MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Laboratory for Computer Science
Cambridge, Massachusetts 02139

Computation Structures Group Memo 199

The Development of a Prototype Router: Design, Implementation, and Test Procedures

by

Joel Lilienkamp

(S.B. Thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, Ma.)

September 1980

The Development of a Prototype Router:

Design, Implementation, and Test Procedures

by

Joel E. Lilienkamp

Abstract

This thesis concerns the design and debugging of a self-timed router to be used as a modular element in the construction of packet switching networks used in a data flow computer. The design is presented and the tests used are described. Also included is a discussion of the prototype implementation. The conclusions indicate what has been done, and what remains unsolved.

Thesis Supervisor: Jack B. Dennis

Title: Professor of Computer Science

CONTENTS

FIGURES

1. Introduction

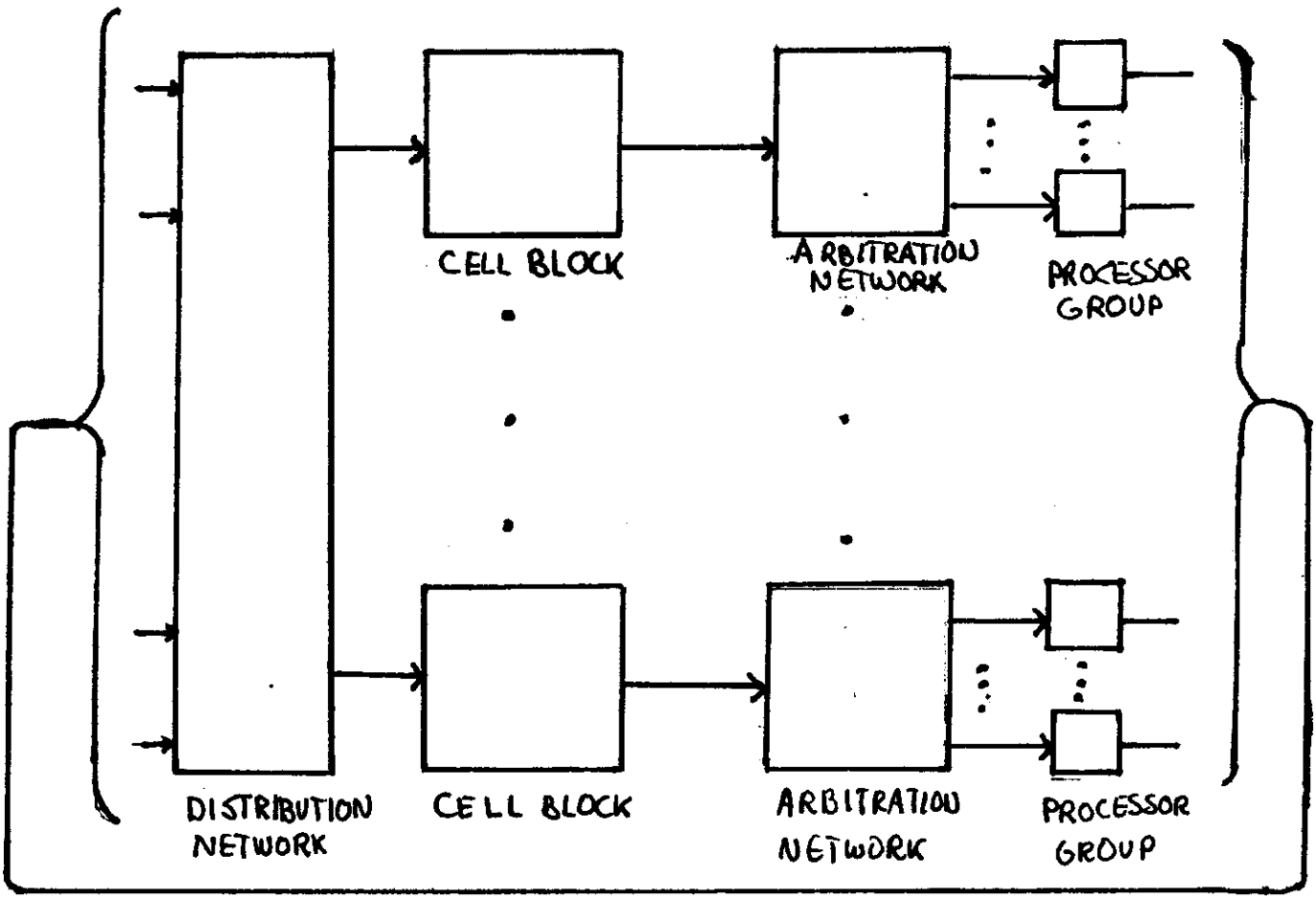
A data flow computer is one in which instructions are executed in response to the arrival of all of their operands, as opposed to a conventional computer that executes instructions based on their position in the program's sequence. Given a suitable machine architecture with multiple processors, this can permit concurrent execution of instructions and thus a decrease in execution time of a program.

A machine that operates on such principles is currently under development at MIT. The configuration of this machine is shown in figure 1. The machine consists of an arbitration network section, a processor section, and a distribution network section. The initial prototype version implements the cell blocks and processors together so that only these elements and the distribution network are required. Future models of the machine will contain specialized processing elements and an arbitration network, so it is beneficial to consider their functions separately here.

Each cell block contains a number of cells in which instructions are stored, and a cell block manager. Each cell contains an operation template, which consists of an operator, space for operands, and a list of destinations for the result. The cell block manager is responsible for receiving result packets from the distribution network (discussed below) and storing the result in the operand template of the designated cell. Furthermore, when it discovers that a cell has all its operands stored in the template, it creates an operation packet that completely describes the operation and operands in a form usable by the processing element, and dispatches it to the corresponding arbitration network. The arbitration network routes the operation packet to the appropriate processors. The processors perform the required operations using the operands from the packet and then create a result packet containing the result of the operation for each destination. The distribution network distributes the result packets to the cell blocks.

One implication of this type of architecture is that local control is required to achieve concurrent operation, but the overall activity must be somehow coordinated. To accomplish this coordination a

Fig. 1. General Form of a Data Flow Computer

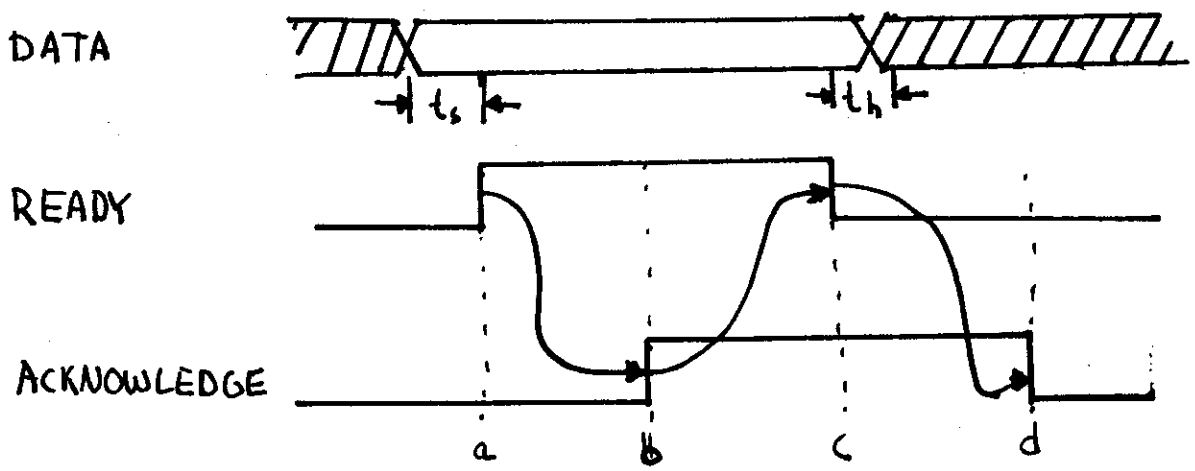


Each cell block is identical. Each arbitration network is also identical.
Each processor in a group is different but each group is the same.

---

uniform communications protocol was adapted, which must be used by all modules when communicating with any other module. It was decided that all packets would be sent in a byte-serial format, using the control handshaking protocol known as reset signalling. This protocol is shown in figure 2. The sender presents a byte of data and raises the ready signal. The receiver gives an

Fig. 2. Reset Signalling Communications Protocol



Data is valid from t_s, called the set-up time, before transition a until t_h, called the hold time, after transition c. The transitions must occur in the order shown but may take an arbitrarily long amount of time.

---

acknowledge when it no longer requires the data. The transmitter can lower ready after the acknowledge goes high. The receiver lowers acknowledge when it is ready to receive another byte. The last byte of the packet is indicated by an additional bit of data, which is one only during the last byte.

A broad introduction to this work can be found in [4]. This paper is concerned with a modular element which can be used to implement the arbitration and distribution networks.

The architecture is expanded rather easily, except that the arbitration and distribution networks can become quite large. Because it is impractical to build large networks as a single circuit and it is desirable to have such networks expandable, a more modular approach to network construction is desirable. One such approach is constructing the networks from two-by-two routers. A router is essentially a two-input two-output switching network. The router receives packets at its two input ports and switches them to

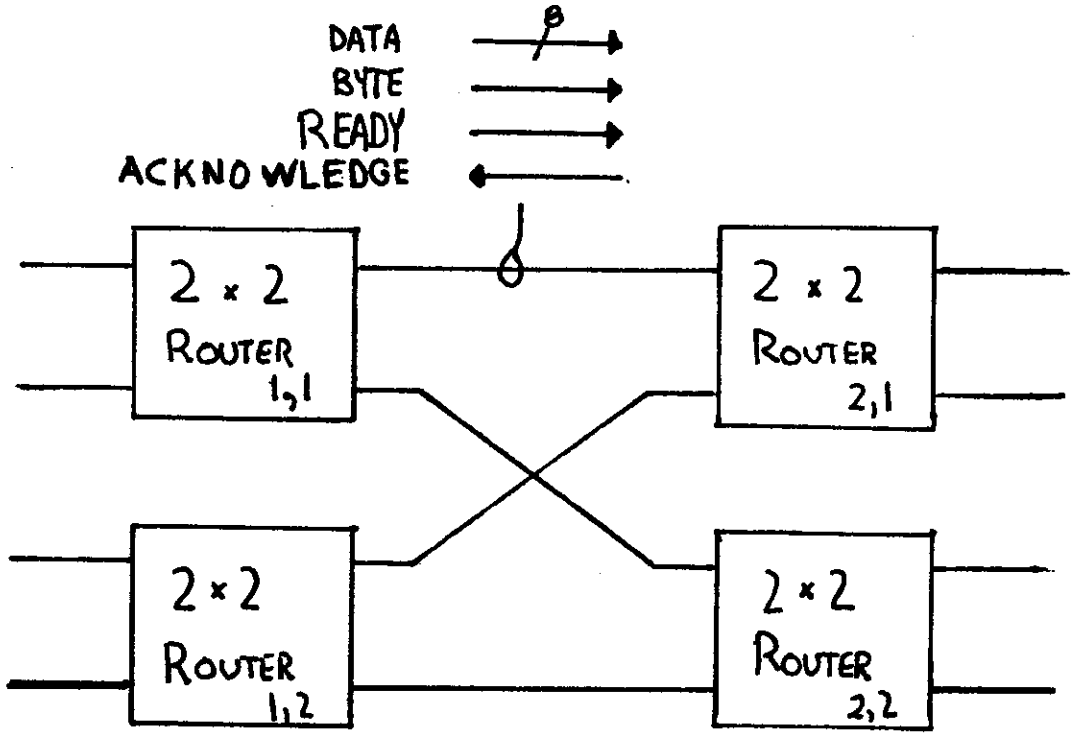
one of the two output ports, depending the value of a bit in the first byte of the packet. If the desired output port is in use by a packet from the other input port already in progress then it must wait until the port is is free. Both input ports can operate concurrently if they require different output ports.

Figure 3 shows a four-by-four network using four routers. Figure 4 shows how to construct an n-by-n network (where n is a power of 2) from routers. A simple analysis will show that n/2 routers are required in each column, and $\log_2 n$ columns are required, or a total of $n/2 \log_2 n$ routers to construct an n-by-n matrix.

The next section discusses a design for an asynchronous self-timed router which can be constructed using standard (low power Schottky) TTL components. Included is a discussion of modular

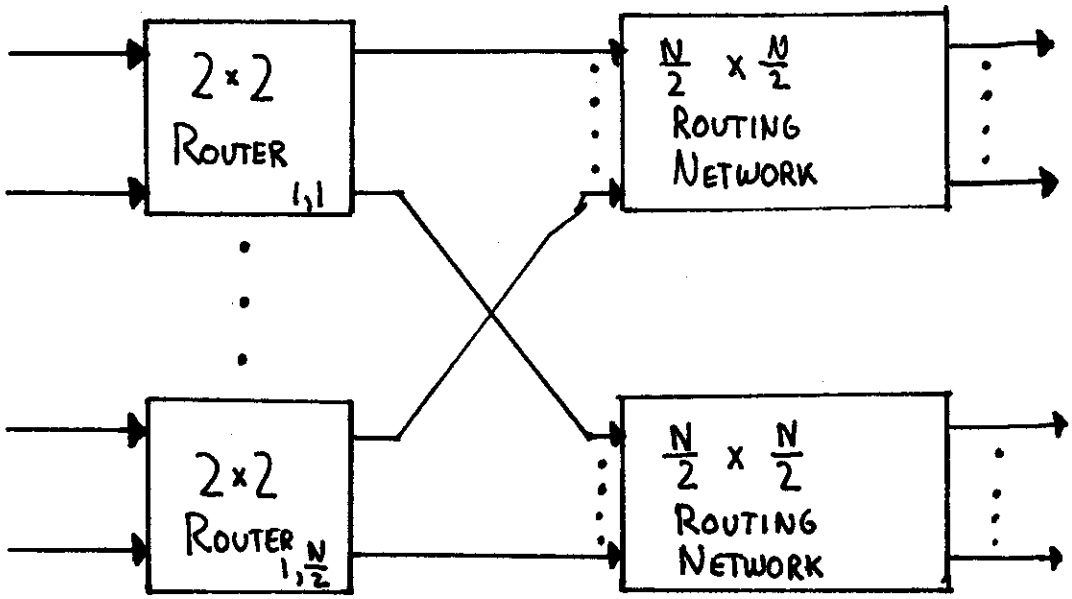
---

Fig. 3. 4 x 4 Routing Network Constructed of 2 x 2 routers



The link detail applies to all connections.

Fig. 4. N x N Routing Network Defined Recursively



N/2 x N/2 routing networks are constructed using 2 x 2 routers.

---

decomposition, a detailed discussion of implementation of all modules, and a top level discussion of operation of the router. The final section discusses techniques to be used to test and debug the router. Appendix I contains a discussion of the implementation of the prototype router, and Appendix II discusses the additional hardware required to perform the testing and debugging.

2. A Design for an Asynchronous Self-timed Router

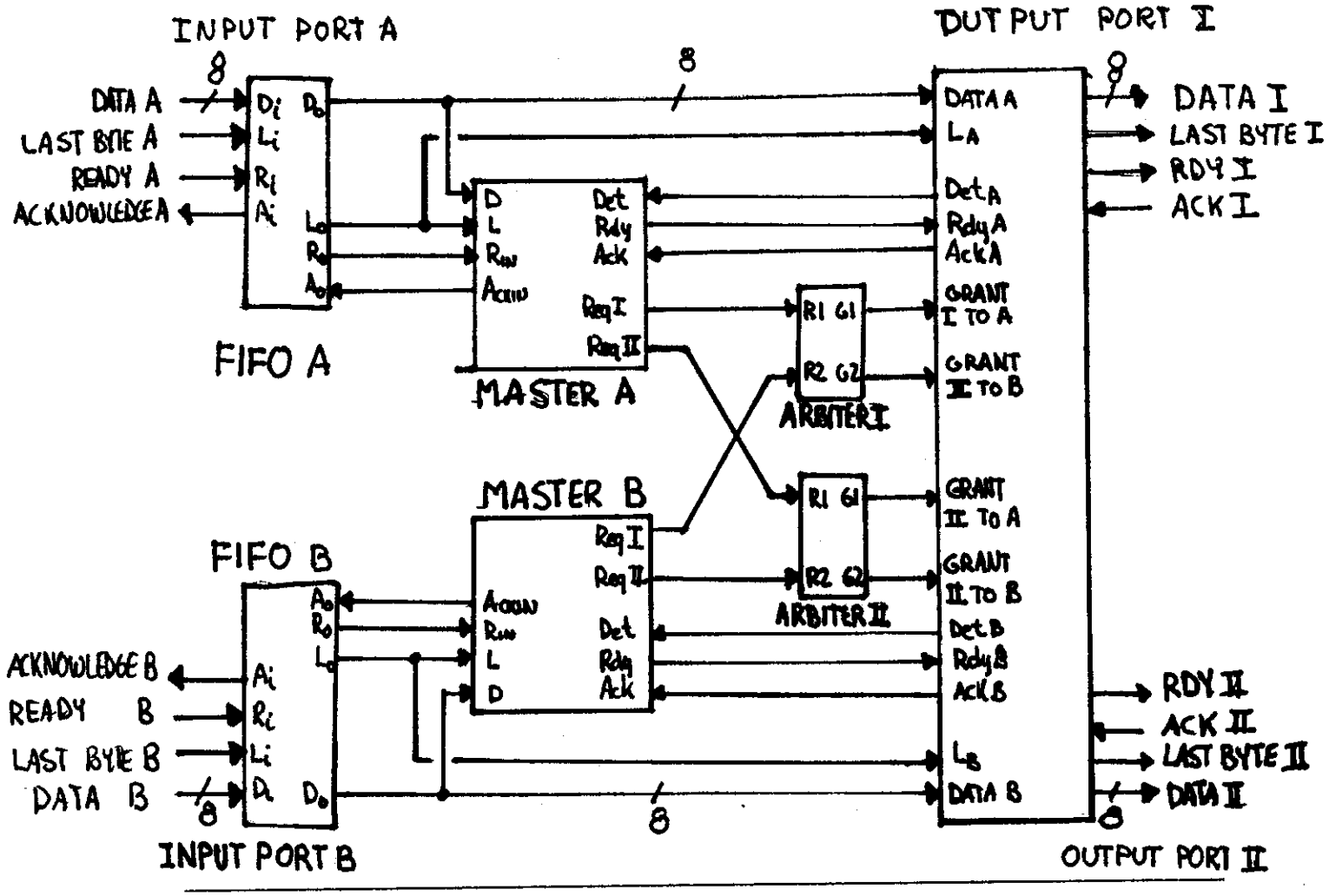
As stated in the previous section the router receives packets at its two input ports and directs them to one of two output ports, providing the necessary arbitration of output ports when conflicts arise and permitting concurrency when they do not. Also, it must send and receive each byte using the reset signalling protocol

Before the design is presented it is necessary to consider a couple of design choices that were made. The major consideration was keeping the design simple, so that the number of components would be small and the cost would be kept down. The major consequence of this is that the router is not very smart and cannot make many decisions. For example, the output port is selected on the basis of the first bit in the first byte of the packet. Since the bit examined by routers in different columns should be different, it is necessary to rearrange the data lines going into the router, and correct them going out. This implies the router cannot use the data of the packet in any other way, because it is likely to be scrambled. Another limitation caused by the simplicity constraint is that the router is particularly dependent on correct use of the reset signalling protocol, and is subject to failure if it is violated.

A block diagram of the design is shown in figure 5. There are four types of modules used in this design: two FIFO modules, two Master modules, two Arbiter modules, and one Multiplexor module.

The FIFO modules are sixteen word first-in-first-out buffers with reset signaling interfaces. There is one of these modules for each input port. Whenever a byte is transmitted to that input, the data and last byte values are stored in the memory and the acknowledge signal is asserted. If the buffer is full then the data is not stored and no acknowledge is given. Whenever the buffer is non-empty the FIFO module generates a ready signal to the Master module. The purpose of this buffer is to allow a packet to be stored in a single router, thus preventing the network from backing up when a packet is forced to wait for an output port.

Fig. 5. Block Diagram of a 2 x 2 Router



The Master module is the input port controller. It examines the first bit of the first byte of the packet and generates a request to the appropriate output port arbiter. It also relays ready and acknowledge signals from the FIFO module to the Multiplexor module during the transmission of all bytes. After the last byte of the packet is sent the Master module drops the request for the output port and resets itself.

The Arbiter module performs a well known function: that of granting mutually exclusive use of a resource, in this case the output ports. It receives requests for the port asynchronously and grants at most

one request the right to use the port. When two requests arrive at about the same time it can decide to grant acknowledge to either but not both requests. The method to determine the winner can be arbitrary, but it should not favor one request over the other. Such a condition could lead to an unbalanced network since a path with many "disfavored" requests could be subject to excessive blocking.

The Multiplexor module is responsible for linking an input port to an output port. An input port is linked to an output port only when all data and control signal paths from the input port are connected to those of the output port. This total link can exist only when the grant signal from the arbiter is asserted. Since each input port has its own Master module and each output port has its own arbiter concurrent linkages can occur without sacrificing mutually exclusive use of the output ports.

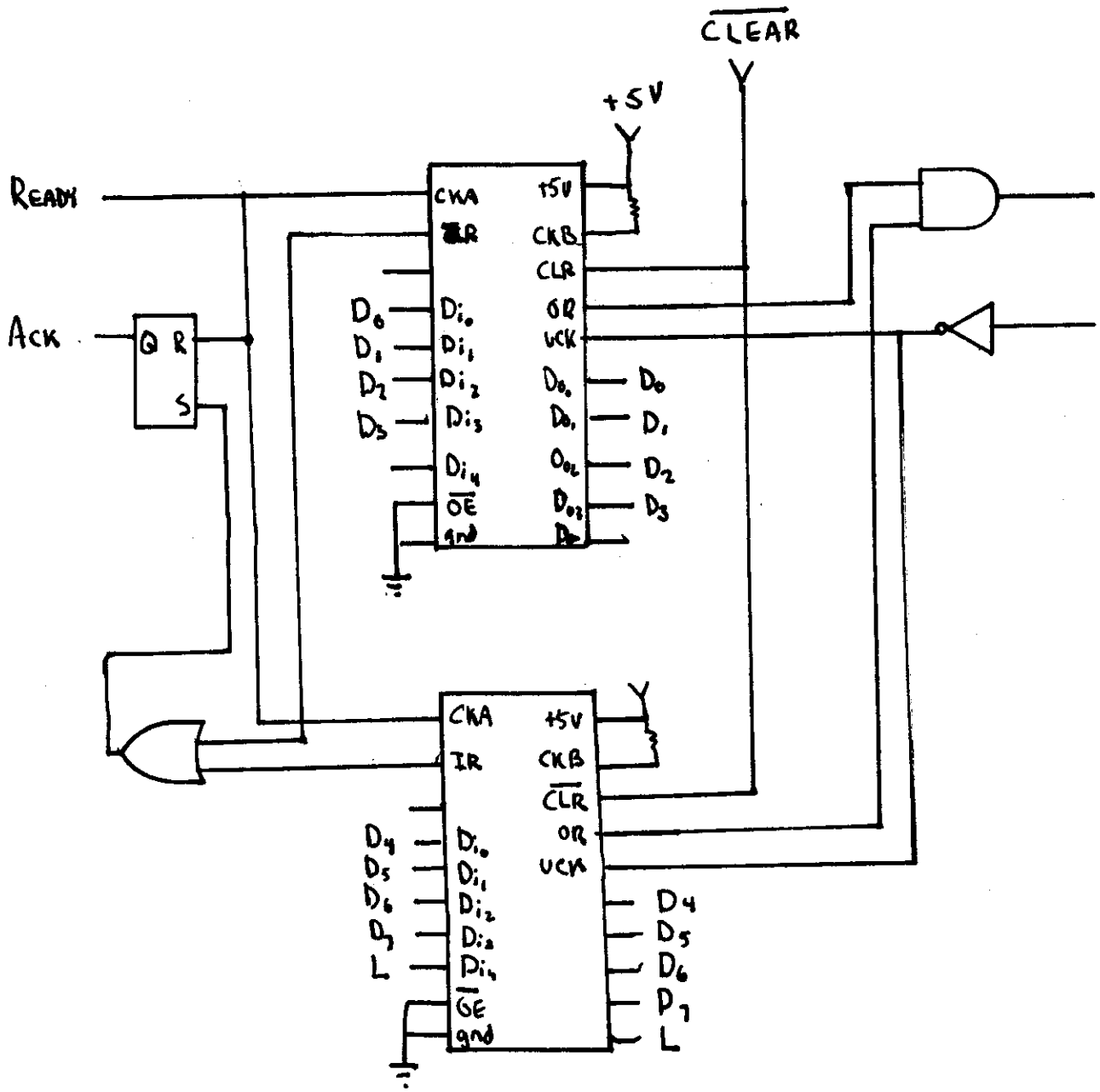
Before a discussion of the overall operation of the router a detailed discussion of the design of each module is given. Included is a discussion of the timing issues when they are relevant.

2.1 The FIFO Module

The circuit diagram for the FIFO module is shown in figure 6. The main element of this design is the 74S225, a single chip asynchronous sixteen-by-five bit first-in-first-out memory. (A detailed discussion of this chip can be found in [11])

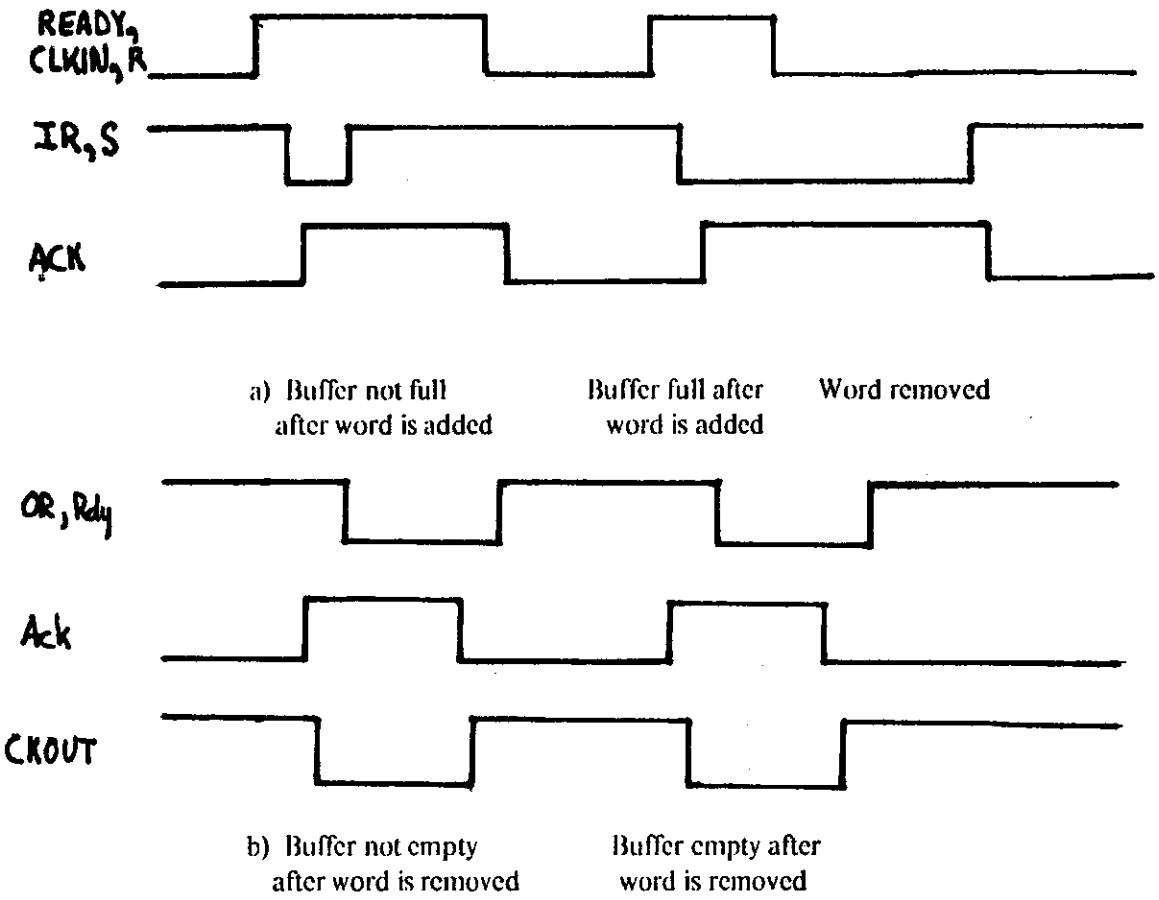
The input timing sequence of the FIFO module is shown in figure 7a. The ready signal is connected directly to the clock input. A positive pulse on the clock input causes the data on the input lines to be stored in the buffer. During this pulse, the IR outputs pulses low, causing the acknowledge flip-flop to be set. This will remain high until the ready signal goes low, causing the flip-flop to be reset. When the buffer is full, the IR signal will remain low. Thus the acknowledge line will remain high if the buffer is filled.

Fig. 6. Circuit Diagram of FIFO Module



The output timing of the FIFO module is shown in figure 7b. The output ready line is connected directly to the ready out, and the Acknowledge signal is inverted and connected to the output clock. The

Fig. 7. Timing Sequence for the FIFO Module



clock is normally high, and data is removed from a low to high transition (a high to low transition of acknowledge). Valid data in the buffer causes the output ready to go high. When the output clock is lowered the output ready goes low. When no valid data is stored in the buffer the output ready remains low.

The difficult aspects of designing an asynchronous FIFO are hidden by the use of the 74S225. All that was added was the reset signalling interfaces.

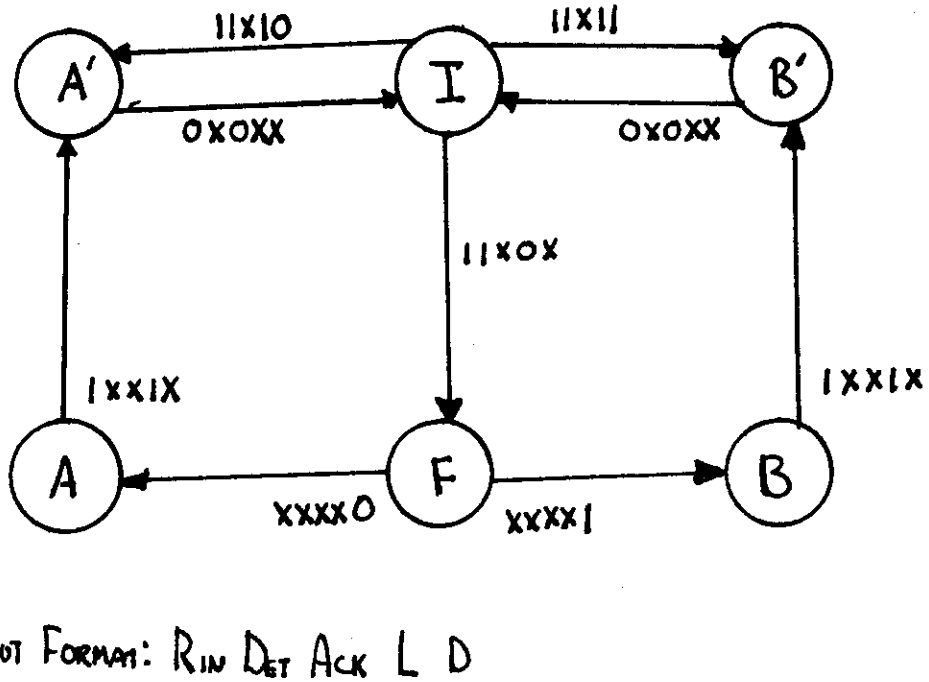
2.2 The Master Module

The most complex module in terms of function is the Master module, which controls the input port, generates request signals for the output ports, and relays the ready and acknowledge signals from the input to the output. This module is implemented as an asynchronous sequential circuit. For this reason the design will be discussed in terms of an abstract Finite State Machine (FSM) before discussing the electronic implementation.

The FSM has five input variables, which are the ready signal (Rin) from the input (actually from the FIFO module), the acknowledge signal (Ack) from the output (actually from the Multiplexor module), the direction (D) and last byte (L) bits from the data lines, and the detach signal (Det) from the Multiplexor module, which indicates if the input port is linked to either output. The FSM also generates four outputs, which are the acknowledge signal (Ackin) to the input, the ready signal (Rdy) to the output, and the request lines (ReqI and ReqII) for each output port.

The state transition diagram appears in figure 8. The machine remains in state I, which is the initial state, until both Rin and Det are one. Then, assuming for the moment that L is zero, the machine changes to state F. This is an intermediate state, which immediately induces another state change. The next state is determined by D. If D is zero the next state is A; otherwise it is B. When the output port is linked the Det input changes to zero. While in State A ReqI is made; while in state B ReqII is made. The machine remains in this state for all bytes of the packet except for the last. When L changes to one the machine changes from state A to A', or from state B to B'. After the transmission of the last byte the Master module removes the request for the output port and resets itself. When the output port is detached (no longer linked), the Det input will return to one. The Master module cannot begin processing another packet until Det is one. If the first byte of the packet is also the last, then the machine goes directly to state A' or B', depending on the value of D. After the byte is transmitted the machine resets itself as it did in the case of a multibyte packet.

Fig. 8. Master Module State Transition Diagram



When the inputs match the pattern on the transition, the machine changes to the next state indicated. For simplicity the outputs are not included.

One implication of constructing an FSM as an asynchronous sequential circuit is that inputs and state variables can change at any time instead of on clock pulses. To insure that such a circuit will work properly it is necessary to consider timing requirements carefully. It is helpful, for example that only one state variable changes at one time, and one change must be allowed to stabilize before the next change is induced.

The state assignment and state transition table for the master module is given in figure 9. The first state variable (Y1) is to one if the Master module is in the middle of a packet. If it is zero then it is

Fig. 9. Master Module State Transition Table

				R_{IN}, Ack															
				L, D															
				00				01				11				10			
S	Y_1	Y_2	Y_3	00	01	11	10	00	01	11	10	00	01	11	10	00	01	11	10
I	0	0	0	(I)	(I)	(I)	(I)	(I)	(I)	(I)	(I)	F	F	B′	A′	F	F	B′	A′
B′	0	0	1	I	I	I	I	(B′)	(B′)	(B′)	(B′)	(B′)	(B′)	(B′)	(B′)	(B′)	(B′)	(B′)	(B′)
E	0	1	1	I	I	I	I	(E)	(E)	(E)	(E)	(E)	(E)	(E)	(E)	(E)	(E)	(E)	(E)
A′	0	1	0	I	I	I	I	(A′)	(A′)	(A′)	(A′)	(A′)	(A′)	(A′)	(A′)	(A′)	(A′)	(A′)	(A′)
F	1	0	0	A	B	B	A	A	B	B	A	A	B	B′	A′	A	B	B′	A′
B	1	0	1	(B)	(B)	(B)	(B)	(B)	(B)	(B)	(B)	(B)	(B)	B′	B′	(B)	(B)	B′	B′
G	1	1	1	(G)	(G)	(G)	(G)	(G)	(G)	(G)	(G)	(G)	(E)	E	E	(G)	(G)	E	E
A	1	1	0	(A)	(A)	(A)	(A)	(A)	(A)	(A)	(A)	(A)	(A)	A′	A′	(A)	(A)	A′	A′

The correspondence between state variables and states is indicated in the table. A circle around the next state indicates it is the same state; that is, no transition occurs.

---

transmitting the last byte of the packet, or it is idling. The second and third state variables (Y2 & Y3) indicate to which output port the packet is being routed: 10 indicates output port I, and 01 indicates output port II. 00 indicates that the Master module is idling or has not yet chosen an output port, and 11 only occurs during an error condition. In both of these cases no request for an output is made. Figure 10 shows the state transition table with the implicants for the Rdy and Ackin outputs. The Reql and ReqII are generated using state variables Y2 and Y3.

The circuit for the Master module is shown in Figure 11. The state variables are implemented using asynchronous SR flip-flops, with the S and R inputs normally high. For this implementation it is desired that if both S and R are low, then the output is zero. This is called Reset overrides Set. The 74LS279

Fig. 10. Output Signal Implicants

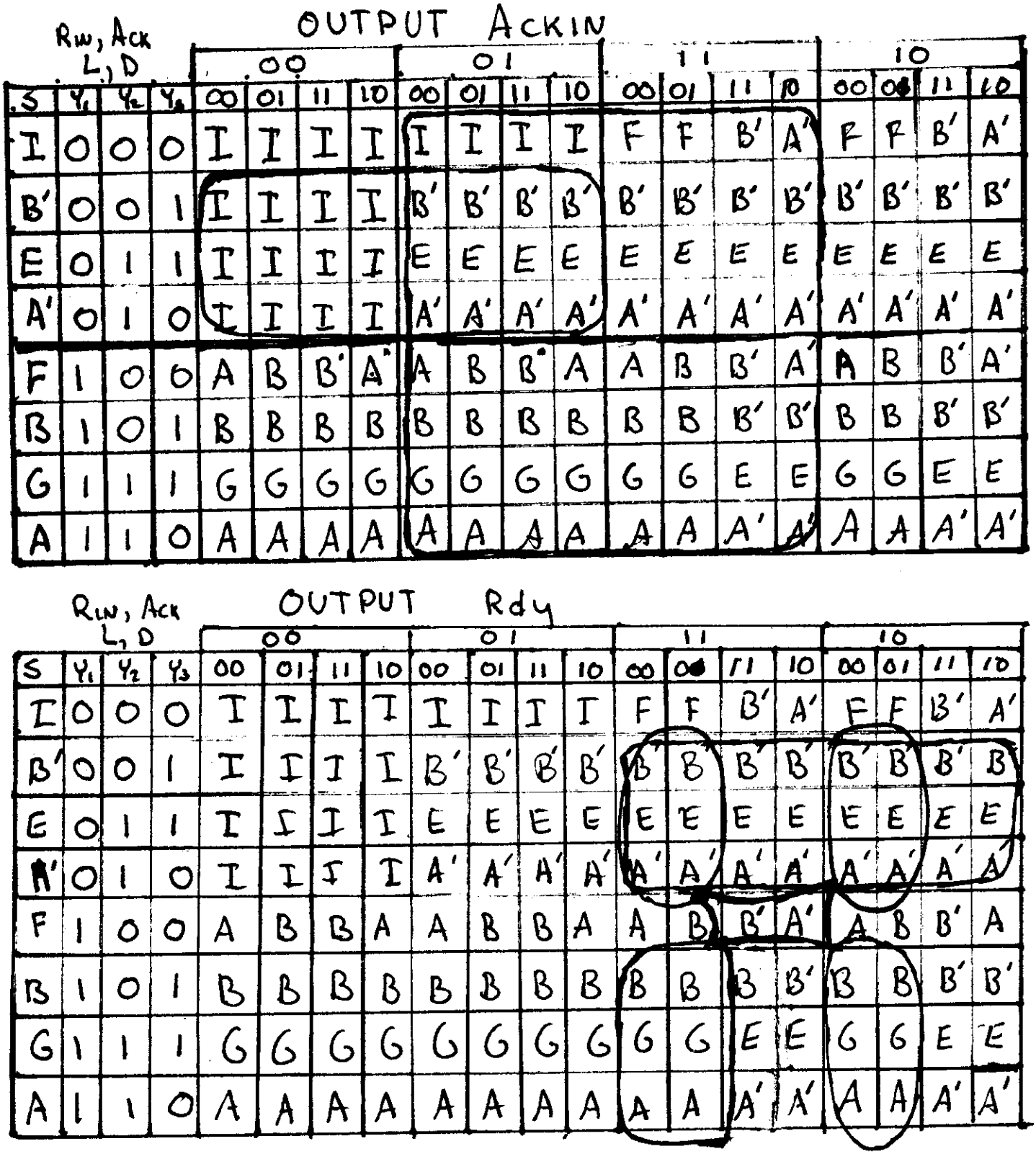
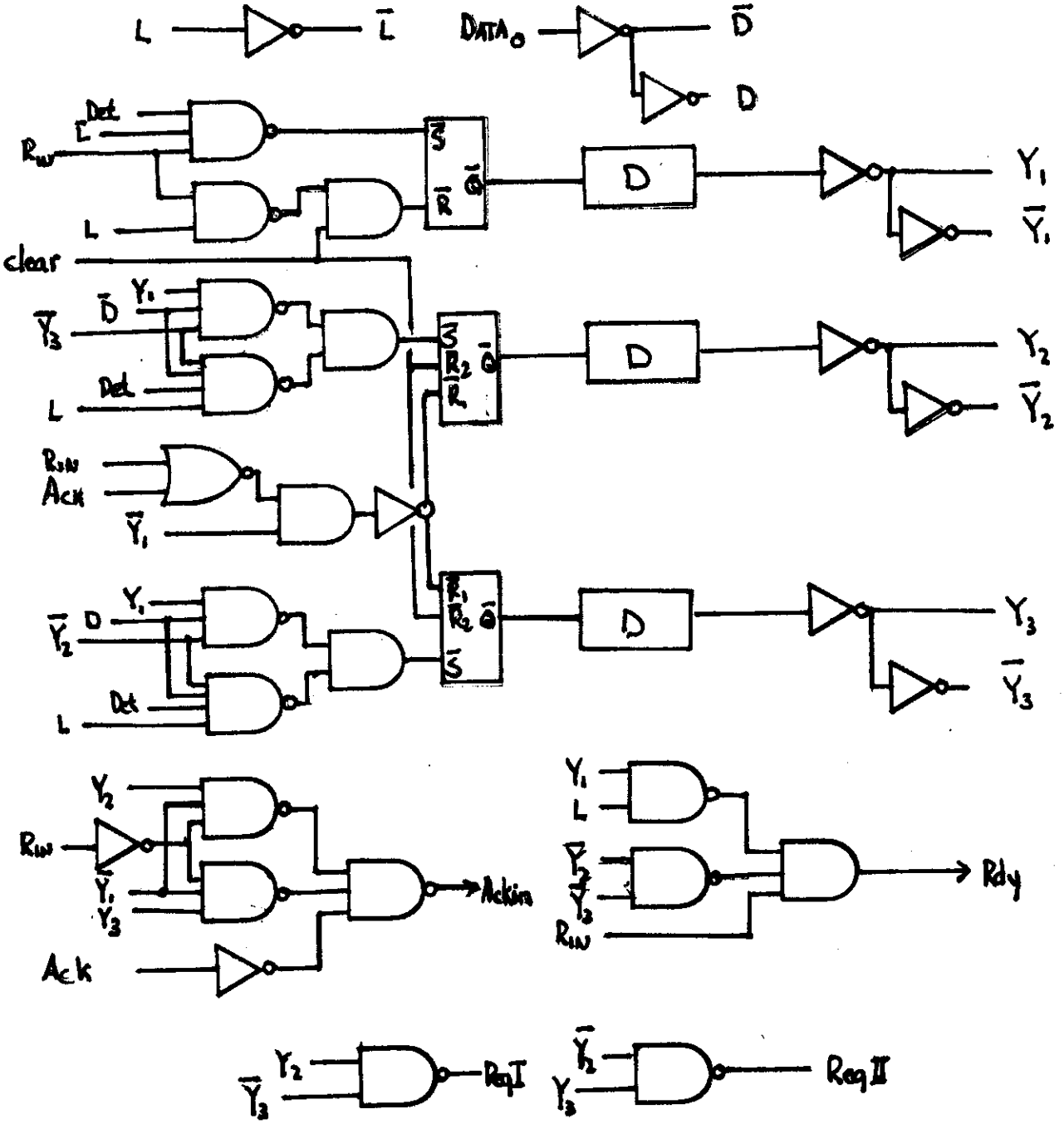


Fig. 11. Master Module Circuit Diagram



contains four asynchronous SR flip-flops, but they are of the type Set overrides Reset. This difficulty can be resolved by reversing the S and R inputs and inverting the output of flip-flop. The delay lines are used

to insure that a state variable change cannot induce a new state variable change quickly.

The timing diagram for the Master module is shown in figure 12. The diagrams represent typical waveforms only. In particular, the response times of the outside devices is arbitrary, with the restrictions of the reset signalling protocol. Part a shows a typical timing for the first byte of a multibyte packet. Part b shows a typical timing for a middle byte of the packet, and part c shows the last byte. Part d shows typical waveforms for a single byte packet. In the example both packets were routed to output I for simplicity; the other case is similar.

In the diagram an arrow from transition A to transition B indicates that the latter transition is dependent on the occurrence of the former. This means that either A directly caused B in the case of internal signals and output changes, or that A precedes B in the reset signalling protocol in the case of input signals. The more interesting cases are indicated by number on the diagram.

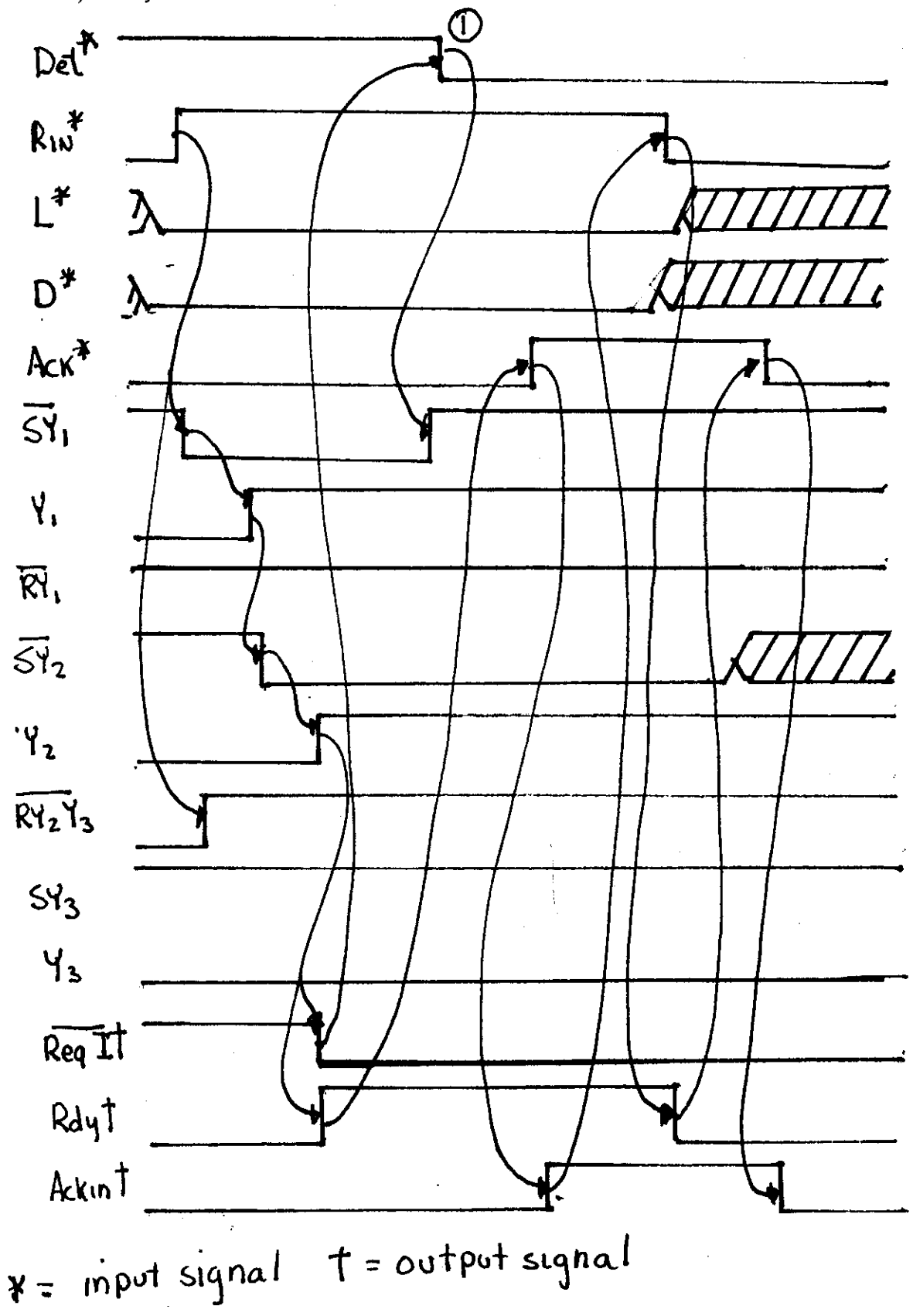
1. This is the transition of the detach signal. It occurs after the arbiter has granted the output port to the Master modules request. After this occurs the ready is sent to the output port and Ack from the output will come after that.
2. In this case the Rin is not automatically relayed to Rin, but delayed until the machine has entered the last byte state, A' or B'.
3. The Ackin signal is held high after the Ack signal goes low, until the Master module has reset itself to the initial state.
4. The detach signal goes high after the grant signal from the arbiter goes low. Although it is possible for Rin to go high during this time, the Master module cannot leave the initial state until Det is one.

Cases 1, 3, and 4 also appear in the diagram for the single byte packet. An additional case is also present.

5. Initially, R and S were low, but since Reset overrides Set the output remains zero. When R goes high, the output changes to one.

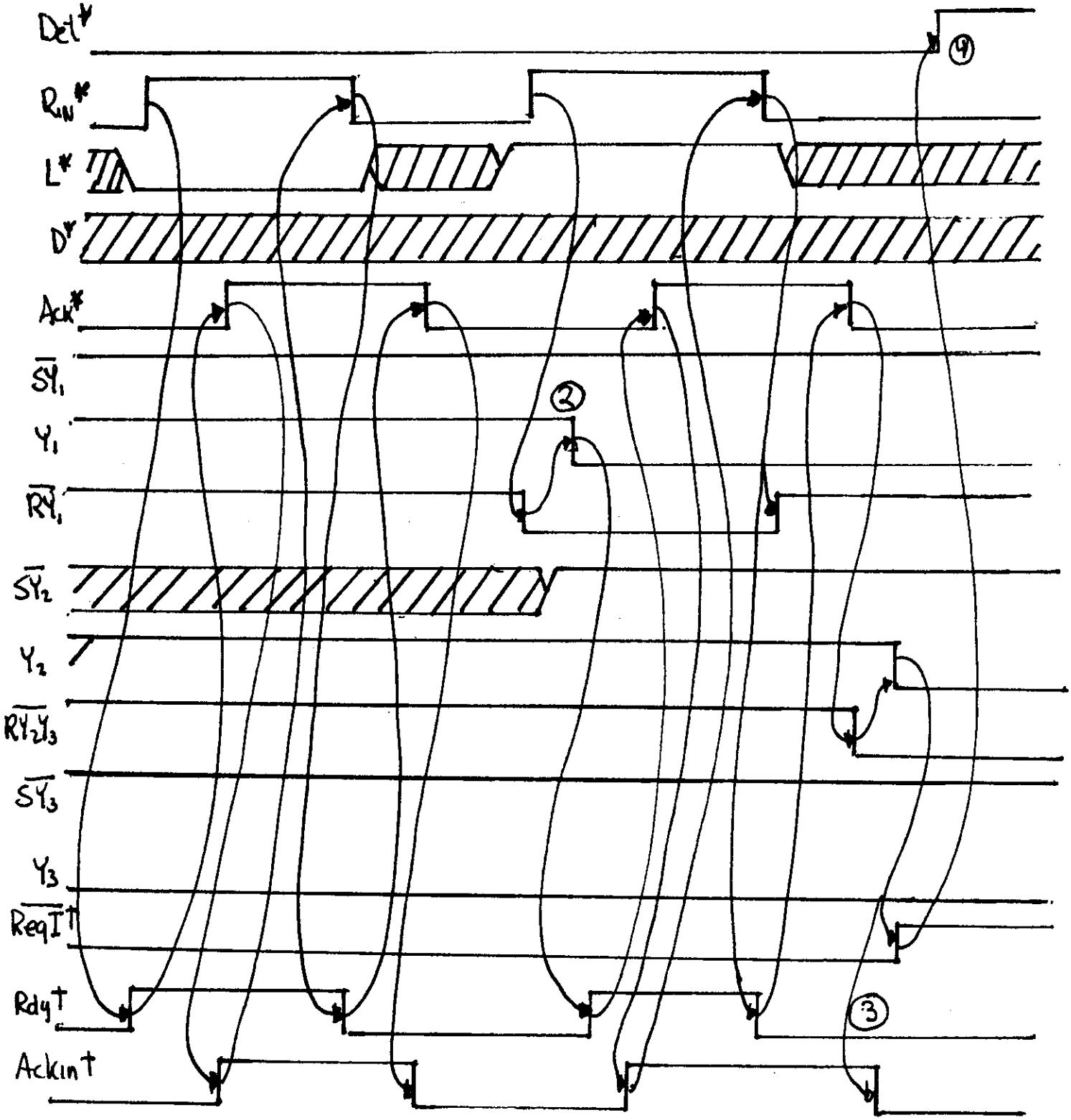
Fig. 12. Master Module Timing Diagrams

a)First Byte

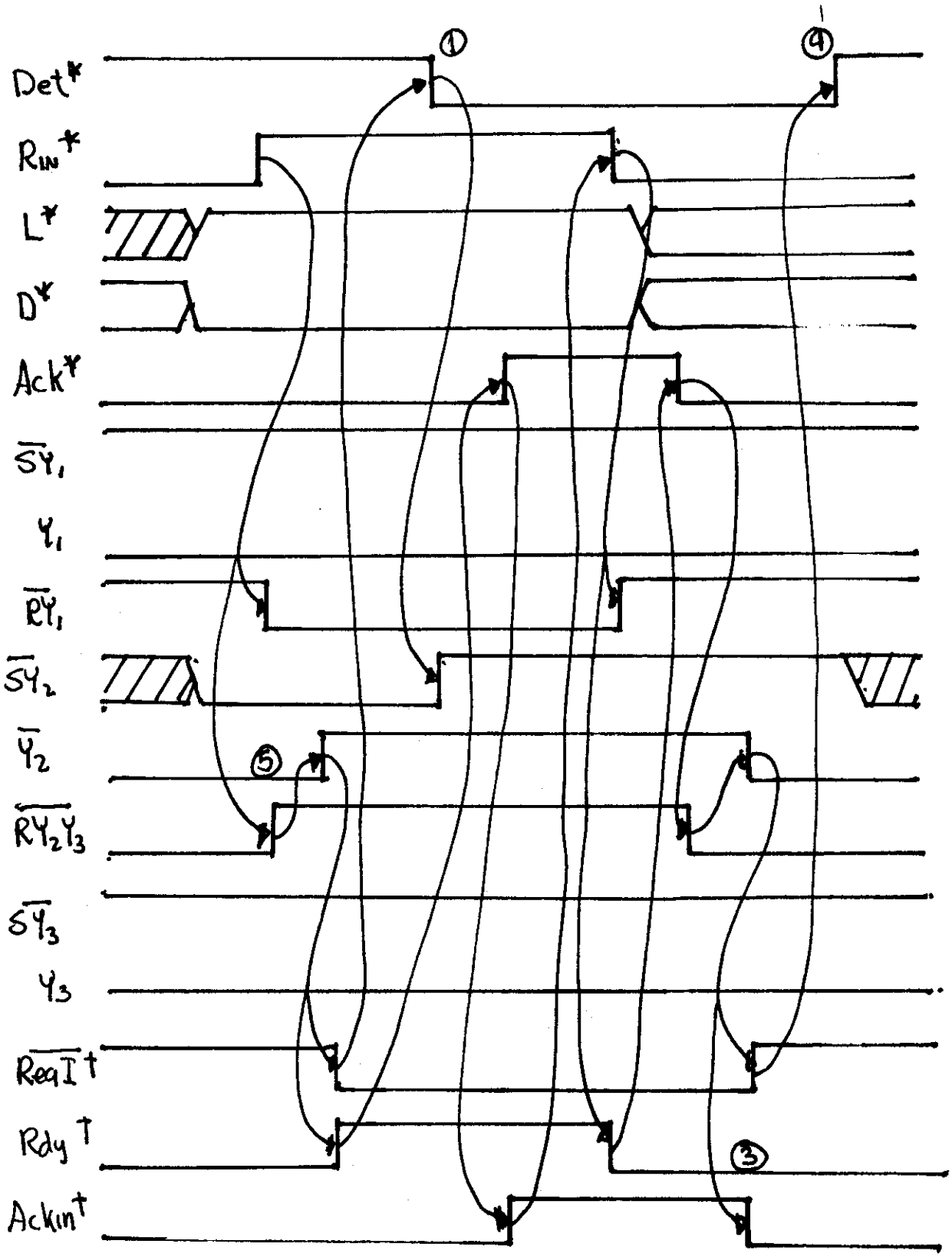


* = input signal t = output signal

b)Middle Byte c)Last Byte



d)Single Byte Packet



These are the only possible types of timing sequences that appear in the Master module.

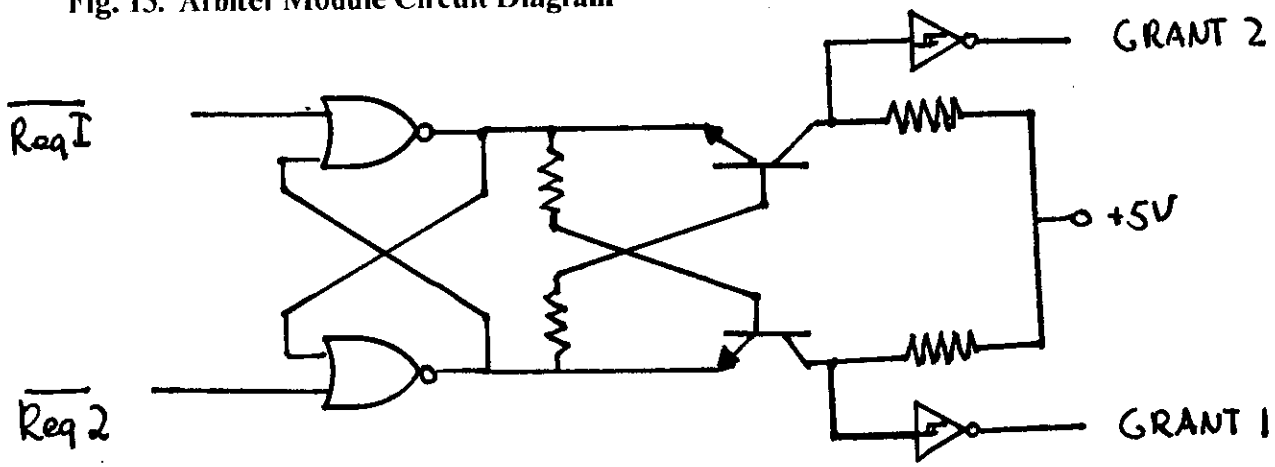
There are two extra, prohibited states that could potentially occur in the Master module. These occur when both state variables Y2 and Y3 variables are one. Abnormal input for the duration of several tens of nanoseconds could put the Master module in one of these states. It is possible, for example, that during the first byte of a packet the value of D might change after one of Y2 or Y3 had been set, but before the change had propagated through the delay. Then the other of Y2 and Y3 would be set. Minimizing the delay reduces the likelihood of such an error occurring. If such a state is entered then no request signal is held to the arbiters. It is not possible to prevent erroneous signals on the ready and acknowledge lines if such a state occurs, at least without adding considerably to the complexity of the Master module. Also, a violation of reset signalling protocol by an outside interface could cause the Master module to violate the protocol in its output signals. For this reason it is important that whatever devices interface the router should be able to function in the presence of erroneous signals. It is not known at this time what affect this will have on the construction of networks.

2.3 The Arbiter Module

An arbiter module is a basic building block in digital systems where two or more concurrent activities wish to share a single resource. The arbiter used in the router is actually a simple arbiter, which has two request signals as inputs and two grant signals as outputs. When a request is made its corresponding grant signal is given, but only if the other grant signal has not been given. When requests come at the about the same time the winner can be decided randomly, but it should not favor one request over the other, and it should never issue two grant signals simultaneously. When a grant signal is given it is held until the request is withdrawn.

The circuit for the arbiter module is shown in figure 13. The circuit is an adaptation for TTL based on an MOS arbiter circuit found in [5]. This adaptation was done by C. Seitz, and a brief discussion may

Fig. 13. Arbiter Module Circuit Diagram



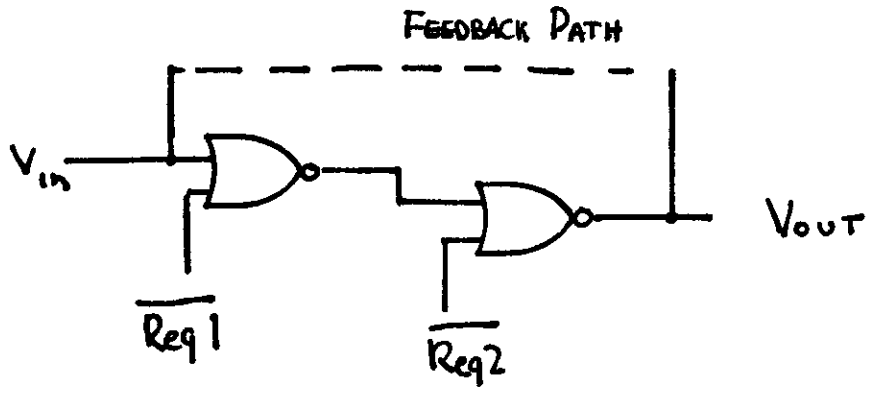
The transistors are of type 2N3648, which is a medium-gain high-speed switching transistor. The base resistors are 2.2K, and the collector resistors are 1K.

---

be found in [7]. It can be divided into two major sections, the front end and the comparator. The front end is a simple SR flip-flop except that the inputs are normally high. When either input goes low while the other remains high, the corresponding output goes low. Thus, when the timing of the asynchronous requests are far enough apart the front end correctly performs the arbitration. The difficulty comes when the requests come at about the same time. Because of the inherent propagation delay in the gates, if both requests go low at about the same time then both outputs will begin to go high. But the output of one gate is connected to the input of the other gate. This can cause a variety of things to happen, but it should be clear that the outputs of the front end cannot be used for the outputs of the arbiter because they both can change (or at least begin to change) at the same time.

To understand what can occur in this situation it is necessary to look beyond the logical models of digital circuits and consider their electronic implementation. Figure 14 shows the front end redrawn as a combinational circuit with a feedback path. Ignoring the feedback path momentarily and assuming that

Fig. 14. Front End of Arbiter Redrawn

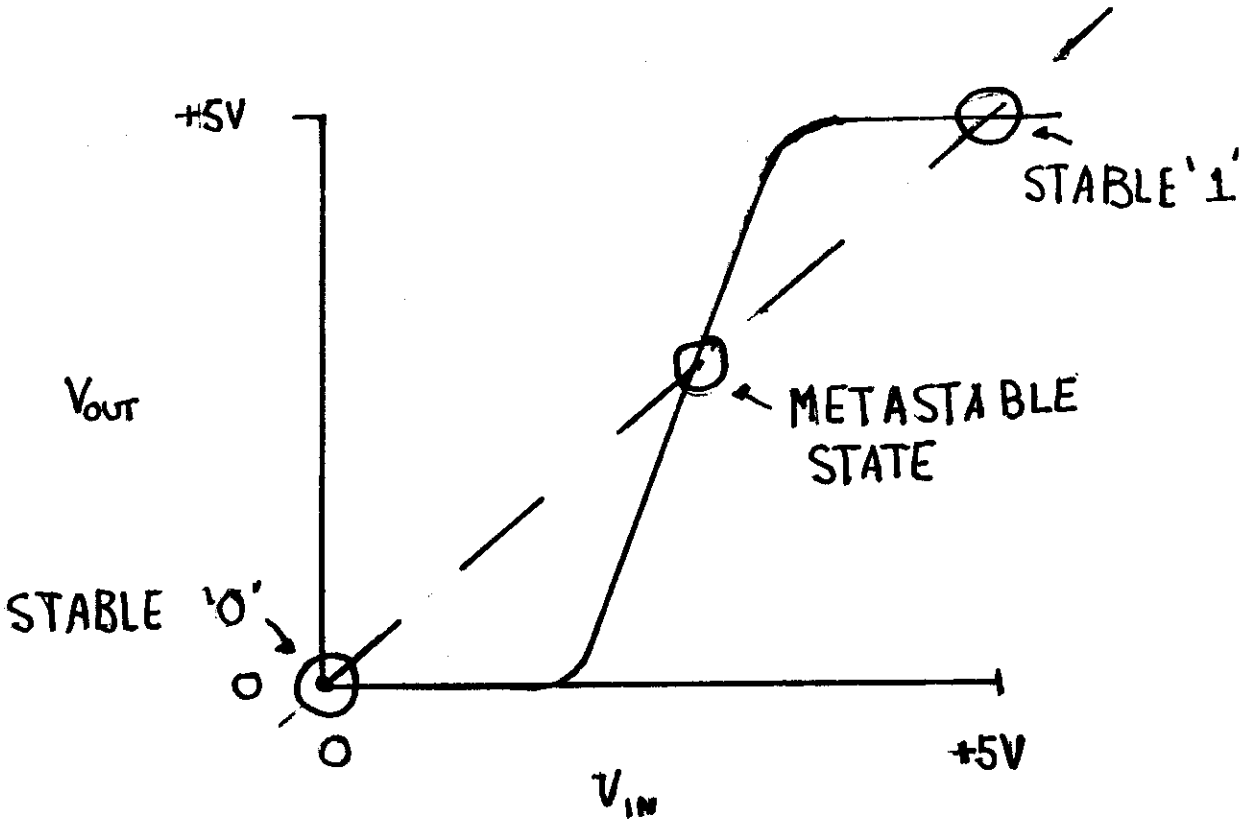


This diagram represents the front end as a combinational circuit with a feedback loop.

---

both requests are low, the path from Vin to Vout would have a transfer characteristic similar to two cascaded inverters, which has the approximate shape of the solid curve on the graph of figure 15. The specific nature of this transfer function is dependent on the specific technologies used to implement the logic gates and is not important to this discussion. Connecting the feedback path adds the constraint that Vin = Vout, which is shown as the dashed line in figure 15. The intersections of the two graphs are the only possible voltages that satisfy both constraints. The anomaly that occurs is that there are three possible voltages, one of which has no meaning in the domain of logic circuits. This point represents an unstable equilibrium, as a small voltage change in either direction due to noise, etc. is likely to cause the voltage to change to one of the other, stable values. This condition is termed the metastable state. The inevitability of noise in real systems will cause the output voltage to eventually change to a valid logic level, but it may take an arbitrarily long amount of time.

Fig. 15. Output of Front End



The solid curve is the transfer function of the combinational circuit, and the dashed line is the constraint added by the feedback path. The intersection of these graphs are the only possible stable values of the output.

The purpose of the comparator section is to detect the metastable condition in the front end and withhold any grant signals until the condition no longer exists. It accomplishes this by comparing the output voltage of the two gates, and asserting the grant signal only when one voltage is more than V volts greater than the other voltage. Choosing the value of V large enough will cause the output not to change until it is certain the front end cannot return to a metastable state. In this circuit the two transistors serve as comparators. When the output voltage of one gate is sufficiently greater than that of the other, then the transistor whose base is connected through a resistor to that gate will turn on and saturate, causing the collector voltage to drop from five volts to almost zero. The voltage differential required to activate the

grant output is approximately 1.2 volts. This value can be increased by .7 volts by adding diodes between the emitters and the gate outputs, as shown in figure 16. Adding more than one diode will raise the low output voltage of the arbiter beyond the allowable limit so the maximum voltage differential is about 1.9 volts. When no metastable condition occurs the comparator will not prevent the arbiter from functioning properly.

2.4 The Multiplexor Module

The Multiplexor module is responsible for switching the data from the input ports to the appropriate output ports, as indicated by the grant signals from the arbiters. It also relays the ready and acknowledge signals from the Master modules to the output ports and generates the detach signals to each Master module. The timing constraint is that all control signals must obey the reset signaling protocol.

This circuit can be implemented entirely with combinational logic elements, which will aid in the testing of this module. The circuit diagram is shown in figure 17. As can be seen the upper and lower

---

Fig. 16. Modified Arbiter Circuit

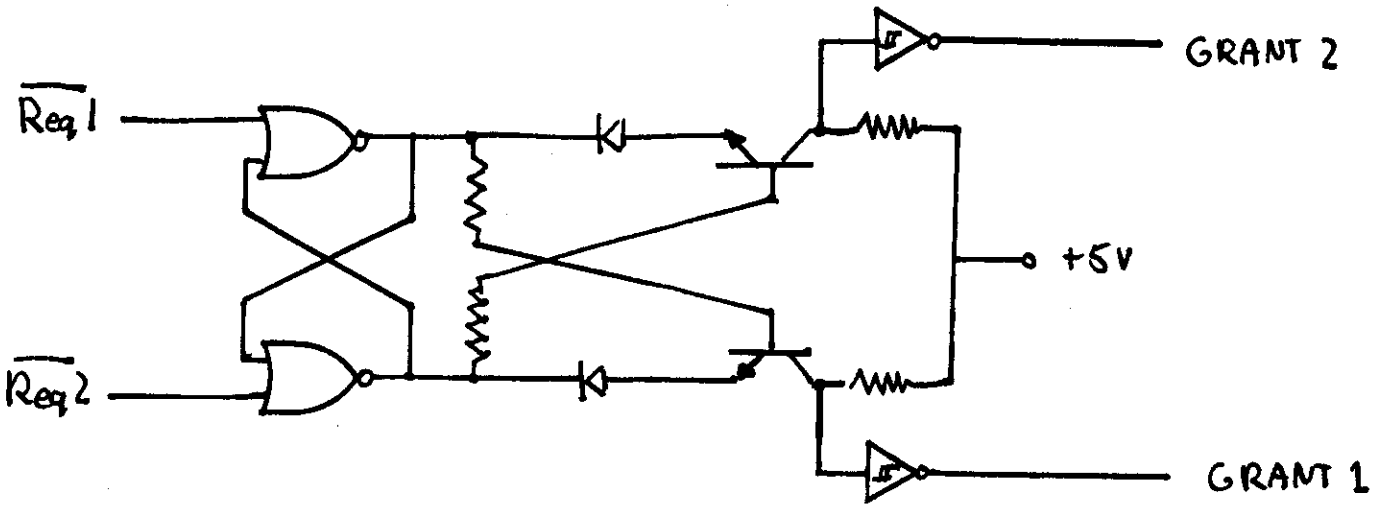
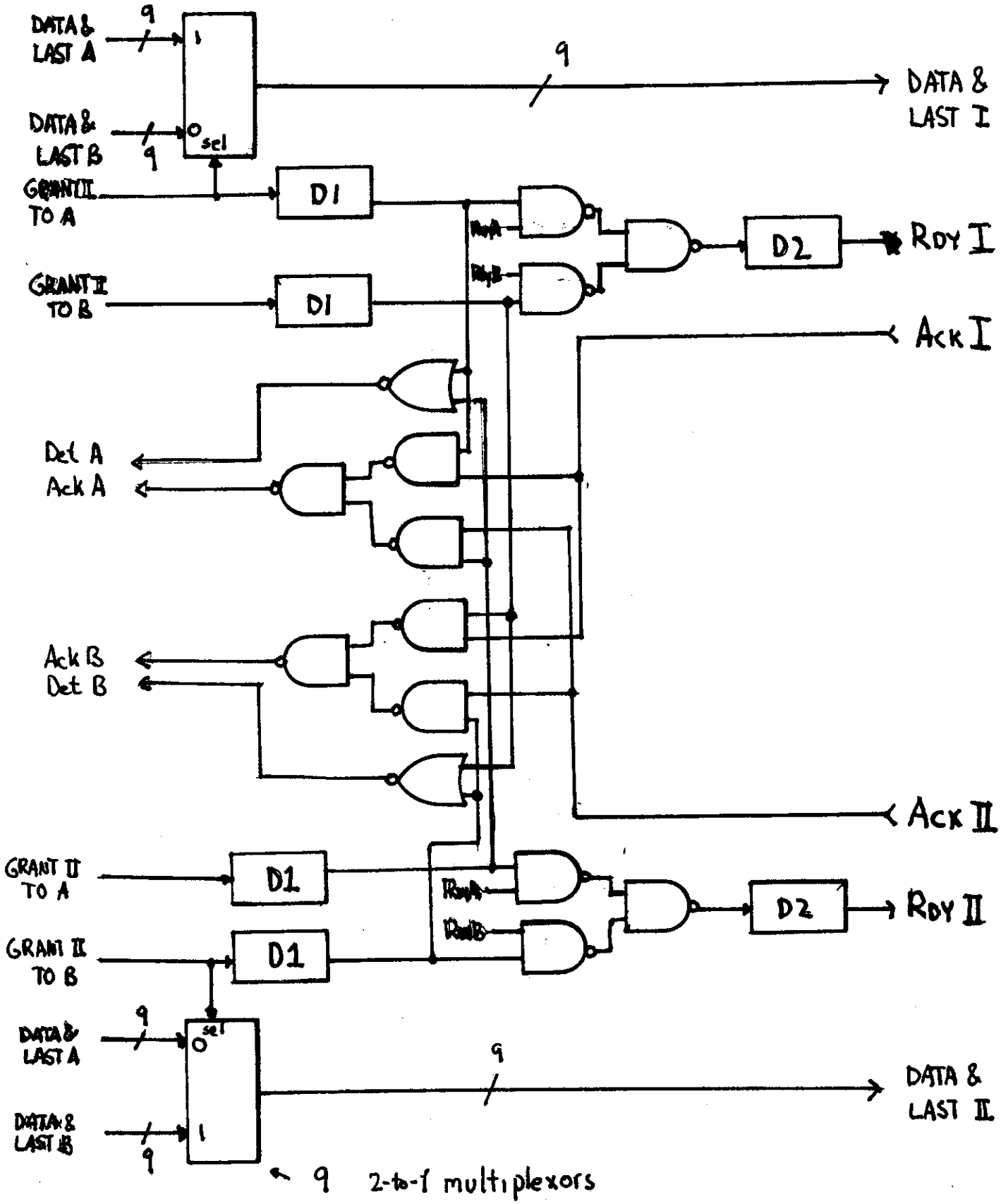


Fig. 17. Multiplexor Module Circuit Diagram



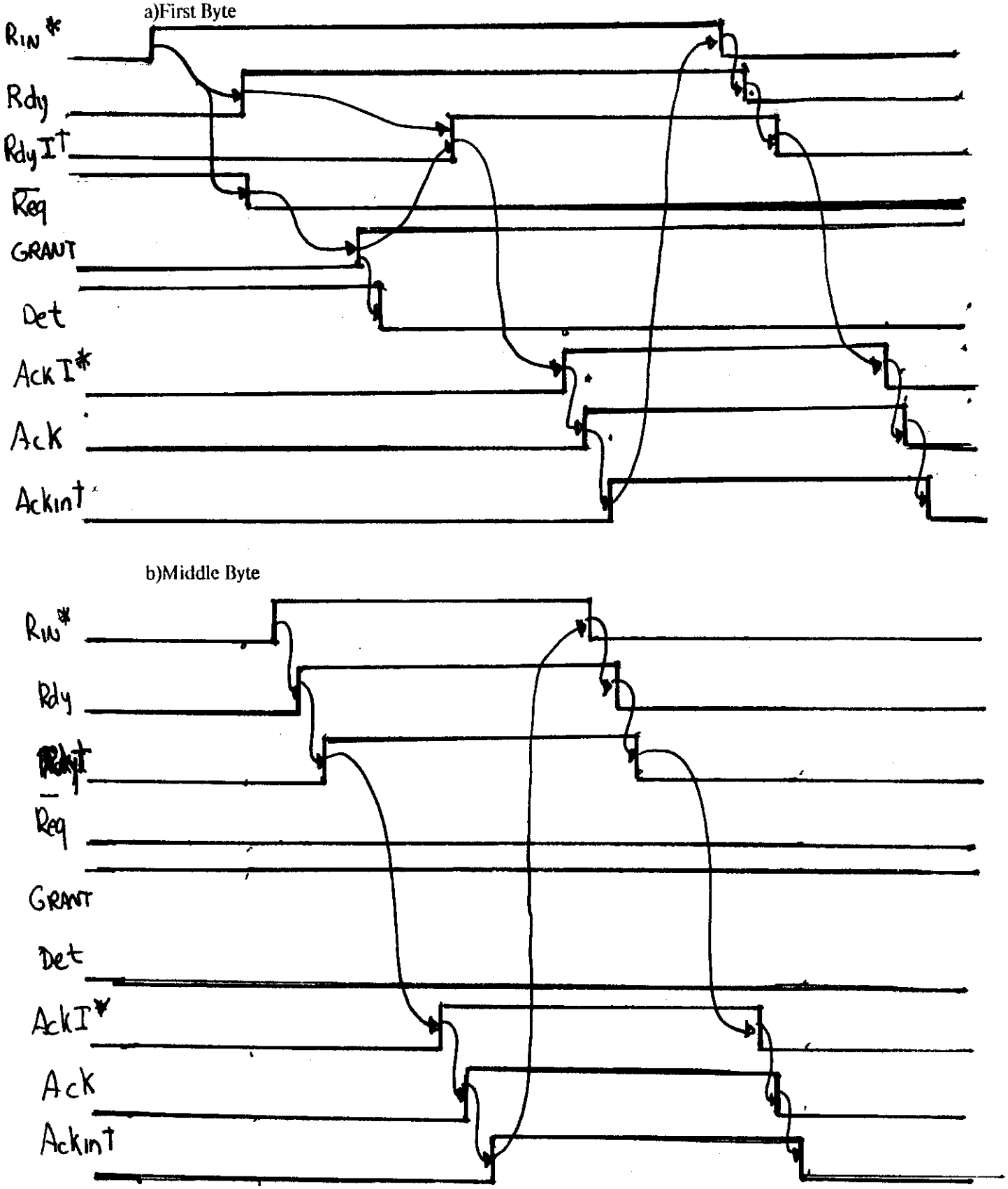
portions of this circuit are nearly symmetrical. For this reason this discussion will center around half of the circuit. The two-to-one multiplexors are implemented with 74LS157's, which contain four such units per chip. The grant signals come from the arbiter and are hence assumed to not both be high at the same time. The data multiplexor select inputs are connected to one of the grant signals, with the corresponding input port's data lines connected to the one inputs of the multiplexors. The other input port's data lines are connected to the zero inputs of the multiplexor and are hence always connected to the output port when the other grant signal is low. This does not matter, because the control signal paths from the input ports are not connected unless the appropriate grant signal is high. The delays labeled D1 are added to insure that the data input lines from the input port are connected to those on the output port before the grant signal is given to the rest of the circuit. The delays labeled D2 are added to the data on the outputs of the multiplexor to be valid for some set-up time before the corresponding ready signal goes high. The acknowledge signals are relayed to the Master module that has the corresponding grant signal asserted. The detach signal to a Master module is high only if both of the associated grant signals are low.

2.5 Top Level Operation of the Router

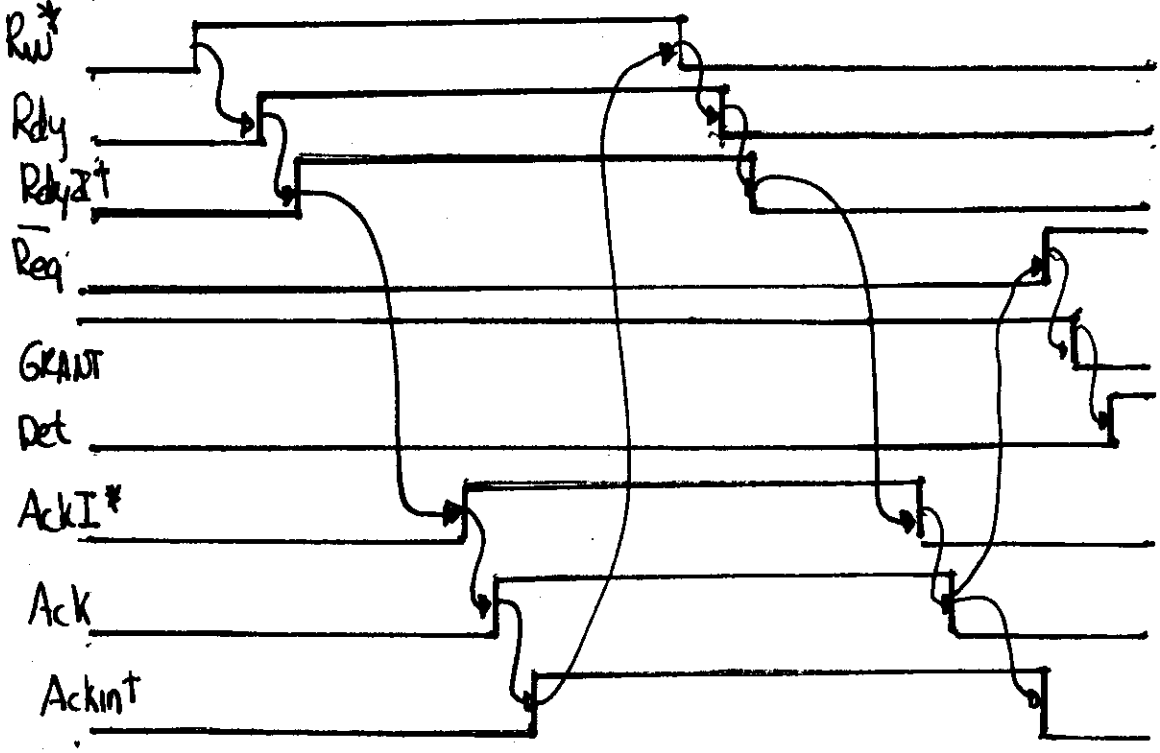
With a detailed understanding of the modules that make up the router it is possible to understand the overall operation of the router. Figure 18 shows the timing diagrams for the top level signals in the router and at the input and output ports. For convenience only the signals for one input port and one output port is shown. Those intervals marked outside interface or arbiter are only typical values and can in practice be arbitrarily long.

When a ready signal for the first byte of a packet arrives, the Master module issues a request for an output port to an arbiter, and presents the ready signal to the Multiplexor module. When the request is granted by the arbiter, the Multiplexor sends the ready to the designated output port. The Acknowledge and Ready signals are relayed during all intermediate bytes. On the last byte of the packet the control

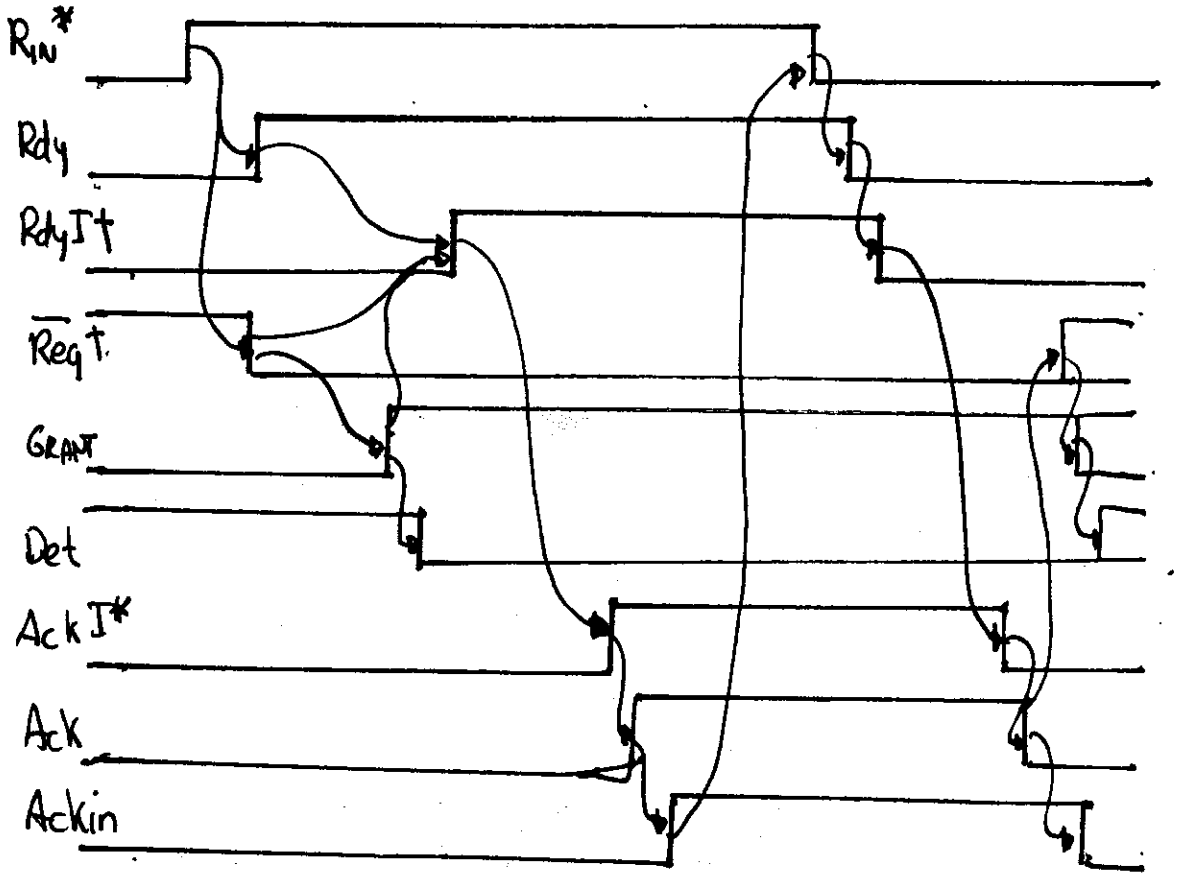
Fig. 18. Multiplexor Module Timing Diagram



c)Last Byte



d)Single Byte Packet



signals are relayed as before, except that Ackin is held high until the Master module has been reset to its initial state. Then the request is withdrawn, the grant signal goes low, and detach goes high. The timing for a single byte packet is similar.

From this discussion it should be clear that the design is capable of routing packets from both inputs to both outputs, providing the necessary arbitration, and allowing maximal concurrency. The next section discusses the issues involved with testing and debugging the modules of the router and the router itself. Information regarding the construction of a prototype router can be found in Appendix I.

3. Testing and Debugging Techniques

The choice of making the router an asynchronous self-timed system has significant impact on the nature of the testing required and this impact cannot be overemphasized. This choice implies, among other things, that the signals from the outside can arrive at any time. The router must be able to respond properly whenever they arrive. Clearly a brute force method that would test all of the possible timings cannot be effected since there are an infinite number of them. A more generalized technique is required.

The approach taken was to exploit the modularity of the design. Thus, it was necessary to design careful tests for each module, and then tests for the overall operation of the router from a modular viewpoint.

First to be presented are descriptions of the tests [to be] used for the testing of each module, along with reasoning for why the tests are useful and what issues remain unanswered. Following that is a discussion of the tests for the overall operation of the router. Appendix II discusses the hardware required to perform the tests on a prototype model.

3.1 Testing the FIFO Module

The FIFO module has two interfaces to the outside. These are the input interface and the output interface, both of which consist of 9 data lines and the corresponding Ready and Acknowledge control lines. The input section stores the data on the input data lines in the buffer when it sees a ready. When there is valid data in the buffer the output section generates a ready signal. At the end of the protocol the data is removed from buffer, and if the buffer is still non-empty another ready is generated.

The main element in the FIFO module is the memory chip, which implements the FIFO buffer without the necessary reset signalling interfaces. The testing described in this section deals with testing

the implementation of the interfaces. Testing the memory chip is of secondary importance.

To test the FIFO interfaces it is necessary to check that the protocol is always observed. To test the input, data should be written into the FIFO by presenting data and raising the ready signal. The acknowledge signal should go high and then the ready can be lowered. After the ready is lowered, the acknowledge should go low. After 16 data words have been stored, the acknowledge should remain high, indicating the buffer is full. Removing a word from the buffer (see next paragraph) should cause the acknowledge to go low.

Sometime after the first word is stored in the buffer the ready on the output interface should go high and the data written should appear on the output data lines. When the acknowledge signal is presented the ready signal should go low. Lowering acknowledge should cause the data outputs to change. If there is another word in the buffer then ready should again go high. If there are no more words then ready should remain low. During this test it should be verified that the data stored in the buffer is not altered.

The final test is to verify that data can be stored and retrieved from the buffer concurrently. No such limitation should exist unless the memory chip is incapable of this action.

The vulnerability of this implementation is the FIFO chip. If the data stored is altered, if concurrent input and output cannot take place, or if the reset signalling interface cannot be constructed then some alternate buffering technique will be required.

3.2 Testing the Master Module

The Master modules are simple sequential state machines and rely heavily on the correct use of reset signalling protocol by the outside interfaces. Because of the complexity of the Master module it is necessary to consider several different types of tests.

The first test can be described as a state transition and output generation test. This test involves removing the delay elements from the circuit and leave the feedback paths open. Logic levels can be applied to these lines so that the state variables can be set like an input. By setting the inputs and state variable inputs to certain values it is possible to observe what actions will occur, without these actions causing additional changes.

Figure 19 presents a testing chart used in this method. The first columns list the present state and the current value of the inputs. The middle columns are the expected output values. The last columns indicates the change to be introduced, and the expected results of that change. These results were obtained from an analysis of the circuit diagram, and can be either a state change or a change in output. If both are listed, then the change in output does not occur until after the state change. The table lists valid states and actions that can occur during the normal operation of the router. Conditions not in the table should not occur during normal operation of the router. Reasons why these actions are illegal include violation of reset signalling protocol or erroneous signals from other modules.

The idea of this test is to input the values and state listed in the chart to the Master module, and then cause the indicated change to occur. The actions caused by the changes can then be observed and compared to the expected results. If they are different then investigation into the implemented circuit is required. This step is completed for every entry in the chart. When the actual results match the expected results in all cases the test is complete.

Fig. 19. Testing Chart for Master Module, First Test

STATE	INPUTS Y1	Y2	Y3	D	Det	Rw	L	Ack	OUTPUTS Ackin	Rdy	ReqI	ReqII	INPUT CHAGE	ANTICIPATED RESULT
I	0	0	0	x	0	0	x	0	0	0	1	1	Det 0→1	No Change
I	0	0	0	x	0	1	0	0	0	0	1	1	Det 0→1	STATE I→F
I	0	0	0	0	0	1	1	0	0	0	1	1	Det 0→1	STATE I→A′
I	0	0	0	1	0	1	1	0	0	0	1	1	Det 0→1	STATE I→B′
I	0	0	0	x	0	0	x	0	0	0	1	1	Rw 0→1	No Change
I	0	0	0	x	1	0	0	0	0	0	1	1	Rw 0→1	STATE I→F
I	0	0	0	0	1	0	1	0	0	0	1	1	Rw 0→1	STATE I→A′
I	0	0	0	1	1	0	1	0	0	0	1	1	Rw 0→1	STATE I→B′
F	1	0	0	0	1	1	0	0	0	0	1	1	none	STATE F→A
F	1	0	0	1	1	1	0	0	0	0	1	1	none	STATE F→B
A B	1→0	0→1	0→1	x	x	0	0	0	0	0	0→1	1→0	Rw 0→1	Rdy 0→1
A	1	1	0→1	x	x	0	1	0	0	0	0→1	1	Rw 0→1	STATE A→A′
B	1	0	1	x	x	0	1	0	0	0	1	0	Rw 0→1	STATE B→B′
A B	1→0	0→1	0→1	x	x	1	0	0→1	0	1	0→1	1→0	Ack 0→1	Ackin 0→1
A B	1→0	0→1	0→1	x	x	1	0	1	1	1	0→1	1→0	Rw 1→0	Rdy 1→0
A B	1→0	0→1	0→1	x	x	1	x	1	1	0	0→1	1→0	Ack 1→0	Ackin 1→0
A′ B′	0→1	1→0	0→1	x	x	1	1	0	0	1	0→1	1→0	Ack 0→1	Ackin 0→1
A′ B′	0→1	1→0	1→0	x	x	1	1	0	1	1	1→0	0→1	Rw 1→0	Rdy 1→0
A′	0	1	0	x	x	0	x	1	1	0	1	0	Ack 1→0	STATE A′→I
B′	0	0	1	x	x	0	x	1	1	0	1	0	Ack 1→0	STATE B′→I

This test checks the combinational aspect of the circuit. It shows the machine will act properly when it is in a given state and a certain input change occurs. The effects of state variable changes on the operation of the circuit has not been carefully checked. In fact, the previous test did not introduce the state variable changes at all, except by manual changes by the tester. The next test requires that the delay lines be reinserted into the circuit so that the state variable changes are introduced by the circuit itself. This will cause the intermediate states (those that immediately cause another state change) to change too quickly to be observed. Other states can be maintained for an arbitrarily long amount of time since they do not change except on the change of input values.

Figure 20 presents a testing chart for the second test. As mentioned earlier the delays are reconnected in the feedback path. The remaining inputs are controlled as before. Since creating certain error conditions in this case is difficult this test is less useful for error analysis. The test procedure is similar to the first case. This test checks the sequential aspects of the circuit.

The major weakness of these tests lies in the restricted nature of the timing signals. While they do, in some sense, represent the entire range of valid timings, in general the timings will occur more quickly than could be produced by manual action. A faster timing sequence could possibly cause an error that would go undetected with the slower tests. A more complex, automatic or computer controlled tester would be required to perform a higher speed test. However it is not clear how such a device would be used to increase the level of confidence in correct in the router. For this reason this option was not explored.

Fig. 20. Testing Chart for Master Module, Second Test

STATE	INPUTS					OUTPUTS					INPUT CHANGE	ANTICIPATED RESULTS
	D	Del	R_{IN}	L	Ack	A_{CKIN}	Rdy	ReqI	ReqII		
I	0	0	1	0	0	0	0	1	1	Del 0→1	RDY 0→1 STATE I→A, REQI 1→0
I	1	0	1	0	0	0	0	1	1	Del 0→1	RDY 0→1 STATE I→B, REQII 1→0
I	0	0	1	1	0	0	0	1	1	Del 0→1	RDY 0→1 STATE I→A', REQI 1→0
I	1	0	1	1	0	0	0	1	1	Del 0→1	RDY 0→1 STATE I→B', REQII 1→0
I	0	1	0	0	0	0	0	1	1	R_{IN} 0→1	RDY 0→1 STATE I→A, REQI 1→0
I	1	1	0	0	0	0	0	1	1	R_{IN} 0→1	RDY 0→1 STATE I→B REQII 1→0
I	0	1	0	1	1	0	0	1	1	R_{IN} 0→1	RDY 0→1 STATE I→A' REQI 1→0
I	1	1	0	1	1	0	0	1	1	R_{IN} 0→1	RDY 0→1 STATE I→B' REQII 1→0
A	X	X	0	1	0	0	0	0	1	R_{IN} 0→1	STATE A→A' RDY 0→1
B	X	X	0	1	0	0	0	1	0	R_{IN} 0→1	STATE B→B' RDY 0→1
A'	X	X	0	1	1	1	0	0	1	Ack 1→0	STATE A'→I ACKIN 1→0
B	X	X	0	1	1	1	0	1	0	Ack 1→0	STATE B'→I ACKIN 1→0

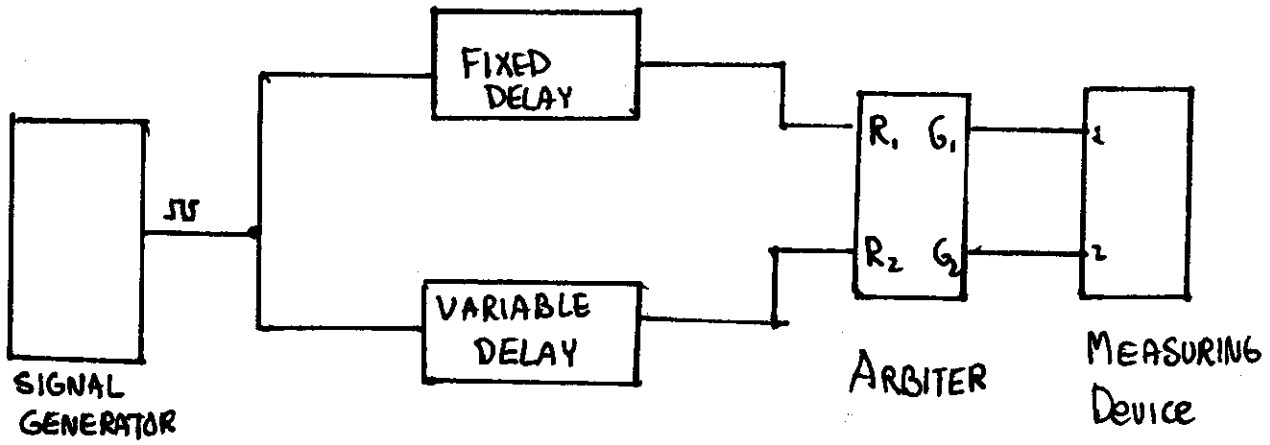
3.3 Testing the Arbiter Module

The question of how to verify if an arbiter works properly is a problem of current research. It seems that almost any arbiter will work when the request signals are spaced far enough apart; the real test is when the signals come at about the same time. In this arbiter, when signals come close together it is likely that the front end will be driven to a metastable condition. It is necessary to check that such a condition does not cause the outputs to change when a grant should not be given.

The method used to test the circuit is based on the technique used at Washington University [1]. A block diagram of the setup used is shown in figure 21. The interesting aspect of the experiment is the variable delay line. The fixed delay is chosen to the value half-way between the upper and lower limits of the variable delay. By connecting the output of the signal generator to both delays and adjusting the variable delay, the timing of the two request signals can be adjusted to arbitrarily close together, until the front end is driven into a metastable state. It is difficult to measure the time between the two signals but

---

Fig. 21. Test Setup for Testing the Arbiter Module



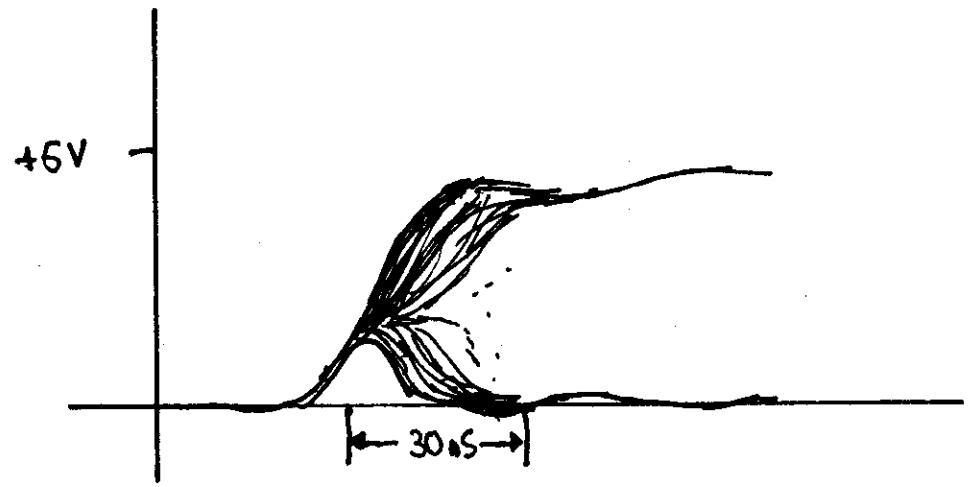
The signal generator gives the same signal to both delays. A measuring device such as an oscilloscope is used to observe the output of the Arbiter.

it is relatively easy to determine if the signals drive the front end of the arbiter into a metastable condition, and this is all that is required.

The delay lines were implemented using lengths of coaxial line; the fixed delay is a fixed length of cable and the variable length delay is an adjustable coaxial line, commonly called a trombone line. The length of the trombone line is changed by adjusting a sliding section to the desired length.

It proved easier than anticipated to drive the front end of the arbiter into a metastable state. A typical oscilloscope waveform of this condition is shown if figure 22. The difference in output voltages of the two gates was measured to be less than .2 volts. The voltage differential required to issue the grant signal was measured to be about 1.2 volts with no additional diodes (see previous section, discussion of arbiter circuit). It appeared that when such a differential occurred the front end did not return to a metastable state Adding diodes changed this voltage to about 1.9 volts as expected, but the circuit appeared to function no differently so it was concluded that they were unnecessary.

Fig. 22. A Metastable Output Waveform



The grant signal of the arbiter did not change until after the voltage differential of the outputs of the two gates was greater than 1.2 volts. A small glitch in the grant signal of the losing request (normally high) was detected, but it never exceeded +.05 volts or -.025 volts from the normal value. This deviation is small enough that it is not likely to cause problems during normal TTL operation. To be safe, Schmitt Trigger buffering is used to interface this signal with the the rest of the router. With this additional protection it was felt this arbiter could successfully serve the requirements of this design.

There was another test considered that also merits mention here. This test involves the construction of a special circuit that generates the two request pulses and adjusts the timing difference between them based on the eventual winner of the previous experiment. The difference between the two signals would very quickly approach that point which drives the front end into the worst metastable condition. The test does not use a different technique to determine if the grant signals change when they should not; it just causes the metastable condition to occur in a different manner. For this reason it was felt that this additional test was not really necessary.

3.4 Testing the Multiplexor Module

The Multiplexor module is purely combinational circuitry so some aspects of the testing are simplified. The timing considerations must still be handled carefully.

The truth table for the Multiplexor module is given in figure 23. The first test is a functionality test. For each entry in the truth table the input signals should be set as specified. Each output of the Multiplexor module should be measured and checked with the value in the truth table. Any deviation from the values in the table must be investigated by tracing signals through the circuit with the aid of the circuit diagram.

Fig. 23. Truth Table for Multiplexor Module

a) Single request cases

GRANT I to A	GRANT II to B	GRANT II to A	GRANT II to B	RdyA	RdyB	AckI	AckII	DetA	DetB	AckA	AckB	RdyI	RdyII
0	0	0	0	X	X	X	X	1	1	0	0	0	0
1	0	0	0	0	X	0	X	0	1	0	0	0	0
1	0	0	0	1	X	0	X	0	1	0	0	1	0
1	0	0	0	1	X	1	X	0	1	1	0	1	0
1	0	0	0	0	X	1	X	0	1	1	0	0	0
0	1	0	0	X	0	0	X	1	0	0	0	0	0
0	1	0	0	X	1	0	X	1	0	0	0	1	0
0	1	0	0	X	1	1	X	1	0	0	1	1	0
0	1	0	0	X	0	1	X	1	0	0	1	0	0
0	0	1	0	0	X	X	0	0	1	0	0	0	0
0	0	1	0	1	X	X	0	0	1	0	0	0	1
0	0	1	0	1	X	X	1	0	1	1	0	0	1
0	0	1	0	0	X	X	1	0	1	1	0	0	0
0	0	0	1	X	0	X	0	1	0	0	0	0	0
0	0	0	1	X	1	X	0	1	0	0	0	0	1
0	0	0	1	1	X	X	1	1	0	0	1	0	1
0	0	0	1	0	X	X	1	1	0	0	1	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0	0	1	0
1	0	0	1	0	1	0	0	0	0	0	0	0	1
1	0	0	1	1	1	0	0	0	0	0	0	0	1
1	0	0	1	0	0	1	0	0	0	1	0	0	0
1	0	0	1	1	0	1	0	0	0	1	0	1	0
1	0	0	1	0	1	1	0	0	0	1	0	0	1
1	0	0	1	1	1	1	0	0	0	1	0	1	1

b) Multiple request case

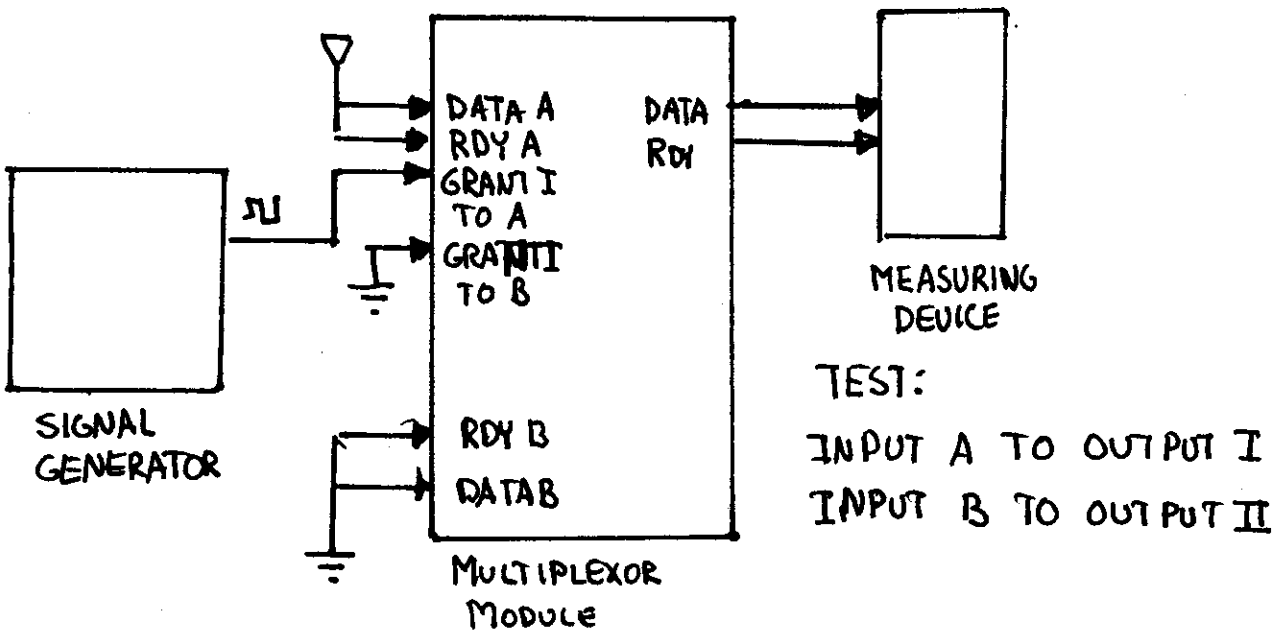
GRANT I to A	GRANT I to B	GRANT II to A	GRANT II to B	RDY A	RDY B	ACK I	ACK II	Det A	Det B	Ack A	Ack B	RDY I	RDY II
1	0	0	1	0	0	0	1	0	0	0	1	0	0
1	0	0	1	1	0	0	1	0	0	0	1	1	0
1	0	0	1	0	1	0	1	0	0	0	1	0	1
1	0	0	1	1	1	0	1	0	0	0	1	1	1
1	0	0	1	0	0	1	1	0	0	1	1	0	0
1	0	0	1	1	0	1	1	0	0	1	1	1	0
1	0	0	1	0	1	1	1	0	0	1	1	0	1
1	0	0	1	1	1	1	1	0	0	1	1	1	1
0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0	0	0	0	0	0	1
0	1	1	0	0	1	0	0	0	0	0	0	1	0
0	1	1	0	1	1	0	0	0	0	0	0	1	1
0	1	1	0	0	0	1	0	0	0	0	1	0	0
0	1	1	0	1	0	1	0	0	0	0	1	0	1
0	1	1	0	0	1	1	0	0	0	0	1	1	0
0	1	1	0	1	1	1	0	0	0	0	1	1	1
0	1	1	0	0	0	0	1	0	0	1	0	0	0
0	1	1	0	1	0	0	1	0	0	1	0	0	1
0	1	1	0	0	1	0	1	0	0	1	0	1	0
0	1	1	0	1	1	0	1	0	0	1	0	1	1
0	1	1	0	0	0	1	1	0	0	1	1	0	0
0	1	1	0	1	0	1	1	0	0	1	1	0	1
0	1	1	0	0	1	1	1	0	0	1	1	1	0
0	1	1	0	1	1	1	1	0	0	1	1	1	1

When the functionality test is complete, a test for the correct timing of signal propagation is required. As mentioned in the previous section, the delays are added to insure that the data at the output port is valid and stable before the ready signal to the output port is given. The first delay guarantees that the data path from the input port to the output port is connected before the grant signal is given to the rest of the Multiplexor module. The second delay is used to insure that the data is valid on the output port for a proper amount of set-up time before the ready is given. The first delay is applicable only during the first byte while the second delay is always active. For this reason two tests are required, one test for the first byte set-up time and one for the set-up time for other bytes. Since in this design the output port data paths are always connected to one of the input ports when not in use by the other (input A to output II, and Input B to output I), the first byte set-up time test is not necessary for those cases because the set-up time will obviously be large.

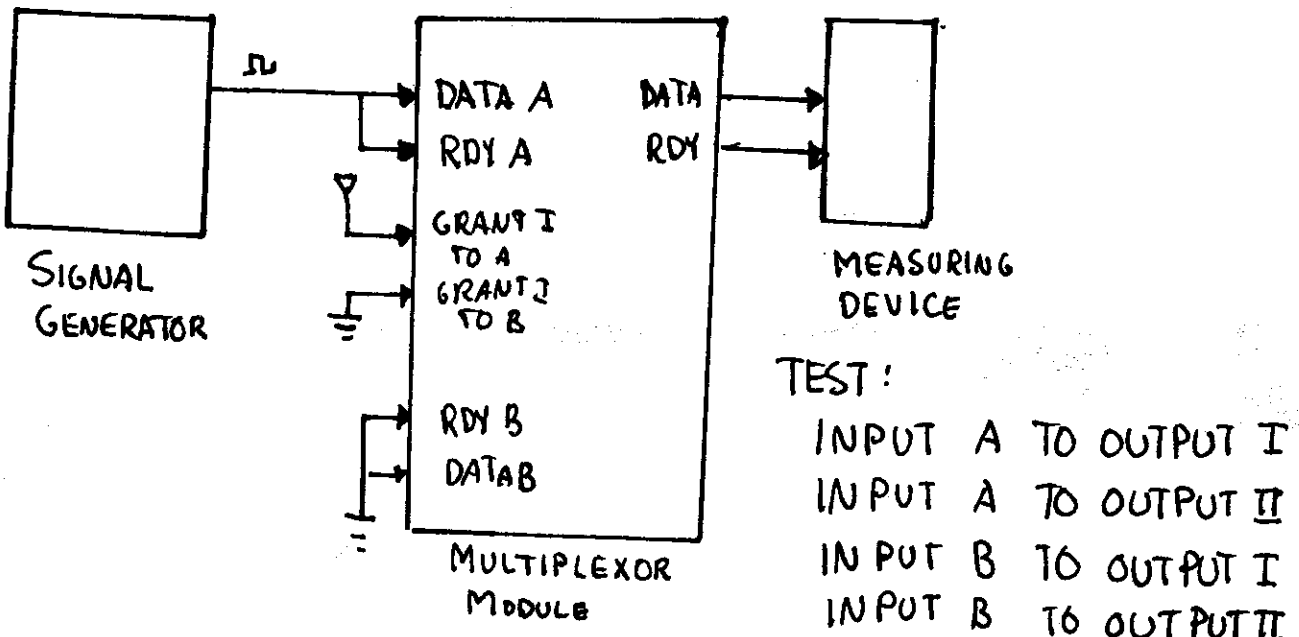
Both tests use the same basic experimental set-up, which is shown in figure 24a&b. During the transmission of the first byte, the ready and request signals from the Master module are issued at the same time. Since a delay exists in the arbiter the ready signal to the Multiplexor module will be one before the grant signal from the arbiter arrives. For this reason the ready input is tied high during this test. The data input lines are also held high while the corresponding data lines of the other input port are held low. The signal generator repeatedly creates pulses which are sent to the grant input. A pulse from the signal generator will eventually cause both the tied high data lines and the ready signal to go high. The oscilloscope is used to measure the timing difference between the two signals. This difference is the first byte set-up time.

The test of the set-up time for other bytes is similar. This time the grant signal is held high while the signal generator is connected to the ready and data inputs. The worst possible case, assuming correct use of the communication protocol, is that the data would arrive slightly before the ready signal, so this measurement actually provides a minimum possible setup time. The setup time is measured the same

Fig. 24. Multiplexor Module Timing Test Diagram



a)First Byte Set-up Time Test



b)Other Bytes Set-up Time Test

way as in the first byte experiment. This test must be used for all inputs to all outputs.

If the first byte setup time is too small but the other setup time is acceptable then the first delay must be increased. If the other setup time is too small then the second delay must be increased.

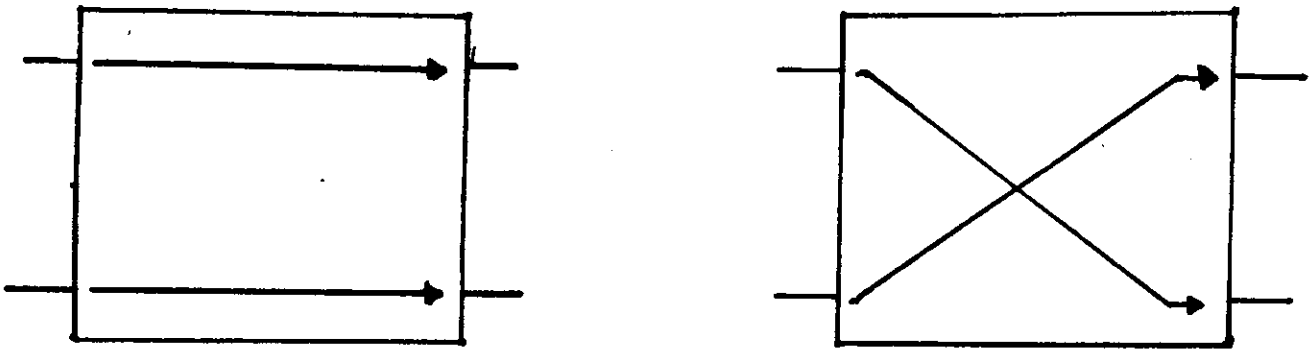
3.5 Testing the Overall Operation

There are two tests of the router as a whole unit. The first actually tests the router without the FIFO modules, which has the effect of making the input and output signal timings dependent on each other, and both are controlled by the human tester. The second test uses the same methods as the first, but the FIFO modules are included. This will isolate the input and output signals but will more resemble the actual operation of the circuit.

The object of the test is to determine if the router functions properly by simulating the transmission of packets from the input. This is accomplished by applying the appropriate control and data signals to the input ports, acknowledging receipt of the signals at the output, and verifying that the data in the packet is not altered. The methods of doing this is discussed in Appendix II.

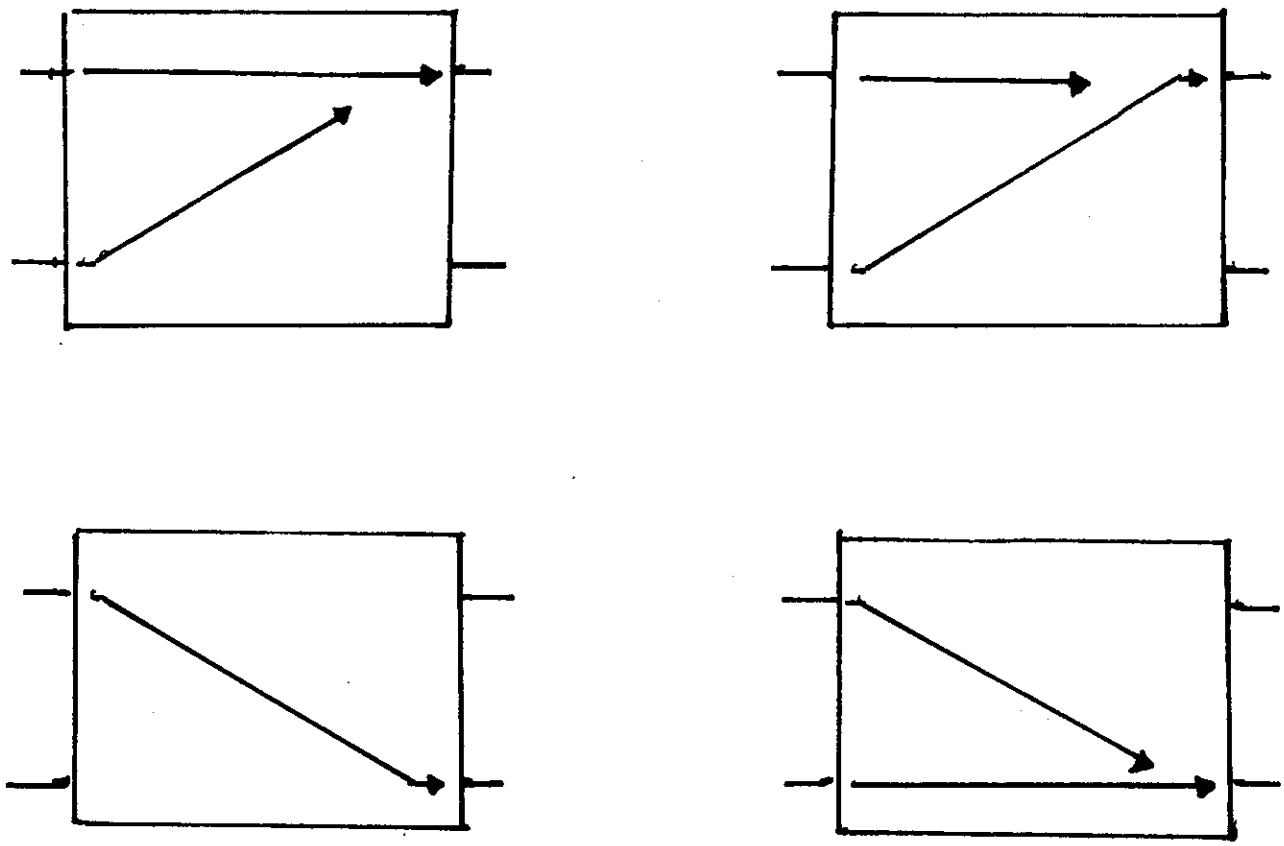
First it should be checked that both input ports can send both single and multibyte packets can be sent to both output ports when usage is uncontested. Then it should be verified that both possible concurrent transmission configurations shown in figure 25. Finally the four possible blocking conditions shown in figure 26 should be checked. This completes the step of the overall test of the router.

Fig. 25. Possible Concurent Configurations of Router



To have concurrent operation, the input ports must use different output ports.

Fig. 26. Blocked Configurations of the Router



When two inputs wish to use the same output one of them must wait.

4. Conclusions

In this concluding section the limits of the tests will be discussed, and the future directions will be explained.

As mentioned previously the tests seem to lack a real time aspect. That is, the test signal timings in general are slower than may normally appear. This seems to indicate that more may be required to test the Router. Perhaps even more important, however, is that the router has not yet been used in the construction of larger networks. It may also be necessary to perform closer studies on the nature of the timing of the router to understand better the nature of how the packets propagate through the networks, and how changing the size of the networks will affect the delay. Finally, it would then be beneficial to determine the nature of how much blocking in the network occurred as the traffic through the network increased.

It seems evident that this method of constructing a router would prove adequate. It may prove expensive, however, because despite the efforts to keep the number of parts down, a fair number of them are still necessary. With increasing capabilities and decreasing cost of LSI technology, this alternative should be examined carefully, especially for more later versions of the router.

Appendix I - A Prototype Implementation of a Router

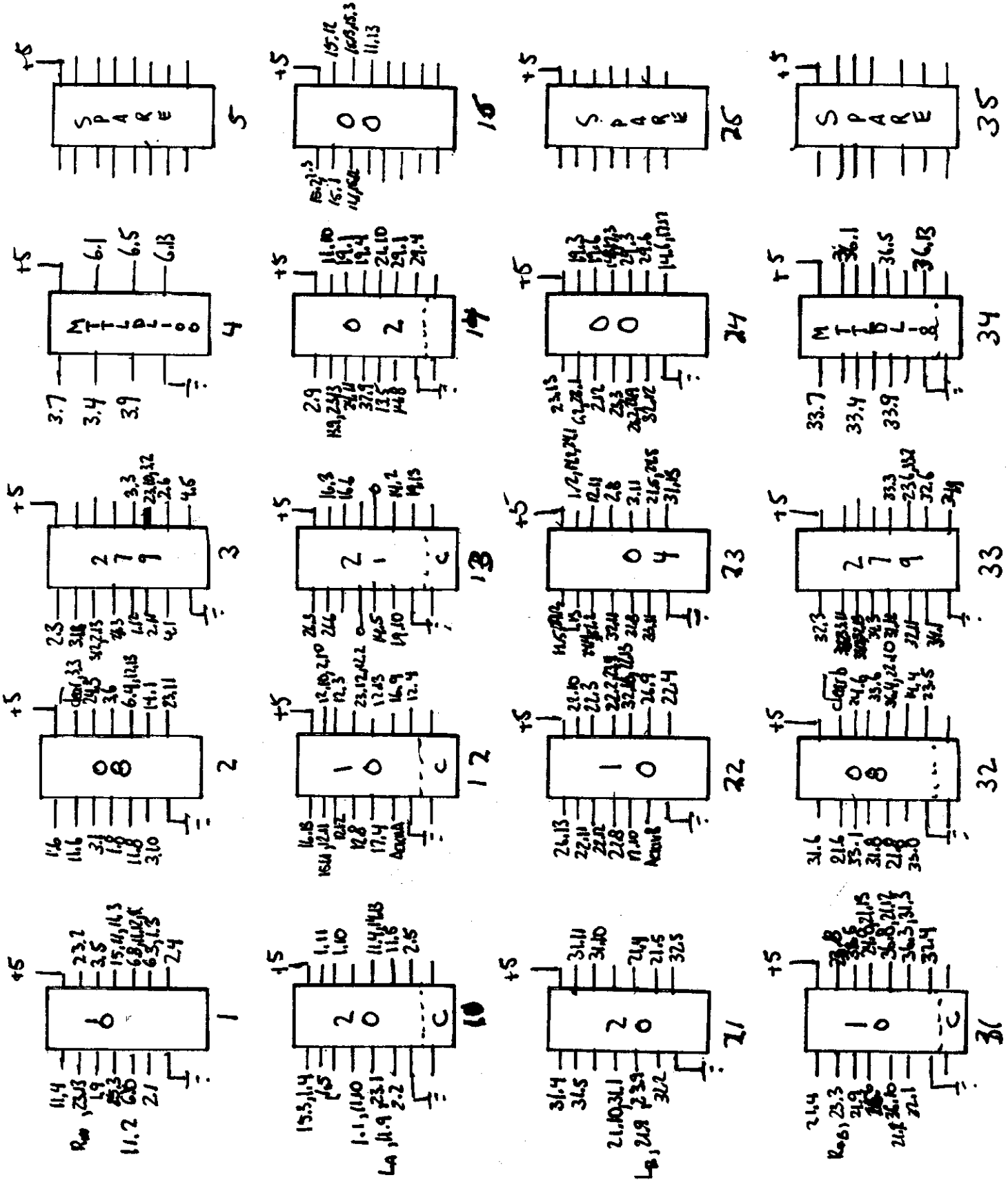
A prototype version of the router was implemented using vector board and wire wrap sockets. The discrete components are readily available, and the digital logic elements are the standard TEXAS INSTRUMENTS' 74LS series Integrated circuits. A complete parts list is given in figure 27. The only unusual elements are the delays. The delays in the feedback paths of the Master module were implemented using the Engineered Components Company (EC^2) MTTLDL-100's, which are precision one hundred nanosecond delay lines. The delays D1 in the Multiplexor module were implemented using extra data paths in the multiplexor chips (74LS157). These chips perform no arbitration, but merely delay the signals the proper amount of time. The delays D2 are implemented using a long chain of an even number of TTL inverters. The delay can be adjusted by adding or removing an even number of inverters to this chain. In a final version of the router it may be desirable to replace these two delays with more precise delays. These simple versions seem to function adequately for this initial version.

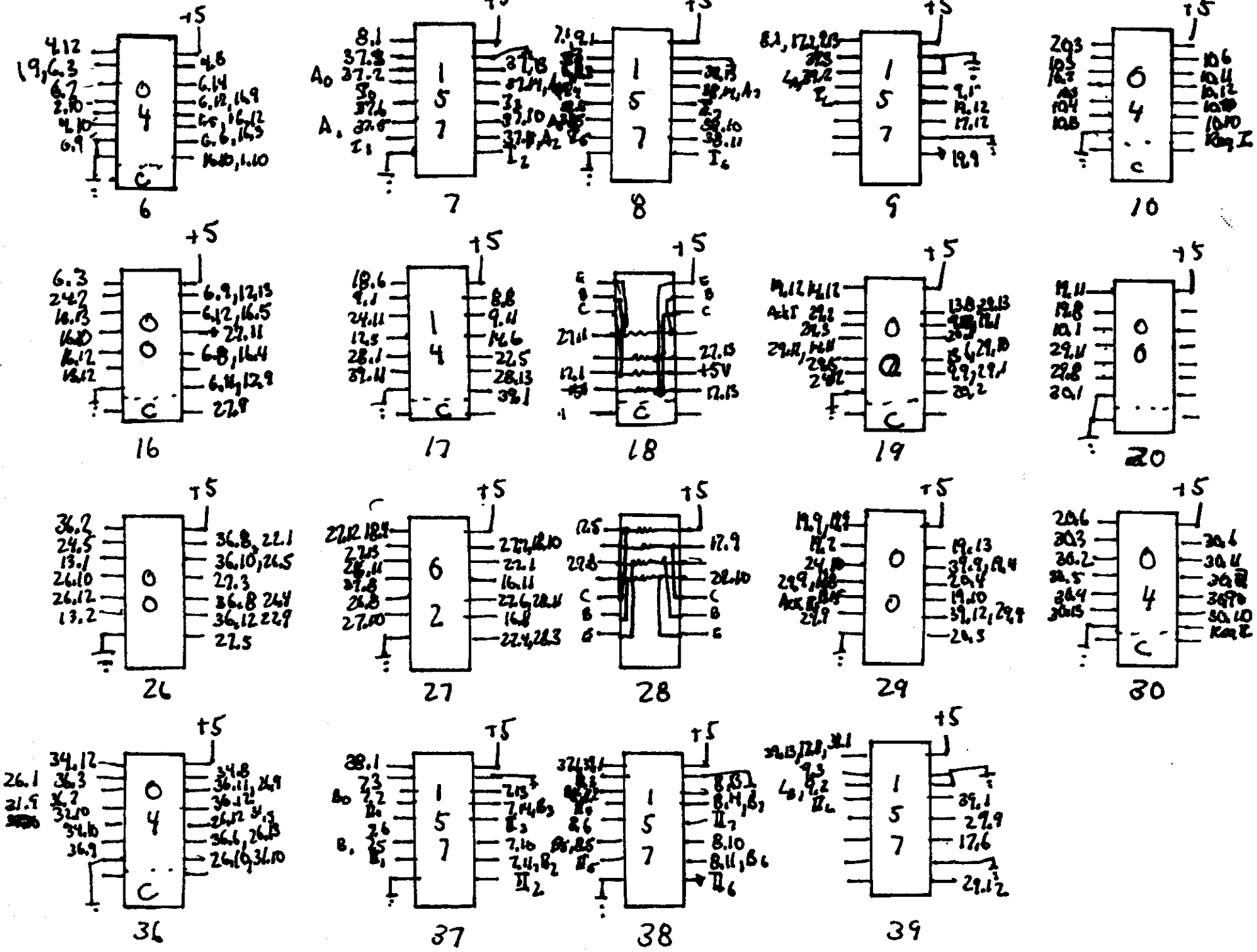
Figure 28 shows the circuit layout and the wiring chart for this prototype version. The circuit is arranged by modules whenever possible while maintaining a chip count The pair(s) of numbers associated with each pin indicates where that pin should be connected. The first number on the wiring chart is the socket number, and the second number is the pin number on that socket that the wire should be connected to. This method was used to help insure correct wiring.

Fig. 27. Parts List for Prototype Router

74LS00	6
74LS02	2
74LS04	4
74LS08	2
74LS10	4
74LS14	1
74LS20	2
74LS21	1
74LS157	6
74LS279	2
MTTLDL-100	2
1KΩ	6
2.2KΩ	4
2N3646	2

Fig. 28. Circuit Layout and Wiring Chart




+5
0 4
1 5 7
6
7
8
9
10
0 0
16
1 4
17
18
0 0
19
0 6
20
0 0
26
6 2
27
28
0 0
29
0 4
30
0 4
36
1 5 7
37
38
1 5 7
39

Appendix II - Hardware Required to Test the Router and its Modules

All of the tests required the ability to hold certain input signals at specific logic levels, generate pulses, and generate reset signalling protocols. To aid in the testing of the modules and the router itself a test module was designed that has all of the capabilities listed.

The module was designed with the end goal of being able to perform a functional test of the entire router so there are part of the module that will not be needed during all of the tests. The block diagram of the test module, which is shown in figure 29, shows this design decision; there are input sections for each input port of the router, and output sections for each output port.

An input section consists of a ready signal generator, which uses the corresponding acknowledge signal as an input, and connections for the Last Byte (L) and direction (D, actually, the least significant bit of the data). Figure 30 shows the circuit diagram for an input section. The push-button switch generates a (debounced) pulse which clocks the flip-flop, whose output is the ready signal. Ready changes on the command of the tester, but only when allowed by the protocol. The pulses from the debounce circuit can also be used directly when non-protocol pulses are required. The toggle switches make it easy to change the values of L and D during the testing. Also provided are extra pull-up resistors and ground connections for tying signals high or low. The LED indicates that an acknowledge signal is present at the input port, and hence clocking the flip-flop will cause ready to go low. This is not required since the values could be measured by the tester, but this facilitates the testing.

An output section provides an acknowledge signal generator which uses the ready as its input, and provides the capability of observing the data at the output port. The output section circuit diagram is shown in figure 31. The acknowledge flip-flop functions in an analogous way to the ready flip-flop in the input section. As in the case of the input section the LED's are optional, included only for the

Fig. 29. Tester/Router Interface

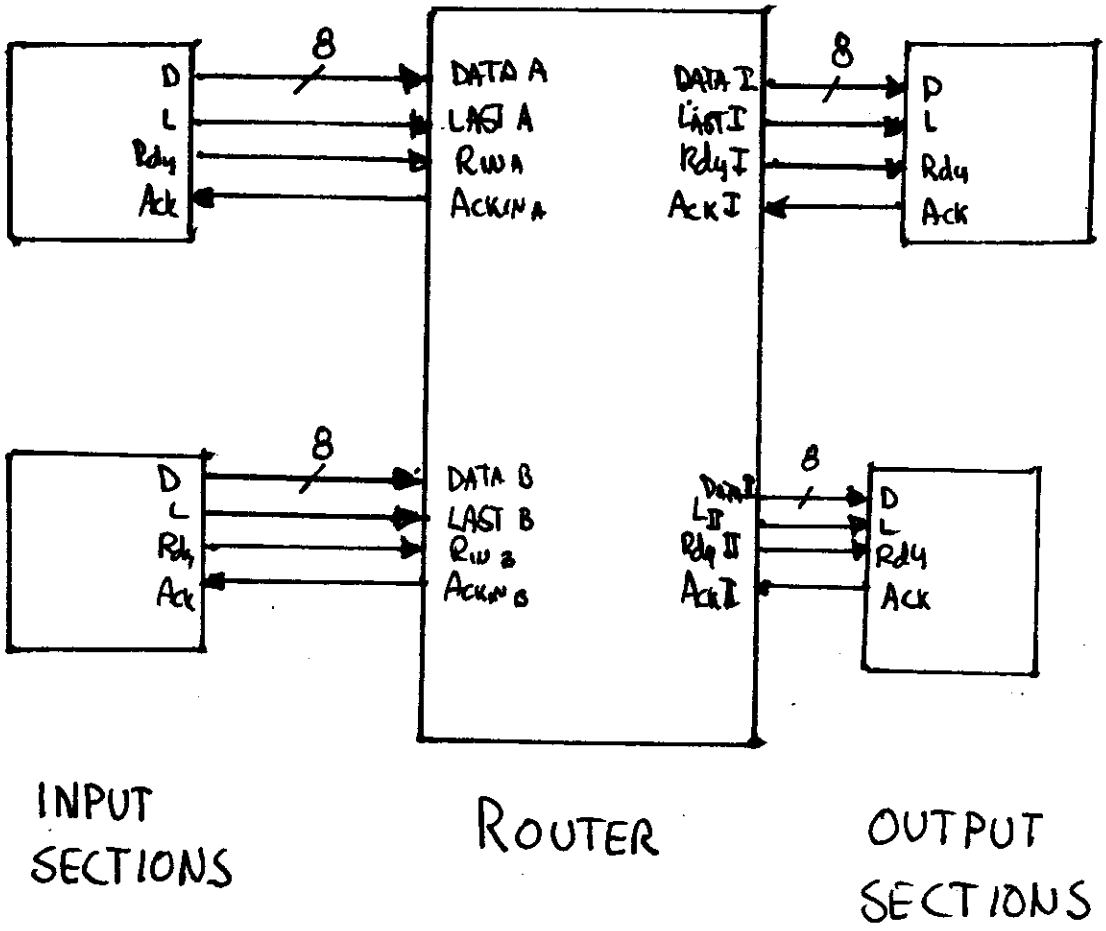


Fig. 30. Circuit Diagram for an Input Section of a Test Module

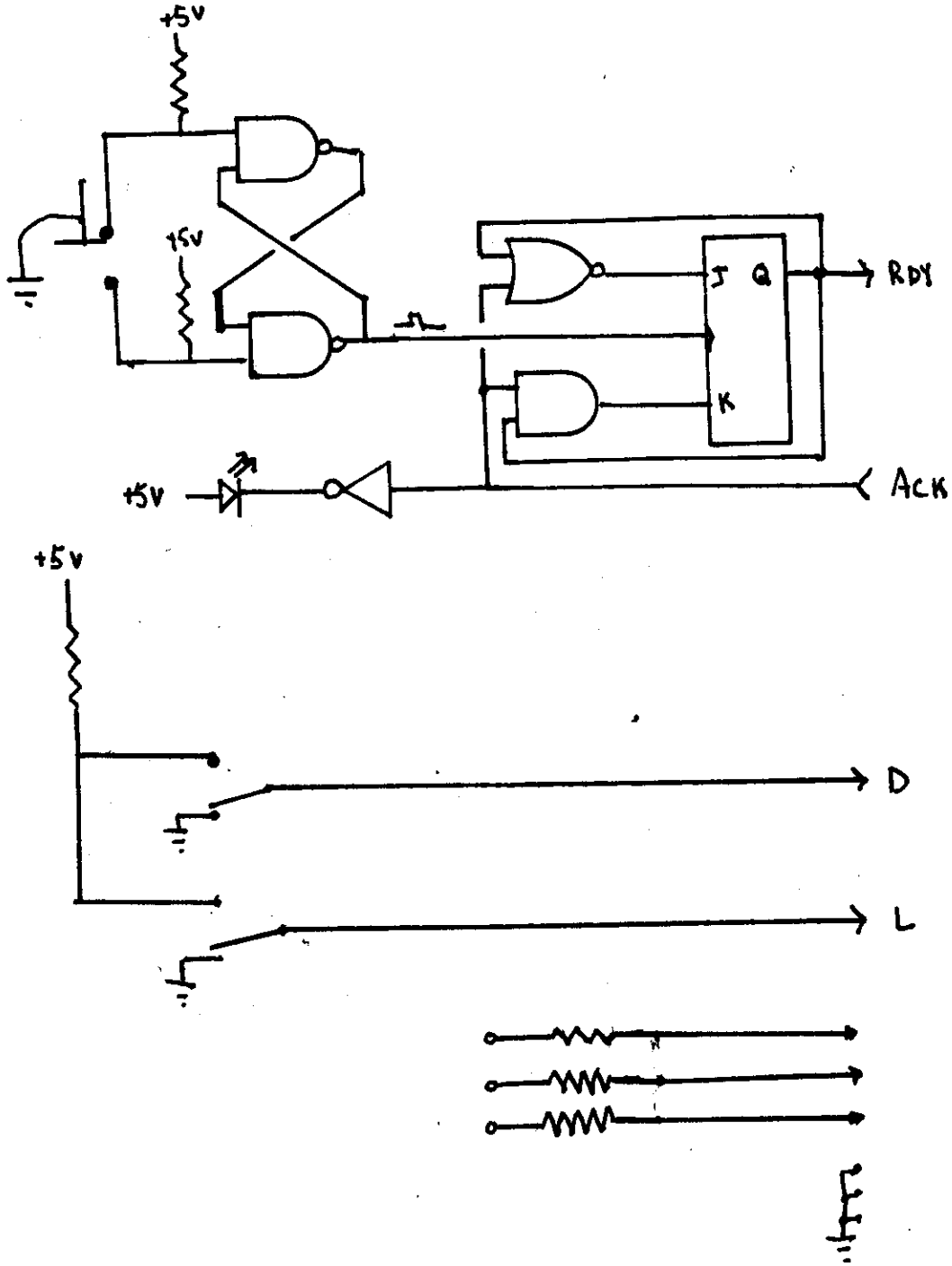
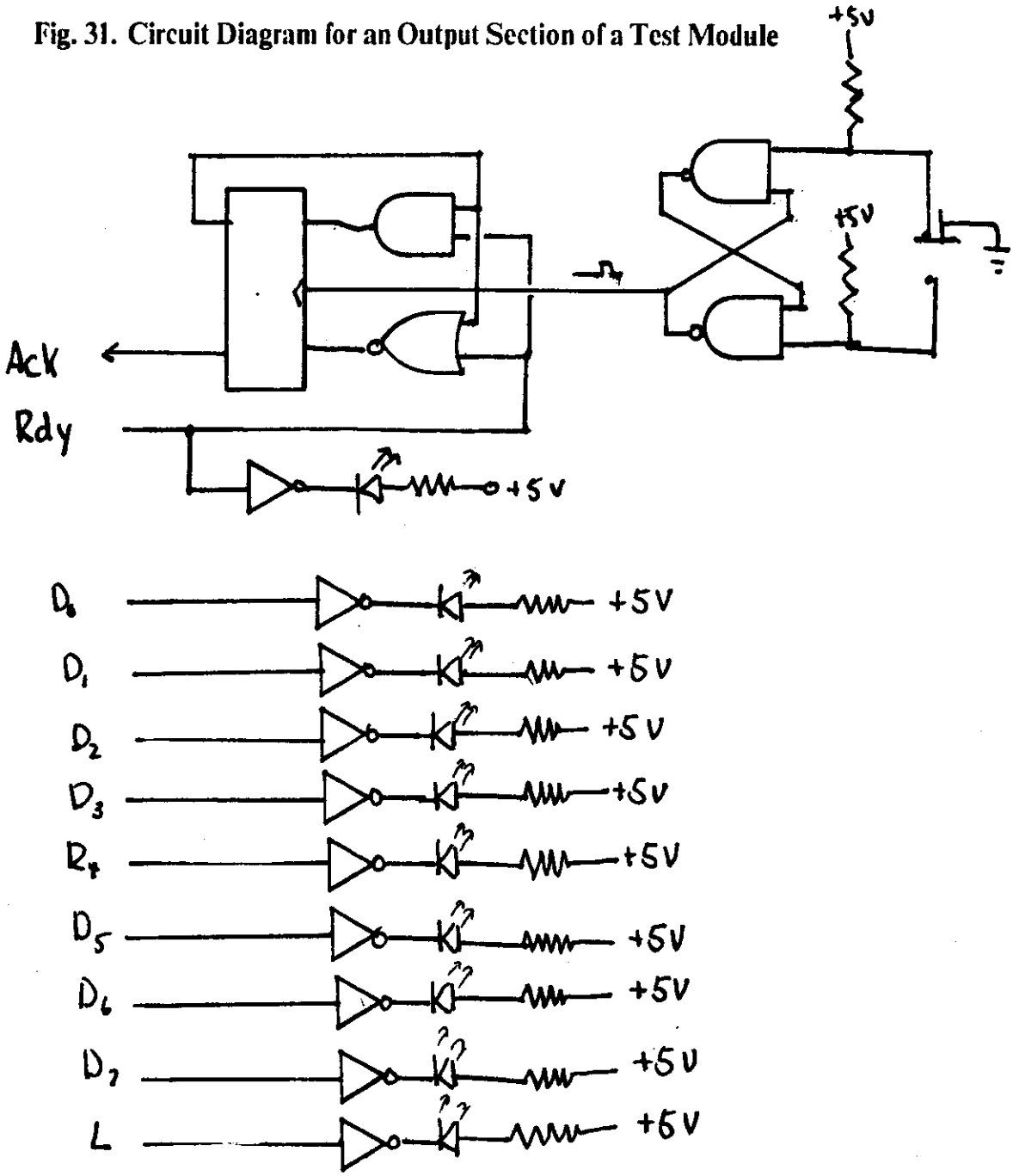


Fig. 31. Circuit Diagram for an Output Section of a Test Module



convenience of the tester.

To simulate the transmission of a packet the following procedure should be used. Assuming the test module is connected to the router, set the L and D values with the switches, set the data inputs as desired, and clock the flip-flop. This will send a ready to the router. Soon the ready should appear at the desired output port, provided that it is not in use. Clocking the flip-flop of the output section causes the acknowledge to be given. At this time the data at the output port should match those that were transmitted. Clocking the input and then the output section flip-flops will cause the ready and acknowledge signals to be reset. Setting L high simulates the last byte of the packet so the router should reset itself when the transmission is complete.

Bibliography

[1] T. J. Chaney, C. E. Molnar, "Anomolous Behavior of Synchronizer and Arbiter Circuits", IEEE Transactions on Computers, C-22, (April 1973)

[2] J. B. Dennis, "The Varieties of Data Flow Computers", Laboratory for Computer Science, M.I.T. CSG Memo 183, (October, 1979)

[3] J. B. Dennis, G. A. Boughton, C. K. C. Leung, "Building Blocks for a Data Flow Prototypes", Laboratory for Computer Science, M.I.T. CSG Memo 191, (February, 1980)

[4] J. B. Dennis, D. P. Misunas, C. K. C. Leung, "A Highly Parallel Processor Using a Data Flow Machine Language, Laboratory for Computer Science, M.I.T. CSG Memo 183, (October, 1978)

[5] Mead, Conway, Introduction to VLSI Systems, Addison-Wesley, Reading, MA, (1980)

[6] J. L. Redford, "A Design for a Routing Module", Laboratory for Computer Science. M.I.T, CSG Note 39, (January 1979)

[7] C. L. Seitz, "A TTL Compatible Arbiter", Memo to G. A. Boughton, (1979)

[8] T. L. Tung, "A preliminary Design for a router", Draft of report on initial work on router, (1979)

[9] E. Vishniac, "A Processor Module for Data Flow Computer Development, Laboratory for Computer Science, M.I.T. CSG Memo 134-1, (June, 1979)

[10] S. A. Ward, R. H. Halstead, D. Reed, Course Notes for 6.032, M.I.T. Department of Electrical Engineering and Computer Science (February, 1980)

[11] Texas Instruments, Bipolar Microcomputer Components Data Book, LCC4270