Graphics and video communications over an Integrated Services network

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:


Metadata Record: [https://dspace.lboro.ac.uk/2134/28220](https://dspace.lboro.ac.uk/2134/28220)

Publisher: © Guojun Lu

Rights: This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 2.5 Generic (CC BY-NC-ND 2.5) licence. Full details of this licence are available at: [http://creativecommons.org/licenses/by-nc-nd/2.5/](http://creativecommons.org/licenses/by-nc-nd/2.5/)

Please cite the published version.
This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

![Creative Commons License](https://creativecommons.org/licenses/by-nc-nd/2.5/)

**Attribution-NonCommercial-NoDerivs 2.5**

You are free:

- to copy, distribute, display, and perform the work

**Under the following conditions:**

- **Attribution.** You must attribute the work in the manner specified by the author or licensor.
- **Noncommercial.** You may not use this work for commercial purposes.
- **No Derivative Works.** You may not alter, transform, or build upon this work.

- For any reuse or redistribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

This is a human-readable summary of the Legal Code (the full license).

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/
<table>
<thead>
<tr>
<th>VOL. NO.</th>
<th>CLASS MARK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>008095202</td>
</tr>
</tbody>
</table>

**LOUGHBOROUGH UNIVERSITY OF TECHNOLOGY LIBRARY**

**AUTHOR/FILING TITLE**
LU G

**ACCESSION/COPY NO.**

**VOL. NO.**

**CLASS MARK**

- **5 JUL 1991**
- **15 JUL 1991**

**Loan copy**

**Date due:**
- **28 JUN 1996**
- **1 JUL 1994**
- **25 JUN 1999**

**Loan 3 wks. + 2 unless recalled**

**Limit**

- **26 JUN 1996**
- **30 JUN 1995**
Graphics and Video Communications over an Integrated Services Network

by

Guojun Lu, B.Eng.

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy of the University of Technology, Loughborough.

December, 1989
Supervisor: Professor J. W. R. Griffiths

Department of Electronic and Electrical Engineering
University of Technology, Loughborough
England

© by Guojun Lu, 1989
To Fengxia

and my parents
ABSTRACT

There is currently much interest in Integrated Services Systems which handle within one network or architecture a mixture of data, graphics, video and voice, although most of the work to date has concentrated on the voice and data services. This thesis is concerned with the provision of graphics and video services alongside the voice and data services over a Wide Area Network (WAN). The network concerned in this thesis is the Unison network, which comprises several Local Area Networks (LANs) interconnected by a primary rate (2 Mb/s) Integrated Service Digital Link.

The first part of the thesis deals with the design and development of an Electronic Blackboard system suitable for use over the network. The system comprises workstations sitting on the network, with a mouse or a graph pad as the control and input device to each workstation. Inputs from a user are displayed on his own workstation monitor and also displayed on the workstation monitor of his partner elsewhere on the network. The system provides the participants, say of a videoconference, with a tool to facilitate exchange of text (entered from keyboards), graphics and free-hand drawings (entered from the mouse or the graph pad).

The second part of the thesis is concerned with the provision of a video service. A versatile video codec (including a transmitter and a receiver) has been designed and implemented. The video codec is based on Inmos Transputers which have the characteristics of fast speed and easy interconnection. A set of video protocols has also been developed to provide multi-point video service over the Unison network using the video codecs, enabling a transmitter to send images to more than one receiver and a receiver to receive and display images on one monitor from more than one transmitter. These protocols are implemented in a concurrent language called OCCAM. Colour and monochrome image transfer is selectable. Two main services, freeze image transmission and slow scan image transmission, are available. To reduce the amount of bandwidth required to transmit images, two simple image compression schemes, namely subsampling and conditional replenishment, have been implemented in software. The operation control of the video codecs is by means of a user interface on workstations connected to the network.

Experiments on graphics and image transmission along with voice and computer data have been carried out successfully over the Unison network.
Acknowledgements

I would like to thank my supervisor Professor J. W. R. Griffiths for his guidance and encouragement throughout the research and for the financial support in the last half year.

I would also like to acknowledge the financial support from the Chinese Government and the British Council.

Special thanks extend to Mr. C. Rodgers who developed part of the codec hardware and to everyone in the Unison Project who contributed the facilities and useful discussion.

I would also like to thank Dr. H. S. Chin, Mr. C. Rodgers, Mr. B. Murphy, Mrs. C. Sun and Dr. S. Jayasinghe for reading and commenting on the final draft of this thesis and to Mrs. S. Clarson, Dr. D. J. Parish and all my friends in the Signal Processing Laboratory of Loughborough University for their help and friendship.

Finally I am greatly indebted to my wife Fengxia and my parents for their love and encouragement.
List of Symbols and Acronyms

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>Analogue to Digital Converter</td>
</tr>
<tr>
<td>ATM</td>
<td>Asynchronous Transfer Mode</td>
</tr>
<tr>
<td>bps</td>
<td>bits per second</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer Aided Design</td>
</tr>
<tr>
<td>CAS</td>
<td>Column Access Strobe</td>
</tr>
<tr>
<td>CCITT</td>
<td>Comité Consultatif International de Télégraphique et Téléphonique</td>
</tr>
<tr>
<td>CFR</td>
<td>Cambridge Fast Ring</td>
</tr>
<tr>
<td>CR</td>
<td>Cambridge Ring</td>
</tr>
<tr>
<td>CRT</td>
<td>Cathode Ray Tube</td>
</tr>
<tr>
<td>DAC</td>
<td>Digital to Analogue Converter</td>
</tr>
<tr>
<td>DCT</td>
<td>Discrete Cosine Transform</td>
</tr>
<tr>
<td>DMA</td>
<td>Direct Memory Access</td>
</tr>
<tr>
<td>DPCM</td>
<td>Differential Pulse Code Modulation</td>
</tr>
<tr>
<td>DRAM</td>
<td>Dynamic Random Access Memory</td>
</tr>
<tr>
<td>EBS</td>
<td>Electronic Blackboard System</td>
</tr>
<tr>
<td>EPROM</td>
<td>Erasable Programmable Read Only Memory</td>
</tr>
<tr>
<td>FIFO</td>
<td>First In First Out</td>
</tr>
<tr>
<td>G</td>
<td>Giga</td>
</tr>
<tr>
<td>ISDN</td>
<td>Integrated Service Digital Network</td>
</tr>
<tr>
<td>k</td>
<td>kilo or one thousand</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>M</td>
<td>Mega or one million</td>
</tr>
<tr>
<td>n</td>
<td>nano</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PCB</td>
<td>Print Circuit Board</td>
</tr>
<tr>
<td>PCM</td>
<td>Pulse Code Modulation</td>
</tr>
<tr>
<td>PPBW</td>
<td>Point-to-Point Bandwidth</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RAS</td>
<td>Row Access Strobe</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote Procedure Call</td>
</tr>
<tr>
<td>RX</td>
<td>Receiver</td>
</tr>
</tbody>
</table>

- III -
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM</td>
<td>Static Random Access Memory</td>
</tr>
<tr>
<td>SSP</td>
<td>Single Shot Protocol</td>
</tr>
<tr>
<td>SSPREQ</td>
<td>Single Shot Protocol Request</td>
</tr>
<tr>
<td>SSPRPLY</td>
<td>Single Shot Protocol Reply</td>
</tr>
<tr>
<td>SysBW</td>
<td>System Bandwidth</td>
</tr>
<tr>
<td>TDS</td>
<td>Transputer Development System</td>
</tr>
<tr>
<td>TX</td>
<td>Transmitter</td>
</tr>
<tr>
<td>µ</td>
<td>Micro or one millionth</td>
</tr>
<tr>
<td>VRAM</td>
<td>(dual-port) Video Random Access Memory</td>
</tr>
<tr>
<td>VSC</td>
<td>Video System Controller</td>
</tr>
<tr>
<td>WAN</td>
<td>Wide Area Network</td>
</tr>
<tr>
<td>WIMP</td>
<td>Windows, Icons, Menus and Pointers</td>
</tr>
<tr>
<td>WS</td>
<td>Workstation</td>
</tr>
</tbody>
</table>
Definitions and Terminology

Asynchronous Transfer Mode

is a packet and connection oriented transfer mode using time division multiplexing technique where data flow is organized in fixed size transport units called cells.

Cell

is the data transport unit in ATM technique. It consists of a header field which contains routing and control information, and an information field for carrying user data, both fields being of fixed sizes.

Call blocking

refers to the situation when a new call request is rejected because of insufficient bandwidth on the network to support the new call, without degrading the service performance of the existing calls.

Client

is a network station which requires services from other stations on the network.

Light-weight virtual circuit (association)

is a logical connection between stations on a packet switched network which preserves the order of packets transmitted, but makes no guarantees about the delivery of these packets.

Minipacket

is the unit of transmission between stations controlled by the ring access mechanism.

Nameserver

is a server on the CR-based Unison network. When presented with the text name of a service, a process or a computer, anywhere on the
network, the name server returns the appropriate network address. It is also capable of performing the converse translation.

Packet Video

is the concept where digital video signals are segmented into blocks of data for transmission over packet switched network.

Portal

is an interface device between client CR or CFR and exchange CFR. It transfers data from one ring to the other and also provides routing mapping.

Ramp

is an interface device between the exchange CFR and the ISDN link. It extracts data from the CFR slots and inserts them into the ISDN slots, and vice versa.

Secretary

is a server on the CFR-based Unison network which provides support for light-weight associations between named Unison services. A client service wishing to exchange data with a peer client uses a simple protocol to communicate with its local Secretary. The Secretary is responsible for locating the peer service and if necessary, for configuring a direct link between the services to be associated.

Server

is a network station which provides service to other stations on the network.

System Bandwidth

is useful data bandwidth of a ring which is independent of the ring size assuming that there are no gap digits between data slots. It is a constant determined by the number of user data bits in a slot, the total number of bits in a slot and the raw bandwidth of the ring.
U-channel

refers to the aggregation of ISDN slots in order to provide variable bandwidth between peer sites for inter-site communications. One ISDN slot is equal to 64 kbits/sec.

Video coding

refers to the application of digital signal processing techniques to reducing the redundancy in video signals to save bandwidth required to transmit images.
# TABLE OF CONTENTS

ABSTRACT I  
ACKNOWLEDGEMENT II  
LIST OF SYMBOLS AND ACRONYMS III  
DEFINITIONS AND TERMINOLOGY V  
TABLE OF CONTENTS VIII  

## Chapter 1 Introduction

1.1 Introduction 1  
1.2 Objectives of the Research 3  
1.3 Overview of the Thesis 3  

## Chapter 2 The Unison Network Architecture

2.1 The Unison Network and Office 5  
2.1.1 The Unison Network 5  
2.1.2 The Unison Office 7  
2.2 The CR-based Network Protocols 9  
2.2.1 Physical Level Protocol 9  
2.2.2 The Basic Block Protocol 11  
2.2.3 The Single-Shot Protocol 12  
2.2.4 The OPEN/OPENACK Protocol 13  
2.2.5 The Network Operation 13  
2.3 The CFR-based Network Architecture 15  
2.3.1 The CFR Protocol 15  
2.3.2 The Unison Secretary Service 17  
  2.3.2.1 S-Association Setup 17  
  2.3.2.2 Association Setup Between Hosts 18  
2.4 Summary 19
### Chapter 3: Investigative Study of Electronic Blackboard System (EBS)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Introduction</td>
<td>20</td>
</tr>
<tr>
<td>3.2 Three Development Stages of Telewriting Systems</td>
<td>21</td>
</tr>
<tr>
<td>3.3 Development Trend</td>
<td>23</td>
</tr>
<tr>
<td>3.4 A Review of Graphics Input Devices</td>
<td>24</td>
</tr>
<tr>
<td>3.5 Design Considerations of the Unison EBS</td>
<td>26</td>
</tr>
<tr>
<td>3.5.1 Design Objectives</td>
<td>26</td>
</tr>
<tr>
<td>3.5.2 Discussion</td>
<td>26</td>
</tr>
<tr>
<td>3.6 Summary</td>
<td>28</td>
</tr>
</tbody>
</table>

### Chapter 4: The Unison Electronic Blackboard System

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 General Description</td>
<td>29</td>
</tr>
<tr>
<td>4.2 User Interface</td>
<td>31</td>
</tr>
<tr>
<td>4.3 Local Graphics Generation</td>
<td>32</td>
</tr>
<tr>
<td>4.4 The Connection Setup</td>
<td>35</td>
</tr>
<tr>
<td>4.5 Graphics Data Transmission and Reception</td>
<td>36</td>
</tr>
<tr>
<td>4.6 Contention Prevention</td>
<td>40</td>
</tr>
<tr>
<td>4.7 The Software Structure</td>
<td>41</td>
</tr>
<tr>
<td>4.8 The CR Framestore as the Blackboard Receiver</td>
<td>45</td>
</tr>
<tr>
<td>4.9 A Note on the ALVEY’87 Exhibition</td>
<td>46</td>
</tr>
</tbody>
</table>

### Chapter 5: Investigative Study on Video Communication

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Introduction</td>
<td>47</td>
</tr>
<tr>
<td>5.2 Video Coding Schemes</td>
<td>48</td>
</tr>
<tr>
<td>5.2.1 Pulse Code Modulation</td>
<td>48</td>
</tr>
<tr>
<td>5.2.2 Differential Pulse Code Modulation</td>
<td>49</td>
</tr>
<tr>
<td>5.2.3 Transform Coding</td>
<td>49</td>
</tr>
<tr>
<td>5.2.4 Vector Quantization</td>
<td>50</td>
</tr>
</tbody>
</table>
5.2.5 Adaptive Coding
5.2.6 Statistics Coding
5.2.7 Practical Coding Schemes
5.3 Error Detection and Recovery
5.4 Video Conferencing and Standardization
5.5 Design Ideas for the CFR Video Codec
  5.5.1 Reasons for Designing a CFR Video Codec
  5.5.2 Design Guidelines for the CFR Video Codec

Chapter 6: The Video Codec

6.1 The Video Coder
  6.1.1 The Analogue To Digital Converter (ADC) Board
    6.1.1.1 The Pixel Clock Generator
    6.1.1.2 The Sync Separator
    6.1.1.3 The ADC
    6.1.1.4 Features of the ADC Board
  6.1.2 The Grabber
    6.1.2.1 The Address Generator
    6.1.2.2 The Frame Memory
  6.1.3 The Controller
    6.1.3.1 The Transputer
    6.1.3.2 The Dynamic RAM
    6.1.3.3 The Grabber Interface
    6.1.3.4 Summary of the Video Transmitter Board
  6.1.4 Summary

6.2 The Video Decoder
  6.2.1 The Video Receiver Board
    6.2.1.1 The Video RAM
    6.2.1.2 The Video System Controller (VSC)
    6.2.1.3 The Clock Generator
    6.2.1.4 The Video Decoder Controller
    6.2.1.5 Features of the Video Receiver Board
  6.2.2 The DAC Board
Chapter 7 Video Protocols

7.1 Introduction

7.2 Connection Setup

7.3 Packetization and Depacketization

7.4 Video Coding Schemes
   7.4.1 Subsampling
   7.4.2 Conditional Replenishment
   7.4.3 Monochrome Image Transfer

7.5 Data Flow Control

7.6 Video Channel Specification

7.7 Video Control Protocol

7.8 Software Structure

7.9 User Interface and Operation

7.10 Summary

Chapter 8 Experiments over the Network

8.1 Introduction

8.2 Video System Configuration

8.3 Experiments over a Local CFR
   8.3.1 Video Transmission without any Compression
   8.3.2 Video Transmission with Subsampling
   8.3.3 Video Transmission with Conditional Replenishment
   8.3.4 Multi-point Communication

8.4 Experiments over Portals
   8.4.1 One Way Video
   8.4.2 Two way Video
   8.4.3 Two way Video with other Traffic

8.5 Experiment over Portals and RAMPs

8.6 Summary
CHAPTER 1

INTRODUCTION

1.1 Introduction

There is currently much interest in the design of so called "integrated services" systems which handle within one network or architecture a mixture of data, graphics, video as well as voice [Ades 87, Gallagher 86, Burren 89]. Most of the work to date has been on either local area networks [Ades 87, Lazar 87] or point-to-point leased lines [Sebestyen 87]. Systems in these categories have limitations because most of the applications of integrated services are in a wide area context. If the services provided on local area networks cannot be extended to wide area networks the usefulness of the services is limited. On the other hand, leased point-to-point services are not flexible, difficult to manage and limited to two partners only. Project UNISON [Clark 86] addresses these issues by designing a general mechanism for interconnecting geographically dispersed local networks for multimedia office applications [Tennenhouse 87].

Project UNISON is a research and development project within the Infrastructure and Communications Programme of the UK Alvey Directorate. The collaborators are the University of Cambridge Computer Laboratory, Loughborough University of Technology (LUT), Logica, Acorn Computers and the Rutherford Appleton Laboratory (RAL). The project has been conducting pre-competitive research into multimedia office applications such as distributed computing, voice, graphics and video.
Chapter 1 Introduction

The objective of the UNISON Project is to build a network and applications which can test the practical aspects of multimedia services. The network architecture and the data and voice services have been the work of others. This thesis concerns the provision of graphics and video services over the Unison network.

Video services, at least slow scan and still image transfer services, are essential for effective communications in a teleconference or similar situation. A full frame rate video requires very wide bandwidth which is beyond the capacity of the Unison network. The slow scan and still image transfer video services are considered in this thesis. Two essential components are required to provide these video services. They are a video codec which is suitable to work on the experimental Unison network and a set of protocols to support reliable and flexible services.

While voice and video services are important in future electronic offices or teleconferencing systems, there is still a lack of a common work space in a teleconferencing system for participants of a teleconference to work in - much like a blackboard in a conventional meeting. Thus it is necessary for a teleconferencing system to provide an interactive medium for the participants to communicate more effectively with each other. The Electronic Blackboard System (EBS) described in this thesis attempts to meet this requirement. The graphics service discussed here is not like the facsimile service which transmits pre-prepared documents. The electronic blackboard system transmits graphics while they are being generated. The system is shared by all the participants who can write on it and receive drawings from others.

The graphics and video services should be integrated into the same network along with voice and computer data. The Unison network is an ATM (Asynchronous Transfer Mode) like network in which data are transmitted in short fixed size packets called cells and each cell has a header with a virtual channel identification [Decina 88, Coudreuse 88, O'Reilly 89]. This kind of network is well suited for multimedia applications. Firstly, there is no channel structure in an ATM network, making it easy for various types of traffic (graphics, video and voice etc.) to share the network bandwidth. Secondly, since multimedia applications usually generate variable bit rate traffic, network bandwidth can be used efficiently. For example, in the case of circuit switched networks, a 64 kbps channel (B-channel) must be assigned for the EBS graphics although on average only several hundred bits per
second are used by the graphics data. Thus most of the bandwidth is wasted. In an ATM network, graphics data shares a large bandwidth with voice, video, and computer data. The bandwidth used by a particular type of traffic is demand-based and is therefore more efficiently used.

In a multimedia electronic office, all facilities should be controlled by one central control device, say a workstation, so that a single common user interface can be used.

1.2 Objectives of the Research

The main objectives of the research work presented in this thesis are:

1) to design and develop an electronic blackboard system for the Unison network;

2) to design and develop a versatile video codec which can make use of the network bandwidth efficiently and can adjust its output data rate according to the loading on the network;

3) to design and develop a set of video protocols to provide reliable and flexible services over the Unison network;

4) to gain some experience in operating the multimedia office.

1.3 Overview of the Thesis

The thesis consists of two main parts, one concerning the provision of graphics service and the other video services. Chapters 3 and 4 are concerned with the work related to the provision of a graphics service over the Unison network, whilst Chapters 5, 6, 7 and 8 deal with the provision of packet video services over the network.

Chapter 2 gives an overview of the Unison network architecture. The Unison network topology and Unison office are described in general. The communication protocols of the Cambridge Ring (CR) based network and Cambridge Fast Ring (CFR) based network are presented. Although this is not the author's work, some knowledge of the network architecture is essential to understand the work to be described in later chapters.
Chapter 3 briefly reviews the development history of electronic blackboard systems, or telewriting systems. It is concluded that an EBS is essential in a future electronic office and is of advantage if it is based on networked workstations. Finally some guidelines are drawn up for the design of the Unison EBS.

Chapter 4 details the design and implementation of the Unison EBS. The EBS is based on Acorn A-500 workstations. The user interface, graphics generation, connection set up and graphics data transmission and reception are discussed. The EBS software is presented in hierarchical layers.

Chapter 5 presents an investigative study on packet video communications. Some commonly used video coding schemes are reviewed first. Then some issues related to packet video, such as error control, coding and transmission standardization, are discussed. Finally, some design ideas for a Unison video codec are introduced.

Chapter 6 details the design and implementation of the Unison video codec (coder and decoder). The codec is based on Transputer technology. The main circuitry of the codec is described. The application of the codec is also discussed. Finally, the interface between the codec and the Unison network is described.

Chapter 7 discusses the video protocols required to provide video services over the network. The protocols include connection set up, video data packetization, data flow and error control and video channel control. The protocols are implemented in OCCAM. The software structure is also briefly described. Finally, the user interface and operation are discussed.

Chapter 8 presents the results of experiments of video transmission along with other traffic over the network. The experiments are carried out over various parts of the network in order to test the applications and the network components.

Chapter 9 summarizes the work described in earlier chapters, proposes further work which could be carried out in this area and concludes the thesis.
CHAPTER 2

The Unison Network Architecture

This chapter gives an overview of the Unison network architecture. The Unison network and office are described in general, followed by a description of the network protocols.

2.1 The Unison Network and Office

2.1.1 The Unison Network

The basic Unison network topology is shown in Fig.2.1. The network comprises a number of Local Area Networks (LANs) interconnected by a primary rate (2 Mbps) ISDN. The pilot ISDN provides each of the sites with a primary rate interface as defined in the CCITT I-series Recommendations. The interface between the client LANs and the ISDN is a Unison Exchange. A Cambridge Fast Ring [Hopper 86] forms the basis of this exchange. A Portal provides the access from the client network to the exchange CFR, while a Ramp provides the primary rate access from the exchange to the ISDN.

If the 64Kbps ISDN B-channels are assigned to individual calls and the separate calls are used by each of the communicating services, no individual service will achieve a throughput greater than 64 Kbps and each service will incur a substantial delay associated with the establishment of a connection through the circuit switched ISDN. In Project Unison all traffic between a pair of sites is aggregated together at the source site and transported along a channel to the destination site. The shared U-channel is multiple B-channel wide allowing the individual service to achieve
Chapter 2 The Unison Network Architecture

Fig. 2.1 The Unison Network Topology
instantaneous bandwidth of \( n \times 64 \) Kbps where \( n \) is the width of the channel in timeslots. An initial interaction between services at peer sites will experience a start of day delay associated with connection establishment. Further interaction between any services at the same sites will not be delayed.

For detailed information about the Unison Exchange, see [Tennenhouse 87].

The Unison Exchange is actually a high speed packet switch. It is well suited to multimedia applications. Firstly, since the bit rates of the multimedia services are variable, transfer of these services over a variable bit rate network is more efficient [Verbiest 86, Littlewood 87, Chin 88]. Secondly, multimedia services usually involve a lot of inter-site interactions. The characteristic of low delay (except the first interaction) of the network is suited to the multimedia environment.

In the initial stage of the Unison project the client local area networks used were the Cambridge Rings (CRs). The CR is a 10 Mbps slotted ring [Hopper 86], the predecessor of the 50-100 Mbps Cambridge Fast Ring (CFR). Because of the limitations of the low CR bandwidth and the low throughput of the Portal between CR and the exchange CFR, the client CFRs were used to replace the client CRs in the later part of the project.

2.1.2 The Unison Office

As mentioned briefly in Chapter 1, the objectives of the Unison project are to develop a network architecture suited to the expected communication needs of the 'office of the future' and to develop the office itself based on the network.

The Unison definition of an office is [Clark 86]:

'Anywhere where human work is performed involving the use of communication and/or the manipulation or processing of data or idea, but excluding anywhere whose primary function is to process materials or living things'.

The multimedia office system is by nature diverse and makes a broad range of demands on its communications system. The Unison office is expected to have the following features:

(1) The office should be based on a LAN which is connected to the outside world through the Unison exchange.
The office should contain a heterogeneous collection of multimedia components such as voice, video, electronic blackboard and file transfer systems. Each component should have its own independent network interface so that the reliability of the office operation is high. The failure of one component should not affect the functioning of others.

Although the components are physically discrete, they are logically bound together in software centred on a workstation so that it is easy to manage and control these components.

The workstation used in the project is an Acorn's A-500 which is based on RISC (Reduced Instruction Set Computer) technology. The A-500 workstation is at the heart of the Unison office, and provides the user interface for the control of various office components.

This thesis is concerned with the provision of video and electronic blackboard services in the Unison office. For a full description of the Unison office, refer to [Griffiths 88, Murphy 89].

As mentioned in the preceding subsection, the CR was used as the office LAN in the first part of the project. The Electronic Blackboard System (EBS) was designed and developed at that stage. Although there were LUT framestores working on the client CR, they are not suited to work on the CFRs (the full reasons are given in Chapter 3). Therefore a video codec was designed and implemented for the CFR. To distinguish between these two types of video codecs, the old LUT framestore is called the CR video codec, the latter is called the CFR video codec, and is one of the main topics of this thesis.

The protocols used in the initial Unison network, when the CRs were used as the client LANs, are different from the protocols used in the later part of the project when the CFRs were used as the client LANs. The EBS has been implemented using the former set of protocols and the video service concerned in this thesis was implemented using the latter set of protocols. These two sets of protocols are discussed in the following sections. To distinguish these two sets of protocols, the former set is called the CR-based network protocols, and the latter set is called the CFR-based network protocols in the following discussion.
2.2 The CR-based Network Protocols [Adams 84, Leslie 83]

The initial Unison network consisted of the CRs interconnected by the ISDN and the Unison exchange. The CR-based network protocols are an extension of the CR protocols. The exchange and the ISDN simply pass data to and from the CRs. The network management system allows a user to talk to a remote service in the same way as he talks to a local service. A user is not aware of whether he is using a local service or a remote service.

2.2.1 Physical Level Protocol [Hopper 86]

The CR has a raw bandwidth of 10 Mbits/s. It works on the empty slot principle. A slot, or minipacket, is a collection of 38 bits with the format shown in Fig.2.2. There is a fixed number of slots circulating continuously around the ring. Stations wishing to transmit wait for an empty slot to arrive, mark it full and fill in the address and data fields. The minipacket continues around the ring to its destination which marks the response bits and normally accepts the data.

The minipacket then returns to the sender where the slot is marked empty and the response bits are read.

Each ring station has a selection register to screen out unwanted traffic. A station selects a particular source to listen to by writing the address of that station into the select register. Alternatively, a station may listen to all the stations by writing the number 255 into the select register, or to no station (a process known as "going unselected"), by writing 0 into the select register. If a minipacket is not accepted because the destination chose not to listen to its source, this is indicated to the sender in the response bits. There are four possible responses:

- **ignored** -- no station with the destination address was active,
- **busy** -- the host attached to the destination station had not yet read the last received minipacket from the station logic,
- **unselected** -- the destination station did not wish to receive from the source, and
- **accepted** -- the minipacket was received by the destination station.
Fig. 2.2 Minipacket Format of the Cambridge Ring

The monitor-passed and parity bits are used to detect error conditions. The monitor-passed bit allows a special station, the monitor, to detect when a transmitter fails to empty a returning slot.

The time delay around the ring is a combination of propagation delay and delay in the repeaters. This time determines the number of bits which fit into the ring. This number may not make up an integral number of slots, so gap digits are inserted between the slots. In fact, ring stations require a minimum of three gap digits in order to synchronize to the slot structure. However, in order to simplify comparisons, it will be useful to speak of rings which are exactly filled by an integral number of slots.

The system bandwidth of a ring, that is the useful data bandwidth of the ring, is independent of the ring size assuming that there are no gap digits. It is a constant determined by the number of user data bits in a slot, the total number of bits in a slot, and the raw bandwidth. This constant is thus \((16/38)*10\) MHz or 4.2 MHz. The point-to-point bandwidth, that is the highest data rate from one station to another, does depend on the ring size, since a station must wait for each minipacket to return before sending another packet. Furthermore, an anti-hogging mechanism prevents a transmitting station from immediately re-using a slot on its return. Thus the point-to-point bandwidth is bounded by \(1/(n+1)\) times the system bandwidth, where \(n\) is the number of slots in the ring. For details of the point-to-point bandwidth calculation, see Appendix 1.
2.2.2 The Basic Block Protocol

The transmission of data across the network is almost exclusively performed using the basic block protocol (BBP). A basic block is a sequence of minipackets and begins with a header minipacket. Following this is a route minipacket and this is followed by between 1 to 1024 data minipackets. Finally, there is a checksum minipacket which is used for error-checking. The format of the basic block is shown in Fig. 2.3. The figure only shows the data part of the minipackets used.

The header minipacket contains a 4-bit pattern to identify it as a header, a 2-bit type field and a 10-bit size field which indicates how many data minipackets there are in the block. The route minipacket contains a 12-bit port number which is used to direct the block to the appropriate place within the destination station. The remaining 4 flag bits are used for various purposes in intelligent ring interfaces.

Ports are generally used by hosts in two ways. A public, or well known port is active (i.e., there is a reception request outstanding for that port,), for a long time, perhaps for as long as a host is switched on. Public ports are used to receive from any station. A private port, on the other hand, is allocated dynamically as the need arises when a connection is established. These ports are used to receive from a specific station.

A ring driver is responsible for dealing with reception and transmission requests from the host. A reception request specifies the source of the transmission and the port number on which the block must arrive. When a basic block header is received

| (4)fixed pattern | (2)type | (10)size |
| (4)flags | (12)port number |
| 1 - 1024 data minipackets |
| (16) checksum minipacket |

Header minipacket
Route minipacket

Fig. 2.3 Format of a Basic Block (a number in brackets refers to the number of bits)
on that port from the source specified, the ring driver will receive the rest of the minipackets in the block and will send the accepted return code to the originating host. A transmission request specifies the destination address, port number and data to be transmitted. The ring driver will make a basic block and send it to the destination. The result of whether the transmission is successful or not will be returned to the host. The BBP makes it easy to transfer bulk data.

2.2.3 The Single-Shot Protocol

The single-shot protocol (SSP) is based on the basic block protocol. It is a request/reply protocol in which the requesting host sends a single basic block containing the request and receives a single basic block in reply. The blocks are identified as belonging to this protocol by reserving the first three data minipackets to contain identification and parameters as shown in Fig. 2.4. The figure only shows the data portion of the basic blocks used, but indicates to which port they are sent. The port to which an SSPREQ is sent is a public port; the reply port contained in the SSPREQ, and to which the SSPRPLY is sent, is a private port.

The receipt of a reply block is taken as evidence that the request was accepted and if no reply is received within a certain time the protocol may specify that the request be made again. It is possible that the request was received but the reply was lost and so it is important that the request should be repeatable.

<table>
<thead>
<tr>
<th>SSPREQ client--&gt; server, service port</th>
<th>SSPRPLY server--&gt; client, reply port</th>
</tr>
</thead>
<tbody>
<tr>
<td>(8)6C hex</td>
<td>(8)FLAGS</td>
</tr>
<tr>
<td>REPLY PORT</td>
<td></td>
</tr>
<tr>
<td>FUNCTION CODE</td>
<td></td>
</tr>
<tr>
<td>ARGUMENTS</td>
<td></td>
</tr>
<tr>
<td>(8)65 hex</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RETURN CODE</td>
<td></td>
</tr>
<tr>
<td>RESULTS</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2.4 Single Shot Protocol Formats
2.2.4 The OPEN/OPENACK Protocol

The OPEN/OPENACK protocol is a light-weight protocol used when a large amount of data needs to be transferred. One station wishing to communicate sends an OPEN basic block to another station which acknowledges with an OPENACK basic block. The formats of the OPEN and OPENACK blocks are shown in Fig.2.5. The only difference from the SSP exchange is in the use of the reply port and the addition of a connection port in the OPENACK block. In the case of the SSP exchange, the reply port is only active until the reply is received. In the case of the OPEN/OPENACK exchange, the reply and connection ports remain active until the connection is closed. The reply port number specified in the OPEN and the connection port number specified in the OPENACK are used for future communication between the two stations. Thus the OPEN/OPENACK sets up a virtual circuit between the two systems which will remain active until closed by the systems.

2.2.5 The Network Operation

Hosts on the network can communicate with each other over the network using the above protocols with the help of network servers. The station address and port number to which an SSPREQ or OPEN is sent is obtained from a name lookup service. This service, often referred to simply as the nameserver, binds service names to addresses. The address of the nameserver is the only one that is written
into application programs. In order to find a service, the name of the service is given to the nameserver in a name lookup request. The name server will reply with a station address, port number and function code. The name lookup process is in fact an example of the use of the SSP.

The simple scheme where each local ring has a nameserver in which all network service names are stored is strained by a network as large as the Unison network. The large network size introduces problems both in storing all the names in all the nameservers and updating the contents of all the nameservers. These problems are solved by the introduction of naming domains. The Unison name space is broken into several subspaces, each called a naming domain. Each ring belongs to one domain, its home domain, although a domain may contain several rings, possibly at several sites. A nameserver stores only the names in its home domain.

Naming domains reduce the number of names each nameserver must store, but perhaps more importantly, they allow each domain to be controlled by separate naming authorities who can specify their own policy for the addition and deletion of names.

Nameservers and portals provide the mechanisms by which hosts are able to send initial connection blocks to hosts in different domains. Instead of merely translating from a name to an address, as in the single ring operation, nameservers also interact with portals to set up paths across the network on behalf of the clients, and may return to the client, not an address, but a path. A client need not be aware of the difference, since a path is essentially a ring address on a portal. The portals provide temporary associations between paths and service addresses on the network.
2.3 The CFR-Based Network Protocol

The CFR-based Unison network is the same as the CR-based Unison network in terms of the network topology, but there are some advanced features in the CFR-based UNISON network. The first feature is that the 50 Mbits/s CFR replaced the 10 Mbits/s CR. Thus the network can provide users with wider bandwidth. The second feature is that the Secretary replaced the nameserver of the CR-based network. In the CR-based network, a two-stage process is used to establish an association. The first stage is the nameserver lookup. This locates the desired peer service (possibly performing a remote lookup) and returns the attributes of the service known to the nameserver. If necessary, a bridge setup is performed to establish a path to the station hosting the peer service. At the completion of this stage the initiating client holds the address of the public port at which the service is believed to be available. The initiating client may then communicate using the SSP protocol or the OPEN protocol to request that a private port be allocated by the recipient. Once a private port has been allocated the peer services can exchange data in a fairly transparent manner. The Unison Secretary combines the two stages of association establishment. In addition to locating a recipient service the Secretary contacts that service and secures a private port for the association. A RPC (Remote procedure Call) mechanism [Hamilton 84] is used in establishing the association. The CFR protocol and association setup process will be discussed in the following section.

2.3.1 The CFR Protocol

The CFR is made up of three types of nodes: stations, which transfer data between devices attached to the ring; bridges, which copy minipackets between rings, and monitors, one of which is required on each physical ring to set up and maintain the slot structure [Hopper 86].

The slots (minipackets) of the CFR are classified into two types according to the way they can be used for transmission. These two types are called normal slots and channel slots, and they are distinguished by a bit in the slot. The normal slots can be used for transmission as on the CR. The channel slot is allocated to a user who has exclusive use of that slot until he no longer requires it. When a slot is not
allocated for channel use it is available for use in the usual way. The monitor is programmed to decide which slots may be used in channel mode and which slots may only be used in normal mode. A typical system might have a number of channel mode slots for bulk data transfers and at least one normal slot to guarantee that some bandwidth will always be available regardless of the usage of the channel slots. The minipacket format is shown in Fig.2.6.

The control bits at the front of the minipacket occupy four bits and comprise a start bit, a full/empty bit, a monitor-passed bit and a channel slot bit. The latter is set in slots which may be used in channel mode and unset in normal slots.

Following the control bits in a slot are the destination and source addresses. These are 16 bits long each and they are followed by 256 bits (32 bytes) of data. At the end of the minipacket is a 12-bit CRC (Cyclic Redundancy Checksum) of all the bits in the minipacket. Encoded in the CRC is a single response bit. It can therefore encode the two states of the receiver and inform the source which of these states is applicable. These states have been chosen to mean 'Try Again' and 'Don't Try Again'. Unlike the CR, this response is not passed back to the host but is used within the station to govern whether an automatic retransmission should be attempted. Whenever the try-again response is seen the transmitter will retransmit the packet, subject to some maximum number of retries. The destination will give the try-again response if its receive buffer is full or if it detects a CRC error in the slot. It will give a don't-try-again response if the select mechanism prohibits reception or if it accepts the packet.
Although the response is not passed back to the host, the host can check whether the transmission has been successful after a predefined number of station retries. If the transmission fails, the host can retransmit the packet if it wants to do so. This kind of retransmission is referred to as software retry. The automatic retransmission by the station is called a hardware retry.

It should be noted that the response is only local. If the transmission passes through a portal (from one CFR to another), the source knows whether the transmission to the portal is successful or not, but it does not know whether the transmission from the portal to the destination (or another portal) is successful or not. If the source wants to know the response from the remote destination, the destination has to send an explicit acknowledgement minipacket to the source to indicate the arrival of the data at the destination.

### 2.3.2. The Unison Secretary Service

The Unison Secretary provides support for light-weight associations between Unison hosts. A client wishing to bind to a service uses the unity RPC mechanism to communicate with its local Secretary. The Secretary is responsible for locating the peer service and for configuring a direct link between the hosts to be associated.

The intent is to separate the establishment of associations from the exchange of data between associated hosts. The associated hosts are provided with a raw end-to-end data link. There are no restrictions on the upper layer protocols that can be used over the association.

#### 2.3.2.1 S-association Setup

A host on the network has to establish an S-association with its local Secretary before the host is able to create associations to other network services or before the host wishes to offer services.

The Secretary has a fixed address and a 'well known port'. When a host is switched on, it allocates a port that is to be used for communication with the Secretary and constructs a single minipacket with the S-allocate indicator and the port allocated in the data field. The minipacket is sent to the Secretary on the 'well known port'. The Secretary receives the minipacket and allocates a port which will be used by
the host on the S-association. The Secretary constructs a single minipacket with the S-reply indicator and the port allocated by the Secretary in the data field and returns the minipacket to the host. An S-association has thus been established between the host and the Secretary with the two ports allocated by the host and the Secretary as the communication ports (S-association ports).

2.3.2.2 Association Creation between Hosts

A host initiates an association by sending a block of data containing an Associate.Create.Request primitive to its local Secretary over the S-association established as above. The request will name the intended recipient service and will specify a private port for the association. If the Secretary can locate this service at the local site it will issue an Associate.Create.Indication primitive to the S-association port of the recipient quoting the names of the intended recipient service and of the initiating service together with the initiator’s station address and the private port specified by the initiator.

If the addressed station supports the service and approves the initiating service it can accept the association. In this case it must allocate a private port for the association and return the port address to the Secretary as part of an Associate.Create.Response primitive. Alternatively, the station can reject the association by returning a negative Create.Response to the Secretary.

When the Secretary finds the station willing to support the association it issues an Association.Create.Confirm to the initiating station’s S-association port. This primitive names the recipient service and quotes its station address and private port number. The two hosts then can communicate using the two private ports allocated in the association creation process.

When the Secretary determines that an intended recipient service is supported at some distant site it will initiate communication with the peer Secretary at the distant site. The Secretary will issue an Associate.Peer.Create.Request primitive to its peer. The peer will attempt to locate the desired service and will perform the Create.Indication/Create.Response sequence with the recipient. The peer will issue an Associate.Peer.Create.Confirm to return the results of its efforts. If the Peer.Create is successful the initiating Secretary confirms the association to the initiating client. Otherwise the Secretary informs the initiating client of the failure.
2.4 Summary

This chapter briefly describes Unison network architecture that was developed to interlink geographically dispersed client local area networks. The network provides a user with similar characteristics to that of a local area network.

The communication protocols of both the CR-based network and the CFR-based network have been outlined in order to give the reader the necessary background for a full understanding of the following chapters on graphics and video data transmissions.
CHAPTER 3

Investigative Study of Electronic Blackboard Systems

3.1 Introduction

An Electronic Blackboard System (EBS), or a telewriting system, is a system which can transmit graphics or drawings while they are being generated. It is different from facsimile communication, which transfers already existing graphics. Telewriting systems were invented in the latter part of the last century, and have been used extensively in the following areas:

(1) when people are discussing something which is difficult to explain in words,

(2) when people who have difficulties in speaking and hearing, communicate with other people,

(3) in a noisy place, such as a factory, where it is difficult to hear,

(4) in fire, police and truck dispatching and parts ordering offices where records of communications need to be kept.

In this chapter, three development stages of telewriting systems are examined, followed by a summary of the development trend in this area. Some common computer graphics input devices are also reviewed. Finally a design guide for the Unison EBS is established.
3.2 Three Development Stages of Telewriting Systems

The development of telewriting systems can be roughly divided into three stages. The first stage is from the invention of the first telewriting system to the late 1960's. The second stage is from the early 1970's to the early 1980's and the third stage, the early 1980's to the present.

The First Stage[Costigan 78]

The first telewriting system was invented in the latter part of the last century by an American inventor, Elisha Gray. The original system used a relatively simple arrangement of a pair of rheostats in the transmitter to vary the current flow through an associated pair of electromagnets in the receiver. The rheostats were operated via mechanical linkages with a hand-held stylus. The movement of the operator's hand thus produced "x" and "y" current variations, which were communicated via separate wire loops to the associated electromagnets in the receiver, where the moving armatures of the electromagnets were similarly linked to a stylus, arranged to record its movements on a piece of paper.

Subsequent telewriting machines continued to employ much the same principle as the original one, but with servomechanisms replacing the crude rheostats of the earlier units.

These systems acted as an electrical extension of the operator's hand. However, they were not very easy to use because of the servomechanism linkages.

The Second Stage

This stage is marked by two features. The first is that coordinate contacting grids were used in transmitters to permit "unshackling" of the writing stylus from its servomechanism linkages. The second is that some kind of display, instead of a piece of paper, was used in the receiver.

A typical example is the electronic blackboard introduced by Bell Labs in 1971. It looks like a conventional chalkboard, but the send terminal transmits its chalked "message" via phone lines to a Cathode Ray Tube (CRT) display, where it is reproduced in near real time. After that, similar systems have been developed using essentially the same principle.
The systems mentioned above have limitations. Firstly, they need a special line to transmit the graphic signals. Secondly, the displays are not shared among the users. For example, user A communicates with user B at a remote site via the telewriting machine. The display at the user A site can only display the drawings from user B. Likewise the display at the user B site can only display drawings from user A. A user cannot write over the other’s drawings. This restricts effective discussion, because it is necessary for one partner to add something to or modify something on the other’s drawings during a discussion.

**The Third Stage**

This stage is marked by the digital transmission of graphics data and integration with other services. The early systems used analogue signals to transmit graphics and needed a separate telephone line. Systems in the third stage use digital transmission. They are combined with telephone sets and/or other data terminals, and the graphics and voice and/or other data are transmitted on one digital line.

An example is the "ISDN telewriting terminal system" from Taiko, Japan, which was exhibited at the Telecom’87 Show in Geneva. This system features voice and drawings transmission using one B channel. It uses a graphics tablet with a wireless stylus as the graphics input device. The displays are shared by the two parties, thereby providing the facility to draw over the other user’s graphics. It also allows a hardcopy of the graphics on the display to be obtained.
3.3 Development Trend

The development trend in telewriting systems may now be summarised as follows:

(1) The graphics generating devices are becoming more user-friendly. Devices have been developed from the servomechanism linkage in the first stage to the graphics tablet with wireless stylus in the third stage.

(2) Common displays are used, such as CRT display and LCD, which can be shared among the communication parties.

(3) Digital transmission techniques are used and the telewriting service is integrated with other services.

In brief, the telewriting systems are becoming more user-friendly and will form part of the ISDN basic teleservices [Sebestyen 87].

There are two areas relevant to the transmission of graphics, which have been developed significantly. They are computer graphics technology and the integrated service digital network. Nowadays, almost all personal computers and workstations are equipped with advanced graphics facilities, and computers are connected by LANs and WANs. If a computer's local graphics facilities are extended over a WAN an EBS can be easily achieved. This is a nice and cheap solution. Firstly, it is easy to generate graphics. In the systems discussed above, the graphics function is limited. The only function available is free-hand drawing. There is no tools to help the user to draw useful shapes, such as rectangles and circles. Using a computer, these functions can be easily introduced. Secondly, since computers are connected to LANs and WANs, the EBS based on these computers can be easily integrated with other services.

The next section gives a review of the computer graphics input devices which are very important in determining whether an EBS is easy to use or not.
3.4 A Review of Graphics Input Devices

A graphics input device which can be used to point to a particular position on the screen easily, for example, to select a particular entity from a menu, is called a pointing device. Similarly, a device which can be used to draw a shape on the screen easily is called a position device [Mcgregor 84].

The most common graphics input devices are graphic tablets, mice, lightpens and touch screens.

Graphics Tablet

A graphics tablet is one of the most convenient input devices. It consists of a flat surface, separate from the display, over which some kind of stylus is moved. It can be used both as a position device and a pointing device. Using a graphics tablet for positioning is much more natural than using other devices, because it is equivalent to drawing with a pen on paper.

There are a range of graphics tablet technologies. But the aims of these technologies are the same: to detect the stylus position as accurately as possible.

Usually a graphics tablet has two or three buttons to provide menu selection and action control signals. Some styli have switches at their tips to give the pen-up and pen-down state. According to the state and the stylus position, the user can do menu selection and action control easily without the need for any other control signals.

Mouse

A mouse is a very good pointing device. There are two types of mice, optical mice and mechanical mice. The former needs a special grid metal flat surface to operate on. The latter does not need a special surface, rather any flat surface will suffice. The x and y coordinates are generated according to the distance the mouse has moved in the x direction and y direction on the bases of the initial x, y coordinates.

A mouse has two or three buttons to provide menu selection and action control signals.
Light Pen

A light pen is both a pointing device and a position device. It contains a photo-detector and is moved over the surface of the display screen. When the raster scan beam comes into coincidence with the light pen, a pulse is generated and detected by the processor. The processor compares the time of this with the start of the raster scan and calculates the x and y coordinates of the stylus point.

The weakness of a light pen is that the keyboard has to be used when menu selection and action control are needed. Furthermore, most of the photo-detectors are not sensitive enough to allow accurate positioning, and hence the resolution is quite low.

Touch Screen

A touch screen is a transparent mask over the screen, which consists of a layer of glass and a resistive membrane sensor. When touched, it causes contact between the cover sheet and the coated glass, and thus the x and y coordinates are generated. The disadvantage of the touch screen is that, since a special stylus is not needed unwanted coordinates might be generated easily. For example, if a finger and palm touch the screen at the same time, an unwanted action may be generated. It is a good position device but is very expensive.
3.5 Design Considerations for the Unison EBS

3.5.1 Design Objectives

The design objectives of the Unison electronic blackboard system are:

1. It should be user-friendly.
2. The data it generates should not interfere with other traffic so that it can be integrated with other services on the network. It means the output data rate should be kept as low as possible.
3. It should support multipoint service.
4. It should be cheap and easy to implement.

3.5.2 Discussion

As mentioned before, an EBS based on a workstation is of advantage. Unison is in a good position to build an EBS based on a workstation since the Unison office is centred on the A-500 workstation which is connected to the network. It is natural that the EBS be based on the workstation instead of another set of hardware.

There are two aspects to user-friendliness. The first is the provision of good graphics facilities, which include the user interface and graphics generating functions. The other is the provision of a user-friendly graphics input device.

The A-500 is well endowed with advanced graphics facilities. In addition, the A-500 has the Arthur WIMP(Window, Icons, Menus and Pointers) system[Acorn 87]. It has been designed to simplify the task of the application programmer who wishes to take advantage of the windowing approach as the user interface. Using this system, windows, menus and icons can be created, redrawn, moved around and deleted easily. Thus the A-500 is well suited to the design of an EBS.

Ideally, a user-friendly graphics tablet should be used in the EBS since a graphics tablet is a good pointing device as well as a good position device, thereby satisfying requirements of pointing and free-hand writing, both of which are
important to an EBS. But there are two problems in using the graphics tablet in this particular work. Firstly if the graphics tablet is used it is difficult to make use of the WIMP system on the A-500. Secondly there is no interface on the A-500 for the graphics tablet to connect to. Since there is no interface available between the A-500 and the CRs, a BBC-MASTER microcomputer is used to connect the A-500 and the CR. Thus, a graphics tablet can be connected to the MASTER and the MASTER then talks to the A-500. A mouse is used by the WIMP system on the A-500, though it is not a good position device, it is a good pointing device. For the above reasons, it was decided that two versions of software be developed, one for the graphics tablet, and the other for the mouse, so that either the graphics tablet or the mouse can be used as the input device. The user may then decide which device to be used.

Since only the vector information of the graphics needs to be transmitted, the bandwidth required by the EBS is very low, on average below one hundred bytes per second. It is easy for these data to be transmitted over the network without interfering with other traffic. Thus, the EBS can be used with other services, such as voice and video, over the network.

There are usually two ways to realise multipoint services. One is that the network provides some kind of agent which will be responsible for setting up connections and broadcasting the data on behalf of the data originator (client). The other is for the client to set up connections with other clients separately and then transmit the same data several times to the individual clients. The former is efficient and reduces the workload of the clients. The Unison EBS should be able to work either way, so that the multipoint service can be realized no matter the network provides the agent or not.

Since the workstation is already interfaced to the network and there is no need for any significant hardware to be built for the EBS, the EBS is cheap and quick to implement.
3.6 Summary

A brief history of EBS development has been given in this chapter. From the general development trend, it can be seen that an EBS based on a personal computer is of advantage. It was also decided that mice and graphics tablets will be used in the Unison EBS.

Unison is in a good position to design and implement an up-to-date EBS because the A-500 workstation is already in the Unison office, and it has an interface to the network. The Unison EBS based on the workstation has several advantages:

Firstly, the workstations have good graphics facilities. The EBS based on them can generate graphics easily and effectively.

Secondly, a user usually does control and communication through the use of the workstation. It is easier for him to operate the EBS based on a workstation than to operate an EBS based on another set of equipment.

Thirdly, since the workstation is already interfaced to the network, the EBS based on it does not need a dedicated interface, including hardware and software, which is usually a complex job.

Fourthly, since no additional equipment is needed the EBS is cheaper and may be implemented quickly.

Graphics are displayed on the monitors or windows of the monitors of the workstations, whether they are generated locally or received from other workstations. The display areas are shared among all the participants just like a normal blackboard is shared among them. The graphics could be typed text from keyboards, regular shapes generated by some kind of graphics software or any free-hand drawings. Although the word "blackboard" is used for historical reasons, the background of a display is not necessarily black. It is up to the user to decide what background colour is used.

In brief, the Unison EBS will be based on workstations with mice or graphpads as input devices. The system will be shared by all participants. All participants may write on it and see exactly the same graphics on their local workstation monitors.
CHAPTER 4

The Unison Electronic Blackboard System

4.1 General Description

A Unison Electronic Blackboard System (EBS) was developed in the first phase of the Unison Project when the CRs were used as the client LANs. The design of the EBS is based on the guidelines established in the previous chapter. The system comprises several workstations connected to the Unison network and uses either the mice or the graphics tablets as the control and graphics input devices. Fig.4.1 shows the configuration of an EBS station. The A-500 workstation is connected to the CR via a BBC-Master microcomputer through the BBC-Master Tube and an A-500 Podule because of the lack of a direct interface between the A-500 and the CR. The BBC-Master is connected to the interface card via the 1 MHz Bus. Since there is no port available on the A-500 for the graphics tablet to connect to, the graphics tablet is connected to the BBC-Master User Port. The BBC-Master collects data from the graphics tablet and passes them to the A-500.

The EBS provides users with an easy to use interface and many useful graphics functions to enable them to input colour graphics easily. These functions are free-hand writing, rectangle drawing, circle drawing, pattern filling and text input. These functions can be used together with different colours and thicknesses. Functions which erase a part of the graphics and clear the whole screen are also provided. The graphics on the screen can be saved onto disc, and the graphics saved on the disc can be loaded onto the screen. The system display is shared
among the users. All the users at different sites may write onto the display, and every one sees exactly the same graphics on his local workstation monitor.

The user interface, generation of graphics on the local workstation, graphics transmission and graphics reception and display are important elements of the EBS. In this chapter these processes are described, followed by an overview of the system in terms of the software implementation. Although both the mice and graphics tablets can be used as the control and graphics input devices, for the sake of simplicity, only the mice are referred to in the following discussion.
4.2 User Interface

A user interface has been designed based on the WIMP system on the A-500 workstation. Items provided for selection are represented by icons. When the cursor is moved over an icon by moving the mouse, the icon is highlighted. If the left hand button on the mouse is clicked when the icon is highlighted, the item represented by that icon is selected.

There are two phases to the user interface. The first one enables a user to select information necessary to set up connection to another station. The second part provides users with graphics functions, such as free-hand drawing and circle drawing.

Since the workstation is also used for controlling other services such as video and voice, running the EBS is only one function of the workstation. The EBS may be selected by moving the cursor over the "blackboard" icon and clicking the left hand button on the mouse.

Once the EBS has been started, the user will be prompted to select the workstation he wants to talk to. If he wants to talk to more than one workstations he needs to select more than once. After selecting the workstations, clicking on the "ok" icon on the screen will lead the EBS to prompt the user to select the blackboard size and position on both the local and remote workstations. The display of the EBS can be any size and at any position on the workstation screen. This information will be sent to the other workstations together with the connection setup request call.

After these interactions, the EBS enters the second phase of the user interface where graphics can be drawn using the functions provided. The functions are represented as menu items at the top of the blackboard display window. When an item is selected by clicking the left hand button of the mouse, it is highlighted and the corresponding function is enabled.
4.3 Local Graphics Generation

The first step in graphics communication is to generate the graphics locally. There are some commercial graphics software packages available, but they are designed for other purposes, such as CAD (Computer Aided Design). It is difficult to make use of these for the EBS because the graphics data they generate are not easily accessible and transmission of these data is difficult. Thus, some graphics generation procedures have been written for the EBS.

Several useful functions for the EBS are identified. The first function is free-hand drawing which allows a user to draw whatever he wants. To aid free-hand drawing, some functions are included to help in the drawing of some useful shapes, such as straight lines, rectangles (including squares) and circles. Two auxiliary functions are COLOUR and THICKNESS which define the colour and thickness of the drawings.

Another useful function that is identified is text input from the keyboard. Using this function, users can communicate in text. This 'text chat' is extremely useful when there is no voice service or when a user has some difficulty in speaking or hearing.

ERASE, CLEAR, SAVE and PRINT are also important functions. ERASE is used to clear part of the screen while CLEAR is used to clear the whole screen. SAVE is used to save the graphics on the blackboard onto disk for future reference. If a hard copy of the graphics is wanted, the PRINT function can be used.

A user-friendly menu with twelve items is provided to the user at the top of the blackboard display window. Their functions and uses are listed below.

WRITING

It provides the free-hand writing facility. When this function is selected (highlighted), the free-hand drawing will be produced by the movement of the mouse when the left hand button of the mouse is kept pressed.
Chapter 4 The Unison EBS

LINE

It is used to draw straight lines. When this function is selected, the starting point of a line is fixed by pressing the left hand button of the mouse. A line is formed by joining the starting point and the point at which the right button of the mouse is pressed after having moved the mouse.

RECTANGLE

It is used to draw rectangles. A rectangle is defined by a pair of diagonal corners. When this function is selected, the first corner is fixed by pressing the left hand button of the mouse. The second corner is fixed by pressing the right hand button of the mouse.

CIRCLE

It is used to draw circles. A circle is defined by its centre and any point on the boundary of the circle. When this function is selected, the centre is fixed by pressing the left hand button of the mouse. A point on the boundary is decided by pressing the right hand button of the mouse and the circle is drawn according to these two points.

TEXT

It provides text input facility from the keyboard of the A-500 workstation. When this function is selected, the starting point of the input text is fixed by pressing the left hand button of the mouse. After the starting point has been fixed, text can be keyed in normally. The text input is terminated by pressing the RETURN key.

PATTERN

It provides a pattern filling facility. It is always useful to fill a certain area with a certain pattern. When this function is selected, several patterns and colours can be chosen. If the left hand button of the mouse is pressed in a closed area, this area will be filled with the coloured pattern selected.
ERASING

It is used to erase a portion of the screen. After this function is selected, if the left hand button of the mouse is kept pressed while moving the cursor, the area traversed by the cursor is erased to the background colour of the screen.

CLEAR

It is used to clear the whole screen. When this function is selected, double clicking the left hand button of the mouse will clear the whole screen to the background colour. The reason for double clicking is to prevent clearing of the screen accidentally, since the action is not reversible.

SAVE

It is used to save the graphics on the screen onto disk. When this function is selected, clicking the left hand button of the mouse will save the graphics on the screen onto disk.

PRINT

It is used to dump the graphics on the screen onto a printer. When this function is selected, a click of the left hand button of the mouse will output the graphics on the screen to the printer. It should be noted that the screen dump is quite slow.

COLOUR

It is used to provide colours for the graphics. It is used together with the WRITING, LINE, RECTANGLE, CIRCLE and TEXT.

THICKNESS

It is used to draw graphics in different thicknesses. It is used together with the WRITING, LINE, RECTANGLE and CIRCLE.
4.4 The Connection Setup

The previous section has discussed the local graphics generation. This section describes the establishment of a communication channel over which the graphics generated on one workstation are transmitted to another workstation.

The OPEN/OPENACK protocol is used to transmit the graphics data. The reason for the use of the OPEN/OPENACK protocol instead of the Single Shot Protocol (SSP) is that the latter is slower and inefficient in utilising the network bandwidth. If the SSP is used every time a block of data needs to be transmitted, the transmitter will perform a name server lookup to the local name server, and if the receiver is at another site, the name server at that site will be consulted via the ISDN link. After the transmission of the block of data, the transmitter would have to wait for the acknowledgement from the receiver. In the case of the OPEN/OPENACK protocol, however, once a communication channel between two stations is established at the beginning of a session, the graphics data can be sent directly down the channel without any other interactions. Both stations involved can use this channel for data transmission and reception since it is a bidirectional channel. The only problem in using the OPEN/OPENACK protocol is that there is a timeout for the channel. If the channel has not been used for a certain period of time it will be terminated. The solution to this problem is to send an empty block down the channel before the timeout, in order to keep this channel active. Since the timeout is set to be over ten minutes, the frequency of the transmission of the empty block is very low.

All the information needed to set up a connection between the workstations, such as the station name, blackboard window position and size, is obtained from the user interface on the workstation. The interface design has been based on the WIMP system. The steps required for station A to set up a channel to station B are as follows:

(I) User A at station A selects station B and the position and size of the blackboard windows on both stations by moving and clicking the mouse on the user interface.

(2) A name server lookup is made by station A to get the address and port number of station B.
(3) Station A sends an OPEN to station B using the address and port number obtained in step (2). The blackboard information (position and size of the graphics window) is sent to station B together with the OPEN.

(4) Station B replies to station A with an OPENACK.

A bidirectional channel is thus created between station A and station B and graphics data can be sent directly over this channel. For details on the network protocols, see Chapter 2.

4.5 Graphics Data Transmission and Reception

Graphics data are transmitted using the basic block via the connection established by OPEN and OPENACK. If the transmission fails, the transmitter will try again until it is successful or until a transmission timeout occurs. The transmission failure information is returned to the user. Since the network error rate is very low, no further error correction method is implemented.

In order to make graphics data transmission as real-time and effective as possible, a basic block is transmitted after every 'action', regardless of whether it is full or not. For network bandwidth saving, a minimum amount of graphics information is transmitted. For instance, to transmit a rectangle, only the coordinates of two opposite corners of the rectangle are transmitted.

Actions have different meanings in different graphics modes. In the "WRITING" and "ERASING" modes, an action starts from the point at which the writing or erasing was started by pressing the left hand button of the mouse, and ends at the point at which writing or erasing was stopped by releasing the button. Naturally, if an action lasts too long, a basic block will not be enough to pack the data generated from this action. But tests have shown that a basic block is usually big enough for a reasonably long action. Whenever one basic block is not sufficient, a second block will be used.

In the "LINE", "RECTANGLE", "CIRCLE", and "CLEAR" modes, an action signifies the finishing of the drawing of a line, a rectangle or a circle, or just the finishing of a clearing command.
In the "PATTERN" mode, an action means finishing filling the area selected. If all the information filled is transmitted, one filling action will take several basic blocks since there is usually a lot of information required to fill an area. However, as every workstation involved in the communication has exactly the same graphics on its screen, the information as to which area is going to be filled and which pattern is going to be used is sufficient for each workstation to fill the area as required. This information only takes a few bytes.

In the "TEXT" mode, an action starts from the start of text entry and terminates when the RETURN key is pressed.

The format of the data part of the basic block used to transmit graphics data is shown in Fig. 4.2.

To preserve consistence the first seven bytes are reserved for every mode, although for some modes some bytes are not used. For instance, the "CLEAR" and "TEXT" modes do not use the THICKNESS byte.

DATA TYPE is used to identify the type of data. Different traffic, such as voice, video and graphics, has different DATA BYTE. It is &6AA0 for the graphics data.

DATA COUNT shows the length of the data block in bytes, excluding the first seven bytes.

```
<table>
<thead>
<tr>
<th>DATA TYPE MINIPACKET</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA COUNT</td>
</tr>
<tr>
<td>WORKING BYTE</td>
</tr>
<tr>
<td>THICKNESS</td>
</tr>
<tr>
<td>COLOUR</td>
</tr>
<tr>
<td>DATA</td>
</tr>
</tbody>
</table>
```

Fig. 4.2 Format of a data block
WORKING BYTE shows the working mode in which the data is generated. Each mode has a unique working byte. They are assigned as follows:

<table>
<thead>
<tr>
<th>graphics mode</th>
<th>working byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRITING</td>
<td>&amp;01</td>
</tr>
<tr>
<td>LINE</td>
<td>&amp;02</td>
</tr>
<tr>
<td>RECTANGLE</td>
<td>&amp;03</td>
</tr>
<tr>
<td>CIRCLE</td>
<td>&amp;04</td>
</tr>
<tr>
<td>TEXT</td>
<td>&amp;05</td>
</tr>
<tr>
<td>PATTERN</td>
<td>&amp;06</td>
</tr>
<tr>
<td>ERASING</td>
<td>&amp;07</td>
</tr>
<tr>
<td>CLEAR</td>
<td>&amp;08</td>
</tr>
<tr>
<td>SAVE</td>
<td>&amp;09</td>
</tr>
<tr>
<td>PRINT</td>
<td>&amp;0A</td>
</tr>
</tbody>
</table>

THICKNESS is useful only in "WRITING", "LINE", "RECTANGLE", "CIRCLE", and "PATTERN" modes. It shows the thickness of the drawings or the number of the pattern used for filling. There are four different thicknesses and seven patterns from which a user can choose. The use of this byte can save a lot of transmission bandwidth. Provided that the minimum necessary information for a basic shape is transmitted to the receiver, the receiver can generate a thicker one. For instance, if the coordinates of two opposite corners of a rectangle are transmitted, the receiver can generate a rectangle of any thickness according to the thickness information provided by this byte.

COLOUR provides the colour information of the drawing or text to the receiver. The receiver can display the colour graphics as required according to this information.

DATA contains the necessary information about the graphics to be transmitted. The content and length of DATA vary with different modes.

For the "CLEAR" mode, a working byte and a colour byte are enough to instruct a receiver to clear the screen to a specified background colour. DATA is not used.
For the "TEXT" mode, the DATA contains the text start position coordinates and ASCII codes of the characters keyed in.

For the "WRITING", "LINE", "RECTANGLE", "CIRCLE", "PATTERN" and "ERASING" modes, the contents of DATA is a sequence of coordinates. On the workstation, the x and y coordinates range from 0 to 1279 and 0 to 1023 respectively. Thus, although one byte is not enough to represent a coordinate, two bytes for each coordinate would mean a waste of several bits. Since the representation of the x coordinate requires 11 bits and the y coordinate requires 10 bits, they can share 3 bytes (24 bits) by a simple coding technique as follows:

- first byte (FB) = x AND &FF
- second byte (SB) = (x DIV &100) OR ((y DIV &100)*32)
- third byte (TB) = y AND &FF.

At the receiver, they can be decoded via

- x = FB OR ((SB AND &7) * &100)
- y = TB OR ((SB AND &F0) * &100).

OR and AND in the above equations are bit operators. The numbers followed by & are hexadecimal numbers.

In the "WRITING" and "ERASING" modes, the coordinates which are transmitted to the receiver are obtained by sampling the cursor movement (caused by the mouse movement) as fast as possible at the transmitter in order to make the writing as smooth as possible and erasing as accurate as possible.

In the "LINE" mode, only the coordinates of the starting point and the end point of a line are transmitted.

In the "RECTANGLE" mode, only the coordinates of two opposite corners of a rectangle are transmitted.

In the "CIRCLE" mode, only the coordinates of the centre and radius of a circle are transmitted. The receive workstation can draw the circle according to these two parameters.

In the "PATTERN" mode, DATA contains only one pair of coordinates of the point at which the left hand button of the mouse is pressed within the area to be filled.
When a receiver receives the information in the format discussed above, it can display exactly the same graphics as at the transmitter where the graphics are generated. It should be noted that every workstation could act as a transmitter and a receiver. It was so designed that a workstation stays in the receiver state unless an action is being initiated by the user of the workstation. When the graphics initiating station has finished an action and transmission, the station will return to the receiver state.

When the EBS was developed, there was no broadcasting server available on the network. The only way to realise multipoint service is for the graphics originating workstation to transmit the same graphics data several times to the individual destinations. It means that one workstation has to set up and keep connections with several other workstations at the same time.

4.6 Contention Prevention

When the EBS was developed, the A-500 was using the single task operating system, the Arthur. The EBS application requires the workstation acting as both the graphics transmitter and receiver. There is a case when graphics data can be lost. Suppose two workstations A and B involved in a graphics communication. If B sends some data to A while A is busy in initiating an action itself, workstation B will get an "other station busy" message and the data it sent will be lost.

A partial solution to this problem is to allow only one station to initiate an action at a time. When one workstation wants to start an action the station should send a signal to other workstations to stop them starting an action. The stations are re-enabled after they have received a block of graphics data. This method prevents the data loss but it is troublesome to both the programmer and the users.

This contention problem can be solved if a multitasking operating system or two processors are used. The use of a BBC-MASTER microcomputer as the bridge between the A-500 and the CR seems clumsy and wasteful, but by making use of the MASTER the contention problem can be solved. It can be done as follows: the reception request is always made outstanding in the MASTER. When the data come in, they are received and buffered in the MASTER. If the A-500 is not busy initiating graphics, the data are passed to the A-500 to be displayed. If the A-500 is busy initiating graphics, the graphics data are transmitted first before the buffered
data in the MASTER are passed to the A-500 to be displayed. In this way data loss can be prevented.

4.7 The Software Structure

The individual aspects of the EBS are discussed in the previous sections. This section gives an overview of the software structure.

There are two versions of the software. One version uses a graphics tablet as the input device and the other uses the mouse. Apart from the difference in the way the coordinates are gathered, another difference lies in the user interface. The interface of the former version uses a question-and-answer method for the user to control the system. The user uses the keyboard to answer the questions prompted. The questions are like "What is the receiver's name?" and "What is the size of the graphics window?" etc. The latter version uses the WIMP system. The user communicates with the system interactively by moving and clicking the mouse.

The program consists of a number of layers. The lower layers are concerned with the implementation of SSP and OPEN/OPENACK protocol. The upper layers are concerned with the graphics generation, transmission and reception. The layer structure is shown in Fig.4.3. Layer 1 and layer 2 are written in assembly code. Others are written in BASIC.

<table>
<thead>
<tr>
<th>Layer 7 : user interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer 6 : graphics generating and displaying</td>
</tr>
<tr>
<td>Layer 5 : graphics transmission and reception</td>
</tr>
<tr>
<td>Layer 4 : implementation of protocols</td>
</tr>
<tr>
<td>Layer 3 : BBD assembly</td>
</tr>
<tr>
<td>Layer 2 : interface between MASTER and A-500</td>
</tr>
<tr>
<td>Layer 1 : ring driver</td>
</tr>
</tbody>
</table>

Fig. 4.3 The Program Structure
Layer 1

The layer 1 code is the 6502 assembly code implementation of the Basic Block Protocol. It sits in the BBC-Master microcomputer in ROM or side RAM. Port management and basic block transmission and reception routines are provided in this layer. The A-500 calls this code via a Tuber driver code (layer 2 program).

The BASIC program makes use of this code through the use of a control block called a Basic Block Descriptor (described in layer 3). The Ring Driver is entered by making an OSWORD call with a value in the accumulator A not used by the MOS (Machine Operating System) or other ROMs. The MOS polls each of the ROMs in turn for a claimant service routine, and control can thereby be passed to the Driver. On entry, the X and Y registers point to the Basic Block Descriptor. If the call succeeds, a code of 0 is returned. Otherwise, other codes are returned. For details see [Girling 82].

Layer 2

As mentioned before the A-500 is connected to the network via a BBC-MASTER microcomputer. If a graphics tablet is used as the input device it is also connected to the MASTER. Layer 2 provides the interface between the MASTER and the A-500. When the A-500 needs to access the Ring Driver or get the coordinates from the graphics tablet, a Tube management routine is called. Tube is the name given to the connector through which the A-500 is connected to the MASTER.

This Tube management routine makes use of some 6502 assembly code which copies data between host and parasite (A-500) and vice-versa.

Layer 3

This layer sets up the Basic Block Descriptors (BBD) required by the Ring Driver. The user requests for basic block transmission and reception and port allocation are passed to the Ring Driver via the BBD. The BBD format is shown in Fig.4.4.

- PORT - either the port on which a reception is expected or the port to which a block is to be sent;

- STATION - the station address from which a reception is expected or to which a request is to be sent;
<TIMEOUT>- the time for which a request is to remain active (a value of 255 means forever);

<ACTION>- specifies the required Ring Driver routine. Bits 0, 1 and 2 specify the number of data buffers;

<FLAGS>- bits 0 and 1 indicate the Basic Block type. When bit 3 is set it means a user event is generated on reception. After submitting a request for reception, a Basic Block Identification (BBI) is returned in this field;

<ROUTINE>- address of the parasite routine to be run on reception;

<BUFFER>- address of the buffer to/from which data is to be transferred;

<MAXLEN>- maximum allowable size of the buffer;

<LEN>- actual length of the buffer.

Layer 4

The single shot protocol and OPEN/OPENACK protocol are implemented in this layer. Layer 3 routines are marshaled to:

-- allocate a reply port in the local station

-- submit a request for reception from the specified remote station

-- transmit a basic block to the remote station

-- poll for reception of a basic block

<table>
<thead>
<tr>
<th>byte 1</th>
<th>byte 2</th>
<th>byte 3</th>
<th>byte 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORT</td>
<td>STATION</td>
<td>TIMEOUT</td>
<td></td>
</tr>
<tr>
<td>ACTION</td>
<td>FLAGS</td>
<td>ROUTINE</td>
<td></td>
</tr>
<tr>
<td>BUFFER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXLEN</td>
<td></td>
<td>LEN</td>
<td></td>
</tr>
</tbody>
</table>

Fig 4.4 Format of BBD
Different headers are formed for an SSPREQ and an OPEN blocks and by using the above routines SSPREQ and OPEN are sent. A basic block received is decoded and the appropriate action will be taken.

Nameserver lookup service is also implemented in this layer. An SSPREQ is formed with a service name included, and the SSPREPLY is decoded to obtain the required station address, port number and function code.

Layer 5

Graphics data are transmitted or received over the connection set up by using the layer 5 routines. The data to be transmitted are generated in layer 6 and the data received are passed to layer 6 to be decoded and displayed.

Layer 6

Graphics generated by the user are displayed locally and the data block is formed for layer 5 to transmit. If data are received from layer 5, they are decoded and displayed in this layer.

The program is responsible for both transmission and reception. When there is no input from either the user or the network, the program stays in a waiting state, ready for data reception. This state can be changed by one of two events. One is when the user enters graphics information, and the other is when graphics data are received from the network. When the former event occurs, the graphics can be generated and transmitted. After one graphics action finishes, the program goes back to the waiting state. If the latter event happens, the data received are decoded and displayed and then the program also returns to the waiting state.

Layer 7

This layer provides a user interface. The connection setup information and graphics window information (size and position) are entered in this layer. If the graphics tablet is used, the information is entered from the keyboard. If the mouse is used, the user interface based on the WIMP system is used. The user can enter information by moving and clicking the mouse.
4.8 The CR Framestore as the Blackboard Receiver

So far, the LUT CR framestore has not been mentioned as the EBS receiver although it is capable of displaying graphics. Since it is difficult to enter graphics from the framestore but it can receive graphics data from the network and display the graphics, the framestore is used as the graphics receiver.

The framestore display is divided into several, typically four, windows. One of them displays the graphics received from participating workstations. If one of the windows displays a still video image, graphics can be drawn over the still image.

A program has been written which sits in the framestore and is used to receive graphics data, decode them and display the graphics on the screen attached to the framestore at the position and size selected by the workstation. The program is combined with the image transmission and reception program in the framestore.

The program consists of two parts, one part was written in Pascal, the other in 8086 assembly language.

The first part, the Pascal program, is mainly concerned with general functions. These functions are setting up a channel descriptor based on the information sent by a transmitter in the OPEN request, sending the OPENACK, and decoding the data received. If the data received are for graphics, they are decoded and the appropriate assembly language procedures are called to display the graphics.

The second part, the assembly program, communicates with the GDP (Graphics Display Processor) in the framestore which displays the graphics. Since the GDP can only draw vectors, the procedures to draw circles and to fill an area are complex.
4.9 A Note on the ALVEY'87 Exhibition

In July 1987 Unison participated in the ALVEY'87 Exhibition in Manchester. Two electronic offices were set up, one in Loughborough University of Technology, and the other in the exhibition hall of UMIST (University Manchester Institute of Science and Technology). People in these two offices communicated effectively with slow scan video, voice and EBS graphics simultaneously. The exhibition was very successful and attracted many visitors. During the exhibition the Unison EBS was tested and attracted much interest.
Chapter 5
Investigative Study of Packet Video Communication

5.1 Introduction

This chapter presents some background knowledge on packet video communication, followed by a general description of the design ideas of the Unison video codec and the provision of video service over the Unison network.

In order to transfer a video signal across a packet switched network such as an ATM network a virtual circuit is first established. The digitised coded video signal is then segmented into fixed-length cells/packets and a header is added to each. This header includes an identifier that distinguishes this video call from other calls currently being transferred through the network. The packet is then sent into the network to thread its own way (using its identifier) through the network along its predetermined path; at each switching node, the header is read and then the cell is forwarded to the appropriate output link. At the destination, the cells are gathered, the headers are stripped off, the video signal and timing are reconstructed, and finally images are displayed.

There are many advantages in transmitting video signals in packetized form. Firstly, future high speed networks should be very flexible, most probably being a dynamic packet switched network. Packet video can take advantage of the flexibility of this kind of network. Secondly, video in packetized form can be easily integrated with other kind of traffic into one network. Thirdly, a video source generates a variable
information rate, and efficient coding usually results in a variable bit rate output. A fast packet mode network can efficiently transport such a bit stream.

It is well known that to transmit digital video without compression needs a very wide bandwidth. For example, to transmit colour video of normal TV quality needs around 100 Mbps (Megabits per second). The present digital network cannot support that much bandwidth economically. Thus it is necessary to compress video data before transmission using some kind of compression methods. Several common video compression schemes are reviewed in section 5.2. Compressed video data are more sensitive to packet errors and packet losses. It is necessary to provide some kind of packet error and loss detection and recovery mechanism in order to guarantee a certain level of quality of service. Some common error detection and recovery mechanisms are briefly described in section 5.3. Video conferencing is a hot topic nowadays. Section 5.4 gives a general discussion on this topic. To make video conference practical standardization of video coding is essential. Section 5.4 also discusses the standardization issue.

The last section of this chapter gives some design ideas of the Unison video codec and video service.

5.2 Video Coding Schemes

Video signals have a lot of spatial and temporal redundancy. Spatial redundancy exists among pixels within a frame. Adjacent pixels are highly correlated. Temporal redundancy exists between frames. Usually one frame does not change very much from the previous frame, at least the background of the image does not change. In most cases, the changed part is caused by an object moving around in the frame. Thus adjacent frames are also highly correlated. All video compression coding schemes are designed to reduce the spatial and temporal redundancy. The compression rate varies with different coding schemes. This section reviews several of the most common coding schemes.

5.2.1 Pulse Code Modulation (PCM)

PCM is just the linear digitization of analogue video signals. The video signal is sampled generally at the Nyquist rate, then quantized using a sufficient number of quantization levels and coded in binary form for transmission. PCM coding is
inefficient because (a) it ignores the spatial and temporal dependency among elements and (b) it treats all quantized amplitude levels as equally likely. Actually, the other coding schemes discussed in the following subsections are designed to remove part of the redundancy in the PCM data.

5.2.2 Differential Pulse Code Modulation (DPCM)

In DPCM systems [Musmann 79], or predictive coding systems, the amplitude of each element is predicted on the basis of the history of previously scanned elements. The predicted estimate is then subtracted from the actual element amplitude and the difference signal is quantized, coded and transmitted.

DPCM coding techniques make use of the picture signal statistics to reduce the bit rate. Video signals are highly correlated, both spatially and temporally. Correlation, or linear statistical dependence, indicates that a linear prediction of sample values based on the values of the neighbouring elements will result in prediction errors that have a smaller variance than the original element. Owing to the smaller variance of the error signal to be quantized, the number of quantization levels can be reduced, resulting in a requirement for fewer bits per element than for a PCM system of the same SNR (Signal to Noise Ratio).

One simple form of DPCM video coding is the conditional replenishment. In such a system, the predicted estimates are compared with the actual element values, if the difference is small enough (below a predefined threshold), that pixel is not transmitted. Pixels with difference greater than the threshold are transmitted. This scheme is relatively easy to implement and also quite efficient.

5.2.3 Transform Coding

In transform coding systems, a unitary transform is taken over an entire image, or repeatedly over subpictures called blocks. For this purpose, Fourier, sine, cosine, Hadamard, Haar, Slant and Karhunen-Loeve transforms have been extensively utilized [Clarke 84].

The primary purpose of these methods is to convert the statistically dependent pixel intensities into 'more independent' coefficients by the use of a unitary transformation. The idea of the above transforms is to concentrate as much of the
image energy into as few transform coefficients as possible. By only sending those coefficients that contain most of the image energy, considerable compression can be achieved.

Among the transform codings, DCT (Discrete Cosine Transform) is very efficient and has some fast computation algorithms. For this reason, DCT is widely used.

5.2.4 Vector Quantization [Gray 84]

In the coding schemes discussed above, the actual quantization or coding is done on scalars, e.g., on individual real-value samples of waveforms or pixels of images. Transform coding does it by first taking the block transform of a block of pixels and then individually coding the transform coefficients. Predictive coding does it by quantizing an error term formed as the difference between the new sample and a prediction of the new sample based on past coded outputs.

A fundamental result of Shannon’s rate-distortion theory, the branch of information theory devoted to data compression, is that better performance can always be achieved by coding vector (a block of pixels) instead of scalars.

This theory had a limited impact on actual system design because 1) the Shannon theory does not provide constructive design techniques for vector coders and 2) traditional scalar coders often yield satisfactory performance with enough adaptation and fine tuning. Only recently vector quantization has been much considered [Yamaguchi 85, Ramamurthi 86].

The basic idea of vector quantization is that the encoder assigns to each input vectors, or block of pixels, a number out of a number set according to a certain criterion, and only that number is transmitted instead of the block of pixel values. For example, if a 32 bit number in a number set is assigned to represent an 8*8 pixel block, 0.5 bits/pixel is needed to transmit that block of pixels. At the receiver, the 8*8 pixel block is recovered from that 32 bit number according to the pre-defined criterion.

For details about vector quantization, see [Gray 84].

Vector quantization is very efficient in the sense that it compresses video data into very low bit rate. But it is difficult to implement in practice.
5.2.5 Adaptive Coding

All the above codings could be made adaptive by adjusting the coding parameters according to the video data statistics during the coding process. As a result of the adaptive coding more video data redundancy can be removed.

5.2.6 Statistics Coding

The most commonly used statistics coding is Huffman coding. It assigns fewer bits to the values that appear more often and more bits to the values that appear less often. It is usually used in combination with other coding schemes.

5.2.7 Practical Coding Schemes

In practice, the coding algorithms discussed above are used in combination. Usually, several coding algorithms are implemented in one codec.

The following basic coding schemes [Santamaki 86, Kuroda 85, Nicol 83] are widely used:
- spatial subsampling
- temporal subsampling
- detection of changed picture areas
- coding of the changed areas

Spatial subsampling can be implemented in two ways. The first is that the decoder displays only the pixels received. This means that the image displayed by the decoder is smaller than the original image before the subsampling. The second way is that the decoder interpolates the pixels dropped by the subsampling process, based on other pixel values received. This coding scheme is easy to implement and is quite efficient. For example, a subsampling ratio of 2 in both horizontal and vertical directions reduces the data rate four times.

Temporal subsampling reduces the update rate of images. Usually update rates of 8 to 10 frames/sec are needed to produce a reasonable motion of eyes, lips and other relevant object areas [Santamaki 86].
The simplest way to detect changed areas is to compare corresponding parts of adjacent frames. If one part in the present frame is quite different from the corresponding part in the previous frame, it is deemed to have changed. A more efficient way is to use motion compensation techniques [Srinivansan 85, Sabri 84].

The changed areas can be coded using DPCM, DCT or vector quantization.

### 5.3 Error Detection and Recovery

In video conferencing codecs the image is coded in such a way as to reduce its inherent redundancy, in order to obtain a lower output bit rate. The video data after compression are more sensitive to transmission errors. There are two types of errors which might occur during the transmission. One is bit error, that is, some bits are corrupted due to noise in the transmission network. The other is packet loss due to traffic congestion at some point in the transmission network. These transmission errors affect the quality of the displayed picture. One effect might be that the video sync signal is destroyed such that the decoder cannot display the picture properly. Another effect might be the propagation of errors over a long period of image reconstruction. These degenerative phenomena vary with the different coding techniques and the amount of bit rate reduction: in any case the image quality can be very poor. Thus in practical codecs, some sort of error detection and recovery mechanisms are needed to compensate part or all of transmission errors. Some common error detection and recovery methods are discussed below.

**Forward Error Correction (FEC) [Freer 88, Ambrosio 85]**

Error detection and correction (EDC) codes, such as Hamming codes, may be added to the transmitted data. These codes contain sufficient redundant information to detect several corrupted bits and in most cases to correct one or more bits. This technique of detecting and correcting errors without retransmission is called forward error correction.

This method can correct up to several error bits, depending on the complexity of the code, but it cannot detect and recover lost packets.
**Cyclic Refresh Method** [Ohta 88, Nicol 83]

In the cyclic refresh method, the refresh mode coding is executed periodically. As the refresh mode coding, an intraframe coding which does not use the correlation between adjacent frames (eg. PCM, intraframe prediction or orthogonal transform coding) is applied. Error propagation, if there is any, between frames is stopped cyclically. This method is usually used in interframe coding systems.

**Demand Retransmission Method**

In demand retransmission method, a coder sends the packet sequence number, parity or CRC (Cyclic Redundancy Check) to the decoder along with the video data packet. If the decoder detects any packet loss or bit error from the sequence number, parity or CRC, it sends a retransmission request to the coder. Upon receiving the retransmission request, the coder sends the packet again.

**Layered Coding** [Nomura 88]

In a layered coding system, the video data are coded into two streams, high priority and low priority data. The high priority data are essential to reconstruct a basic image. The low priority data are used to improve the image quality of the basic image. Transmission networks should distinguish the priorities of the data. They should keep the packet loss rate of the high priority data as low as possible. Packet loss on the low priority data is allowed if the network cannot handle all the traffic. The loss of the low priority data should not cause much annoying effect on the reconstructed image.

### 5.4 Video Conferencing and Standardization

Video conferences are a special type of teleconferences. Two or more groups of people meet at different places and are in contact via a telecommunication link. Therefore, a desirable feature is a motion video transmission. This is like bringing the participants, often sitting hundreds of miles away, to one conference table. This new kind of communication, the videoconferencing, permits a conference to be held without the time-consuming, tiring and expensive business trips. It is primarily suited for frequent business conferences. Of course, video conferences cannot replace conventional conferences in which personal contact between the partners is
necessary for success. On the other hand, conferences are now possible which otherwise would have been canceled because of the external circumstances.

Apart from motion video service, a videoconference system usually provides still image, voice, text, graphics and data services. The traffic of all these services is transmitted over the same transmission network. The Unison office is aimed at supporting videoconference services, although individual service can be used as its own.

At the moment, video conferencing is available on a reservation basis only at a limited number of locations. This is limited by the cost of the codecs and unavailability of a broadband ISDN. But this situation is changing very rapidly. Within the next five years, office-to-office video conference might be a reality with the camera and the codec being two of the options available to the office workstation [Douglas 89].

To hold videoconference among several different sites, often in different countries, the videoconference systems used by the participants must be compatible. To ensure compatibility, an international standard for these systems is essential. There are two aspects to the standard. One concerns the codecs and the other the transmission network. These standards are actively pursued by international standard bodies, such as the CCITT.

There are two major problems in the standardization of the codecs. One is that there are several video standards in the world, such as PAL, SECAM and NTSC. The other is concerned with the video data compression. The first problem might be solved by including a pre-processing system in the codec that produces a common intermediate video format which is used for processing and coding purpose. Then at the receiver side, the decoder can display pictures in any standard from the common intermediate video format.

As regard to coding, a hybrid coding scheme might be used. Coders identify which parts of the picture have changed significantly frame-by-frame and after compression only transmit these parts to the receiver. The changed parts are coded by the use of motion compensated prediction combined with DCT.
Only video service standardization has been discussed above. Other videoconference services, such as voice, graphics and text are also being standardized.

Concerning network compatibility, the problem occurs in the constitution of international 'digital pipes', for example between 2048 and 1544 kbit/s networks or 64 and 56 kbit/s networks. It may be solved by the use of network converters. The transmission framing structure within a network also needs to be decided. A frame structure of m*64 kbit/s and n*384 kbit/s might be used for videoconferencing.

5.5 Design Ideas for CFR Video Codec

This section first discusses the need for a new video codec for the Unison project. Design guidelines for the CFR video codec and video service over the Unison network are then described.

5.5.1 Reasons for Designing a CFR Video Codec

Video codecs already exist which work on the CR-based network[Lee 86, Brendan 87]. But it is slow and there is no compression scheme implemented. It cannot make use of the large bandwidth of the network efficiently. Furthermore, the CR video codec uses the multibus interface. A lot of work is needed to make it work on the CFR.

Commercial products are available but they are complex and thus very expensive. They are not suitable to work on the Unison experimental network because a) the output bit rates of the commercial video codecs are usually fixed. It is difficult to adjust output bit rate to suit the network. b) There is no interface available between commercial codecs and the CFR. To design the interface would require a lot of efforts.

In view of the above reasons, it was decided to design and develop a new video codec using up-to-date technology.
Personnel management practices in Kuwait libraries.

Ph.D., Loughborough - 43-0001

Checkland's soft systems methodology is adapted to investigate and suggest improvements to personnel management in Kuwait libraries. An initial study investigating relevant elements of personnel management in Kuwait, such as motivation, communication, etc., was conducted primarily by interviewing. A model was constructed on this basis to deduce relevant important issues, such as library services and motivation of library staff. These issues were further investigated in a second survey again primarily by interviewing. Because of the Gulf War, a third interview survey then took place to update data and to identify important changes regarding library management.

Four activity-based models were then constructed to determine factors relating to the improvement of personnel management in Kuwait libraries and as guides for data analysis, as follows:

- a system of increasing the supply of competent information workers;
- a system to enhance communication inside and outside the library;
- a system which meets users' needs and encourages the use of the library;
- a system to enhance staff motivation.

The conclusion examines possible solutions regarding personnel management problems in Kuwait libraries. In addition, the value of Checkland's soft systems methodology for this kind of analysis is examined.
There is currently much interest in Integrated Services Systems which handle within one network or architecture a mixture of data, graphics, video and voice, although most of the work to date has concentrated on the voice and data services. This thesis is concerned with the provision of graphics and video services alongside the voice and data services over a Wide Area Network (WAN). The network concerned in this thesis is the Unison network, which comprises several Local Area Networks (LANs) interconnected by a primary rate (2 Mb/s) Integrated Service Digital Link. The first part of the thesis deals with the design and development of an Electronic Blackboard system suitable for use over the network. The system comprises workstations sitting on the network, with a mouse or a graph pad as the control and input device to each workstation. Inputs from a user are displayed on his own workstation monitor and also displayed on the workstation monitor of his partner elsewhere on the network. The system provides the participants, say of a videoconference, with a tool to facilitate exchange of text (entered from keyboards), graphics and free-hand drawings (entered from the mouse or the graph pad). The second part of the thesis is concerned with the provision of a video service. A versatile video codec (including a transmitter and a receiver) has been designed and implemented. The video codec is based on Inmos Transputers which have the characteristics of fast speed and easy interconnection. A set of video protocols has also been developed to provide multi-point video service over the Unison network using the video codecs, enabling a transmitter to send images to more than one receiver and a receiver to receive and display images on one monitor from more than one transmitter. These protocols are implemented in a concurrent language called OCCAM. Colour and monochrome image transfer is selectable. Two main services, freeze image transmission and slow scan image transmission, are available. To reduce the amount of bandwidth required to transmit images, two simple image compression schemes, namely subsampling and conditional replenishment, have been implemented in software. The operation control of the video codecs is by means of a user interface on workstations connected to the network. Experiments on graphics and image transmission along with voice and computer data have been carried out successfully over the Unison network.
5.5.2 Design Guidelines for the CFR Video Codec

Considering the Unison requirement and the time restriction, the CFR video codec should have the following features:

(1) It should use a fast processor. Preferably the processor should be easy to be interfaced to the network.

(2) DMA (Direct Memory Access) should be used so that access to the memory is fast and flexible.

(3) At least the spatial subsampling and temporal subsampling schemes should be implemented.

(4) It should be flexible in terms of resolution and update rate so that it can be easily fitted into the network.

(5) Since the network is an experimental ATM network, it is unavoidable that there will be some packet losses when it is overloaded. The codec should be able to tolerate some packet losses.

(6) It should provide some means for the workstation to control it.

(7) It should have a simple split-screen facility so that several pictures from different sources can be displayed on one monitor.

Fast Processor

Since Transputers [Inmos 89] have features of fast operation and easy interconnection and there were interfaces available between transputers and the CFR, it was decided to adopt it in the CFR video codec.

Direct Memory Access

DMA is not only fast but also flexible. Spatial subsampling can be done easily with DMA. The transputer has a very large memory space, the video memory can be easily mapped into the transputer memory space.
Coding

Spatial and temporal subsampling are simple and fairly effective in reducing the bit rate, although they do not exploit the intraframe and interframe redundancy fully. Among the coding methods mentioned in Section 5.3, conditional replenishment, transform coding and vector quantization are very efficient. But to implement them in hardware would take quite a long time. Unison required video codecs to run on the CFR-based network as soon as possible. So it was decided that no hardware coding would be implemented, but conditional replenishment could be implemented in software.

It should be noted that the main interest of Unison is not to build a high compression rate video codec, although it would be preferable to have high speed video. The main interest is to put video of reasonable speed onto the network and to study some characteristics related to the video service and the integration between video and other traffic, and meanwhile to assess the performance of the network in handling multi-media traffic.

Flexibility

The Unison network is an experimental network. The exchange is still under development and its performances (delay, bandwidth) are not known. It is required that the bit rate transmitted over the network be easy to adjust. It is better to transmit video with slower update rate and no or low packet loss than to transmit video with faster update rate and large packet losses. The CFR video codec should be flexible in terms of picture resolution and update rate so that its output bit rate can be accommodated on the network.

Packet Loss Tolerance

To tolerate packet loss means that the receiver should maintain synchronization even though there are packet losses. To realize that a pixel address should be sent with each packet so that the data in the packet can be put into the right position in the video memory (and on the screen). Packet loss only results in a small area of the screen not being updated but displayed with the corresponding data from the previous frame. Some mechanism should also be provided to adjust the transmission rate according to the network loading.
Workstation Control

One of the main features of the Unison Office is that all the office components should be controlled by workstations. The video codec should provide some channels to talk to the workstations over the network. It should be noted that the office components are separate entities and are separately interfaced to the network. The only way they talk to each other is via the network.

Split-Screen Facility

Multi-point communication involves more than two parties. The transmitter can transmit information to multi-point in two ways. One way is to transmit the same information several times to several different destinations. The other way is to transmit the information to an "agent" once, the agent then broadcasts the information for the transmitter. Usually the former is slower and needs more bandwidth. The latter is effective, but needs more complex network management. The CFR video coder (transmitter) should be programmable to do either. To the receivers the information transmitted by these two methods is the same. But the receiver side can display the information differently. One simple way is that one receiver accepts the information from only one source. Several receivers are required to accept the information from several sites. It is expensive. An alternative is that one receiver accepts information from several sources and display the information on different windows on one monitor. This is the so called split-screen technique. The CFR video decoder (receiver) should have this split-screen ability.

The next two chapters detail the design and implementation of the video codec and video protocols based on the above guidelines. Chapter 6 concentrates on the video codec hardware and Chapter 7 on video protocols.
Chapter 6

The Video Codec

This chapter is devoted to a description of the principle of operation and implementation of the video codec (coder and decoder). The coder will be discussed first, followed by a description of the decoder. The general applications of the coder and the decoder is also presented. Finally, the interface between the codec and the CFR is discussed.

6.1 The Video Coder

The functional diagram of the video coder is shown in Fig 6.1. The output of the camera is a PAL composite signal. This signal is decoded into RED, GREEN and BLUE components, and composite sync signal by the PAL decoder. These analogue R, G and B signals (or their transformed signals: luminance signal $Y$ and chrominance $U$, $V$) are converted into digital data by the ADC (Analogue to Digital Converter). The grabber generates pixel addresses according to the sync signal and pixel clock from the ADC, and stores the digital video data in memory. The

![Fig.6.1 Functional Diagram of the Video Coder](image-url)
controller is used to control the operation of the grabber and to process and transmit the data stored in the grabber.

The design principle and implementation of the ADC, the grabber and the controller will be detailed in the following sub-sections. The grabber and the controller can be implemented either on the same board or on separate boards. Both versions were implemented and worked well. For simplicity, the former version is described only. The board consisting of the grabber and the controller is called the video transmitter board.

6.1.1 The Analogue to Digital Converter (ADC) Board

Fig.6.2 shows the functional diagram of the ADC board. This board has three functions: generation of pixel clock, separation of the composite sync signal into line sync (composite sync), field sync and odd/even field indicator signals, and conversion of analogue video signals into digital signals. In the diagram, only RED, GREEN and BLUE components are shown. For better picture quality and some other reasons (see later), R, G and B components can be transformed into a luminance (Y) and chrominance components (U and V). Then, Y, U and V may be converted into digital data.

![Functional Diagram of the ADC Board](image)

Fig.6.2 Functional Diagram of the ADC Board
6.1.1.1 The Pixel Clock Generator

The pixel clock is used by the ADC's as the sampling clock and by the grabber to generate pixel addresses. There are two attributes to be considered, the frequency of the pixel clock and phase relationship between the pixel clock and the composite sync signal.

The frequency of the pixel clock affects picture resolution (pixels per line), memory speed required and the amount of memory required to store a frame of video data. The higher the frequency, the higher the picture resolution, provided that the resolution of the display monitor is sufficiently high. A pixel clock of 10 MHz can produce reasonable picture resolution and suits most monitors in common use.

The higher the clock frequency, the higher the video memory speed required. Static memories having access speeds of 100 nanoseconds are currently widely available. In the past when memories of this speed were very expensive, a buffer was used to store data of several pixels, say four pixels. Then after every four pixels' time the data of these four pixels were transferred into the main frame memory. By using this method, the memory used to store video data can be four times slower, but this method did add some hardware complexity and inflexibility.

![Fig.6.3 Vertical Zigzag Effect](image-url)
The higher the frequency, the more pixels there are on one line and the more memory is required to store one frame of image. Considering the effect of the frequency on the picture resolution and the memory speed required, a clock frequency of 10 MHz (100 nanoseconds per cycle) is sensible. Since the effective time of each line is 52 µs, it was decided 512 samples (pixels) per line are stored in the video memory. The decision is made because of two reasons. The first one is to make use of the picture width efficiently. 512 samples take 51.2 µs which is over 98% of 52 µs. The second reason is that 512 is a whole power of 2. It is easier to manipulate data block of n-th power of 2 (n being an integer). Also, memory space is usually configured as m-th power of 2 (m being an integer), so the memory space can be used more effectively.

As to phase relationship between the pixel clock and the sync clock, the pixel clock has to be phase locked to the sync signal. Otherwise the position of the starting

---

![Diagram of pixel clock generator and waveforms](image)

(a)

(b)

Fig.6.4 The pixel clock generator and the waveforms
pixel captured by the grabber on each line varies randomly within one pixel space. Thus the vertical edge of an captured image is a zigzag line. This is illustrated in Fig. 6.3.

Two methods have been considered to generate the pixel clock required. The first one is to use a crystal module of higher frequency, say 40 MHz. This clock is divided by 4 to get the 10 MHz clock required. By doing this the maximum time difference, relative to the start point of the line, between corresponding pixels on each line is reduced to 25 nanoseconds instead of 100 nanoseconds. This method can not entirely eliminate the vertical zigzag effect, but this method does reduce the zigzag effect to one fourth.

The second method is to use some kind of oscillator which is controlled by the incoming sync signal. One implementation is shown in Fig. 6.4(a). When line sync is low (blank period), the oscillator ceases to operate. The output remains high. When line sync is high (effective line period), the circuit oscillates. It produces a square wave the frequency of which is determined by $1/(1.38*R*C)$. The time $T$ from the sync going high to pixel clock first time going low is fixed to $0.69*R*C$. This means that pixel positions are fixed relative to the line sync. So if the line sync is stable, there is not any zigzag effect. $R$ is adjustable so that the correct frequency required can be obtained.

The second method is implemented on the ADC board.

### 6.1.1.2 The Sync Separator

The function of the sync separator is to obtain field sync and an odd/even field indicator signal from the composite sync. The composite sync and field sync are used by the grabber to generate pixel addresses. The odd/even field indicator tells the grabber the field identity, odd or even. Although the grabber can grab frames in real time, it is not operated in continuous grabbing mode. When one frame has been captured, the data of this frame are processed and transmitted. After that the grabber is ready to capture another frame. When the grabber is ready to capture another frame, there is uncertainty as to whether the odd field sync comes first or the even field sync comes first. Pixel addresses are generated sequentially. If the odd field is captured first in one frame and the even field is captured first in another frame, the pictures recovered from these two frames are slightly different.
(a) an image captured in the normal order

(b) the same image as (a) but captured in the wrong order (odd lines becomes even lines)

Fig.6.5 Comparison of images captured in the different order

even though the original images are the same. An exaggerated illustration is shown in Fig.6.5. In practice, although the problem is not so serious, the edge of a recovered image could be blurred if the fields are captured in the wrong order. This problem also affects the compression rate if an interframe coding scheme is used since even the still areas of a scene are changed slightly between frames if the frames are captured in the incorrect order. To solve this problem, it is necessary to

Fig.6.6 LM1881 Video Sync Separator
Chapter 6 The Video Codec

guarantee that the odd field is always captured before the even field using the odd/even field indicator signal.

There are several methods to realize the sync separation. The simplest way is to use an integrated circuit, LM1881 Video Sync Separator, as shown in Fig.6.6. The odd/even field indicator signal is high when a field is the odd field. It is low when a field is the even field.

6.1.1.3 The ADC

The PAL decoder outputs analogue Red, Green and Blue signals. These signals can be directly converted to digital data, or can be transformed into luminance (Y) and chrominance (U, V) signals, and then converted into digital data. These two methods are discussed respectively below.

R, G and B analogue signals can be directly converted to digital form. In the implementation of the conversion, there are two points to be considered. The first is how many bits one pixel should take. The second one is how these bits are to be assigned among the R, G and B components. In theory, the more bits one pixel takes, the better the quality of the picture recovered, but more bandwidth is required to transmit the picture. A good compromise is that each pixel takes 16 bits. Since the green component contributes more to the luminance signal than red and blue components, it is better to assign 6 bits to the green signal, and 5 bits each to the red and blue signals.

6-bit monolithic A/D Flash Converters, CA3300, are used in the implementation. The analogue signals are amplified to make full use of the converter input range before they are applied to the converters to increase the signal-to-noise ratio. For detailed circuitry, see Appendix 2.

The R, G and B analogue to digital conversions are simple, but it is difficult to do image compression coding with R, G and B components. If these components are converted to luminance (Y) and chrominance signals (U, V) as detailed below, it is much easier to do some compression coding with Y, U and V, such as conditional replenishment;

\[ Y = 0.299R + 0.587G + 0.114B, \]
\[ U = -0.147R - 0.289G + 0.437B \]

and
$V = 0.615R - 0.515G - 0.100B.$

Furthermore, since the human eye is less sensitive to $U$ and $V$ than to $Y$, $R$, $G$ and $B$ signals and less bits may be used for $U$ and $V$, YUV representation produces better picture quality than RGB representation if the same amount of bits are used to represent each pixel. Also, $Y$, $U$ and $V$ representation is more flexible. For example, to obtain a black and white picture, only the $Y$ signal needs to be considered.

The transformation from $R$, $G$, $B$ to $Y$, $U$, $V$ is performed in real time on the analogue signals. Assuming $R$, $G$, $B$ values in the range 0 to 0.7 volt (the standard for baseband components), then the maximum and the minimum values for the transformed components are:

- $Y$ : 0.000 to 0.700 volt
- $U$ : -0.306 to 0.306 volt and
- $V$ : -0.431 to 0.431 volt

Since $U$ and $V$ are bipolar signals, they are level shifted so that their minimum values are zero before they are applied to the converters. If $U$ and $V$ are 0 volt in analogue form before the level shifting, they are represented by 16 (10000 in binary) in digital form. For detailed circuitry, see Appendix 2.

One pixel still takes 16 bits in the YUV representation. Since the human eye is more sensitive to the luminance signal $Y$ than to chrominance signals $U$ and $V$, 6 bits are used to represent the $Y$ signal and 5 bits each are used to represent the $U$ and $V$ signals.

For flexibility, both RGB and YUV ADC's are implemented on the final PCB version. They are switchable depending on user requirements. It is arguable that since the YUV representation has advantages over the RGB representation, there is no point to use the RGB representation. But in practice, the $Y$, $U$ and $V$ ADCs are much more difficult to be implemented than $R$, $G$ and $B$ ADCs. The $R$, $G$ and $B$ ADCs are useful to test other parts of the video codec before the $Y$, $U$ and $V$ ADCs have been implemented.

The digitized video data bits for a pixel are arranged as follows:
6.1.1.4 Features of The ADC Board

The PCB version of the ADC is implemented on a double height, standard depth Eurocard. The board takes in standard R, G, B and PAL composite sync signals from four separate Phono connectors at the front edge of the board. YUV representation or RGB representation is selected by a switch on the board. The video signals are converted to digital data at a frequency of 10 MHz. The board outputs digital R/U (5 bits), G/Y (6 bits) and B/V (5 bits) as well as the 10 MHz pixel clock, composite sync, field sync and odd/even field indicator signal to the video transmitter board through a forty-way ribbon cable. The board is powered by

![Fig.6.7 Functional Diagram of the Grabber](image-url)
Chapter 6 The Video Codec

+/-12 volts and 5 volts through a rear connector which is VME bus compatible. For detailed pin connections, see Appendix 2.

6.1.2 The Grabber

Fig. 6.7 shows the functional diagram of the grabber. The grabber is controlled by the control processor. The memory, control register and status register are memory mapped to the controller processor allowing the processor direct access. Both the control register and the status register have 8 bits, although only 2 bits in the control register and 1 bit in status register are used. The two bits in the control register control grabbing enable/disable and the resolution. The grabbing enable/disable bit indicates whether video is to be grabbed or not; when it is high (1), the grabber is enabled, otherwise it is disabled. The resolution bit indicates whether to grab one field or one frame (2 fields). When it is high, one frame is grabbed, otherwise only one field is grabbed. The one bit in the status register indicates whether the grabber has finished grabbing a frame (or a field depending on the resolution bit). When it is high (1), it means the grabber has finished grabbing a frame, otherwise the grabbing is still in process. This bit is controlled by both the controller processor and the pixel address generator. When the processor enables the grabber, this bit is reset to 0. When the grabber has finished grabbing a frame, this bit is set to 1.

To capture a frame, the controller sets the resolution bit and enables grabbing. When the odd field sync comes the address generator starts generating pixel addresses sequentially and video data are written into the memory. Meanwhile, the controller checks the status register to see whether grabbing has finished. If grabbing has not finished, the controller is not allowed to access the memory unless the controller disables the grabbing by writing 0 into enable/disable bit. This usually does not happen. Normally the controller waits until the grabbing finishes. As soon as one frame (or field) has been grabbed, the grabber stops automatically. The controller then can access the memory. After the controller has finished processing and transmitting the data in the memory, the controller can enable the grabber again and the process repeats. The buffers are used to isolate the address generator bus and the controller bus so that only one of them can access the memory at a time.
Fig. 6.8 Operation flow chart of the address generator
The address generator and memory unit are described separately below.

6.1.2.1 The Address Generator

Basically, the pixel addresses are generated by counting the pixel clock on each line, the line number in each field and the field number in each frame. The combination of these three counts forms a unique address for each pixel.

If picture resolution is 512 lines by 512 pixels, an 18 bits address is needed. The least significant 9 bits are used to represent the pixel position on a line. The next 8 bits represent in which line in a field the pixel lies. The most significant bit indicates on which field the pixel is. The odd field is indicated by 0. The even field is indicated by 1. For example, if a pixel x is the 5th (0000101 in binary) pixel on the 100th (01100100 in binary) line of the odd field(0), the address of x is (0 01100100 000000101) in binary.

For several reasons, the pixel address generator does not generate addresses from the very beginning of a field and the very beginning of a line. The first reason is that there are line and field blank periods. During these periods the pixel address generator should not generate any address, that is, no data should be captured in these periods. The second reason is that there are 575 effective lines (excluding lines in the field blank period). Since only 512 lines are to be captured for a frame, it is better to capture the 512 lines in the middle of the 575 lines. Thus an offset of 32 lines vertically and an offset of 64 pixels horizontally are used.

Control logic is used to control when to start counting and when to stop counting. The operation flow chart to capture an odd field is illustrated in Fig.6.8. The controller processor writes 0 into the resolution bit of the control register to request the grabber to capture the odd field only, and writes 1 into the grabbing enable/disable bit to enable the grabber, and the status register is reset to 0. The grabber waits for the start of the next odd field. When the next odd field comes, the vertical offset counter counts the line sync to 32. After an offset of 32 lines, the line counter starts counting lines. Meanwhile, the horizontal offset counter counts the pixel clock to 64 from the beginning of each line, and then the pixel counter starts counting pixels to 512 on a line. The combination of contents of the line counter and pixel counter forms a pixel address. The line counter stops and resets when it reaches 256, and the grabber is disabled automatically and writes 1 into the
grabbing status register. When the controller processor detects this grabbing finish signal (1), the processor starts processing and transmitting the data of this captured field. The grabber is enabled again when the processor has finished the processing and transmission. For detailed circuitry, see the left part of the transmitter diagram on page 2 in Appendix 3.

6.1.2.2 The Frame Memory

Since high capacity and high speed static RAMs are available, to simplify hardware implementation, static RAMs are used instead of dynamic RAMs. The SRAMs used are HMS41664 from Hybrid Memory Products Limited, which is configured as 65536 by 16 bits at an access speed of 100ns.

The frame memory consists of four these chips. It can store a frame with 512 lines by 512 pixels and 16 bits per pixel. Fig.6.9 shows the schematic diagram of the frame memory.

Video data can be written into the frame memory under the control of the pixel address generator. The frame memory can also be accessed by the controller processor. But only one of these two processes can be active at one time. The data bus from the ADC, the address bus from the pixel address generator and the data and address buses from the controller processor are all buffered. The buffer output enables/disables are controlled by the grabber's status signal. The processor can access the frame memory only when the grabber has finished capturing one frame or by stopping the grabbing process by written 0 into the grabbing enable/disable bit of the control register.

Pixel addresses are generated sequentially. Pixels in the even field are stored in the higher memory page. Pixels in the odd field are stored in the memory from addresses 0 to (256*512)-1 sequentially. Pixels in the even field are stored in the memory from addresses 256*512 to (512*512)-1 as shown in Fig.6.10. Lines 1 to 256 relate to the odd field and lines 257-512 relate to the even field.

The frame memory is memory mapped to the controller processor allowing the processor easy access to any pixel in a frame. For example, it is very easy to do subsampling, or just transmit part of a frame.
(note: a0-a17 are from pixel address generator; d0-d15 are from the ADC board; A0-A17 and D0-D15 are from the controller board.)

Fig. 6.9. Schematic Diagram of the Frame Memory
6.1.3 The Controller

For flexibility, the controller should be based on a fast processor. Since the transputer has many advantages and the interface between the transputer and the CFR is already available, the transputer is used as the controller processor. The processor is also responsible for some simple data processing.

The transputer is a complete computer on a chip. There is a processor, a small amount of fast (50ns cycle), on-chip static RAM, four serial communication 'Links' (for external communication), and a programmable memory interface (which allows up to 4Gbytes of physical memory external to the transputer). The block diagram of the transputer is shown in Fig. 6.11.

The transputer, and the programming language Occam [Inmos 88, a], evolved together. Occam is designed to simplify the task of concurrent programming. Occam consists of three primitive processes, which are combined to create larger processes:

- \( v := e \) assign the value of expression \( e \) to variable \( v \);
- \( c ! e \) output expression \( e \) to channel \( c \);
- \( c ? v \) input a value from channel \( c \) into variable \( v \).

These primitives are combined to create processes using the Occam constructs:

- \( SEQ \) operate on the component processes in sequence

---

**Fig. 6.10. Relationship between pixel position and address**

<table>
<thead>
<tr>
<th></th>
<th>Pixel 1</th>
<th>Pixel 2</th>
<th>...</th>
<th>Pixel 512</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line 1</td>
<td>(0)</td>
<td>(1)</td>
<td></td>
<td>(511)</td>
</tr>
<tr>
<td>Line 2</td>
<td>(512)</td>
<td>(513)</td>
<td></td>
<td>(1023)</td>
</tr>
<tr>
<td>Line 256</td>
<td>(130560)</td>
<td>(130561)</td>
<td></td>
<td>(131071)</td>
</tr>
<tr>
<td>Line 257</td>
<td>(131072)</td>
<td>(131073)</td>
<td></td>
<td>(131583)</td>
</tr>
<tr>
<td>Line 512</td>
<td>(261632)</td>
<td>(261633)</td>
<td></td>
<td>(262143)</td>
</tr>
</tbody>
</table>
Chapter 6 The Video Codec

System Service

2 k bytes of on-chip RAM

MEMORY INTERFACE

Processor

ProcClockOut
notMemS0-4
notMemB0-3
notMemRd
notMemRf
MemWait
MemConfig

32 bits

32 bits

32 bits

32 bits

32 bits

Event

MemReq
MemGrnd
MemAD0-31

LinkIn0
LinkOut0
LinkIn1
LinkOut1
LinkIn2
LinkOut2
LinkIn3
LinkOut3
EventReq
EventAck

Fig. 6.11 T414 Transputer Functional Block Diagram
PAR operate on the component processes concurrently

ALT operate on the first process to become ready

IF and WHILE constructs are also available. Each construct is itself a process, and can be used inside other construct to create larger processes. Concurrent processes, which cannot use shared resources, communicate across occam channels. These channels are single direction, point to point connections between processes, and give synchronised message communication.

Concurrent programming evolved as it became clear that many programs could be split into a number of tasks which could be processed on independently, and use some form of message exchange for passing results, parameters, and synchronisation.

On standard sequential machines, implementing concurrence was solved by making the processor share its time between each task. This required a complex software 'Kernel' be written, which would control the switching of tasks (including itself) in and out of the processor, and handle the passing of messages. Switching the tasks frequently gives the user the impression that all of the processes are running simultaneously.

On transputers, the kernel has been implemented in hardware, giving a sub-microsecond task switch time, compared to a few milliseconds on software driven multi-tasking machines. Occam processes can be mapped onto a transputer,
which shares its time between them, or onto multiple transputers, each taking a subset of the processes. The Occam channels are mapped onto the transputer links for processes on separate processors. In this way, programs can be developed on a single transputer, and extended to multiple transputers as more performance is required.

The T414 transputer's links normally operate at 10MBAud, full duplex, and each link is capable of supporting two Occam channels, one into the transputer, and one out from the transputer. Each link is implemented as an autonomous DMA (Direct Memory Access) engine, and so can perform communications with external devices as background tasks to the processor with negligible performance degradation.

The functional diagram of the controller is shown in Fig.6.12. The controller itself can act as a single board computer, which can run Transputer Development System (TDS). The 1 Mbyte DRAM is used to run control and processing software. The bus driver demultiplexes the data and address lines of the transputer. Data lines D0-15, address lines A0-15, and the w/r signal are provided externally to the controller. The external grabber circuitry is memory mapped to the transputer by means of the data and address buses provided.

### 6.1.3.1 The Transputer

The transputer used in the controller board is the T414, a 32 bit processor, capable of 10 MIPS (Million Instructions Per Second). The 32-bit multiplexed address/data bus of the T414 allows up to 4 GBytes of directly addressable memory external to the transputer, as well as the 2 KBytes fast static RAM on board the transputer itself. The memory map of the transputer is signed, with the internal RAM starting

<table>
<thead>
<tr>
<th>Resources</th>
<th>AD31</th>
<th>AD30</th>
<th>AD29</th>
<th>AD22</th>
<th>AD21</th>
<th>AD20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic RAM</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Video Frame SRAM</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>grabbing control register</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>grabbing status register</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig.6.13 The Transputer Memory Map
at the most negative address (Hex 80000000 to Hex 800007FF). The external memory starts at Hex 80000800.

For the purpose of addressing the various resources of the coder, the transputer address space is divided into eight equal segments by decoding the three most significant address lines (AD29-AD31). Only two segments are used. One segment is used by the DRAM on the controller board and the other is used by the grabber. The segment used by the grabber is divided again into eight sub-segments. The various resources on the grabber are mapped into three sub-segments by decoding the address lines (AD20-AD22). The resources which are mapped are shown in Fig.6.13, together with the corresponding values on the address lines (AD20-AD22, AD29-AD31).

In terms of the current implementation of OCCAM 2, the resources could be declared as follows:

\[
\begin{align*}
\text{PLACE} & \quad \text{dynamicRAM} \quad \text{AT} \quad \$00000000:\n\text{PLACE} & \quad \text{frameSRAM} \quad \text{AT} \quad \$08000000:\n\text{PLACE} & \quad \text{control.register} \quad \text{AT} \quad \$08040000:\n\text{PLACE} & \quad \text{status.register} \quad \text{AT} \quad \$08080000:
\end{align*}
\]

Since the address lines (AD23-AD28) are not used by the grabber, the addresses of the video frame SRAM, the control register, and the status register declared above are not unique. But since all accesses of these memory mapped resources are software controlled and if the above declarations are always used, there should not be any addressing contention.

The memory interface of the T414 uses a 32-bit wide multiplexed address/data bus and its controller may be configured on reset, to suit a wide variety of different memory systems. For the video controller, the configuration of MemAD6 is used. The writing and reading timing diagrams for this configuration are shown in Fig.6.14 and Fig.6.15.

The transputer memory interface was designed so that the external DRAM can be used easily. The DRAM RAS (Row Address Strobe) and CAS (Column Address Strobe) can be obtained easily from notMemS0-S4. For an example of the use of these signals, see the subsection below.
Tstate | T1 | T1 | T2 | T3 | T3 | T4 | T5 | T5 | T6 | T7 | T1 | T1 |

MemADO-31

notMemRD

notMemS0

notMemS1

notMemS2

notMemS3

notMemS4

Read Data latched

Fig.6.14 Transputer Read Cycle Timing Diagram
Fig. 6.15 Transputer Write Cycle Timing Diagram
Fig. 6.16 The DRAM Unit Circuit Diagram
6.1.3.2 The Dynamic RAM

One megabyte dynamic RAM is provided for the transputer. Four DRAM modules, TM4256F8-12L, are used. Each module is organized as 262144 by 8 bits, and has row access time of 120 nanoseconds. The one megabytes memory is organized as shown in Fig.6.16. They are configured as 262144 by 32 bits, although individual bytes can be accessed.

IC1 and IC2 are used to latch row addresses and column addresses under the control of notMemS0 and notMemS2. IC3 decodes the AD29-AD31 address lines to divide the transputer memory space into eight equal segments, one of which is used by the DRAM. Four writing enable signals notWRB0, notWRB1, notWRB2 and notWRB3 are used so that any byte may be accessed individually as well as one word (four bytes) may be accessed at one time. Memory refresh is automatically performed by the transputer at an interval of 14 microseconds in this configuration.

The signals notMemS0-S4 are used for control purposes. The notMemS0 signal is used to latch row addresses and column addresses. The notMemS1 signal is used as RAS. The notMemS2 signal is used to switch from the row address enable to the column address enable, that is before the notMemS2 signal goes low the row address is enabled and after the notMemS2 signal goes low the column address is enabled. The notMemS3 signal is used as the CAS.

The refresh of the DRAMs is performed by the transputer. After every 14.4 μs, the transputer pulls the notMemRef signal low and outputs the row address to be refreshed. RAS is forced low and the memory cells in that row are refreshed.

6.1.3.3 The Grabber Interface (the bus driver)

The data and addresses are demultiplexed as shown in Fig.6.17. There are 21 address lines and 16 data lines which are used to access the resources on the grabber.

Again, the notMemS0 signal is used to latch addresses. To access memory used on the grabber board, 21 address lines are used. Two bidirectional buffers are provided for 16 data lines. The direction of these buffers is controlled by the w/r signal. The
Chapter 6 The Video Codec

Fig. 6.17 The Interface between the Controller and the Grabber
address latches and the data buffers are controlled by y5, the grabber access signal. They are enabled only when the controller processor accesses the memory segment assigned to the resources on the grabber.

6.1.3.4 Summary of the Video Transmitter Board

The video transmitter board consists of two main parts, the grabber and the controller. The grabber is capable of capturing frames with a resolution of 512 lines by 512 pixels and 16 bits per pixel in real time under the control of the controller. The controller is based on the transputer T414. The controller itself acts as a single board computer and can run TDS. The board takes in R/U (5 bits), G/Y (6 bits), B/V (5 bits) and 10 MHz pixel clock, the composite sync, the field sync and the odd/even field indicator from the ADC board through a forty-way ribbon cable. It is not important whether the video signals are represented by R, G, B or Y, U, V, unless some kind of software coding is carried out by the transputer. The board is interfaced to the outside world (the network) via four transputer links. The wire-wrap version was implemented on a double height, standard depth Eurocard. The board gets power (5 volts) from the rear connector which is VME bus compatible. For detailed pin connections, see Appendix 3.

6.1.4 Summary

The principle and implementation of the video coder have been detailed in this section. The coder consists of two boards, the ADC board, the transmitter board (the grabber and the controller). The grabber is capable of grabbing frames of 512 lines by 512 pixels, 16 bits per pixel, in real time under the control of the controller, although the coder is normally not used in this mode. It is used to capture a frame (or a field) and to process and transmit this frame, and then to capture a new frame.

The coder can be easily connected to the CFR via the transputer based CFR interface card, developed by CAMAN Consultants Ltd. For detailed information on the interface card see Section 6.4. The connection between the coder and the interface card is by means of transputer links, which provide a data transfer rate of about 3 Mbits per second. An image of 256 lines by 256 pixels and 16 bits per
pixel has 1 Mbit of information, so the coder can transmit a maximum of about three images of this size per second.

The transmission speed of the coder is limited, but the transmission speed is more significantly limited by the network interface and the network itself. Without complex hardware compression coding, this is the fastest frame update rate which may be achieved. Since the data is not driven out by some kind of rigid hardware, i.e. the transputer is used to control the grabber and to read and transmit the data, the transmission speed is limited, but the user is given considerably greater flexibility.
6.2 The Video Decoder

Fig 6.18 shows the functional diagram of the video decoder. It consists of three main functional units: the receiver controller, the display unit and the digital to analogue converter (DAC). The display unit consists of two main parts, the VRAM (dual-port video RAM) and the VSC (Video System Controller). The receiver controller receives data from the network and writes the data into the video memory in the display unit. At the same time, the data in the video memory are continuously read out to the DAC board in real time, 25 frames per second. The digital video data are converted into analogue signals and are displayed on a monitor.

The controller and the display unit are implemented on one board, which is referred to as a video receiver board in the following description. In the following sub-section, the video receiver board and the DAC board will be detailed respectively.

It is significant that in the video coder described in the previous section, the grabber captures a frame, then the coder controller transmits the video data from this frame. The grabber will not capture another frame until the controller has finished the transmission. In the video decoder, however, the decoder controller receives data and writes the data into the memory, at the same time, the monitor
has to be refreshed at 25 frames per second by the contents in the VRAM memory no matter whether the contents in the memory have changed or not. If ordinary RAMs are used as the video memories, it is difficult to control the memory operation and the controller has little time to access the memory. Thus dual-port video RAM is used as the video memory in the implementation of the video receiver board.

6.2.1 The Video Receiver Board

The function of the video receiver board is to receive data from the network, write the data into video memory, and display the data from the memory in real-time.

![Functional Diagram of the Video Board](image)

Fig.6.19 Functional Diagram of the Video Board

The diagram of the video board is shown in Fig.6.19. As shown in the diagram, the VRAM has two ports, one port is used by the controller to randomly access the RAM, and the other port is controlled by the VSC to output video data continuously to the DAC. The clock generator produces several clocks required by the VRAM and the VSC. The VRAM, the VSC, the clock generator and the controller will be detailed in the following sub-sections.

6.2.1.1 The Video RAM

Conventionally, ordinary DRAMs have been used for video memories because of the low cost. However, DRAMs cannot meet the requirements for high speed
operation and fine resolution display. Recently, a new type of memory, dual-port video RAM, was developed [Hitachi 86]. It can independently receive the data from a controller processor and send the data to a CRT monitor to be displayed.

Fig.6.20 compares the block diagram of dual-port video RAM to that of conventional DRAM. The dual-port video RAM provides a random port for random access and a serial port for serial access, both of which operate almost independently. The dual-port video RAM is organized with a RAM which is connected to the random port (compatible with the conventional DRAM), and a SAM (Serial Access Memory) which is connected to the serial port.

A conventional DRAM has only one port for both writing data into the memory and displaying data in the memory. Display data are transferred to a CRT display continuously to refresh the display. Writing can be performed only during the blanking periods, and as a result, it offers about 30% writing efficiency. On the other hand, the dual-port video RAM realizes almost 100% writing efficiency.

The dual-port video RAM provides the following functions (Fig 6.20(b)):

1. Read/write access on random port
(2) Read/write access on serial port

(3) Data transfer between RAM and SAM

A unique aspect of dual-port video RAM is that a large block of data can be transferred between RAM and SAM, hence using the SAM as a temporary buffer. In the continuous display process, a block of data, say 256 words (the length of a word depends on the organization of the dual-port video RAM), are transferred from RAM to SAM. Then the data in SAM are shifted out through the serial port word by word. In this way, 99.6% of the time is left for the random port to access the RAM (256 out of 257 access cycles, assuming one data transfer cycle takes the same time as SAM access cycle).

The dual-port video RAM used in the video board is D41264c-12, from NEC. The main features of the dual-port video RAM are (1) RAM is organized as 64k by 4 bits at an access time of 120ns, (2) SAM is organized as 256 by 4 bits at an access time of 40 ns.

For our application, sixteen of these dual-port video RAMs are used to store a frame of 512 lines by 512 pixels, 16 bits per pixel. The relationship between the memory addresses and the pixel positions displayed on a monitor is illustrated in Fig.6.21.

<table>
<thead>
<tr>
<th>line1</th>
<th>pixel1</th>
<th>pixel2</th>
<th>pixel512</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>511</td>
</tr>
<tr>
<td>1024</td>
<td>1025</td>
<td></td>
<td>1535</td>
</tr>
<tr>
<td>261120</td>
<td>261121</td>
<td></td>
<td>261631</td>
</tr>
<tr>
<td>512</td>
<td>513</td>
<td></td>
<td>1023</td>
</tr>
<tr>
<td>261632</td>
<td>261632</td>
<td></td>
<td>262143</td>
</tr>
</tbody>
</table>

Fig.6.21 Relationship between pixel position and the address
Comparing Fig. 6.21 with Fig. 6.10 in the previous section, it can be seen that it is necessary to perform an address conversion as follows:

\[
\begin{align*}
\text{pixel address at receiver} &= \text{pixel address at transmitter} + (\text{line.number}-1) \times 512 \\
& \quad \text{(if pixel is in the odd field, line number < 256)} \\
\text{pixel address at receiver} &= \text{pixel address at transmitter} - (512 \times 256) \\
& \quad + (\text{line.number} - 256) \times 512 \\
& \quad \text{(if pixel is in the even field, line number } \geq 256) 
\end{align*}
\]

The diagram in Appendix 4 (Receiver Board, page 3) shows the circuitry of the video RAM unit on the video board. The video RAM unit consists of four banks of VRAM, each providing four bits for each pixel. For convenience, only bank 0 is used for the following discussion.

The SAM of each VRAM device is organized as 256 by 4 bits, so the data transferred from RAM to SAM on one chip comprise only half of a displayed line. If two transfers on a single chip are performed separately to display one complete line on a monitor, there is a gap in the middle of the line because after the last pixel data of the first transfer has shifted out, the VSC has to get permission from the controller to transfer the second half-line of data from RAM to SAM, and then transfer data if permission is obtained. This process takes more than one pixel time, 100 ns. Furthermore, when the VSC requests the permission for data transfer, the controller processor may be in a refresh cycle. The controller processor will not grant memory access until it has finished the refresh cycle, so the data transfer from RAM to SAM should ideally occur during the blanking periods. In the circuitry of Appendix 4 ICs 57-60 are configured so that the RAM-SAM transfer on each chip takes place at the same time. Thus data comprising two complete lines are transferred in one operation from RAM to SAM. The transfer is performed by providing ROW addresses on A0-A7 and the correct DT, WR, CAS and RAS signals. The ROW addresses and other signals are provided by the VSC. If data stored in IC57 are tagged as (1), IC58 as (2), IC59 as (3) and IC60 as (4), the pixel sequence displayed on the monitor is as shown:

line1: \((1)(2)(3)(4)(1)(2)(3)(4)\) ...
line2: \((1)(2)(3)(4)(1)(2)(3)(4)\) ...
...
It is required that the controller processor should write data to the VRAM in the same sequence as the display sequence to obtain a meaningful displayed image. If the addresses of data received by the controller processor are sequential, the processor should write first data into IC57, the second data into IC58, third data into IC59, fourth data to IC60, fifth data into IC57 and so on. This is realized by enabling RAS0, RAS1, RAS2 and RAS3 sequentially. RAS0, RAS1, RAS2 and RAS3 are obtained by decoding the least significant two address lines of the processor.

ICs61-64 are 4-bit shift registers. When RAMCLK (SC) rising edge occurs, four bits (one pixel) on each SAM are shifted into the output buffer on the serial port of the VRAM. When SHIFTEN (SO) goes high and the rising edge of PIXCLK (CLK) occurs, the data in the serial output buffer of the VRAMs are shifted into the four shift registers. Then the data in the registers are shifted out pixel by pixel. Note that the first bit of the every pixel is shifted into IC61, the second bit into IC62, the third bit into IC63 and the fourth bit into IC64. The timing diagram of PIXCLK, SHIFTEN and RAMCLK is shown in Fig.6.22.

**6.2.1.2 The Video System Controller (VSC)**

A video system controller is used to generate video sync signal and to produce ROW addresses (which should be in step with the sync signal) and other signals to transfer data from the RAM to the SAM. For simplicity and flexibility, a
programmable device, TMS34061 video system controller is used to realize these functions [Texas 86]. To avoid confusion between the controller processor and the video system controller, the controller processor is referred to as the host processor in the following description.

Highly programmable, the TMS34061 supports a broad range of raster-scan display systems with various resolutions and scan rates. Some of the major functions of the TMS34061 VSC are:

- Generates all control signals necessary to control VRAM devices.
- Generates the video sync and blanking signals necessary to control a CRT monitor.
- Supports both interlaced and non-interlaced displays of essentially any display resolution (from 256 to greater than 4096 per line).
- Automatically generates the special display-update cycles required by VRAM memories to maintain the CRT display.
- Automatically performs periodic DRAM refresh cycles necessary to maintain data stored in the VRAMs.

The TMS34061 contains eighteen programmable registers. The values in the registers are set according to the functions required. Each register is nominally 16 bits wide and can be written to and read from by the host processor through an 8-bit data path one byte at a time.

The host processor accesses the programmable registers within the VSC by means of special read and write cycles. A register-access cycle is selected by setting the function-select input pins FS0-FS2 to 000. One of 18 registers is selected by the 5-bit register address input on column-address input pins CA2-CA6. Binary codes 00000 through 10001 are valid register addresses. The high or low byte of each register is selected by the value input on CA1. If CA1 is zero, the register low byte is selected, otherwise the register high byte is selected. Thus, the eighteen 16-bit programmable registers can be seen by the host processor as thirty-six 8-bit programmable registers, and they can be accessed by the address lines CA1-CA6. These registers are memory mapped into the host processor memory space. For simplicity of implementation, these registers occupy one memory space only and they are addressed by the second byte of the host processor data lines. The first
data byte carries the real data for the registers. For VSC circuitry, refer to the circuit diagram in Appendix 4 (Receiver Board, page 2).

These registers are memory mapped to the host processor memory space at hex 18000000. In OCCAM, they can be declared as below:

```
INT vsc.registers :
PLACE vsc.registers AT $18000000:
```

The registers are programmed as following:

<table>
<thead>
<tr>
<th>register address value in register</th>
<th>register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>vsc.registers:=( 0*256) + 10</td>
<td>--lower byte of horizontal end sync</td>
</tr>
<tr>
<td>vsc.registers:=( 1*256) + 0</td>
<td>--higher byte of horizontal end sync</td>
</tr>
<tr>
<td>vsc.registers:=( 2*256) + 25</td>
<td>--lower byte of horizontal end blanking</td>
</tr>
<tr>
<td>vsc.registers:=( 3*256) + 0</td>
<td>--higher byte of horizontal end blanking</td>
</tr>
<tr>
<td>vsc.registers:=( 4*256) + 155</td>
<td>--lower byte of horizontal start blanking</td>
</tr>
<tr>
<td>vsc.registers:=( 5*256) + 0</td>
<td>--higher byte of horizontal start blanking</td>
</tr>
<tr>
<td>vsc.registers:=( 6*256) + 158</td>
<td>--lower byte of horizontal total</td>
</tr>
<tr>
<td>vsc.registers:=( 7*256) + 0</td>
<td>--higher byte of horizontal total</td>
</tr>
<tr>
<td>vsc.registers:=( 8*256) + 1</td>
<td>--lower byte of vertical end sync</td>
</tr>
<tr>
<td>vsc.registers:=( 9*256) + 0</td>
<td>--higher byte of vertical end sync</td>
</tr>
<tr>
<td>vsc.registers:=(10*256) + 37</td>
<td>--lower byte of vertical end blanking</td>
</tr>
<tr>
<td>vsc.registers:=(11*256) + 0</td>
<td>--higher byte of vertical end blanking</td>
</tr>
<tr>
<td>vsc.registers:=(12*256) + 37</td>
<td>--lower byte of vertical start blanking,</td>
</tr>
<tr>
<td>vsc.registers:=(13*256) + 1</td>
<td>--higher byte of vertical start blanking</td>
</tr>
<tr>
<td>vsc.registers:=(14*256) + 56</td>
<td>--lower byte of vertical total</td>
</tr>
<tr>
<td>vsc.registers:=(15*256) + 1</td>
<td>--higher byte of vertical total</td>
</tr>
<tr>
<td>vsc.registers:=(16*256) + 4</td>
<td>--lower byte of display update</td>
</tr>
<tr>
<td>vsc.registers:=(17*256) + 0</td>
<td>--higher byte of display update</td>
</tr>
<tr>
<td>vsc.registers:=(18*256) + 0</td>
<td>--low byte of display start</td>
</tr>
<tr>
<td>vsc.registers:=(19*256) + 0</td>
<td>--higher byte of display start</td>
</tr>
<tr>
<td>vsc.registers:=(20*256) + 0</td>
<td>--lower byte of vertical interrupt</td>
</tr>
<tr>
<td>vsc.registers:=(21*256) + 0</td>
<td>--higher byte of vertical interrupt</td>
</tr>
</tbody>
</table>
vsc.registers:=(22*256) + 0 --lower byte of control register 1
vsc.registers:=(23*256) + 0 --higher byte of control register 1
vsc.registers:=(24*256) + 0 --lower byte of control register 2
vsc.registers:=(25*256) + 48 --higher byte of control register 2
vsc.registers:=(26*256) + 0 --lower byte of status register
vsc.registers:=(27*256) + 0 --higher byte of status register
vsc.registers:=(28*256) + 16 --lower byte of x-y offset register
vsc.registers:=(29*256) + 0 --higher byte of x-y offset register
vsc.registers:=(30*256) + 0 --lower byte of x-y address register
vsc.registers:=(31*256) + 0 --higher byte of x-y address register
vsc.registers:=(32*256) + 0 --lower byte of display address register
vsc.registers:=(33*256) + 0 --higher byte of display address register

The last register, the vertical count register, can be read from but may not be written to. Thus it is omitted in the above list. The values in the registers are worked out to produce the standard PAL sync signals and correct ROW addressing and other signals required by the VRAM.

In the circuit diagram in Appendix 4 (Receiver Board, page 3), IC27 and IC28 are disabled for much of the time. IC25 and IC26 are enabled and the transputer has free access to the VRAM. When the VSC wants to access the VRAM, it pulls the MEMREQ to low state (0) to request memory access from the transputer. After the transputer has finished the outstanding cycle, BUSFREE is taken low. IC25 and IC26 are disabled and IC27 and IC28 are enabled. The VSC can access the VRAM. When the VSC has finished the access, MEMREQ is taken high by the VSC and the BUSFREE is taken high by the transputer. IC27 and IC28 are disabled and IC25 and IC26 are enabled, giving the transputer access to the VRAM again. The VSC accesses the VRAM once every two lines for about 300ns. Thus the transputer has access to the VRAM for most of the time.

6.2.1.3. The Clock Generator

The clock generator provides video clock and system clock signals for the VSC, and provides SHIFTEN, RAMCLK and PIXCLK for the video RAM. These signals are generated based on a 10 MHz clock from a crystal module as shown in the
lower part of the circuitry diagram in Appendix 4 (Receiver Board, page 3). The timing diagram for these signals is shown in Fig 6.22.

6.2.1.4 The Video Decoder Controller

The controller (host processor) in the video decoder is the same as the controller in the video coder, except that the memory-request and the memory-granted signals are used in the decoder. The transputer and the VSC share the VRAM, although the VSC only performs one RAM-SAM transfer cycle every two lines. When the VSC wants to transfer data from RAM to SAM, the memory-request output from the VSC goes high. When the transputer detects that the memory-request signal is high, if there is no refresh cycle outstanding, the transputer takes the address and data lines high impedance and gives permission to the VSC to access the VRAM by taking memory-granted high. After the VSC has finished one RAM-SAM transfer cycle, both the memory-request and the memory-granted signals are taken low, and the transputer can access the VRAM again. If a refresh cycle is outstanding when memory-request is sampled high, the refresh cycle is completed before memory-granted is taken high.

The VRAM and the registers in the VSC are memory mapped to the transputer memory space. The VRAM is mapped at hex 08000000, and the registers are mapped at hex 18000000.

6.2.1.5 Features of the Video Receiver Board

The video receiver board consists of a transputer based controller, 4-Mbits VRAM and a programmable VSC. The controller receives data from the network and writes the data into the VRAM through the random port of the VRAM, meanwhile the data in the VRAM are continuously read out to refresh the display under the control of the VSC.

The board is a double height, standard depth Eurocard. The board is implemented in wire-wrap. The controller receives data through one of four transputer links. The board outputs R/U (5 bits), G/Y (6 bits), B/V (5 bits) composite sync and 10 MHz pixel clock to the DAC through a 40-way ribbon cable. The board is powered by a 5 volt supply through the rear connector, which is compatible with the VME bus. For detailed pin connections, see Appendix 4.
6.2.2 The DAC Board

The DAC board receives R/U (5 bits), G/Y (6 bits), B/V (5 bits), composite sync and a 10 MHz pixel clock from the video receiver board through a 40-way ribbon cable. The board converts the digital data received into analogue signals at a conversion speed of 10 MHz. If the digital video data are in Y, U, and V form, they are transferred back to R, G, B form after the D/A conversion. The board outputs R, G, B analogue signals and composite sync to the monitor through four separate phono connectors at the front of the board. The functional diagram of the DAC is shown in Fig. 6.23. For detailed circuitry, see Appendix 5.

![Functional Diagram of the DAC Board]

**Fig. 6.23 Functional Diagram of the DAC Board**
6.3 Applications of the video Codec

The video codec was designed and developed to transfer slow-scan video over the Unison network. This application will be dealt with thoroughly in later chapters. The codec also has applications in non-networked image processing. This is possible because transputers can be interconnected easily and the TDS (Transputer Development System) runs on any transputer interfaced to an IBM PC or a compatible machine. The coder is interfaced to a PC via transputer links. An interface board, based on a link adaptor known as a Sension board is used to connect the PC and the coder. The Sension board is plugged into a PC slot and connected to the coder via one of the transputer links, and the coder is connected to the decoder either directly or via a network. The controller in the coder can run TDS and can be programmed to capture a frame of image and process the image. The general term 'process' is used here to imply that there are numerous applications of the video system. Firstly, the frame of image can be stored on the PC disc for future reference using the TDS. Secondly, image statistics and compression algorithms can be investigated on the captured frame, and the transputer's parallel processing feature makes it extremely useful in investigating parallel processing algorithms. Thirdly, a still image can be data compressed before being transferred to a remote site via the network to save network bandwidth. The compressed image can be either transferred immediately after the compression or stored on the PC disc for future use. So a user can prepare still images for a teleconference in advance.
6.4 The Interface between the Codec and the CFR

The last section briefly discussed some side-applications of the codec. The main application to transfer slow-scan video over the Unison network will be dealt with in Chapters 7 and 8. This section discusses the physical interface between the codec and the CFR. Although this is not the author's work, for completeness, it is briefly discussed here.

The CFR interface card is also based on a T414 transputer. It is similar to the controller unit in the video coder or in the video decoder. The major difference is in the usage of the transputer memory space. The transputer address space is also divided into eight equal segments by decoding the three most significant address lines (AD29-31). There is also 1-Mbyte DRAM on the interface card to run the TDS and the application software. The difference is that one segment is assigned to EPROM and two segments are assigned to CFR registers and a CFR disable latch.

Up to 256 Kbytes of EPROM may be fitted on the card in the form of four 28-pin devices. The following types of EPROM may be fitted: 2764, 27128, 27256 and 27512. The main purpose of fitting an EPROM is so that the transputer can be bootstrapped from it rather than from one of the transputer links. This is very useful to the video application. Without EPROM, every video coder and decoder has to have a PC attached to allow bootstrapping from a link. This is expensive and inconvenient in use. With EPROM bootstrapping, the video application code can be blown into the EPROM and the video coder and decoder are booted on reset of the transputers. Note that the software running in the controllers of the video coder and decoder is also loaded from the EPROM via transputer links.

Thirty-two CFR station registers are also memory mapped into the transputer's address space. The transputer can directly access them. In this way, the transputer is interfaced to the CFR.

The CFR disable latch is used to enable and disable the CFR station. The latch value when zero indicates that the station is disabled, and when one indicates enabled.

The resources which are mapped are shown below, together with the corresponding values on the upper address lines.
<table>
<thead>
<tr>
<th>Resource</th>
<th>AD31</th>
<th>AD30</th>
<th>AD29</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic RAM</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EPROM</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CFR registers</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CFR disable latch</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The video coder and decoder are connected to the interface cards via transputer links.
Chapter 7

Video Protocols

7.1 Introduction

The previous chapter discussed the hardware of the CFR video codec. This chapter is devoted to the description of the protocols and software needed to provide video service over the Unison network using the video codec.

Section 7.2 describes the way the codecs set up associations with each other. Section 7.3 discusses how video data are packetized at the transmitter and depacketized at the receiver. The implementation of two simple coding schemes, sub-sampling and block conditional replenishment, are discussed in section 7.4. Section 7.5 discusses the flow control strategies. Video channel specification is described in section 7.6. Section 7.7 describes the protocol used by the workstations to control the video stations. The structure of the software running in the codec is presented in section 7.8. Section 7.9 is about the user interface and codec operation. Section 7.10 summarizes the chapter.

7.2 Connection Setup

Before a transmitter can send any data to a receiver, a virtual connection must be set up first between the transmitter and the receiver.

The Secretary service has been described in Chapter 2. When a video station (a transmitter or a receiver) is switched on, it sets up an S-association with its local Secretary. After that the video station registers its service with the Secretary. A transmitter registers "video-c" and "video-d" with the Secretary, while a receiver also registers "video-c" and "video-d" with the Secretary. "video-c" is used by the
workstation to send control messages to the video station. "video-d" is a service advertised on the network and is used by video stations to talk to each other.

A workstation is used to control the operations of the video stations. When a video connection is required, the workstation selects a transmitter and sets up an association with it. The workstation then sends a "video-start" command with the connection attributes (see Section 7.6) and the destination receiver name to the transmitter. When the transmitter receives the "video-start" command, it sends an associate-request RPC to the local Secretary with the destination receiver name and connection specification. After that it is the Secretary's responsibility to negotiate with the receiver whose name is specified in the association-request call. If the receiver is at a remote site, the Secretary at that site has to be consulted and that Secretary will negotiate with the receiver. If the receiver rejects the request for some reason, the Secretary will inform the initiating transmitter. Usually the receiver will accept the request and the Secretary will confirm the acceptance to the initiating transmitter with the address and a private port of the receiver. If the receiver is at a remote site, the address and the port returned to the transmitter are those of its local portal. The portal has the routing information of the connection.

Fig7.1 An Example of Connection Setup
The private ports allocated by the transmitter and the receiver during the connection setup process are used for future data communication.

A video station can set up more than one connection with other video stations. Each connection has its own identity number. Video stations keep a list of the connection identity numbers together with their connection specifications. It should be noted that in theory both a transmitter and a receiver can initiate a video call to other video stations. For implementation simplicity only a transmitter is allowed to initiate a video call.

Fig 7.1 shows an example of a workstation initiating a video connection between a transmitter and a local receiver. There are six transitions:

1. The local workstation sends a "video-start" command to the transmitter with the receiver name and connection attributes.

2. The transmitter allocates a private port for the video connection and sends an association-request RPC with the port number, receiver name and connection attributes to the local Secretary down the S-association.

3. The local Secretary passes the association-request to the receiver whose name is specified by the initiating transmitter together with the connection attributes, transmitter address and connection port allocated by the transmitter down the S-association between the receiver and the Secretary.

4. If the receiver accepts the association request it keeps the transmitter address, port number and connection attributes. The receiver also allocates a private port for the connection and confirms the association request to the Secretary with the port number it allocated.

5. The Secretary passes the association confirmation to the transmitter with the receiver address and port for the connection.

6. The transmitter stores the receiver address and port, and assigns a connection identity number to the connection, it then sends a positive reply to the workstation with the connection identity number (handle).

The connection is thus established and the private ports allocated by the transmitter and the receiver are to be used for future communication between the transmitter
and the receiver. In step (4), if the receiver rejects the association request then it will reply to the Secretary negatively. The transmitter and the workstation are informed of the failure.

In this way, a transmitter can set up more than one association with one or more receivers, and a receiver can have more than one association with one or more transmitters.

### 7.3 Packetization and Depacketization

Video data are assembled into packets which are transmitted to the receiver individually. The size of the packet has several effects on the quality of reconstructed images. The larger the packet, the longer the time taken to assemble one packet. If the packet is too large, burstiness in the image reconstruction can be seen. The size of the packet also affects the image reconstruction in the case of packet losses. The loss of a larger packet will affect a larger part of the reconstructed image.

It was decided that the size of the video data packet should be the same as a minipacket on the CFR. This size was chosen for three reasons. Firstly it fits into a single CFR slot. Transmission and reception in a video station will therefore be very simple, since there need be no procedure for blocking groups of slots into packets. Secondly, the packetization delay is minimum. Thirdly since higher level protocol is not needed to transmit video data in minipackets, the network bandwidth can be used efficiently without the high level protocol overhead. Video data can be transmitted directly over the association established as described in the previous section.

Pixel addresses are needed to reconstruct the image at the receiver from the video data received. There are two ways to obtain the pixel addresses. One scheme is for the receiver to generate the pixel addresses itself according to the number of minipackets it received and the channel specification. The receiver is synchronized by the transmitter sending a frame starting signal at the start of each frame. The second scheme is for the transmitter to send the pixel address of the first pixel in the minipacket along with the video data to the receiver, the receiver then works out the addresses of other pixels in the minipacket from the address of the first pixel.
The first addressing scheme is more efficient in terms of network bandwidth usage since no addressing data are sent along with the video data to the receiver. But this scheme is more sensitive to packet losses. If there is a single packet loss, the packets following in that frame will be put in the wrong place. The picture thus reconstructed is distorted totally. This distortion will be corrected when the start signal of the next frame is received. But this new frame could be distorted again by a single packet loss. Therefore, this scheme should only be used in a very low packet loss environment. In Unison, this scheme is suited to the local CFR only. If the traffic goes over a portal there is a relatively higher chance that a minipacket might be lost due to congestion at the portal. A frame starting signal is sent to synchronize the receiver. The starting signal is contained within one minipacket and the value pattern is so chosen such that the video data will very rarely take this pattern. The pattern used is &FFFF0000...FFFF0000. Considering the correlation between pixels the video data will very rarely take this pattern.

In the second addressing scheme, two bytes are used in each minipacket to transmit the address of the first pixel in the minipacket. In this scheme, even if there are packet losses, the packets received can still be placed at the right position using the addressing information in the minipackets. The lost data are substituted with the corresponding data from the previous frame. If the movement in the frame is not very big, the distortion caused by the minipacket losses is quite difficult to see.

The data parts of the minipackets are shown in Fig.7.2 for the both addressing schemes discussed above. There are 32 bytes in the data part of a CFR minipacket. In the first addressing scheme, one minipacket contains 28 bytes of video data whilst 4 bytes are used to transfer the port number (and control bits). In the second

<table>
<thead>
<tr>
<th>4 bytes</th>
<th>28 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>port</td>
<td>video data</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>4 bytes</th>
<th>2 bytes</th>
<th>26 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>port</td>
<td>pixel address</td>
<td>video data</td>
</tr>
</tbody>
</table>

(b)

**Fig.7.2 Data Part of a Minipacket**
addressing scheme one minipacket contains 26 bytes of video data, with the port number (and control bits) and the address of the first pixel in the minipacket taking 4 bytes and 2 bytes respectively. Because of the packet loss tolerance characteristic of the second scheme, it is almost always used in the Unison video services.

At the receiver, the port number in the incoming minipacket is checked to determine the channel the minipacket belongs to because one receiver can have several video channels at one time. The video data are then placed at the correct position in the video memory according to the addressing information and the channel specification.

7.4 Video Coding Schemes

The simplest way for a transmitter to send video data to a receiver is to transmit every pixel data in the area of a frame specified in the connection attributes by the workstation. In that case, one pixel is represented by 2 bytes of data. One minipacket can convey 13 pixels if the second addressing scheme is used. This method needs a lot of bandwidth to transmit a picture. For example, 1 Mbits (plus addressing and protocol overhead) are required to transmit a colour picture of 256 lines by 256 pixels.

There are two basic video services: slow scan video and still image transfer. For still image transfer it is acceptable to transmit every pixel in a frame to a receiver without any compression coding, because this transfer is not continuous and still image usually needs very high quality. For slow scan video, some coding schemes can be used to reduce the bandwidth required and still keep the picture quality acceptable. The coding schemes implemented in software in the video codec are: subsampling and block conditional replenishment. In the case where monochrome image is sufficient, monochrome image transfer can be used to save bandwidth.

7.4.1 Subsampling

Since the video memories in both the transmitter and receiver are memory mapped to the Transputers, subsampling is very simple to implement. At the transmitter one pixel out of every m pixels on one line and one line out of every n lines are transmitted. At the receiver the picture is reconstructed by simple repetition of pixels received. There are various effects for different m, n and number of
replication. Fig.7.3 shows some typical effects with fixed m and n but different p and l, p being the horizontal repetition number, l being the vertical repetition number. Pixels are represented by numbers in the diagram. m, n, l and p are selected by a user when he starts a video channel on a workstation.

In the first case, the receiver displays the received pixels only, so the picture size is reduced by m*n times from the original picture. In case two, pixels received are repeated twice horizontally. The picture recovered is horizontally expanded. In case three, pixels received are repeated twice vertically. The picture recovered is vertically expanded. In case four, m=p and n=l, the picture recovered has the same size of the original one though the resolution is reduced. In case five, p>m, l>n and l=p, the picture recovered is expanded by (lxp)/(mxn). In practical use, pictures recovered should keep the same horizontal/vertical ratio as the original picture. So normally, m is equal to n, and p is equal to l.

If case four is used and the same amount of network bandwidth is provided, the picture updating rate using the subsampling transmission method is four times faster than that using direct (non-subsampling) transmission method.

Some complex interpolation algorithm could be used, as in [Mallappa88]. Since the coding aim here is to increase the picture updating rate and it is implemented in software, the simple pixel repetition at the receivers is adopted to increase the transmission rate. The result shows that the picture quality obtained using this simple subsampling method is acceptable.

![Fig.7.3 Subsampling Effects](image)

Chapter 7 Video Protocols
In view of the low bandwidth of the chrominance signal, the chrominance signal can be transmitted once for several pixels to further reduce the data amount required to transmit a picture.

7.4.2 Conditional Replenishment

In this coding method, an image is divided into pixel blocks and only pixel blocks which have changed significantly from the previous frame are transmitted to the receiver. Those pixels which have not been transmitted by the transmitter are substituted by the corresponding pixel values from the previous frame at the receiver. This method can reduce the bandwidth required significantly. For detailed result, see Chapter 8.

The pixel block size is the same as the size of a minipacket, that is 13 pixels, because the second addressing scheme is to be used in this case. For simplicity, the 13 pixels in a block are 13 successive pixels on a line. In the simple implementation, each pixel in a block is compared with the corresponding pixel in the previous frame. The comparison is based on the luminance component, Y, only, since the Y component is the most sensitive one among the Y, U and V components. The absolute values of these thirteen differences are accumulated. If the sum of these values is greater than a pre-defined threshold, this pixel block is regarded to have changed significantly and is transmitted to the receiver. The threshold is selected from the workstation when the video channel is being established.

Since the coding is implemented in software, this method can reduce the amount of data required to transmit a frame of picture but can not increase the picture updating rate. One way to increase the speed is to explore the pixel correlation further. Instead of calculating the differences of all thirteen pixels in the block, only the differences of every other pixels are calculated. That is, only pixels 1, 3, 5, 7, 9, 11, 13 are compared with the corresponding pixels in the previous frame. Whether the block should be transmitted or not is based on the sum of the absolute values of these seven differences. So long as the threshold is reduced to the half of the threshold used in the "full comparing" method, the quality of the pictures recovered at the receiver using this method has no noticeable degradation.
7.4.3 Monochrome Image Transfer

In some applications, monochrome image display is sufficient, such as the image displayed locally to monitor the user's position in a frame. For monochrome image, only the Y component needs to be transmitted. The Y component has 6 bits, but for the convenience in assembling a minipacket, 8 bits (1 byte) for each pixel are transmitted. That is, the monochrome image transmission uses half the bandwidth of the colour image transmission.

7.5 Data Flow Control

The amount of data corruption and packet loss is very low in networks of the Unison type if the traffic loading is within the network capacity [Siddiqui 88a]. Thus the data loss (error) control is quite simple. There are two possibilities where minipackets might be lost. (1) After a video channel has been established, the receiver or the bridge components of the network go down. Since the network does not provide acknowledgement to the transmitter while the minipackets are transmitted, it is up to the application to provide some kind of mechanism to detect any fault in the network and the receiver. (2) A receiver and the network bridging components have to handle several traffic streams from different sources. The network might be overloaded and the receiver might not be able to keep up with the speed at which the minipackets are coming in. If that happens, when the portal or the receiver is on the same CFR as the transmitter, the minipacket will return to the transmitter and request the transmitter to re-send the minipacket. If the receiver still cannot accept the minipacket after a certain number of re-transmissions, the minipacket will be lost.

The simplest way to detect any fault in the network components and receivers is to design the transmitter to expect an "alive" minipacket from the receivers after a certain time. If the minipacket comes on time, the transmitter knows the network and the receiver are all right and it can keep on transmitting video data. However, if the minipacket does not come as expected, the transmitter cannot assume that there is definitely something wrong with the network or the receiver, because there are possibilities that the "alive" minipacket is lost due to network overloading. The transmitter has to wait patiently for another "alive" minipacket to come. If it comes
at the expected time, the transmitter does not take any action and keeps on transmitting video data. If the "alive" minipacket does not come at the expected time again, the transmitter will conclude that there is something wrong with the network components or the receiver. The transmitter will stop transmitting video data to the receiver and the video channel will be shut down. The reason to do so is to prevent bandwidth wasting if there is something wrong with the receiver or some part of the network.

If the network and the receiver are all right but cannot handle the traffic fully, some minipackets will be discarded. For the slow scan video service, if the rate of minipacket loss is very small, the quality of the picture recovered might be still acceptable for two reasons. (a) If a minipacket is lost, the pixels in that minipacket will be substituted by the corresponding pixel values of the previous frame. There is a chance that these two corresponding pixels are not very different. (b) The lost minipacket will be replenished in the next frame. It is rare that the minipacket at the same position will be lost in two consecutive frames. But if packet losses are too many, or for the still image transfer, the picture quality will not be acceptable. In the case of still image transfer, the lost minipacket should be retransmitted. In the case of slow scan video transfer, retransmission is not necessary, but the transmitter must be informed to reduce the transmission speed. Retransmission can be achieved by sending minipacket sequence number along with the video data minipacket. The receiver can tell which minipacket has been lost and how many minipackets have been lost from the sequence numbers of the minipackets. These statistics are sent back to the transmitter after a certain time (perhaps after every frame) and the transmitter should take action accordingly: to reduce the transmission rate if the video service is slow scan video and to retransmit the lost minipacket if the service is still image transfer. This statistics minipacket can also serve as the "alive" minipacket so that the "alive" minipacket does not need to be transmitted separately. The minipacket sequence number can be sent in the control bits of the port number. Actually only two bytes are used for the port number although four bytes are assigned. So the two unused bytes can be used for the minipacket sequence number.

When the association involves portal(s), one must be very careful in deciding the transmission rate to be used. Since there is no acknowledgment from the portal, if it is overloaded, minipackets will be lost. The minipacket containing the loss statistics
(also used as the "alive" minipacket) may also be lost. As a result the transmitter would not slow down its transmission rate as required because of the lack of the statistics and further minipacket will be lost. If two consecutive "alive" minipackets are lost the channel will be shut down automatically. Thus the bit rate to be used by a channel should not be too high in relation to portal's capacity.

On a local CFR, the minipacket sent to the receiver will return to the transmitter with the "receiver busy" information if the receiver is busy, and the CFR chip in the transmitter will re-send the minipacket. This is called hardware retry. The number of hardware retries is set by programming a register on the CFR chip, the maximum number being 11. Apart from the hardware retry, a user can implement retransmission in software. The software retry is used when a pre-fixed number of hardware retries have failed. The number of software retries is decided by the implementor and there is no upper limit on this. In the video application, the number of hardware retries is set to 7 and software retries to 5. One hardware retry causes about 20 μs delay so total delay caused by the hardware and software retries is about 700 μs. That is to say if a minipacket cannot be accepted by the receiver within 700 μs it will be lost.

A 32 Kbyte buffer is used at both the transmitters and receivers. At the transmitter, the buffer is used to smooth and increase the minipacket transmission rate. When the transmission process is busy, the video data of the current frame are transferred into the buffer and the grabber can proceed to capture a new frame. Meanwhile the data in the buffer are being transmitted. The transmission process is run in the transputer on the CFR interface board, and the grabbing process is run in the grabber control transputer. This is a typical example of parallel processing. At the receiver, when the displaying transputer is busy decoding some of the incoming data, the other incoming data can be stored in the buffer to avoid being discarded.

7.6 Video Channel Specification

A video channel is an association between a video transmitter and a video receiver on which video data are transferred. The channel attributes are selected on a workstation before it sends the "video-start" command to the selected transmitter. Every video channel has a channel identity number (handle) which is returned to
the workstation by the transmitter after it has successfully established the video channel specified by the workstation. The channel attributes are as follows:

```c
INT    handle;
INT    mode;
STRING  transmitter, MAXLENGTH=100;
STRING  receiver, MAXLENGTH=100;
INT    tx.position ARRAY LENGTH=4;
INT    tx.resolution ARRAY LENGTH=2;
INT    rx.position ARRAY LENGTH=4;
INT    rx.resolution ARRAY LENGTH=2;
INT    delay;
INT    threshold.
```

These attributes are kept in both the transmitter and the receiver associated with the channel. The attributes in the transmitter and the receiver are the same except the handle. The handles are assigned independently by the transmitter and the receiver. The handle assigned by the transmitter is returned to the workstation which can later manipulate the channel with the handle.

The second attribute defines the mode of the channel of the service. There are two basic transmission modes, freeze image transfer and slow scan video transfer. For freeze image transfer, the channel will be shut down automatically after one image has been transferred. Both colour and monochrome freeze images can be transferred. For slow scan video, four transmission methods can be used: colour, monochrome, colour conditional replenishment and monochrome conditional replenishment. These modes are assigned as follows:

- **0(000)**  colour freeze image transfer
- **1(001)**  monochrome freeze image transfer
- **2(010)**  colour slow scan
- **3(011)**  colour slow scan with conditional replenishment
- **4(100)**  monochrome slow scan
Both freeze image transfer and slow scan video can use subsampling transmission method. But it is not specified in the channel mode. It is specified implicitly in `tx.resolution` and `rx.resolution`.

The third and fourth attributes are transmitter and receiver names which are 100 byte character long strings.

The fifth attribute is a four-integer array which defines the picture position and the area in a frame at the transmitter to be transmitted. The array is defined as `[x0, y0, x1, y1]`. `[x0,y0]` defines the position of the pixel in the top left corner. `[x1,y1]` defines the position of the pixel in the bottom right corner. `x0, y0,x1` and `y1` are in the range 0 to 511 as shown in Fig.7.4. When the transmitted area required is smaller than 256 lines by 256 pixels, only pixels in the odd field are transmitted so that a wider scene can be seen instead of a small corner in a frame (two fields).

The sixth attribute is a two-integer array `[tx.res1, tx.res2]`, which defines the horizontal and vertical sub-sampling ratio. The `tx.res1` and `tx.res2` correspond to the `m` and `n` respectively as defined in Section 7.4.1.

The seventh attribute is a four-integer array `[rx0, ry0, rx1, ry1]`, which defines the position where the received video data are displayed. The definition of the array is the same as in Fig 7.4. If the area of the transmitted image, `tx.resolution` and `rx.resolution` are decided, the area of the recovered picture at the receiver is also

![Fig. 7.4. Illustration of picture area to be transmitted](image)
decided. So rx1 and ry1 are actually not used. Only rx0 and ry0 are used to define the top left corner of the recovered image.

The eighth attribute is a two-integer array [rx.res1, rx.res2], which defines the horizontal and vertical repetition numbers of the received pixels. The rx.res1 and rx.res2 correspond to the p and l respectively as defined in Section 7.4.1.

The ninth attribute is an integer which defines the transmission delay between video data minipackets. The delay value is the delay time between two successive minipacket transmissions. The delay is defined in unit of Transputer ticks. One Transputer tick in a low priority process is 64 microsecond. Thus, the bigger the value, the slower the transmission speed. It is worth noting that the transmission rate is determined not only by the delay value but also by the software execution time and the network loading and the receiver speed. If the network is very busy or the receiver cannot keep up with the speed of the transmitter, the transmitter will try to re-send the minipacket, thus the transmission rate will slow down. For example, if the delay value is 2, the delay between two successive minipacket transmissions is 128 microseconds plus some software execution time. This attribute is used to adjust the transmission rate according to the network loading. The portal can be overloaded even when there are only two active channels (one in each direction) if the maximum transmission rate is used.

The tenth attribute is an integer which defines the threshold in conditional replenishment transmission mode. It is used only when the conditional replenishment channel mode is used.

Once a channel has been established between a transmitter and a receiver, the channel can be manipulated by the workstation. For example, the workstation can modify and stop the channel.

### 7.7 Video Control Protocol

This section describes which workstation can directly control a particular device on the network and what kind of control commands are available for the workstation to control the video transmitters and receivers.

Every workstation has a list of devices which are under its control. Although there is nothing to prevent a workstation from controlling a device which is not in its
control list, normally, a workstation only directly controls the devices in its control list. Control of a device not in its control list is done through the workstation which can directly control that device.

See Fig. 7.5, TX1 (transmitter 1) and RX1 (receiver 1) are under the direct control of WS1 (workstation 1). TX2 and RX2 are under the direct control of WS2. If WS1 wants to send video from TX1 to site 2, WS1 has to send a connection request to WS2 and WS2 checks which video receiver is available. If RX2 is available, WS2 will send positive reply to WS1 with the name of RX2. Then WS1 can send a "video-start" command to TX1 with the destination receiver name RX2 and channel attributes. The channel will be established as described in Section 7.2.

If WS1 wants a video display on RX1 from site 2, WS1 will send a request to WS2 with the receiver name RX1. If there is a transmitter available at site 2, WS2 will positively reply to WS1 and send a "video-start" command to TX2 to establish a video channel with RX1.

If WS2 wants to initiate a video channel to site 1, WS2 has to go through WS1 just as WS1 has to go through WS2 when WS1 wants to initiate a video channel.

A workstation can logon (select) to a transmitter under its direct control and can start a new channel from the transmitter, modify an existing channel, stop an already existing channel or reset the transmitter. When a workstation starts a new

Fig. 7.5 Video Station Control
channel from the transmitter or modifies an existing channel, the receiver has to be informed. When a workstation stops an existing channel or reset the transmitter, the receiver is not necessarily informed because when the channel has been stopped, the receiver will not receive anything from that channel, and will eventually kill the channel at its end automatically after a certain time.

A workstation can logon to a receiver under its direct control and can stop an existing channel, clear the receiver display screen or reset the receiver, but it cannot start a new channel and modify an existing channel from the receiver. When a workstation stops a channel from the receiver's end or reset the receiver, the transmitter does not need to be informed because the transmitter will stop the transmission over the channel if it has not received an "alive" minipacket from the receiver within a certain time.

7.8 Software Structure

The software in the video transmitter and receiver is implemented in OCCAM. The software consists of many processes. Communications between processes are via channels. The structures of the software in a transmitter and the software in a receiver are similar except that the transmitter grabs pictures and transmits the video data to the network and the receiver receives the data from the network and displays them on the screen. In this section, only the software in the transmitter is analysed.

Fig.7.6 shows the block diagram of the structure of the software in the transmitter. Some details are omitted in the diagram, such as some small processes and channel specifications and names.

The grabber process is run in the transputer on the grabber board, other processes are run in the transputer on the CFR interface board. The grabber process communicates with other processes via the transputer links.

RPC Process

Several low level processes are broadly called RPC process. RPC process has the following functions:

(1) To transmit and receive minipacket to and from the network;
Fig. 7.6 Software Structure in the Transmitter
(2) To construct the UDL (Unison Data Link) protocol which is used by Remote Procedure Calls (RPC);

(3) To construct RPCs (Remote Procedure Calls);

(4) To establish S-association with the local Secretary using an RPC;

(5) To refresh the S-association using an RPC after every certain time;

(6) To pass video data and control information between the network and the video processes.

**Video.master**

Video.master is the key process in the program. It makes most of the decisions. The functions that the Video.master performs are as follows:

(1) When the workstation logs on to the transmitter, the video.master sends a list of handles and attributes of channels existing on the transmitter to the workstation. The workstation gets a map of the channels on the transmitter and can manipulate the channels and start new channels from the transmitter.

(2) When a "video-start" command comes from the local workstation (through the network and the RPC process), the video.master assigns a port number for the association (channel) to be established and initiates an association request to the receiver specified by the workstation with the channel attributes. If the association establishment fails, the video.master will reply to the workstation negatively. If the channel is established successfully, the video.master will assign a channel handle to the channel and return it to the workstation. Meanwhile, the video.master will send the receiver address and port number and channel attributes to the grabber process. The receiver address and port number are also sent to the rx.packet.handler.

(3) When the workstation sends a "stop" command with a channel handle to the transmitter, the video.master will eliminate the channel with the handle from the channel list and send commands to tx.packet.handler and the grabber processes to disable the transmission over that channel.

(4) When the transmitter receives a "modify" command with a channel handle already existed in the transmitter, attributes of the channel with the handle will be modified as required by the workstation. The old channel will be stopped and a channel with the new attributes will be established with the same receiver. The
video data will be transmitted to the receiver through the new channel. In another word, the "channel modify" is so implemented that the existing channel is stopped and a new channel with the same channel handle number is established with new channel attributes sent by the workstation with the "modify" command.

(5) When the transmitter receives a "reset" command from the workstation, the video.master will eliminate all channels from the channel list and send commands to the tx.packet.handler and the grabber processes to disable the transmission over all the channels established in the transmitter.

**rx.packet.handler**

This process is used to receive "alive" minipacket from the active channels associated with the transmitter. If this process does not receive an "alive" minipacket from a channel for a certain time, the process will conclude that there is something wrong with the receiver or the network components and the process will ask the video.master to stop the transmission over the channel. As discussed in section 7.5 the "alive" minipacket contains minipacket reception statistics, the video.master will adjust the transmission speed over the channel, if necessary, according to the statistics. If too many minipackets have been lost during freeze image transmission, the video.master will ask the grabber and transmission process to transmit the whole image again.

When the rx.packet.handler receives other control information it will pass it to the video.master.

**tx.packet.handler**

This process passes the video data minipackets from the grabber process to the RPC. There is a 32 Kbyte buffer in this process to store video data from the grabber. When video.master has any control minipacket to send, the rx.packet.handler relays the minipackets to the RPC.

**Screen.buffer and key.buffer processes**

A PC is connected to the transmitter to enable software development and load the software code into the transputers. It is also used to display debugging and other information, and to input some simple commands to the transmitter using the
keyboard. Of course, when the software is finalized, the PC is not required. The code can be loaded from the CFR interface's on-board EPROMs.

Screen.buffer is used to pass information from the video.master to a PC for display.

Key.buffer is used to receive the keyboard input from a PC. Thus a simple command can be entered from the keyboard to the video.master. For example, a video channel can be started from a PC.

**Grabber**

This process is run in the transputer on the grabber board. It carries out four functions:

1. It receives channel information from the video.master via a transputer link and keeps a list of the active channels and channel attributes;
2. It controls the grabber to capture video data from the ADC board;
3. It carries out video codings and packetization. The subsampling and block conditional replenishment codings are implemented in this process;
4. Packetized video data are passed to the tx.handler via a transputer link.

Function (1) runs in parallel with functions (2), (3) and (4). Functions (2), (3) and (4) run sequentially. That is, the grabber captures a frame of image first, then the video data are coded, packetized and transferred to the tx.handler. If there are several active video channels associated with the transmitter, the video data in a frame are coded and transmitted separately to the different channels since different channels can have different attributes. When the transmission of video data in that frame to all the active channels has finished, the grabber captures a new frame of image, and the coding and transmission processes repeat.

**7.9 User Interface and Operation**

All operation control of the video codecs is carried out on the A-500 workstations. It was intended to implement a user-friendly interface using X Window System [Scheifler 86], but due to the time limitation and unavailability of X-window on the Tripos operating system [Knight 82] within the Unison life time, the user interface was implemented in Tripos commands instead.
The software running in the codecs is loaded from the EPROMs on the CFR interface boards. To start the video codec, all that needed is to press the reset button on the interface boards. Then the workstation has full control over the video codec. A typical example to start a video channel via a workstation from transmitter "fs00" to receiver "fs01" is as follows (bold text is prompted by the workstation and italic text is typed in by a user):

```
select fs00
select lut-blue *fs01
```

Enter channel attributes at the receiver:

```
x0: 10
y0: 10
x1: 240
y1: 250
x res: 1
y res: 1
delay: 2
```

Enter channel attributes at the transmitter:

```
x0: 100
y0: 50
x1: 400
y1: 400
x res: 1
y res: 1
```

Parameters validated, about to make call ...

Then the workstation will try to start the specified video channel. If the channel is set up successfully, the handle of the channel returned from the transmitter will be printed out. Otherwise, an error number will be given.
7.10 Summary

This chapter has discussed the protocols required to provide video services over the Unison network. One video station can set up more than one video channel with one or more other video stations. So multipoint service is realized by the transmitter sending the video data in a frame to several different receivers separately.

Video data are transmitted in packetized form. The size of the packet used is the same as the minipacket of the CFR which is 32 bytes. Pixel address and packet sequence number are sent along with the video data to synchronize the receivers. This protocol tolerates packet losses since the receiver will not lose synchronization even when there are packet losses and the data from the previous frame will be displayed where data have been lost.

To reduce the amount of data required to transmit a picture, two simple coding schemes, subsampling and block conditional replenishment, have been implemented. The subsampling ratio and the threshold in conditional replenishment are selected by user on a workstation.

"alive" minipackets are sent from the receiver to the transmitter to inform the transmitter of the status of the network and the receiver. The transmitter will adjust the minipacket transmission rate according to the loading of the network and will stop the transmission over that channel if "alive" minipacket has not been received for a certain time. Hardware and software retries are used to prevent avoidable packet losses. A 32-Kbyte buffer is used at both the transmitter and the receiver to smooth the transmission rate, and to some extent to increase the maximum transmission rate by making use of the parallel processing power of the transputers.

Video channel attributes, such as the picture size and position, coding schemes and transmission delay are selected from a workstation when the channel is initiated by a user.
Chapter 8

Experiments over the Network

8.1 Introduction

Various video services can be supported by the codec hardware and video protocols described in Chapters 6 and 7. A transmitter can establish more than one association with more than one receiver, and a receiver can have more than one association with more than one transmitter. So a transmitter can send images to several receivers sequentially and a receiver can receive video data from several transmitters and display them on one monitor at the same time. In this way, multi-point communication can be achieved with one transmitter and one receiver at each site.

Colour and monochrome still image transfers and slow scan video services are available. Image size, position in a frame and resolution at both the transmitter and the receiver are selectable when a video channel is initiated via a workstation. The maximum size (resolution) is 512 lines by 512 pixels. A typical display on the receiver monitor is shown in Fig.8.1.

The delay between minipacket transmissions is selectable to control the data rate. Two coding schemes, subsampling and block conditional replenishment, are used to reduce the amount of data required to transmit an image. The pixel subsampling ratio and the threshold for the block conditional replenishment coding are also selectable to suit the network loading and application.

The video system is packet loss tolerant in the sense that any packet loss will not cause the receiver to lose synchronization. Lost data will be substituted by the corresponding pixel data from the previous frame.
Fig. 8.1 A Typical Display on the Receiver Monitor

Fig. 8.2 The Video Station Crate
In this chapter, the performance of the various services over different parts of the network is presented.

8.2 Video System Configuration

A video station (a transmitter or a receiver) can be connected to any client CFR on the Unison network. A video transmitter station consists of four boards, the ADC board, the transmitter (grabber and control) board, the CFR interface board and the CFR station board. A video receiver station also consists of four boards, the CFR station board, the CFR interface board, the video receiver board and the DAC board. The connections among these boards have been described in Chapter 6. The video stations are housed in a crate with VME bus compatible power rails. One crate can house three video stations. Fig 8.2 shows a crate with two stations fitted in.

Fig 8.2 shows a video system setup on two client CFRs which are connected via two portals through an exchange CFR. Each client CFR has a transmitter (TX) and a receiver (RX) connected to it.

---

Fig.8.3 A Setup of the Video System over the Network
The PC connected to the TX is used to collect and display experiment results, such as the transmission rate. During the development stage, the PCs are used to develop software and to load the compiled codes into the codec. When the software is finalized, PCs are no longer needed in normal use. Codes needed to operate the video codecs are loaded from EPROMs on the CFR interface board.

The workstation (WS) is used to control the operation of the video codecs. A user does all controls using the user interface on the workstation.

Several configurations have been used for the experiments. (1) Video codecs communicate with each other over a local CFR. (2) Video codecs on two different CFRs communicate with each other over two portals and an exchange CFR (shown in Fig. 8.3). (3) Video codecs on two different CFRs communicate over two portals, two ramps and an ISDN link.

8.3 Experiments over a Local CFR

In these experiments, two transmitters and two receivers were connected to a CFR. The two transmitters can send images to both receivers. A receiver can accept images from both transmitters and display them on one screen. A transmitter can set up several channels with one receiver so that images with different channel attributes can be displayed on one screen and be compared.

The CFR used is clocked at about 50 MHz. If five stations are active at the same time on the ring, the minimum point to point bandwidth available to a station is over 5 Mbits/s (see Appendix 1). The maximum output bit rate of a transmitter is less than 2 Mbits/s. Thus considering the very low bit corruption rate over a local CFR and provided the receiver can cope with the minipacket incoming rate, the rate of packet error and loss should be very low. Results of experiments confirmed this.

Experiments of video transmission along with other traffic over a local CFR were also carried out. Two video channels with transmission rate of about 1 Mbps each, one voice channel with 64 kbps in both directions and one computer data channel with traffic of over 1 Mbps were set up over the CFR. The results showed that there were no interference among traffic. This was as expected because there was sufficient bandwidth available over the CFR and the CFR prevents any bandwidth hogging by any station.
In the rest of this section, performance of the video system using different coding schemes is presented.

8.3.1. Video Transmission without any Compression

In this transmission mode, video data are transmitted without any compression coding. One minipacket carries 13 pixels of data. The best picture quality is obtained using this transmission method, although the bandwidth required is the largest.

Picture updating rate can be controlled by using different delay between minipacket transmissions. Table 8.1 shows the amount of time taken to transmit a colour picture (16 bits per pixel) of various sizes with various delays. Note that the delay is measured in transputer ticks and one tick is equivalent to 64 μs. It can be seen that the longer the delay, the slower the picture updating rate. The maximum bandwidth required with a delay of 0 is \( \frac{(256*256*16)}{0.97} \) plus pixel address (2 bytes for each minipacket), port number (2 bytes for each minipacket) and control code ( 2 bytes for each minipacket) overheads \( \frac{6*8*256*256}{(13*0.97)} \), which is about 1.33 Mbps.

The table provides a guideline for channel setup. A user can set up a video channel with the appropriate channel attributes according to his requirement and the bandwidth available on the network. This is important only when the channel goes through portals, because if the channel requires too much bandwidth the portal(s) will be overloaded and packets will be lost. This has been explained in the previous

<table>
<thead>
<tr>
<th>delay size</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 * 64</td>
<td>0.07</td>
<td>0.09</td>
<td>0.11</td>
<td>0.13</td>
<td>0.16</td>
<td>0.18</td>
</tr>
<tr>
<td>128 * 128</td>
<td>0.29</td>
<td>0.33</td>
<td>0.42</td>
<td>0.50</td>
<td>0.58</td>
<td>0.67</td>
</tr>
<tr>
<td>256 * 256</td>
<td>0.97</td>
<td>1.03</td>
<td>1.29</td>
<td>1.55</td>
<td>1.8</td>
<td>2.06</td>
</tr>
<tr>
<td>512 * 512</td>
<td>3.95</td>
<td>4.10</td>
<td>4.85</td>
<td>5.61</td>
<td>6.36</td>
<td>7.13</td>
</tr>
</tbody>
</table>

Table 8.1. Picture Transmission Times (in seconds)
chapter. It should be noted that the time shown in Table 8.1 is the minimum time required to transmit an image of a certain size when the receiver is always ready to accept data. When the receiver is busy, such as when there are two or more video channels going to the same receiver, the time taken to transmit an image of the same size might be slightly longer than those shown in the table due to the probable retransmissions in the transmitter.

8.3.2. Video Transmission with Sub-samplings

In these experiments, pixel sub-sampling ratio of 2 was used in both horizontal and vertical directions although use of other sub-sampling ratio is possible. The data amount required to transmit a certain picture was reduced to one quarter of that required when no compression is used.

Experiments with three different combinations of resolutions were carried out. In combination 1, the transmitter transmitted pictures of 256 lines by 256 pixels with sub-sampling ratio of 2 in both horizontal and vertical directions. The receiver displayed what it received without interpolating the "missing pixels". Thus the picture displayed was one fourth of the size of the picture transmitted. With a delay of 0 second the time taken to transmit a colour picture of 256 lines by 256 pixels was about 0.25 second. Compared with the figures in Table 1, it can be seen that the transmitter can output at a similar bit rate but the picture update rate was four times faster when the sub-sampling coding was used.

In combination 2, the transmitter transmitted pictures of 256 lines by 256 pixels with a pixel subsampling ratio of 2 in both horizontal and vertical directions (the same as in combination 1). The receiver not only displayed the received pixels but also interpolated the "missing pixels" by simply duplicating the received pixels three times. Thus the displayed picture had the same size as the transmitted picture although some detail was lost. With a delay of 0 second the time taken to transmit a colour picture of 256 * 256 was about 0.64 second. The transmission rate was lower than that in combination 1 because the receiver had more computation to do and so the transmitter was not allowed to transmit as fast as it can. When the receiver was busy doing computation, the minipacket sent by the transmitter would be turned back and the transmitter would be requested to re-transmit the minipacket. That is why the maximum output rate was lower. It is obvious that
when the receiver is too slow in accepting minipackets, some minipackets will be lost after a pre-fixed number of re-tries (hardware and software re-tries). But this situation rarely occurs because the receiver is slightly faster than the transmitter and the time taken to complete a fixed number of re-tries is quite long. The only chance it might happen is when several transmitters send data at full speed to the same receiver.

In combination 3, the transmitter sent pictures of 128 lines by 128 pixels. At the receiver the pixels received were duplicated as is shown in Fig.7.3 case (4). Thus the displayed picture was twice the size in both dimensions. The time taken to transmit the picture was about 0.6 second.

Experiments showed that quality of pictures reconstructed from above combinations was sufficient for transmission of head and shoulders images in a video conference.

8.3.3. Video Transmission with Conditional Replenishment

In this transmission mode, only pixel blocks which have changed significantly (difference being greater than a certain threshold) from the previous frame are transmitted. To reduce the calculation overhead, only 7 pixels out of 13 pixels in a block are used in the calculation (see Chapter 7).

Typical images of head and shoulders were used for the experiments. For a threshold of 13, statistics of block changes were collected from a sequence of 100 images transmitted. The result showed that on average 53% of the blocks (pixels) have changed significantly and were transmitted. For a threshold of 20, statistics of block changes were collected from a sequence of 133 images transmitted. The result showed that on average 26.6% of the blocks were taken to have changed significantly from the previous frame and were transmitted. For a threshold of 26, statistics were collected from a sequence of 149 images. The result showed that 19.3% of the blocks were taken to have changed significantly from the previous frame and were transmitted. Fig.8.4 shows the rough relationship between the threshold and percentage of changed blocks in a frame. When the threshold is too small, some blocks will be transmitted unnecessarily due to the noise in the video signals. When the threshold is too large, the quality of reconstructed picture will become unacceptable. Note that the threshold values referred above are used when 7 pixels in a block are involved in the calculation. If all 13 pixels are used in the
calculation, similar results can be obtained when the threshold values used are the corresponding threshold values multiplied by a factor (13/7). For example, the threshold of 26 referred above is corresponding to a threshold of 48 when all pixels are involved in the calculation.

The time taken to transmit a colour image of 256 lines by 256 pixels varies from 1.28 seconds to 1.6 seconds depending on the amount of changed blocks. The more the changed blocks, the longer the time taken to transmit the image. There was no added delay between packet transmissions. The slow image updating rate is due to the computation power required for conditional replenishment coding.

8.3.4 Multipoint Communication

To do this experiment three transmitters (TX1, TX2 and TX3) and three receivers (RX1, RX2 and RX3) were connected to a local CFR. TX1 established associations with RX2 and RX3, TX2 with RX1 and RX3, and TX3 with RX1 and RX2. For simplicity, take TX1 and RX1 for example. TX1 sent images to RX2 and RX3 sequentially over separate video channels which probably had different channel attributes. TX1 sent a whole image first to the receiver, to which the association was established earlier, then to the other receiver. So there was a quiet gap at the receiver when the transmitter was not sending data to it. RX1 received video data
from TX2 and TX3 and displayed them on one monitor simultaneously. In this situation, the video channel attributes should be selected so that the two image display windows at the receiver do not overlap, otherwise the video data from these two channels will overwrite each other and a sensible display can not be obtained. The appropriate channel attributes can be selected by "selecting" the receiver and checking what channels have already existed on the receiver before establishing a new channel to the receiver.

The experiments showed that a receiver can accept data from two transmitters which were sending data as fast as possible, i.e., with a transmission delay of 0 second. The receiver is not twice as fast as the transmitter. This data handling ability is attributed to the buffering in the receiver and hardware and software re-transmissions in the transmitters.

8.4 Experiments over Portals

The video system in these experiments was configured as shown in Fig.8.3. Several experiments have been conducted: (1) One way video from one client CFR to another; (2) Two way video from TX1 to RX2 and from TX2 to RX1; (3) Two way video with other traffic. Since the most important factor concerned in these experiments is data rate, and the transmitter has maximum data rate output when no compression is used, results of the video transmission experiments without data compression are presented in this section.

8.4.1 One Way Video

In this experiment, only one video channel went through portals, either from TX1 to RX2 or from TX2 to RX1. The results showed that the transmitter can transmit data at the maximum data rate (around 1.3 Mbps) without data losses and no difference on the receiver displays was observed compared with that of the video transmission over a local CFR. This is as expected because the portals have a one-way throughput around 1.5 Mbps as reported in [Siddiqui 89b].

8.4.2 Two Way Video

In this experiment, two video channels were set up through the portals, one from TX1 to RX2 and the other from TX2 to RX1. The results showed that when both
transmitters sent data with a minipacket transmission delay of 64 µs, data losses were seldom observed and when both transmitters sent data at the maximum data rate, i.e., with a transmission delay of 0 second, many data losses were observed and the picture quality at the receiver was quite poor. This showed that the portal’s one way throughput is lower when it is active in both directions which confirms Siddiqui’s observation [Siddiqui 89b].

8.4.3 Two Way Video with Other Traffic

In this experiment, two video channels, one voice channel and one "monitoring" data channel were set up through the portals. Voice data are more sensitive to packet delay and losses, hence they were transmitted as high priority traffic. The data rate of the voice channel is fixed at 64 kbps in both directions. The data rates of the video channels and the "monitoring" data channel were adjustable. The experiment results showed that when the total data rate of the traffic through the portals was within the limit of the portal’s throughput, around 1.2 Mbps, packet losses were seldom noticed and no interference among different traffic was observed. But when the portals were overloaded, the results were unpredictable, sometimes the portals stopped functioning totally, sometimes one or more channels were stopped.

8.5 Experiment over Portals and RAMPs

Due to time limitation and the unavailability of an ISDN link at the later stage, no proper experiment of this type has been carried out. The only experiment was done during the ITEX’88 exhibition at Barbican Exhibition Centre, London, in November 1988. The setup of the demonstration is shown in Fig.8.4. One office was set up at LUT and the other at Barbican Exhibition Centre, and they communicated via voice and video.

At that time, the implementation of the portal was not mature and its throughput was quite low. Video and voice transmissions were achieved over the network but not very reliable due to the portal breaking down. In order to keep the connection as long as possible, the data rate of the video channel was kept very low, at around 150 kbps.
Fig. 8.4 Setup of the Demonstration

At a later stage, the performance of the portals and video codec were improved, but no proper experiment has been carried out. Since the RAMP has a larger throughput than the portal and so long as the ISDN link has sufficient bandwidth available, it is expected that similar results to that of the experiments over portals only will be obtained, although the transmission delay will be slightly longer.

8.6 Summary

In this chapter the experiments and results of video communication over the Unison network in a multimedia environment have been presented. Satisfactory results were obtained so long as the network was not overloaded. Since video is transmitted in packets (cells) and the transmission data rate is variable, the video services can be integrated with other traffic easily and network bandwidth can be used efficiently.

Since there is not any call blocking mechanism implemented in the network, and the network bridging components function unpredictably when they are overloaded, a user must be careful in setting up a call with the appropriate data rate.
Chapter 9

Discussion and Conclusions

9.1 Introduction

The work described in this thesis is an attempt to provide a common workspace in a teleconferencing system for participants to work in and to provide reliable and flexible video services along with other services over the Unison network. The main work carried out is as follows:

(1) The Unison Electronic Blackboard System (EBS) has been designed and implemented based on Acorn A-500 workstations connected to the CR-based Unison network;

(2) The Unison video codec has been designed and constructed based on transputers;

(3) A set of video protocols has been implemented in a concurrent language, OCCAM, to provide reliable and flexible services over the Unison network;

(4) Experiments on transmission of multimedia services have been carried out over the Unison Network.

The next section gives some discussion and proposals for further work. The last section summarizes the main work and concludes the thesis.

9.2 Discussion and Proposals for further Work

Electronic Blackboard System (EBS)

The EBS was designed and implemented based on the A-500 workstation running the Arthur operating system over the CR-based Unison network. It is a useful
facilit for person to person communication in a multi-media environment. People showed much interest in using it. Unfortunately, as Unison moved to the CFR-based network soon after the full implementation of the EBS, no further use was made of the EBS. Consequently, there was no time and chance to obtain feedback on the usability of the EBS, and to further improve the system.

Since the EBS is required to operate in a many-to-many mode, there is a question of how data should be distributed. The mechanism implemented in the EBS is that every blackboard establishes associations with all the other participating blackboards and maintains a list of the associations. Each blackboard could then explicitly transmit data to everyone on the list. One alternative is for the network to provide a multicast server and each blackboard then talks to the server and the server distributes the data to every participants.

Due to the lack of time and effort, an EBS has not been implemented on the CFR-based network although it is desirable to have the EBS in the multimedia office to complement the video and voice services.

The EBS on the CR-based network was implemented using the WIMP system under the Arthur operating system. The WIMP system is a powerful window system, but it does not support networking. There is currently much interest in networking window systems, such as the X window system [Scheifler 86]. Unlike most window systems, the base system in the X window system is defined by a network protocol: asynchronous stream-based inter-process communication replaces the traditional procedure call or kernel call interface. An application can utilize windows on any display in a network in a device-independent, network-transparent fashion. Interposing a network connection greatly enhances the utility of the window system, without significantly affecting its performance. Recently, Olivetti Research Laboratory implemented the X window system under the Tripos operating system [Knight 82] using the RPC mechanism on the CFR. So it is sensible that an EBS on the CFR-based network should be implemented, if ever, using the X window system.

User Interface

The operation control of the video codec is via Tripos command lines entered from the A-500 workstation. It is not user-friendly because a user must know the values
of various parameters required to set up a video channel. It is desirable to have a user interface based on some kind of window system so that a video codec can be controlled by just moving and clicking the mouse. There is a great amount of interesting research and development work to be done in the area of user interface.

There are various service resources in the office, such as video and voice. They should be controlled under one common interface. Since the EBS is also based on workstation there is the question of how to share the workstation monitor. And a management mechanism needs to be designed to control who is allowed to do what and when. The interface should be user-friendly: flexible, simple and natural to use.

Again, the X window system seems to be a sensible tool to use to implement the user interface.

**Video Coding**

The introduction of video service is always confronted with economic problem because of the vast amount of data required to transmit real time video. To economically transmit video, some kinds of compression techniques are required. In the CFR video codec, two simple compression schemes, sub-sampling and block conditional replenishment, were implemented in software. Because of the computation power required by the coding, the picture update rate was not increased in the case of block conditional replenishment although data rate was reduced significantly. It seems that the only practical approach to obtain a high picture update rate with low data rate is to implement compression in hardware.

A study has been conducted into the hardware implementation of the block conditional replenishment and revealed a relatively simple and effective approach. But because of time restriction of the project, it was not implemented. The design idea is as follows: the odd field is captured into video memory in real time. Then during the time of the even field, the captured video data are compared with the data in the reference video memory block by block. If a block is taken as having changed significantly from the corresponding block in the reference video memory, it is put into a FIFO (First In First Out) buffer to be transmitted. So the maximum update rate that can be achieved is 25 fields per second. According to [Chin 86], for a typical head and shoulders image sequence about 10% of the blocks needs
Chapter 9 Discussion and Conclusions

updating on average. So if a network bandwidth of 1 Mbps is available, around 10 fields per second can be transmitted. This would be an interesting experiment.

Another alternative is to use some specific DSP (Digital Signal Processor) for intraframe coding, such as IMS A121, a 2-D discrete cosine transform image processor from INMOS. It should be noted that it is always better to follow the international standards on video coding whenever possible.

Network

While implementing and experimenting with the EBS and video transmissions, it was strongly felt that the network should provide a multicast server and call blocking mechanism. Both topics were considered in the Unison project and a prototype of the former was developed although it has not been put into use.

There are several advantages in having a multicast server on the network for multipoint multimedia communications. Firstly, it is simple and easy to implement an application with the server. A station does not need to establish several associations with other stations concerned and keep a list of them rather a single association with the server is needed. Secondly, bandwidth could be reduced. Thirdly, application speed can be increased without repeating transmissions of the same data to several destinations.

Without call blocking mechanism implemented in the network, a new call could be established when there is not enough bandwidth available on the network, and data losses could occur to all calls existing in that part of the network used by the new call. The worse case is that the part of the network concerned could crash totally and all calls existing in that part of the network will be stopped. To prevent this from happening, it is suggested that when a call is set up via the secretary, the average data rate requirement of the new call should be reported to the secretary, the secretary will check the bandwidth available in the network and will grant the call if there is enough bandwidth for the new call, otherwise the new call will be rejected.
9.3 Summary

The Electronic Blackboard System

In response to the requirement of a common work space for people communicating from different sites, just like a blackboard used when people are discussing together, the EBS has been developed over the CR-based Unison network. The EBS is shared among participants. All participants see exactly the same graphics on their own local workstation monitors and they can enter graphics on their own workstations via mice or graphpads, and the graphics are distributed to all other participants.

The EBS is based on the A-500 workstations connected to the network and made use of the WIMP system under the Arthur operating system. A blackboard is a window on the workstation monitor and the background colour is selectable by a user. Ten facilities are provided. They are free-hand drawing, line drawing, rectangle drawing, circle drawing, pattern filling, text input, erase, clearing screen, saving and printing graphics. The colour of graphics is selectable and the thickness of free-hand drawing, line, rectangle, and circle is also selectable. There are six patterns for a user to select in pattern filling mode. For text input, the start position of the text on the blackboard and the characters' ASCII codes are transmitted. For free-hand drawing and erasing, the cursor movement is sampled as fast as possible and the coordinates of the samples are transmitted. For a circle, the coordinates of the centre and the radius of the circle are transmitted. For a rectangle, the coordinates of the two opposite corners of the rectangle are transmitted. Of course, the colour and thickness information of the graphics are also sent along with these graphics data. For pattern filling, only the pattern number and coordinates of a point in the area to be filled are transmitted and the pattern is generated at the receiver according to the pattern number.

The participants are associated with OPEN and OPENACK protocol. A blackboard establishes associations with all other participants and multi-point service is realized by explicitly sending graphics data to every blackboard associated. Graphics data are transmitted in basic blocks.
The EBS is a useful facility in a multimedia office and is interesting to use, although further work is needed to make it more user-friendly.

The Video System

The architecture of the video system is heavily based upon the Inmos T414 Transputers. The complete video system comprises a transmitter and a receiver, each having a separate CFR station. The transmitter is responsible for capturing video data, storing them in video memories, coding and transmitting them. The receiver is responsible for receiving data from the network, decoding them, putting them into video memories and displaying them on a monitor. Since the video memories and control registers are memory mapped into the transputers’ memory space, the video system is very flexible.

A transmitter can set up more than one association with more than one receiver, and a receiver can have more than one association with more than one transmitter. So a transmitter can send images to several receivers sequentially and a receiver can receive video data from several transmitters and display them on one monitor at the same time. In this way multi-point communications can be achieved with one transmitter and one receiver at each site.

The image size, position in a frame and resolution at both the transmitter and the receiver are selectable when a video channel is initiated from a workstation. the maximum picture size (resolution) is 512 lines by 512 pixels with 16 bits per pixel. Monochrome image transfer is achieved by transmitting the Y component only. Slow scan video and still image transfer services are available.

Delay between minipacket transmissions is also selectable to control the output data rate. The maximum data rate is about 1.33 Mbps when the delay between minipacket transmissions is 0 second.

Data flow control is performed in three ways. The first is that hardware and software re-transmissions are implemented at the transmitter to avoid unnecessary packet losses when the receiver is busy momentarily. The second way is for a transmitter to expect an "alive" minipacket from the receiver after a certain time. The "alive" minipacket contains packet loss statistics obtained from the packet sequence numbers sent by the transmitter along with the packets. The transmitter will adjust its output data rate according to the statistics. If the "alive" minipacket
does not arrive within a certain time, the transmitter will assume that there is something wrong with the network components or the receiver and will stop sending data over that channel to avoid bandwidth wastage. The third is that a 32 kbytes buffer is provided at both the transmitter and the receiver to make use of parallel processing power of the transputers and thus to increase transmission rate.

The pixel subsampling ratio and threshold for the block conditional replenishment are also selectable. For a threshold of 48 for a block of 13 pixels, experiments on typical head and shoulders type image sequence show that about 20% of the blocks have changed significantly from the previous frame and need transmission, and to transmit a colour image of 256 lines by 256 pixels takes about 1.5 seconds. The picture update rate does not increase with the block conditional replenishment coding because the transputer spent a lot of computation power on the coding although the data rate is reduced significantly. Maximum picture update rate is achieved with subsampling coding. With a pixel subsampling ratio of 2 horizontally and 2 vertically, a colour image of 256 lines by 256 pixels takes about 0.64 second to transmit.

The video system is packet loss tolerant in the sense that any packet loss will not cause the receiver to lose synchronization. Lost data are substituted by the corresponding pixel data in the previous frame. Since there is a strong correlation between frames, it is sometimes difficult to perceive the packet loss effect.

Although the video codec was meant for video transmission over the network, it can be used as a stand alone video simulation system. Simulation work can be carried out using the TDS on a PC. A user can program the system to capture a frame of image, to process the image using a video coding algorithm under investigation and to display the processed image. Since transputers are used in the system it can be extremely useful when some sort of parallel algorithm is to be investigated.

Since the video coder is based on one single transputer, it is flexible and easy to control although the picture update rate is limited by the processor speed. To increase the picture update rate, other architecture has to be used. Nevertheless, the video system described in this thesis meets the main objective of the project: to provide flexible video services which can make use of the network bandwidth.
efficiently and which can adjust its output data rate according to the loading on the network.

**Experiments of multimedia communication**

Experiments on packet video, packet voice and computer data transmission over a local CFR showed that there is almost no packet loss and the traffic does not interfere with each other. When the traffic goes through one or more portals satisfactory results are achieved so long as the portals are not overloaded.

**9.4 Conclusions**

A common work space for people at different places to work in is becoming increasingly important for effective communications. The work described in Chapters 3 and 4 shows that it is feasible to build an electronic blackboard system based on networked office workstations.

The work on the video codec shows that a video codec based on microprocessors is very flexible although the picture update rate is limited. Transputers are strong candidates for this purpose because of their high operation speed and ease of interconnection to each other.

The video codec described in this thesis is a variable bit rate (VBR) system. The experiments show that VBR video transmission over an ATM network is of advantage in terms of efficient utilization of network bandwidth and easy integration with other traffic.

Finally, the work described in this thesis demonstrates that the Unison network architecture is suitable for multimedia applications.
REFERENCES

Acorn Computers Limited, 1987

[Adams 84] C. J. Adams
"Universe Network Protocol Architecture"

[Ades 87] Stephen Ades
"An Architecture for Integrated Services on the Local Area network"

[Ambrosio 85] R. Ambrosio et al
"Error Correction Codec for Video Conference System"

[Burren 89] J. W. Burren and C. S. Cooper
"Project Universe-An Experiment in High-Speed
Computer Networking"

[Chin 86] H. S. Chin
"Transmission of Image Data over Computer Network"
Internal report, Department of Electronic and Electrical

[Chin 88] H. S. Chin et al
"Statistics of the Video Signals for Viewphone Type Picture"
International Workshop on Packet Video,
Torino, Italy, 8-9th September 1988.

[Clark 86] P. Clark et al
"Unison - Communications Research for Office Applications"
References

[Clarke 84] R. J. Clarke
"Transform Coding of Image"

[Costigan 78] D. Costigan
"Electronic Delivery of Documents and Graphics"

[Coudreuse 88] J. P. Coudreuse
"ATM: State of Definition and Major Issues"
International Workshop on Packet Video,
Torino, Italy, 8-9th September 1988.

[Decina 88] M. Decina
"Broadband ISDN, Whither ATM?"
International Workshop on packet Video,
Torino, Italy, 8-9th September 1988.

[Douglas 89] S. Douglas
"Why Travel When You Can Call?"
Telephony (USA), 3 April 1989.

[Freer 88] John Freer
"Computer Communications and Networks"

[Gallagher 86] I. D. Gallagher
"Multi-service Networks"

"Variable Bit Rate Transmission of Television Signals"
[Girling 82] C. Gray Girling and Chris Cooper
"Proposed Ring Software Interface for BBC Micro"
Project Universe Demonstrations Working Group Report,
November 1982.

[Gray 84] R. M. Gray
"Vector Quantization"

"The Unison Demonstration Electronic Office"
Proceedings of EUSIPCO-88, Fourth European Signal

[Hamilton 84] Kenneth Graham Hamilton
"A Remote Procedure Call System"


[Hopper 86] A. Hopper et al
"Local Area Network Design"

[Inmos 88a] Inmos Limited, Edited by C. A. R. Hoare
"Occam 2 Reference Manual"

[Inmos 88b] Inmos Limited
"Transputer Development System"

[Inmos 89] Inmos Limited
"The Transputer DataBook"
References

[Knight 82] B. J. Knight
"Portable System Software for Personal Computers on a Network"

"1.5 Mb/s Interframe Codec For video Teleconferencing"
Review of the Electrical Communication Laboratories,

[Lazar 87] Aurel A. Lazar and John S. White
"Packetized Video on MAGNET"

[Lee 86] B. S. Lee
"Image Transmission over the Cambridge Ring"

[Leslie 83] I. M. Leslie
"Extending the Local Area Network"
Technical Report, No.43,
University of Cambridge Computer Laboratory, 1983.

[Littlewood 87] M. Littlewood et al
"Evolution toward an ATD Multi-service Network"

[Mallappa 88] B. Mallappa and N. R. Cox
"Still-Frame Image Transmission Using DPCM/Interpolation Coding"

[Mcgregor 84] J. Mcgregor and A. Watt
"The Art of Microcomputer Graphics for the BBC Micro/Electron"
References

[Murphy 87] B. J. Murphy
"Video Transfers on the Unison Network"
Internal Report, Department of Electronic and Electrical
Engineering, Loughborough University, October, 1987

[Murphy 88] B. J. Murphy et al
"Multi-media Applications over a Wide Area Network"
UK IT 88, Conference Publication,
University College Swansea, 4-7 July 1988.

[Murphy 89] B. J. Murphy
"Multimedia Services in a Distributed Office"
To be submitted as a PhD Thesis, Loughborough University

[Musmann 79] H. G. Musmann
"Predictive Image Coding"
Advances in Electronics and Electron Physics, Suppl. 12,

[Nicol 83] R. C. Nicol and T. S. Duffy
"A Codec System for Worldwide Videoconferencing"
Professional Video, November 1983.

[Nomura 88] M. Nomura et al
"Layered Packet Loss Protection for Variable Rate Video
Coding Using DCT"
International Workshop on Packet Video, Torino, Italy,

[Ohta 88] M. Ohta et al
"Error Recovery Procedures for Packetized Interframe Video Coding"
International Workshop on Packet Video, Torino, Italy,
<table>
<thead>
<tr>
<th>Reference</th>
<th>Author(s)</th>
<th>Title</th>
<th>Publication Details</th>
</tr>
</thead>
</table>
References

[Siddiqui 89a] M. H. Siddiqui et al
"Performance of Local and Remote Bridge Components in Project UNISON: a 50 Mbits/s ATM Network"

[Siddiqui 89b] M. H. Siddiqui
"Performance Measurement Methodology for Integrated Service Network"

[Srinivasan 85] Ram Srinivasan and K. R. Rao
"Motion Compensated Coder for Videoconferencing"


[Tennenhous 87] D. Tennenhouse et al
"Exploiting Wideband ISDN: The Unison Exchange"

[Verbiest 86] W. Verbiest
"Video Coding in an ATD Environment"

[Yamaguchi 85] H. Yamaguchi
"Vector Quantization of Differential Luminance and Chrominance Signals"
Appendix 1

Cambridge Ring Point-to-Point Bandwidth Calculation

Although "Cambridge Ring" is used in the title of this appendix, the discussion and conclusion applies to both Cambridge Ring (CR) and Cambridge Fast Ring (CFR). "Cambridge Ring" is used here to mean a set of protocols used in CR and CFR. As far as client rings are concerned, if channel mode is not used in CFR, the only difference between CR and CFR is that CFR uses larger minipacket and faster clocking rate. In the following discussion, it is assumed that the channel mode is not used in CFR.

The system bandwidth (SysBW), the total data bandwidth available on the network, is calculated as follows:

\[ \text{SysBW} = \frac{\text{Number of data bytes on ring}}{\text{Total number of bytes on ring}} \times \text{clocking rate} \]

Provided that the gap between slots is small (less than 5 bits), then SysBW is roughly independent of the number of slots on the ring.

The point-to-point bandwidth (PPBW) is dependent on the number of slots on the ring for the following reason. Consider a ring with N slots, when a station transmits it must wait for the minipacket to return (N slot times plus gap time) and then wait a further one slot time before transmitting again. This wait is a feature of the Cambridge Ring to prevent bandwidth hogging. Thus the minimum time between transmissions is \((N + 1)\) slot times (Plus the gap time). Assuming the gap to be small then

\[ \text{PPBW} = \frac{\text{SysBW}}{(N+1)} \]
This assumes that no other stations are using the ring. If M stations are transmitting as fast as the ring will allow, then the PPBW available to each is as follows:

$$PPBW = \frac{SysBW}{(M+N)} \quad \text{where } M>1$$

Note that two stations may transmit with the maximum PPBW without interfering with each other.

For a typical CFR, the clocking rate is about 50 MHz and in a typical situation has 2 slots with small gap, then

$$SysBW = \frac{(2*32*50)}{(2*38)} = 42 \text{ Mbps}$$

There are maybe many stations connected to a CFR. But only a few stations, say 3 stations, are transmitting as fast as the ring will allow, then the PPBW available to these stations is as follows:

$$PPBW = \frac{42}{(3+2)} = 8.4 \text{ Mbps}$$

It should be noted that the PPBW calculated as above is the maximum point-to-point bandwidth allowed by the Cambridge Ring protocols. In practice, the point-to-point bandwidth might be limited by the hardware and software implementations of stations and interfaces.
Appendix 2

Circuit Diagram of the ADC Board
Appendix 3

Circuit Diagram of the Video Transmitter Board
ALL TTL AND SIL RAM HAVE 0.1NF CAP ONBOARD SOCKET
Appendix 4

Circuit Diagram of the Video Receiver Board
ALL VRAM DEVICES AND
TTL CHIPS HAVE 0.1uF
CAP ONBOARD SOCKET

RECEIVER BOARD
PROJECT UNISON
T. ROGERS / S. J. LU
APRIL 1989
PAGE 3 OF 3
Appendix 5

Circuit Diagram of the DAC Board
Appendix 6

Instructions for Operation of the Video Codec

The software running in the video codec described in Chapter 8 is very complex and is not listed here. The simple program in this appendix shows how images are sent from the transmitter to the receiver through a transputer link (link 2). The intention of this appendix is to demonstrate how the registers in the codec are controlled.

The procedure Transmitter() runs in the transmitter control transputer and Receiver() in the receiver control transputer.
PROC transmitter()
CHAN OF [7]INT c.tra: --transmission link
PLACE c.tra AT 2: --link 2
INT i, j, line.num, pixel.num:
[]INT16 ty RETYPES tx: --pixel block
[1] INT enable: --grabber enable register
PLACE enable AT $08040000:
PLACE status AT $08080000:
[512*1024]INT16 frame: --frame memory
PLACE frame AT $08000000:
WHILE TRUE
SEQ -- to capture a frame
  enable[0] := 4 --enable grabber
  j := 0
  WHILE j = 0 --till status = 1
  SEQ
    j := status[0] \1
    enable[0] := 0 --disable grabber
  SEQ i = 0 FOR 256 --to transmit a frame
  SEQ
    line.num := i*1024
    SEQ j = 0 FOR 39 --the odd field
    SEQ
      pixel.num := j*26
      ty[0] := frame[line.num + pixel.num]
      ty[1] := frame[line.num + (pixel.num + 2)]
      ty[2] := frame[line.num + (pixel.num + 4)]
      ty[3] := frame[line.num + (pixel.num + 6)]
      ty[4] := frame[line.num + (pixel.num + 8)]
      ty[5] := frame[line.num + (pixel.num + 10)]
      ty[6] := frame[line.num + (pixel.num + 12)]
      ty[7] := frame[line.num + (pixel.num + 14)]
      ty[8] := frame[line.num + (pixel.num + 16)]
      ty[9] := frame[line.num + (pixel.num + 18)]
      ty[10] := frame[line.num + (pixel.num + 20)]
      ty[12] := frame[line.num + (pixel.num + 24)]
      ty[13] := ((INT16 i) << 1) + ((INT16 j) << 9) --pixel address
      c.tra!tx
    . SEQ
      j = 0 FOR 39 --the even field
    SEQ
      pixel.num := (j*26)+262144
      ty[0] := frame[line.num + pixel.num]
      ty[1] := frame[line.num + (pixel.num + 2)]
      ty[2] := frame[line.num + (pixel.num + 4)]
      ty[3] := frame[line.num + (pixel.num + 6)]
      ty[4] := frame[line.num + (pixel.num + 8)]
      ty[5] := frame[line.num + (pixel.num + 10)]
      ty[6] := frame[line.num + (pixel.num + 12)]
      ty[7] := frame[line.num + (pixel.num + 14)]
      ty[8] := frame[line.num + (pixel.num + 16)]
      ty[9] := frame[line.num + (pixel.num + 18)]
      ty[10] := frame[line.num + (pixel.num + 20)]
      ty[12] := frame[line.num + (pixel.num + 24)]
      ty[13] := ((INT16 (i*2)) PLUS 1 (INT16)) + ((INT16 j) << 9)
      c.tra!tx
PROC receiver()
[262144]INT x: --frame memory
PLACE x AT $08000000:
[50]INT yy:
PLACE yy AT $18000000:
CHAN OF [7]INT c.rec: --receive link
PLACE c.rec AT 6: --link2
[]INT16 ry RETYPES rx:
[262144*2]INT16 frame RETYPES x:
PLACE frame AT $08000000:
INT k, i, j, h, li, pi:
SEQ
yy[0]:=(0*256) + 10 -- set VSC register values
yy[0]:=(1*256) + 0
yy[0]:=(2*256) + 25
yy[0]:=(3*256) + 0
yy[0]:=(4*256) + 155
yy[0]:=(5*256) + 0
yy[0]:=(6*256) + 158
yy[0]:=(7*256) + 0
yy[0]:=(8*256) + 1
yy[0]:=(9*256) + 0
yy[0]:=(10*256) + 37
yy[0]:=(11*256) + 0
yy[0]:=(12*256) + 37
yy[0]:=(13*256) + 1
yy[0]:=(14*256) + 56
yy[0]:=(15*256) + 1
yy[0]:=(16*256) + 4
yy[0]:=(17*256) + 0
yy[0]:=(18*256) + 0
yy[0]:=(19*256) + 0
yy[0]:=(20*256) + 0
yy[0]:=(21*256) + 0
yy[0]:=(22*256) + 0
yy[0]:=(23*256) + 0 --2:interlaced
yy[0]:=(24*256) + 0
yy[0]:=(25*256) + 48
yy[0]:=(26*256) + 0
yy[0]:=(27*256) + 0
yy[0]:=(28*256) + 16
yy[0]:=(29*256) + 0
yy[0]:=(30*256) + 0
yy[0]:=(31*256) + 0
yy[0]:=(32*256) + 0
yy[0]:=(33*256) + 0
SEQ a=0 FOR 262143 --set background colour
x[a]:=SD200
WHILE TRUE --receive and store video data
SEQ
c.rec ?rx
i:=INT (ry[13] \511(INT16)) --get pixel address
j:=INT (ry[13] >>9)
li:=(i TIMES 1024) PLUS (j TIMES 26)
pi:=2
SEQ h =0 FOR 13 --store data into frame memory
frame[[(li PLUS (h TIMES pi))]:=ry[h]