NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

DESIGN OF THE H-NETWORK FOR THE HOMOGENEOUS MULTIPROCESSOR SYSTEM

Nando Di Giambattista

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the degree of Master of Engineering at
Concordia University
Montreal, Quebec, Canada

August 1989

ISBN 0-315-51365-9

ABSTRACT

Design of the H-Network
for the Homogeneous Multiprocessor System

Nando Di Giambattista

The Homogeneous Multiprocessor system is a hybrid of tightly coupled and loosely coupled architecture. This work presents the design of the packet switched H-Network used in the Homogeneous Multiprocessor system. The H-Network uses distributed control in a bus architecture.

It is similar to a Carrier Sense Multiple Access with Collision Detection (CSMA/CD) network. However collisions during data transmission are completely eliminated through prior confirmation of a single network master. This improves network utilization since no data is ever lost due to a collision.

The logical design of the H-Network and the H-Station is presented. The network protocol and the packet format is detailed. A custom microcontroller is developed to implement the network access protocol.

ACKNOWLEDGEMENTS

Sincere thanks to Dr. Nikitas Dimopoulos, for all his guidance, advice, and incredible patience throughout the preparation of this thesis.

TABLE OF CONTENTS

		Page
ABSTRACT		iii
ACKNOWLEDGEMENTS		iv
TABLE OF CONTENTS		v
LIST OF FIGURES		vii
CHAPTER 1:	Introduction	1
1.1	Architecture of The Homogeneous Multiprocessor	1
CHAPTER 2:	Local Area Networks	4
2.1	Overview	4
2.2	Elements of A Local Area Network	4
2.3	Network Topology	5
2.3.1	Star Topology	5
2.3.2	Ring Topology	7
2.3.3	Bus Topology	8
2.4	Network Control Mechanism	9
2.4.1	Ring Networks	9
2.4.2	Bus Networks	11
2.4.3	The Ethernet Network	14
CHAPTER 3:	The Homogeneous Network	17
3.1	Overview	17
3.2	H-NETWORK Design Objectives	17
3.3	H-NETWORK Components	19
3.4	H-Station Architecture	22

		Page
3.4.1	H-NETWORK Bus Definition	22
3.4.2	H-Network Protocol	24
3.4.3	H-NETWORK Packet Structure	29
3.4.4	H-NETWORK Transmit Module	30
3.4.5	H-NETWORK Receive Module	34
CHAPTER 4:	Homogeneous Network Implementation	40
4.1	Overview	40
4.2	Transmit Module	40
4.2.1	Output FIFO Buffer	42
4.2.2	Transmit Controller	45
4.2.3	Test & Set	56
4.2.4	Transmit ID Compare	58
4.2.5	HBUS CLOCK Module	61
4.3	Receive Module	67
4.3.1	Input FIFO Buffer	69
4.3.2	Receive Decision Logic	69
4.3.3	Input FIFO Buffer	69
4.4	Transmit Controller Microcode	75
CHAPTER 5:	Conclusion	82
5.1	Conclusions and Discussion	82
REFERENCES		84
APPENDIX A:	Transmit Algorithm Microcode Listing	87
APPENDIX B:	Schematic of H-Network Node Prototype	90
APPENDIX C:	Microcontrollers and Microcoding	99

LIST OF FIGURES

		Page
Figure 1.1	The Homogeneous Multiprocessor	3
Figure 2.1	Star Topology	6
Figure 2.2	Ring Topology	6
Figure 2.3	Bus Topology	12
Figure 2.4	Bus Topology with Central Controller	12
Figure 2.5	Ethernet Network	15
Figure 3.1	The Homogeneous Multiprocessor Architecture	18
Figure 3.2	H-Station	20
Figure 3.3	H-NET Packet Structure	28
Figure 3.4	H-NET Transmit Block Diagram	31
Figure 3.5	H-NET Receive Block Diagram	35
Figure 4.1	H-NET Transmit Module Block Diagram	41
Figure 4.2	Output FIFO for n bit wide host	43
Figure 4.3	FIFO Timing Waveforms	44
Figure 4.4	Generic Microcontroller Block Diagram	46
Figure 4.5	Transmit Microcontroller Block Diagram	48
Figure 4.6	Microcode Instruction Breakdown	52
Figure 4.7	H-NET Transmit TEST & SET Module	57
Figure 4.8	H-NET Transmit ID Compare Module	59
Figure 4.9	HBUS-CLOCK Module	62
Figure 4.10	HBUS-CLOCK Timing at Start of Packet Tx	65
Figure 4.11	HBUS-CLOCK Timing at End of Packet Tx	66
Figure 4.12	H-Network Receive Module Block Diagram	68
Figure 4.13	Output FIFO for n bit wide host	70

		Page
Figure 4.14	Receive Decision Logic Block Diagram	73
Figure 4.15	Packet Not Destined to this Node	76
Figure 4.16	Packet Destined, FIFO is Free	77
Figure 4.17	Packet Destined, but FIFO not Free	78
Figure 4.18	Microcode State Transition Diagram	79

CHAPTER 1

INTRODUCTION

1.1 ARCHITECTURE OF THE HOMOGENEOUS MULTIPROCESSOR

All multiprocessor systems provide inter-node communication paths. The various approaches yield direct communication paths only to neighboring nodes or direct paths between any two nodes through either crossbar switching or a bus structure.

The Homogeneous Multiprocessor [8,9,10] is a tightly coupled MIMD architecture, as seen in figure 1.1. It includes a network to directly provide complete connectivity. All nodes consist of identical processing units, each with local memory. A node has direct communication to its two neighboring nodes through shared memory. Access to the shared memory is at full processor speed. Communication between all non-neighboring nodes is through a high speed packet switched network.

Such an architecture is most effective when software tasks requiring a significant amount of communication are assigned to neighboring nodes. Neighboring nodes can directly access the others memory by closing a switch to form an extended bus. Once the switch is closed, memory access is at full processor speed. The station controller operates the switch and guarantees mutually exclusive and deadlock free operation. The algorithm for switch operation is found in reference [10].

In the Homogeneous Processor Network, the H-Network, provides the pathway for communication between any distant nodes. the H-Network is a packet switched network. Only one node may transmit at a time. One or more nodes can receive the transmitted packet. The throughput of the H-Network, including packet processing overhead, is lower compared to the extended bus data rates.

This work will deal with the design of the H-Network. It will include the logic design of the FIFOs, the network, and the microcontroller.

An overview of existing networks is presented in Chapter 2. The various topologies and control mechanisms are described. The CSMA/CD protocol is also defined.

Chapter 3 presents the architecture of the Homogeneous Network in detail. Each module of the network is described. The low level algorithms needed to implement the network protocol are also presented.

Chapter 4 presents the detailed design of the actual network. Emphasis is on the logical design and not on the specific implementation.

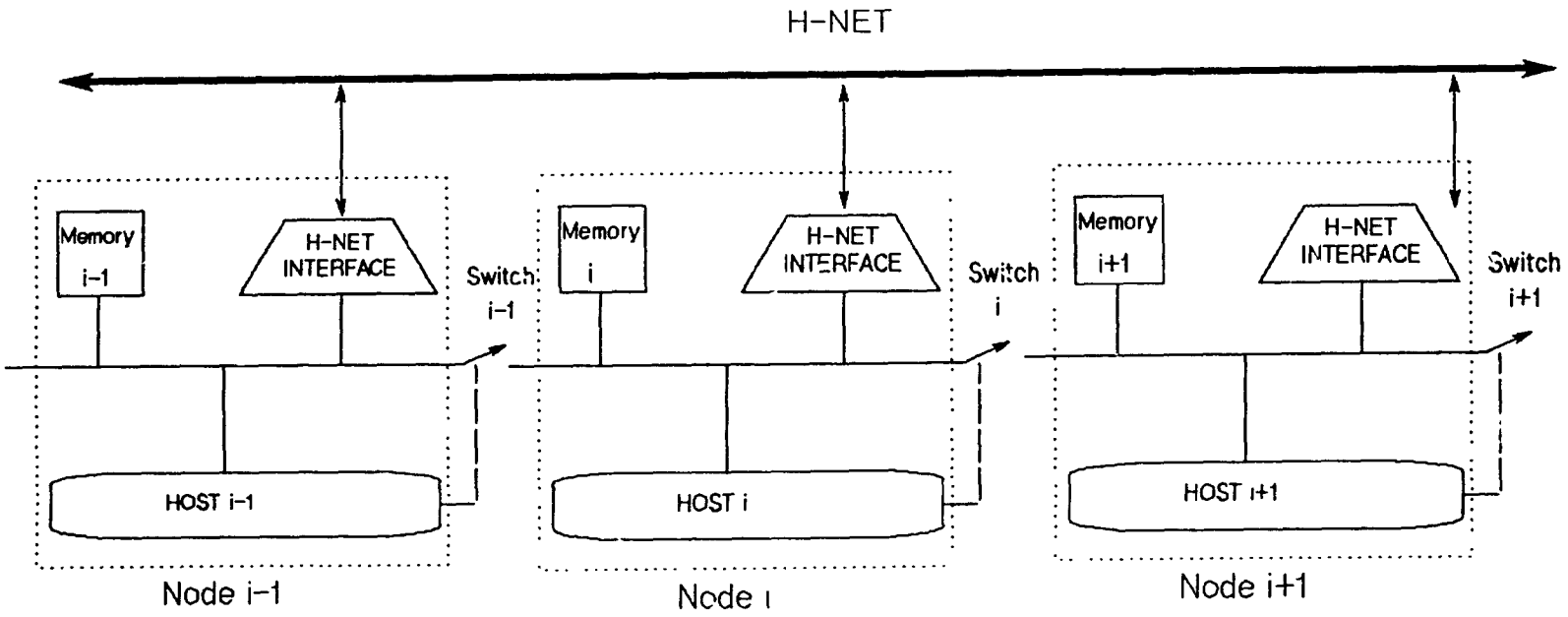


Figure 1.1 The HOMOGENEOUS MULTIPROCESSOR

CHAPTER 2

LOCAL AREA NETWORKS

2.1 OVERVIEW

This chapter is a survey of a number of existing local area networks. The various topologies and access schemes will be examined. The strengths and weaknesses of each will be highlighted. With this background, the H-Network can be understood in a proper perspective.

2.2 Elements of a Local Area Network

Networks are considered Local Area Networks if they are geographically restricted to between 10 and 100 meters with bit rates on the order of 10 Mbps. At its most basic level, a Local Area Network , LAN, consists of a transmission medium, a mechanism for control of transmission, and an interface to the node [6]. A network protocol is implemented at a higher level, through software, to access the actual hardware.

The transmission medium can be twisted pair wire, coax, or fiber optics. Criteria for choosing the medium includes data rates, noise immunity, attenuation and cost. The mechanism for controlling the network is strictly a function of the protocol and topology. Its function is to coordinate the actual transmission onto the network. The low level algorithm for network usage is implemented in the controller. The node interface provides connection to the node. It implements the node-network interface. This

interface may adhere to some data transmission standard, such as RS232, or implement a custom protocol at the node.

2.3 Network Topology

Network topology is the pattern of paths between the processor[7]. The most general case is the complete graph solution which provides a dedicated path between any two nodes. A STAR TOPOLOGY will have all nodes connecting to a central node, where switching takes place. A RING TOPOLOGY connects each node to exactly two others so that a circular path is formed. Messages are forwarded in a unidirectional sense around the network. A BUS TOPOLOGY allows all nodes to simultaneously receive a transmitted message.

2.3.1 Star Topology

The star topology is characterized by a dedicated path between a central node and any other node. The central node can communicate with all other nodes individually, or it may be a switching center providing communication between any two nodes. Figure 2.1 shows a typical arrangement.

A star network is often used to connect many peripherals to a mainframe computer. The mainframe forms the central node and communicates with each peripheral individually. Any two nodes could communicate by passing messages through the mainframe.

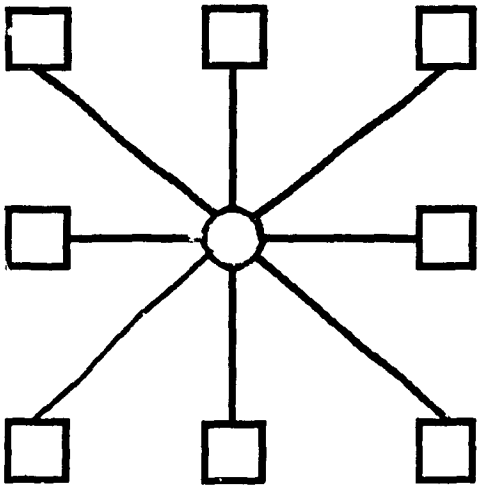


Figure 2.1 Star Topology

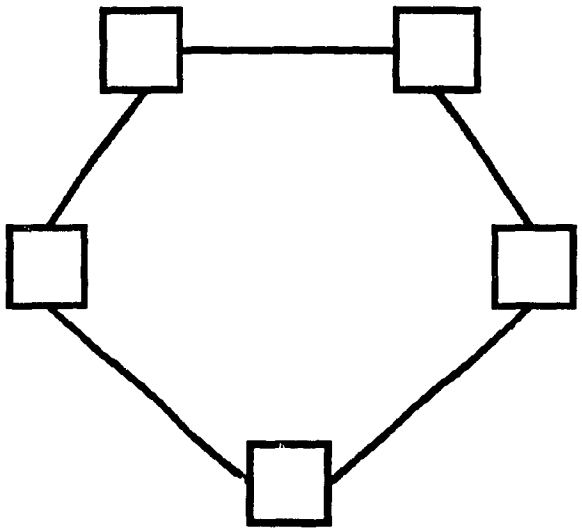


Figure 2.2 Ring Topology

Alternatively, the central node could be a pure message switching center. Its role would be to receive messages from one node and transmit to the destination node. The central node could consist of a store and forward computer or simply a circuit switching device.

The strength of a star network is flexibility in defining the interface between the central and peripheral nodes. A computer based central node could support many peripherals each with different speeds and protocols. Addition and removal of nodes is trivial since it doesn't affect other nodes. Message delivery time is not a function of the topology but of the central node. A failure in a given path or paths would only compromise communications to the respective nodes. On the negative side, the central node can become a bottle neck. Failure in the central node would mean the collapse of the entire network.

An example of a Star Network is the Local Backbone Network [1] designed at Concordia University. It uses optical links running at 40 Mbs to connect the nodes to a central switch. The system is suited for integrated voice and data applications.

2.3.2 Ring Topology

In a ring topology, each node is connected to exactly two other nodes, forming a circular path. An example is shown in figure 2.2. Most rings provide for a

unidirectional flow along these paths. A message is passed to the next node in the path. Each message will contain a destination address. The destination node will recognize the address and capture the message. All other nodes will simply send the message unaltered to the next node.

The obvious weakness in the ring topology is any single failure in either a node or the transmission media collapses the entire network. Addition or removal of nodes is intrusive to the network. Message delivery time increases as more nodes are added.

2.3.3 Bus Topology

In a bus topology, all nodes connect to a common bus, as shown in figure 2.3. A node can seize the bus and broadcast a message. The message contains a destination address. All nodes that are addressed will receive the message.

Failure of any node will not affect the network. The message delivery time is a function of the physical length of the bus and not the number of nodes. Addition and deletion of nodes is not disruptive. Failure of the node-bus interface such that the bus is held in a fixed state will collapse the entire network. A break in the bus can result into two smaller working networks.

2.4 Network Control Mechanisms

Rules are needed to decide which node is allowed to transmit at a given time. In a star network, the central node handles the task by servicing one node at a time. A ring or bus network may employ a central or distributive control scheme. The following sections will examine central control, ring control and contention control mechanisms for the ring and bus networks.

2.4.1 Ring Networks

A distributed control mechanism is inherent in a Ring topology. Each node is capable of copying messages with its destination address. A node receiving a message will either mark the message as received and send it on to other nodes or remove the message from the network.

Many schemes exist for authorizing a node to send a message. In the MESSAGE SLOT method, a fixed set of empty message slots circulates around the network. A node can insert a message only into an empty slot. The message is passed to all nodes where it may be copied and marked as received by all addressed nodes. The originating node can then remove the message and determine which nodes have successfully received the message. A similar scheme allows a node to simply add its message to the end of the train of messages. The Cambridge Digital Ring [15] follows this scheme. It includes a dedicated Monitor Station as one of

the nodes, to supervise the network, maintain statistics and initialize the network when needed.

TOKEN CONTROL consists of passing a token around the ring. Only upon receiving the token, may a node transmit its message followed by the token. If the node has no message, it simply forwards the token to the next node. The IBM Token-Ring [2] is an example of a Token control mechanism.

A third method, REGISTER INSERTION, uses shift registers to hold the message. The node can splice the shift register in series with the network at the end of a message or token. Alternatively, the shift register is switched into the network during a quiet period. The message then shifts out onto the network. The register is switched out of the network path when the transmitted message has returned.

Ring networks have inherent complications and reliability problems. Both the message slot and token method require initial generation of these control entities. Furthermore, since the token or message slots can be corrupted or lost, a method of regeneration is needed. A large waiting delay is experienced under light loads as the token or slots circulate through all the idle nodes. The REGISTER INSERTION reduces reliability since it places a register directly in series with the network path. In general, any significant failure at any node will collapse the network control mechanism.

LOOP NETWORK [7], have a similar topology to ring networks but have a centralized control mechanism. This avoids some of the shortcomings of a distributed control mechanism.

2.4.2 Bus Networks

A CENTRAL control strategy is possible in a bus network by using a topology similar to figure 2.4. The controller would coordinate the use of the bus. A DISTRIBUTED strategy could be based on token passing or contention principles. The token passing approach would yield roughly the same weakness as a ring network. Contention networks allow nodes to transmit onto a common bus. A COLLISION occurs if two or more nodes transmit at the same time.

In a contention network, detection of a collision indicates failure of the attempted packet transmission. Upon detecting a collision, that node will jam the bus so that all nodes realize a collision has occurred. The collided packet is re-queued for transmission at a later time.

The simplest strategy is to transmit a packet as soon as it is available [5]. If after a fixed interval no acknowledgment is received, the packet is retransmitted. A high rate of collisions occurs when many nodes have packets to transmit. Improvement is seen if packets are transmitted only during fixed time slots. In such a case,

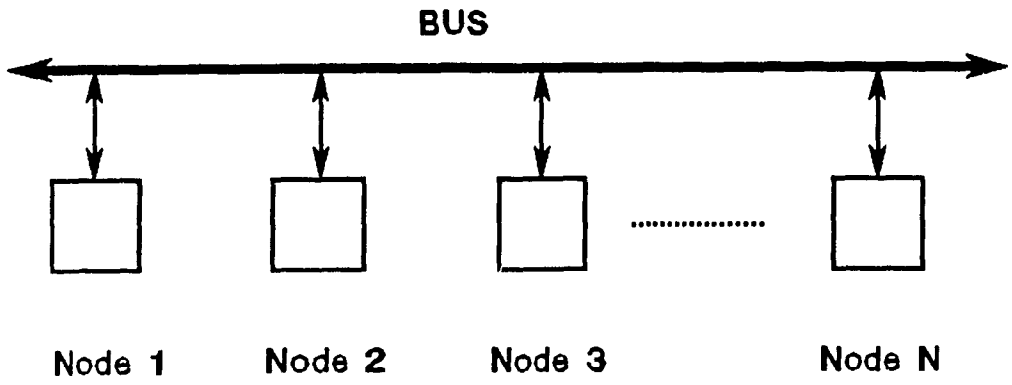


Figure 2.3 Bus Topology

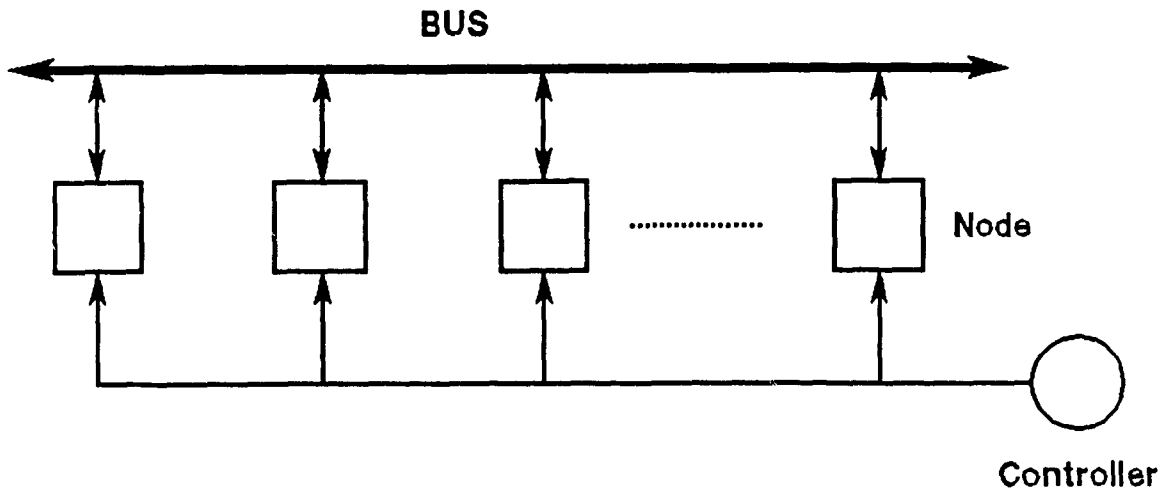


Figure 2.4 Bus Topology with Central Controller

colliding packets will overlap entirely, reducing the chance of mid-packet collision.

A further improvement is found in Carrier Sense Multiple Access, CSMA, systems [24]. CSMA requires nodes to check the network for activity prior to transmitting. This entirely avoids packets colliding midway. Collisions will occur when two or more stations find the network inactive at the same time. In such a case, those nodes will tie up the network for the duration of packet transmission even though the packets are garbled.

Carrier Sense Multiple Access with Collision Detection, CSMA/CD [17,19], adds a listen during transmission utility to CSMA. Collisions during transmission are thus detected and an abort takes place. This frees up the network as soon as a collision is detected. Performance improves since the duration of a collision is shortened to a minimum.

Upon finding the network busy, a node has several options. It can reschedule, or keep trying till the network is idle and then decide to transmit or not. NONPERSISTENT CSMA will transmit if the network is idle or reschedule the attempt for a later time.

In 1 PERSISTENT CSMA [17], the node transmits if the network is idle otherwise it persists in checking the network till it is idle. As soon as the network is idle the node transmits. However, if two or more nodes find the network busy, they will all be waiting for the network to

become idle. When the network does become idle, all waiting nodes will begin transmitting and a collision will result.

The more general case is P PERSISTENT CSMA. Upon finding the network idle, the node will transmit with probability p, or wait with probability 1-p. If after the waiting period the network is still idle the decision process is repeated, otherwise the attempt is rescheduled. The decision process effectively reduces the chances of collision when several nodes are waiting for the network to go idle.

Bus networks with distributed control allow easy addition and removal of nodes. The media tends to be reliable since it has no active elements. In contention networks there is no risk of failure due to loss of token. A failed node will not impair network operations. However failure in the control mechanism to release the bus would collapse the entire network.

2.4.3 The Ethernet Network

The Ethernet Network,[19,22], was developed by the XEROX Corporation. It is instructive to examine this implementation. The Ethernet was designed as a low cost, high speed LAN. It can support 100 nodes along a kilometer of coaxial cable. Packets are sent using CSMA/CD techniques. The network has an unrooted tree topology allowing extension from any point in any direction.

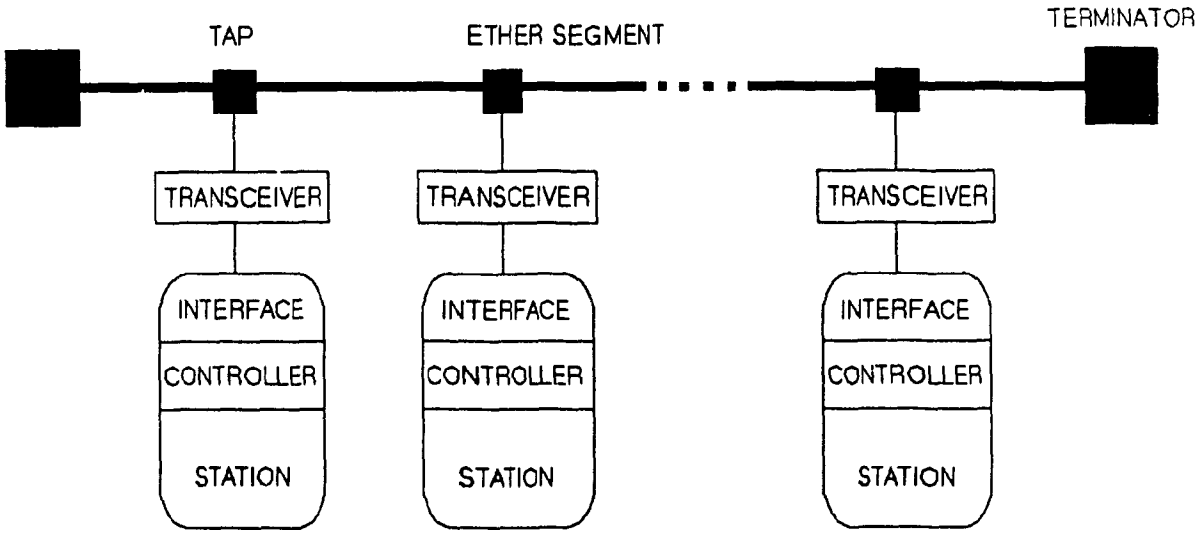


Figure 2.5 ETHERNET NETWORK

The network is shown in figure 2.5. Each node connects to the ETHER, a passive medium. It can take the physical form of coaxial cable, twisted pair or even optical fiber. Each station interfaces to the network through a controller and interface. The interface includes CRC generation hardware. It accepts only packets destined for its node. The controller implements the protocol for sending and receiving packets. It also controls rescheduling of collided packets.

Control is distributed among the nodes. Each packet has the destination address at the head of the packet. Addressed nodes receive the broadcast packet. A 16 bit cyclic redundancy checksum is appended to the end of the packet by the interface module.

A node will first check that the network is idle before beginning transmission. If two or more stations are transmitting, the collision of their packets will be detected and an immediate abort will take place. Once a collision has occurred, the next attempt will be after a dynamically chosen random delay. This random delay minimizes the chances that the just collided stations will all attempt to retransmit at the same time.

CHAPTER 3

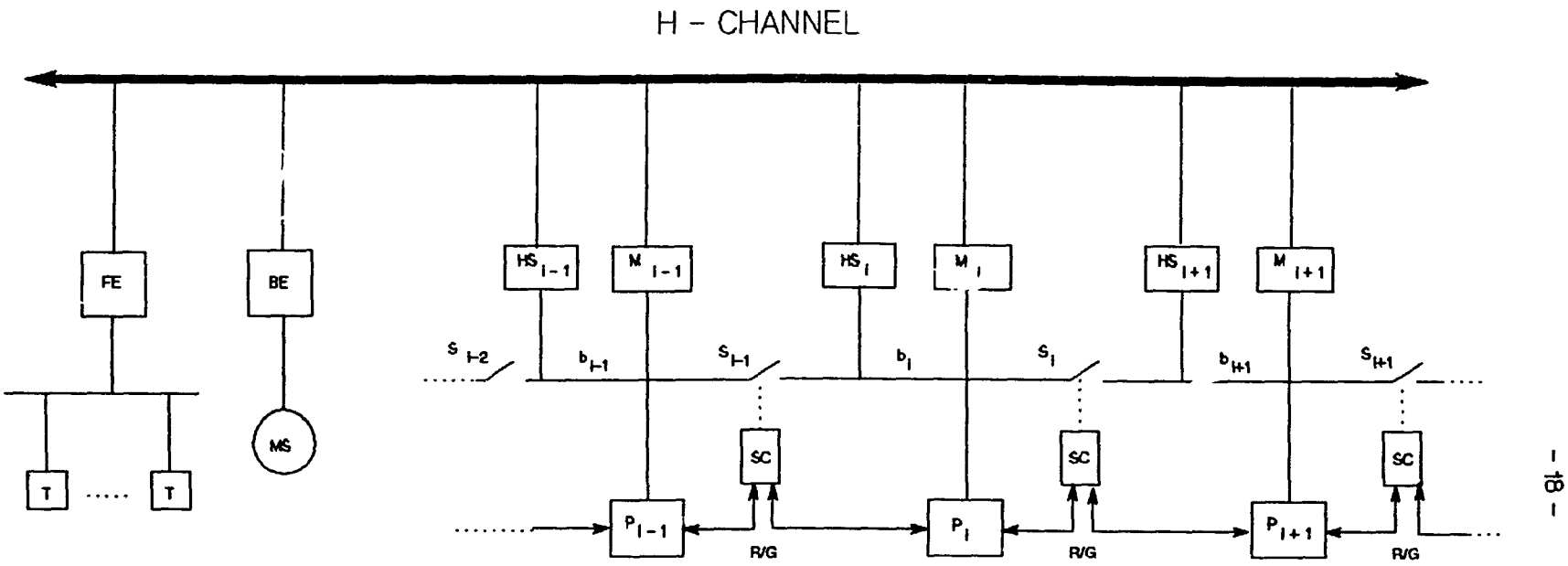
THE HOMOGENEOUS NETWORK

3.1 OVERVIEW

This chapter describes the function of the Homogeneous Network, H-Network. The architecture to implement these functions will be explained. A network acquisition algorithm will also be defined.

3.2 H-NETWORK DESIGN OBJECTIVES

The Homogeneous Multiprocessor is a tightly coupled MIMD machine. An overview of the system is shown in figure 3.1. The H-Network ties all components of the system together. The system includes a front end for terminal access and a back end for mass storage support. Each node consists of a processor with local memory. Each processor is tightly coupled to its two immediate neighbors through a switched memory bus. A node must close the switch on the bus to form an EXTENDED BUS, allowing direct access to its neighbor's memory. The switch controller ensures mutually exclusive switch closures at any given node. Detailed descriptions of the Homogeneous Multiprocessor and its architecture is found in references [8,10,11]
Communications between any two non-neighboring nodes is through the H-Network, a packet switched network.
The H-Network is a bus topology local area network having distributed control. The access protocol is a variation on CSMA/CD. Separate lines are used for data paths and for



M: Memory
P: Processor
FE: Front End
BE: Back End

MS: Mass Storage
T: Terminal
S: Bus Switch
SC: Switch Controller

HS: Network Station
R/G: Bus Request/grant
b: Local Bus

FIGURE 3.1: The Homogeneous Multiprocessor Architecture

each control signal. The data path is as wide as the host computer's word width. Using separate lines for each signal allows for baseband operation, easy adaptation to different word widths and higher throughput. This parallel approach avoids having to serialize data and control signals. This in turn allows for the parallel operation of both data transmission and control, which leads to performance improvements. The data rate exceeds 7 Mbytes per second.

The bus topology yields full connectivity and ease in addition or deletion of nodes. The bus is highly reliable as it is strictly passive. Failure at one node will not collapse the bus.

3.3 H-Network COMPONENTS

The H-Network provides a complete packet delivery system. As seen in figure 3.2, each H-Network station consists of:

- a host specific interface
- a transmit FIFO buffer
- a Station Controller
- transmission media
- receive arbitration logic
- a receive FIFO buffer

The H-STATION provides the host specific interface between an H-Network and the host. It provides any logic level translations, buffers and returns the H-Network status

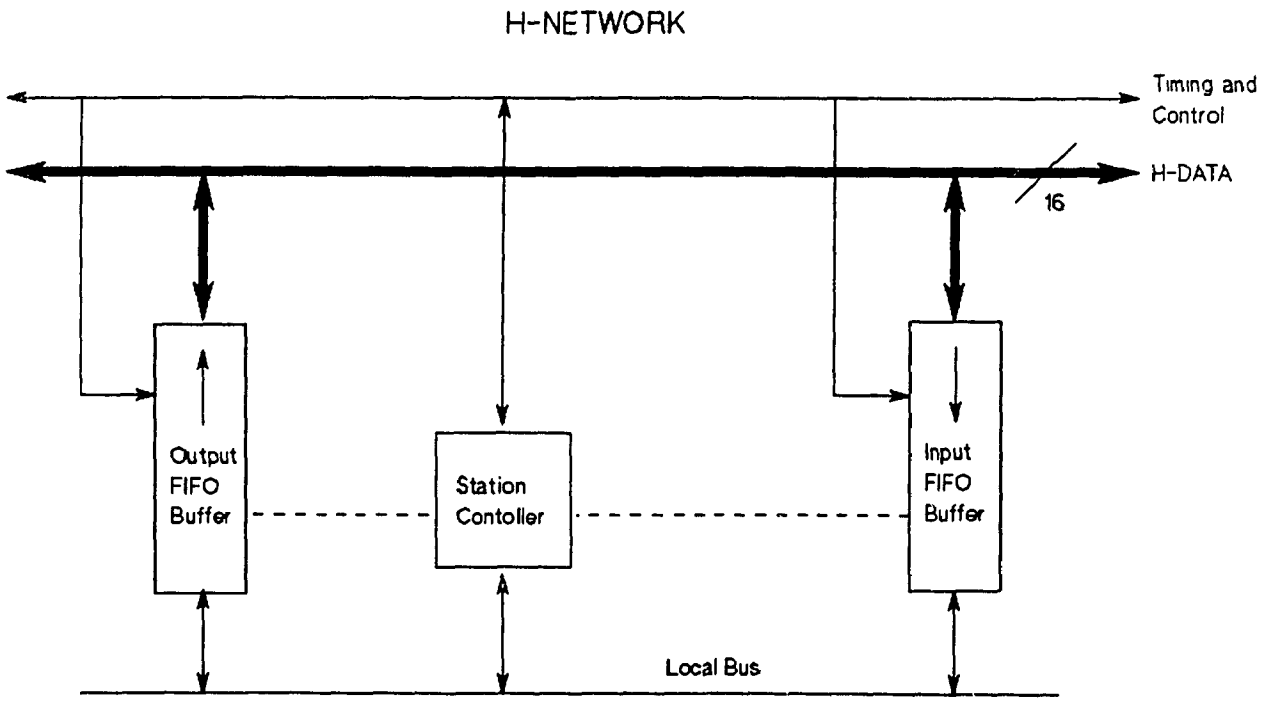


Figure 3.2 H-Station.

bits. The interface is customized for the chosen host processor. Host processor access to the H-Network is strictly limited to the H-Network-host interface.

The transmit FIFO buffer receives a packet from the host and holds it till permission to transmit is granted. The packet is then clocked out at a fast rate onto the H-Network data lines. This particular FIFO is built to have the same word size as the host processor. The depth of the buffer is presently set to 128 words and the width to 16 bits for this design. The FIFO design allows easy expansion of word width in 4 bit increments. Similarly the FIFO depth may be incremented in steps of 64. For higher performance, the design can support multiple transmit FIFOs.

The Station Controller uses a microcontroller to implement the network protocol algorithm. It acquires mastership of the network, initiates transmission and releases the network. The microcontroller uses horizontal coding [13,18] of instructions. It provides the means to experiment with different protocols. The microcontroller could be replaced with a hardwired state machine for a fixed network access algorithm.

The transmission media used is a twisted pair ribbon cable. Signals on the lines are TTL logic levels from either tri-state or open collector drivers. An alternate approach would be to use differential drivers. Although

the number of control lines are set, the number of data lines is chosen to equal the word size.

On the receive end, discrete logic is used to decide if the packet should be received. The decision is based on the destination of packet and availability of an empty receive FIFO. If an empty receive FIFO is not available when needed, then an abort of the transmission is forced. The transmitting station will release the network and retry at a later time. Improved network performance may be achieved by having multiple receive FIFOs.

3.4 H-STATION ARCHITECTURE

This section will define the architecture of an H-STATION. Although each node will have receive and transmit capability, the two functions are modular at a node. The support logic and controller for the transmit section has no direct interaction with the receive section of the interface.

3.4.1 H-NETWORK BUS DEFINITION

The H-Network must carry both data and control signals. Separate lines are used for each signal, allowing for high speeds and simplified node hardware. Signal names preceded by the backslash symbol, / , denote an active low signal.

All signals are either TRI-STATE or WIRED-OR logic. Only one node may drive a TRI-STATE line at any given instance by enabling the driver. All other TRI-STATE

drivers must be disabled or contention will occur. A disabled driver presents a high impedance load to the line.

WIRED-OR logic uses drivers with Open Collector outputs. Such a driver can only drive a line to a logic LOW by grounding it. A logic HIGH is provided by a pull up resistor. Many open collector drivers may be directly connected to the same line. The line will be low if any of the drivers are asserted. In this fashion, a logical OR function is provided.

/HBUS-ACTV (Wired-OR, Active Low) Asserted by the network master to indicate network is active.

/HBUS-COLLIS (Wired-OR, Active Low) Asserted to indicate a network collision. A collision occurs if more than one node believes it is the Network Master. A collision also arises if a destination node has no free receive FIFO

/HBUS-DEST (Wired-OR, Active Low) Asserted to indicate the Packet Destination Address is on the HBUS-DATA lines.

/HBUS-DECIS (Wired-OR, Active Low) Asserted when all nodes have decided to permit packet transmission

HBUS-DATA (TRI-STATE) Data Bus for packet transfer. This bus will have 16 lines. This bus is also used to carry the Node ID number

during the Master Contention phase.

HBUS-EOP (TRI-STATE) Line carries the End Of. Packet bit.

HBUS-DCLK (TRI-STATE) Clock used to shift packet data on HBUS-DATA into the receive FIFOs

3.4.2 H-NETWORK PROTOCOL

The protocol used here is an evolution of the design advanced by Kehayas [16]. Kehayas proposed using a secondary line for collision detection. A transmitting node would repetitively send out a unique ID number in a serial format, on a dedicated line, during the data transmission phase. By receiving this serial ID, a node can detect a collision due to multiple network masters. If a collision occurs on the serial ID line, then the data packet has also collided. Once detected, the host can retransmit that packet. The design presented in this work avoids packet data collision altogether by first confirming that one and only one network master exists. Further the handshaking scheme between FIFOs has been eliminated, and the data transfer rate increased.

A recent proposal for improving the H-Network performance has been proposed and analyzed by Wong [25] and Phung [20,21]. The scheme calls for ready stations to contend for the network in parallel with the current data packet transmission. Thus the next network master will have been chosen prior to the end of data packet

transmission, and data collisions are eliminated. These ideas are not implemented in this work, but can easily be added on later. Once added, the network utilization will increase over the present protocol.

There are two aspects to the H-Network protocol. The low layer protocol implements the actual packet transmission. The higher layer protocol implements the algorithm for network acquisition. The lower layer protocol implemented by the H-STATION design is rigid as it is done directly in hardware for optimal performance. The lower layer consists of the data FIFOs, the actual data transfer clock signal and the network Test & Set circuitry.

The higher layer protocol is microcoded, allowing easy adaptation to different algorithms. It is based on a microcontroller, which can react to input signals and generate appropriate control signals. This work will implement a CSMA type protocol. It is roughly based on work done by Kehayas [16]. Many modifications and improvements have been added and are presented in this work.

There are four states to the H-Network transmit protocol:

- waiting for a packet from the host
- attempting to become sole master of H-Network
- checking if the destination node is ready to receive
- transmitting the packet.

The normal transmit cycle begins with the H-STATION waiting for a packet from the host. Upon receiving a packet, it will then attempt to become the network master. A test and set module checks the H-Network activity status line. This line will be repeatedly tested till the network is found to be idle. The station then verifies that it is the sole master of the network. A unique node ID number is placed on the data lines and read back after a specified interval. If more than one station is putting out an ID number, a collision will occur. At least one of the stations will detect this collision and assert the HBUS-COLLISION line. A collision will terminate the transmit cycle and the H-Network will become idle again. If the station does read back its own ID and the HBUS-COLLISION line is not asserted, then it has successfully become the sole network master. By resolving multiple masters at this point, absolutely no collisions can occur during the actual packet transmission phase.

Before the packet can be transmitted, a check must be made that all destination nodes have a free receive FIFO buffer. After a node has become the network master, it places the packet destination address on the H-Network data bus. Each node on the network signals its decision to allow the packet to be transmitted through a "wired OR" status line,(ie. HBUS-DECIS). All non-destination nodes will decide to let the packet be transmitted. All destination nodes with a free receive buffer will also

decide to let the packet be transmitted. However any destination node with no free receive buffer will decide to block the transmission and assert the collision line. Depending on the state of HBUS-DECIS and the HBUS-COLLIS line, the packet will either be transmitted or rescheduled for a later attempt. In either case the network is released and becomes idle again.

A checksum is generated at the transmit end and sent as part of the packet. Each receiving station calculates a checksum for the received packet and compares it to the transmitted checksum. A mismatch would indicate a garbled transmission. A retransmission would be requested by the host processor itself. The H-STATION would only signal the host that the checksum was not correct. Similarly, any acknowledgement of received packet would be handled by the host and its operating system.

Whenever a station relinquishes the network due to an ID number collision or receive buffer full condition, it attempts retransmission immediately. Alternatively the protocol may implement a backoff mechanism to reduce the number of failed transmission attempts.

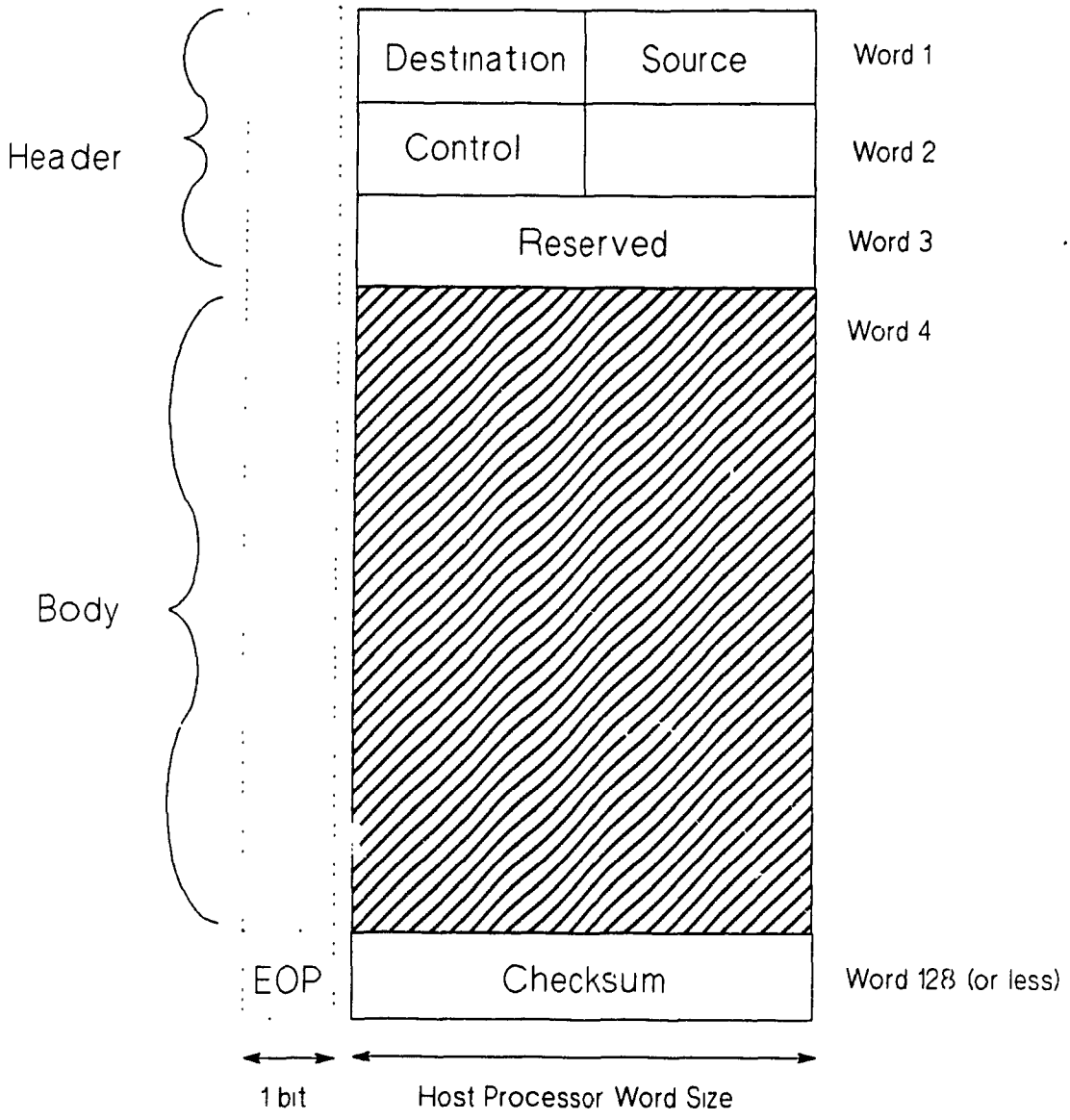


Figure 3.3 HNET Packet Structure

3.4.3 H-NETWORK PACKET STRUCTURE

All communication between nodes over the H-Network take the form of packets. The H-Network packet structure is seen in figure 3.3 [9]. The first word at the top of the packet contains the source and destination address. Nodes make their decision to receive a packet based on the destination address. A destination address of zero denotes a broadcast packet and would be received by all nodes.

The second word denotes the type of packet, the length of the packet, etc. The third word is reserved for use by the higher layer protocol. The body of the packet contains the actual message. The body may be of variable length, depending on the host Operating System.

The final word of the packet is reserved for the checksum. The checksum of the packet header and body is calculated and appended to the packet by the H-STATION. The host Operating System need only restrict the length of packets to one less than the maximum. Thus with a FIFO depth of 128 words, only the first 127 words may be used by the host.

The FIFO depth is chosen to match the packet length. The FIFO width is chosen to be one bit wider than the host processor word size. For example a 17 bit wide FIFO is used for a 16 bit processor. The additional bit is used to hold the End Of Packet (EOP) bit. This EOP bit is set to TRUE only for the last word of the packet. The H-STATION

will set this bit and insert it into the FIFO as part of the checksum word.

The transmitting node uses the EOP bit to terminate transmission. The host at the receiving end will use this bit to detect the end of packet when reading in variable length packets. The receiving node H-STATION will calculate the checksum of the inbound packet. Since the transmitted checksum is included in this calculation, the result will be zero for an intact transmission. The packet must be discarded if a non zero checksum is calculated. Retransmission is requested by the receiving node host.

3.4.4 H-NETWORK TRANSMIT MODULE

The host processor communicates with all non-neighboring processors by sending packets on the H-Network. A long message, such as a file, would be broken up into a number of packets. The host sends a packet by storing it in an empty output FIFO buffer and signaling the H-STATION controller to transmit. Each packet will contain the destination address. The destination may be a single node, a group of nodes or all nodes. A block diagram showing the H-STATION transmit section is shown in figure 3.4. Transmission of a packet originates at the host. The host follows this sequence:

Step 1) Break up the message into packets of 127 words or less. The first word in the packet will be the destination address. The Host

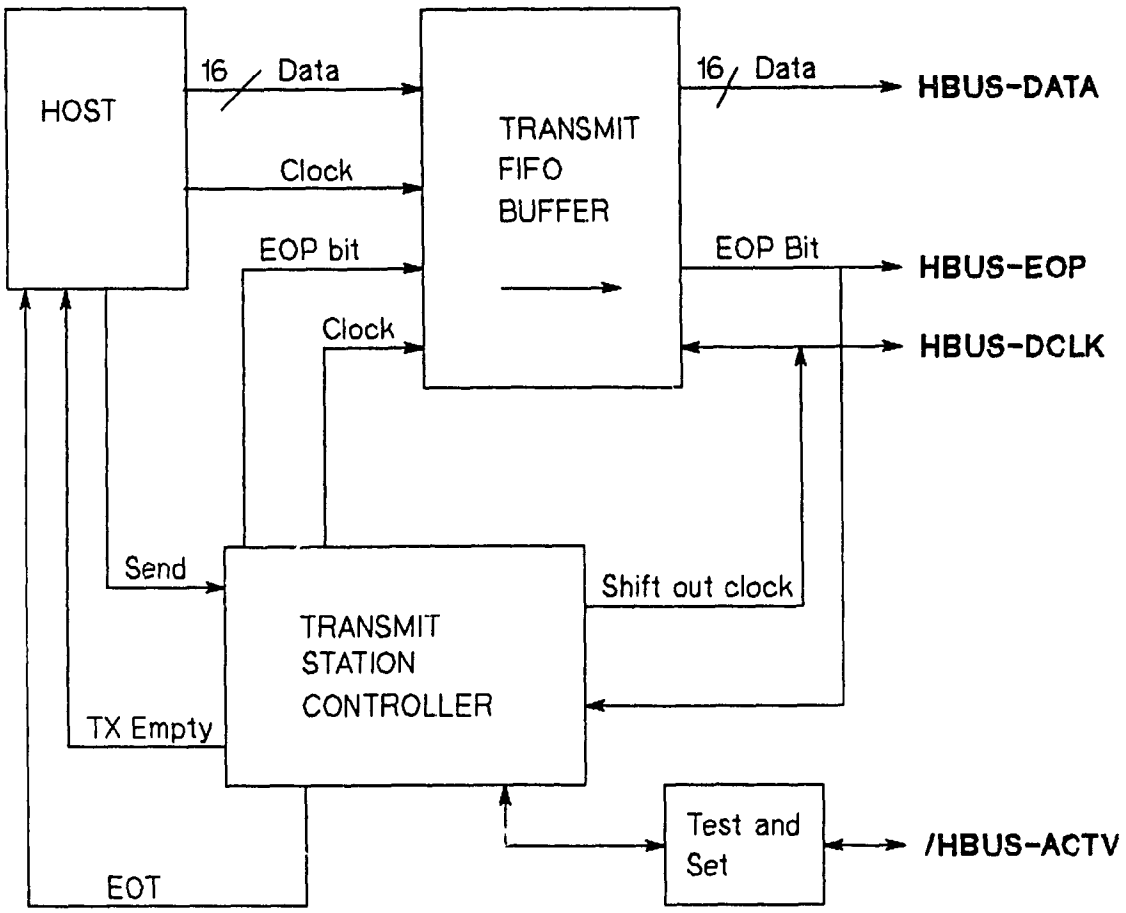


Figure 3.4 HNET Transmit Block Diagram

also adds a sequence number to each packet.

2) Wait till the TRANSMIT FIFO is empty.
   This is done by checking the TX EMPTY status bit from the STATION CONTROLLER.
3) Clock the data into the TRANSMIT FIFO.
4) Signal the STATION CONTROLLER to transmit the packet (by asserting the SEND line).
5) Monitor the End Of Transmission (EOT) status bit, and jump back to step 1 when it is set.
   If the EOT bit is not asserted after a fixed timeout, then host can force a transmit clear by asserting /XTCLR line and jump to step 1

The H-STATION controller implements the network access protocol. The following algorithm is implemented:.

Step 1) Set the TX EMPTY status line to TRUE
   Set the EOT status line to FALSE.
2) Wait for the host to signal that a packet has been loaded into the TRANSMIT FIFO.
   (ie. Monitor the SEND line from the HOST
3) Clock into the transmit FIFO the End Of Packet bit (EOP) as the final word of the packet.
4) Do a TEST and SET of the H-Network active line,/HBUS-ACTV.
5) If the result of the TEST and SET operation

shows that the network was active then reset the TEST and SET module, and go to step 4 back to step 4, else go to step 6.

6) The STATION CONTROLLER has succeeded to become a network master. However, at this point, other nodes may think themselves as masters. To confirm sole mastership, the STATION CONTROLLER places its ID number on to the network data bus (ie. HBUS-DATA).

7) After a time interval, the ID number on the HBUS-DATA lines is read back. If it matches then jump to step 11

8) Since the ID did not match, assert the network collision line, /HBUS-COLLIS, to force all nodes to abort transmission and release the network. Reset the TEST and SET to release the station's mastership of the network

9) Keep the collision line asserted till the network goes IDLE (ie /HBUS-ACTV = FALSE). This will ensure all nodes have relinquished mastership of the network.

10) Remove the ID number from data bus. Remove the network collision condition (ie set the /HBUS-COLLIS = FALSE). Jump back to step 4.

11) At this point we are the sole network master. Remove the node ID number from the network

data bus

12) Enable the first word in the transmit FIFO onto the network data bus. This word is the destination address for this packet

13) Strobe the destination line, /HBUS-DEST. This will latch the destination address into the receiving section of the STATION CONTROLLER at all nodes.

14) Check the network collision line, /HBUS-COLLIS for a TRUE condition. If a collision has occurred, (ie. due to destined node not having a free receive FIFO;, abort the transmission and jump to step 4. Note the packet is still intact in TRANSMIT FIFO.

15) Since no collision has occurred, the STATION CONTROLLER now waits for the decision line to go TRUE to start the transmission.

16) The STATION CONTROLLER need only wait for the End Of Transmission (ie EOT = TRUE), and then jump to step 1.

3.4.5 H-NETWORK RECEIVE MODULE

The H-Network RECEIVE module of the H-STATION is shown in figure 3.5. The Receive FIFO sits between the HBUS and the host processor. The Station Controller implements the receive portion of the H-Network protocol. The receive

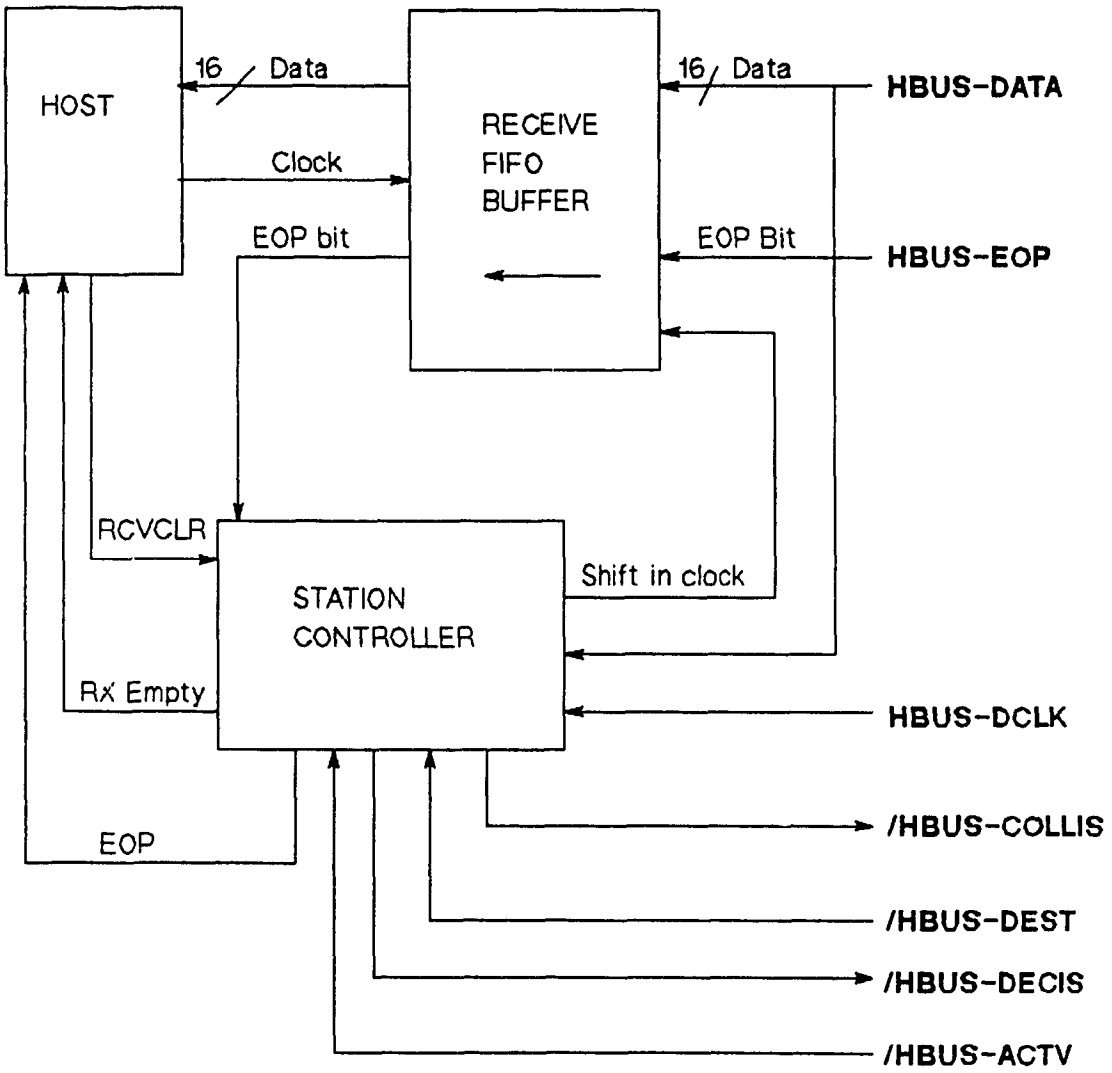


Figure 3.5 HNET Receive Block Diagram

portion of the Station Controller is less complex and is best implemented in discrete logic.

The receive FIFO is built to have an identical width and be at least as deep as the transmit FIFO. It accepts data directly from the H-Network DATA bus. It also accepts the End Of Packet bit from the HBUS. The EOP bit will be stored as the last word of the packet. This bit is needed to allow for variable length packets. The Station Controller clocks data into the FIFO at the appropriate time. The host can directly shift the received packet out of the FIFO one word at a time.

The Station Controller waits for a Transmit cycle to begin by monitoring the network activity line (ie /HBUS-ACTV). It then checks the transmit packet destination ID and the status of its receive FIFO. It responds with a decision on the network decision line (/HBUS-DECIS) or the network collision line (/HBUS-COLLIS). If the packet is to be received, then the Station Controller provides the shift in clock to the FIFO. The shift in clock is derived from the network data clock (ie HBUS-DCLK). The Station Controller also provides a FIFO Empty status bit (RX-EMPTY) and an End Of Packet bit to the host processor.

The specific host interface may take the form of interrupt or polled driven and DMA or non-DMA transfers. The host monitors the RX-EMPTY bit to determine when a packet has been received into the RX FIFO. The host then shifts out the packet from the FIFO into its own memory.

When the last word of the packet has been shifted out, the EOP bit will be set. The host monitors the EOP bit when variable length packets are allowed. The EOP bit need not be monitored if only full length packets are used.

The host follows this sequence as part of the reception of a packet:.

Step 1) Reset the Receive Station Controller by pulsing the receive clear line (ie RCVCLR). This will empty the receive FIFO, and place the station controller in the ready to receive state.

2) Wait till the Receive Empty status bit goes false.

3) Read the packet into local memory. If variable length packets are used, then monitor the EOP bit to determine the End of Packet.

4) Jump to step 1.

The H-STATION implements the following algorithm as part of the network receive protocol. Note that a Receive Clear request from the host will force the station controller to step 1.

Step 1) Set the RX EMPTY status line to TRUE Set the EOP status line to FALSE.

2) Wait for the network to go active. (ie. /HBUS-ACTV = TRUE.) While the

network is inactive set both the H-Network collision line (/HBUS-COLLIS) and the H-Network decision line (HBUS-DECIS) to FALSE.

3) When the /HBUS-DEST line is TRUE, the packet destination address will be on the H-Network data lines. Compare this destination address to the node ID number. Also check if a receive FIFO is free to accept the packet.

4) Three outcomes are possible:

   [A] If the packet is not destined to this node then the decision is to allow the packet to be sent. Set the HNET decision line (HBUS-DECIS) TRUE.

   [B] If the packet is destined to this node and a receive FIFO is free then the decision is to receive and HBUS-DECIS is set TRUE.

   [C] If the packet is destined to this node and a receive FIFO is not free then the decision is to abort transmission. A network collision is forced by setting HBUS-COLLIS TRUE.

5) Receive the packet if it is destined for this node. Mark this FIFO as used by setting the RX-EMPTY status bit FALSE.

6) Wait for the network to go inactive (ie /HBUS-ACTV = FALSE).

7) Set /HBUS-COLLIS = FALSE and HBUS-DECIS to TRUE.

8) Wait for the host to issue a receiver clear pulse on the RCVCLR line.

9) Jump to step 1.

CHAPTER 4

HOMOGENEOUS NETWORK IMPLEMENTATION

4.1 OVERVIEW

This chapter provides the Homogeneous Network implementation details. The description is at the logical function level. The logical model may be translated to various technologies for a physical implementation.

4.2 TRANSMIT MODULE

The Transmit Module accepts packets from the host. It then broadcasts the packet over the network to the Receive Module of each destination node. The Transmit Module, seen in figure 4.1, consists of the following 5 submodules :

1) Output FIFO Buffer : Accepts packet from host and holds until transmission is possible.
2) Transmit Controller: Software based controller implementing the transmit algorithm.
3) Test & Set : Part of semaphore mechanism to ensure single master of the network.
4) Tx ID Compare : Performs additional check to confirm single master of the network.
5) HBUS Clock : Generates clock used to shift data out of the Output FIFO and into the Input FIFO(s)

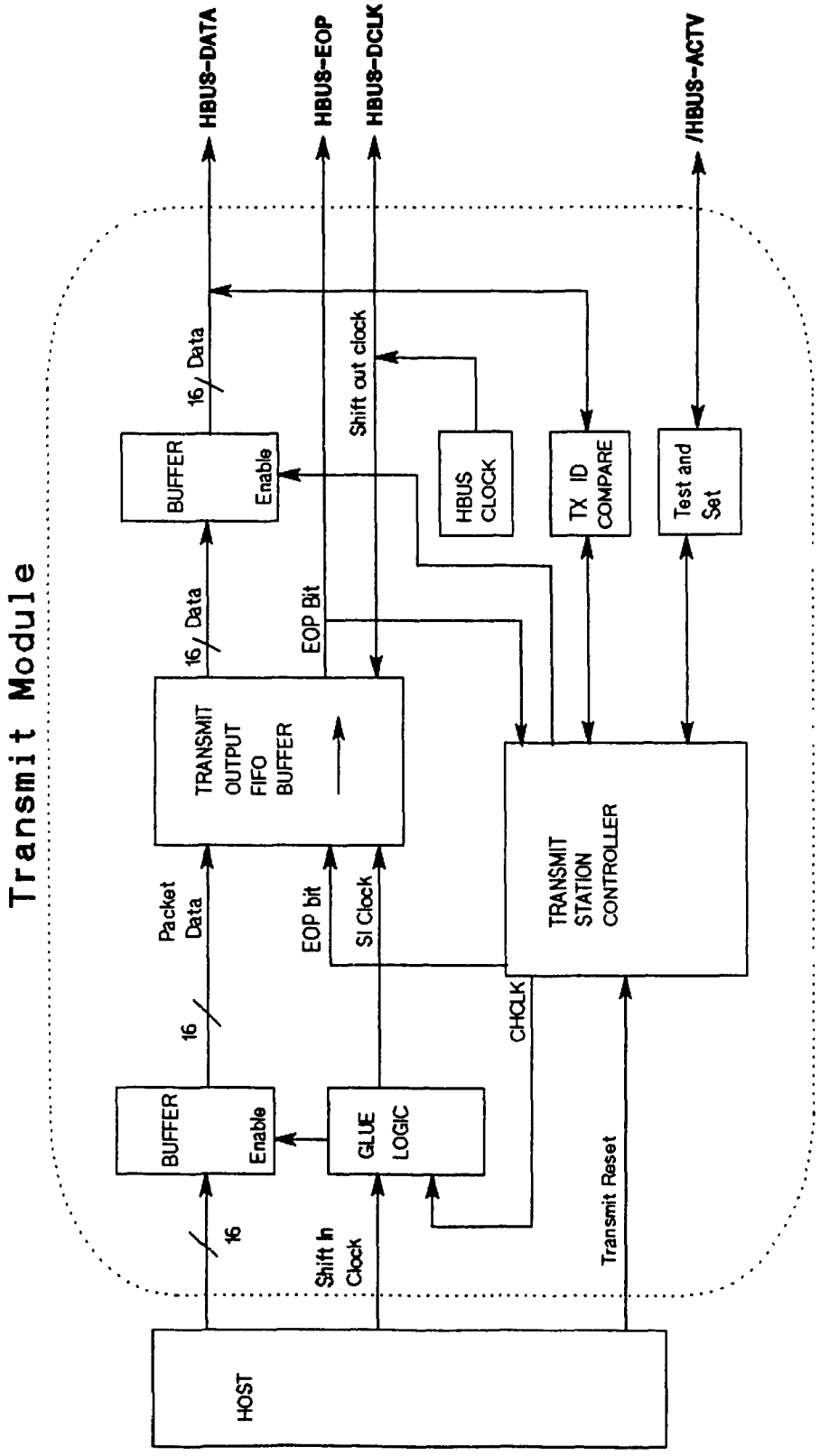


Figure 4.1 H-Network Transmit Module Block Diagram

4.2.1 OUTPUT FIFO BUFFER

The Output FIFO is shown in figure 4.2. The FIFO is 1 bit wider than the host word size. For a host word size of n bits, the FIFO is (n+1) bits wide. The extra bit is used to indicate End Of Packet. The FIFO depth is 1 word longer than the longest packet length that the host can generate. For example, a system with a 16 bit host using a maximum of 127 word per packet would use a 17 bit wide FIFO with a depth of 128 words.

The FIFO has 3 control signals. The RESET input is used to reset the FIFO, which in effect empties the FIFO of all data. The host can flush the Output FIFO simply by resetting it. The SI input is the Shift In control line. The data inputs on D0-Dn are shifted into the FIFO on the rising edge of the SI control line.

A FIFO contains a set of memory locations. The first word shifted into an empty FIFO will propagate to the last FIFO location. Subsequent words shifted into the FIFO will propagate towards the next free location. Words may be shifted in until all locations are occupied.

The word at the last FIFO location appears on the FIFO output pins. The falling edge of the Shift Out line, SO, causes the next word in the FIFO to appear on the Q0-Qn outputs. Data may be shifted out of the FIFO until it is empty.

In practice, the host processor assembles a packet of data in its local memory. When the Output FIFO buffer is

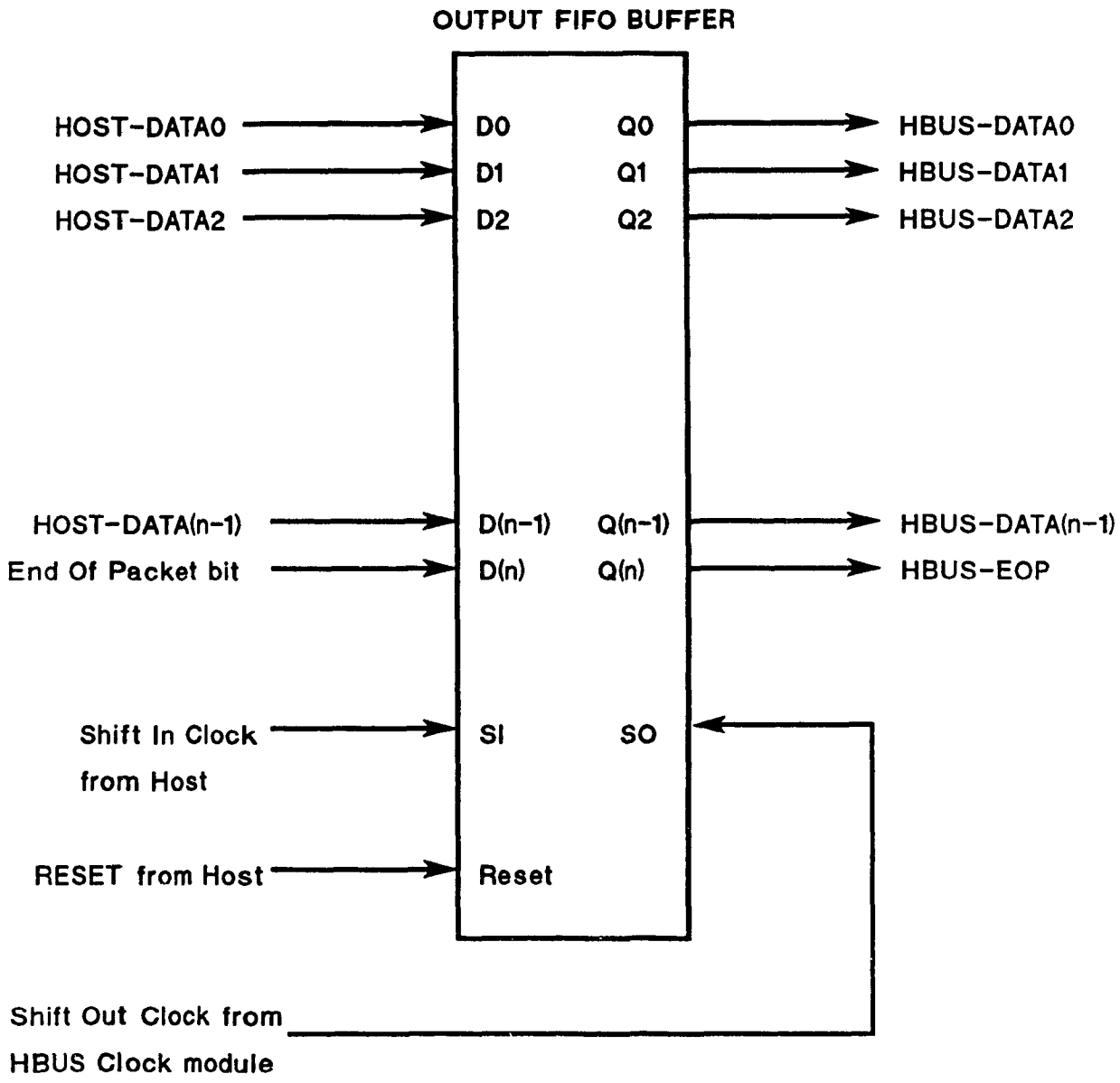
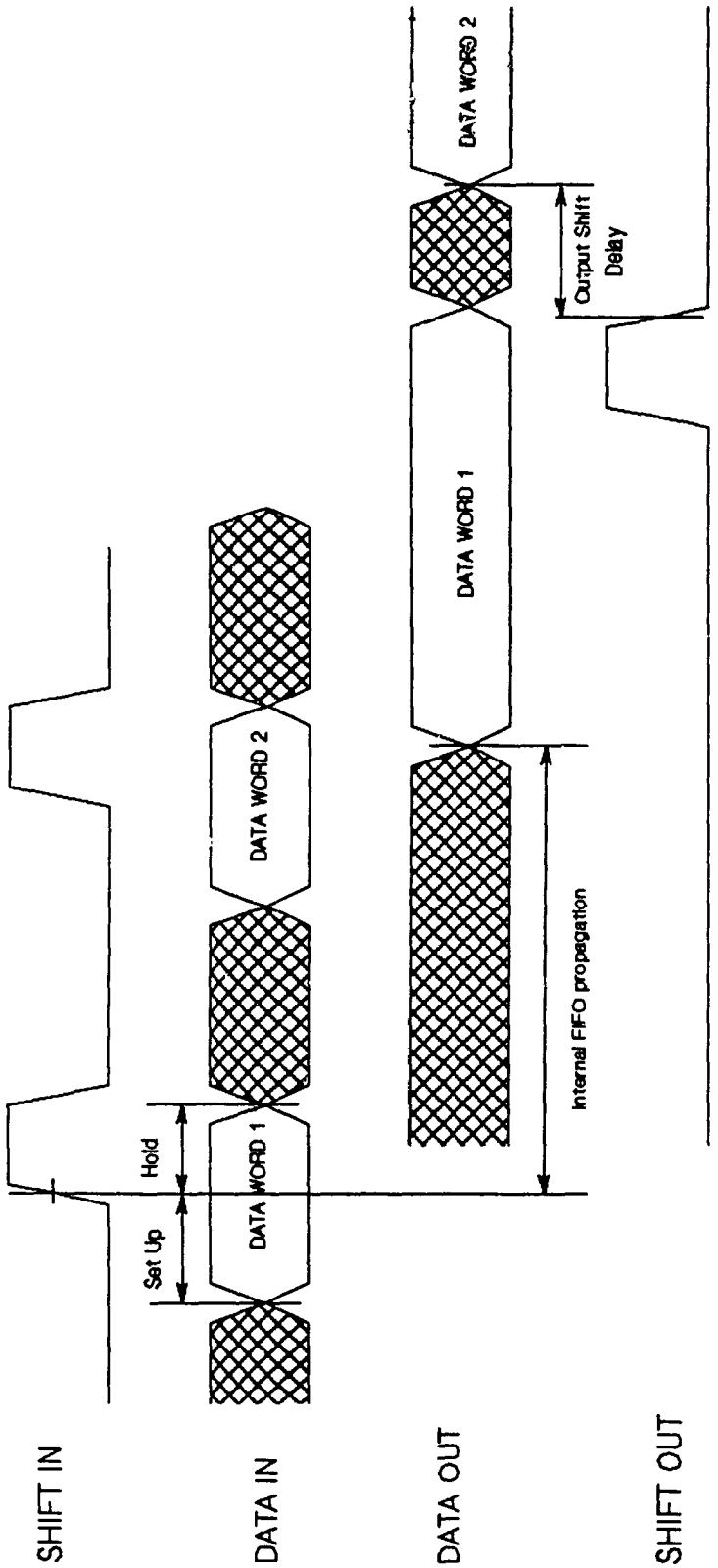


Figure 4.2 Output FIFO for n bit wide host.



Note: Timing not drawn to scale.

Figure 4.3 FIFO Timing Waveform

free, the host transfers the packet to the FIFO, one word at a time. The interface between the host and the FIFO consists of the data bus and a Shift In Clock line, SICLK. The SICLK line should only pulse for writes to the FIFO. Thus in a memory mapped environment, the SICLK line will be the host write to a specific memory location. A data word will be shifted into the Output FIFO for every SICLK pulse. Once the host has shifted the packet into the FIFO, it signals the controller, on the SEND line, to transmit the packet. The controller will shift into the FIFO the one more word with the End Of Packet bit set. The End of Packet bit is detected when the packet is being shifted out. It signals that the packet has been sent and mastership of the network must be released.

After the node becomes sole master of the network, it shifts out the packet from the Output FIFO directly to the H-Network data lines. The Shift Out clock, SO, is generated by the HBUS Clock module.

4.2.2 TRANSMIT CONTROLLER

The transmit algorithm could be implemented using discrete logic. However to allow for evolution and improvements of the algorithm, a software implementation is preferred. A solution using the Signetics 8X305 Micro Controller had been proposed by Kehayas [16]. The 8X305 is a 16 bit processor with a minimum 200 nanosecond clock cycle. A cross assembler would be used to develop the code.

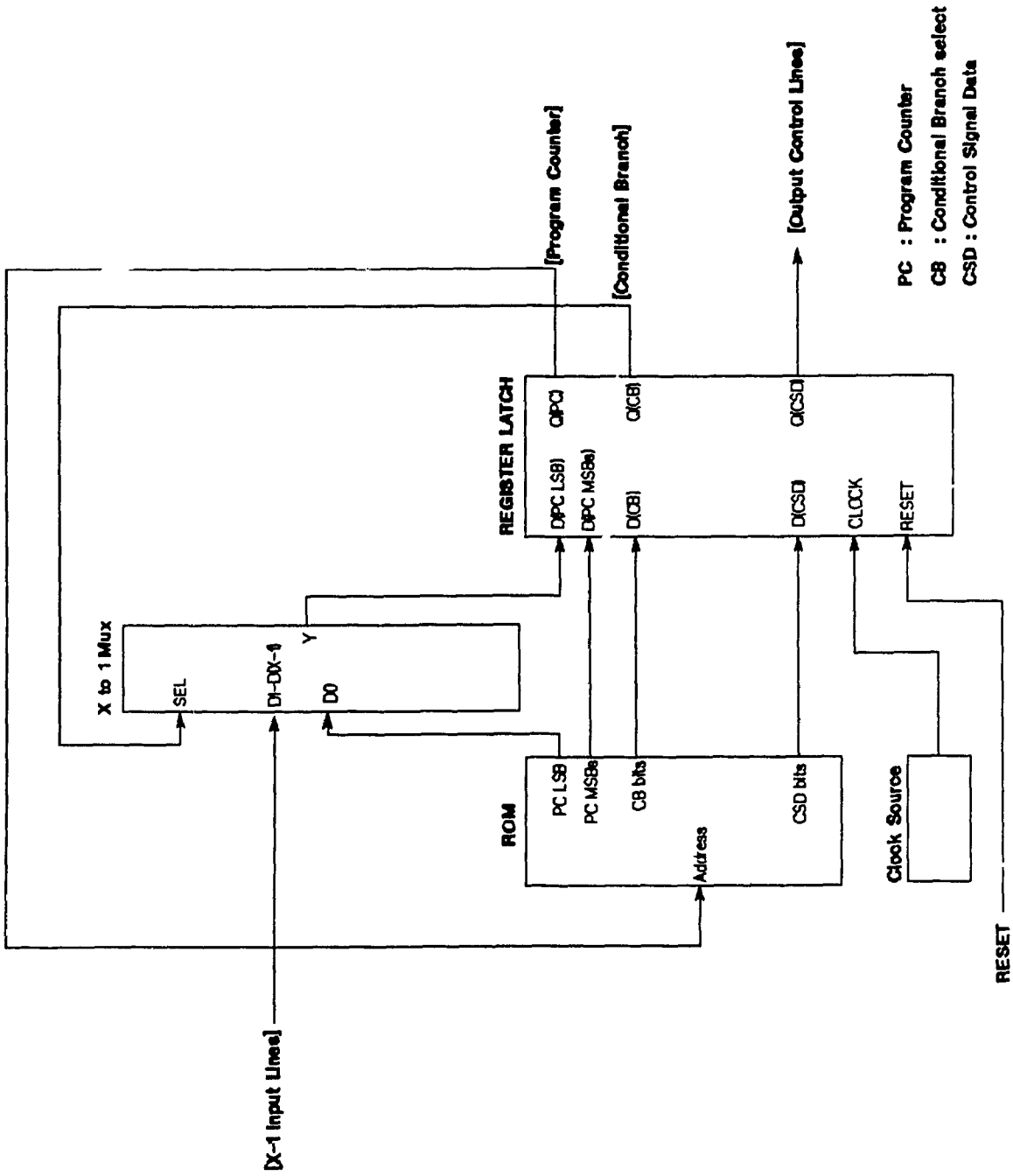


Figure 4.4 Generic Microcontroller Block Diagram

This work proposes using a custom built controller instead of the 8X305. Although not as versatile as commercially available microcontrollers, the custom microcontroller is highly suited for implementing the transmit algorithm. It can operate significantly faster than its commercial counterpart. Since it is a discrete design, it can easily be implemented using new and improved technology of the future. More importantly, it can be integrated, along with the other circuit modules, into a custom VLSI H-Network controller chip. The use of commercially available controller chips would preclude the VLSI option.

The overall architecture of the controller is shown in figure 4.4. It accepts (X-1) input lines and generates the required number of output lines. The Program Counter width depends on the length of the code. The Conditional Branch data is used to select 1 out of X inputs.

Figure 4.5 shows the specific details for the actual Transmit Controller. It implements the transmit algorithm in a state machine fashion. It accepts 7 input status lines and generates 8 output control signals. The controller is clocked by a free running clock source. A reset input allows the host to force the controller to the initial state.

The core of the controller is a fast ROM to hold the microcode. The microcode word read from the ROM is latched

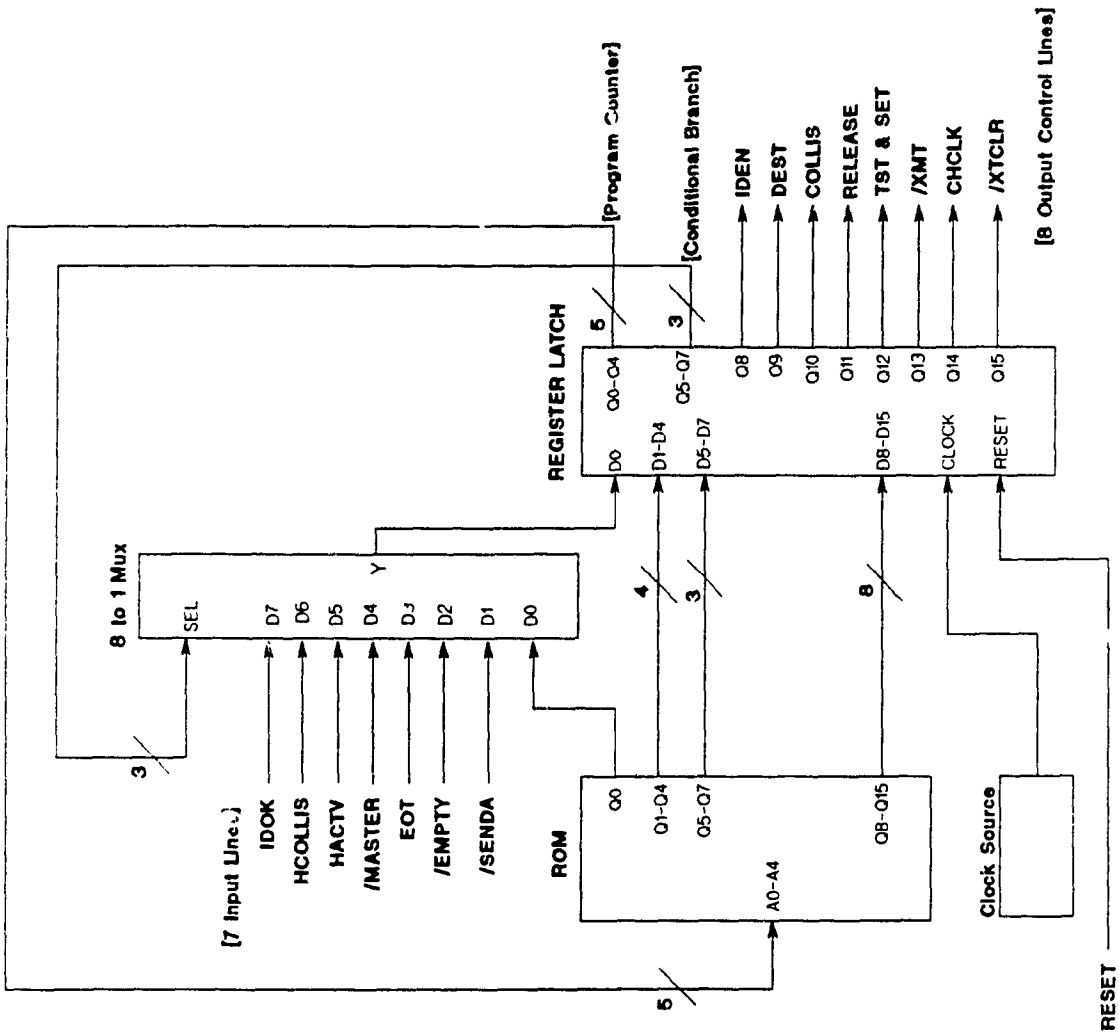


Figure 4.5 Transmit Microcontoller Block Diagram

by a register. The 5 bit Program Counter , PC, found at the output of the register. Note that the PC width can easily be increased if more than 32 locations are needed.

The register also outputs 8 control signals and a 3 bit Conditional Branch selection. The 8 control signals are used to control the network and circuits within the Transmit Module. The Conditional Branch Selection, CBS, controls an 8 input mux. It allows conditional branching depending on the state of the selected input.

The microcode instruction is 16 bit wide. Figure 4.6 shows the instruction breakdown. The microcode instruction has 3 fields as follows:

1) **PROGRAM COUNTER (PC)** 5 bit value, points to next expected address location. The PC Least Significant bit read from ROM may be replaced by one of the seven external input signals. LSB substitution is used for conditional branching and occurs only when CB of the previous instruction was not zero

2) CONDITIONAL BRANCH (CB) 3 bit value, used to choose 1 of 8 possible sources for the LSB of the Program Counter of the following instruction. This allows conditional branching on the next instruction based on 1 of 7 external inputs. CB is set to zero if the next instruction is not a conditional branch. The following lists the external signal chosen, (in the current implementation), for each value of CB.

0 : PC LSB untouched

1 : /SENDA

2 : /EMPTY

3 : EOT

4 : /MASTER

5 : HACTV

6 : HCOLLIS

7 : IDOK

3) CONTROL SIGNAL DATA (CSD) An 8 bit field which directly sets the 8 output control lines high or low. For the current implementation, each control line has a dedicated bit as follows:

(LSB) 0 : IDEN
1 : DEST
2 : COLLIS
3 : RELEASE
4 : TST&SET
5 : /XMT
6 : CHCLK
(MSB) 7 : /XTCLR

The host can reset the microcontroller directly. At reset the Program Counter, the Conditional Branch and the Control Signal Data are all set to a logic low level. On the next rising edge of the microcontroller clock, the instruction stored at location zero will be executed. Since CB was 0, the first instruction will always be an unconditional branch.

The clock cycle needs only be as long as the ROM data access time plus propagation delay of the latch. Thus a fast ROM and latch could be used to achieve a quick response. And since each instruction executes in 1 clock cycle, the controller can operate at high speed.

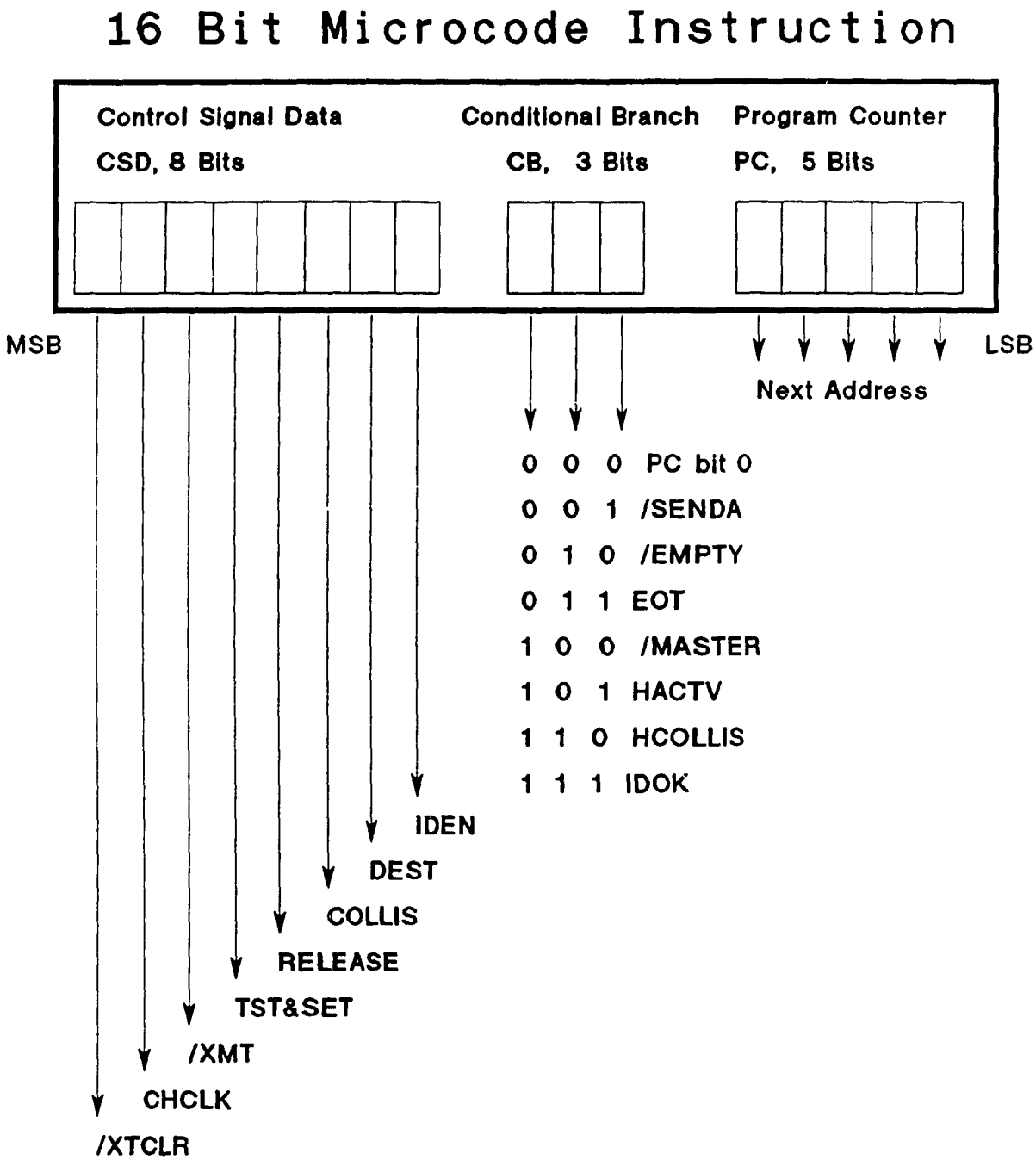


Figure 4.6 Microcode Instruction Breakdown

For example, the 63S081A, a 32 by 8 PROM from Monolithic Memories [3], has a maximum Address Access Time (T_{AA}) of 15 ns. A suitable octal latch from the TTL FAST family [14], such as the 74F273, has a clock to output data propagation delay of 12 ns. The controller clock cycle can be comfortably set at 20 ns, yielding a throughput of 50 million microinstructions per second.

During software development, it may be desirable to slow down the controller speed to facilitate debugging. Due to the static nature of the design, it is very easy to substitute a manual switch for the clock signal to allow single step execution of the algorithm.

The following simple example illustrates the controller in action. It sets the output control signal IDEN high for one controller clock cycle and low for 3 clock cycles. The first column of the listing, PC, is the actual ROM address. The second column is the Microcode Word at that ROM location. The Microcode word is broken down into its 3 fields. Only unconditional branches are used so CB is always 0. With CB=0, the 5 bit data from the PC field defines the next Program Counter value (ie. next ROM location). The CSD defines each of the 8 output control lines. IDEN is bit 0 of the CSD field, so CSD =1 will set the IDEN line high. All values are in hex.

PC	Microcode Word			Comments
00	CSD=0	CB=0	PC=01	Set IDEN=0, jump to 1
01	CSD=0	CB=0	PC=07	Jump to 7
.	.	.	.	
.	.	.	.	
07	CSD=1	CB=0	PC=08	Set IDEN=1, jump to 8
08	CSD=0	CB=0	PC=00	Set IDEN=0, jump to 0
.	.	.	.	
.	.	.	.	

At reset, the PC counter is set to 0. The instruction at location 0 sets all control lines 0, (CSD=0), and calls for a jump to location 1, (PC=01). The next instruction will be an unconditional branch since (CB=0). This instruction will execute on the rising edge of the first clock cycle after reset. Instruction at address 1 calls for a jump to location 7. The Instruction at location 7 calls for IDEN to be set high, (CSD=1) and for a jump to location 8. The instruction at location 8 sets the IDEN line low, (CSD=0) and jumps to location 0. The sequence then repeats. This produces a 25% duty cycle signal on the IDEN line.

Similarly we could pulse each of the 8 output control lines with the next example. At each step one control line is set high and all other are set low.

PC	Microcode Word			Comments
00	CSD=01	CB=0	PC=01	Set IDEN=1, jump to 1
01	CSD=02	CB=0	PC=02	Set DEST=1, jump to 2
02	CSD=04	CB=0	PC=03	Set COLLIS=1, jump to 3
03	CSD=08	CB=0	PC=04	Set RELEASE=1, jump to 4
04	CSD=10	CB=0	PC=05	Set TST&SET=1, jump to 5
05	CSD=20	CB=0	PC=06	Set /XMT=1, jump to 6
06	CSD=40	CB=0	PC=07	Set CHCLK=1, jump to 7
07	CSD=80	CB=0	PC=00	Set /XTCLR=1, jump to 0

The two examples above illustrate how each of the 8 output control lines are controlled. Each microcode instruction called for an unconditional jump to the next instruction. Since CB was always 0, all 5 bits of the PC were taken from the PC field. The next example illustrates the remaining important feature, the conditional branch. During a conditional branch, only the 4 most significant bits of the PC are taken from the microcode instruction. The LSB of the PC comes from 1 of 7 external sources, as chosen by the CB field of the previous instruction.

This IDEN line is initially set low. The program then loops until the input line SENDA goes high, then it sets the IDEN line high. The IDEN line is the LSB of the CSD field. The SENDA input line is selected by setting the Conditional Branch field, CB, to 1.

PC	Microcode Word			Comments
00	CSD=00	CB=1	PC=02	Set IDEN=0, select Conditional Branch for next instruction based on SENDA, jump to 2.
01	.	.	.	.
02	CSD=00	CB=1	PC=02	While SENDA=0, select Conditional Branch for next instruction based on SENDA, jump to 2. When SENDA =1 jump to 3.
03	CSD=01	CB=0	PC=04	Set IDEN=1, select Unconditional Branch, jump to 4.
04	.	.	.	.

As demonstrated, writing code for the microcontroller is straight forward. At each step the 8 output lines are set and the address for the next instruction is specified.

Additionally, the next instruction may be specified to be a conditional branch based on 1 of 7 external inputs. With these simple features, the complete transmit algorithm may be implemented.

4.2.3 TEST & SET

The H-Network is a single resource shared by all the nodes in the system. Only one node can be the master of the network for successful transmission. A semaphore structure is used to ensure only one node has exclusive use of the network. The semaphore is formed by the H-Network control signal /HBUS-ACTV and the TEST&SET module of each node. The /HBUS-ACTV line is a wired-OR logic line with a pull-up resistor. A logic low on /HBUS-ACTV indicates the network is active, while a logic high indicates the network is idle. The /HBUS-ACTV line is held low by the node that has become master of the network.

The TEST & SET module is shown in figure 4.7. The Station Controller attempts to become the network master by pulsing the TST&SET line. This latches the status of the /HBUS-ACTV line. If the network was idle, then the Q output of the FLIP-FLOP would be a high. This would cause the /HBUS-ACTV line to be brought low by the open-collector inverter. However if the network was active at the time of the TST&SET pulse, then the FLIP-FLOP would latch a low. While the Q output of the FLIP-FLOP is low, the open-collector inverter will have no effect on the /HBUS-ACTV line.

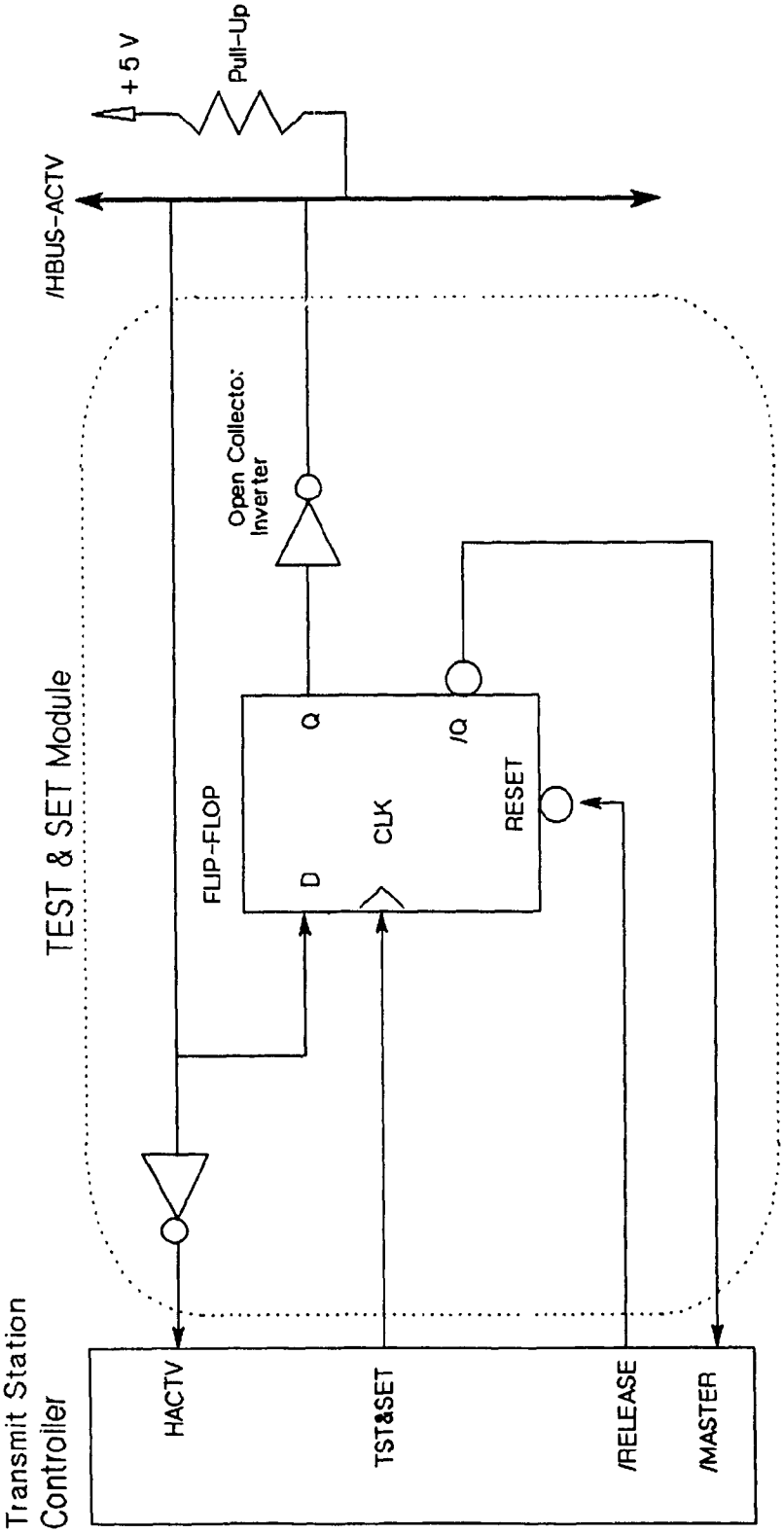


Figure 4.7 H-Network Transmit TEST & SET Module

The result of the test & set operation is determined by reading the /MASTER line. The /MASTER line will be low if the network was idle, (ie. /HBUS-ACTV = high) at the time the TST&SET line was pulsed. If the /MASTER line is found low, the controller has succeeded in becoming the network master and proceeds to the next step. If the /MASTER line is high, then the network was found busy.The controller releases mastership of the network by resetting the flip-flop through the active low /RELEASE line. The network is released at either the end of transmission or if a collision is detected.

It is quite possible for 2 or more nodes to succeed at the test & set procedure due to propagation delays. For example, consider two nodes separated some physical distance on the network. If both nodes do a test & set operation on an idle network at the exact same time, then both will see the /HBUS-ACTV line high. Both nodes will think they are masters and hold the /HBUS-ACTV line low

This problem can be handled by checking for collision of the packets during actual transmission. Every time a collision is detected, all transmitting nodes would release the network. The packets that collided must be retransmitted by the host. Collisions significantly degrade the performance of the network.

4.2.4 TRANSMIT ID COMPARE

The original network protocol is modified to avoid collisions altogether. The test and set step is followed

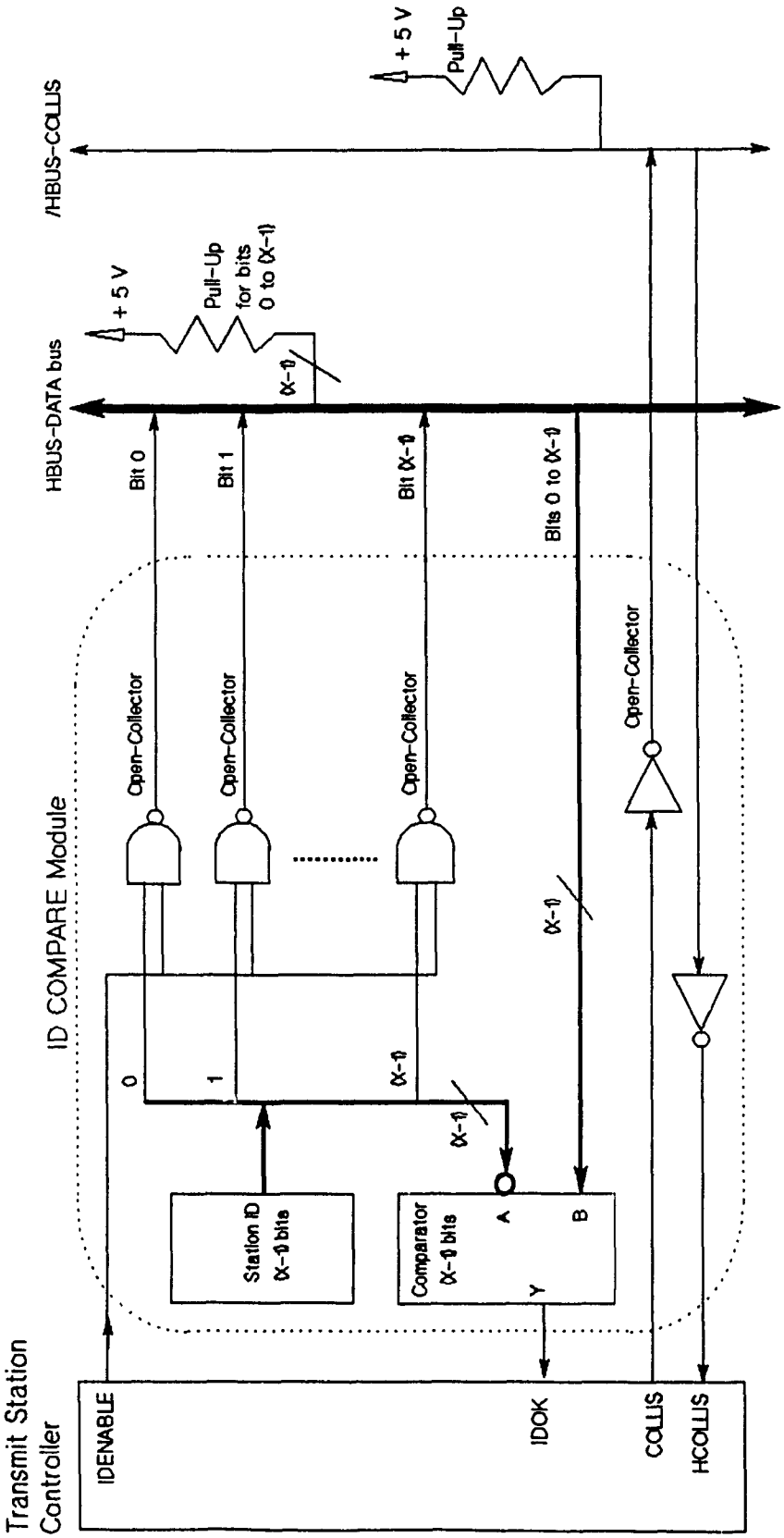


Figure 4.8 H-Network Transmit ID Compare Module

by a confirmation of only a single network master prior to any packet transmissions. If more than one node believes it is the network master, then a collision is forced and all nodes relinquish control of the network. The packets are still intact in the output fifo buffers for later transmission. No additional interaction from the host is needed.

Each node in the network is assigned a unique identification number. Following a successful Test & Set operation, the node ID is placed on the HBUS-DATA lines using Open-Collector drivers. After a suitable delay, the ID on the HBUS-DATA lines is read back and compared to the expected node ID. If it matches, then the node is the single master of the network and packet transmission may take place with no danger of a collision. However if two or more nodes are driving their node ID onto the bus, at least one node will not read back its ID. That node then asserts the /HBUS-COLLIS line to indicate a collision. All nodes will see the collision condition and release the network by clearing their Test & Set module.

Consider an example where 2 stations, with node IDs of 3 and 7, have successfully completed a Test & Set operation. Each would then place its node ID through open collector drivers onto the network HBUS-DATA lines. The wired-OR logic would produce a value of 7. The station with ID 7 would read back a 7 and see no problem. However the station with ID 3 would read back a 7 and declare a

collision by asserting the /HBUS-COLLIS line. Both stations would then release the network, avoiding a packet collision.

Figure 4.8 shows an overview of the ID Compare module with (X-1) ID bits. The number of bits, X, needed for the ID depends on the number of stations on the network. X must be less than or equal to the width of the HBUS-DATA bus. The node ID may be hard wired or set by switches. The station controller drives the ID onto the bus by setting the IDEN line high. Since NAND gate drivers are used, the inverse of the node ID is driven on to the bus. The comparator submodule compares the inverse of the Node Id to the data on the HBUS-DATA bus. The controller reads back the result of the comparison on the IDOK line. The controller declares a collision by setting COLLIS high and can read the status of the /HBUS-COLLIS line on the HCOLLIS input.

4.2.5 HBUS CLOCK

The HBUS-CLOCK module produces the HBUS-DCLK signal. This clock is used to shift data out of the transmitting node's Output FIFO onto the network and into the receiving node's Input FIFO. It feeds the Output FIFO Shift Out input of the Output FIFO module.

Figure 4.9 shows an overview of the HBUS-CLOCK and relevant control lines. It contains a gated crystal oscilator with an enable input. The oscillator is free running. With then Enable low, the output, CLK, is

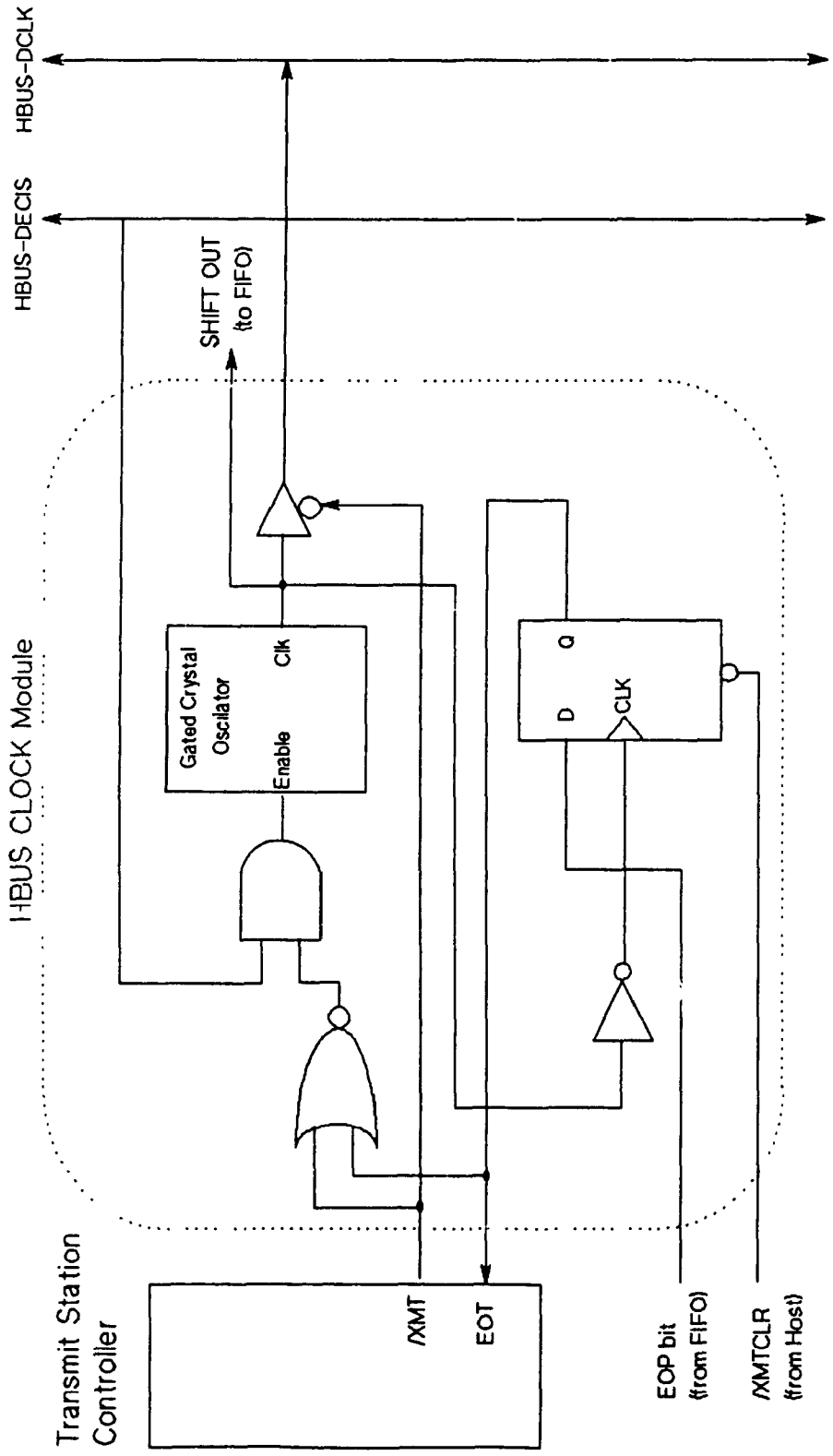


Figure 4.9 HBUS-CLOCK Module

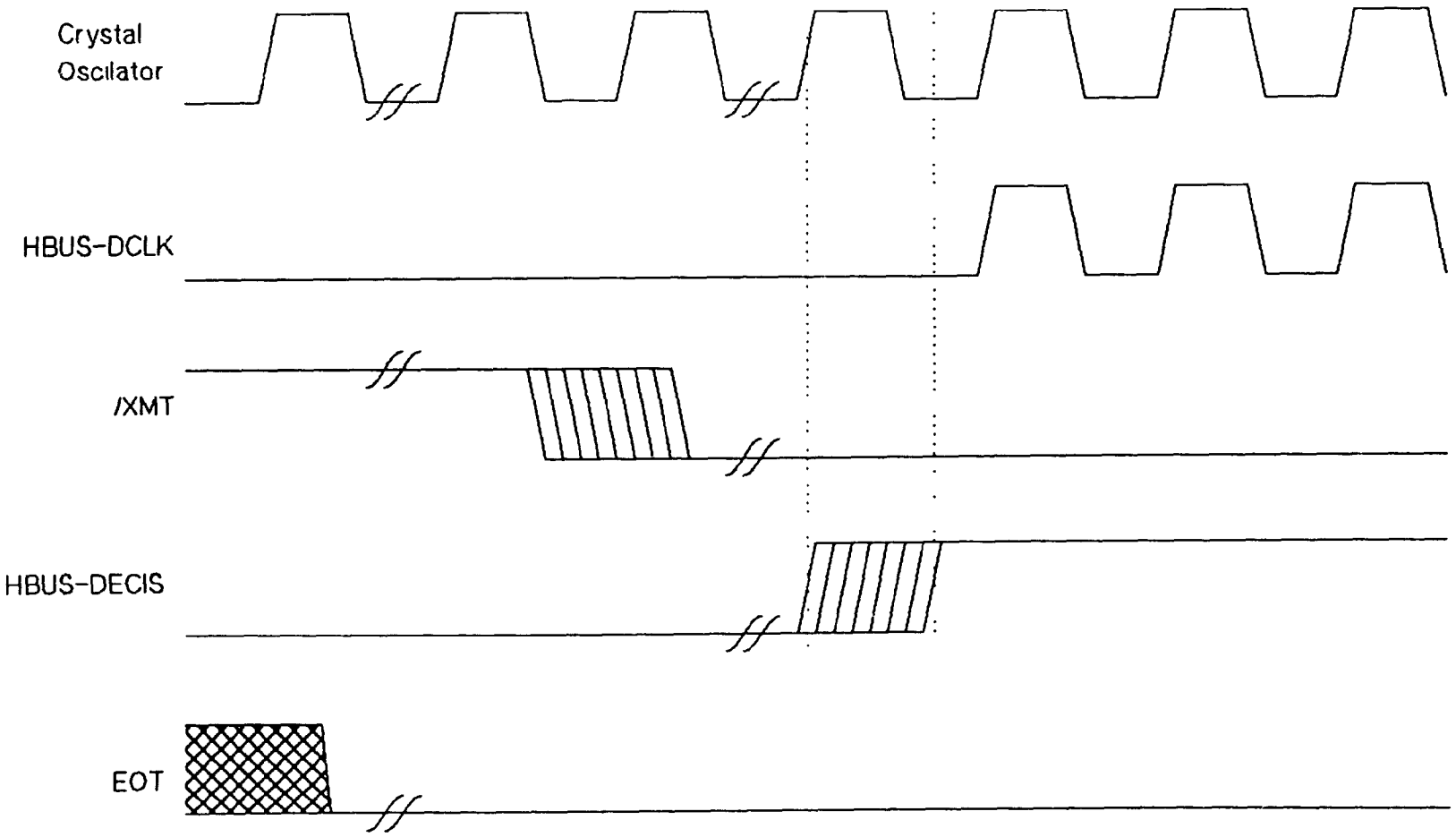
disabled (ie. low). When Enable is high, the oscillator appears on the output. However it is crucial not to change the active high pulse width of the CLK signal. Since the enabling signal is asynchronous to the free running signal, care must be taken to ensure that the enabling function allow only complete clock cycles through. Thus if the enabling signal were to arrive while the oscillator clock was high, the output would remain low for the duration of the clock cycle.

Three signals are needed to enable the clock. First, the Station Controller sets the /XMT line low, indicating that this node is the sole master and a packet transmission should take place. The /XMT line enables the tri-state HBUS-DCLK driver. Second the End Of Transmission flag, EOT, must be low. The EOT flag is set low by the host transmit clear line, /XMTCLR, prior to the host transferring the packet to the Output FIFO. The EOT flag will be set high when the last word of the packet is shifted out since the last word of the packet has the End Of Packet bit, EOP, set high. Lastly the HBUS-DECIS line must go high to start the packet transmission. HBUS-DECIS is a wired-or logic line. Prior to a transmission, all nodes hold the HBUS-DECIS line low. If a node is not a destination for the packet or if it has a free Input Buffer, it stops driving a low onto the HBUS-DECIS line. However if a node is a destination for the packet but has no free Input Buffer, it blocks transmission by holding the

HBUS-DECIS line low (and asserting HBUS-COLLIS). As soon as all station decide to allow the transmission, the oscillator is enabled and the packet is transferred. The EOP bit of the last word of the packet is latched by a flip-flop as the EOT flag. The EOT flag in turn disables the oscillator. The station controller release the network when the EOT bit is read high.

Figure 4.10 shows the timing details at the start of packet transmission. The HBUS-CLOCK crystal oscilator is free running. The EOT flag goes low after the host has issued a transmit clear pulse on the /XMTCLR line. At a later point, the station controller becomes the sole network master and sets the /XMT line low to initiate a packet transmission. After all nodes decide that the transmission may take place, the HBUS-DECIS line goes high, and in turn the oscillator enable line goes high. Note that HBUS-DCLK stays low for the duration of the current oscillator clock cycle. It only goes high at the start of the next clock cycle. Thus only complete clock cycles appear on the HBUS-DECIS line.

The timing details at the end of transmission are seen in figure 4.11. The falling edge of HBUS-DCLK during the last word of the packet latches the EOT flag high. The Station controller releases the network and sets the /XMT line high. Also, all nodes recognize an end of packet and set the HBUS-DECIS line low.



Note: Timing not drawn to scale.

Figure 4.10 HBUS-CLOCK Timing at Start of Packet Transmission

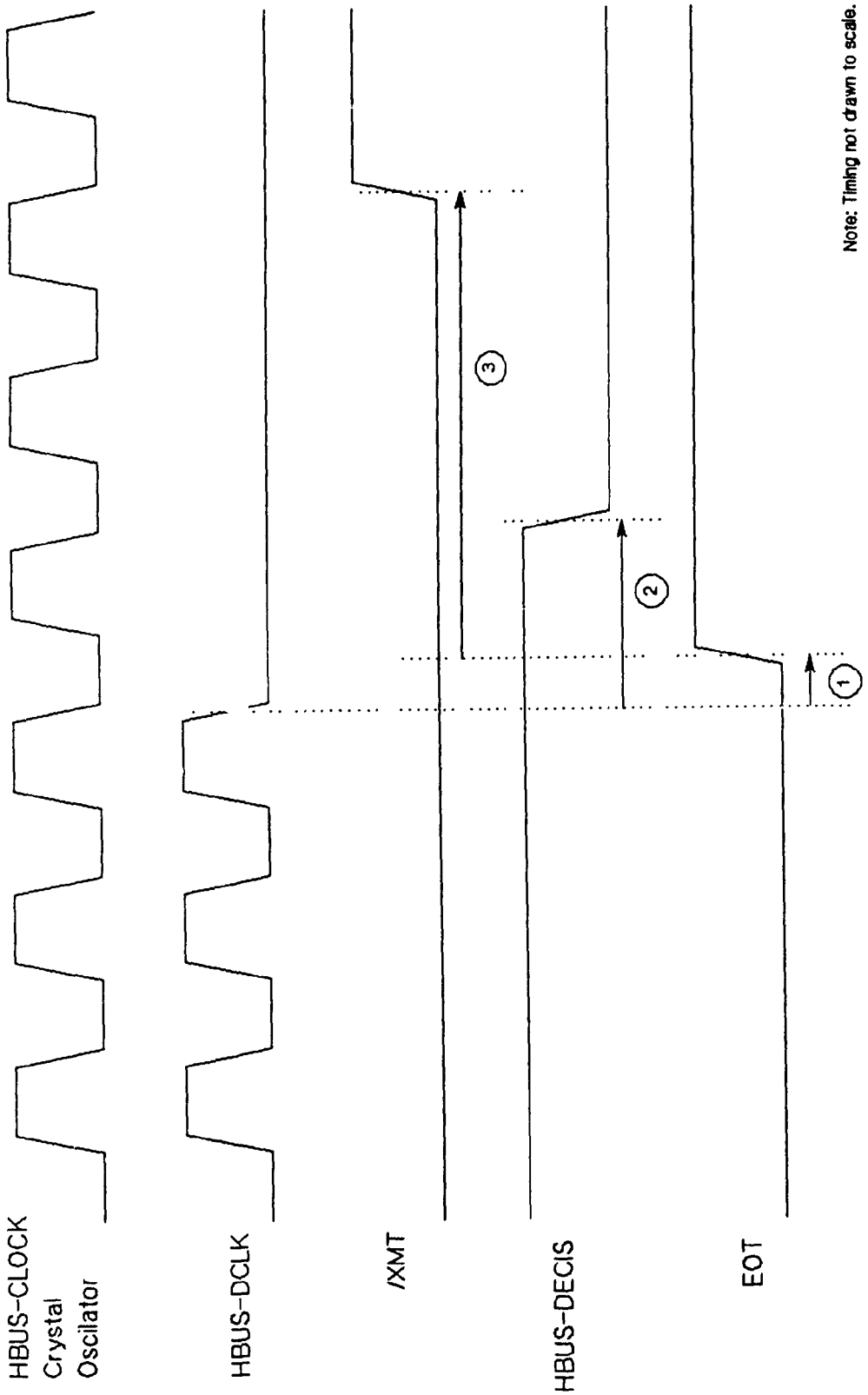


Figure 4.11 HBUS-CLOCK Timing at End of Packet Transmission

The frequency of the HBUS-CLOCK module is chosen to meet the speed of the FIFOs and of the net··k. It allows easy evolution path as faster devices become available. It also roughly doubles the throughput of the packet transfer rate compared to the scheme proposed by Kehayas [16]. That scheme used full handshaking between transmit and the receive node.

4.3 RECEIVE MODULE

The Receive Module , shown in figure 4.12, accepts data packets destined for this node from the network. The data packet is stored in the Receive Input Fifo. The host then reads the packet from the Input Fifo. The submodule, Receive Decision Logic, implements the receive aspects of the network algorithm The receive algorithm is much simpler than the transmit algorithm and is best implemented using discrete logic. There would be no advantage in using a microcontroller, in fact a speed penalty would be incurred.

The Receive Module accepts as inputs the HBUS-DATA lines, HBUS-EOP, /HBUS-ACTV, /HBUS-DEST and HBUS-DCLK. It outputs the 2 network control lines, /HBUS-COLLIS and /HBUS-DECIS. It also monitors the End Of Packet bit as the host is reading data out of the fifo to generate the Receive EOP, RCVEOP, signal. The RCVEOP is checked by the host to determine the end of packet.
The host may flush out the receive fifo and reset the Receive Decision circuit through the /RCVCLR line.

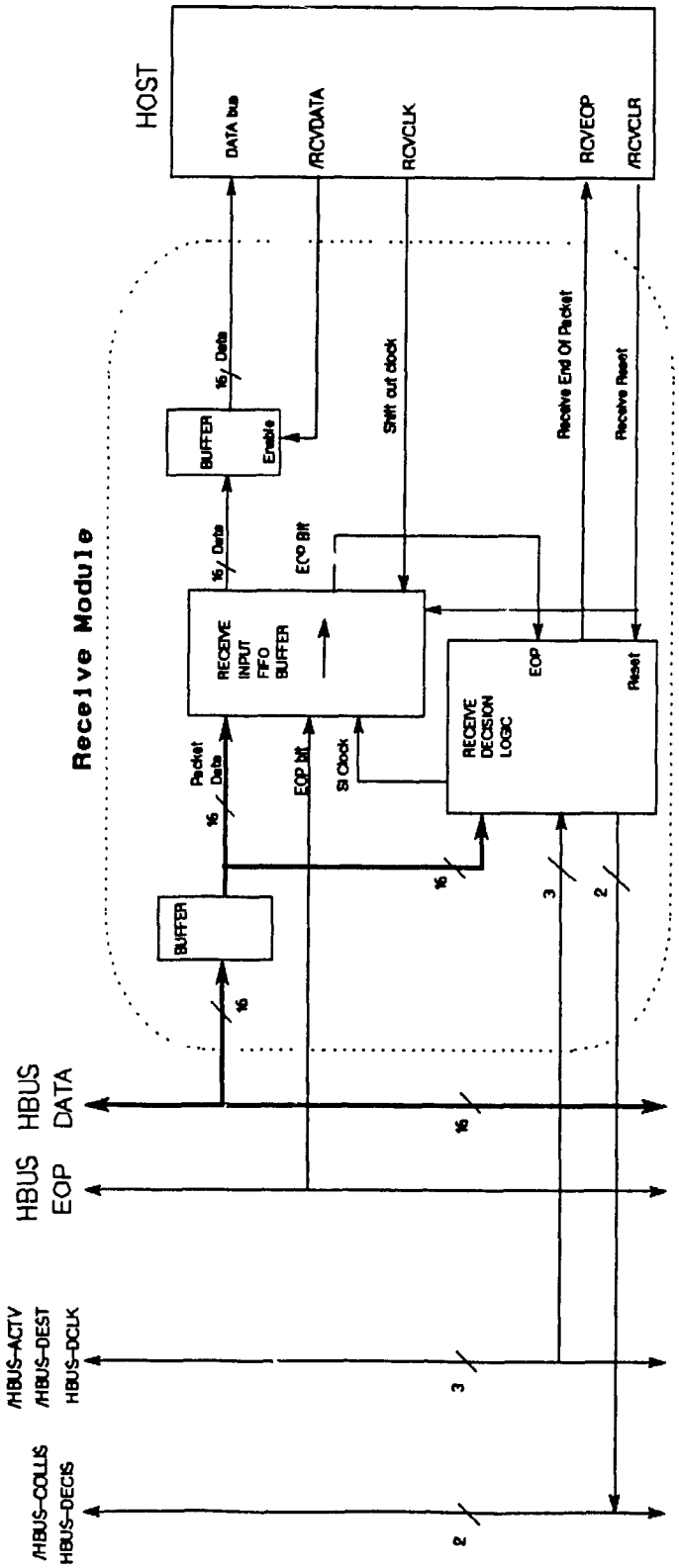


Figure 4.12 H-Network Receive Module Block Diagram

Typically the host issues a /RCVCLR pulse after each packet read in. A tristate buffer is placed between the Input Fifo and the host, allowing direct interfacing to the host's data bus. The host enables the tristate buffer through the /RCVDATA line. This allows the flexibility of enabling the buffer for the whole packet transfer or only during each packet data read cycle.

4.3.1 INPUT FIFO BUFFER

The Input Fifo Buffer is shown in figure 4.13. The fifo is identical to the Output Fifo Buffer found in the transmit section. A data word is shifted into the Input Fifo from the network data lines, (ie. HBUS-DATAx), for every HBUS-DCLK pulse. Note that the End Of Packet bit on the HBUS-EOP line is also shifted in. The host reads the packet data directly from the fifo output line. The packet is shifted out under host control. The host stops reading once it detects the End of Packet. The timing waveforms are identical to those of the Output Fifo Buffer, shown in figure 4.3.

4.3.2 RECEIVE DECISION LOGIC

Before the network master may transmit a packet, all nodes on the network must decide to allow it. The network control line HBUS-DECIS, a wired OR logic, is normally held low by all receiver nodes on the network. It goes high only after all nodes have decided that the packet may be transmitted. Alternatively, one or more nodes may abort

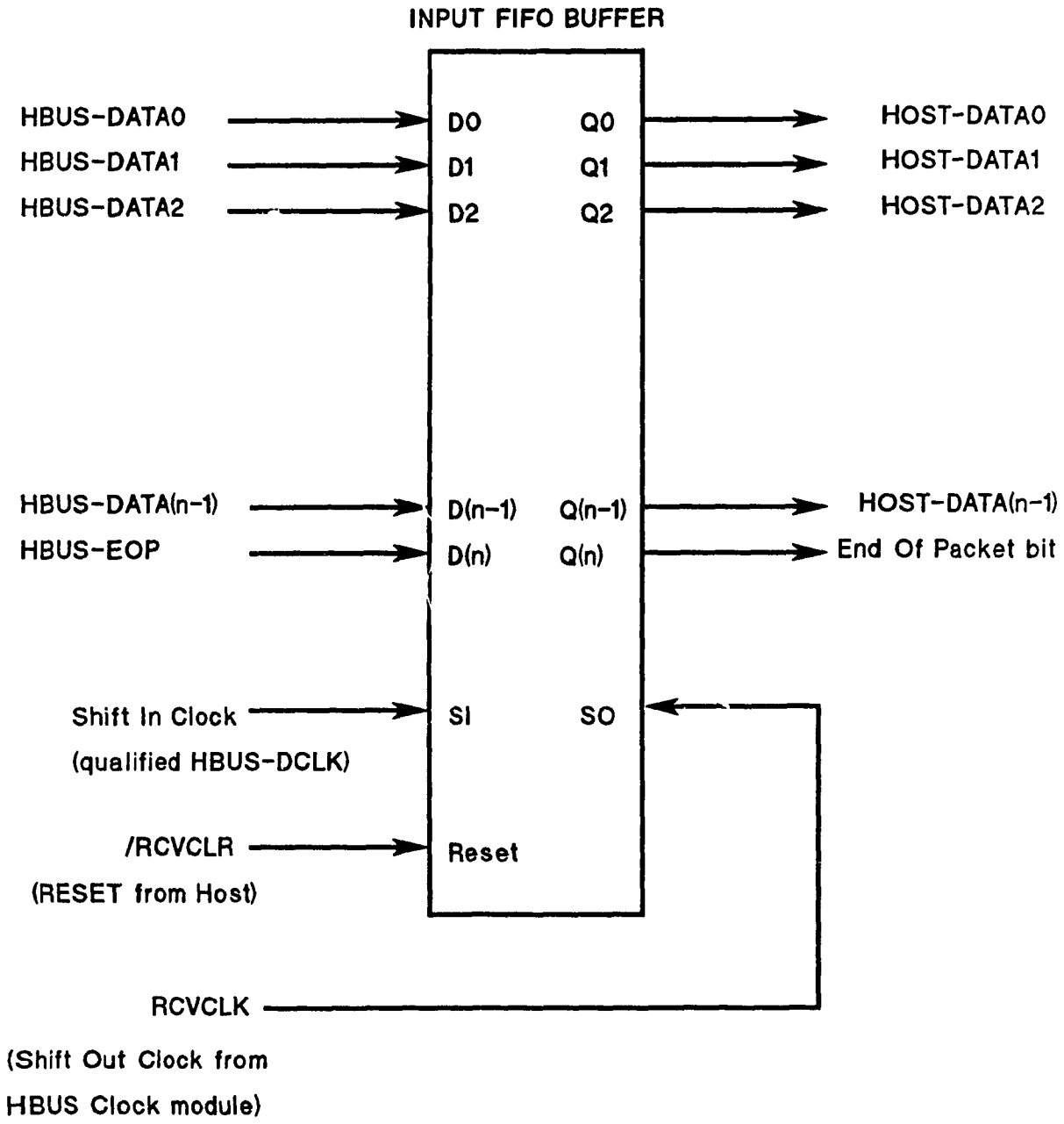


Figure 4.13 Input FIFO for n bit wide host.

the transmission attempt by asserting the /HBUS-COLLIS line low. The network master either transmits or releases the network based on the status of HBUS-DECIS and /HBUS-COLLIS. After a node becomes the sole network master, it places the first word of the packet onto the HBUS-DATA lines. The first word of each packet is the node destination address. Next the /HBUS-DEST line is pulsed to indicate to all receive nodes the destination of this packet. Each receive node checks if the packet is destined for it. If it is not, then the decision is to allow the transmission to proceed and the HBUS-DECIS is set to the open collector high state. Similarly, if the packet is destined for this node and a Input Fifo Buffer is free, then the decision is to allow transmission and HBUS-DECIS is set high. However if the node is a packet destination but no Input Fifo is free, a collision is declared and the /HBUS-COLLIS is asserted low. The transmitting node aborts the transmission when /HBUS-COLLIS is found low. The packet remains intact in the Output Fifo Buffer. The action of the Receive Decision Logic module is summarized in the following table.

Case for this node	HBUS-DECIS	/HBUS-COLLIS
Packet not destined	HIGH	HIGH
Packet destined, Fifo Free	HIGH	HIGH
Packet destined, No Fifo Free	LOW	LOW

A logical model of the Receive Decision Logic is shown in figure 4.14. It consists of a 3 bit latch, a flip-flop, a Destination ID Compare module, and glue logic. The ID Compare module compares the node ID to the packet destination ID on the HBUS-DATA lines. The output, MATCH, is high when this node is a destination. This module may be implemented in many ways. It could consist of a simple comparator with a fixed node ID. A more versatile scheme would use RAM or ROM as a lookup table, allowing a node to have more than one destination address. This could be used, for example, to broadcast a packet to a group of nodes in a single transmission. Although not shown, the host would have means to access the Destination ID Compare module.

The flip-flop output Q, is status flag indicating whether the Input Fifo is free, and is low when the fifo is empty. It is set high whenever a word is shifted into the fifo. It is cleared by the host through the /RCVCLR line, which also resets the Input Fifo.

The 3 Bit Latch outputs 3 status lines, /HBUS-COLLIS, HBUS-DECIS and Receive Enable. The input to the latch is from glue logic implementing the receive rules. On the rising edge of /HBUS-DEST, the /HBUS-COLLIS, /HBUS-DECIS and Receive Enable will be set. Receive Enable is high if the decision is to receive the packet at this node. This enables the network HBUS-DCLK to reach the Input Fifo as the Shift In clock.

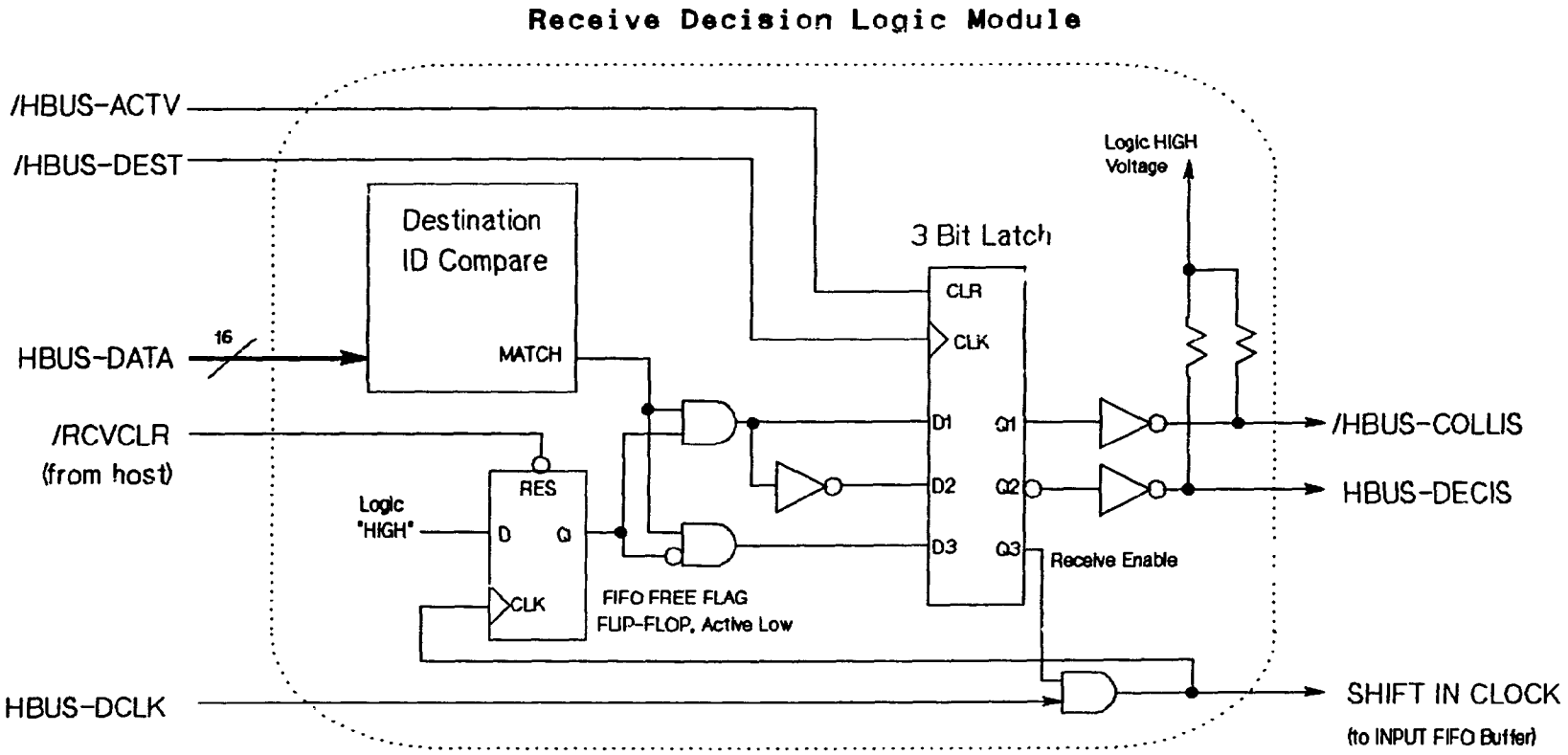


Figure 4.14 Receive Decision Logic Block Diagram

The 3 Bit Latch is cleared when the network goes inactive as indicated by /HBUS-ACTV being high. This ensures /HBUS-COLLIS is set high, and HBUS-DECIS is set low in preparation for the next packet transmission.

Figure 4.15 shows the timing diagram for the case of a node not being a destination. Since at the rising edge of HBUS-DEST, the ID Compare Match is low, the node, as well as other nodes, sets the HBUS-DECIS line high. The packet is then shifted in to the destination nodes, one word for each pulse of the HBUS-DCLK line. Since this node was not a destination, the Shift In Clock signal stays low. At the end of the packet transmission, the network master releases the network by setting /HBUS-ACTV line high. This causes all nodes to set the HBUS-DECIS line low in preparation for the next network transaction.

The timing diagram for the case when the packet is destined for this node and the Input Fifo is free, is shown in figure 4.16. In this case, the ID Compare Match is high at the rising edge of HBUS-DEST, indicating the packet is destined for this node. Since /FIFO FREE indicates that the Input Fifo is free, the HBUS-DECIS is set high. The Shift In Clock line then follows the HBUS-DCLK line, and the packet is shifted in to the Input Fifo.

Lastly figure 4.17 shows the case where the packet is destined for this node, but the Input Fifo is not free to accept it. In this case the HBUS-DECIS line is held low. The /HBUS-COLLIS is also brought low to declare a

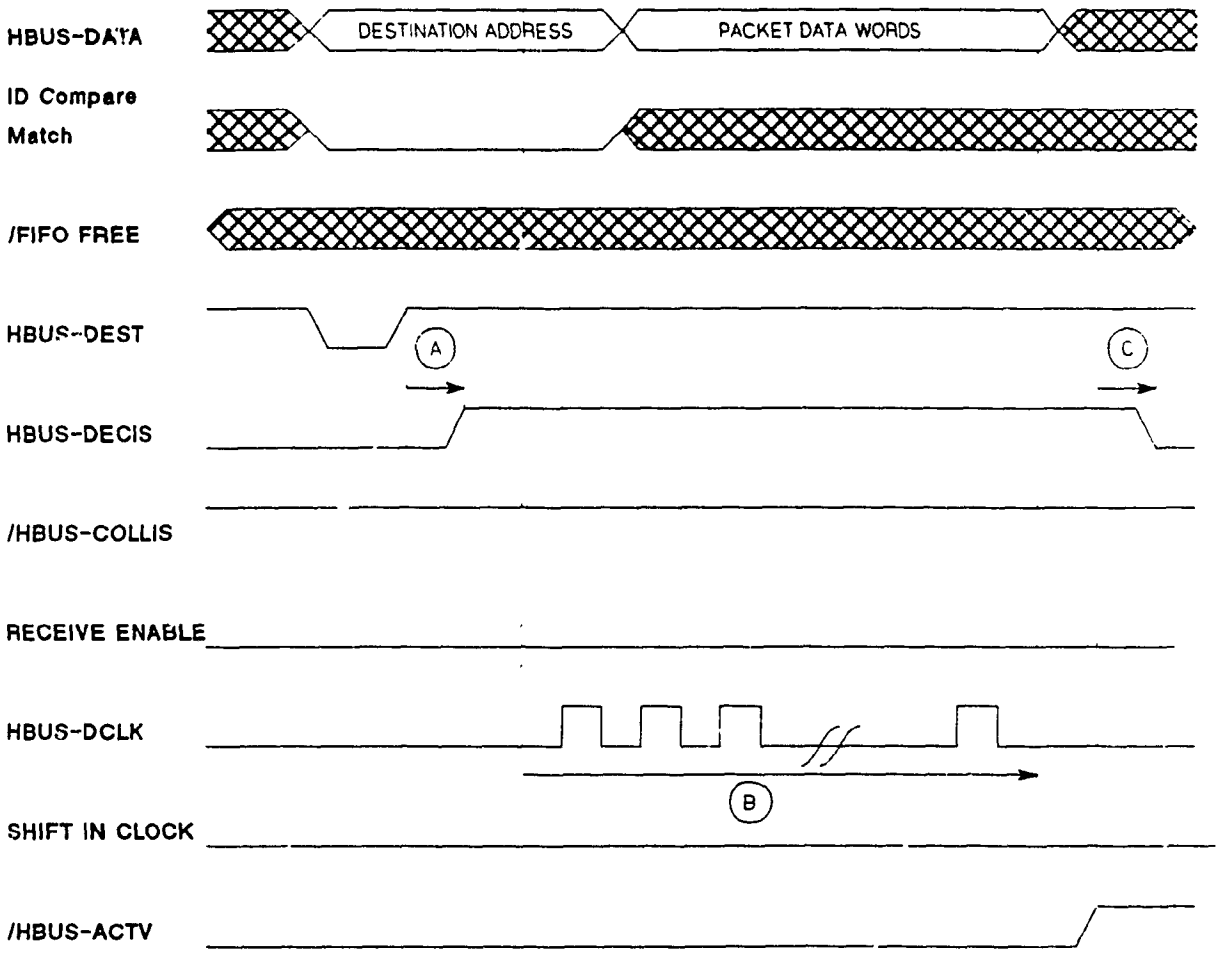
collision. The network master aborts the transmission and releases the network by setting /HBUS-ACTV high. In turn, the collision condition is removed, and /HBUS-COLLIS is set high.

Figures 4.15, 4.16, and 4.17 show the only three cases that can arise. The design is safe because the /HBUS-ACTV line is used to directly set the HBUS-DECIS and HBUS-COLLIS lines to their idle states at the end of all network transactions.

4.4 TRANSMIT CONTROLLER MICROCODE

The transmit controller hardware was described in section 4.2.2. The network algorithm was presented in section 3.4.4. This section presents the actual microcode. The algorithm is presented in figure 4.18 as a state transition diagram.

Each box represents a state. The text inside each box describes the action performed at that state. The action consists of setting the control lines high or low. Control lines not mentioned at a particular state maintain their old setting. Each Controller Clock cycle advances the algorithm to the next state pointed to by an arrow. The text beside each arrow describes the necessary condition needed before the algorithm follows that path. Arrows with no text represent unconditional branching. Above each box is the state number in hexadecimal, which is also the ROM address location.

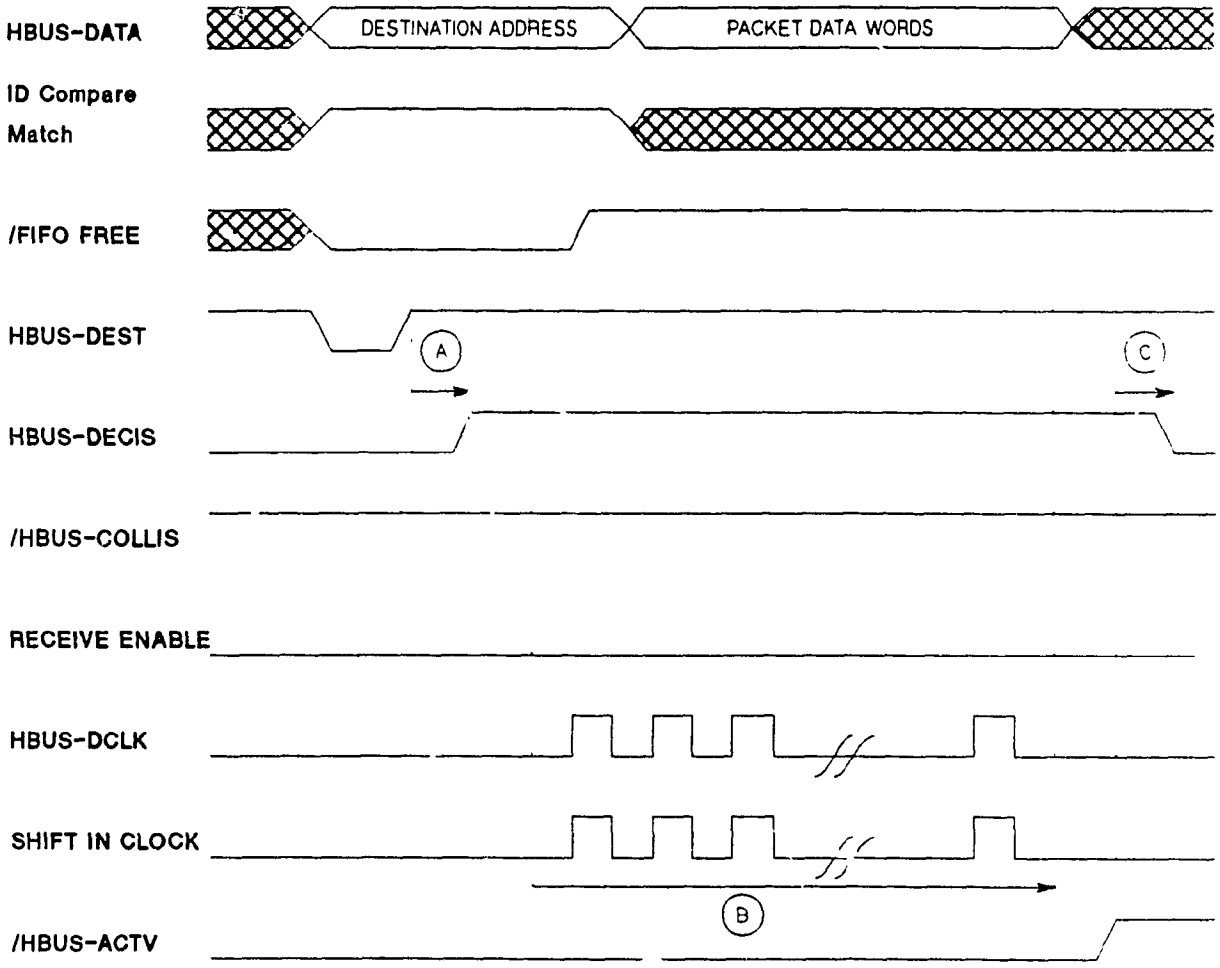


(A) Packet not destined for this node, ID COMPARE MATCH low, HBUS-DECIS set high

(B) Packet shifted in to destined nodes but not to this node, SHIFT IN CLK held low

(C) End of transmission cycle, /HBUS-ACTV goes high causing HBUS-DECIS low

Figure 4.15 Packet Not Destined to this Node

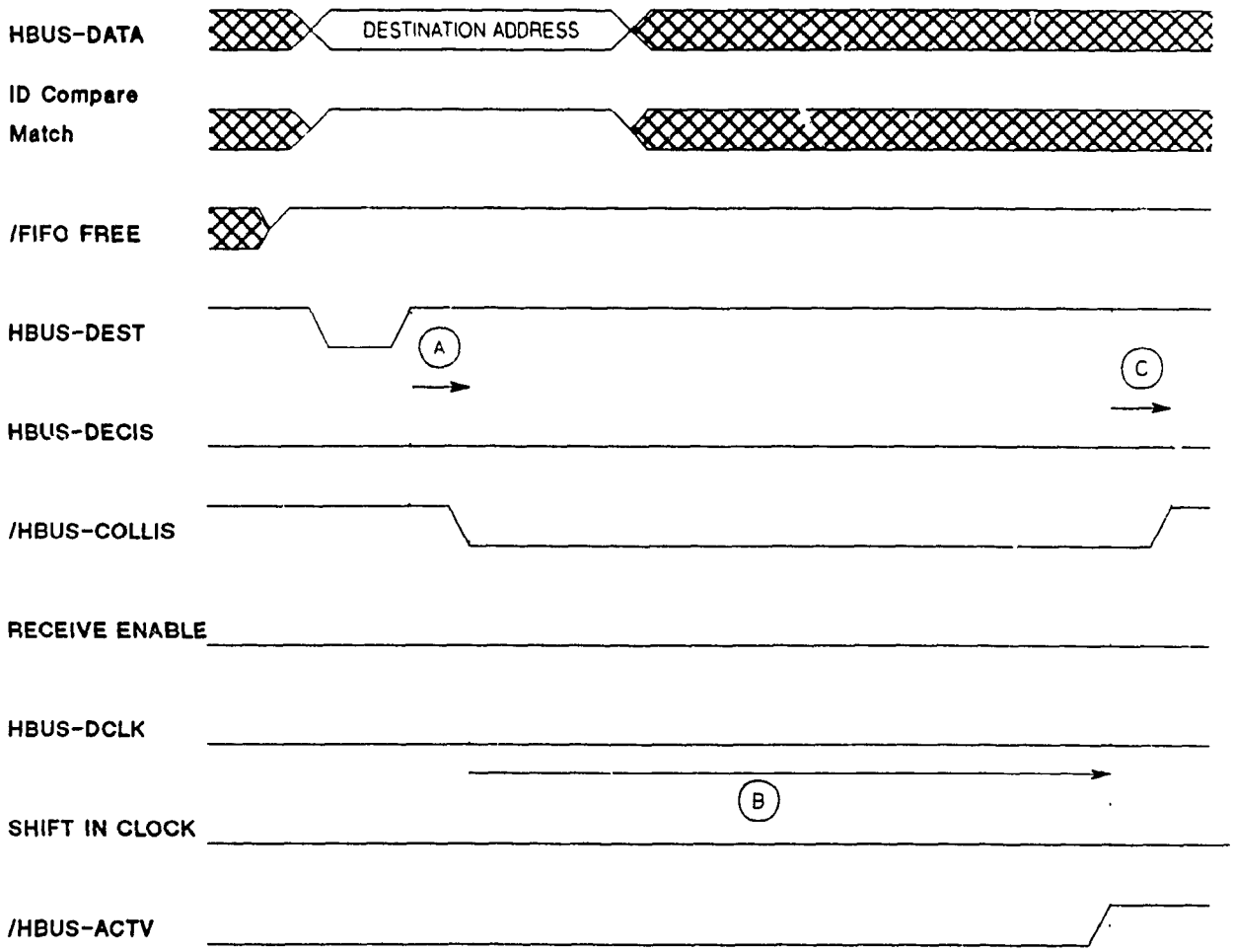


(A) Packet is destined for this node, ID COMPARE MATCH high, HBUS-DECIS set high

(B) Packet shifted in to this node, SHIFT IN CLK follows HBUS-DCLK

(C) End of transmission cycle, /HBUS-ACTV goes high causing HBUS-DECIS low

Figure 4.16 Packet Destined to this Node and FIFO is FREE



(A) Packet destined for this node but FIFO not free, /HBUS-COLLIS set low

(B) The network master recognizes HBUS-COLLIS, releases network, /HBUS-ACTV high

(C) /HBUS-ACTV goes high causing HBUS-COLLIS to go high

Figure 4.17 Packet Destined to this Node but FIFO not Free

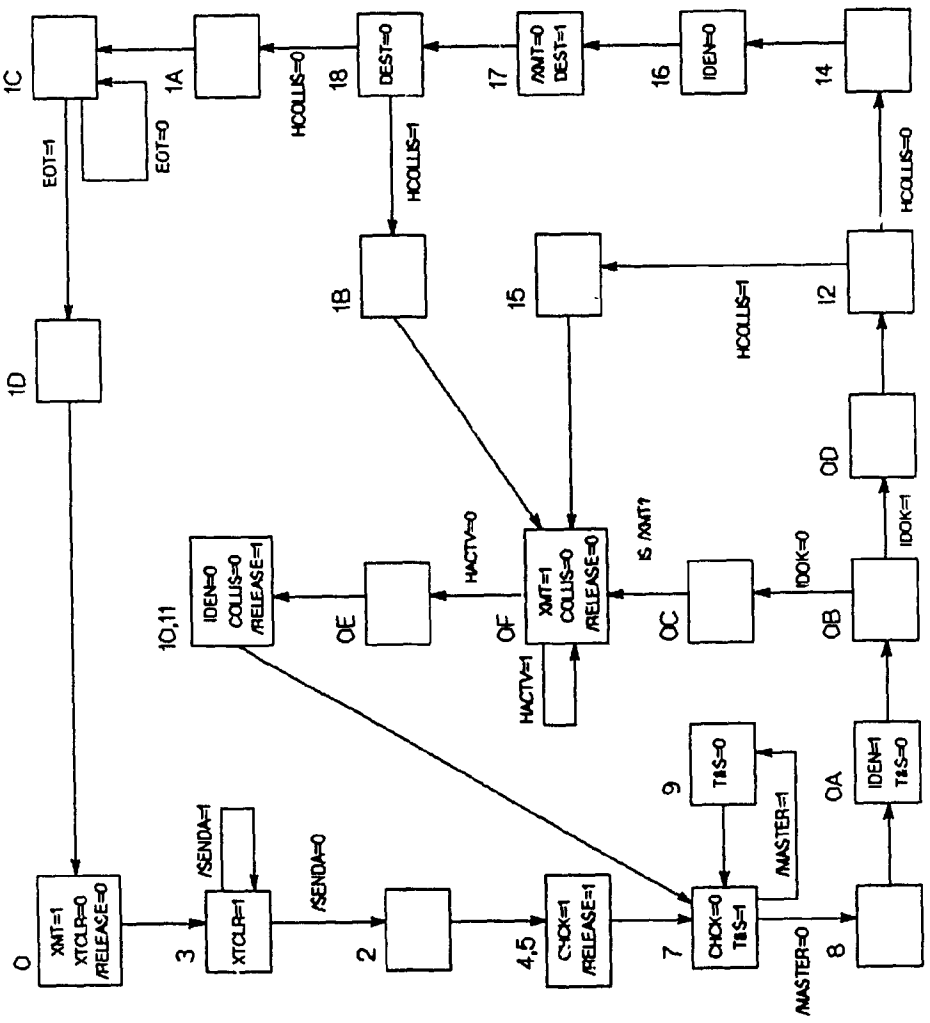


Figure 4.18 Microcode State Transition Diagram

When the host resets the controller, all eight control lines are set to low. When the host releases the transmit reset, the next controller clock will cause the algorithm to jump to state 0. Although not needed, the host should issue a reset at the end of the packet transmission to ensure everything is in a known state.

The following table lists the action at each algorithm state. A detailed knowledge of the hardware and the access protocol, as described in previous sections, is needed to understand each step.

State	Comments
0	/XTCLR=0 resets Output FIFO and control lines.
3	/XTCLR=1 ends reset, loop to this state until host request a transmit by setting /SENDA=0.
2	(Dummy State) Host has requested transmission.
4,5	CHCK=1 shifts last word, with EOP=1, into fifo.
7	T&S=1 performs TEST & SET. Jump to state 8 if network was free (ie. /MASTER=0), else goto 9.
9	Did not find network free, set T&S=0 and goto 7.
8	(Dummy State) Have become a network master.
0A	Confirm sole network master, IDEN=1.
0B	(Dummy State) If sole master, IDOK=1, goto 0D. If IDOK=0, then multiple masters, goto 0C.
0C	(Dummy State) Network has multiple masters.
0F	Release the network /RELEASE=0, Assert COLLIS=0 to cause network collision. Jump to 0F as long as other nodes are on the network (ie. HACTV=1).
0E	(Dummy State).
10,11	Collision has been recognized by all nodes, so COLLIS=0 ends collision. Jump to 7 and try again.

State	Comments
0D	Was able to read back node ID correctly.
12	Check if any other node has declared a collision. If no collision, HCOLLIS=0, goto 14 else goto 15.
15	(Dummy State) Must abort transmission attempt.
14	(Dummy State) Node is confirmed network master.
16	Remove node ID from network, IDEN=0.
17	Enable the Output Fifo data onto the network. Indicate that the destination address is on the network by setting DEST=1.
18	Set DEST=0. If HCOLLIS=1 then a collision has occurred, abort by going to state 1B else 1A
1B	(Dummy State) Packet will not be transmitted. Goto 0F to abort the attempt.
1A	(Dummy State) No collision, packet will be transmitted.
1C	Loop until whole packet has been transmitted. Goto 1D at End Of Transmission (ie. EOT=1)
1D	(Dummy State)

The actual microcode is listed in appendix A.

CHAPTER 5

CONCLUSION

5.1 CONCLUSIONS AND DISCUSSION

Multiprocessing systems are becoming an increasingly popular manner to build high performance computers. Many solutions exist for interconnecting the individual nodes. As seen in chapter 2, each communication architecture has strengths and limitations. A multi-access bus structure is an attractive architecture because it is reliable, inexpensive and very robust. It is easily the most appropriate strategy for the Homogeneous Multiprocessor System.

Distributed control of the network, as in the Ethernet network, is also desirable. However, while Ethernet uses a single coax cable, the H-Network uses separate network control and data lines to achieve significantly higher performance.

Chapter 3 defined the H-Network architecture and the individual components forming the network interface. Also the H-Network packet format, the network access algorithm, and the H-Network bus were defined. Lastly, the interaction between the host and the H-Station was explored. The design presented does not define the data path width, which is dependant on the host chosen. No restrictions are placed on the technology that may be used for an actual implementation.

In chapter 4, the logical design of the H-Network was detailed. Key to this design is a custom microcontroller to implement the transmit portion of the network protocol. The algorithm intended to run on the microcontroller was presented as a state transition diagram. Appendix A lists the actual microcode that would be stored in the ROM.

The design presented in this work has significantly improved on the original H-Network proposals. One important feature is the complete elimination of packet data collisions during transmission. This is assured by confirmation of a single network master prior to any data transmission. Also, by eliminating the handshaking between the Input and Output FIFO, the data transfer rate has been effectively doubled.

The intent of this thesis has been to design a high speed network for use in the Homogeneous Multiprocessor. This goal has been achieved. Further, the logical design in chapter 4 was translated to a physical implementation based on readily available technology, as seen in the schematic in appendix B. Major portions of this schematic were wired up to allow testing of the concept on the lab bench. Specifically, packets were transferred successfully, and the microcontroller was single stepped to validate the microcode. The next step will be to wire up a full system running at full speed.

REFERENCES

[1] Ali, M.M. and J.F. Hayes, "An Optical Fiber Based Local Backbone Network", Concordia University.

[2] Andrews, D.W., and G. Shultz, "A Token Ring Architecture for Local Area Networks", IBM Corporation, Communication Product Division, North Carolina, USA.

[3] "Bipolar LSI Data Book", Monolithic Memories 1986.

[4] "Component Data Catalog", Intel 1983.

[5] Abramson, N., "The Aloha System", AFIPS Conf. Proc., Vol. 37, pp. 281-285, FJCC, 1970.

[6] Clark, D.D. el al., "An Introduction to Local Area Networks", Proceedings of the IEEE, vol. 66, no. 11, pp 1497-1517, November 1978.

[7] Cotton, I., "Technologies for Local Area Computer Networks", Computer Networks, vol. 4, pp. 197-208, November 1980.

[8] Dimopoulos, N., "The Homogeneous Multiprocessor Architecture - Structure and Performance Analysis", Proc. of the 1983 Int'l Conf. on Parallel Processing, pp. 520-523, August, 1983.

[9] Dimopoulos, N. and D. Kehayas, "The H-Network A High Speed Distributed Packet Switching Local Computer Network", Proc. on MELECON'83 Mediterranean Electrotechnical Conf. Athens, Greece, p. A01,02, May 1983.

[10] Dimopoulos, N., "On the Structure of the Homogeneous Multiprocessor", IEEE Trans. on Computers, 1985.

[11] Dimopoulos, N., K.L. Li, and C.W. Wong, "Simulation and Performance of the Homogeneous Multiprocessor", Summer Computer Simulation Conf., Boston, M.A., July 1984.

[12] Dimopoulos, N. and C.W. Wong, "Collision-free Protocol for Local Area Networks", Computer Communications, Vol. 11, No. 4, pp. 208-214, August 1988.

[13] Ercegovac, M.D. and Lang, T., Digital Systems and Hardware/Firmware Algorithms, John Wiley & Sons Inc, 1985.

[14] "FAST Data Book", Signetics 1986.

[15] Franta, W.R., Chlamtac, I., "Local Networks", D.C. Heath and Company, Massachusetts, 1981.

[16] Kehayas, D., "The H-Network - A Packet Switching Local Area Network", Major Technical Report, Master of Electrical Engineering, Concordia University, 1983.

[17] Kleinrock, L. and F.A. Tobagi, "Packet Switching in Radio Channels: Part 1", IEEE Trans. on Communications, COM.23, Dec. 1975.

[18] Lewin, Morton H., "Logic Design and Computer Organization, Addison-Wesley Publishing Company, 1983.

[19] Metcalfe, R.M. el al., "Ethernet: Distributed Packet Switching for Local Computer Networks", Communications of the ACM, vol. 19, pp. 395-404, July 1976.

[20] Phung, V.P.T., "Throughput Analysis of CSMA/CF Systems", Master of Electrical Engineering Thesis, Concordia University, 1987.

[21] Phung, V.P.T., Dimopoulos, N., "Throughput Analysis of A Collision Free Protocol for Local Area Network", 13th Conference On Local Computer Networks, Minneapolis, MN, October 1988.

[22] Shoch, J., Dulal, Y., Redell, D., "Evolution of the Ethernet Local Network", Computer, 15, pp. 10-24 August 1982.

[23] "TTL Logic Data Manual", Signetics 1982.

[24] Tanenbaum, A.S., Computer Networks, Prentice-Hall Inc., 1981

[25] Wong, C.W., "A Collision Free Protocol for LANs Utilizing Concurrency for Channel Contention and Transmission", Master of Electrical Engineering Thesis, Concordia University, 1985.

APPENDIX A

TRANSMIT ALGORITHM MICROCODE LISTING

The following table lists the microcode instructions needed to implement the transmit algorithm described in chapters 3 and 4. It consists of a bit for each of the eight control lines XTCLR, CHCLK, /XMT, T&S, /RELEASE, COLLIS, DEST, and IDEN. Three bits are used for the select field and 5 for the address field. The ROM DATA column holds the microcode instruction in hex for the ROM address indicated in the ROM ADDR column.

An X entry indicates a "don't care" condition and are set to zero in the ROM.

ROM ADDR	ROM DATA	XTCLR	CHCLK	/XMT	T&S	/RELEASE	COLLIS	DEST	IDEN	SELECT2	SELECT1	SELECT0	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0
00	A023	1	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1
01	A023	1	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1
02	E804	1	1	1	0	1	0	0	0	0	0	0	0	0	1	0	X
03	A022	1	0	1	0	0	0	0	0	0	0	1	0	0	0	1	X
04	B887	1	0	1	1	1	0	0	0	1	0	0	0	0	1	1	1
05	B887	1	0	1	1	1	0	0	0	1	0	0	0	0	1	1	1
06	0000	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
07	A808	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	X
08	A90A	1	0	1	0	1	0	0	1	0	0	0	0	1	0	1	0
09	B887	1	0	1	1	1	0	0	0	1	0	0	0	0	1	1	1
0A	A9EB	1	0	1	0	1	0	0	1	1	1	1	0	1	0	1	1
0B	A90C	1	0	1	0	1	0	0	1	0	0	0	0	1	1	0	X
0C	A5AF	1	0	1	0	0	1	0	1	1	0	1	0	1	1	1	1
0D	A9D2	1	0	1	0	1	0	0	1	1	1	0	1	0	0	1	0
0E	A810	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	X
0F	A5AE	1	0	1	0	0	1	0	1	1	0	1	0	1	1	1	X
10	B887	1	0	1	1	1	0	0	0	1	0	0	0	0	1	1	1
11	B887	1	0	1	1	1	0	0	0	1	0	0	0	0	1	1	1
12	A914	1	0	1	0	1	0	0	1	0	0	0	1	0	1	0	X
13	0000	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
14	A816	1	0	1	0	1	0	0	0	0	0	0	1	0	1	1	0
15	A5AF	1	0	1	0	0	1	0	1	1	0	1	0	1	1	1	1
16	8A17	1	0	0	0	1	0	1	0	0	0	0	1	0	1	1	1
17	88D8	1	0	0	0	1	0	0	0	1	1	0	1	1	0	0	0
18	881A	1	0	0	0	1	0	0	0	0	0	0	1	1	0	1	X
19	0000	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
1A	887C	1	0	0	0	1	0	0	0	0	1	1	1	1	1	0	0
1B	A5AF	1	0	1	0	0	1	0	1	1	0	1	0	1	1	1	1
1C	887C	1	0	0	0	1	0	0	0	0	1	1	1	1	1	0	X
1D	2000	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	X
1E	0C00	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
1F	0000	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Notes: 1) First column is the ROM address.
2) Second column is the ROM data.
3) Remaining 16 columns contain the microinstruction broken down, bit by bit. The LSB is at the extreme right (ie. ADDR0).
4) The / symbol denotes active low.
5) X entries in the table denote "don't care" and are set to 0.

Microcode Listing for H-Network Transmit Algorithm

APPENDIX B

SCHEMATIC OF H-NETWORK NODE PROTOTYPE

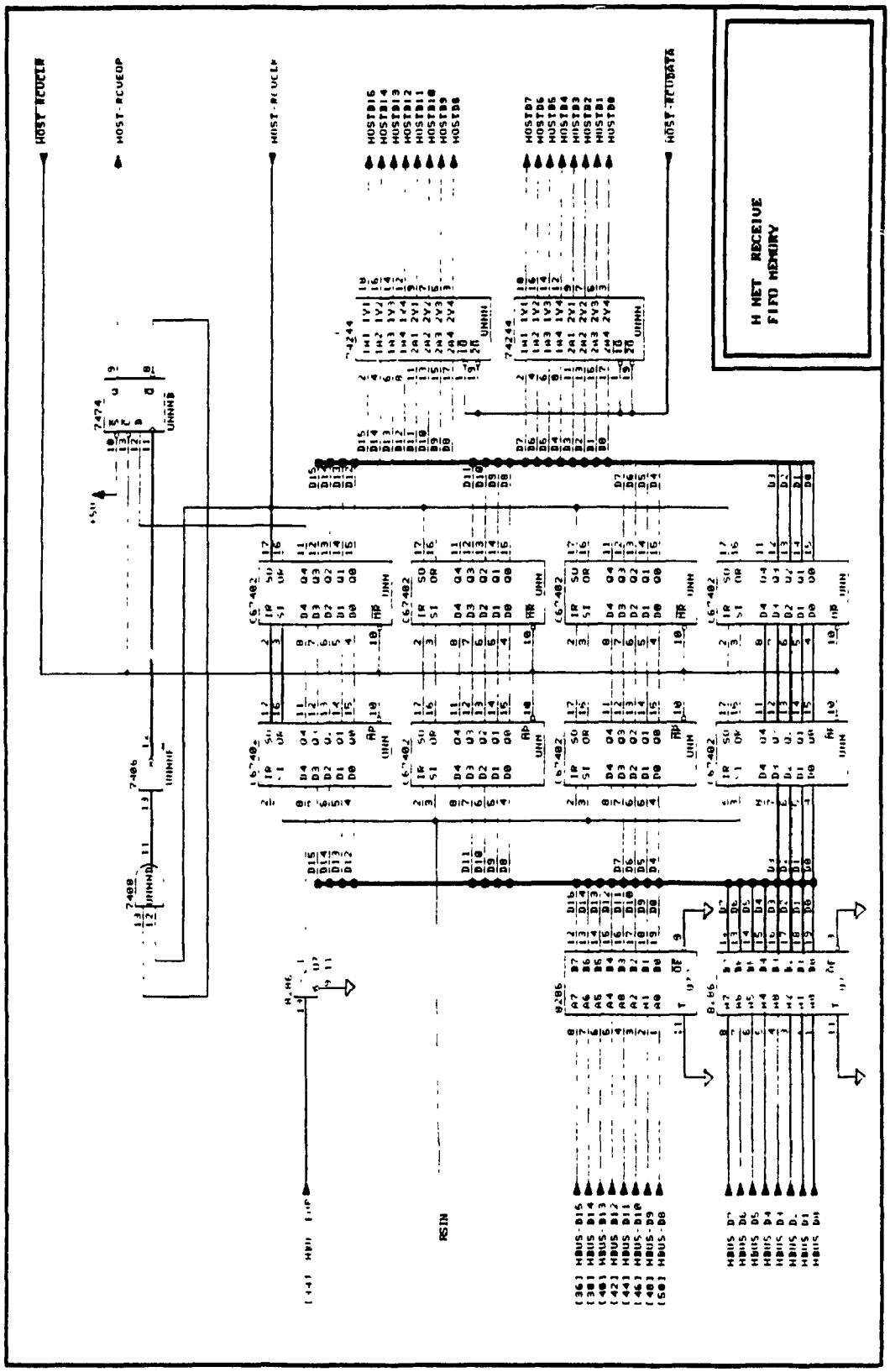
The following schematic is an implementation based on the logical model described in chapter 4. It uses readily available TTL devices. The Input and Output FIFOs are built using 64 by 4 and 64 by 5 FIFOs devices to produce a 128 by 17 FIFO buffer. The ROMs of the microcontroller contains the microcode listed in appendix A.

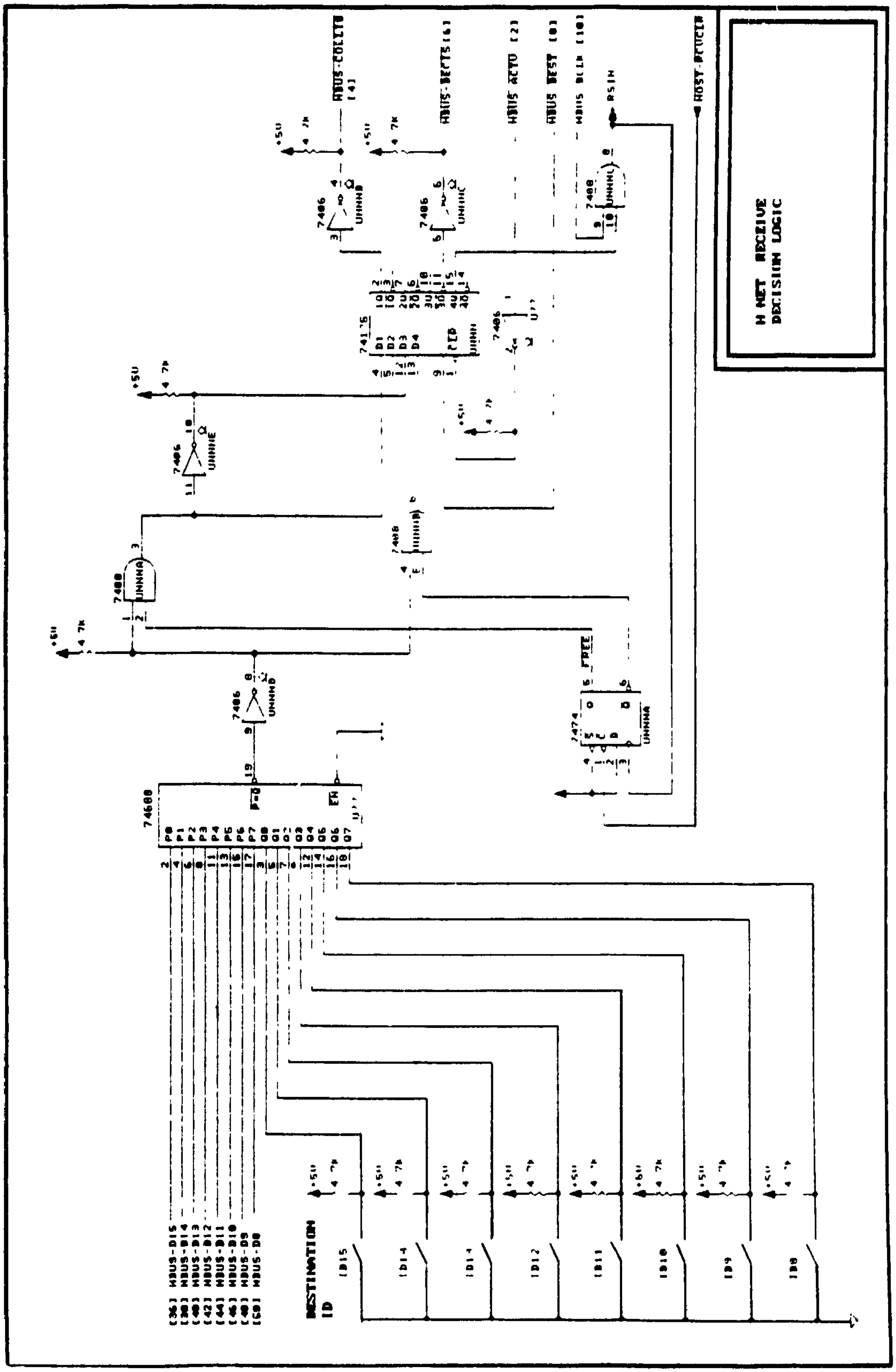
Key sections of the schematic were wired up. The main motive was to verify the microcode and the validity of the design in the lab. It was not intended as the optimal or fastest version of the H-Network.

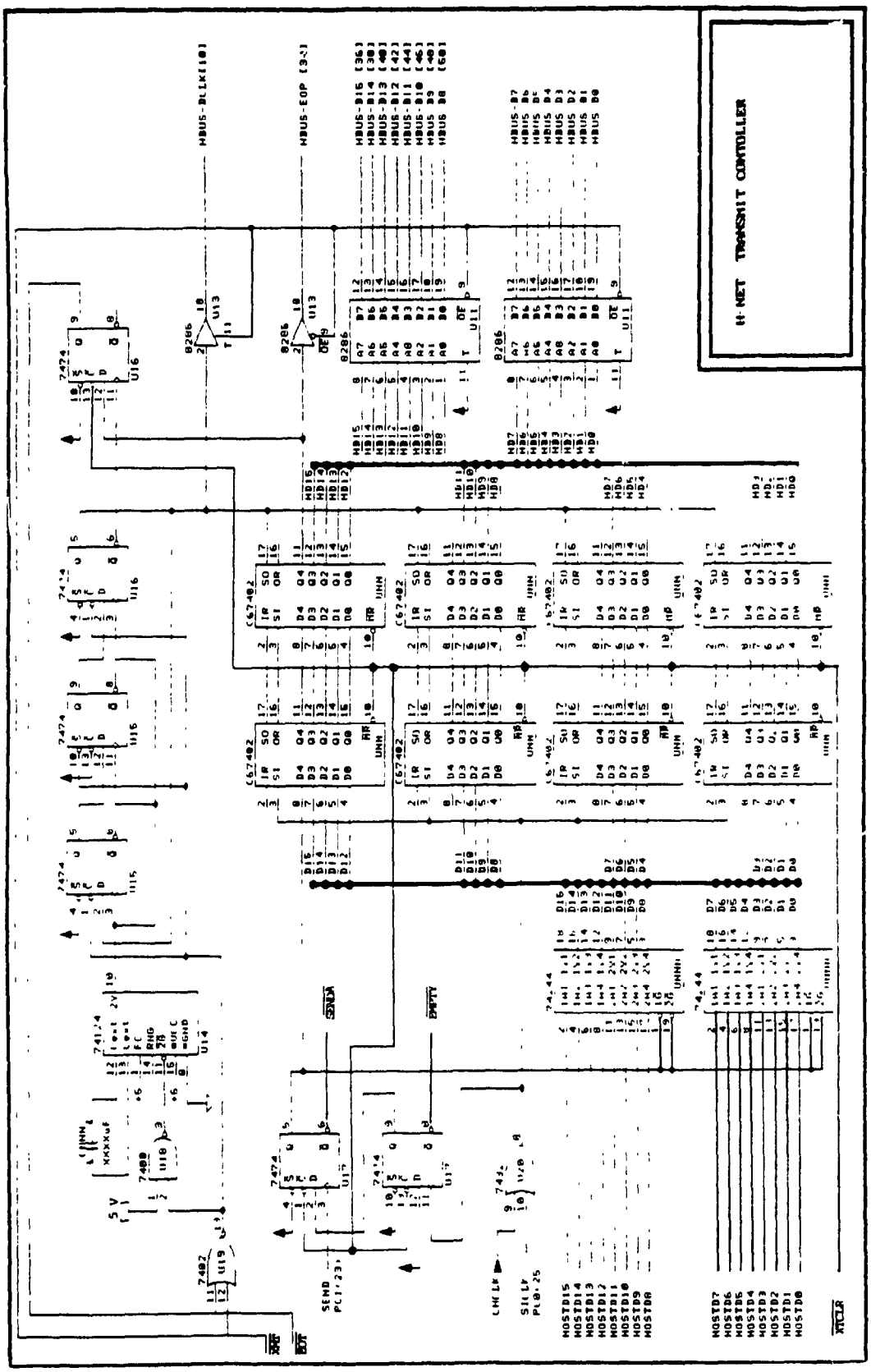
All signal names with the preface "HBUS-", connect directly to the H-Network. Signals connecting to the host have names with a "HOST" preface. Other signals connect to off page circuitry. A bar over the signal name denotes the signal as active low. Inverters of type 7406 have open-collector outputs. Technical data on the C67402 FIFO devices is found in reference [3]. Data on the discrete logic parts is contained in references [4,23].

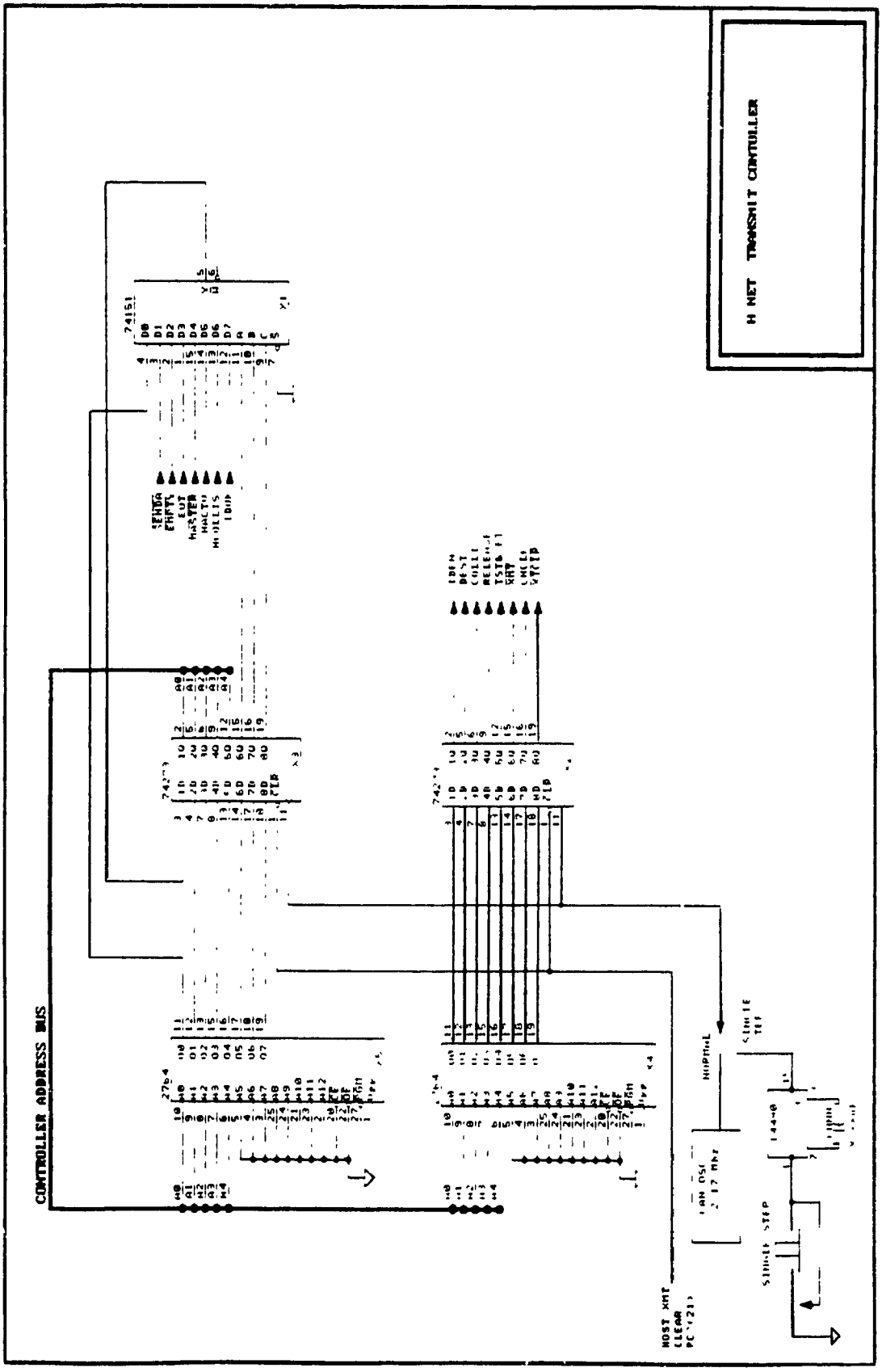
The main intention of the lab test was to prove in the microcontroller and the microcode. Only a single transmit station and a single receive station were constructed. They were interconnected through short cables. The host was simulated manually using a commercial microprocessor. Data was successfully transferred. Initial tests were done by single stepping the microcontroller, so that signals could be observed in a controlled fashion. Further tests were successful with the microcntroller free running at 2.17 Mhz.

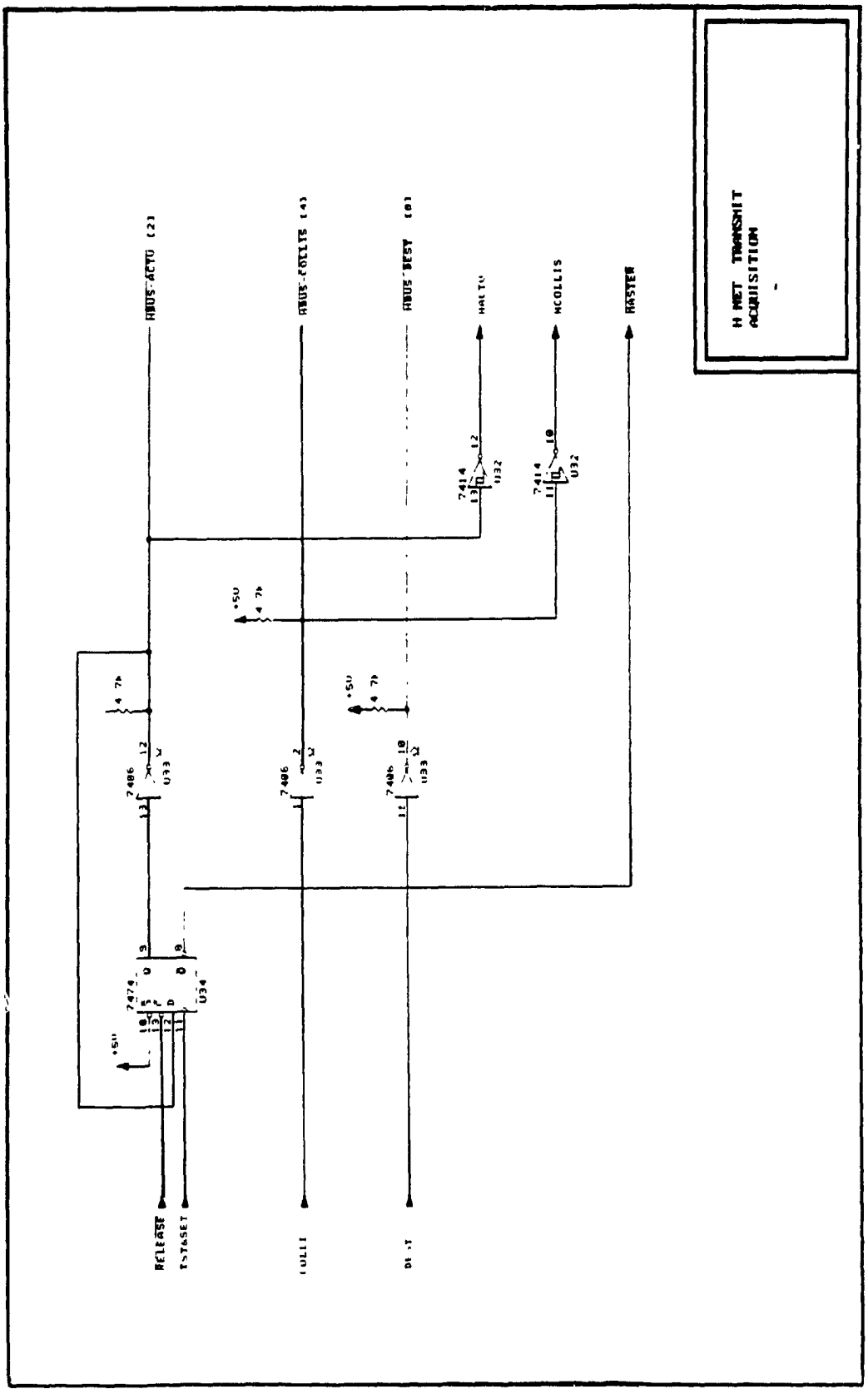
All that has been proven so far is that the protocol and the microcontroller concepts work. The next step is to wire up at least 3 complete stations and run them at full speed. This would allow proving the design under various conditions. The most important would be to confirm that collision are handled properly. The network itself should also be wired up so that data transfer rates may be confirmed.

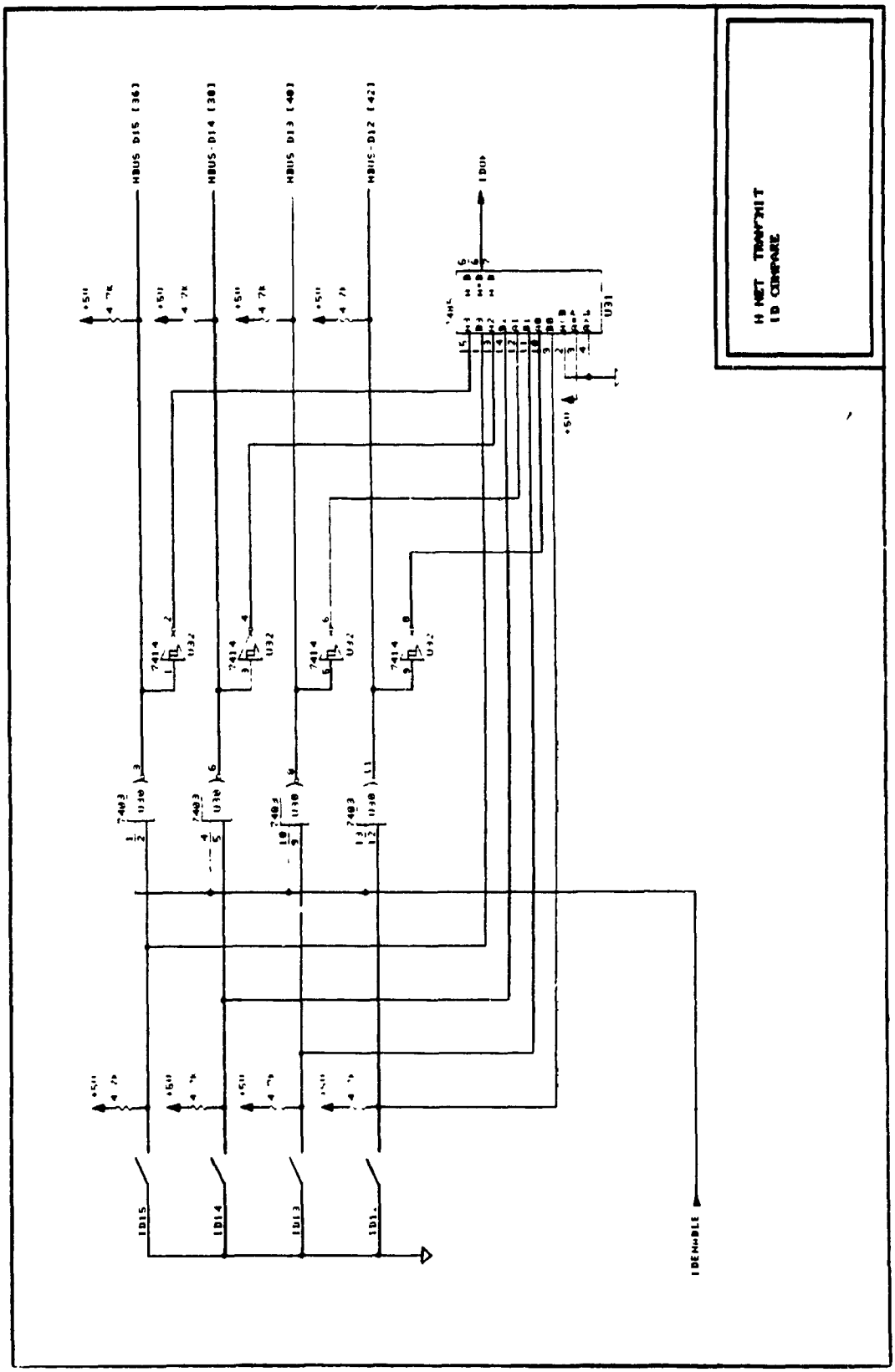

H NET RECEIVE
FIFO MEMORY
HOST RCVCLK
HOST-RCVEOP
HOST-RCVDATA
HOST-RCVCLK
HOSTD15
HOSTD14
HOSTD13
HOSTD12
HOSTD11
HOSTD10
HOSTD9
HOSTD8
HOSTD7
HOSTD6
HOSTD5
HOSTD4
HOSTD3
HOSTD2
HOSTD1
HOSTD0
74244
74S74
7406
7408
67402
8286
RSIN
HBUS-D15
HBUS-D14
HBUS-D13
HBUS-D12
HBUS-D11
HBUS-D10
HBUS-D9
HBUS-D8
HBUS-D7
HBUS-D6
HBUS-D5
HBUS-D4
HBUS-D3
HBUS-D2
HBUS-D1
HBUS-D0


H NET RECEIVE
DECISION LOGIC
DESTINATION ID


H-NET TRANSMIT CONTROLLER


H NET TRANSMIT CONTROLLER
CONTROLLER ADDRESS BUS


H MET TRANSMIT ACQUISITION
ABUS-ACTU (2)
ABUS-POLLS (4)
ABUS-RESY (8)
MALTU
MCOLLIS
MASTER
+5V
4.7k
7414
U32
7486
U33
7474
U34
RELEASE
TESTSET


H NET TRANSMIT
ID COMPARE
HBUS D15 (36)
HBUS-D14 (38)
HBUS D13 (40)
HBUS-D12 (42)
+5V
4.7K
7414
U32
7403
U30
74LS
U31
ID15
ID14
ID13
ID12
IDENABLE

APPENDIX C

MICROCONTROLLERS AND MICROINSTRUCTIONS

Control circuits may be hardwired using standard logic design techniques. If the circuit is complex or subject to change, a microcontroller approach is more suitable.

Microcontrollers perform the desired control function by executing microinstrunctions stored in ROM. The sequence of microinstruction executed is based on a set of input status lines. At each step, the microcontroller sets the output lines to the desired state.

A microinstruction can be either horizontal or vertical [13,18]. In horizontal coding, the microinstruction will have a bit defined for each control signal that needs to be generated. This allows setting any combination of control signals for each microinstruction executed. However if there are many control signals that are mutually exclusive, then a vertical microinstruction may be more suitable. This results in a more narrow microinstructions as compared to the equivalent horizontal microinstruction. But since only one control signal changes state per vertical microinstructions, more clock cycles are needed to perform the same task.

As an example, say we need to control 64 lines. A horizontal microinstruction would consist of 64 bits in the control field, plus X bits for the other fields. A vertical microinstruction would only need 6 bits for the control field, plus X bits for the other field. If during execution we needed to set 5 lines to a particular state, it would take 5 vertical instructions but only 1 horizontal instruction. If instead we needed to toggle the 5 lines one at a time, then both the

vertical or horizontal approach would require the same amount of instructions.

The microcontroller proposed in this work as the station controller for the H-Network is essentially a horizontal microcontroller. A bit is dedicated in the CSD field for each of the 8 control lines. However the 3 bit Conditional Branch selects one of eight signals at each step (ie. vertical coding).