the bit and pass it down stream. If there is no FIFO space, the bit is cleared and packet is forwarded only. If the bit is not set, the node only passes the packet downstream. If all bits are cleared in the Multicast Field, then the stream is eliminated from the ring.

3.2.3.6 Routing mechanism

When the messages are multicasting to nodes in other ring, the corresponding bit of the bridge hop must be set so that the bridge hop can transfer the message to other ring. When a multicast message passes through a bridge hop, the modifications of the group field and the multicast field are handled by an external conversion table. The design of the conversion table for our system is presented in Chapter 4.

3.2.3.7 Performance Model

In Multicast mode, the packets are transmitted continuously. When the data is transmitted from the processor, it will be loaded into the client port of the QuickRing controller. The QuickRing controller will then transfer this data to the ring via its ring port. The time required for this operation is represented by *Tp* (refer to Table 3-6).

The packets will pass through the bridge hop, and circulate in the ring consists of high level processors. All the specified nodes in this ring will receive the packets. Since the QuickRing supports unidirectional data flows, the delay for the packets to bypass the nodes before it is received by the last node is $Tph + Tpb \times (N - 1)$. The propagation delay is equal to:

$$Tdelay = Tp + Tph + Tpb \times (N\text{-}1) + Tpt. \quad (6)$$

Let *Treceive* be the total clock cycles required to receive all symbols,

$$Treceive = S + H \quad (7)$$

This is equal to the total number of symbols, including data and headers, because the client processor requires 1 clock to receive 1 symbol.

The total clock cycles required to transmit a frame from the low level layer controller to the high level processors equal to the propagation delay plus the clock cycles required to receive all symbols. Hence,

$$Ttotal = Tdelay + Treceive = 68876 + 5N. \quad (8)$$

Total clock cycle required to transmission 25 frames of data is:

$$Tm_{25} = Ttotal \times 25 = 1721800 + 125N \quad (9)$$

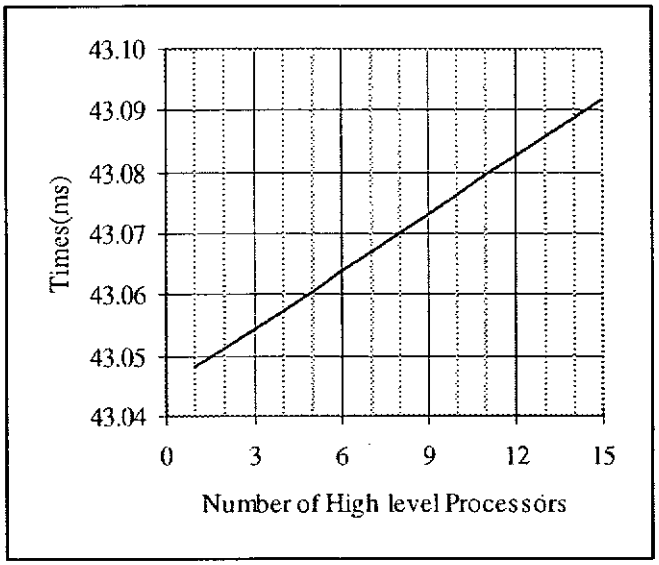


Figure 3.12 Calculated Data Transmission Time in Multicast Mode

Converting the unit of (9) into (sec), we get:

$$Tm_{25} = 43.045\times10^{-3} + 3.125\times10^{-6}N \qquad (10)$$

Figure 3.11 and Figure 3.12 show the times required to transmit 25 image frames under the different communication modes using model given in (5) and (10). In Figure 3.11, every line represents the performance of the network with different amount of processors. In Directed mode, the time required to transmit image frames increases with the total number of high level processors connected to the network (refer to Figure 3.11) even the position of destination processor is unchanged. However, this is not important in Multicast mode. In Multicast mode (refer Figure 3.12), the time required to complete the transmission is much shorter than that in Directed mode.

3.3 Simulation and Results

In order to verify the model presented in Section 3.2, we have used software packages to simulate the behaviour of the QuickRing network. A simulation programs is developed to simulate Multicast Mode and Directed Mode. With the program, we have simulated the performance of the network with 2 different topological configurations as illustrated in Figure 3.9 and Figure 3.13. In our simulation, we have assumed that there are 15 High Level Processors in the processors network.

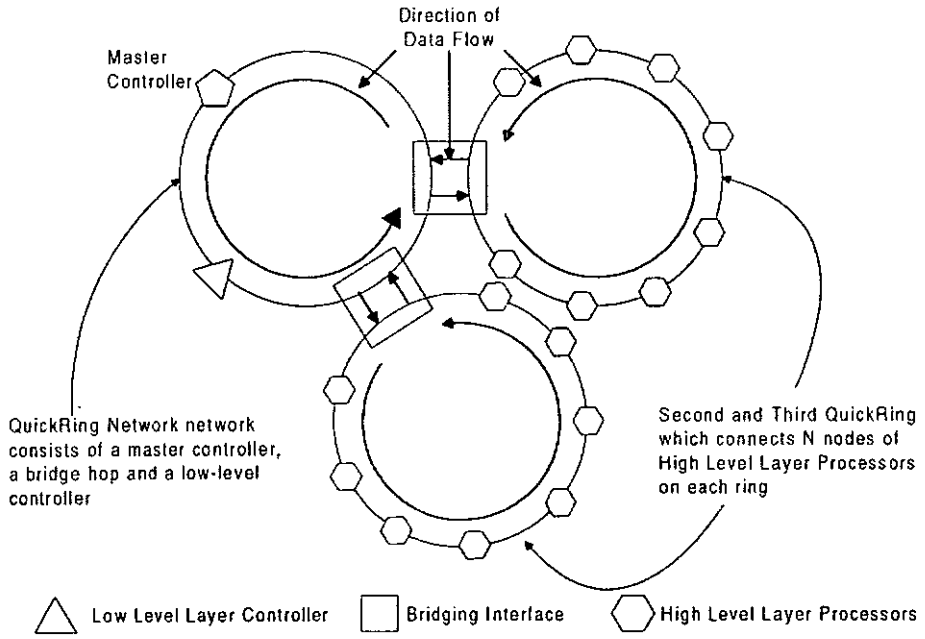


Figure 3.13 The Network Topology of The Expanded Heterogeneous Image Processing System

Figure 3.14 illustrates the transmission time to transmit 25 image frames in Directed Mode based on the software simulator under the topology shown in Figure 3.9 and Figure 3.13. The time difference between sending data to the second High Level Processors Ring and to the first High Level Processors Ring is too small and cannot distinguish in the chart. The slope of transmission time line suddenly increases if the destination is greater than 5 nodes away. We have found that this slope increases when the sum of the number of nodes and the destination node is larger than 20. The slope of the line segment *AB* (refer to Figure 3.14) is given by:

$$\text{Slope}_{AB} = \left[\left(\frac{N_{packet}}{3} \times 2 + 4 \right) \times 25 \right] / 40Mhz \qquad (11)$$

Transmission Time for 25 Frame in Directed mode, Td_{25} can be found by equation:

$$Td_{25} = \begin{cases} 5.74 \times 10^{-2} + 1.88 \times 10^{-6} N + 3.13 \times 10^{-6} n, if\ N + n \le 20 \\ 5.74 \times 10^{-2} + 1.88 \times 10^{-6} N + 1.37 \times 10^{-3} n, if\ N + n > 20 \end{cases} \qquad (12)$$

Figure 3.15 depicts the transmission time in Multicast mode. The results show that the transmission time in Directed mode is longer than in the Multicast mode. Therefore, Multicast mode is more effective for the proposed two-layer image processing system.

3.4 Conclusion

QuickRing is a ring-based communication network that provides a high data transfer rate. The internal architecture of the QuickRing controller, the data formats, Directed data transmission mode and Multicast mode have been introduced. The design of QuickRing limited the number of nodes in a ring to only 16 but this is not significant in our current design since our system does not have many nodes. The design of the system architecture for the proposed system will be presented in Chapter 4.

QuickRing offers two different data transmission modes. Directed mode provides a voucher/ticket buffer reservation protocol that ensures reliable data transmission. Multicast mode provides broadcast capability and reduces latency in data transfer. Multicast mode of the QuickRing can highly utilise the bandwidth of the network. Therefore, it will be applied in our communication network in order to achieve better performance. However special hardware, control and communication protocol have to be designed to increase the reliability of the system.

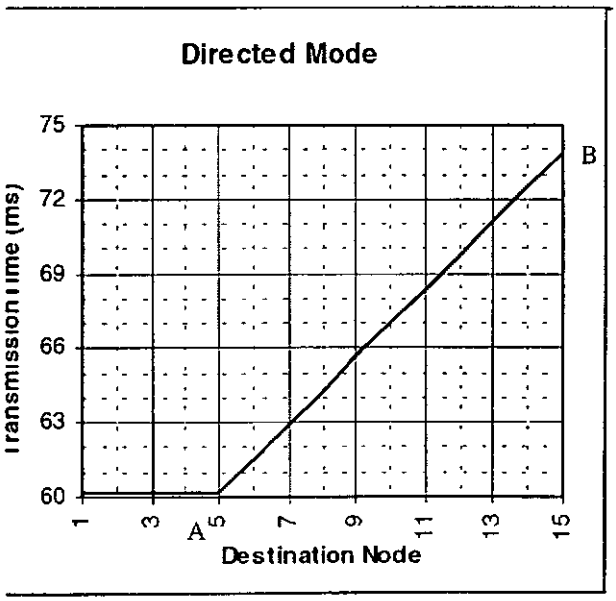


Figure 3.14 Data Transmission Time in Directed Mode

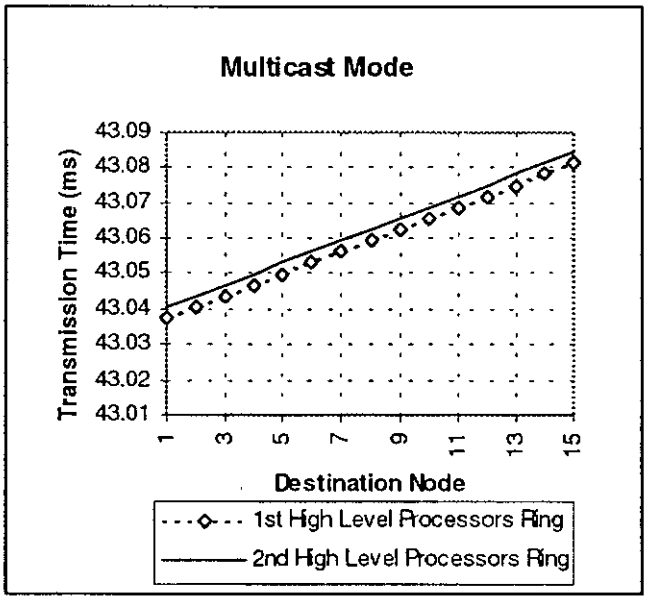


Figure 3.15 Data Transmission Time in Multicast Mode

Chapter 4

4. HARDWARE DESIGN

4.1 Overview

In the last Chapter, we have discussed the internal structure of the QuickRing controller as well as the communication mechanism provided by QuickRing. Based on our studies, we have identified that the multicast mode of communication is more efficient when comparing with the Directed mode, therefore, it will be applied in our system.

In this Chapter, the architectural design of the proposed two-layer system is presented. We will concentrate on the communication mechanism and components of the system that are devised to enhance the communication efficiency.

4.2 System Architecture

As discussed in Chapter 2, image processing and machine vision problems have different computational requirements at different stages of the process. A heterogeneous system, which consists of different types of computing devices, provides an effective solution for solving these problems. Moreover, in many applications [39][40], a continuous stream of images has to be processed, therefore, a heterogeneous system provides functional parallelism for the image pipeline. As presented in Chapter 2, there are different heterogeneous systems designed for solving vision problems.

Our proposed system, as mentioned in Chapter 1, consists of two layers. In the first layer, it is an array of DSP processors connected as a ring and it is applied in solving image-based operations. For the second layer, it consists of a network of powerful personal computers, such as Pentium II machines, and this layer is targeted for symbolic operations. In addition to the processing layers, there are control and communication components. Based on the QuickRing technology, our proposed system has an architecture as depicted in Figure 4.1.

As shown in Figure 4.1, the major components in the system include the low-level layer, the high-level layer, the Master Controller (MC), the Bridge Hop, and the DSP layer controller (DLC). The Master Controller, the DSP layer controller and the Bridge Hop are also connected as a ring network which is called the Control and Communication Network (CCN).

The Master controller, as its name implies, is responsible for distributing processing tasks to the layers. In addition, it also acts as the user-interface and providing image data to the processing layers. The Bridge Hop is the device for connecting two QuickRing networks together, details regarding its structure are presented in Section 4.2.4. The DSP controller provides an interface between the DSP layer and the QuickRing network. In our design, it consists of two DSP processors and a QuickRing controller and the data transfer mechanism between the DSP layer and QuickRing network is given in Section 4.2.3.

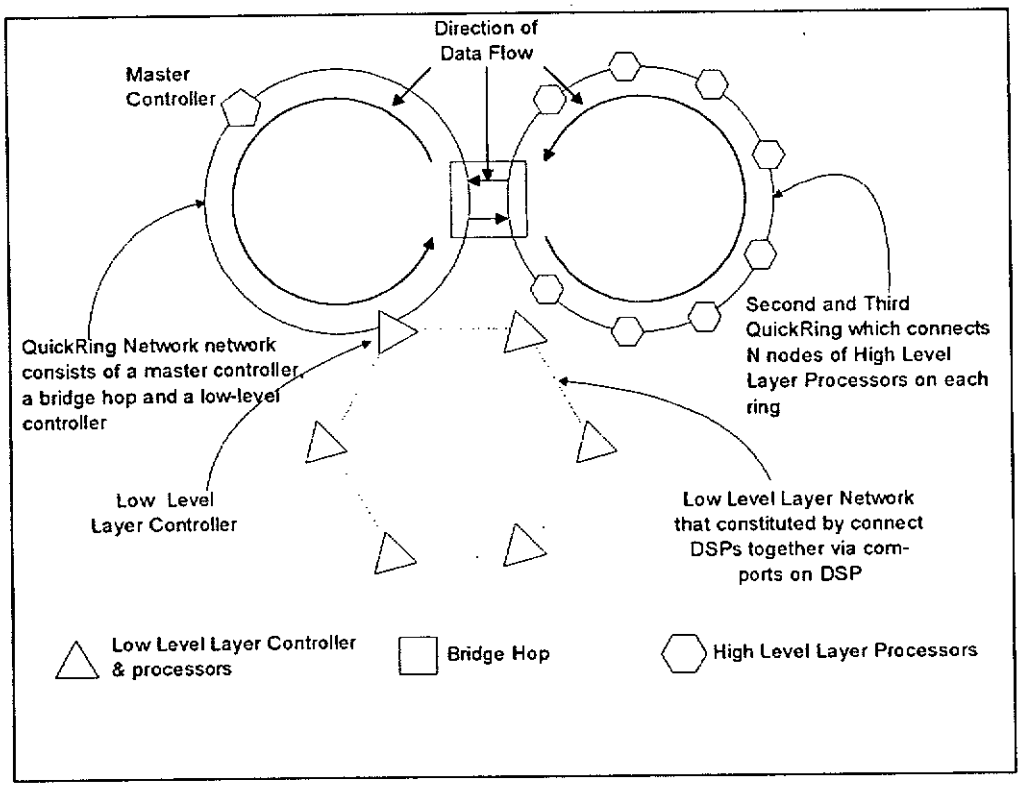


Figure 4-1 The Block Diagram of The Heterogeneous image Processing System

4.2.1 Low-level Layer

The low-level layer consists of an array of DSP processors connected in a ring network, as shown in Figure 4.1. The advantage of linear array is its simplicity and this allows us to expand the system very easily. Moreover, the structure is easy to implement with the latest DSP which has built-in communication links, such as TMS320C40 [41], Transputers [16]. With the computing power provided by modern DSP processors, coarse grain parallelism is employed, comparing to the traditional fine-grain design for image based processing. In our prototype system, the TMS320C40 digital signal processor is used.

The most important features of the TMS320C40 (C40) processor are the six communication links and the high-speed memory access. Other features of C40 are depicted in Figure 4.2. The communication port of the C40 provides a simple mechanism for C40-to-C40 connection. The maximum throughput of each port is 20Mbytes per second. In addition, the C40 can sustain a data transfer rate of 100 Mbytes/sec via its memory bus and this allows us to build a high-speed interface between the QuickRing controller and the DSP layer, details in Section 4.2.3.

40-ns and 50-ns instruction cycle times
operations per cycle throughput, resulting in massive computing parallelism and sustained CPU performance
40/32-bit single-cycle floating-point/integer multiplier for high performance in computationally intensive algorithms
Six communication ports with 32 bits data bus
Each port with 20Mbytes/s asynchronous transfer rate
Two identical external data and address buses for Global memory and Local memory
Memory access provides 100Mbytes/s of data-transfer rate

Figure 4.2 Summary of Features of TMS320C40

4.2.2 High Level Layer

Operations at the high level layer include decision-theoretic methods for recognition, structural methods for recognition and image interpretation [2][36]. Decision-theoretical methods deal with patterns quantitatively where structural methods seek to achieve pattern recognition by capitalising precisely on structural relationships. Image interpretation involves incorporating human knowledge into processing task. Most of these kinds of operations involve a large amount of database or knowledge base operation. With the enhancement of silicon technology and operation system, the processing power of a cluster of PC is capable to achieve high-performance computing in an effective and economical manner [20] [21] [22]. In our case, the high-level layer will initially consist of 15 PCs which are connected in a ring. The maximum number of node supported in a QuickRing network is only 16. One node in the PC network is occupied by the

interfacing component so that only 15 PCs will be provided. The number of high-level processing elements can be increased by adding another ring network to the Control and Communication Network, as shown in Figure 3.13. The network between the PCs will be supported by QuickRing. Among the 15 PCs, one of them will be co-ordinating data transfer between the PC layer and with the bridge hop, details will be given in Section 4.2.4.

4.2.3 DSP Layer Controller (DLC)

The DSP Layer Controller (DLC) is designed to provide an effective connection between the low-level layer and the Control and Communication Network (CCN). The DLC can initiate tasks within a layer according to control sent by the Master Controller. It can also gather results obtained from image based processing operations performed by other DSPs and send them to the Master Controller or High Level PCs. The major design criterion for the DLC is to maximising the data transfer rate between the QuickRing controller and the DSP layer.

The DLC, as shown in Figure 4.3, consists of two DSP processors and a QuickRing controller (QC). The QC is responsible to communicate with the MC and high level layer PCs via the QuickRing Network. The two DSP processors, the Receive Controller (RC) and the Transmit Controller (TC), are responsible to distribute data and gather result from other DSP and they are connected to the QC via their global memory port that supports a 100Mbytes/s transfer rate. The RC and TC are connected together by their communication ports so that control signals can be exchanged between them. In addition, the RC and TC are connected to other DSP processors, forming a ring, via their communication ports as depicted in Figure 4.4. Data received by the RC from the QC will be transmitted to other DSP within the layer. The RC will partition the image data if necessary. On the other hand, processed results will be collected by the TC and then passed to the QC. This arrangement increases the complexity of the control algorithm, but it increases the data transfer rate in both incoming and outgoing directions of the layer to 100Mbytes/s since the QuickRing controller can process data in it's Receive Port and Transmit port simultaneously. Therefore, the overall data transfer rate between the two layers can be achieved at 200Mbytes/s.

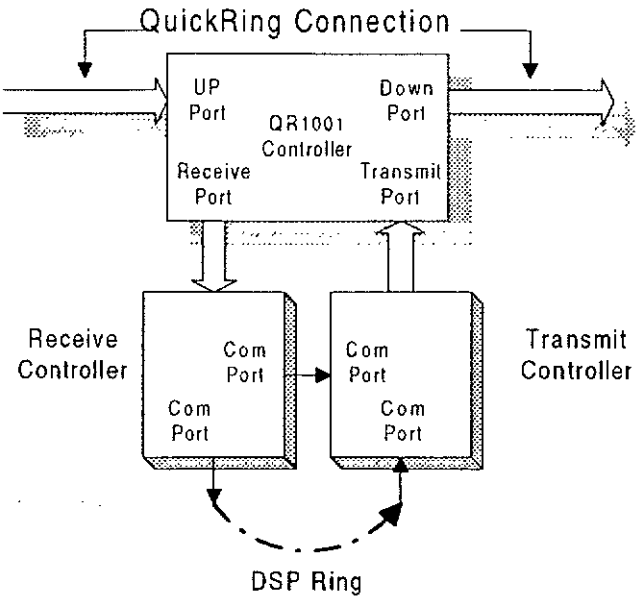


Figure 4.3 The DSP layer controller(DLC)

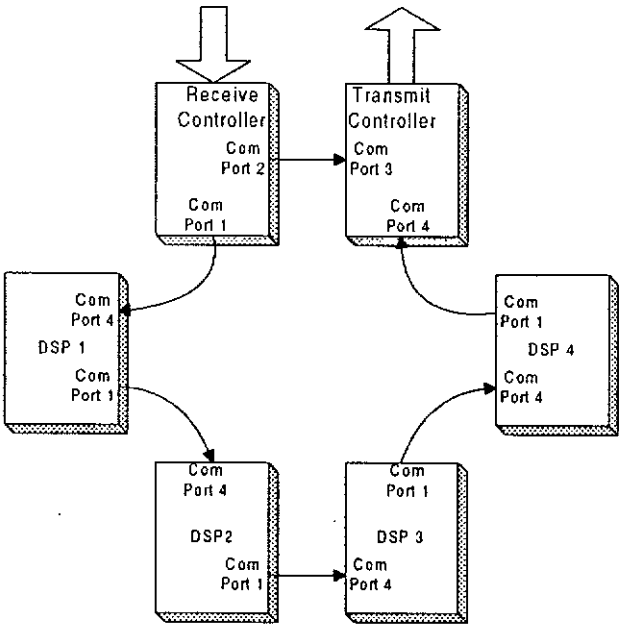


Figure 4.4 Formation of DSP Ring

In order to provide reliable communication under Multicast transmission mode, see Chapter 3, both the TC and RC are configured with large buffers to store image data. Image data coming from the MC will be first stored in the buffer of the RC. Similarly, data is first stored in buffers of the TC before sending either to the MC or the high-level layer. Since the buffers are important to the operations of the system, therefore, the status, such as full or not full, of buffers must be monitored in order to prevent buffer overflow. A buffer monitoring mechanism has been designed and this is part of the control mechanism which is covered in the next Chapter.

4.2.4 The Bridge Hop

The high level layer is connected to the CCN via a bridge hop instead of directly connecting to the CCN. This arrangement can localise data exchange between the PCs without affecting traffic in the CCN. In addition, this also allows us to expand the number of processing elements by adding other processing ring to the CCN.

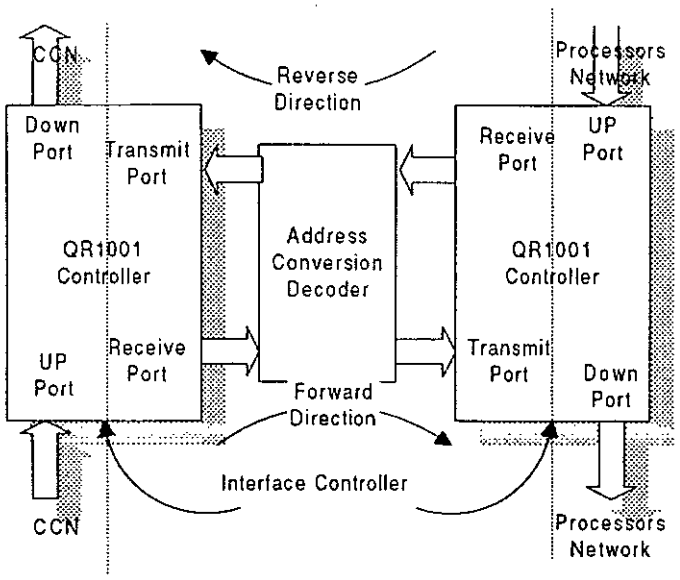


Figure 4.5 Bridge Hop of QuickRing

As shown in Figure 4.5, the bridge hop consists of two QCs connected back-to-back via an Address Conversion Decoder (ACD). Such an arrangement minimises the latency for data transfer between the two ring networks and maintains a high-throughput. The ACD provides a mechanism for forwarding data from the CCN ring to the high-level ring under the Multicast Mode. If Directed mode is employed instead then the ACD is not required.

When message arrives at the interface controller in the first ring, the controller will check the destination set in the header of the message. If the controller is in the destination, see Table 4.1, it will copy the message and send it to ACD. ACD will modify the destination set of the message and transfer the message to another interface controller. Then the message will be sent to the second ring.

4.2.5 Address Conversion Mechanism

As mentioned in Chapter 3, data transferred in a QuickRing network is in the form of a packet. The destination for a packet is identified by an address field. In Multicast mode, the addressing field is divided into a Group Field (8 bits) and a Multicast Field (16 bits). The Multicast Field represents a subset of nodes, which will receive the packet. In order to send data to the high-level layer then the node ID of the bridge hop must be included in the Multicast Field, as shown in Table 4.1. When the packet reaches the bridge hop, the QC of the bridge hop located in the CCN will copy the packet and forward it to the QC of the hop on the high-level layer. Since the original Multicast field only contains the destination set for the original ring, i.e., the CCN ring, the Multicast field has to be redefined for the high-level ring network. The content of the new Multicast Field can be encoded

and stored in the Group Field, which is not used in Multicast mode, when the message is being prepared in the Source Controller. An external conversion decoder is added, see Figure 4.5, to convert the original Group Field to the new Group Field and Multicast Field. The decoding mechanism is in the form of a lookup table. The ACD is divided into 2 parts for the two directions of data flow. The forward direction refers to the flow from the CCN to the high level layer while the reverse direction means the opposite.

Table 4-1 Example of Multicast field of a Message that have to route from CCN to high level layer

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

**Since node2 is the bridge hop connecting to the processor network, bit-2 must set to be 1 to route the message to the processor network.

The conversion table, as shown in Table 4.2, for the reversed direction is very simple because there are only three nodes in the CCN. As there are 8 bits in the group field, each bit in the group field can be used to represent node in the CCN. In the CCN, node 0 is the MC and node 1 is the DLC. They will receive messages from high level layer. Node 2 is the return path to the high level layer and will not receive data originated from the high level layer. As shown in Table 4.2, the upper 8 bits of the new Multicast Field are hard wired to 0 since they are not applicable. In the second row, it demonstrates how the group field is set if the packet is targeted for node 0. Similarly, in the third row, it shows the case for sending data to node 1. The pattern in the last row is for broadcasting data to both nodes.

Table 4-2 Address Conversion Table for the reverse Direction

E.g. No.	Original Group Field	New Multicast Field
1)	00000000	0000000000000000
2)	00000001	0000000000000001
3)	00000010	0000000000000010
4)	00000011	0000000000000011

The design of the conversion mechanism for the forward direction is much

more complicated comparing to the reverse mechanism. It is because the maximum number of high level processors in the ring is 15 and the number of destination set that has to be represented by the Multicast Field is equal to 2^{15}. However, the Group Field is only 8-bits wide and can only represent 2^8 destination sets. Therefore, a direct mapping from bits in the Group Field to the multicast field is not possible and there are three choices to deal with this problem.

In the first case, we could limit the number of processor in the high-level layer to 8 processors. Therefore, we could map the Group field to the multicast field directly. Certainly, this reduces the computing power. In order to maintain the number of processors at 15, we could connect two rings of high-level processors to the CCN, with 8 processors on each ring. The same address conversion mechanism must be provided in each bridge hop. Having two high level processing rings, the overall high level processing power will not be affected. However, the communication between processors in different rings becomes a problem. However, the two rings can be set to process different frames. For example, when ring 1 is processing data of frame 1, the ring 2 is available to process data of image frame 2. The overall performance of the system will not be affected significantly.

A part of the Address Conversion Table of this scheme is shown in Table 4.3. As there are 8 bits in the group field, each bit in the group field can be used to represent processor node in the high level layer network. In the high level layer network, node 0 is the return path to the CCN and will not receive data originated from CCN. Node 1 to Node 8 are the processor nodes. They will receive messages from CCN. As shown in Table 4.3, bit 0 of the new Multicast Field is hard wired to 0. Bit 1 to bit 8 is directly connected to Bit 0 to Bit 7 of the original Group Field. The upper 7 bits of the new Multicast Field are hard wired to 0 since they are not applicable. In the second row, it demonstrates how the group field is set if the packet is targeted for node 1. Similarly, in the third row, it shows the case for sending data to node 2. The pattern in the last row is for broadcasting data to all HP nodes.

Table 4-3 Part of Address Conversion Table for the Forward Direction with 8 node in processor ring

E.g. No.	Original Group Field	New Multicast Field
1)	00000000	0000000000000000
2)	00000001	0000000000000010
3)	00000010	0000000000000100
4)	00000011	0000000000000110
5)	00000100	0000000000001000
.	.	.
.	.	.
.	.	.
256)	11111111	0000000111111110

In the second option, we could route all messages to the first high-level processor node. This node is responsible to forward the message to the desired nodes. The bit 1 of the new Multicast Field must be hard wired to '1' and other bits are hard wired to '0'. This arrangement allows us to configure the conversion table dynamically, since the table can be sent to the first high-level processor by a packet. However, two problems will be raised by this arrangement. First, higher level of control mechanism must be provided for the routing node. Second, the data transfer rate for messages sending from ring of CCN to the network of high-level processors will be limited by the data transfer rate of PC-QuickRing interface.

Table 4-4 Routing Table in NODE 1

	Key	Destination Set
Entry 1	1	0010000000000000
Entry 2	2	1001001001001001
Entry 3	3	0010100000000000

As shown in Figure 4.6, MC will define the set of destinations and send a

message to the first high-level processor (NODE 1). This information will be stored in a routing table, see Table 4.4. The message contains the set of destinations and a key for indexing. Every message that sent to high level layer network has a key. Node 1 will compare the key with the routing table and multicast the message to the matched set of destinations.

In the last case, we could predefine 2^8 sets of destinations. Table 4.5 illustrates one possible combination for the conversion table. This arrangement requires that groups of destination high-level processors must be defined before the system is initialised. Since the address conversion table cannot be changed, it should be well planned to meet the grouping requirements for high-level Processing.

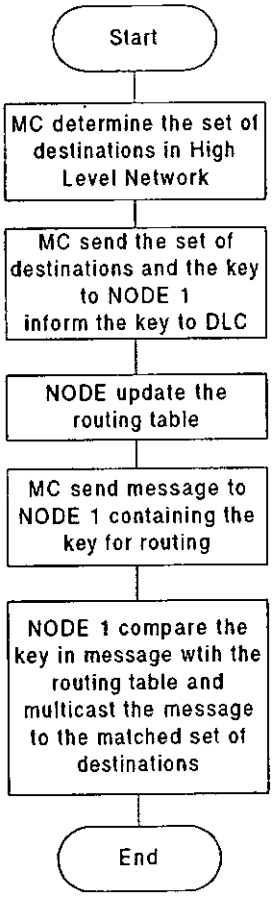


Figure 4.6 Flowchart of operation of Routing Node

Table 4-5 Part of Address Conversion Table for the forward Direction

E.g. No.	Original Group Field	New Multicast Field
1)	00000000	0000000000000000
2)	00010000	1010101010101010
3)	00100000	0101010101010100
4)	00110000	1001001001001000
5)	01000000	0110110110110110
6)	01010000	0100100100100100
7)	01100000	0010010010010010
8)	01110000	1101101101101100
9)	10000000	1011011011011010
10)	10010000	1000010000100000
11)	10100000	0100001000010000
12)	10110000	0010000100001000
13)	11000000	0001000010000100
14)	11010000	0000100001000010
15)	11100000	1110011100111000
16)	11110000	1111011110111100
17)	00010001	0000000000000010
18)	11110001	1111111111111110
19)	11100010	0111111111111100
20)	00011111	1000000000000010
21)	00110111	1111111110001110

Referring to Table 4-5, the Original Group Field is divided into two 4-bits fields. In our system architecture, Node 0 of the high-level layer is the Bridge Hop and will not receive message. Therefore, bit 0 in the Multicast Field is hard wired to '0'. When all the lower 4 bits of the original Group Field are zeros, the upper 4 bits define the distribution pattern of the destination set. Since only 16 patterns can be represented by 4 bits, we have chosen the patterns that can evenly distribute data to processors. When the lower 4 bits of the original Group Field contain 1s, row 17 to 21 in Table 4-5, it indicates the binary value of the starting node position. The upper 4 bits indicate the ending node position; the Multicast message is then broadcast to the range of nodes as described. For example, in Table 4-5, row 18, the starting position indicated 1 and the ending position indicated 15, so that the bit 1 to bit 15 in the New Multicast Field will be set. For row 21 in Table 4-5, the starting position indicates node 7 and the ending position indicates node 3, so that the bit 7 to bit 15 and bit 1 to bit 3 in the New Multicast Field are set to 1.

The Address Conversion Decoder can be implemented by a high performance Programmable Array Logic (PAL). The ACD will not reduce the throughput of the network but it will cause a short transmission delay.

4.3 Conclusion

In this Chapter, the architecture of the two-layer system and the communication components of the system have been introduced. Currently, our system consists of three ring networks which include the low-level processing ring, high-level processing ring and the control and communication ring. In the current design, only two processing layers are connected to the control and communication ring (CCN), however, as supported by the QuickRing controller, more processing layers can be included as long as the number of nodes in the CCN is not more than 16.

In order to connect the different processing layers together, dedicated interfacing components have been designed and their basic functions have been described. In the next Chapter, the control and communication mechanism for the system will be presented.

Chapter 5

5. THE SYSTEM CONTROL MECHANISM

5.1 Overview

In previous chapters, the network technology, the system architecture and the design of the interfacing components have been described. Our system consists of two processing layers and a communication and control ring network. The Multicast mode data transfer mechanism is utilised in our system. In this Chapter, details regarding the control mechanism will be presented.

The control mechanism governs the flow of information, including both control signals and image data, between the layers and the Master controller. Based on the multicast mode mechanism of QuickRing, we have derived a control protocol. In our design, we have minimised the communication overhead and therefore, improved the utilisation of the network bandwidth.

5.2 The Control Mechanism

The control mechanism mainly deals with the flow of information between the processing layers and the Master Controller (MC). Information includes both control signals and data. The MC is responsible to assign processing tasks to the two processing layers. Each processing task will embody different types of processing operations. Once the processing operations have been determined, the MC will send the task number, the task identity (ID), to both low-level and high-level processing layers. The task ID is stored in a table located in both the DLC and all HPs (High-level Processors) so that status of a task can be traced. After the distribution of the task ID, image data and processing operation ID, the image processing operation will be initiated. By receiving the image data, the low-level layer will carry out the proper operations and then forward the results to the high-level layer. When the high-level operations complete, result will be returned to the MC. The task ID is used to identify the data and it is included in all data packets transmitted between different components.

In order to achieve the control mechanism, a number of software modules must be provided in the MC, DLC and the HPs, as shown in Figure 5.1. In the MC, there are the Resource Manager (RM), and the Task Manager (TM). A Resource Agent (RA) and a Task Agent (TA) are found in each HP and the DLC.

The modules located on HPs and DLC, the RA and the TA, are responsible

for monitoring tasks that are processed by HPs and the DLC, and to report to the MC. On the other hand, modules located on MC have several responsibilities including getting instructions and image data from input device, partitioning the jobs into low-level and high-level processing operations, and maintaining the status of tasks distributed to HPs and DLC.

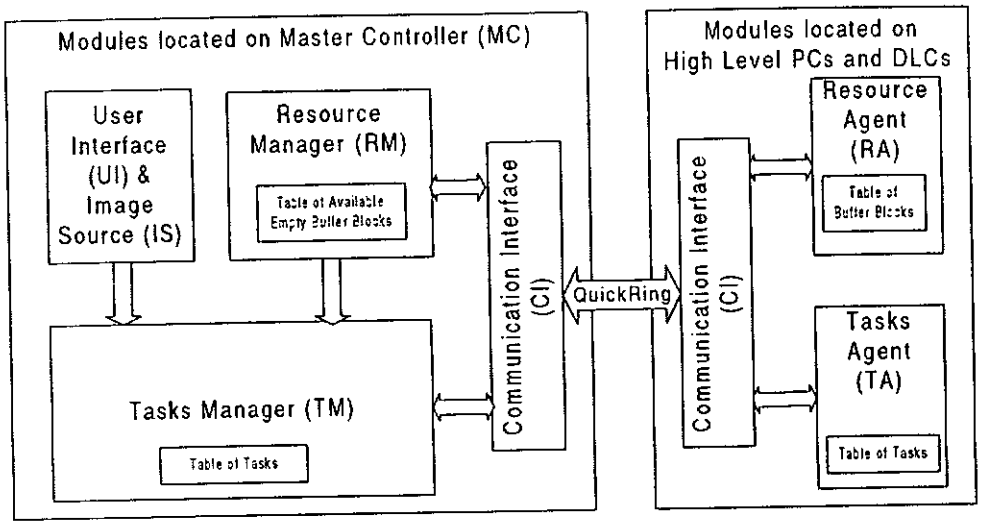


Figure 5.1 Modules of Our Communication and Control Protocol

5.2.1 *Resource Manager (RM) and Resource Agent (RA)*

The term resource refers to the storage buffer provided in the DLC and HPs. As discussed in Chapter 3 and 4, although the Multicast mode provides a more efficient mechanism for data transfer, it also imposes a problem in data security. New incoming message cannot be stored in QuickRing's internal buffer if the internal buffer is full. To solve this problem, each device must have sufficient memory storage and data stored in QuickRing's buffer will be moved to the memory continuously. If the memory storage in a device is full, then the sending of data must be suspended to prevent data loss. The function of the Resource Manager (RM) and the Resource Agent (RA) is tailored for solving this problem. The memory storage is divided into blocks with the same number of storage space and each of such blocks is regarded as a buffer. The Resource Agent (RA) is responsible for buffer monitoring and reporting. The Resource Manager (RM) is responsible for buffer managing.

After system initialisation, the Resource Agent (RA) will count the number of empty buffers that is ready for incoming jobs. The RM will be notified by the RA about the quantity of available buffers and this information is stored in the Table of AEBB (Available Empty Buffer Blocks) inside the RM, as shown in Table 5-1. If buffer is not available in a particular processing element then it will be regarded

as busy and no processing task will be assigned. The operation sequence of this operation is depicted in Figure 5.2. RAs will check the number of empty buffer blocks again when the node receives an instruction. The RA will provide the RM with the latest information of available buffer so that the RM can update the Table of Available Empty Buffer Blocks accordingly.

5.2.2 Task Manager and Task Agent

Image Processing tasks are divided into high-level and low-level sub-tasks and distributed to the two processing layers. The assignment of processing tasks is carried out by the TM resided in the MC. The Task Manager will first check the Table of AEBB in the RM and then divides tasks according to the status of available buffer. The Task Manager distributes the tasks to the processors, both the DLC and the HPs, via the Communication Interface (CI), as shown in Figure 5.1.

Table 5-1 Table of AEBB (Available Empty Buffer Blocks)

	Entry 1	Entry 2	Entry 3	Entry 4		Entry n-1	Entry N
Processor Name	DLC	HP1	HP2	...		...	HP_N
No. of empty Buffer Block	2	3	0	...		...	1

**Number of entries in this table is equal to the number of DLC plus the number of high-level Processors in the system

The processing status of a task has to be monitored so that inactive tasks can be terminated or re-activated by the TM. The Task Agent in each processing device is designed to monitor the status of sub-tasks and it will report to the Task Manager when a task is completed in that node. On the other hand, if the TM does not receive message from a TA then it will probe the processing device by sending a control message. If a TA cannot response to the TM within a defined period, TM will send a control signal to clear the task and in the mean time, trigger an error handling procedure. The operations sequence of the TM and TA is shown in Figure 5.3.

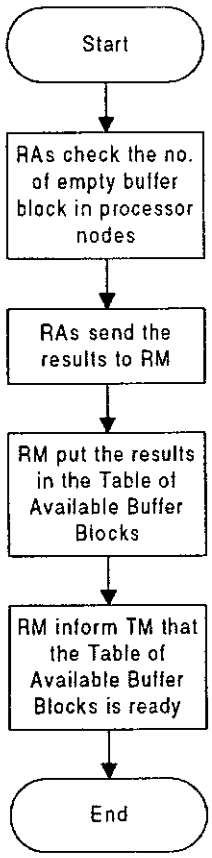


Figure 5.2 Procedures of Buffer Control operation

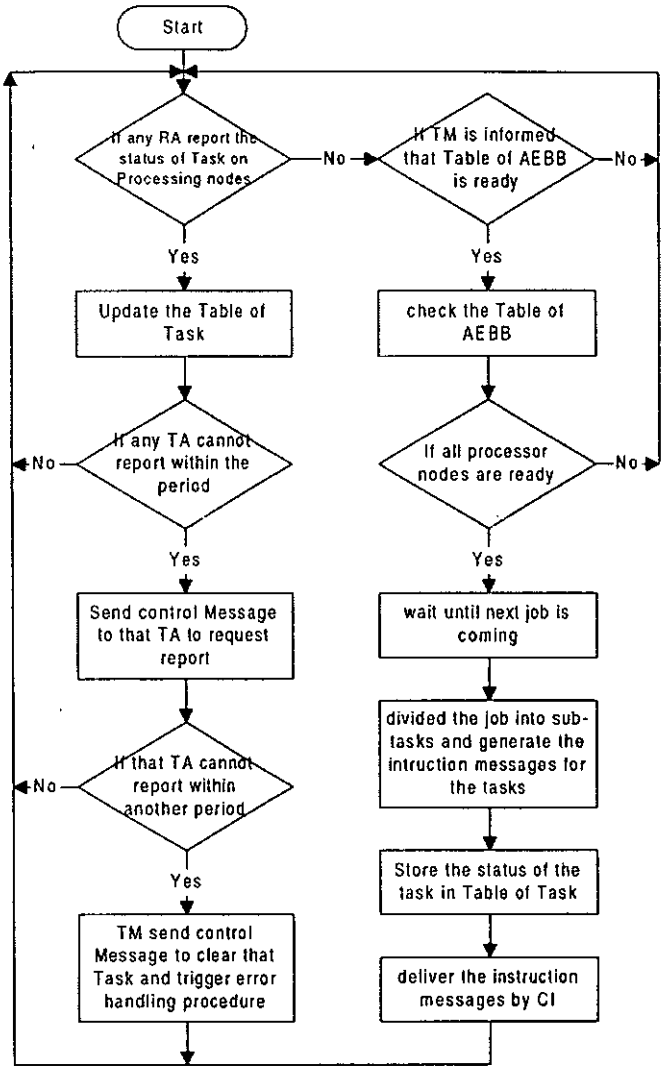


Figure 5.3 Procedure of Operation of TM and TA

5.3 Message Format

In this section, we will describe the message format used in the control mechanism. Since the mechanism is based on the multicast transmission mode of QuickRing, the message format must be compatible with the original data format. The original QuickRing packet consists of 80 bytes of data and 4 bytes of header information. The header information cannot be altered. The 80 bytes of data will be used to convey both data and control information required for our control mechanism. In our design, 4 bytes are used to carry control information and the other 76 bytes are for data.

Different types of messages are defined to handle different communication and controlling operations and functions, as shown in Table 5-2. Five types have been derived and the first 3 types of messages are responsible for the delivery of instruction and data, while type-4 and type-5 messages are for control and communication purposes.

The type-1 message consists of 5 fields that include the message type, Task ID (TID), Operation ID and Destination. The structure of a type-2 message is very similar to the type-1 message with the exception that a data field is added in the message. A Type-3 message consists of a message type field, the TID and a data field. Since the size of image data is too large to be contained in a single message, a type-2 message is always followed by a number of type-3 messages in order to convey a complete image data. Examples of how communication proceeds between different components with different message types are given in Section 0.

A Type-4 message consists of a Message type field, the Task ID (TID) and Control Key field. Finally, the Type-5 message includes a Task ID (TID), a Task Status field, the PID and NAB field. The Type-4 message is used to transmit buffer information from a processing device to the MC. If the TID value is equal to zero then no specific instruction is being referred. Messages of type 1, type 2, and type 4 are transmitted from MC to DLC and HPs. Type-3 message is transmitted from any data sources to destinations. Type-5 message is transmitted from DLC and HPs to MC.

Table 5-2 Component Fields of different Message types

Mes-sage Type	Function	Component Fields						
1	Deliver Instruction	**Field Name**	Message type	Task ID (TID)	Operation ID	Destination (of the result)	Not Used	
		Location [bit no.] symbol	[31:29] 1st symbol	[28:24] 1st symbol	[23:0] 1st symbol	[31:4] 2nd symbol	[3:0] in 2nd symbol and all bits in the following symbols	
2	Deliver Instruction	**Field Name**	Message type	Task ID (TID)	Operation ID	Destination (of the result)	Not Used	Data
		Location [bit no.] symbol	[31:29] 1st symbol	[28:24] 1st symbol	[23:0] 1st symbol	[31:4] 2nd symbol	[3:0] 2nd symbol	all bits in the following symbols
3	Deliver Data	**Field Name**	Message type	Task ID (TID)	Not Used	Data		
		Location [bit no.] symbol	[31:29] 1st symbol	[28:24] 1st symbol	[23:0] 1st symbol	all bits in the following symbols		
4	Instruction Control	**Field Name**	Message type	Task ID (TID)	Control Key	Not Used		
		Location [bit no.] symbol	[31:29] 1st symbol	[28:24] 1st symbol	[23:21] 1st symbol	[20:0] in 1st symbol and all bits in the following symbols		
5	Reporting Status	**Field Name**	Message type	Task ID (TID)	Task Status	PID	NAB	Not Used
		Location [bit no.] symbol	[31:29] 1st symbol	[28:24] 1st symbol	[23:21] 1st symbol	[20:4] 1st symbol	[3:0] 1st symbol	all bits in the following symbols

Description of the field name:

1. Message type: a 3-bits number to show the type of the message.
2. Task ID (TID): a 5-bits ID is assigned for image processing job. All processing sub-tasks for an image job are assigned with same TID.
3. Operation ID: a 24-bits key to tell processor node to perform an assigned operation.
4. Destination: a 24-bit pattern to represent the set of destinations. It forms the address field in the header symbol of the result messages.
5. Not Used: is the empty-space that do not contain any value.
6. Data: is the space to contain data.
7. Control key: represent the operation TM assigns to TA to carry out the task with the matched TID.
8. Task Status: contain the status of the task that being processed in processor node.
9. Processor ID (PID): contain the ID of processor node. For example, 1 represents the DLC, and 2 represents the first HP.
10. Number of Available Buffers (NAB): contain the number of buffers on the processor node that is available for new processing tasks.

During system initialisation, the Resource Agent (RA) located on HPs and the DLC check for buffer blocks that are empty and send the results to the Resource Manager (RM), refer to Figure 5.2. RM located on MC will update the Table of AEBB, refer to Table 5-1, to identify which HPs and DLC that have empty buffer blocks for receiving instruction. According to the Table of AEBB, the Task Manager can divide image processing jobs into sub-tasks. Then sub-tasks will be distributed to the available HPs and DLC. After receiving the instruction, RA will check the number of empty buffer blocks in HPs and DLC again, and then they will send the results to RM. With the results, the RM can update the Table of AEBB and will prepare to distribute other tasks to the HPs or DLC that with empty buffers. The steps performed by a processor node to handle an incoming message is depicted in Figure 5.4.

After the HPs or DLC have processed the data and transmitted the result to the destination that is defined by the Operation ID and Destination, the RA will check the buffer again and send the results to the RM. With this arrangement, the RM can always update the Table of AEBB.

The TA will send a type-5 message to the TM to report the processing status of different tasks. The TM will then update its Table of Task based on the

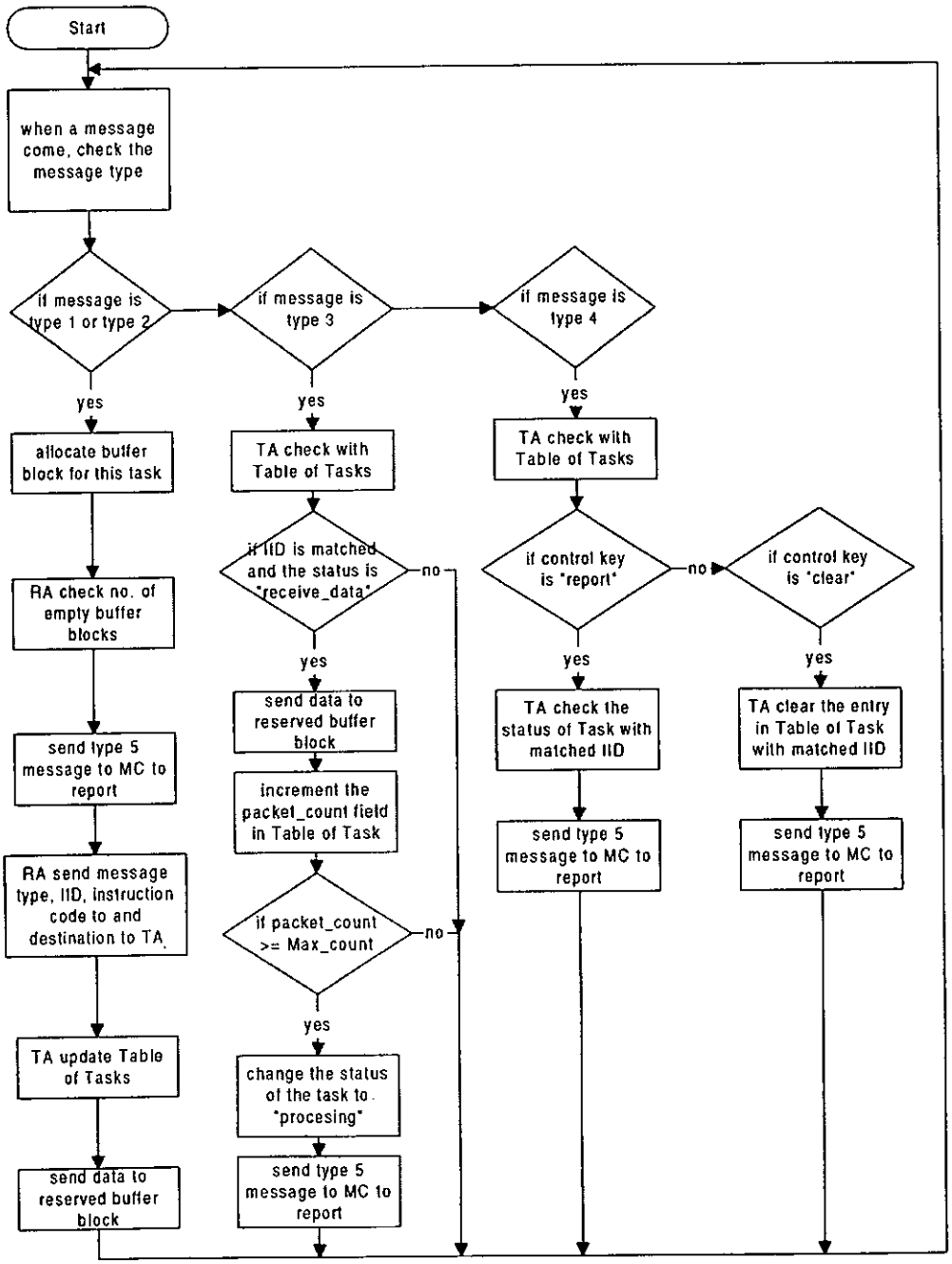


Figure 5.4 Procedures of Processor Node to handle incoming Message

message. If the TM suspected that a specific task is not functioning properly in either DLC or HPs, TM will send a type-4 message to that suspected DLC or HPs.

The sequence of message transmission is illustrated in Figure 5.5.

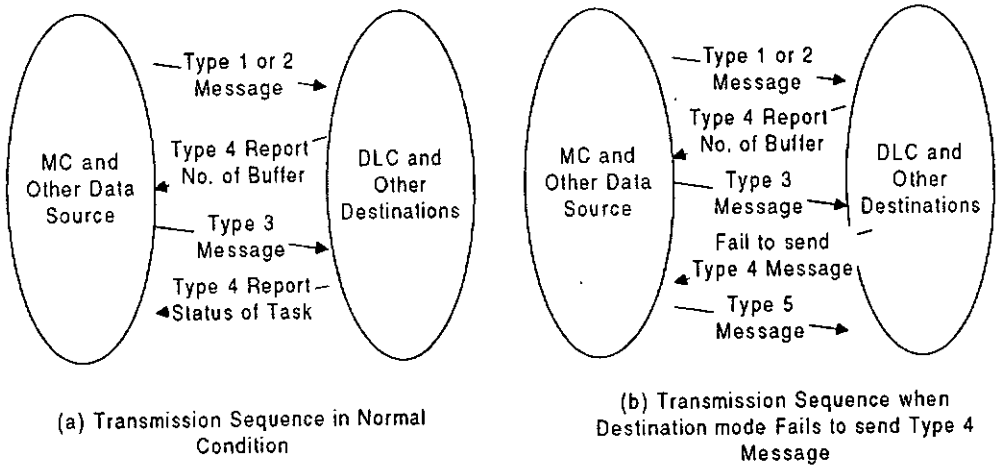


Figure 5.5 Message Transmission Sequence

5.4 Control Algorithm in DSP Layer Controller

As described in Chapter 4, the DLC consists of two DSPs, the Transmit controller (TC) and the Receive controller (RC), and a QuickRing Controller (QC). The RC and TC are connected together through a communication port and they are also connected to other DSPs via their communication ports. The control algorithm designed for DLC also make use of the RA and the TA. The RA is located in the RC while the TA is located in the TC.

Since each TMS320C40 processor is allocated with 1Mbyte of local buffer, which is large enough to store data for 4 image frames. Buffer is divided into 4 memory blocks with 256kbytes each. Each block is assigned with a unique identity number. During system initialisation, the QuickRing controller, via its receiving port, will inform the RC its node number and the total number of nodes connected in the Control and Communication network. Such information is necessary for the generation of packet header when data is being sent to other components in the system. The RC will forward the information to the TC which is responsible for sending data.

In the next step, the RA will check for buffer blocks that are available for receiving data. It will inform the RM through the Communication Interface located on the TC. The RA will maintain a Table of Buffer Blocks, as shown in Table 5-3. The function of this table is to keep track of the status of each block located in the RC. On the other hand, the TA will also maintain a Table of Buffer

Blocks for the TC. The table contains information regarding status of buffers allocated for the processed results which are waiting for transmission in the buffer.

Table 5-3 Table of Buffer Block

Field Name	Block Number	Starting Address	Status (1byte)
Entry 1	1	00001100 (hex)	01
Entry 2	2	00041100 (hex)	02
Entry 3	3	00081100 (hex)	00
Entry 4	4	000C1100(hex)	00

**The number of entries in this table equal to the number of buffer blocks in the processor node.

Descriptions of Field Name:

1. Block Number: is an integer assigned to the buffer block
2. Starting Address: represent the starting address of the corresponding buffer block.
3. Status: is an integer representing the status of the corresponding buffer block. 0 means the buffer is not used. Other values represent the Task ID of that job that this buffer is assigned to.

As depicted in Figure 5.4, when a message arrives, the message type will be examined. If the incoming message is either a type-1 or a type-2, the information including the message type, the TID, Operation ID and destination address will be sent to the TA. The RA will reserve enough buffer blocks for that instruction and inform the TA the starting address of the reserved buffer blocks. The TA will then add an entry in the Table of Tasks, see Table 5-4, and fill the different fields with correct information. The TID and Max_count are extracted from the message while the address information for the buffer block is provided by the RA. Moreover, the status entry will be set to "*getting_data*", see Table 5-5. The value of Max_count, see Table 5-4, is equal to the total number of packets required for that task. Since the DLC is responsible to distribute processing task to other DSPs for parallel processing, the RC will decide the parallel method according to the Operation ID.

Table 5-4 Table of Tasks

Field Name	ID	Status	Starting_address	Current_address	packet_count	Max_count
Entry 1	01	*"send_result"*	00001100 (hex)	00001240 (hex)	4	3276
Entry 2	02	*"processing"*	00041100 (hex)	000810C0 (hex)	3276	3276
Entry 3	03	*"getting_data"*	00081100 (hex)	000BBA80 (hex)	3000	3276

**The number of entries in this table equal to the number of tasks being processed in this node

Description of field name:

1. Task ID and identity key assigned to each Task (refer to Table 5-2)
2. Status: status of Task (refer to Table 5-5)
3. Starting_address: Starting address of the buffer block. (refer to Table 5-3)
4. Current_address: The address of the data that being transferred.
5. packet_count: To count the number of packets that have been transferred.
6. Max_count: The Max number of packet that required to transferred in this task.

If it is a type-2 message then data stored in the data field will be moved to the reserved buffer in the RC. Under a normal situation, type-3 messages will follow either a type-1 or type-2 message. The type-3 messages, see Table 5-2, will carry the image data to the DLC. When a type-3 message arrives, the TA will extract the TID from the message and search for the TID in the Table of Tasks. If the Table of Tasks has an entry of the matching TID of the incoming message and the status field is *"getting_data"*, see Table 5.5, then the data field of that message will be stored in the reserved buffer block. At the same time, the packet_count field of the Table of Tasks will be incremented by one. When the value of packet_count equals that of Max_count, the TA will set the processing status to *"processing"* and then send the data to fellow DSPs through communication ports. The TA will send a type-5 message to MC to inform TM the status of the tasks when all packets related to the task have been received.

Table 5-5 Processing Status of Task in HPs and DLC

Processing Status	Description
getting_data	ready to get data and store it in buffer
Processing	data in buffer is being processed
send_result	Processed results are in buffer and are sending to destination.

If the TM does not get a Type-5 message from the TA located on the DLC, the TM will send a type-4 message to DLC. When the message arrives, the RC will extract the TID and forward it to the TA. The TA will then check the TID against the Table of Tasks and carry out the operation according to the control code embodied in the type-4 message. For example, if the control code is "*delete*", the TA will clear the entry with the matched TID in Table of Tasks and inform the RA to release the reserved buffer block. The RA will then inform the TA the current number of empty buffer blocks. Finally, TA will send out a type-5 message to tell the TM that the task has been deleted.

In previous paragraphs, we have described how incoming messages are being handled by the DLC and now operations for sending data from the DLC will be discussed. When other DSPs have completed their processing operations, results will be pumped to the TC. The TA will gather the results into the reserved buffer and based on the TID accompanying the result, the status field in the Table of Tasks will be set to "*send_result*". Eventually, the result will be sent through the QuickRing controller to other processing devices, most likely the high-level processing layer.

The Starting_address field of the Table of Tasks will be replaced by the buffer block address where the processing results are being stored. After the transmission of a packet, the current-address field of Table of Tasks will point to the address of the next packet to be transmitted. When the transmission is completed, the TA will delete the record from the Table of Tasks. Finally, the TA will also send a type-5 message to the MC informing the termination of the task.

5.5 Control Algorithm in HPs

Communication mechanism in HPs is similar to that in the DLC. After the HPs receive a message, they will first check the message type. If the incoming message is type-1 or type-2, then information including the message type, the TID,

the Operation ID and the destination address will be sent to the TA. The RA will reserve enough buffer block for that task and send the information of the reserved buffer blocks to the TA. The TA will then add an entry to the Table of Tasks using the TID as a key and set the processing status for the new entry to "*getting_data*". For a type-2 message, its data field will be sent to the allocated buffer and the Max_count field will also be stored in the Table of Tasks.

When a type-3 message arrives, the TA will check the message's TID with the Table of Tasks. This is similar to the process performed by the DLC. When all packets related to that task are stored in the buffer block, the processor will process the data. The TA will set the processing status, based on the TID, in the Table of Tasks to "*processing*" and send a type-5 message informing the MC the status of the task and the RM will update the information in the Table of AEBB.

When the TM cannot get a type-5 message from a HP, it will send a type-4 message to that node. When the message arrives, the TA will check the TID and control code conveyed in the message. For example, if the control code is "*clear*", the TA will terminate the process and clear the record in the Table of Tasks with the matching TID. In such situation, the RA will release the reserved buffer block and it will then inform the TA the available number of empty buffer blocks. Finally, the TA will send a type-5 message to notify the TM that the task has been deleted.

If the processing operation completes normally then the HP will store the results to the reserved buffer block. The TA will set the processing status to "*send_result*" in the Table of Tasks and start transmitting the result to the destination using the QuickRing controller. Once the transmission is done, the TA will delete the corresponding record from the Table of Tasks.

5.6 Conclusion

In this Chapter, we have introduced the communication and control mechanism of our communication network. The mechanism provides several features including prevention of buffer overflow and task monitoring.

In order to provide a reliable data transmission, both the Resource Manager and the Resource Agent are monitoring the number of empty buffer blocks on all processing nodes and the DLC. They can prevent buffer overflow and ensure the integrity of data. With the information provided by the Resource Manager, the Task Manager can divide the incoming jobs according to the amount of available resources in the system. A single Task Manager is responsible for dividing incoming Job and distributes the sub-tasks to processor nodes with different Task ID. Furthermore, Task Manager and Task Agent are monitoring the status of task distributed to processor nodes. This arrangement ensures the safety of the sub-task

and Task Manager can perform error handling.

Based on our designed communication mechanism, the communication overhead is low implying that the mechanism is efficient. When a task is being handled, 2 control messages are transmitted over the network. In addition, all data messages come with 4 bytes of header. Therefore, when we transmit an image frame which includes 3450 packets, the percentage of the communication overhead induced is:

$$\frac{2}{3452}+\left[\left(\frac{4}{80}\right)\times\left(\frac{3450}{3452}\right)\right]=5.055\% \qquad (1)$$

This communication and control mechanism is efficient and reliable. It can satisfy the communication requirement for our Heterogeneous System. However, the performance of the components and the communication mechanism has not been presented. We have simulated functions of both the hardware component and the communication mechanism. The results are presented in the next Chapter.

Chapter 6

6. PERFORMANCE EVALUATION

6.1 Introduction

In previous chapters, the system architecture and the system control mechanism of the proposed two-layer architecture have been described. In order to verify the functionality and efficiency of the communication mechanism, we have simulated the operation of the system and studied its performance. We have simulated the flow of information between the layers. The performance of the system depends on the DSP Layer Controller (DLC) because it is the bridge connecting the low-level processors network and the high speed QuickRing network. Therefore, we have also simulated the function of the DLC to evaluate its performance. Details of the studies and simulation will be presented in following sections.

6.2 Simulation of the System's Operation

The simulation of the system's operation will reflect the system's performance and is important for verifying our design. In order to carry out the simulation, we have assumed that a frame of 512x512 pixel image is being processed. The processing operations include both image based (low-level) and symbolic processing (high-level). The image is stored in the Master Controller (MC) and first transferred to the DSP layer for low-level processing. When low-level operations complete then the resultant image is passed to the high-level layer for further processing. After high-level processing, the results are returned to the MC. To analyse the performance of the system, the operation steps involved in the above process are identified, as shown in Figure 6.1. In addition, the duration of each processing step is labelled and shown in Table 6.1.

The processing steps depicted in Figure 6.1, include the initialisation sequence of the MC, the DLC and all high-level processors (HPs), as described in Chapter 5. Initially, all HPs and DLC will first check for the number of empty buffer blocks and send the results to MC. The MC will divide image-processing job into sub-tasks and generate instruction messages of that job, which are then transmitted to the DLC and HPs. The duration for the MC to send the instruction and data to DLC is equal to t4 while t5 represents the time to broadcast the instruction to the HPs. In this simulation, the messages, carrying the Operation ID, being sent to the DLC is a type-2 message and it is followed by many type-3 messages which contain data of the image frame (256k bytes). The message broadcast to the HPs is a type-1 instruction. After receiving the instruction and

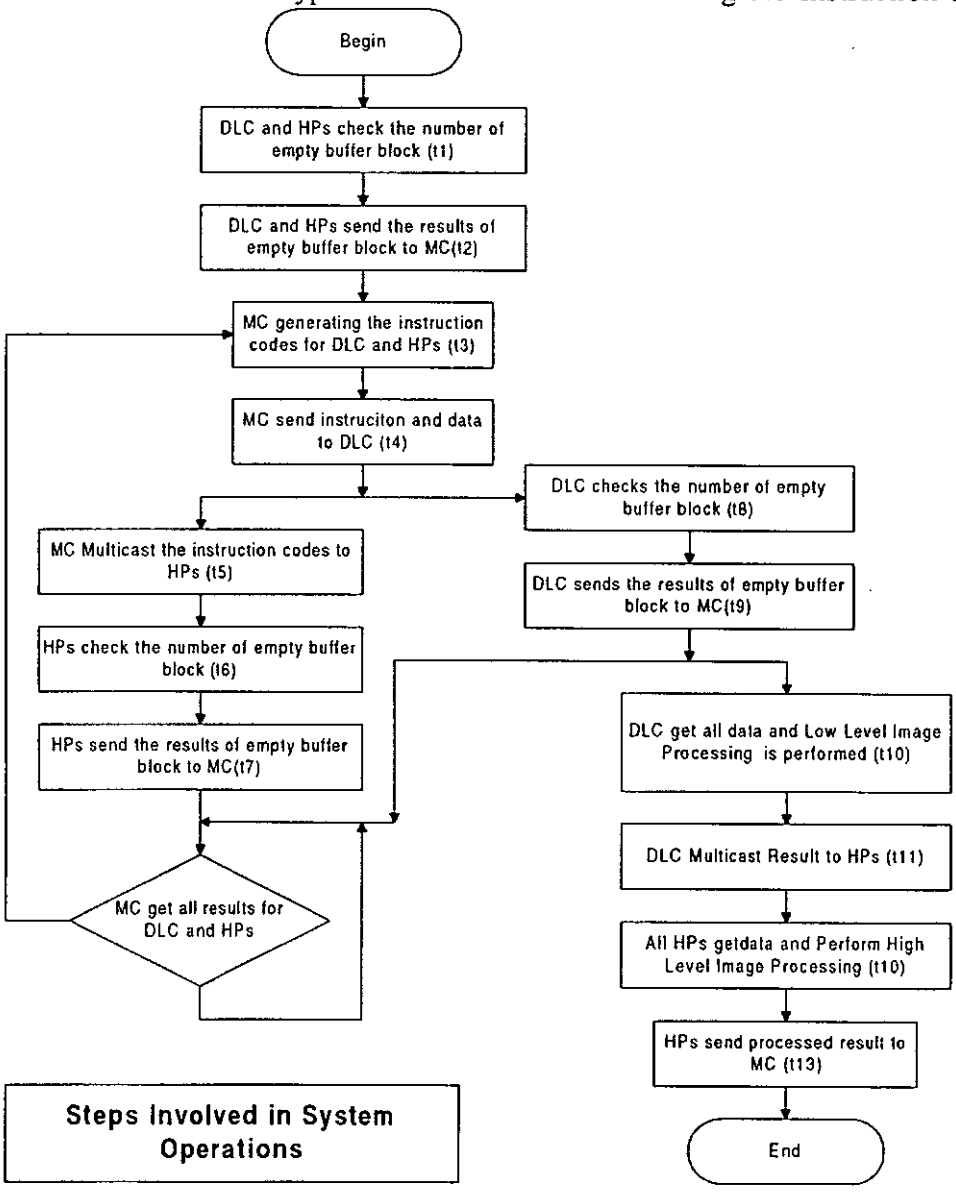


Figure 6.1 Flow-Chart of System Operations

data, the DLC and the HPs will check the number of empty buffer blocks again and send the results to the MC. Once the DLC has sent the results to the MC, low-level processing is performed. We have assumed that the processing time for the operation is t10. At the same instant, a user may submit another job for the MC. In such a case, the MC will generate another set of instruction messages for the DLC and the HPs. When low-level image processing completes, the DLC broadcasts the processed results to all specified HPs. Then HPs will perform high level image-processing. Finally, all HPs send the processed results back to the MC.

The duration for the processing steps presented in Figure 6.1 can be determined based on the number of operation cycles and information [23] of the QuickRing controller. The results are listed in Table 6.1. For the simulation, we have made three assumptions:

Table 6-1 Time required by Communication in QuickRing

Symbol	t2	t4	t5	t7	t9	t11	t13	t4'	t5'
Value	8.8 µs	1.82 ms	3.35 µs	1.20 µs	8.6 µs	1.84 ms	86.42 µs	1.818 ms	3.35 µs

** All primed symbols (e.g. t4', t5') represent periods of the second cycle of the operation steps, symbols without prime, refer to Figure 6.1, represent the periods of the first cycle of the operation steps.

1. The quantity of data transferred to the low-level image processing layer is equal to 256 Kbytes and the amount of data transmitted by HPs to the MC is equal to 800 bytes. In Figure 6.6 , we have shown the communication time when the data transmitted is not equal to 800 bytes.

2. Transmission of high level processing results will begin when all HPs have finished high level processing.

3. The processing time for the same process operation in different cycles is equal.

Since the quantity of resultant data obtained from most high level image-processing are much less than data conveyed by an image frame, the assumption on the size of results transmitted by HPs is practical. The second assumption is the worst case situation for the transmission of high level result. In practice, each HP can send its results to the MC when it has finished high level image processing, i.e. asynchronously. Therefore, the duration for t13 should be shorter in practice than in our simulation result. The third assumption can highly reduce the complexity of the simulation and is not far from real-life cases.

For a ring network, a problem that may occur during data transfer between nodes is congestion. Congestion refers to the situation when a node is transmitting its own messages and at the mean time being requested to forward message to

other nodes. This will delay the delivery of messages. In our design, there are three situations where congestion may occur.

In the first case, it is when the MC broadcast message to HPs (t5) since the message must pass through the QuickRing Controller of DLC. If, at the same instant, the DLC has finished checking the number of empty buffer block (t8) and is ready to send the result to the MC (t9), network congestion may occur. However, the effect of congestion can be ignored because the duration of t5 and t9 are very short.

In the next case, it occurs when the MC has generated the next instruction (t3') and is sending the instruction and data to the DLC (t4'). The DLC has processed the data (t10) and sending the result to HPs (t11). Since the DLC can receive and transmit data simultaneously, therefore, the overlapping of (t4') and (t11) will not cause congestion. Since the data transmission operations in t5' and t11 may use common network path, the overlapping of t5' and t11 may cause congestion. However, the duration of t5' is insignificant when comparing to t11, the effect of congestion also can be ignored.

The final case, congestion may take place in the HP network when HPs are transmitting high level image-processing results to MC (t13) and the DLC multicasting the processed results of the next image-frame to HPs (t11'). In the worst case, congestion may cause delay of operation t11' by the period spent by t13. Since the period of t13 is shorter than 0.1ms, the effect of congestion can be ignored.

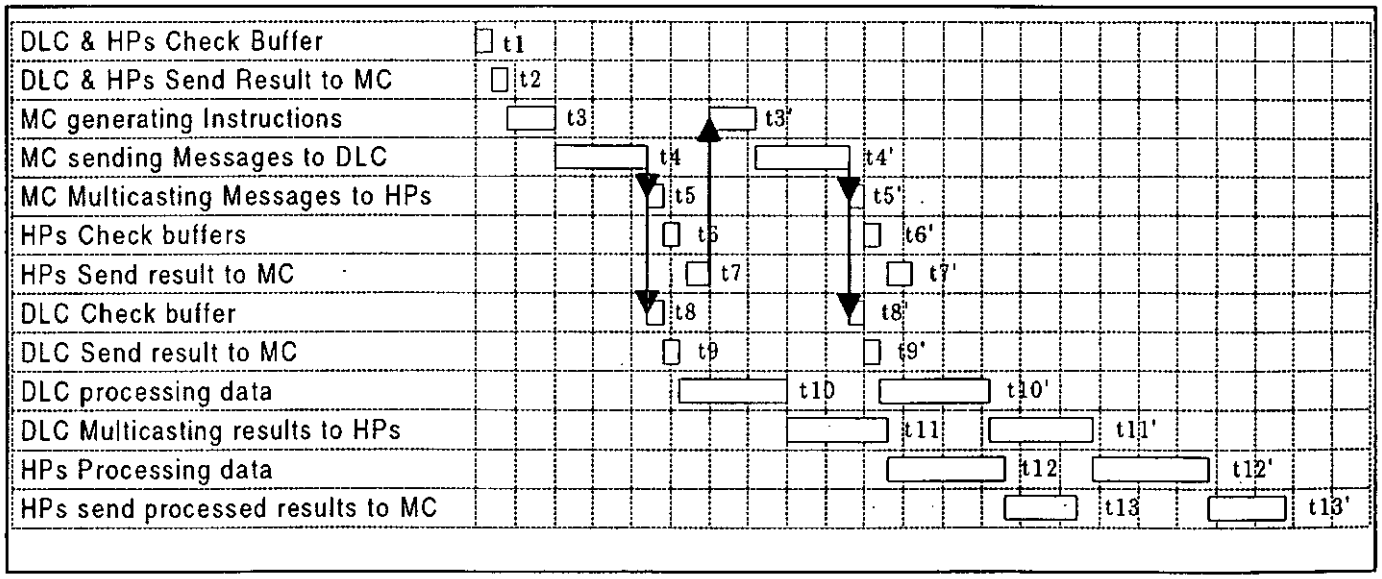


Figure 6.2 Timing Diagram of the communication

Figure 6.2 illustrates the timing relationship in all processing steps and communication steps without considering the network traffic problem. Details of performance analysis will be presented in Section 6.3.

6.2.1 Performance of the DLC

In the last section, ideal values for the duration of the processing operations have been presented, see Table 6.1. The total communication time of an image-processing job is estimated to be 4ms. However, in the previous simulation, we have assumed that the data transfer rate between the QuickRing controller and the DLC is 160 Mbytes/s. In fact, the maximum data transfer rate of the DLC is 100Mbytes/s in one direction and this is limited by the memory access speed of the C40 as described in Chapter 4. In addition, the data transfer rate will also be reduced by overheads of the communication protocol. Therefore, the total communication time will be affected, especially in the step when MC sends messages to the DLC (t4) and when the DLC broadcasts results to the HPs (t11). Therefore, it is necessary to investigate the performance of the DLC by software simulation. Based on the results, we can then adjust the values of t4 and t11 so that more accurate communication overhead can be determined.

As described in Section 4.3, the DLC consists of two C40 DSPs and a QuickRing Controller. The global memory ports of DSPs are connected to the Transmit port and the Receive port of the QuickRing Controller. The maximum data transfer rate of QuickRing controller is 160 Mbytes/s. Data transmitted by QuickRing is packed into 80-byte packet. A program is written, using the C40 development kit [42] [43], to simulate the process of the DLC. Figure 6.3 shows the flowchart of the simulation program.

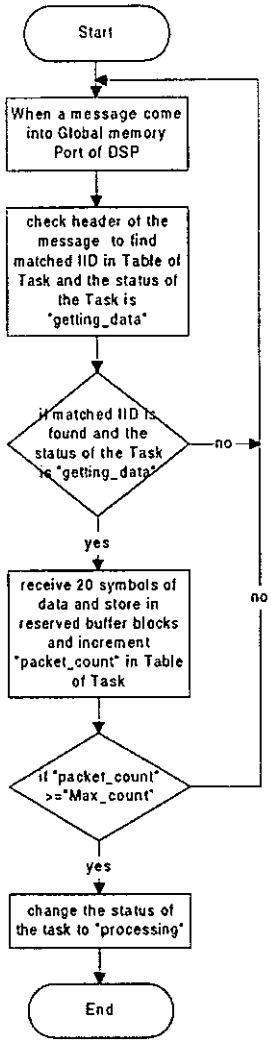


Figure 6.3 Flowchart of Simulation on DLC

We have simulated the process of a DSP for reading different amount of 80-bytes packets from its global memory port. In addition, the DSPs have to check the header information of the packet with the Table of Tasks, see Section 5.4, and then transfer the data of the packet to an appropriate buffer block. Since this control mechanism is applied to each packet, we can accurately estimate the time required by a DSP to handle each data transaction produced by the QuickRing controller.

Based on our experimental results, the data transfer rate of the DLC is presented in Figure 6.4 and it is estimated that 6.44ms is required for the DLC to deliver 256k of image data to the QuickRing controller. For the processing step t4 and t11, 1.82ms and 1.84ms are required to deliver data in a QuickRing network. With the additional time that is requested by DSPs to receive and transmit data, the modified values of t4 and t11 become 8.26ms and 8.28ms respectively. We have also found that the period required by the DLC to check the number of empty buffer block is less than 1.25μs (t6). The adjusted values of the processing steps are list in Table 6.2.

Table 6-2 Time required by Communication Steps in Heterogeneous Image-processing

Symbol	t2	t4	t5	t6	t7	t9	t11	t13	t4'	t5'
Value	8.8 μs	8.26 ms	3.35 μs	1.25 μs	1.20 μs	8.6 μs	8.28 ms	86.42 μs	1.818 ms	3.35 μs

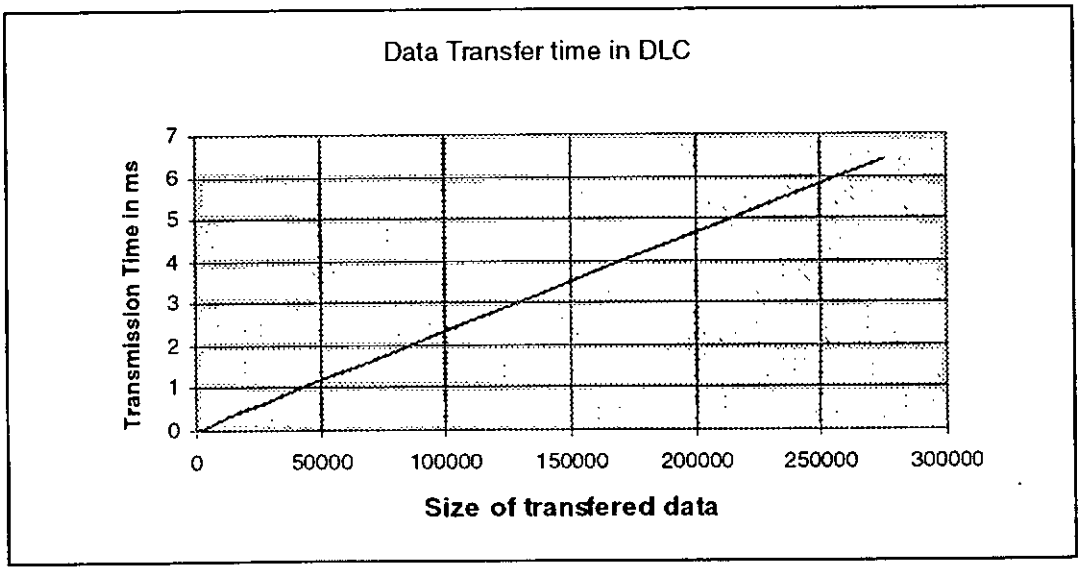


Figure 6.4 Data Transmission Time in DLC

6.2.2 Results Analysis

In order to simplify our analysis, we have assumed that the time required by HPs to check the empty buffer blocks is equal to that required by the DLC. Since HPs is Pentium type processor that is running at a clock speed of 200 MHz, or

higher, with high-speed data cache, the practical value should be much smaller than this assumption.

The capability of the proposed heterogeneous image processing system can be measured based on the time between MC receiving results of two successive image frames, namely Ttotal. In order to find the value of Ttotal, we have to identify steps which dominate the time spent on the image processing operation. We have assumed the time required by a HP to check its buffer is equal to that required by DLC, so that the values of t1, t6 and t8 are equal to 1.25μs. In addition, the values of t2, t5, t7, t9 and t13 are smaller than 0.1ms, see Table 6-2.These values are very small comparing to the value of t4 and t11, so that they can be ignored. Therefore, the value of Ttotal depends on t3, t4, t10, t11 and t12.

Ttotal is equal to the time between MC receiving the processed results of two successive image-frames from HPs. Because of the pipeline effect, Ttotal is equal to the maximum operation steps. This value may be equal to the time between the ending of two successive cycles in step t4 which is equal to t3 + t4, t10 or t12, as shown in Figure 6.5. They will be equal to the maximum value of (t3 + t4), t10 and t12.

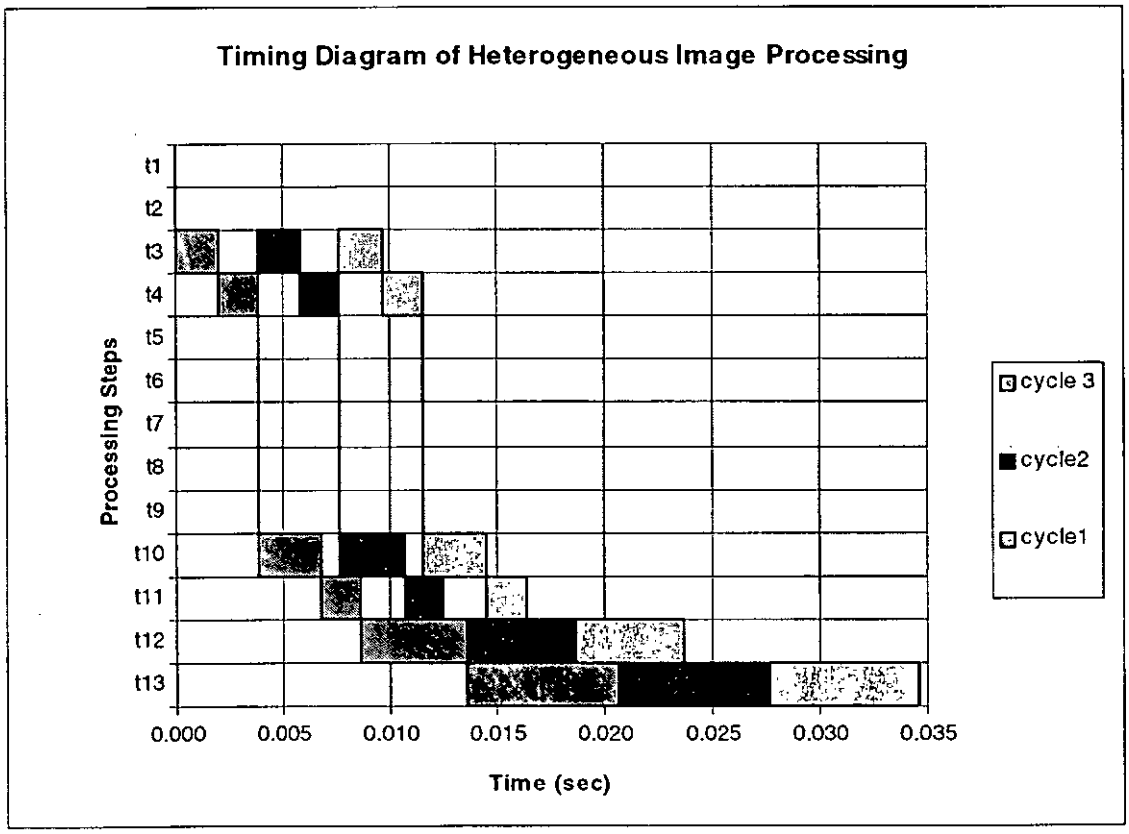


Figure 6.5 Timing Diagram for the Heterogeneous Image-processing

Referring to Figure 6.5, Ttotal can be modelled as follow.

$$Ttotal = Max\{t3 + t4, t10, t12\} \quad (1)$$

If Ttotal is shorter than 40ms, the proposed system will be able to handle real-time image processing tasks. To meet such a requirement, the value of (t3 + t4), t10 and t12 must be shorter than 40ms. Since only the value of t4 is known so that value of t3 must be shorter than 31.74ms. In addition, the value of t10 and t12 must be shorter than 40ms, see Table 6.3.

Table 6-3 Summary of Limitation of Time Step

Time Step	Description	Maximum limitation
t3	MC generating instructions that are going to send to DLC and HPs.	31.74ms
t10	DSP Layer Controller processing the received data.	40ms
t12	HPs processing the results came from DLC.	40ms

In the previous analysis, we have assumed that the congestion, which is caused by simultaneous data transmission from HPs (t13) and DLC (t11), can be ignored since the value of t13 is too small compare with t11. However, when the data size of results transferred by HPs increases, this assumption may become invalid because the increase in data size will increase the value of t13. As mentioned in Section 6.2, the completion time of the operation t11 will be delayed by a duration equals to that of t13. Therefore, the time required to complete the data transmission of t11 would be the sum of t11 and t13. Therefore, Ttotal will become dependent on the sum of t11 and t13, equation (1) is then modified into (2).

$$Ttotal = Max(t3 + t4, t10, t12, t11 + t13) \quad (2)$$

For a real-time system Ttotal < 40ms, so that t13 < 40ms - t11 and t13 < 31.72ms.

Based on the simulation on QuickRing's Multicast mode mechanism, we have found that the time required to complete the data transmission for step t13 is depended on the size of data being transmitted. In Figure 6.6, it illustrates the values of t13 depends the quantity of data transmission. It shows that 26.97ms is required to transmit 256k bytes of data.

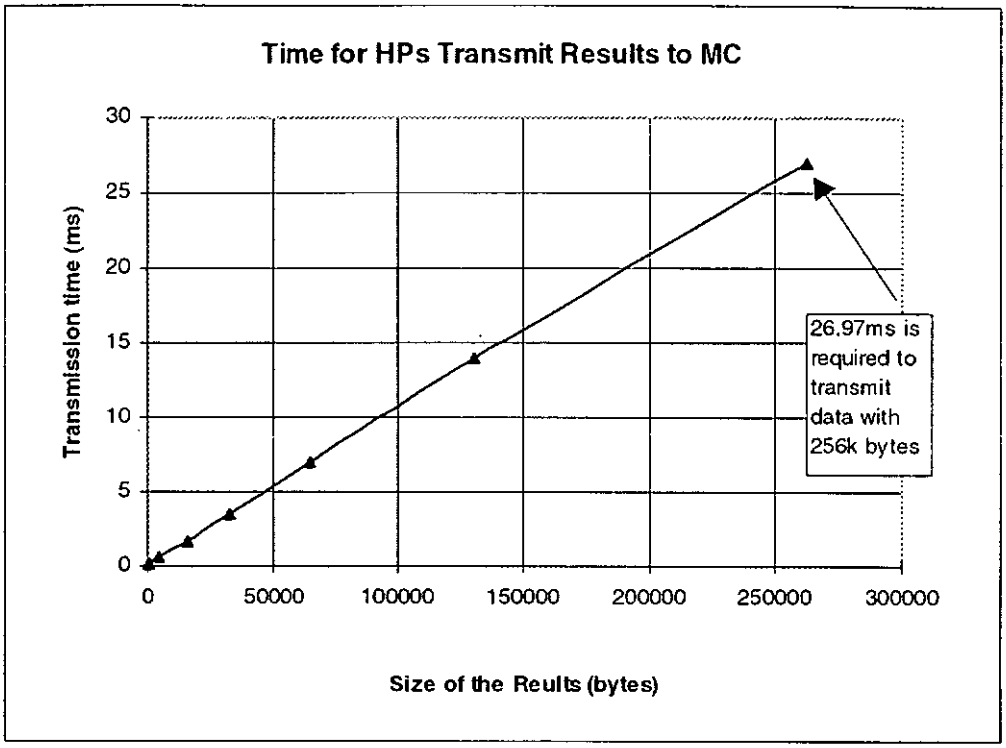


Figure 6.6 Transmission Time for HPs to send results to MC

Referring to Figure 6.6, we can derive the equation to determine the processing time for step t13.

$$t13(ms) = 1.03 \times 10^{-4} \times size\ of\ data(bytes)$$

$$size\ of\ data(bytes) = \frac{t13(ms)}{1.03 \times 10^{-4}} \qquad (3)$$

Considering the real-time processing requirement, we can conclude that the size of data transmitted from HPs to the MC must be less than 307960 bytes.

6.2.3 Efficiency Analysis

Efficiency of communication and control mechanism is defined as the ratio of the time that are not spent on operations that unrelated to data transmission over the total operation time except the time spends on low-level and high level image-processing. In our simulation, the time spent on the operations that not related to data transmission can be classified into two classes. In the first class, the time steps are totally spent on overhead. Operations of the checking buffers, sending buffer information and the MC generating instructions, belong to this class. Class

1 includes steps t3, t8 and t9. Class 2 includes the time steps with only part of the period being spent on overhead. The operations of transmitting data or instruction belong to this class. For our communication protocol, one out of twenty of these periods are spent on overhead. Class 2 includes steps t4, t11, and t13.

Therefore, efficiency of our communication network can be determined from the following equation.

$$
\begin{aligned}
efficiency &= 1 - \frac{Class\ 1 + Class\ 2}{summation of all time steps except shigh level and low level proces\sin g} \\
&= 1 - \frac{t3 + 0.00985ms + \frac{16.63}{20}ms}{t3 + 16.28ms} \\
&= 1 - \frac{t3 + 0.841ms}{t3 + 16.28ms}
\end{aligned}
\tag{4}
$$

We found that the processing period of t3 dominates the time spend on operation unrelated to data delivery. The efficiency of the system can be increased by employing a powerful processor in MC and reduce the time required to generate instructions. The effect of t3 on the efficiency is illustrated in Figure 6.7

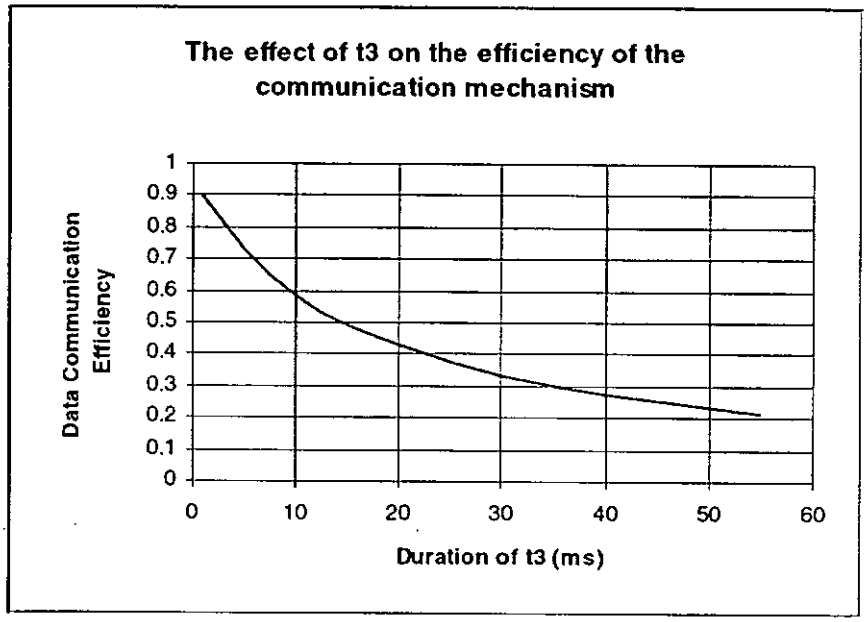


Figure 6.7 The effect of t3 on the efficiency of the Communication Mechanism

6.3 Application for the Two-layer System

In previous discussion, we have evaluated the performance of the communication mechanism based on the size of data to be transmitted in the QuickRing Network. In this section, we will study an application, a pattern recognition problem, of the system. The purpose of this study is to illustrate operations to be carried out at the two processing layers. In addition, performance of the system based on the study will be presented.

The problem being considered is to process a grey level image and recognise 2-D objects carried by the image. Feature points are first extracted from the image and they are then used as to match with a database, where object models are stored. The operations applied and the process sequence are depicted in Figure 6.8. Image processing tasks included in this example can be divided into 2 levels, namely low-level and high-level. In low-level, image is transmitted to the Low Level Layer and several operations, such as thresholding, edge detection, Hough transform [45] and feature points extraction are being performed. The extracted feature points are broadcast to high level layer PCs and geometric hashing [36] is used as a pattern recogniser based on the extracted feature points. The results obtained from geometric hashing are sent to MC and this will be objects being recognised.

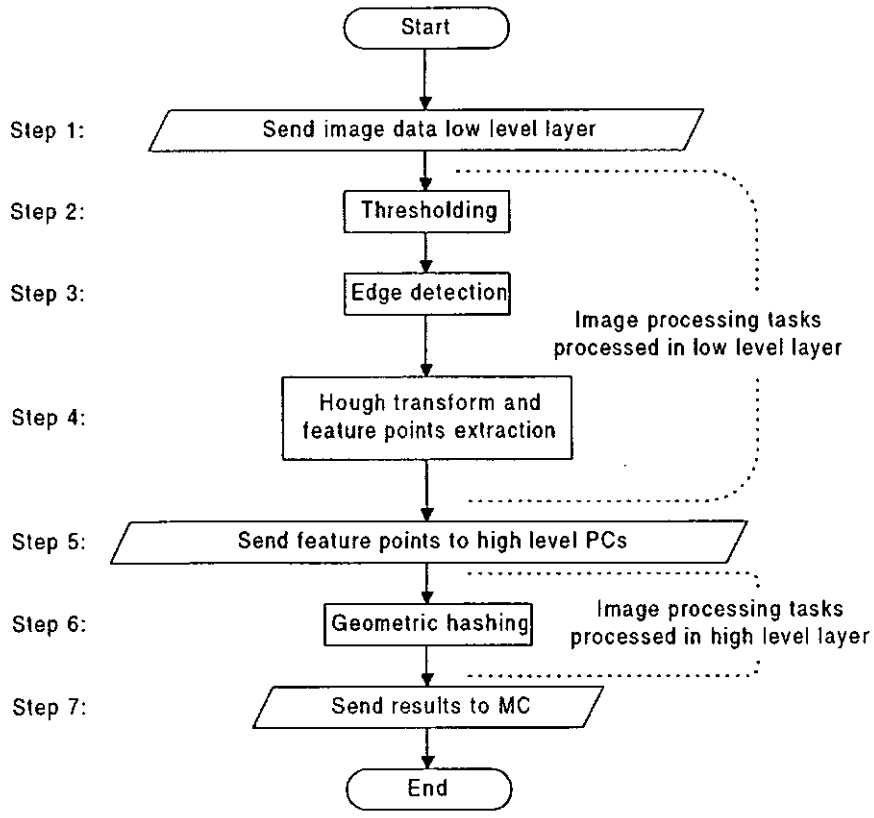


Figure 6.8 Practical process executed on proposed system

6.3.1 Thresholding

Thresholding is the first operation being applied and it converts the grey level input image to a binary image. The threshold level of each pixel is determined by the intensity of its neighbourhood pixels. The maximum and minimum intensity level of the 3x3 neighbours are first determined. If the difference between the maximum and the minimum is greater than a preset value, the threshold level (T) is the average of maximum intensity and minimum intensity. Otherwise, the threshold level is equal to the difference of the two intensity levels over two, see equation (5).

$$\begin{aligned} &T = (\text{minimum} + \text{maximum})/2, \text{ if maximum-minimum} > \text{preset value} \\ &T = (\text{maximum} - \text{minimum})/2, \text{ if maximum-minimum} < \text{preset value} \end{aligned} \quad (5)$$

Once the threshold level is determined, the intensity value of that pixel is compared with T. If the intensity value is larger than T, the value of that pixel is set to 1. Otherwise, the value is set to 0. After thresholding, pixels, which lie within the region of an object is highlighted.

6.3.2 Edge detection

After thresholding, the Sobel [45] 3×3 edge operator is applied in order to select edges, or boundary points, of objects. After the process, the thickness of an edge of the object is two pixels because the pixels at both sides of the boundary are highlighted. A logical "AND" operation between the resultant image and the source image is applied and this will reduce the thickness of the edges into a single pixel.

6.3.3 Hough transform and feature points extraction

After edge detection, the image contains the boundaries of objects in the scene. The edges are straight line segments. Hough transform is applied to detect straight lines in the image. First the image function $f(x,y)$ is converted into the Hough space (ρ,θ) using the following equation:

$$\rho = x\cos\theta + y\sin\theta \quad (6)$$

This transform provides L co-linear edge points on the line $\rho = x\cos\theta + y\sin\theta$ to generate L sinusoidal curves in the (ρ,θ) space and these curves intersect at a single (ρ,θ) point. By incrementing θ from -90° to +90°, corresponding ρ values are obtained for L edge points and corresponding points in (ρ,θ) space are

accumulated. These operations are carried out for all the edge points found in the image. The significance or strength of lines can be determined by referring to the magnitude at (ρ,θ). The points in (ρ,θ) space, which magnitude are greater than a threshold value, are considered as lines of the great significance. Then all (ρ,θ) values with great significance are used to compute the pixels on the line using the following equation:

$$y = (\rho - x\cos\theta)/\sin\theta, \text{ for } 0°<\theta<45°, \text{ and } 135°<\theta<180° \quad (7)$$

$$x = (\rho - y\sin\theta)/\cos\theta \quad , \text{for } 45°<\theta<135°$$

The computed pixels are compared with the resultant image after edge detection so that the starting point and the end point of a line can be extracted. The two points are regarded as feature points and their co-ordinates are results generated from low level image processing.

6.3.4 Geometric hashing

In geometric hashing, objects are represented as sets of geometric features, such as points. Their geometric relations are encoded using a minimal set of such features. It can be used to recognise flat objects. The algorithm consists of two phases, pre-processing and recognition. The pre-processing phase is executed off-line and only once. In pre-processing phase, a set of models is specified using their feature points. For each model, all possible pairs of feature points are designated as a basis set. The co-ordinates of other feature points are computed relative to each of these basis. These co-ordinates are then stored in a hash table. The entries in the hash table comprise of (*model, basis*) pairs. In the recognition phase, the feature points extracted in low level layer are sent to the high level layer, an arbitrary pair of feature points is chosen as a basis and the co-ordinates of the other feature points in the scene are computed relative to this basis. The new co-ordinates are used to compare with the entries in the hash table. The computation of co-ordinate will continue with a different pair of feature points chosen as the basis. Votes are accumulated for the (*model, basis*) pairs stored in the hashed locations. The pair winning the maximum number of votes is chosen as a candidate for matching.

6.3.5 Performance Evaluation

An image (refer to Figure 6.9) is chosen as a test image. The image has 128 × 128 pixels with 8 bits grey level. As depicted in Figure 6.8, the complete process consists of 7 steps. However, we mainly concentrate on the time spent in data exchange between the two processing layers and Master Controller. The steps,

referring to Figure 6.8, contribute to data exchange include the followings.

Step 1, Master Controller sends image data to low level layer. The image data is collected by DLC in low level layer.

Step 5, DLC broadcasts feature points to high level PCs.

Step 7, high level PCs send results to MC.

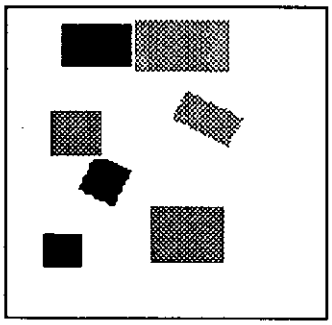


Figure 6.9 Images used to test the system

Refer to section 6.2, time spent in Step 1, Step 5 and Step 7 are represented by t4, t11 and t13 respectively. The data communication time is mainly depended on the size of data. In step 1, size of the picture shown in Figure 6.9 is 16384 bytes. The time spent in Step 1 (t4) is 0.42ms.

After low level image processing, the feature points will be packed in the following format (refer to Table 6.4) before being broadcast to High Level PCs. The first symbol of message represents the number of feature points which have to be processed in Geometric Hashing. Each following symbol in the packet represents two feature points. Starting from the second symbol to the end of message, byte 0 and byte 1 represent the coordinates of the feature point, while byte 2 and byte 3 represent the coordinates of next feature point.

Table 6.4 Data format of feature points to be broadcasted to High Level PCs

	Byte 0	Byte 1	Byte 2	Byte 3
Symbol 1	[31-0]: number of feature points			
Symbol 2 to Symbol N	x coordinate of feature point	y coordinate of feature point	x coordinate of next point	y coordinate of next point

The number of extracted feature points is 44, for the image shown in Figure 6.9. Then, they will be further packed in QuickRing Packet. Therefore, results can be transfered in two packets. The data transfer times are shown in Table 6.5.

Table 6.5 Transfer Time for Extracted Feature Points to High Level Processors(in μsec)

Number of Points	Size of data (bytes)	Number of Packets	Transfer time (t11) μs
44	92	2	3.875

After HPs have completed Geometric Hashing, the result will be sent to MC in the format shown in Table 6.6. The first symbol represents the number of Objects found. Every two bytes, after the first symbol, represent an object name.

Table 6.6 Data format of result transmitted from PCs to Master Controller

	Byte 0	Byte 1	Byte 2	Byte 3
Symbol 1	Number of objected found			
Symbol 2 to Symbol n	Object Name		Object Name	

During pre-processing in Geometric Hashing, we have used 4 object models with 4 feature points each for generating the hash table.

Models with maximum votes and the number of their votes over certain criteria will be considered as results. The results will be sent to MC. After the feature points extracted from the image have processed by the high level layer, 3 objects are recognised.

Table 6.7 Transfer Time of final results transmitted from PCs to MC (in μsec)

Number of Objects	Size of data (bytes)	Number of Packets	transfer time (t13) μs
3	8	1	3.350

For the test image, the time required by DLC to send results to HPs is 3.875 μs (t11). Time required by HPs to send the results to MC is 3.350μs (t13). The total time required to transfer results from low level layer to high level layer and from high level layer to the Master Controller is very short for our current case. In this section, we have illustrated one possible application for the two-layer system and how processing operations are being distributed between the layers is presented.

6.4 Conclusion

In this Chapter, a simulation of the system's operations has been performed. The communication network is able to support the bandwidth requirement for real-time image processing task if the size of resultant data generated by high level image-processing is less than 300kbytes. In addition, the simulation on the DLC produces accurate results for transmission time on different operation steps. Also efficiency analysis shows that t3 (the time spent by MC to generate messages to be distributed to DLC and HPs) has significant effect on the efficiency of the communication network.

Due to the pipeline feature, Ttotal equals to the maximum value of the operation steps, refer to equation (1). From the case study, we have determined that the time spent on data communication (t4, t11 and t13) is all less than 0.5 ms. The results show that timing requirements of communication operations are much shorter than high level and low level image processing operations. The operations of message delivery will not affect the value of Ttotal. Therefore, the communication and control mechanism is effective and is capable to handle communication requirements generated by a real-time image-processing job.

Chapter 7

7. CONCLUSION

7.1 Summary

In this thesis, the architecture and interlayer communication mechanism of a heterogeneous image processing system have been presented. It is based on the Multicast mode communication mechanism of the QuickRing network. A problem emerges from our design is related to the buffer utilisation. Buffer is used to store data and control commands passing between elements of the system. In order to handle the buffer properly, we have implemented a buffer checking algorithm in High Level Processors and the DSP Layer Controller. The algorithm makes use of a table to maintain the availability of buffers in both High Level Processors and the DSP Layer Controller. Such an arrangements increase the reliability of the network.

In Chapter 2, the requirements for the interlayer communication for heterogeneous image processing tasks were defined. A survey of the current communication technologies is presented. Features of different network technologies are compared. The reasons for choosing the QuickRing network are based on its speed and flexibility to be adopted in our design.

Chapter 3 describes the QuickRing network and explains design criteria that were made. The main characteristics of QuickRing are explained in detail, especially the Multicast data transmission mechanism and the routing scheme for the multi-ring topology. The advantages and disadvantages of the network are discussed. Based on software simulations, it is proved that the Multicast mode is more efficient then the Directed mode when data is required to broadcast to several nodes in the network. However, Multicast mode of data transmission is not reliable.

The system architecture and components of the communication network are presented in Chapter 4. System architecture and hardware components including both the DSP Layer Controller and the bridge hop are presented. The DSP Layer Controller consists of two DSPs and a QuickRing Controller. The global memory ports of the two DSPs are connected to the client port of the QuickRing Controller in order to double the data transfer rate. The bridge hop consists of two QuickRing Controllers connected back-to-back via an addressing conversion table, see **Figure 4.5**. The functions of the hardware in the system and how they deal with the problem of reliability of QuickRing are described.

Chapter 5 describes the communication and control mechanism for the heterogeneous image processing system. Different software components required to carry out the task are also discussed. The Master Controller is responsible to maintain a table which stores information regarding the buffer availability. Then it can divide the image processing job into sub-task and distribute them to the DSP Layer Controller (DLC) and High Level Processors. The processed result will then be transferred back to the Master Controller.

Finally, in Chapter 6 we presented the simulation and the analysis on the performance of the proposed communication mechanism. Simulation of system's operations was performed to illustrate the timing relation between each communication and processing step. The simulation on the communication of the DLC reflects its performance under the proposed control mechanism. The performance of the communication and control mechanism is analysed in the Chapter.

7.2 Conclusion

In this dissertation, a two-stage multi-layer architecture is proposed. The low level layer is a Digital Signal Processors (DSPs, TMS320C40) array and high level layer is a cluster of Personal Computers (PCs). Each processor in the DSP layer has six communication ports, each of them can sustain a maximum throughput of 20Mbytes/s. Since operations at high level image processing involve AI-based computations and it may require access to databases, therefore, PCs are well suited for this level.

In the pre-processing phase, image data is delivered to the low level processors and operations are performed under a SPMD control paradigm. The processed data is delivered to the high level processors where feature extraction and recognition tasks are performed. As data is distributed among processors in the low-level layer, under the SPMD paradigm, processors in the high-level layer must collect all processed results available in the low-level in order to analyse the whole image. Therefore information from the low-level layer must be broadcast to all processors in the high-level.

A simulation model of QuickRing has been developed to study its data transmission behaviour and the model also assists us to verify the performance of QuickRing. From our studies, we can conclude that the QuickRing controller provides a feasible solution for the implementation of the communication network for a heterogeneous system.

We have presented an efficient and reliable interlayer communication mechanism, which is based on QuickRing, to deal with the communication requirement of the heterogeneous image processing system. A high speed local

area network is proposed. The simplicity of the control algorithm and the high data transfer rate of QuickRing reduces the communication overhead. QuickRing's architecture is flexible and it enables the expansion of the network easily. Besides, QuickRing provides a mechanism, the Multicast mode, for broadcasting data. This is an important feature for implementing machine vision tasks. From our simulation results, we have found that the communication mechanism can complete data transmission tasks within the limitation that is imposed by real-time image operations. So it is feasible to apply the system for real-time image processing.

Although the communication mechanism of the proposed image processing system is reliable and efficient, it also has disadvantages. Since the data throughput of the DSP (TMS320C40-40) is only 100Mbytes/s, the DSP Layer Controller becomes a communication bottleneck between the Low Level Layer and the QuickRing network although based on our studies, this bottleneckdoes not affect our system's performance significantly. The problem can be improved if DSP with higher data transfer rate is employed, such as TMS320C40-60 [44] which has a throughput of 120Mbytes/s. In addition, congestion may occur when two or more nodes are transmitting data through a common network path. This problem cannot be prevented and we can only limit the effect of congestion by limiting the size of data being transmitted.

In order to realise the system, hardware components are specially designed for the implementation of the communication network and these include the interface between DSP layer and QuickRing network and bridge hop between two QuickRing networks. Due to the flexibility of QuickRing, these components can be realised with a simple design and their functions are tested using a software simulator. In addition to the hardware components, a new communication and control mechanism has been developed to achieve reliable and efficient data communication. A simulation model has been implemented for performance analysis. The communication mechanism also avoids buffer overflow at the receivers thus the reliability of the network can be maintained.

7.3 Future Work

The maximum number of nodes in a single QuickRing network is limited to 16, so that 15 HPs and one bridge hop could be connected in the high-level processor ring network. To increase the number of HPs, an additional ring can be connected to the control ring. Similarly, the control ring can support 16 nodes and 15 of those can be used by processing layers. Currently, the low level layer consists of an array of TMS320C40 processors. The number of DSP can be increased by simply inserting additional DSPs in the existing DSP array. However, this arrangement will increase the propagation delay within the DSP layer. Alternatively, expansion can be achieved by connecting an additional DSP

Layer to the control ring. With this arrangement, the Master Controller can broadcast instructions to the two DSP Layer Controllers. Further research should be conducted to investigate the performance of the system when different expansion schemes are applied.

Hardware components proposed in this project should be built to verify their performance. Also, further investigation for the improvement of the data transfer rate of the DLC should be performed. In Chapter 5, code generation and load partitioning algorithm of Master Controller were not defined. The operation time of code generation is limited to 31.74ms for handling real-time image processing operation. Further research should be focused on the design of the code generation algorithm and its performance. Besides, effective load partitioning scheme can improve the performance of both high level image processing and low level image processing operations and the effect of load partitioning scheme on the performance of image processing should be investigated.

APPENDIXES

A. QuickRing Technical Details

This chapter mainly description the details of QuickRing technology and Multicast operation.

A.1 General Description

QuickRing technology is a point to point data transfer architecture designed to facilitate high speed data streams between devices, boards and systems. The QuickRing architecture can be applied both inside the chassis as well as outside the chassis to increase data throughput. Each QuickRing Controller in a ring is capable of streaming up to 280M samples per second on 6-bit links, including protocol overhead. Multicast, Broadcast, and Fixed Packet size are provided for data transmission. This device is intended for use in applications that handle high bandwidth data streams associated with graphics, compressed and uncompressed video, disk arrays, high-speed localised networks, multiprocessor systems, and peripherals over cable. The controller can be used to augment the performance of traditional backplane buses in personal computers, workstations, and high-end systems. It is also useful for routing high-bandwidth streams in systems that are larger or topologically more complex than bus-based systems.

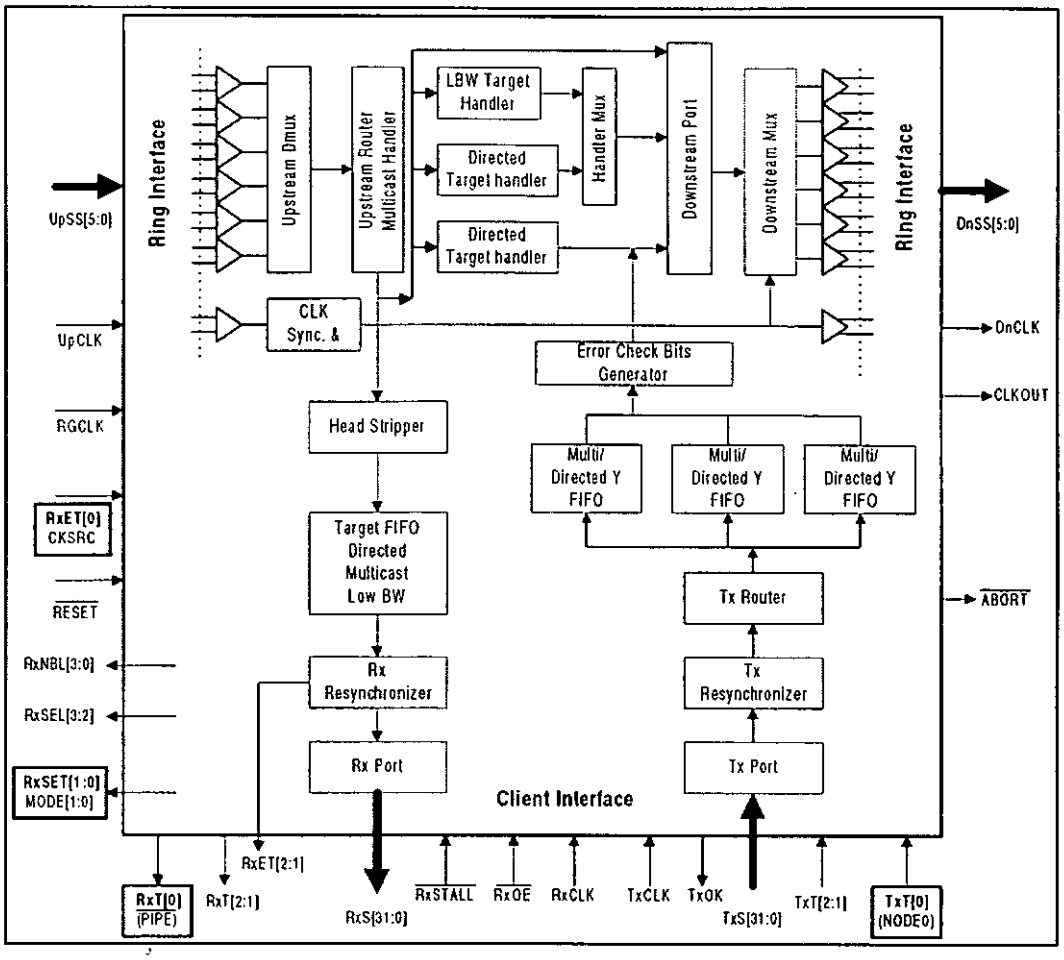


Figure A-1 Block diagram of QuickRing Controller

As shown in Figure A-1, QuickRing have Ring interface and client interface. Correspondence pins assignment are described in Table A-1. Each interface has two ports. All ports are unidirectional so that incoming and outgoing data can be queued simultaneously.

The Ring Interface is consists of upstream port for arriving traffic, and downstream port for departing traffic. The Ring Interface forms the link to other nodes on the point-to-point QuickRing architecture. QuickRing connects multiple nodes by attaching the upstream port of each node to the downstream port of another node. The ring ports, upstream and downstream, are 6 bits wide plus a clock. The ring interface is implemented using LVDS[23] drivers and receivers. The Ring Interface signals are not accessible from the board except through the controller. The on board logic connected to the controller via the Client Interface.

Table A-1:Signal Description

Pin Name	I/O	No.	Description
$\overline{\text{RESET}}$	I	1	RESET: When this input is released, the initialization sequence begins.
$\overline{\text{ABORT}}$	O	1	ABORT: When asserted, it indicates that a failure was detected. $\overline{\text{ABORT}}$ is negated by asserting Reset.
$\overline{\text{PIPE}}$	I	1	PIPE: When $\overline{\text{PIPE}}$ is negated (non-pipelined timing), at the Client ports, both the symbol and type fields correspond to each other during the same clock cycle. When PIPE is asserted (pipelined timing), the timing of the Type field leads by one clock at the receive port and trails by one clock at the transmit port. (The type and symbol fields are pipelined.)
RxT[0]	O		EXTENDED RECEIVE TYPE FIELD: The function of this pin is the same as PIPE until the release of reset. This pin becomes the third receive port type bit. This gives 8 symbol functions. They are used to identify the multicast heads and explicit tail symbols.
NODE0	I	1	NODE0: When asserted, the controller is configured as having Node ID 0. Node 0 is responsible for governing the initialization process of the ring.
TxT[0]	I		EXTENDED TRANSMIT TYPE FIELD: The function of this pin is same as NODE0 until the release of reset. After release of reset, this pin becomes the third transmit port type bit. This identifies the multicast head and the explicit tail symbol used to fix stream and packet length on the client and ring ports.
RGCKL	I	1	RING CLOCK: This clock input is the time-base for the ring interface. A clock input should be present when the CKSRC pin is asserted. When CKSRC is negated, RGCLK should be tied low.
CKSRC	I	1	CLOCK SOURCE: Designates the source of the ring interface. A clock input should be present when the CKSRC pin is asserted. When CKSRC is negated, RGCLK should be tied low.
RxET[0]	O		Extended Receive Early Type: This completes the Early Type field.
CLKOUT	O	1	CLOCK OUT: If CKSRC is asserted, the CLKOUT is frequency-locked to the RGCLK.
UpCLK	I	2	UPSTREAM CLOCK: This LVDS input clock comes from the neighbour upstream node and drives the ring interface when CKSRC is negated.
UpSS[5:0]	I	12	UPSTREAM SUB-SYMBOL: These 6 LVDS inputs for the Ring interface carry the divided 42-bit symbol from the downstream port of the previous node.
DnCLK	O	2	DOWNSTREAM CLOCK: This LVDS output clock signal is derived from the clock that drives the Ring interface. The transitions on the DnSS are in phase with transitions on the DnCLK signal.
DnSS[5:0]	O	12	DOWNSTREAM SUB-SYMBOL: This LVDS outputs for the Ring interface carry the divided 42-bit symbol for the upstream port of the next node.

Pin Name	I/O	No.	Description
TxT[2:1]	I	2	TRANSMIT TYPE: On the Client interface, this field defines (as head, data, frame or null) the contents of TxS: in the previous clock cycle when $\overline{\text{PIPE}}$ is asserted, pipelined timing. in the current clock cycle when $\overline{\text{PIPE}}$ is negated, non-pipelined timing.
TxS[31:0]	I	32	TRANSMIT SYMBOL: On the Client interface, these signals form the data path of the transmit port.
TxOK	O	1	TRANSMIT OKAY: On the Client interface, this is the transmit port status signal. It tells the client whether or not another non-null symbol can be accepted. Loading of non-null symbols must cease within 20 symbols of the negation of TxOK. Transmission may not resume until TxOK is reasserted.
RxCLK	I	1	RECEIVE CLOCK: On the Client interface, all receive port signals are synchronous to the rising edge of this clock. Except $\overline{\text{RxSTALL}}$, which is sampled on the following edge of RxCLK.
RxT[2:1]	O	2	RECEIVE TYPE: On the Client interface, this field defines (as head, data, frame or null) the contents of RxS: in the next clock cycle when $\overline{\text{PIPE}}$ is asserted, pipelined timing. in the current clock cycle when $\overline{\text{PIPE}}$ is negated, non-pipelined timing.
RxS[31:0]	O	32	RECEIVE SYMBOL: On the Client interface, these signals form the data path of the receive port.
$\overline{\text{RxSTALL}}$	I	1	RECEIVE STALL: On the Client interface, when $\overline{\text{RxSTALL}}$ is asserted: When $\overline{\text{PIPE}}$ is asserted, pipelined timing: RxS shall remain for the next clock cycle. When $\overline{\text{PIPE}}$ is negated, non-pipelined timing: RxT will indicate a null for the next clock cycle and RxS shall remain.
$\overline{\text{RxOE}}$	I	1	RECEIVE OUTPUT ENABLE: On the Client interface, when asserted this signal enables outputs RxS[31-0]. When negated, the RxS are TRI-STATE.
RxET[2:1]	O	2	RECEIVE EARLY TYPE: On the Client interface, this field identifies in advance whether the information entering the Rx Port block is a head, data, frame or null.
RxNBL[3:0]	O	4	RECEIVE NIBBLE: On the Client interface, it contains one of the 16 selectable fields of two readable internal areas (Diagnostics bits, RxS driver).
RxSEL[3:0]	I	4	RECEIVE SELECT: On the Client interface, selects one of the 16 fields appearing on the RxNBL. Codes from 0 to 7 select 4-bit fields at the current output driver of RxS, codes of 8 or above select internal diagnostics status bits.
MODE[1] (RxSEL[1]) MODE[0] (RxSEL[0])	I I		OPERATION MODE SELECTION (Either QR0001 or QR1001): The operation mode selection occurs on the release of reset. If MODE0 is sampled low and MODE1 is sampled high on the release of reset, then the part will operate in the Enhanced QuickRing mode. If on the release of reset, these two pins are sampled in any other condition, then the part will operate in the QR0001 mode.
Vcc	n/a	13	POWER PINS
GND	n/a	29	GROUND PINS
Note 1: $\overline{\text{SignalName}}$: The indicates that the signal is active low.			

The Client Interface is consists of a transmit port for locally generated symbol streams, and a receive port for locally-absorbed symbol streams. The transmit and receive ports have a 32-bit data path which use TTL compatible I/Os. The Transmit(Tx) and Receive(Rx) ports each have a separate clock plus control signals for information flow. Also, some internal status bits can be read through the receive interface. All on board circuitry interfaces to the Client transmit and receive ports, never to the Ring ports. QuickRing transmits data streams between nodes on the ring. The goal of QuickRing is to pipeline data streams and not just to facilitate memory access. QuickRing is logically equivalent to placing a large FIFO between pairs of QuickRing nodes.

A.2 Client Interface

Type and symbol fields at the client ports

The QuickRing client can multiplex multiple independent data streams onto and from the transmit (Tx) and receive (Rx) ports of the controller. The type fields (TxT, RxT) distinguishes the contents of the symbol fields(TxS, RxS). The type field identifies the nature of the symbol field information at the 32-bits ports as: Directed Head, Multicast Head, Data, Data-Tail, Frame, Frame-Tail and null. Details are explained in **Error! Reference source not found.**.

Table A-2. Client Type Field Definitions (TxT/RxT)

Type	Name	Description
0	Directed Head	First symbol of a packet, specifying the path to a single target. This stream is reservation based.
1	Multicast Head	First symbol of a stream destined for one or more targets. This stream does not use the reservation based ring protocol.
2	Data	32-bit payload symbol is a Data.
3	Data-Tail	Data symbol which is the last symbol in a stream.
4	Frame	Specially marked 32-bit payload symbol is a Frame.
5	Frame-Tail	Frame symbol which is the last symbol in a stream.
6	Reserved	Future use
7	NoSymbol	No associated symbol

A.2.1 Client Transmit Port

The transmit port can be thought of as the input to a bank of fast, deep FIFOs connected to other nodes on the ring. The receive port can be treated as the output of the bank of FIFOs connected to other nodes on the ring. The transmit block of the controller is formed by: Tx Port, Tx Resynchronizer, Tx Router, and 3

independent FIFOs. All of these blocks form the transmit pipeline.

1. The Tx Port is the first stage into the transmit pipeline. The Transmit port is a 4 deep pipeline.

2. The Tx Resynchronizer is a 32-deep asynchronous FIFO in the path between the Tx Port and the Tx Router.

3. The Tx Router directs the streams to the appropriate channel efficiently.

4. Multicast/Directed FIFOs X and Y are meant for handling one independent high bandwidth stream each, and the LB (Low Bandwidth) FIFO is meant for low bandwidth transmissions. The FIFOs contain the data/frame part of the client stream. The stream can be directed or multicast.

The sole purpose of providing two normal (high bandwidth) FIFOs (X and Y) is so that the client may switch from transmitting one stream to another without slowing down or wasting available ring bandwidth during the context switch.

On release of $\overline{\text{RESET}}$ any payload symbols at the transmit port are ignored until the first head symbol is presented at the input of the Tx Port. QuickRing controller always checks for consecutive heads and ignores all redundant heads. The type and symbol fields are latched internally according to the timing specified by the state of the PIPE signal.

When the clients starts a transmission, it writes a head followed by a stream of payloads. QuickRing receives these symbols through the transmit port and directs them to either the X, Y or LB FIFO. Any head symbol with the CONN field equal to 1 is always routed to the LB FIFO, as is every payload symbol following such a head. Any other head with the CONN field equal to 0 and all payloads following such a head are routed to either the X or Y FIFO.

QuickRing controller can handle one independent data stream through each of the X and Y FIFOs, a total of two streams at once. Even if the FIFO is not full, the FIFO will store data associated only with a single head. Multiple data streams with various heads will not be held in a single FIFO. The subsequent data streams, with different heads, will be held in the Tx pipeline, until either FIFO X or Y empties, and the data (with the different head) is allowed to further proceed in the pipeline.

5. The LB FIFO. Several streams with different heads can flow through the LB FIFO at one time. When several payloads are loaded, following a signal head, a head will be generated for each payload.

For low bandwidth, multicast and directed transmissions, controller will keep TxOK asserted for as long as there is space for 20 or more symbols in the transmit pipeline. As soon as the transmit pipeline has space for only 20 more symbols, TxOK negates. The initial negation of TxOK indicates to the client interface that it must stop transmitting non-null symbols soon. TxOK is the only handshake mechanism at the transmit port. If TxOK asserts again, the count is voided and the client can write to the TxPort as many symbols as it wants. If TxOK negates again, the client must stop writing non-null symbols within 20 valid transactions.

The client may pause transmission at any time by presenting the null symbol code to the transmit port. When the client interface wishes to begin transmission of a data stream, the client first writes a head to the transmit port. From then on, every payload symbol (type = data or frame) sent to the transmit port is unbounded. However, if a new head is written to the transmit port, the data stream that follows is associated with the new head. If at any time the client is not prepared to transmit either a payload or a new head, a null symbol may be introduced into the transmit data stream. Null symbols do not propagate into the QuickRing controller. Logically distinct data streams can be multiplexed together and loaded into the Transmit port. The client is free to switch between source streams at its convenience, as long as it introduces a new head when the switch occurs.

Directed Transmission

The directed packet is transmitted with buffer reservation scheme. The transmit port evaluates the connection, target, and all hop fields,(refer to Table A-3). The ACCess field should be set to zero for variable stream size and one for fixed stream size. The Conn field indicate if the packet is Low Bandwidth packet or normal packet. The controller adds its own ID to the source field internally. The Trgt field contains the node ID of the target node. The Hop fields (Hop1 to Hop4) contain the node ID of the bridge hops that the packet will pass through. When the packet pass through a RxPort, the position of Srce, Trgt, and the Hop fields will shifted(refer to Table A-4). Therefore, the position will shifted when the packet pass through a bridge hop. This allows for an easy response packet to be generated by the client logic of the Target node. As shown in Table A-5, the address for returning header can be generated by reverse the value in the Trgt field and hop fields. The Hop Count Field (HCNT) enables a bridging node to be addressed as a node. The source node will identify the number of bridge hops in the header. Each Head Stripper decrements the HCNT field. If the HCNT is Fh when received at a node that also functions as a bridge, then the stream is not passed on to the bridged ring. If there is a value 0h or greater (not Fh) in the HCNT, then the stream is bridged to the next ring. The HCNT field is not shifted when the other hop fields are shifted.

Table A-3 TxPort Head Format

TxT[2:0]	TxS[31:0]								
2:0	31:30	29:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0
Type	**ACC**	**Conn**	**Srce**	**Hop**					
0	0	Conn	XXXX	Trgt	Hop1	Hop2	Hop3	Hop4	HCNT
0	1	0	XXXX	Trgt	Hop1	Hop2	Hop3	Hop4	HCNT
1	XX	0	XXXX	Group Field		Multicast Field			

Table A-4 Shift of Hop field when Directed head crosses Bridge Hop

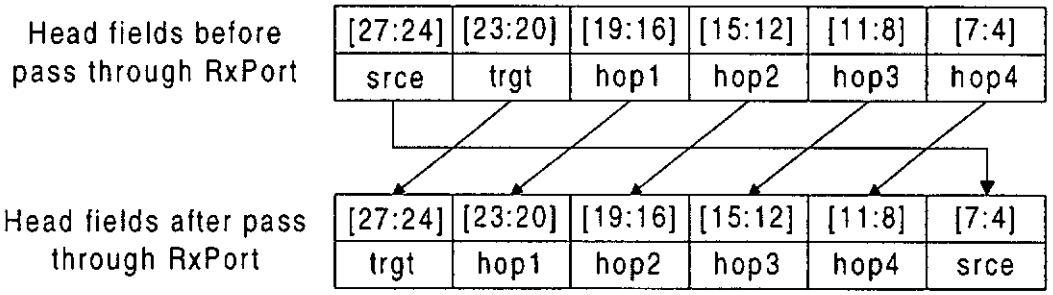
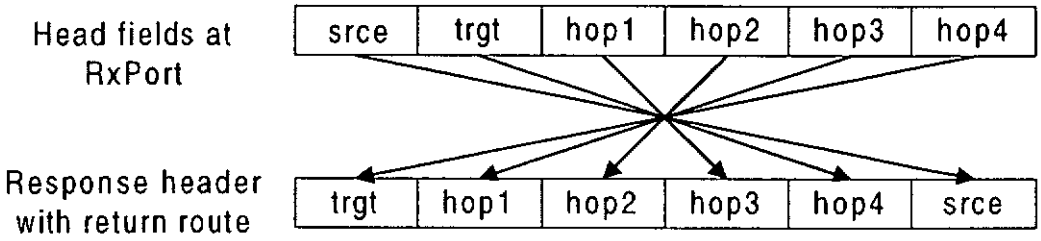


Table A-5 Client Logic Transformation Head fields to Return Route



Multicast transmission

The Multicast transmission is used to send non-reservation based streams to a subset of all nodes on the system. If all nodes are in the subset, then the multicast is broadcast. Non-reservation means the buffer in destination are not reserved to received the stream. This makes transfer unreliable.

As depicted in Table A-3, multicast uses the stream header to address the delivery pattern. There is a multicast field has 16 bits, one for each node

in the local ring. If the bit corresponding to the node address is set and there is Target FIFO space available, then that node will absorb the multicast packet, clear the bit and pass it downstream. If there is no FIFO space, the bit is cleared and packer forwarded only. If the bit is not set, the node only passes the multicast packet downstream. If all bits are cleared in the Multicast Field, them the stream is eliminated from the ring.

The Group field is available to designate multicast targets in rings beyond bridge hops. Intelligent bridges must identify the multicast field patterns in ring hops. These group field maps must set up during configuration, prior to transmitting the multicast. The group field is passed through the target RxPort exactly as it was entered at the source TxPort. The bridge has the intelligence to look up the group in a table and set the appropriate multicast bits for the next ring. If another bridge hop is in the route, the group field must be correct for the next intelligent bridge.

Multicast packets are always fixed in length and never Low bandwidth on the ring. The length is set by one of 3 occurrences at the Tx Port.

1. The loading of a tail, either frame or data.
2. The loading of another head of any type.
3. The loading of the twentieth payload symbol of any type. This occurrence also attaches the previous head to the next multicast packet.

Fixed Stream Size

Fixed client stream size allows the client to control the contiguous packet size onto the ring. This benefits the client design because the client can guarantee delivery of contiguous fixed size streams at the Target RxPort. The fixed stream is limited to the maximum ring packet size of 20 payload symbols. Any shorter size contiguous packets can be used such as 16 symbols for 64-byte payloads. The fixed stream size is accomplished by setting the appropriate value in the access field of the stream header. Table A-3 shows this access field value in the second row for the fixed stream size. The field can be one of 4 values as represented in the following Table A-6.

If a header with a fixed stream ACC field is written to the TxPort, it signals the controller to wait for a tail symbol to send the packet onto the ring. If a head should arrive before a tail, the previous payload is automatically changed to a tail. The next tail symbol or head symbol will indicate the packet is ready to launch. In Directed transmission, if the ticket arrives before the tail symbol is written to the transmit port, then the tail symbol is the trigger to launch. This gives the client complete control over packet continuity on the ring. The packets then travel to the destination and appear at the receive port as one continuous stream (no interleaved streams or nulls) ending with the tail symbol.

Tail Symbols

Tail symbol is a feature that gives a lot of flexibility to the device. The tail symbol is useful for the fixed length, directed stream and multicast operation. It is optional for variable directed stream operations.

Table A-6 Directed Head Access Field Description

ACC[1,0]	Name	Description
0	Variable Directed	The stream marked by this head shall follow a specified path to a single target using the voucher-ticket protocol within each ring and may modify packetization. Streams may arrive at target RxPort non-contiguously.
1	Fixed Directed	The stream marked by this head shall follow a specified path to a single target using the voucher-ticket protocol within each ring and shall not modify packetization. This access fields dictates that a Tail symbol, either data or frame, be used as the final payload symbol. Streams less that 20 payload symbols will be contiguous throughout the ring.
2	Reserved	Future use
3	Reserved	Future use

A.2.2 Client Receive Port

The receive port uses the Target FIFO for the Directed, Multicast and Low Bandwidth Streams. The function are accommodate the Multicast streams and the Fixed Directed Streams. A Multicast Handler is added to the Upstream Router to accommodate the special functions.

The Upstream Router, LB Target Handler, Directed Target Handler and Ring/Multicast FIFO are part of the forwarding path. The LB Target Handler processes LB vouchers targeted to this node into tickets. These tickets are forwarded to the source node through the downstream port.

The Target Handler processes vouchers targeted to this node into tickets. These tickets are forwarded to the source node through the downstream port.

The Ring/Multicast FIFO stores incoming data from the upstream port that is intended to be forwarded to other nodes on the ring, when the downstream pipe is allocated to launching local data generated by this node.

Upstream Router

The Multicast stream handling first. The multicast stream does not use the

voucher-ticket protocol on the ring. For this reason, multicast ring packets will arrive at a target without a reservation.

At the Multicast handler, the multicast head is checked for the Multicast Field bit to be set that corresponds to that nodes ring address. For example, Bit 1 corresponds to Node 1. If the node bit in the multicast field is set, the packet is routed to the Head Stripper and to the Ring/Multicast Bypass FIFO. The multicast field bit is cleared and if there are still any multicast bits set, the packet is immediately forwarded onto the next node. If there are no more bits set, then the packet is eliminated from the ring. If the node address bit is not set, the multicast packet is routed to the Bypass FIFO and immediately forwarded. This allows for the minimum delay through each node and the almost simultaneous multicast to each destination. If the packet should return to the source node, and the bit is set for that node is set, the packet is routed to head stripper. The packet is not forwarded under any circumstances from the source node.

Head Stripper

The Head Stripper has the function of shifting hop fields, refer to Table A-4. It only shifts the 4 Hop fields in Directed heads. The HCNT is never shifted. Instead, the HCNT is decremented as it passed the Head Stripper. No shifting occurs for the heads of multicast or fixed-length packets. In directed transmission, The Head Stripper removes all heads except those identifying the beginning of a stream.

Receive Pipeline

The Target FIFO reserves space for 3 directed packets and 6 LB packets. Therefore, a maximum of 3 tickets can be generated when vouchers are targeted at this node. The Rx Resynchronizer is an 8-deep FIFO in the path between Target FIFO and the Rx Port. The Rx Resynchronizer handles the discontinuity between the client interface and the ring logic of the controller.

If a ticket is able to be issued by the Directed Target Handler, the ticket counter is decremented and the packet is copied into the Target FIFO. If all tickets are outstanding. The packet is not copied locally. This may result in some multicast packets being lost due to RxPorts not unloading the Target FIFO in time to accept the next multicast packets. When the multicast packet in the Target FIFO is unloaded through the RxPort by the client, then the ticket counter is incremented and another multicast packet can be copied.

Fixed Packets

The fixed length streams will have a header at the beginning of each stream that emerges from the RxPort. The head stripper that usually eliminates redundant heads used in the ring packetization, has no function in the fixed length packets. It is then deactivated when Fixed length packets are traversing the receive pipeline. It is only variable, directed, redundant heads that are stripped.

Multicast Packets

Multicast Packets are fixed in length. Just as with the Fixed packets, they will always appear at the RxPort beginning with a head symbol and ending with a tail symbol.

A.3 Ring Interface

The ring is formed by connecting Up and Dn ports of adjacent QuickRing controllers, carries one 42-bit symbol every 25ns. The 42-bit symbol is composed of : 32 bits of data, 3 type bits and 7 bits of EDC. To transmit 42 bits in 25 ns, QuickRing divides the 42-bit symbol into 7 sub-symbols, each sub-symbol is 6 bits wide. The controller then multiplexes the sub-symbols onto the 6 LVDS pairs on the downstream port. A 7^{th} LVDS clock signal, at 40 MHz, accompanies every 42-bit symbol transmission.

Type and Symbol Field at the Ring Ports

On the ring path, upstream and downstream ports, type and symbol fields organise data transmissions. Data on the ring flows in bounded streams called packets. Before data flows in the ring, packets are formed by each controller internally. Packets have one head and one or more payload symbols. Since multiple independent packets can be found inside one controller and multiplexed at the downstream port, a type field accompanies each symbol. Inside a controller, packets can be found that may originate from any other node on the system. The type field marks each symbol as a head, payload, tail, or access.

Symbol Flux on Ring

At the transmit and receive ports, the length of a data stream that is uninterrupted by a head is unbounded. On the ring upstream and downstream ports, data is bounded; there is an upper bound that gives the concept of a packet. There are two types of packets: normal and low bandwidth (LB). There is a ring protocol defining the symbol sequence. For normal packets, the maximum number of payload symbols associated with one head is fixed at 20 symbols. The largest packet is 21 symbols in all; however , packets may be less that 21 symbols. The

LB packet consists of a Head and one payload/Tail.

Data on the Ring (Head, Payload, Tail)

QuickRing transports streams of payload symbols from source nodes to target nodes through the ring interconnect. QuickRing internally assembles packets from the data that the client writes into the transmit port. This data is eventually transmitted in packets of 1 to 20 payload symbols. A head symbol precedes the packet and the last payload symbol of a packet is specially marked as the tail of that packet. The head holds the source and the destination node Ids, plus other information that uniquely identifies the stream to which the payload symbols belong.

Payload symbols consists of 32 bits of user defined information plus one 33rd user controllable Frame identifier. A payload symbol whose frame bit is set to 1 may be called a Frame symbol. Otherwise it may be referred to as a Data symbol. The Frame identifier, the logical 33rd user defined bit is mapped in SS[3] in sub-symbol 1.

Access on the Ring (Voucher, Ticket, Abort, Null)

In directed data transfer, before a source node can send a packet, permission to transmit must first be granted by the target node. To get permission to transmit, the source node sends a voucher to that target node. To grant permission to transmit, the target node sends a ticket back to the source node. This is done only in response to a voucher. When the source node. This is done only in response to a voucher. When the source node sends a voucher, the target node may (1) absorb the voucher and return a ticket or (2) return the voucher. If the source receives the ticket, then it may send one packet to the target that returned the ticket. If the source received its own returned voucher, then it will sink it and retransmit the voucher after 100 clocks for a new request to transmit. The number of retries for the voucher is unlimited until the target returns a ticket. Under normal circumstances the target should return a ticket in response to a voucher, even if it must save accumulated vouchers in a queue and issue corresponding tickets with significant delays. The return of a voucher to its source should occur only if resources for queuing vouchers in the target node are exhausted. An abort symbol identifies the occurrence of a failure such as (1) an illegal symbol sequence, (2) a corrupted symbol or (3) a node ID for which node is present. The node that creates the abort symbol deletes it once the abort symbol has circulated the ring.

For every tick of the ring clock, every node in the ring receives on symbol at its upstream port and transmits one symbol at its downstream port. In the absence of any other symbol, a null symbol is transmitted.

Mapping of Type, Frame, Data and EDC Code on the Ring

On the ring path, a 42-bit symbol is transferred on every tick of the 40 MHz clock. The 42-bit symbol is divided into 7 sub-symbols, 6 bits wide. Each sub-symbol is transferred sequentially at a rate of 1 sub-symbol every 3.6ns. In all, 42 bits are transferred every 25ns.

As shown in Table 3-4, the type field, frame bit and the 3 MSB of the data are mapped to sub-symbol 1. Data[28 to 0] are mapped onto sub-symbols 2 to 6. The EDC field, CB[6:0], is mapped onto the last two sub-symbols of 6 and 7.

Ring Interface Field Definitions

QuickRing organizes data with a combination of access symbols and packet symbols. Access symbols are vouchers, tickets, nulls, and aborts. Packet symbols are those that form packets such as Multicast or Directed heads and payloads.

These symbols can also be grouped into routing symbols and payloads symbols. Routing symbols include access symbols and heads of packet symbols for Directed and Multicast transfer. Access symbols and Directed heads hold source and target address. Multicast head hold set of destinations node's ID.

Payload symbols are data or frame; they hold the information the client are trying to transmit. At the client ports, a type field [01X]represents a data symbol, a type field of [10X] a frame symbol. A data or frame symbol are the ring ports are distinguished by an additional frame bit. The payload symbol, frame or data, at the end of a packet is called a tail and it is encoded as such in the type field.

Access Symbols on the Ring

There are four kinds of access symbols on the ring: vouchers, tickets, nulls, and aborts. These are identified in Table A-7 through Table A-10.

Voucher: Vouchers are sent by source nodes to obtain permission to launch packets of client generated payload symbols. QuickRing will send a voucher as soon as there is a head and a data are loaded at the transmit port. Table A-7 shows the format for a voucher. A Type field value of 3 indicating an access symbol and the ACCess field value of 1 indicates that the symbol is a voucher.

Table A-7 Ring Port Voucher Field Format

Type	F	31:30	29:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0
3	0	[0,1]	Conn	Srce	Trgt	Hop1	Hop2	Hop3	Hop4	HCNT

Ticket: Tickets are returned by target nodes in response to vouchers. The indicate that the target has reserved FIFO space for one packet. QuickRing will send one packet upon receipt of one ticket. The format for a ticket is shown in Table A-8. As in all access symbols, the type field has a value of 3. In tickets, a value of 2 in the ACCess indicates that the symbol is in fact a ticket.

Table A-8 Ring Port Ticket Field Format

Type	F	31:30	29:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0
3	0	[1,0]	Conn	Srce	Trgt	Hop1	Hop2	Hop3	Hop4	HCNT

Null: Null symbols are sent to consume idle time on the ring. The symbol format is shown in Table A-9. When ever a controller does not have any payload or other access symbol to send, it will send a null symbol at its downstream port.

Table A-9 Ring Port Null Field Format

T(1:0)	F	31:30	29:0
3	1	3	XXX

Abort: An abort symbol is sent by any node that detects a failure of addressing, protocol, or data integrity during normal operation the QuickRing ring. The symbol format is shown in Table A-10.

Table A-10 Ring Port Null Field Format

T(1:0)	F	31:30	29:0
3	0	0	XXX

A.4 Operation Flow

Ring Traffic Flow Priorities for DnSS Port Transmission

When all nodes on the QuickRing are ready to transfer packets on the ring. Data streams can be queued/de-queued into/from the controller through the Client Interface on all nodes. As traffic builds on the ring, the controller prioritizes the information flow through the node that goes onto the ring. Following is the list, in descending order, of the paths inside the controller that process information to be sent out onto the ring through the downstream (DnSS) port (Figure A-1).

1. The highest priority is given to tickets/vouchers to/from other nodes This controller is simply forwarding the access symbols on the ring that are destined for other nodes.

2. LB Target Handler: Sends out low bandwidth tickets generated by this node. (Includes voucher rejects when exceeding storage capacity.)

3. Directed Target Handler: Sends out normal tickets generated by this node. (Includes voucher rejects when exceeding storage capacity.)

4. Local sourced vouchers launched by this node.

LB FIFO: Generates a voucher when the LBW symbol gets to the head of the FIFO.

X or Y FIFO: Generates a voucher as soon as the first symbol of packet directed transmission is loaded into the FIFO. Also, generates another voucher as soon as the 21st symbol (associated with the same head) is loaded into the FIFO. Further, continues to generate a voucher for each packet for directed transmission(maximum bundle of 20 symbols).

5. Ring/Multicast FIFO: is forwarding data destined to other nodes.

6. Sends out locally generated packets from this node's X, Y, or LB FIFO. At the beginning of each packet, the traffic flow priorities are checked. (The source (X, Y, or LB) FIFO can transmit when the Ring/Multicast FIFO has at least 28 empty positions. Locally sourced packets will be held until the Ring FIFO has no more than 12 data symbols.)

Inside the Source Node

For directed transmission, as soon as the source controller latches a head and

a payload symbol, in the X or Y FIFO, it sends a voucher to the target node. The source node waits until the target node sends a ticket back before transmitting a packet. During this time the client interface can write payloads into the controller. For multicast transmission, packet will be transmitted to ring as soon as the ring network is available to transfer a complete packet.

When the controller detects that a single packet will not be enough to transmit all data in FIFO X or Y, another voucher is sent to the target node. Additional vouchers are sent as soon as the controller deems it necessary to complete the transmission. This action is intended to hide the latency between the transmission of vouchers and the receipt of tickets. A maximum of 7 (3X, 3Y and 1 LB) vouchers can be outstanding from the source node. Vouchers have higher priority than payloads, and they can be launched interleaved in current outgoing. At the source node, only when a ticket is received is a packet sent. A packet is formed by 1 head and anywhere from 1 to 20 payload symbols. The largest packet is 21 symbols, 1 head symbol and 20 payload symbols. The last payload symbol of a packet is always identified as a tail. This is encoded in the type field.

Inside the Target Node

At the target node, if the receiving controller has space for one packet in the Target FIFO, it will send a ticket immediately to the source node in response to a voucher. Target FIFO has space for 3 normal packets and 6 low bandwidth packets; therefore, the controller can have only 3 outstanding normal tickets and 6 outstanding low bandwidth tickets. When a multicast packet targeted at this node arrived, the number of available ticket is reduced by 1. If all tickets have been given, the receiving controller will queue incoming vouchers in one of two special buffers, called Target Handler and LB Target Handler and the incoming multicast packet will be forwarded to next node directly. The Target Handler can store 30 vouchers and the LB Target Handler can store 10 vouchers for low bandwidth transmission. At the target node, a new ticket is released as soon as a packet has exited the Target FIFO to the Rx Resynchronizer. This is determined internally by matching the tickets given and the tails exiting the target FIFO.

If the target node cannot return a ticket or store the voucher to handled later, it will return a voucher rejected to the source node. (The source will sink the voucher and the node will the re-send the voucher after 100 clock cycles.)

B. Summary of the QuickRing Software Simulator

Software simulator of QuickRing has been built to evaluate the performance of QuickRing for different network sizes and data sizes. The simulation is based on the networks configuration depicted in Figure 3-9. We evaluate the performance of the QuickRing network by changing the number of High level PCs (N), destination node position and size of data to be transferred in the network.

The software simulator is written in C++. Operations performed by components of controller are simulated step by step. Definition of network configuration and the data to be transmitted are stored in data files. The simulation program will read the configuration files first, then transfers the data. The simulation program counts the clock cycles required for transferring the data and the number of clock cycles is recorded in a result file.

Configuration Files

Several configuration files are required to hold different aspects of network configuration including node number of controller that form the ring network, the node number of controller that is used to form the bridge, and the node number of controller that connect different Ring network together. They are Bridge.ini, Initsend.ini, Ring.ini, Testnode.ini

Ring.ini: The configuration file "Ring.ini" stores the number of rings that constitutes the network and the number of nodes in each ring. As shown in Figure B-1, the first number in each line represents the ring number and the next number represents the total number of nodes in that ring. The example depicted in Figure B-1 represents the network consists of two rings. The first ring consists of 3 nodes and the second consists of 16 nodes.

```
1 3
2 16
```

Figure B-1 Rings Definition File

Bridge.ini defines the nodes in different rings that are connected to form a bridge. As depicted in Figure B-2, the first two numbers in each line represent the ring number and the node number in the first ring. The next two numbers represent the ring number and the node number in the second ring. The sample shown in Figure B-2 illustrates that node 2 of ring 1 is connected to node 0 of ring 2 forming a bridge.

```
1 2 2 0
```

Figure B-2 Bridges Definition File

Testnode.ini stores all other nodes in the network that are not defined in Bridge.ini. As shown in Figure B-3, the first number represents the ring number and the second value represents the node number.

```
1 0
1 1
2 1
2 2
.
.
.
2 15
```

Figure B-3 Node Definition File

Initsend.ini defines the source node, bridge node and the destination node of transmission action. As depicted in Figure B-4, the first number represents the ring number of the source node and the second number represents the node number of source node. The third and fourth number tells us the node 2 of ring 1 is the bridge node. The fifth number indicates the transmission mode: 0 stands for normal, 1 represents low bandwidth transmission. The following 5 digits-hexadecimal number represent the hop field and hop count field in the header of packet. The string "Testx1.doc" is the file name of the data file that contains the data to be transmitted.

```
1 1 1 2 0 0X80000LTestx1.doc
```

Figure B-4 Route Definition File

The process of the simulator can be divided into 3 phases: initialisation, iteration, and result recording. In the initialisation phase, configuration file is read in order to configure the network being simulated. In the iteration phase, data are read from data file word by word and transmitted to destinations. The simulation program starts counting the number of clock cycles when the data reach transmit port of QuickRing controller. The program will stop counting when all data in the data file is received by destination node. The operations in QuickRing controller are simulated based on information provided in the QuickRing Handbook[23]. When the last data in the data file is received by the destination, the number of clock cycle stored in the counter will be written to the result files. Since the QuickRing chip is operating at 40 MHz, time (sec) consumed for data transfer can be found by multiply the number of clock cycle by 1/40,000,000.

The number of clock cycles spent in data transfer is stored in the output file. As shown in Figure B-5, the number in first column represents the starting clock

cycles that the data reach the Tx port, the number in the fourth column represents the end of data that is received by the destination node. The results can be imported to other analysis software such as Excel for further analysis.

```
2       --- 11537    --- 11546    --- 12004          ---->Mon Apr 21 01:00:09 1997
```

Figure B-5 Output File Example

REFERENCES

[1] G. Funka-Lea, R. Bajcsy, "Vision for Vehicle Guidance Using Two Road Cues", *Proceedings of the Intelligent Vehicles Symposium 1992*, pages 126-131, 1992

[2] C. C. Weems, "Architectural Requirements of Image Understanding with Respect to Parallel Processing", *Proceedings of the IEEE*, Vol. 79, No. 4, April 1991.

[3] Fountain T. J., Mathews K. N. and Duff M. J. B., "The CLIP7A Image Processor", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 10, page 310-319, 1988.

[4] Batcher K. E., "Design of a Massively Parallel Processor", *IEEE Transactions on Computers*, Vol. 9, pages 836-840, 1980.

[5] Hunt D. J., "The ICL DAP and Its Application to Image Processing", in *Languages and Architectures for Image Processing,* edited by Duff M. J. B. and Levialdi S., Academic Press, New York, pages 275-282, 1981.

[6] C. C. Weems, D. Rana, A. R. Hanson, E. M. Riseman, D. B. Shu, J. G. Nash, An Overview of Architecture Research for Image Understanding at the University of Massachusetts, CH2898-5/90/0000/0379$01.00 IEEE 1990.

[7] G. R. Nudd et al, WPM: A Multiple-SIMD Architecture for Image Processing, *Proc IEEE Int. Conf. On Computer Vision and Pattern Recognition,* pages 92-97, San Diego, June 1989.

[8] Duff M. J. B., "CLIP4: A Large Scale Integrated Circuit Array Parallel Processor", *Proceedings of 3rd International Joint Conference on Pattern Recognition*, pages 728-733, 1976.

[9] T. Collette, H. Essafi, D. Juvin, J. Kaiser, "Low and Intermediate Level Image Processing on SYNPATIX, a SIMD Parallel Computer", 0-8186-2925-8/92 IEEE 1992.

[10] N. D. Francis, G. R. Nudd, T. J. Atherton, D. J. Kerbyson, R. A. Packwood, J. Vaudin, Performance Evaluation of the Hierarchical Hough Transform on an asociative M-SIMD architecture, CH2898-5/90/0000/0509 IEEE, 1990

[11] D. J. Kerbyson, T.J. Atherton, G. R. Nudd, "An MSIMD Architecture for Feature Tracking", *IEE Colloquium on 'Medium Grain Distributed Computing' (Digest No. 070)*, pages 7/1-6, 1992.

[12] Eshaghian M.M., Nash J.G., Shaaban M.E., Shu D.B., "Heterogeneous Algorithms for Image Understanding Architecture", *Proceedings 1993 Computer Architectures for Machine Perception*, pages 286-292, 1993.

[13] Fountain T. J., "The Use of Linear Arrays for Image Processing", *Proceedings of International Conference on Systolic Arrays*, San Diego, USA, pages 183-192, 1988.

[14] P.P. Jonker, "Why linear arrays are better image processors", *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, Vol. 3, pages 334-338, IEEE, 1994.

[15] Fung Y.F., Fountain T. J., "The Design of a Linear Array Image Processor", *Proceeding of IEEE Tencon '92*, Melbourne, Australia, pages 998-1002, 1992.

[16] Ercan M.F., Fung Y.F., "Low Level Image Processing on a Linear Array Pyramid Architecture", *IEEE Computer Society, Technical Committee on Computer Architecture (TCCA) Newsletters*, pages 9-15, June 1996.

[17] May D., Shepherd R. And Thompsion P., "The T9000 Transputer", in *Proceedings of IEEE International Conference on Computer Design: VLSI in Computer and Processors*, pages 209-212.

[18] Maxfield C., "The Pentium II revealed", *Byte*, Vol. 22, Issue 9, pages 51-52,

McGraw-Hill, Sept 1997.

[19] C.C. Feng, S.N. Yang, "A Parallel Hierarchical Radiosity Algorithm for Complex Scenes", *Proceedings IEEE Symposium on Parallel Rendering (PRS '97)*, pages 71-77, 1997.

[20] Y.F. Fung, T.H. Heung, M.F. Ercan, "Distributed image processing tool for a WindowsNT network", *Proceedings of the SPIE- The International Society for Optical Engineering*, Vol. 3166, pages 300-307, 1997.

[21] H. Hellwagner, W. Karl, M. Leberecht, "Enabling a PC cluster for high-performance computing", *SPEEDUP*, Vol. 11, Issue 1, pages 18-23, June 1997.

[22] M. Oguchi, T. Shintani, T. Tamura, M. Kitsuregawa, "Characteristics of a parallel data mining application implemented on an ATM-connected PC cluster", *High-Performance Computing and Networking. International Conference and Exhibition. Proceedings*, pages 303-317, 1997.

[23] *The National QuickRing Design Handbook*, National Semiconductor, June 1994.

[24] Nudd G. R., Kerbyson D. J., Atherton T. J., Francis N., Packwood R. A., and Vaudin G. J., "A Massively Parallel Heterogeneous VLSI Architecture for MSIMD Processing"; in *Algorithms and Parallel VLSI Architectures*, edited by Deprettere F. and Van Der Veen A., Elsevier, Amsterdam, pages 463-472, 1991.

[25] Hunt D. J., "The AMT DAP- A Processor Array in a Workstation Environment", *Computer Systems Science and Engineering*, Vol. 2, pages 107-114, 1989.

[26] D.B. Shu and J.G. Nash, A Multiple-Level Heterogeneous Architecture for Image Understanding, IEEE *International Conference on Application Specific Array Processors*, pages 615-627, 1990.

[27] Anyanwu C. D. and Jalowiecki I. P., "ASTRA: A Multilayer Parallel Processing System", *Proceedings of 20th EUROMICRO Conference on System Architecture and*

Integration, LiverPool, pages 565-572, 1994.

[28] Lea R. M., "The ASP: A Cost-Effective Parallel Microcomputer", *IEEE Micro*, Vol.8, pages 10-29, October 1988.

[29] Jaloweicki I., "WASP the Associative String Processor". in *Parallel Computing Principles and Practice*, edited by Fountain T. J., Cambridge University Press, Cambridge, pages 296-308, 1994.

[30] M. Pietikäinen, T. Seppänen P. Alapuranen, "A Hybrid Computer Architecture for Machine Vision", *CH2898-5/90/0000/0426* IEEE, 1990.

[31] Hancock, Bill, *Advanced Ethernet/802.3 Network Management and Performance*, Boston, Digital Press, 1996.

[32] L. G. Kazovsky, G. F. Barry, M. J. Hickey, C. A. Noronha, P. T. Poggiolini, "A Multi-Gbit/s Optical LAN Utilizing a Passive WDM Star: Towards and Experimental Prototype", 0743-166X/93 $03.00 IEEE, 1993.

[33] Dell'Acqua, Alexa A., *IBM's Token-Ring network*, Charleston, S.C. : Computer Technology Research Corp., 1992.

[34] K.M. Lye, T. C. Wong, K. C. Chua, "Two-connected Multiple-Token Multiple-Ring Network", *IEE Proc.-Commun.*, Vol. 142, No. 6, December 1995.

[35] Johnson H.W., *Fast Ethernet: dawn of a new network*, Prentice Hall, 1996

[36] H.J. Wolfson, I. Rigoutsos, "Geometric Hashing: An Overview", *IEEE Computational Science & Engineering*, Vol. 4, Issue 4, pages 10-21, Oct-Dec 1997.

[37] V. O. K. Li, J. F. Chang, K. C. Lee, T. S. Yang, "A Survey of Research and Standards in High-Speed Networks", *International Journal of Digital and Analogue Communication Systems*, Vol. 4, pages 269-309, 1991

[38] R. Händel, M N. Huber, S Schröder, *ATM Networks*, Addison-Wesley, 1994.

[39] G. D. Hager, D. Kriegman, E. Teh, C. Rasmussen, "Image-based prediction of landmark features for mobile robot navigation", *Proceedings of International Conference on Robotics and Automation*, Vol. 2, pages 1040-1046, IEEE,1996.

[40] T.H. Tseng, J.N. Hwang, F.H. Sheehan, "3-d heart contour delineation and motion tracking of ultrasound images using continuos distance transform neural networks", *Neural Networks for Signal Processing VI. Proceedings of 1996 IEEE Signal Processing Society Workshop*, pages 361-370, IEEE, 1996.

[41] *TMS320C4x User's Guide*, Texas Instruments, 1993.

[42] *TMS320C3x/C4x Assembly Language Tools*, Texas Instruments, 1997.

[43] *TMS320C3x/C4x Optimizing C Compiler*, Texas Instruments, 1997.

[44] *TMS320C40-60 Datasheet (online document)*, Texas Instruments, 1996.

[45] E. R. DAVIES, *Machine Vision: Theory, Algorithms, Practicalities*, Academic Press, 1990.