Design and Implementation of a Collision Avoidance Multiple Broadcast Tree Network [1]

Technical Report 91-51

Hung Khei Huang, Tatsuya Suda
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717
Phone: (714) 856-5474
FAX: (714) 856-4056

Abstract

Packet collisions and their resolution create a performance bottleneck in random access LANs. A hardware solution to this problem is to use collision avoidance switches [1, 2]. Collision avoidance switches allow the implementation of random access protocols without the penalty of collisions among packets.

In this paper, we describe a design and implementation of a local area network architecture based on collision avoidance, called the Collision Avoidance Multiple Broadcast (CAMB) tree network. Our implementation includes CAMB tree switches, station/network interface boards, and support of transport protocols. Our implementation of the CAMB tree network follows the protocol layering architecture of the IEEE 802 local area networks.

[1]This material is based upon work supported by the National Science Foundation under Grant No. NCR-8907909. This research is also in part supported by University of California MICRO program and Omron.

1. Introduction

Advantages of random access protocols such as ALOHA and CSMA/CD include simplicity and ease of implementation at the stations in a network. Random access protocols also exhibit small transmission delays under light traffic conditions. However, they have a performance bottleneck under heavy traffic conditions. In that situation, a large number of collisions occur, resulting in low channel utilization. Channel capacity is wasted in the transmission of collided packets [3].

In order to solve this performance bottleneck in random access protocols, a new network architecture based on collision avoidance, called collision avoidance tree network, has been proposed and investigated [2,4,5,6] by the authors. The tree network uses collision avoidance switches. These switches allow implementation of random access protocols without the penalty of collisions among packets.

Collision avoidance can be implemented with very little circuitry. Implementations of collision avoidance hardware are proposed in [1,4,6,7]. Various station and switch protocols for collision avoidance networks are discussed in [2,4,5,8]. An experimental broadcast star network is discussed in [9,10]. In [11], synchronous operation of a broadcast star is considered, and performance is analyzed. An exact analysis is developed for the network with an infinite number of stations, and an approximate analysis is developed for the network with a finite number of stations. Papers [12,13] assume a broadcast star that operates under asynchronous mode, where transmissions of a packet are not confined to the beginning of slots. They also model a broadcast star as a polling system, and develop an approximate analysis.

Suda et. al. extended the idea of collision avoidance to a tree network [2], called the Collision Avoidance Multiple Broadcast,(CAMB) tree network. In a CAMB tree network, collision avoidance switches are used to resolve contentions among packets. The important feature in a switch is that there is no collision. When two or more packets contend for the right to use the switch, it is always guaranteed that one packet will be successfully transmitted. Thus, no channel time is wasted in the transmission of collided packets, eliminating the main disadvantage of a traditional random

access network. In the CAMB tree, the switches are organized into a tree topology. The switches are the internal nodes, and the stations are the leaves of the tree. A packet reaches its destination by climbing the tree and being broadcast by the switch that is the root of the minimal subtree containing both the source and the destination stations. Concurrent transmissions are possible in the CAMB tree network, due to the segmented nature of the network.

Station and switch protocols for the CAMB tree are studied in [4,5]. A switch design based on photonic devices and a simulated performance analysis for this network are given in [4,6]. A performance study on the CAMB tree network through theoretical analysis and simulations is given in [14. 15]. In comparison with a broadcast star, the CAMB tree shows a better performance.

In this paper, we present a design and implementation for the CAMB tree network. In section 2, a description of the the CAMB tree network is presented. A design and implementation of the CAMB tree network is given in section 3. In section 4, concluding remarks are given.

2. The CAMB Tree Network

The CAMB tree network consists of collision avoidance switches organized in a tree topology, where each switch is a node in the tree. The stations are the leaves of the tree. Each transmission line consists of uplinks and downlinks. A station or a switch uses the uplink to send packets and receives packets through the downlink. A station sends a packet as soon as one is available. A packet reaches its destination by climbing the tree and being broadcast by its proper ancestor. The switch that is the root of the minimal subtree containing both the source and the destination stations is the proper ancestor (Fig.1). Note that each switch on the tree is responsible for transmitting a packet to its parent switch and for broadcasting the packet to its children switches (if the switch is the proper ancestor of the source and the destination of the packet). Fig.2 gives an example of a transmission in a CAMB tree network. Station 1 transmits a packet. Switch E, which is the proper ancestor of this packet, sends it to the parent switch G, and broadcasts it to its subtree (switch A and B). If the switch is not the proper ancestor a packet is only passed to the parent switch (in Fig.2 switch A passes the packet to its parent switch E only).

CAMB Collision Avoidance Switch Protocol

A CAMB switch is connected to children switches (or stations) by uplinks and downlinks. It is also connected to a parent switch by an uplink and a downlink. A switch can receive packets either from its children's uplinks or from its parent's downlink. When a switch receives packets from its children's uplinks, the switch executes the following protocol:

1. The switch selects one of the incoming packets and checks the header of the selected packet to see if it is the proper ancestor of the packet.

2. IF the switch is not the proper ancestor, the packet is transmitted to the parent uplink,

3. ELSE

   (3.1) IF there is a packet from the parent downlink being broadcast, the packet from a child is discarded (this is called broadcast preemption).

   (3.2) ELSE, if there is no packet from parent switch, the packet from a child is broadcast to the children's downlinks and to the parent uplink.

When a switch receives a packet from its parent's downlink, the switch executes the following protocol:

1. IF the switch is idle, the packet from the parent is broadcast to the children's downlinks.

2. ELSE if it is busy broadcasting a packet from a child, this packet from a child is aborted and the packet from the parent is broadcast to the children's downlinks.

By the switch protocol, priority is given to the parent's downlink over the children's uplinks. This is an efficient priority scheme for the following reasons. A packet is broadcast by its proper ancestor to both the source and the destination stations. Successful reception of a packet by the source implies that the packet is also successfully received by the destination station. Therefore, proper ancestor switch gives priority to a packet coming from the parent [5]. This prevents a situation where the source station receives the broadcast but the destination does not Another

factor indicating parent switch priority is that a packet coming from the parent is likely to have been in the network for a longer time than a packet coming from the children.

Station Protocol

The CAMB station protocol is based upon the station monitoring its downlink for the broadcast of its packet. It is very simple and similar to random access protocols such as ALOHA. The CAMB station protocol is as follows:

1. A station transmits a packet as soon as one becomes available, starting at say, time t.
2. The station monitors its downlink for the broadcast of its packet.
3. IF the station does not see the start of its packet by time $t + R_{pa}$, where R_{pa} = round trip propagation delay time between the station and the proper ancestor,
4. THEN it retransmits the packet immediately,
5. ELSE (the start of the packet is seen within this time)
   (5.1) IF it sees the broadcast of the packet truncated by the broadcast of another packet (abortion), then it retransmits the packet immediately,
   (5.2) ELSE the station sees the broadcast of the whole packet (the transmission is successful and the station can transmit a new packet).

Note that the CAMB tree protocol requires that a station see the broadcast of its entire packet before assuming a successful transmission. This is because of the priority mechanism described above, which gives the possibility of a packet broadcast being cut short (aborted in the middle).

Concurrency of Transmissions

In the CAMB tree, concurrent transmissions are possible. This is due to the segmented nature of the network, which allows broadcasting to non-overlapping subtrees of the tree network. In the CAMB tree, the switch protocol is designed in such a way that, regardless of the level of a switch (either above or below the proper ancestor), a packet is always passed to the parent switch. Every

time the climbing packet busies a switch above its proper ancestor, it prevents stations beneath that switch from using the switch as a broadcast point. In other words, every time a packet busies the switch, it creates partitions that are the children subtrees of that switch. Within each of these partitions, broadcasts may occur. Any attempt to transmit from within a partition to a destination outside the partition does not succeed. This is illustrated in Fig.1. In [6], it is shown that the delay decreases significantly as the degree of concurrency increases.

3. Implementation of a CAMB Tree Network

Currently we have a completely operational prototype of the CAMB tree network. In this section, a design and an implementation of a CAMB tree network is described.

In our implementation, we followed the protocol layering architectures of the IEEE 802 local area networks. Fig.3 shows the layering architecture of the CAMB tree network and the components that implement each layer. The Physical layer consists of CAMB switches, transmission cables and transceivers located on the station/network [2] interface board. The MAC layer and part of the LLC layer are implemented by the interface board installed in the station. We have also provided the TCP/IP protocol on the top of the above protocol stack.

In the next subsections, we describe the design and implementation of the CAMB tree network (i.e., CAMB switch, station/network interface board, and TCP/IP).

3.1. CAMB Switch

The CAMB switch can be realized by the following three switch components. See Fig.4.

- Uplink Selector (US), which is responsible for detecting an incoming packet and selecting one in case of simultaneous packet arrivals.

[2] The station used in this implementation is a LUNA workstation. The LUNA is a UNIX based workstation that uses the 32 bit CPU M68030 (20 MHz) and has a 68881 (20 MHz) floating point co-processor. The processing speed is 4 MIPS. The workstation supports communication protocols such as TCP/IP, NSF, and X.25.

- Address Recognizer (AR), which checks the header of the selected packet to see if the switch is the proper ancestor. If it is, the AR sends the packet to the DS and to the parent switch, otherwise the AR sends the packet only to the parent switch.

- Downlink Selector (DS), which receives a packet from the AR and the downlink from the parent switch, and broadcasts the packet to all the children switches. In case of simultaneous arrivals of a packet from the AR and a packet from the parent downlink, priority is given to the packet from the parent downlink. In other words, if the DS is broadcasting a packet received from the AR upon the arrival of a packet from the parent downlink, the transmission of the packet from the AR is aborted, and the packet from the parent is broadcast.

We present here a design of the CAMB switch that has the following packet format and addressing scheme.

Packet Format

The packet format used in the current implementation of the CAMB tree network has the following seven fields: Destination Address, Control Bit, Reserved, Source Address, Packet Size, Data and CRC (Fig.5).

The Destination Address field (8 bits long) indicates the destination address. The next field, Control Bit, is used to indicate if the packet has reached its proper ancestor. This bit is initially set to 1 by the sending station. It remains 1 until the packet reaches the proper ancestor. When this happens, the proper ancestor switch resets this bit to 0. The Reserved field is not being used. The Source Address field indicates the source address. The Packet Size field indicates the number of bits contained in the Data field. This information is used by the stations to determine whether a packet has been aborted. The CRC field is used to check the correctness of the received packet (header and data). The switch examines only the Destination Address and the Control Bit field. All the other fields are used by the receiving station.

Addressing Scheme

The addressing scheme assumed in our switch design is shown in Fig.6. Stations are ordered

in ascending order from the left to the right of the tree. Addresses are assigned to the switches in the following way. Each switch has an eight bit address. However, at level i, only the $2 \times (i-1)$ left most bits are significant. The other bits are ignored. In our addressing scheme, an address is assigned to a switch in such a way that the first $2 \times (i-1)$ bits of the address of a switch on level i match with the first $2 \times (i-1)$ bits of the addresses of all of the stations in its subtree.

Thsi numbering of stations and switches makes it easy for a switch to know whether it is the proper ancestor of a packet. A switch on level i checks the $2 \times (i-1)$ leftmost bits of the address field of a packet. If it is equal to the $2 \times (i-1)$ leftmost bits of its own address, then it is the proper ancestor of the packet. In this case, (as indicated before), the packet is sent to both the parent switch and the children switches. Otherwise, the packet is only sent to the parent switch.

In the following, we use an example to show how packets are transmitted on a CAMB tree using the above addressing scheme and packet format. Fig.6 shows a transmission of a packet from station A (address 00000000) to station B (address 00000100) as an example. When station A sends a packet, its immediate parent switch, Z (address 000000xx), checks the destination address of the packet. As the six leftmost bits of the destination address in the packet header are different from the switch address, it sends the packet only to its parent switch (Y). Switch Y (address 0000xxxx) then checks the four leftmost bits of the destination address of the packet. This time, as they match with the switch's address, switch Y knows that it is the proper ancestor of the packet. Thus, the packet is broadcast to its children. The packet is also sent to the parent switch (X). Now, the switch X (address = 00xxxxxx) checks the two leftmost bits of the destination address field. They agree with the switch address. Therefore, the switch X thinks that it is the proper ancestor of the packet and becomes ready to broadcast the packet to its children. However, if the switch X actually broadcasts the packet, the packet is broadcast twice (once by switch Y and once by switch X). To avoid this undesirable situation, a control bit has been added in the packet header (Fig.5). First, a switch checks the control bit in the packet header. If it is 1, the switch knows that the packet has already been broadcast by its proper ancestor, so it will transmit the packet only to the parent switch. Otherwise, it will check the address field to see if it is the proper ancestor or

the packet. In the example shown above, the switch X first checks the control bit. As it is 0, the switch does not broadcast the packet.

We present a switch design which supports four children switches (Fig.4). Thus each leaf switch can interface with four devices (stations) – the entire tree structure is a 4-ary tree. Note that this design can be easily extended to support more than four children per switch. Each line (either up- or down-link) consists of a control path and a data path. The control path signals the beginning and end of each packet. This line is high whenever there is a valid packet transmission on the data line. Otherwise it is low, signaling the switch that there is no packet being transmitted. The data path carries the packet sent to the switch.

In the following, we present a possible implementation of the CAMB switch. First using TTL devices and then a VLSI implementation. In the design, we assume the packet format and the addressing scheme discussed above.

3.1.1. TTL Switch Design

The design of each of the three components (US, AR, DS) of the switch is explained below.

Uplink Selector (US)

Fig.7 schematizes the possible design of the US section of the CAMB switch. The US has been broken down into four units.

- The *Synchronizer* – which aligns an incoming packet with the switch's internal clock. Note that if several switches in the network share the same clock signal, this unit is not required between interconnected switch nodes. Removal of the unit in this case will speed up packet transmission.

- The *Start Recognizer* – which detects the start of a new packet by checking the control line. This unit is also responsible for blocking all other uplinks while a selected packet is being transmitted.

- The *Priority Resolver* – which selects one packet at random, when more than one packet

arrives simultaneously at the switch. Note that the Start Recognizer and the Priority Resolver are exclusively control path units, i.e., they do not have connections to the data path. So the Priority Resolver selects the packet based on the signal from the control lines and signals the Multiplexer which packet was selected. The *New'* line from the AR signals the Priority Resolver the end of a packet transmission, resetting the Priority Resolver.

- The *Multiplexer* – which allows only the packet which the Priority Resolver selected to go through to the Address Recognizer.

In summary, the synchronizer receives packets from uplinks. The start recognizer detects the beginning of the packets (control lines are high), activating the priority resolver, which will randomly select one of the packets, based on the control lines. The selected packet is allowed to go through the multiplexer.

Address Recognizer (AR)

Fig.8 shows the schematic of the AR section of the switch. It consists of 5 units.

- The *Shift Register* – which allows the Comparator to check the destination address of a packet. An incoming packet goes through the shift register and is always sent to the parent switch. The destination address of the packet is contained in the Shift Register for one clock cycle. During that cycle, the Comparator can check the the destination address of that packet.

- The *Comparator* – which generates a signal (*Cmp*) to indicate whether the destination address of the incoming packet matches with the switch address, indicating the switch is a proper ancestor of the packet. The switch address is configured in a block of 8 SPST switches.

- The *Routing Logic* – which generates a signal (*Broadcast*) to indicate if the incoming packet will be sent to the Downlink Selector to be broadcast or not. This signal is generated based on three signals received by the routing logic: *Cmp*, *Control Bit*, *Parent Int'*. When *Cmp* is low, and Control Bit of a packet is high, and *Parent Int'* is high, a high *Broadcast* signal will be generated. This means that the address of the incoming packet and the address of the

switch match, and the packet has not been broadcast by its proper ancestor, and there is no transmission from the parent switch. Otherwise, the *Broadcast* signal will be low. *Ddn* and *Cdn* lines send to the DS the data and control lines, respectively, received from the US.

- The *Reset Control Bit Logic* – which resets the control bit (see packet format – Fig.5) in the incoming packet whenever the switch recognizes that it is the proper ancestor of the packet. As explained in previously, this will prevent broadcast of a packet by a switch at higher level than the proper ancestor.

- The *End Recognizer* – which detects either the end of a packet going through the shift register (by checking if the control line from the US is low) or a transmission abort (by checking if the *Parent Int'* line is low). When one of these events happens, the End Recognizer sends a reset signal (*New'*) to the Uplink Selector. Upon the reset the US waits for new incoming packets. Whenever there is a transmission from the parent, the *Parent Int'* line is low.

Downlink Selector (DS)

The DS section is shown in Fig.9. It is divided into two units.

- The *Selector* – which selects a packet from the parent downlink or from the Address Recognizer (*Cdn*, *Ddn*) and broadcasts it to all the children switches. As described previously, priority is given to the parent's downlink over the children's uplink. If a packet from the parent downlink arrives during transmission of a packet from the AR, the Selector will abort the packet transmission from the AR. In addition, any packet from the AR will be blocked during the entire period of transmission of a packet from the parent switch.

- The *Flip-flop* – which is used to set the control lines to children switches low in case of an abort. If the control line goes low it indicates the end of a transmission. It can be either the end of a transmission of an entire packet or a truncated packet (abort). To detect an abort, station needs to check the packet size indicated in the packet header and the actual size of the packet received.

A full schematic of the TTL tree switch based on the above design is given in Fig.10.

3.1.2. VLSI Switch Design

In order to reduce the size of a switch and to decrease the delay incurred in a switch, a VLSI implementation of the CAMB switch is presented.

Layout

To obtain a layout for the CAMB switch, a standard cell layout synthesis tool, Autocells [16] (GDT[16] Version 4.0.1g) is used. Autocells is an automatic placement and routing tool for laying out circuits using standard cells (polycells). Standard cells are small, predefined rectangular layout blocks that perform simple logic functions that correspond to low-level icons in a schematic. Autocells is provided as a part of the integrated GDT system.

The layout produced by this system is shown in Fig.11. This layout is in 3 microns P-well standard CMOS technology. Routing is done in only one layer of metal. And to obtain the smallest layout area that can possibly be generated, all transistors are set to nominal size.

The resulting layout is 1497.5 microns (width) by 1549.5 microns (height). Input and output ports of the switch are arranged as shown in Fig.12.

Performance Test of the Layout

Circuit and layout testing is accomplished with Lsim [16] simulator (GDT version 4.0.1g). Three basic algorithms are supported by the Lsim simulator. These include system or logic simulation with single direction signal flow, bi-directional switch level simulation, and ADEPT circuit and analog simulation. In this CAMB switch implementation, the bi-directional switch level simulation is used as the functional testing method. Performance of the layout is obtained by using the Lsim's ADEPT mode simulation.

In our performance tests, a switch is driven with the clock speed of 10MHz, which is the normal Ethernet channel speed. The result obtained shows correct functionality with a maximum input to output delay of 32.78 ns.

3.2. Station/Network Interface Board

The purpose of this section is to provide a description of the CAMB Station/Network interface board. This interface board is designed to connect a station to the CAMB tree network. As described in the beginning of section 3, the interface board provides the functions corresponding to part of the physical layer, MAC layer and part of LLC layer (Fig.3).

First we will describe the transmission and reception mechanisms in the interface board.

Transmission

The interface between the station and the interface board can be modeled as two independent mailboxes. One mailbox is controlled by the station (referred to as a station mailbox), and the other mailbox controlled by the interface board (referred to as an interface mailbox). When a station wishes to send a packet over the network, the station places a packet into the interface mailbox, and raises the electronic equivalent of a mailbox flag. This flag signals the interface board that the interface mailbox is full.

The interface board responds to the raised interface mailbox flag by performing the following steps.

- re-lowers the interface mailbox flag;
- removes the packet from the interface mailbox;
- drops a note which says "Interface mailbox emptied, ready for one more packet" to the station in the station mailbox;
- raises the station mailbox flag;
- processes the packet (adding header) and places the packet in an interface board internal transmission buffer;
- transmits the packet using the protocol described in section 2.

The station responds to its raised mailbox flag by executing the mail fetching steps similar to the ones performed by the interface board:

- re-lowers the station mailbox flag;
- removes and processes the note in its box;
- puts another packet into the interface mailbox if there is one available.

Reception

Packet reception is basically the packet transmission in reverse. The only difference is that the interface board provides one additional function. The interface board checks the destination and CRC of the packets and passes only those which are error free and correctly addressed.

When the interface board starts to receive a packet from the network, it immediately begins to store this packet in a buffer within the interface board, and also performs CRC error checking. When the end of packet is received, if the packet contains an error, it is discarded. Otherwise (no errors), the following two cases are possible.

- In the first case, the incoming packet is the same as the one which the interface board just transmitted. As described in a previous section, the source station identifies a successful transmission by receiving a broadcast of the transmitted packet without error. Thus, after the interface board receives a correct broadcast of a complete packet, it looks at the internal transmission buffer for the next packet to be transmitted. If there are more packets in the queue, the interface board starts the transmission of the next packet.
- In the second case, the interface board receives a broadcast from another station. If this packet is addressed to another station, then the packet is discarded. If the packet is addressed to the station, then it will be moved to station mailbox and the station mailbox flag is raised. When the station sees its raised mailbox flag, it removes the packet for processing and re-lowers the flag. The station then places a "station mailbox empty" note in the interface mailbox and raises the interface mailbox flag. This note signals the interface board that the station is ready to receive another packet.

3.2.1. The Interface Hardware

Interface Board Design

The interface board functions described above can be realized by the following four sections (Fig.13).

- the *Station Interface Section (SIS)*, which is responsible for the transfer of data and control information between the station and the interface board. It implements the mailbox system described in the previous section.
- the *Microprocessor Control Section (MCS)*, which manages data movement, data buffering, and header processing for transmission and reception.
- the *Transmission Section (TS)*, which accepts a packet from the MCS, generates CRC for it, and transmits the packet to the switches.
- The *Reception Section (RS)*, which receives incoming packets, performs CRC error checking, and buffers them for transfer to the MCS.

Next we describe the detailed design and implementation of the four sections just described.

The Station Interface Section (SIS)

Fig.14 schematizes the possible design of the SIS section of the interface board. The SIS links directly with the station. SIS has been broken into the following units.

- The *Bus Transceiver* – which is required to provide an asynchronous two-way communication between the address and data busses within SIS and the address and data busses within the station. It is also used to reduce the noise across the interface connector (See Fig 13).
- The *Interface Buffer* – which stores packets to be transmitted. This Interface Buffer and the Interface Status Register described below collectively function as the interface board mailbox described in section 3.2.

- The *Interface Status Register (ISR)* – which stores a control word from the station. Whenever a control word is written in this register, the interrupt line (*INT0'*) will go low, interrupting the MCS. (This corresponds to raising the mailbox flag.) Then, the MCS reads the control word from Interface Status Register.

- The *Station Buffer* – which stores a packet to be passed to the station. This Station Buffer and the Station Status Register described below collectively function as the station mailbox described in section 3.2.

- The *Station Status Register (SSR)* – which stores a control word from the interface board. Whenever a control word is written in this register, the interrupt line (*INT5*) will go high, interrupting the station. (This corresponds to raising the mailbox flag.) The station then reads the control word from Station Status Register.

Note that the interrupt line (*INT0'* or *INT5*) is deactivated when a read is completed in the correspondent Status Register (Interface or Station, respectively). This corresponds to the operation of lowering the mailbox flag. The device used to implement the buffers and status registers is a Dual Port RAM (DPRAM).

The Microprocessor Control Section (MCS)

The block diagram of the MCS is given in Fig.15. The MCS controls the other three sections of the interface board. The MCS consists of the following units:

- The *Central Processor Unit (CPU)* – which implements the control logic that drives all the other units in the interface board. It executes a small firmware code contained in the Read Only Memory (ROM). It is responsible for packet processing. such as creating and appending a header to outgoing packets, processing the header of incoming packets, moving packets among DPRAM (in SIS), RAM and FIFO (in TS and RS); and controlling transmission and reception of packets. The CPU is a 16-bit microprocessor, operating at 10 MHz. Internally it has the following modules.

- Two DMA (direct memory access) modules. used to transfer data from Random Access Memory (RAM) to FIFO (in TS) and vice-versa (in RS). One DMA is used for transmission and the other is used for reception. The DMA can be programmed to transfer a certain number of bytes. When this number is reached, an internal interrupt is sent to the CPU.
- A clock generator module, used to generate a clock signal to the CPU and to the other units.
- A programmable timer module, used as a timer to start packet retransmissions. When the timer counter reaches zero, an internal interrupt is sent to the CPU, and a retransmission takes place.

PCS0' line is used by the CPU to reset the TS, *PCS1'* line is used to signal the TS to start the transmission of a packet, and *PCS2'* is used to signal TS to retransmit a packet. The *DRQ1* line received from the RS indicates that a packet is available in the FIFO, and *INT1'* indicates the end of a packet.

- The *Read Only Memory (ROM)* – which contains the code that is executed by the CPU.
- The *Random Access Memory (RAM)* – which provides the internal transmission buffer and the internal reception buffer. It is also used to store information necessary for the operation of the CPU such as variables and interrupt vectors.

The Transmission Section (TS)

The functions of the Transmission Section are provided by the following three units (Fig.16):

- The *Transmission FIFO* – which is a First-In First-Out memory used to store the packet that is currently being transmitted. The MCS loads the packet into the FIFO and starts the transmission by activating *PCS1'*. It also receives a signal from the retransmission control line (*PCS2'*). If the broadcast of the transmitted packet is not received before the timeout, the MCS sets this line high, triggering a retransmission of the packet in the FIFO.

- The *CRC Generator* – which is used to generate on the fly the CRC code for the current transmitted packet.

- The *Transmission Transceiver* – which is used to interface the board with the transmission cable (of a tree network).

The Reception Section (RS)

Fig.17 shows the block diagram of a design of the Reception Section. It consists of the following three units:

- The *Reception FIFO* – which is a First-In First-Out memory used to store incoming packets from the network (i.e., CAMB switch). When the control line from the network is low, all three units of the RS are held in wait state, and *INT1'* and *DRQ1* are inactive (high and low, respectively). When the control line (from a CAMB switch) goes high, indicating the beginning of a packet, the CRC checker (described below) starts calculating the CRC, and the incoming packet is stored in the FIFO. As soon as the first byte of the packet is stored in the FIFO, the *DRQ1* signal is activated (goes high), informing the DMA module (in the MCS) that there is data to be transferred to the MCS internal reception buffer. When the DMA reads the last byte of an incoming packet, an interrupt signal (*INT1'*) is generated and sent to the MCS, informing MCS of the end of transfer.

- The *Interrupt Logic* – which generates the *DRQ1* signal to activate the DMA module in the MCS when a packet arrives. It also activates the *INT1'* interrupt signal when the end of a packet is transferred to the reception buffer by the DMA.

- The *CRC Checker* – which is used to check on the fly the CRC code of the packet being received.

- The *Reception Transceiver* – which is used to interface the cable with the board.

A complete circuit schematic of the Station/Network interface board is shown in Fig.18.

3.2.2. The Interface Software

In this section, the firmware (contained in the ROM, located in the MCS) that the CPU executes is described. It consists of two concurrent processes: a packet transmission process, and a packet reception process. These processes are both interrupt driven. In other words, an appropriate process is triggered either when a packet is left in the DPRAM and the interrupt line (*INT0'*) goes active, or when the RS signals that a packet is being received.

As described in section 3.2, the DPRAM functions as the mailbox system. The DPRAM is divided into two separate mailboxes: one for the station, and the other for the interface board (Fig.14). Each mailbox is designed to hold one packet and one word of status information at a time. In this implementation the size of each mailbox is 1K $\times$ 16 bits. Note that this means that the maximum packet length supported by the interface board is 2046 bytes.

In this implementation, a 32K $\times$ 16 bits RAM is used (Fig.15). The RAM in the interface board is partitioned into three regions. The first region (4 Kbytes in our implementation) contains variables and tables used by the firmware that controls the interface board. The second and third regions (30 Kbytes each in our implementation) are used as a reception and a transmission buffer respectively.

The Transmission Process

The transmission process starts when the station writes a packet to the Interface Buffer in the SIS. The station also writes a control word to the Interface Status Register. This forces the interface mailbox flag to be raised. This means that the DPRAM's interface interrupt line, *INT0'* (Fig.14), goes low. This interrupts the microprocessor within the MCS board to start the following transmission algorithm.

1. The microprocessor reads the control word at the Status Register, clearing the interrupt signal *INT0'*.

2. The control word indicates that there is a packet in the Interface Buffer. This causes the microprocessor to transfer the packet either to the TS if the TS is not busy, or to the MCS

internal transmission buffer otherwise.

3. The microprocessor writes a control word in the Reception Status Register, indicating that the Interface Buffer is available to receive another packet.

4. This write forces the DPRAM's station interrupt, *INT5*, to be activated.

The station responds to the active signal on *INT5* by activating a process that reads the control word written by the MCS. Since the control word informs the station that the interface board is available for another packet, the station will copy a new packet to the mailbox if it is ready, starting the transmission process again.

At this point, a packet is now moved from the station and is stored in the Interface Buffer in the SIS. In the following we describe how the packet is transmitted to the network from the SIS mailbox.

As described before, if the TS is busy transmitting a packet, the incoming packet will be stored in the transmission buffer in the RAM (within the MCS on the interface board). Otherwise, the microprocessor resets (via *PSC0'*) the TS and programs its DMA (in the MCS) to transfer the packet from the Interface Buffer (DPRAM) to the TS's FIFO. When the transfer is complete, the microprocessor starts the timer and signals the FIFO to start the transmission. If a broadcast of the packet is not received by the RS when the time-out occurs, the microprocessor resets the timer and retransmits the packet. This process continues until a broadcast of the packet is received by the RS.

When a broadcast of the transmitted packet is received, the reception process will set a flag to stop the transmission process. The transmission process resets the TS and looks in the transmission buffer for a new packet to be transmitted.

The Reception Process

Before the arrival of a packet, the control line from a CAMB switch is low, and *INT1'* and *DRQ1* are inactive (Fig.17), indicating the RS is in an idle state. The DMA unit (in the MCS) is initialized to fetch three bytes from the reception FIFO (in the RS).

The reception process starts when a packet (from a CAMB switch) arrives at RS on the interface board. Upon the arrival of a packet at RS, the control line changes from low to high, moving the first byte of the incoming packet into the reception FIFO. This causes *DRQ1* to change to high, and the high on the *DRQ1* line in turn triggers the DMA module to start the byte-by-byte transfer of the packet from the FIFO to the reception buffer in MCS.

When the last byte of the packet is transfered from the FIFO to the MCS reception buffer, an interrupt (*INT1'*) is generated to signal the reception process to check the last bit of the received packet. If the bit is equal to one and the number of bytes received matches that of the Packet Size field, the packet has been received correctly. Otherwise, a CRC error has been detected.

In the case where a CRC error has been detected and the station is the source of the packet, the erroneous packet is discarded, and the flag is set to inform the transmission process to retransmit the packet.

If the packet has no error (CRC and packet size correct), then the following three cases are possible.

a) The destination address matches that of the receiving station. In this case, the packet will be kept in the reception buffer for transfer to the station.

b) The source address matches that of the receiving station. In this case, the reception process notifies the transmission process of the successful transmission of the packet.

c) Neither the source nor the destination address of the packet matches that of the station. In this case, the incoming packet is discarded from the reception buffer.

Note that after the first three bytes are moved to the reception buffer, the reception process examines the first three bytes of the packet (the destination address, a reserved byte and the source address) and determines which of the above is the case.

In the first case, the incoming packet is transferred from the reception buffer in the MCS to the Station Buffer in the SIS. When the station finishes reading the packet from the Station Buffer,

it writes a control word in the Interface Status Register, generating an interrupt (*INT0'*) for the interface board. This interrupt signals the reception process to search for more packets in the reception buffer (in MCS) to be transferred to the station.

3.3. Network Protocol

In the previous sections, we described the design and implementation of the Physical layer and Data link layer (MAC and LLC layers) of the CAMB tree network. In this section, we describe transport protocol support for the CAMB tree network.

In our CAMB tree network implementation, we used Unix-based LUNA [3] workstations as network stations. LUNA workstations support TCP/IP and UDP/IP. To support such transport protocols (TCP/IP and UDP/IP) on the top of our CAMB tree network layering architecture, we implemented an interface between IP and the interface board.

The most efficient way to support IP on our CAMB tree network is to design and implement a device driver for this purpose. However, this will make the implementation hardware dependent and not easily portable to other machines. Therefore, our approach is to provide an interface that is hardware independent by using the LUNA workstation's general device driver as a base (Fig.19). A similar design approach was used at the University of California at Irvine to connect tty lines into the ARPA Internet [19].

In our implementation, TP passes a packet to IP, and IP then passes the packet to the pseudo-device called the Raw Packet Interface (RPI). The packet is then handed to the Tree Network Daemon process through a raw socket interface. Finally, the packet is handed to the Station Device Driver that sends it to the Tree Network interface board. (The reception procedure is the reverse of the transmission procedure described above.)

3.3.1. Station Device Driver

The LUNA station general device driver allows a process to map its virtual memory address

[3] LUNA is a trademark of Omron.

space into the physical I/O address space. It also allows a process to wait for an interrupt signal from an I/O interface. When an interrupt happens, a device driver internal interrupt routine is called to handle it and the process that is waiting for that interrupt is awakened. The *ioctl()* system call provides access to the device driver.

3.3.2. Tree Network Daemon

The Tree Network Daemon interfaces with the Raw Packet Interface and the LUNA Device Driver. In the case where a packet is transmitted towards the CAMB network, the tree network daemon reads a packet from the Raw Packet Interface and subsequently writes them to the LUNA Device Driver interface. In receiving a packet, the tree network daemon reads packets from the LUNA Device Driver interface and writes them to the Raw Packet Interface. In order to handle both transmission and reception of packets, the daemon forks into two processes.

The daemon is in effect a software link between the LUNA Device Driver and the Raw Packet Interface. The UNIX raw sockets are used as an interface between the Tree Network Daemon and the Raw Packet Interface.

The daemon program is rather simple in concept. It consists of an initialization sequence and infinite loops where it waits for inputs from the Raw Packet Interface and/or the LUNA Device Driver.

First the raw socket and the hardware are initialized. A raw socket is established, the connection to the LUNA device driver interface is made, the DPRAM address space is mapped into the Daemon memory address space, and several constants and pointers are initialized.

After the initialization of the socket and device interface, the daemon forks into two processes. The child process handles packet reception. It reads a packet from the device driver and writes it to the raw socket. The parent process handles packet transmission. It reads a packet from the raw socket and writes it to the device.

After creating a child process, the parent process enters into a wait state by calling a *recvfrom()* system call. When a packet is sent to the socket by the RPI, the process is awakened, and reads the

packet from the raw socket. The 4 byte header is added to an IP data unit to construct a tree net data unit (a packet). This 4 byte header contains the packet size and its destination. The packet is then moved to the DPRAM, and the control word is written to the Interface Status Register to indicate the existence of a packet in the DPRAM. Then the process enters into a wait state again.

The child process enters into a wait state by calling an *ioctl()* system call. When a packet is received from the network, the LUNA device driver will generate a signal, waking up the child process. Then the process reads the control word from the Station Status Register. If the control word indicates that there is a packet waiting to be received, then the size of the packet is read from the header (stored in the DPRAM). The packet is then copied from the DPRAM into a local buffer (in the tree network daemon). Next, a control word is written to the Interface Status Register to indicate that the Station Buffer is available. Finally, the packet is sent to the raw socket after stripping the tree network header. Then, the process enters into a wait state again.

3.3.3. The Raw Packet Interface

The Raw Packet Interface is a pseudo device driver which is used to provide a connection between the IP protocol layer and the Tree Network Daemon. The Raw Packet Interface appears exactly like a standard network interface (such as Ethernet) to the IP protocol layer. However, unlike an Ethernet device driver which interfaces with a real hardware component, the RPI interfaces with a software process (Tree Network Daemon).

In a packet transmission, IP sends a packet to RPI by invoking an output routine for the RPI (*rp_output ()*). The RPI output routine uses the UNIX raw socket interface to pass this packet to the Tree Network Daemon.

In a packet reception, when a packet is sent to the raw socket by the Tree Network Daemon, another routine in the RPI is invoked. This routine simply places the packet on the IP input queue.

4. Concluding Remarks

In this paper, a design and implementation of the CAMB Tree Network is described. CAMB switches, Interface Board and station protocols provide the physical layer, data link layer and

higher layer protocols (TCP/IP) for the Tree Network. Currently we have a completely operational prototype of the Tree Network. The next step is to run performance tests on the Tree Network with different configurations.

References

[1] A. Albanese, "Star Network with Collision-Avoidance Circuits," The Bell System Technical Journal, Vol. 62, No.3, March 1983.

[2] T. Suda, Y. Yemini and M. Schwartz, "Tree Network with Collision Avoidance Switches," Proc. of the IEEE Infocom, 1984, pp.105-113.

[3] J. Kurose, M. Schwartz and Y. Yemini, "Multiple-Access Protocols and Time-Constrained Communication", ACM Computing Surveys, Vol.16, No.1, March 1984.

[4] T. Suda, S. Morris and T. Nguyen, "Tree LANs with Collision Avoidance: Protocol, Switch Architecture and Simulated Performance," Proc. of the ACM SIGCOMM Symposium, 1988.

[5] T. Suda and S. Morris, "Tree LANs with Collision Avoidance: Station and Switch Protocol," Computer Networks and ISDN Systems, 17, 1989.

[6] S. Morris, T. Suda and T. Nguyen, "Tree LANs with Collision Avoidance: Photonic Switch Design and Simulated Performance," Computer Networks and ISDN Systems, 17, 1989.

[7] F. Closs and R. P. Lee, "A Multi-Star Broadcast Network for Local-Area Communications," in Local Networks for Computer Communications, A. West and P. Janson eds., North-Holland Publishing Company, cp IFIP, 1981.

[8] Y. Yemini, "Tinkernet: or, Is There Life Between LANs and PBXs?," Proc. of the IEE ICC, 1983.

[9] E. S. Lee and P. I. P. Boulton, "The Principles and Performance of Hubnet: A 50 Mbit/s Glass Fiber Local Area Network," IEEE Journal on Selected Areas in Communications, Vol. SAC-1, No.5, November 1983, pp.711-720.

[10] E. S. Lee and P. I. P. Boulton, "Hubnet Performance Measurement," IEEE Journal on Selected Areas in Communications, Vol. SAC-1, No.5, November 1983, pp.711-720.

[11] T. Suda and K. Goto, "Performance Study of a Tree LAN with Collision Avoidance." Proc. of the IEEE Infocom, 1989.

[12] A. E. Kamal, "A Performance Model for a Star Network," Proc. of the IEEE Globecom, 1986.

[13] A. E. Kamal, "Star Local Area Networks: A Performance Study," IEEE Trans. on Computers, Vol. C-36, No. 4, April 1987.

[14] S. Marano and A. Volpentesta, "Performance Evaluation of Alberonet by Simulation and Theoretical Analyses," Proc. of the IEEE Infocom, 1987.

[15] V. Ielapi, S. Marano and A. Volpentesta, "A Simulation Study for a Tree Local Area Network with Concurrent Transmissions," in Local Communications Systems: LAN and PBX, Elsevier Science Publishers BV, cp IFIP, 1987.

[16] " GDT – LCELLS Generator Library", Silicon Compiler Systems Corporation, 1988, 1989.

[17] A. E. Kamal and V. C. Hamacher, "Analysis of a Star Local Area Network with Collision Avoidance," in the IEEE INFOCOM, 1988.

[18] H. S. Hassanein and A. E. Kamal, "Performance Evaluation of Prioritized Collision-Avoidance Star Local Area Networks," IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, June 1989.

[19] M. T. Rose, "Low Tech Connections into the ARPA Internet: The Raw Packet Split-Gateway," Technical Report No. 216, University of California, Irvine, Feb. 1984.

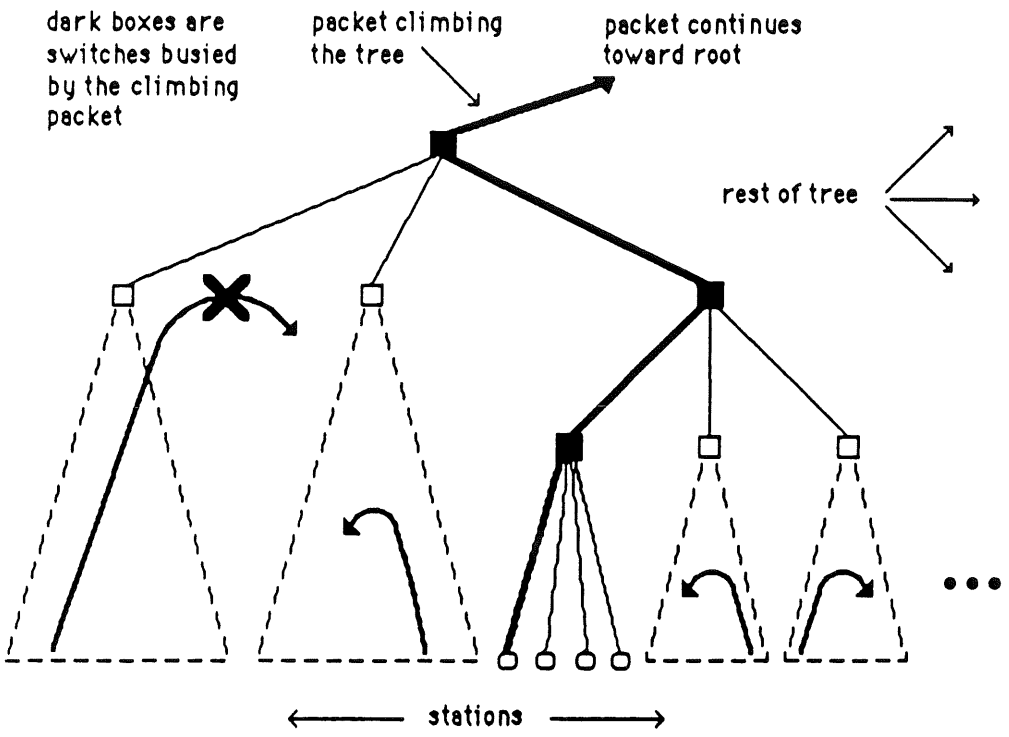

Fig.1 Partitioning of CAMB Tree by Climbing Packet

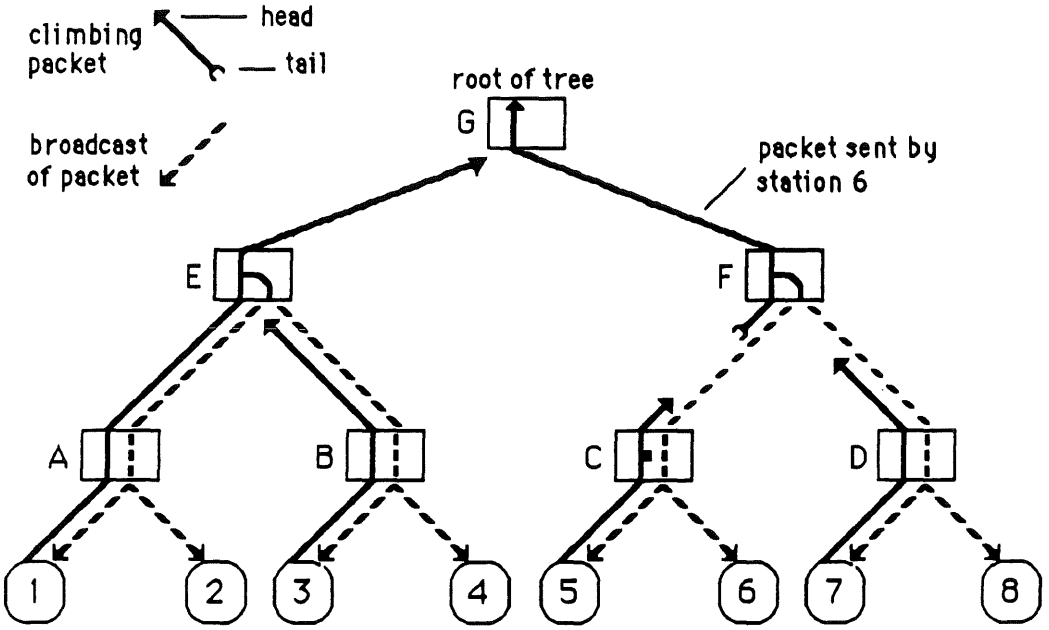

Fig.2 Trasmission in a CAMB Tree

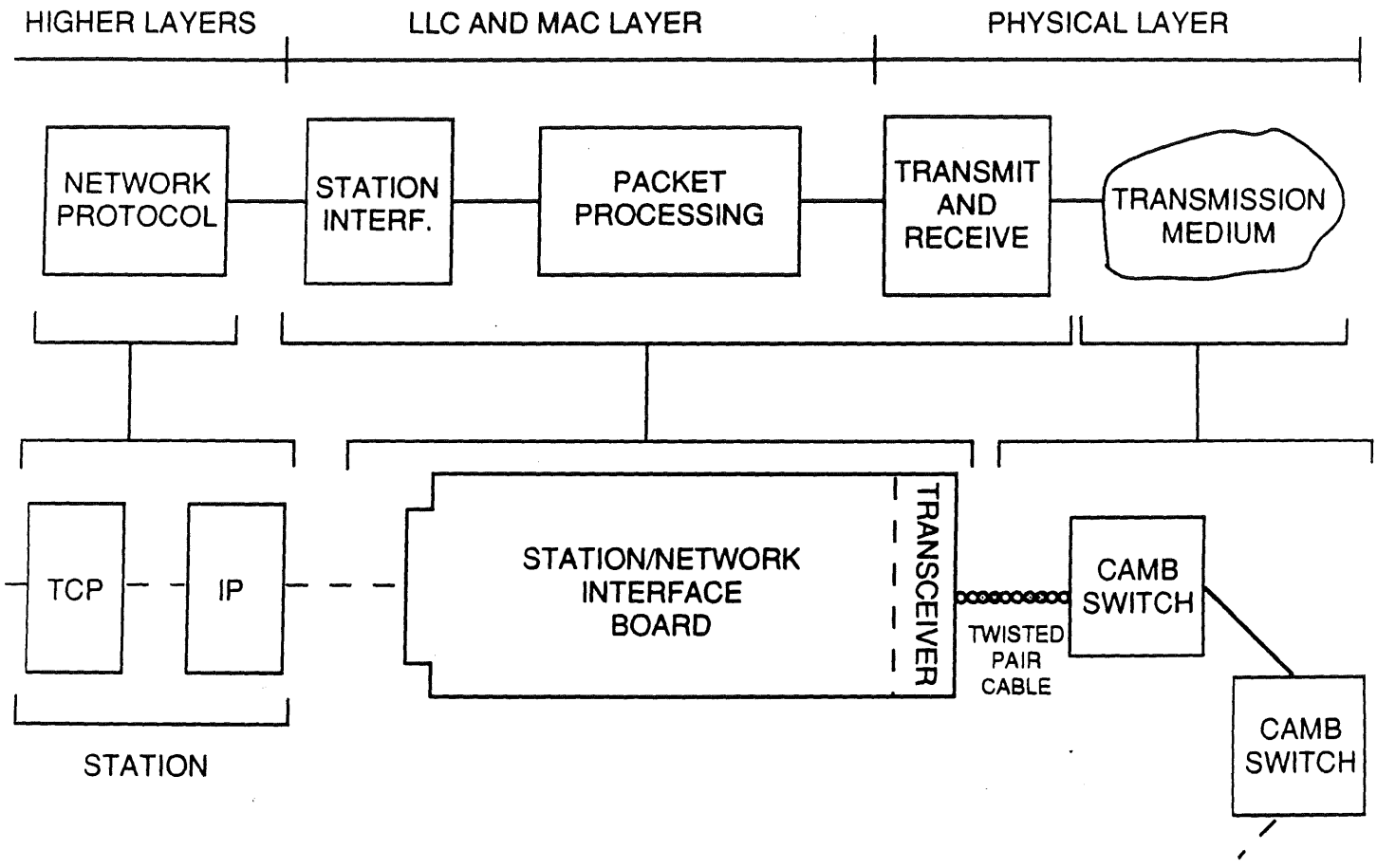

Fig.3 Layering Achitecture of the CAMB Tree

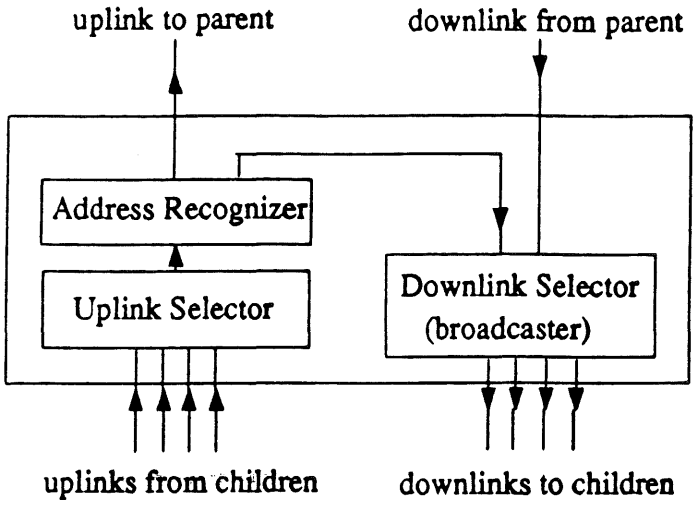

Fig.4 CAMB Switch

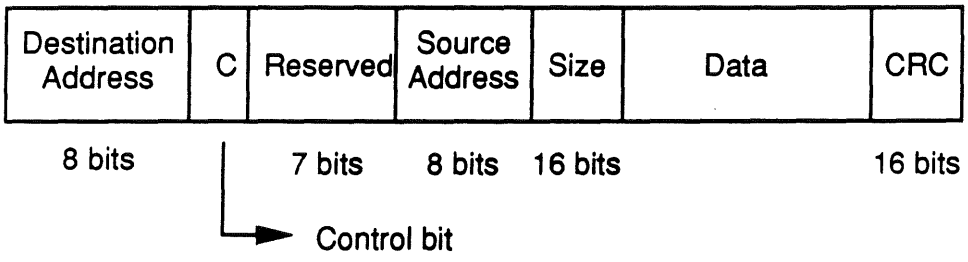

Fig.5 Packet Format

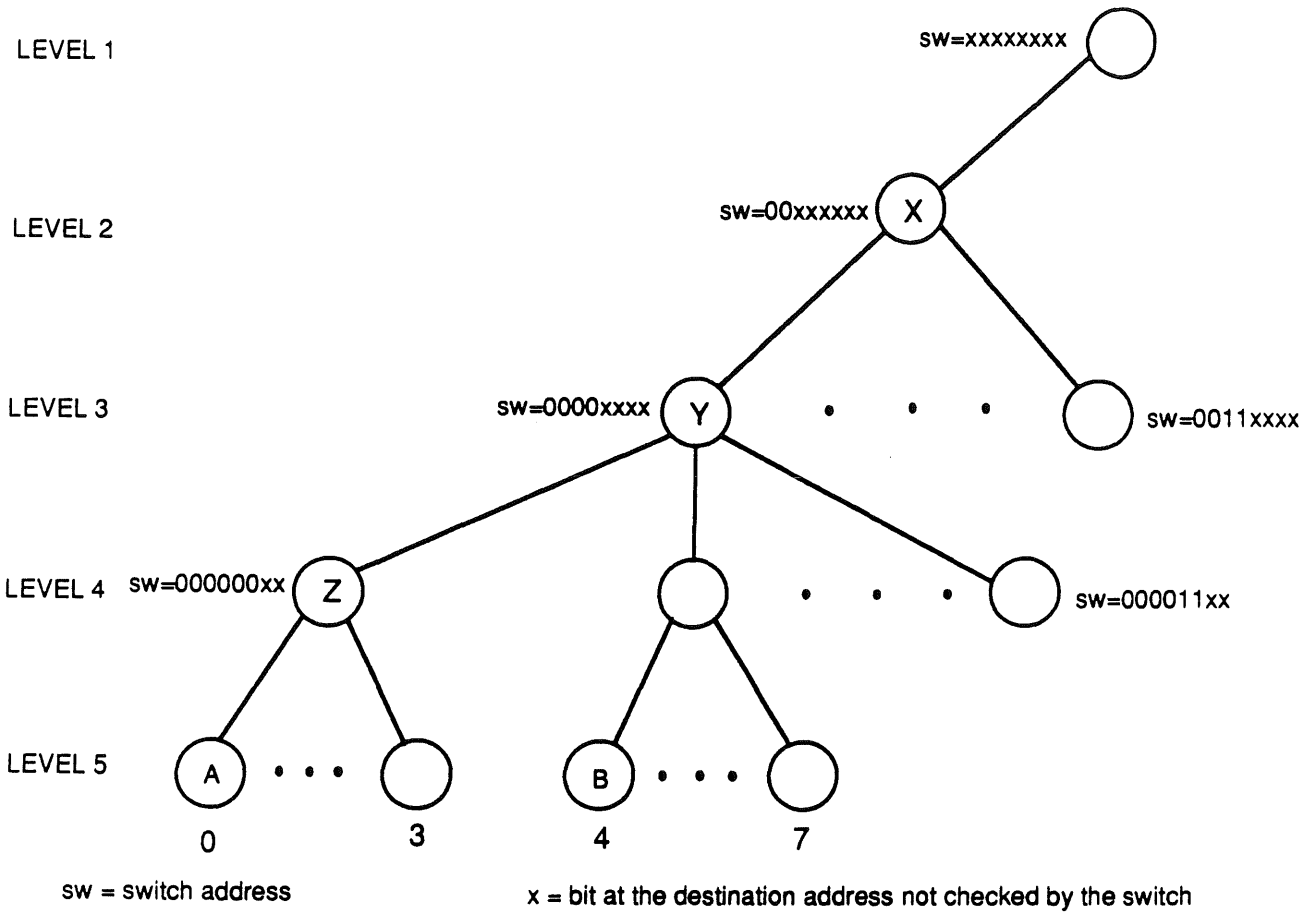

Fig.6 Addressing Scheme

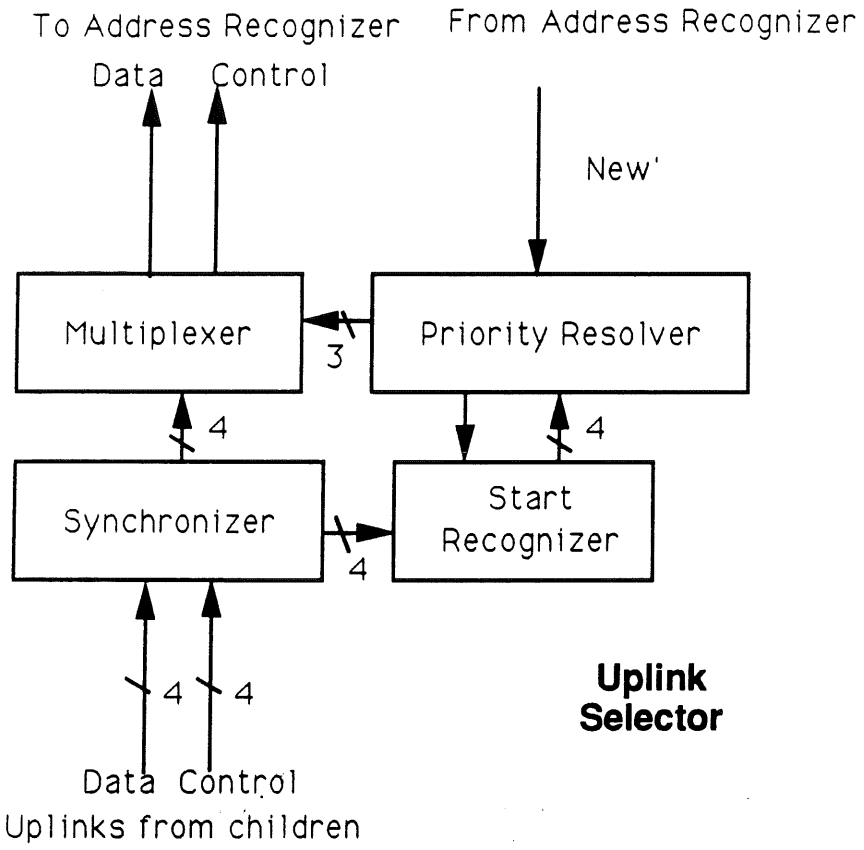

Fig.7 Uplink Selector Block Diagram

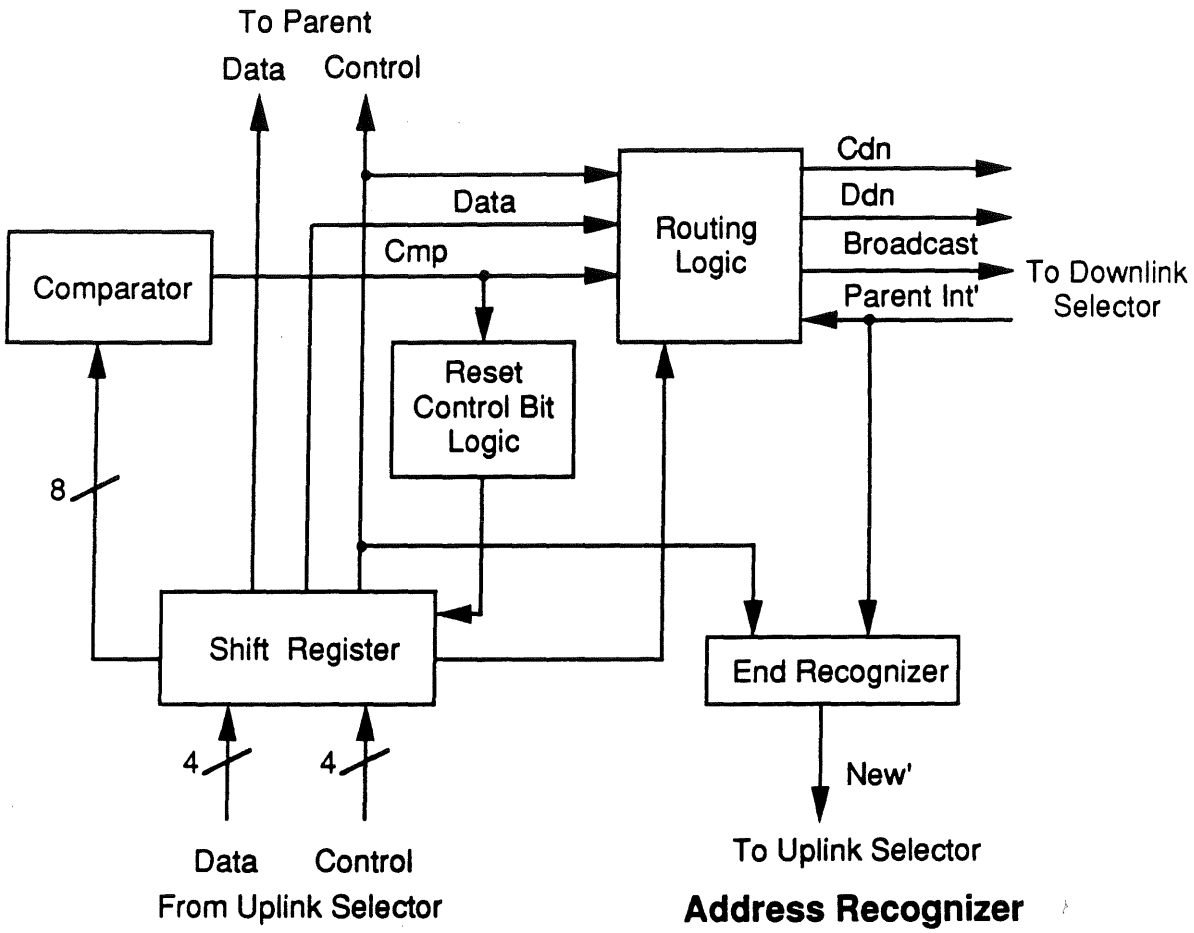

Fig.8 Address Recognizer Block Diagram

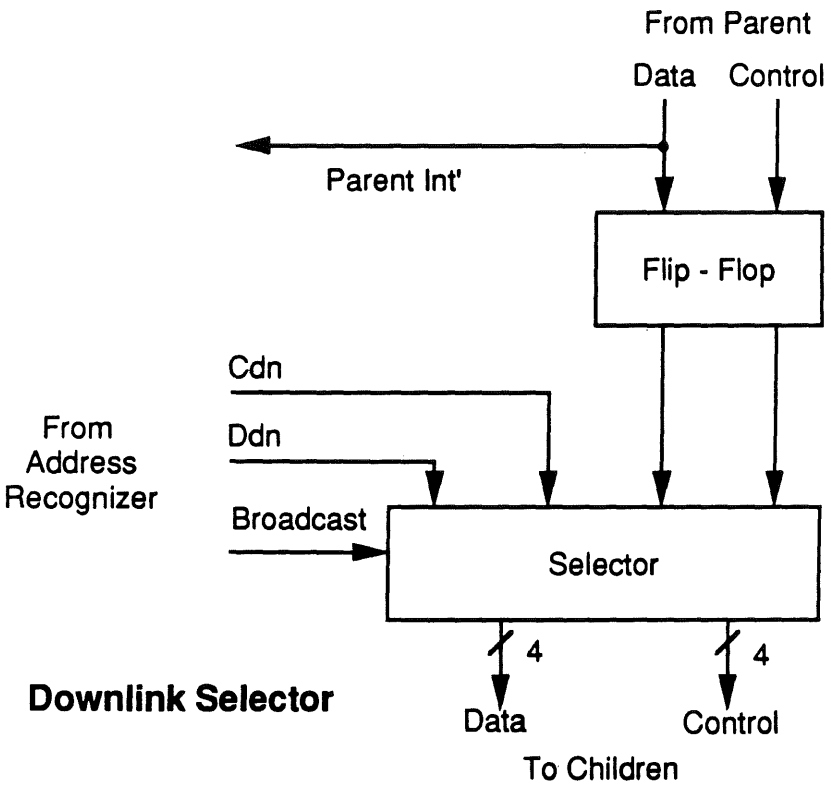

Fig.9 Downlink Selector Block Diagram

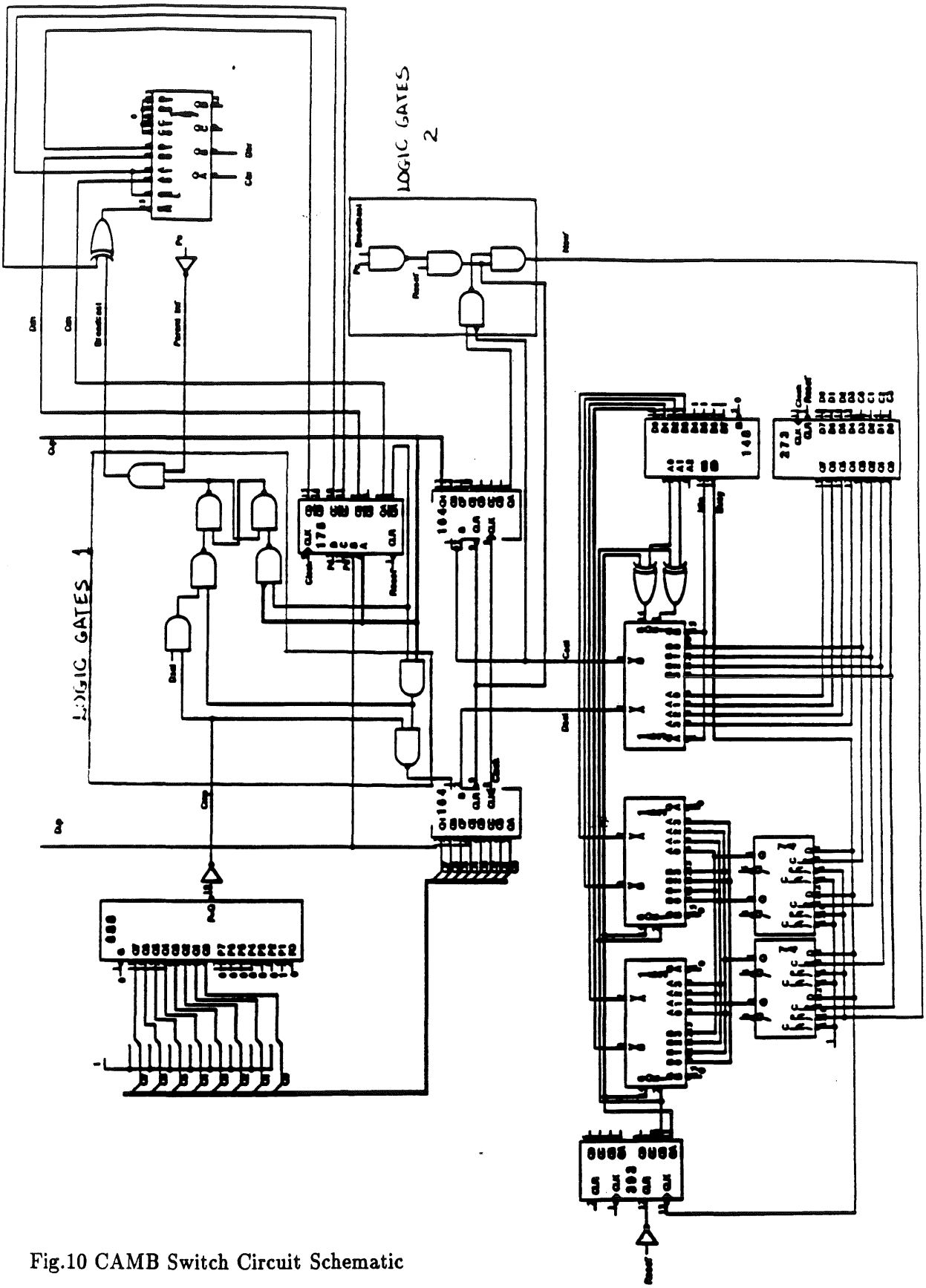

Fig.10 CAMB Switch Circuit Schematic

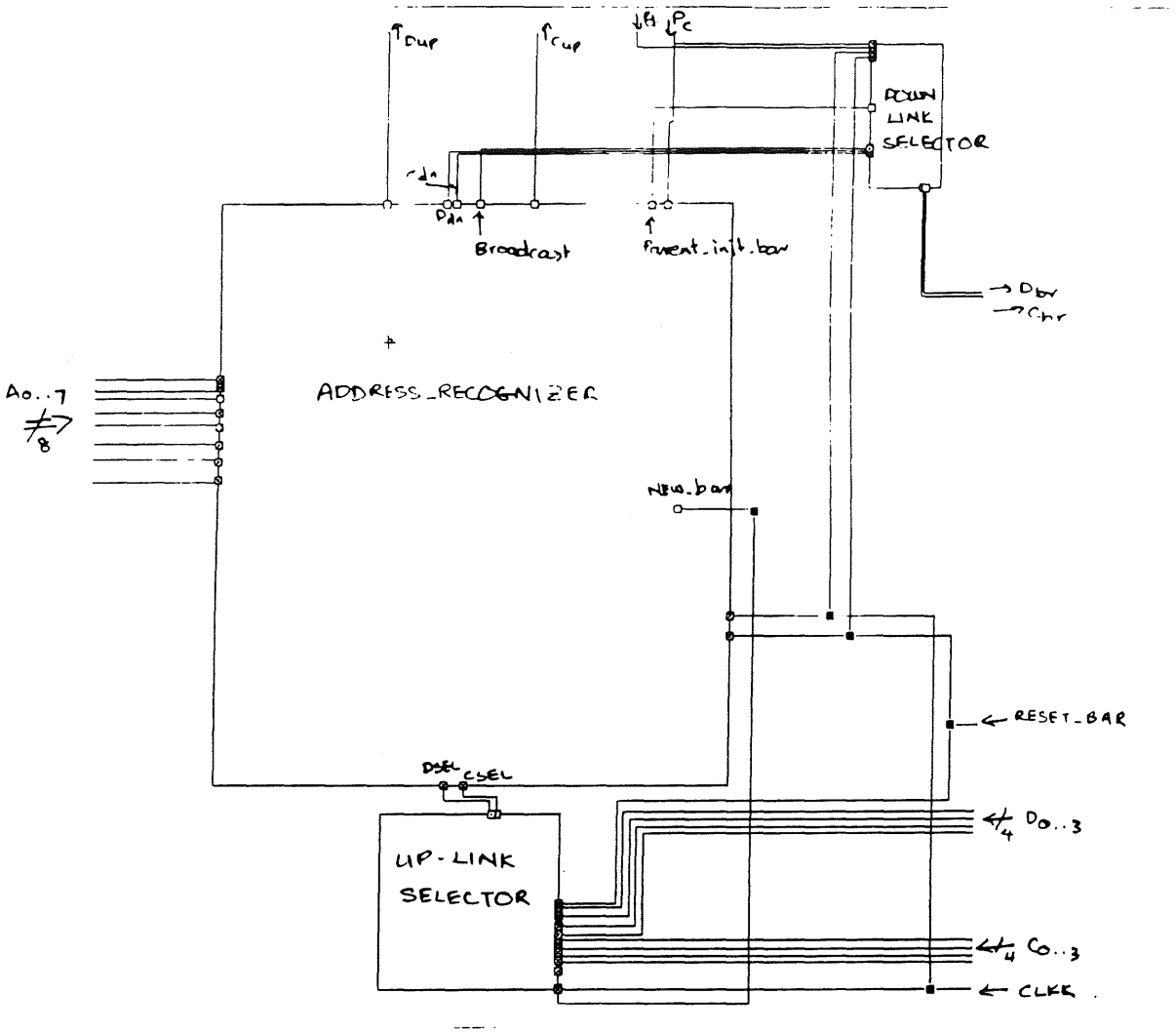

Fig.11a VLSI Implementation of the CAMB Switch

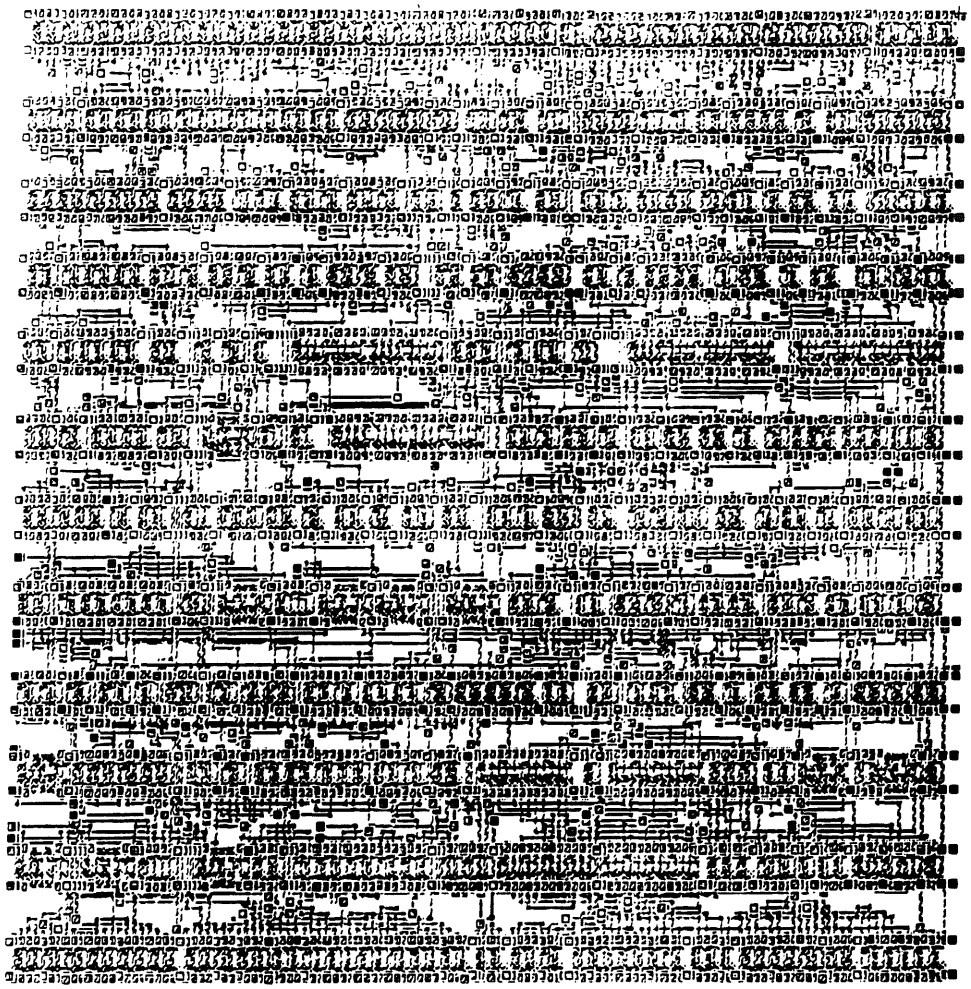

Fig.11b VLSI Implementation of the CAMB Switch

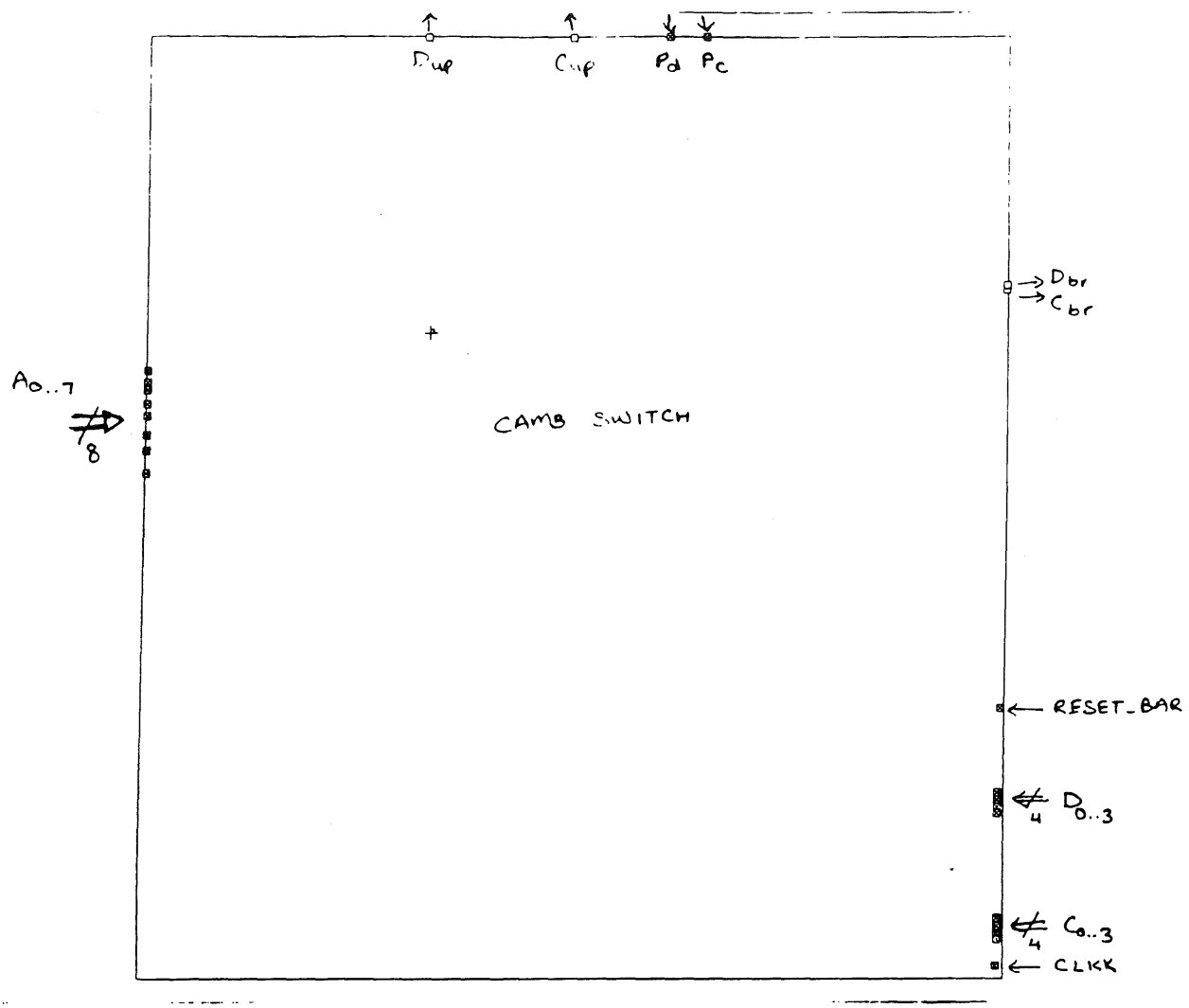

Fig.12 Input and Output Ports

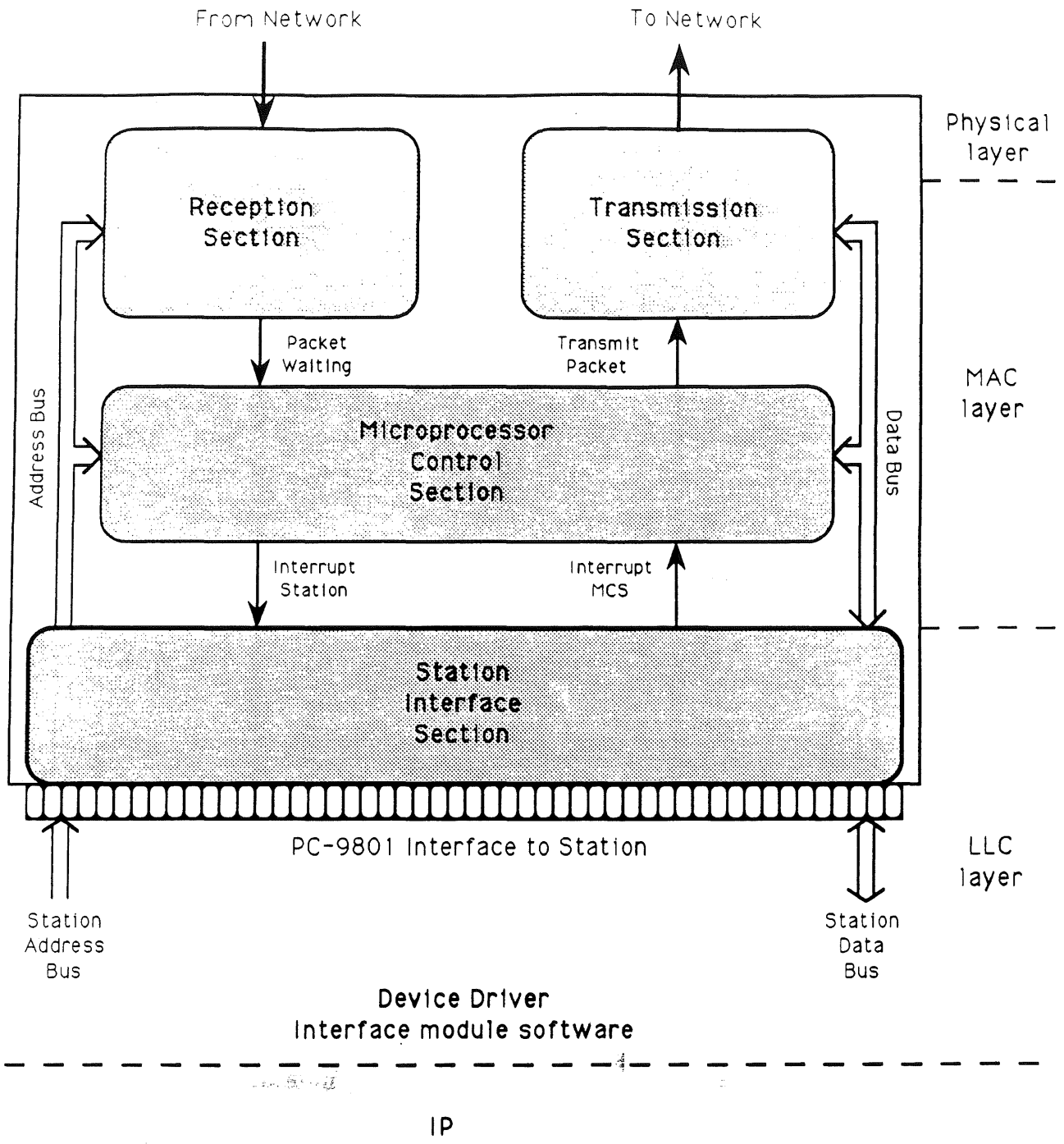

Fig.13 Station/Network Interface Board

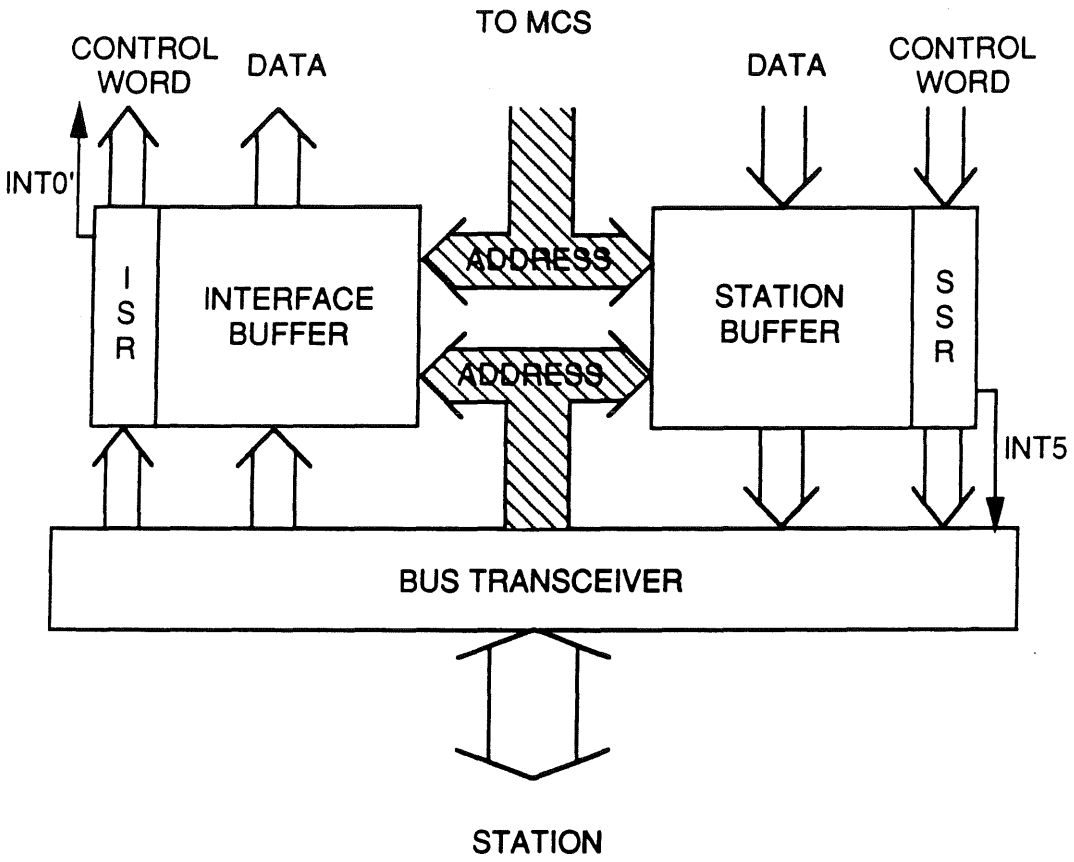

Fig.14 Station Interface Section Block Diagram

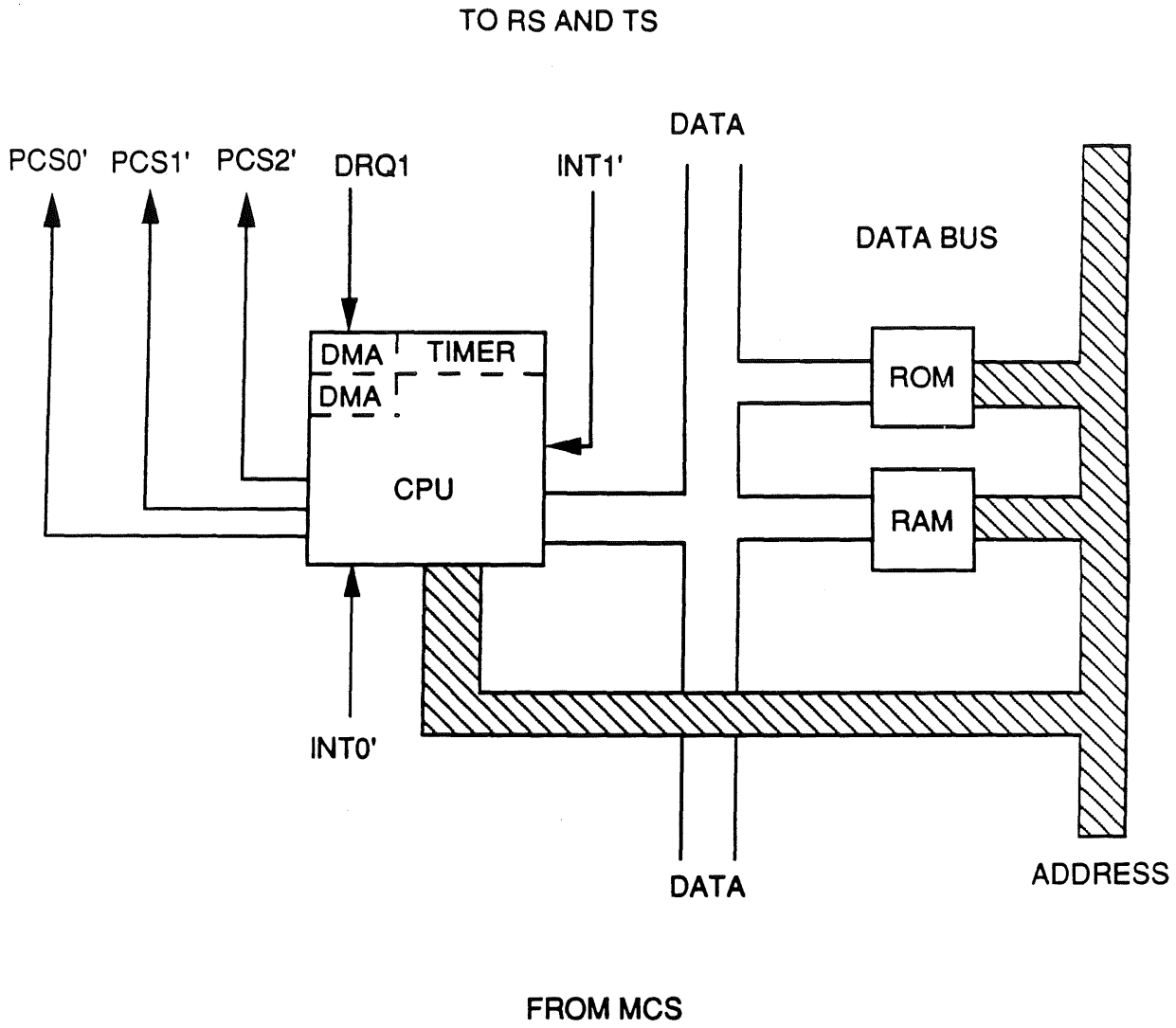

Fig.15 Microprocessor Control Section Block Diagram

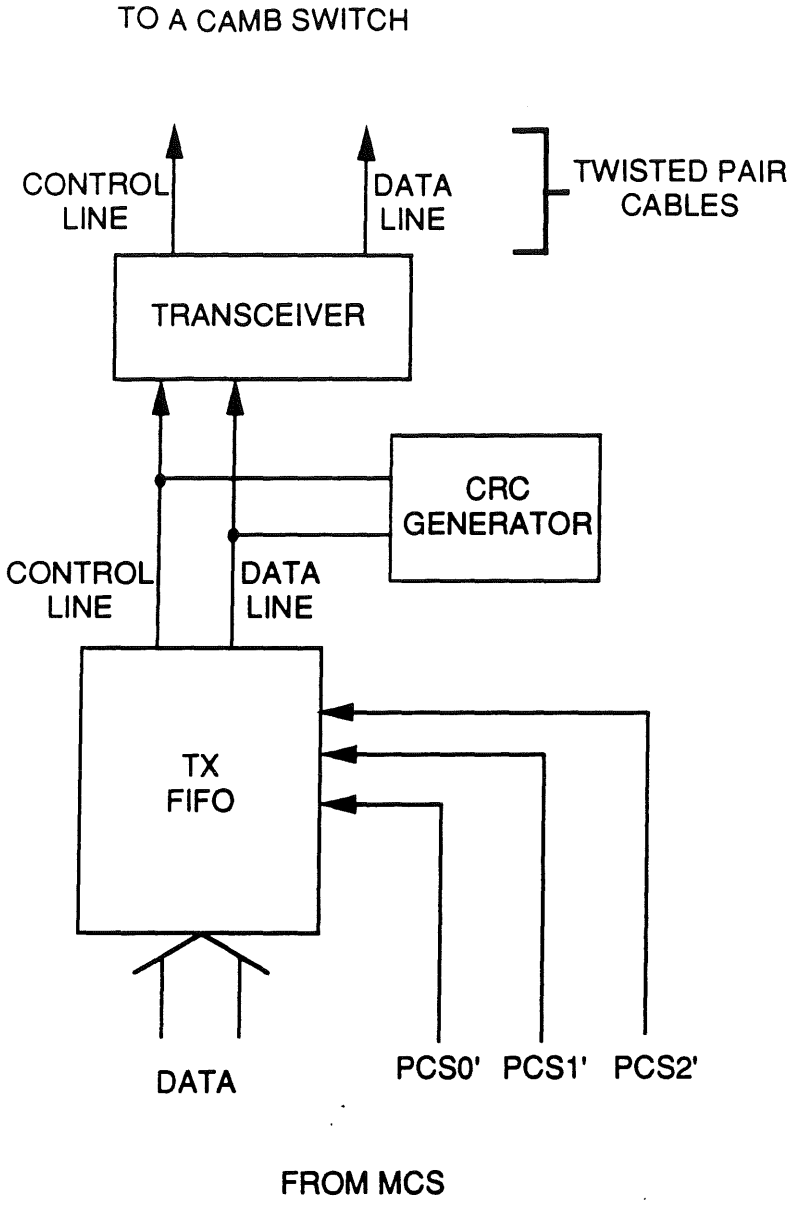

Fig.16 Transmission Section Block Diagram

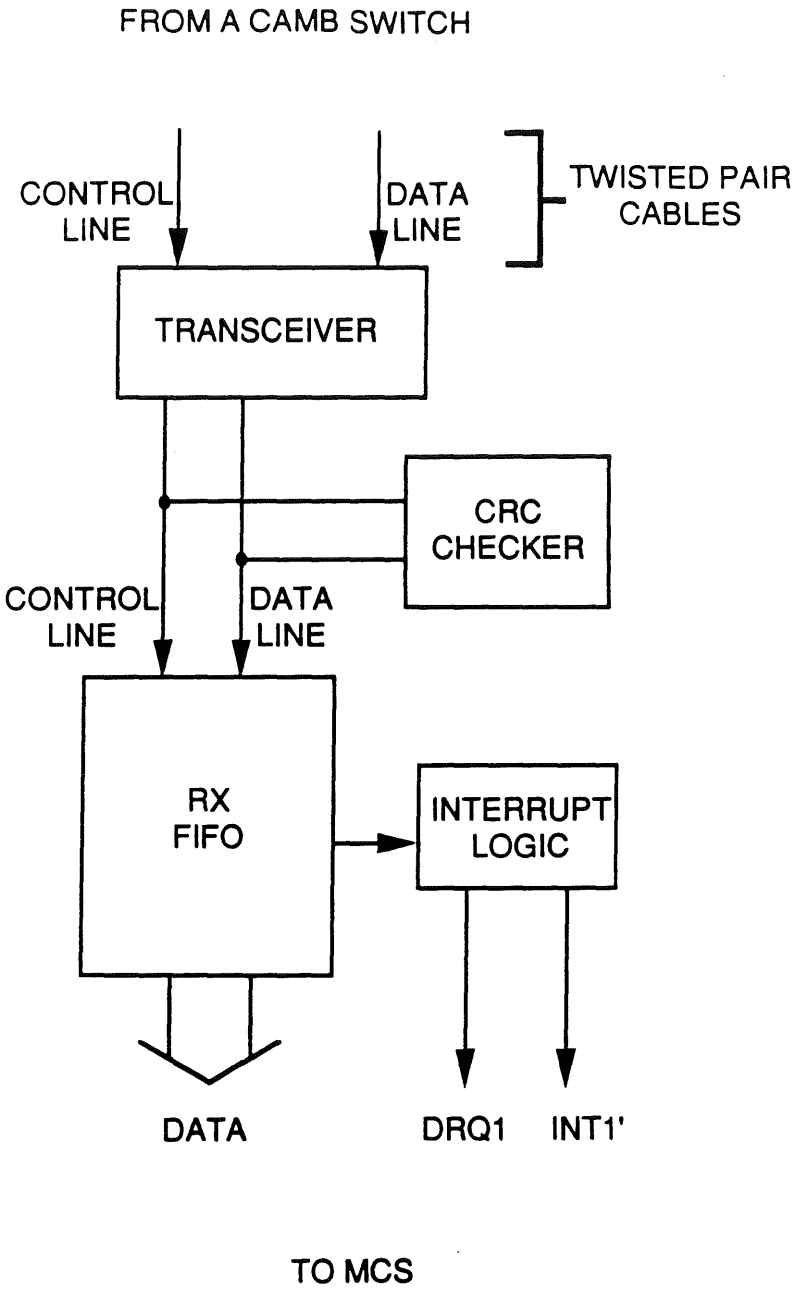

Fig.17 Reception Section Block Diagram

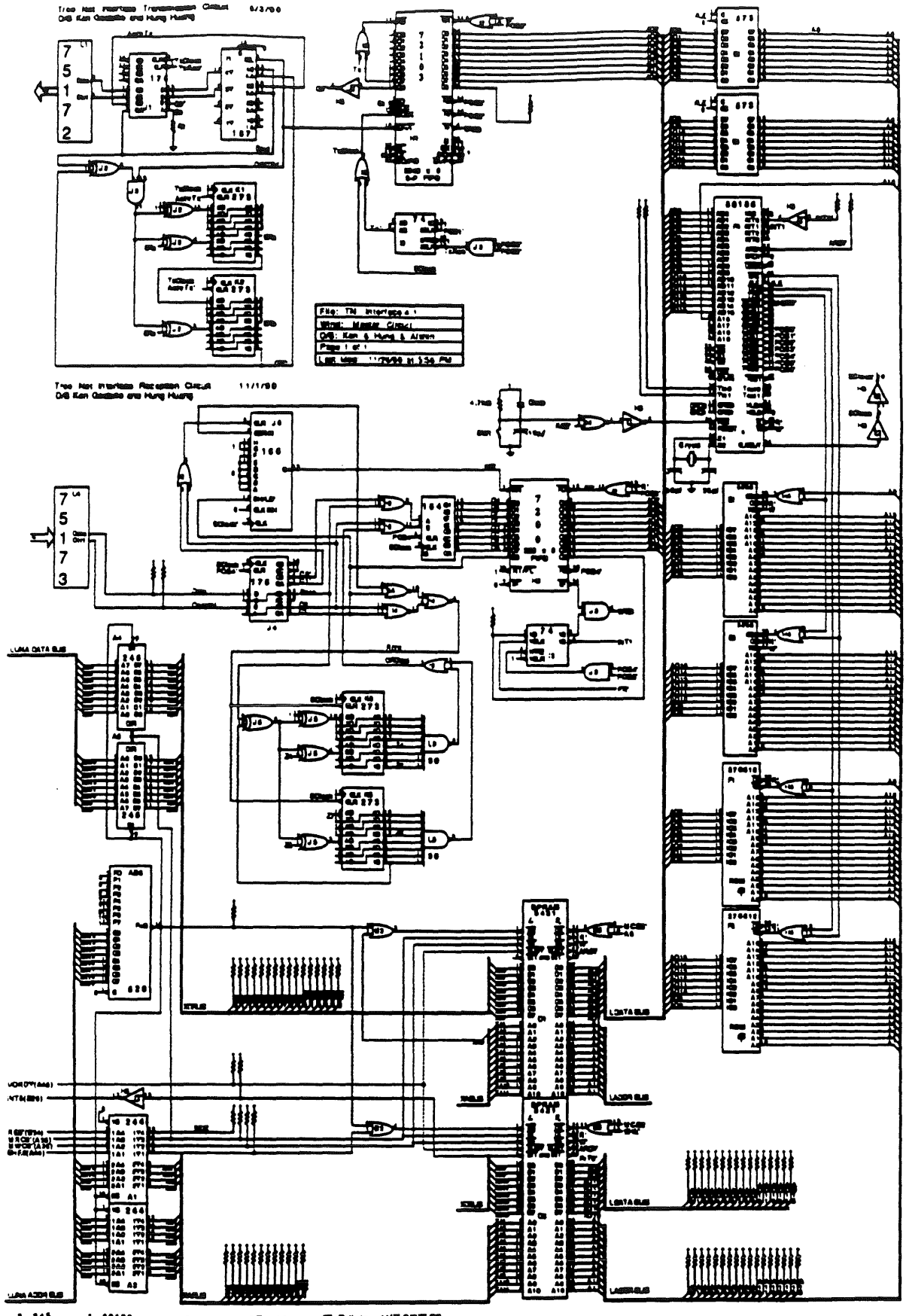

Fig.18 Station/Interface Circuit Schematic