PACKET SUPPORT IN THE TEXAS RECONFIGURABLE ARRAY COMPUTER

BY

MATTHEW C. SEJNOWSKI, B.S.

May 1981 CS-TR-204 TRACTR-39

PACKET SUPPORT IN THE TEXAS RECONFIGURABLE ARRAY COMPUTER

BY

MATTHEW C. SEJNOWSKI, B.S.

REPORT

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF ARTS

THE UNIVERSITY OF TEXAS AT AUSTIN

MAY, 1981

TABLE OF CONTENTS

1.Ø INTRODUCTION

This report describes the hardware support for packet switching capability in the Texas Reconfigurable Array Computer (TRAC). This support consists of subsystems within the switch, processor and memory modules of TRAC. It is based on work previously published by the TRAC project. This report assumes familiarity with the basic concepts of reconfigurable interconnection network architectures in computer systems and some specific familiarity with the architecture of TRAC. Of particular relevance is work done by Dujari and Horne [1], Lipovski [2], and Tripathi and Lipovski [3].

Included here is a description of the concept of TRAC packets, as well as a description of the subsystems of TRAC that support packet communication within the various components of the TRAC system. The report excludes detailed description of the exact nature of packet content and packet processing by tasks running on TRAC. This section does, however, provide justification for packet communication as well as an overview of the packet communication system.

Section 2 contains an overview of the packet subsystem hardware, and Section 3 resolves the hardware design and operation to the logical function level. Finally, Appendix A has gate-level schematics of the implemented hardware.

Packets are an essential element in the total communication capability of a computer architecture structured by a reconfigurable but blocking interconnection network. Two communication requirements will illustrate the functions served by packet communication.

1.1 Intratask Communication

Within a task, each processor has control of its own data tree which is completely disjoint from other processors' data trees. The only entity that ties the processors together is the instruction broadcast bus.

It often occurs in a task that data from some or all processors must be sent to other processors within the same task. This requirement arises in many applications, including numerical methods, data bases, etc. If N processors are in the task, there is a potential requirement for up to (N x N) communication channels. In a typical operation data from a set of processors must be transferred to another set; many mappings are possible, ranging from a data permutation among a set of processors (e.g., row/column matrix access), to a data collection job in which all processors send local data to a common "data collection" processor. In any case, up to N sets of data may need to be sent in a given operation. If the instruction bus (or some other single bus linking the processors, such as a shared memory bus) were to be used for communication, data transfer would be costly in terms of time since the data would have to move serially on the bus.

Packet communication solves this problem. Data to be sent from a processor is partitioned into fixed-size blocks. Each of these blocks, which includes a prefix indicating the destination processor address, is called a packet. Figure 1.1 shows the TRAC packet data structure. All processors comprising the task operate in instruction lockstep. Each processor executes a MAP instruction [4] which initiates the packet sending process. Data to be sent

PACKET DIRECTION BYTE
PACKET DATA BYTE 1
PACKET DATA BYTE 2
PACKET DATA BYTE 3
PACKET DATA BYTE 4
PACKET DATA BYTE 5
PACKET DATA BYTE 6

FIGURE 11 PACKET DATA STRUCTURE

is partitioned into packets, including the destination information. Each processor then sends the first packet over its data bus towards the memory modules. A designated memory module has been preconditioned to receive the packet, and this module loads its packet transmitter/buffer with the packet contents.

Immediately, all packets simultaneously begin to make their way, switch by switch, to their respective destination processors. It should be noted that the sequence of switches used by a particular packet is unrelated to the use of the switches as bus connectors. The network can be regarded as being circuit-free during the clock phases used for node to node packet movement. After the memory module's buffer has been flushed onto the switch network, the sending processors can load successive packets into the buffer, until all data has been sent.

When two packets attempt to enter the same switch node simultaneously, a priority circuit in the switch forces one to wait and allows the other to pass.

Since, in the circumstances described above, all processors are dependent on receiving all data before they can proceed, they idle (after sending all their own packets) until all packets arrive. When all packets have arrived at their destination processors, the processors resume execution of instructions in lockstep.

Thus it is seen that packets are a powerful vehicle for intra-task communication. In TRAC terminology, this type of packet is called a "mapping" or "local" packet.

1.2 Intertask Communication

A second use for packet communication is for inter-task communication. It is often necessary for one task to send information to another task. A shared memory can be used to accomplish this, but it may be costly to do so if the amount of data to be transferred is small. Furthermore, in some cases it may be impossible to allocate a shared memory to a set of tasks. Another point is that packets are hierarchiclly a lower form of communication than shared memory. While shared memories must be allocated and acquired, the right to send packets is intrinsic and non-maskable. Indeed, packets will be the primary communication vehicle between tasks and the TRAC operating system.

Packets for inter-task communication are called "interrupting" or "global" packets. As with mapping packets, interrupting packets are first written into a special buffer in the sending task's memory. From there, they make their way from switch to switch towards the designated destination processor. Upon arriving, the interrupting packet lives up to its name by generating an interrupt to signal its arrival (unlike mapping packets, the receiving task is running asynchronously with the sending task and is therefore in an unknown state). The receiving processor can then read the packet data and handle it accordingly.

While the two types of packets move through the system in much the same way, their movement is multiplexed in time so that they cannot conflict with each other. One phase of the system clock cycle is dedicated for mapping packet movement, another for interrupting packet movement. It is anticipated that the mapping packet channel may sometimes become heavily loaded, so interrupting packets have bren given a separate channel so that their flow (which is anticipated to be relatively light but must be fast) is unimpeded by mapping packet traffic.

It is important to note that the switch-to-switch journey of packets from a memory module to a processor module does not interfere with the TRAC machine fetch-execute cycle in any way. The phases in which the packet moves (phases Ø and 1) are phases in which neither the data busses or instruction busses are in use; the processor module and the memory module are performing local computations.

2.Ø HARDWARE OVERVIEW

The packet support hardware consists of three major circuits: The packet buffer/transmitter associated with the memory modules at the base of the switch network, the packet switching circuit at each switch node, and the packet buffer/receiver in the processor module at the apex of the computer. They are described briefly here and then in more detail in section 3. This section concerns itself primarily with the way all three components work together to effect packet communication.

The standard packet for TRAC consists of a string of 7 bytes, as shown in Figure 1.1.

The first byte contains the destination processor number, and the remaining 6 bytes are user-defined data.

2.1 Packet Buffer/Transmitter

This circuit resides in the lowest level of switch modules. At this level, the spread of the switches is only one, so that each memory module has its own dedicated switch.

When a processor executes a send-packet instruction, the packet is serially (byte by byte) sent over the data bus to this packet buffer. The moment the first byte is buffered up in the buffer/transmitter, the transmitter logic starts trying to move the packet bytes onto the switch network.

As previously mentioned, packet movements on the switch network occur during phases Ø and 1 of the TRAC clock. Loads of the buffer/transmitter, however, occur during the remaining phases of the clock. Therefore, loads of the packet buffer and emptying of buffer contents onto the switch network are non-conflicting so that throughput is maintained at a maximum.

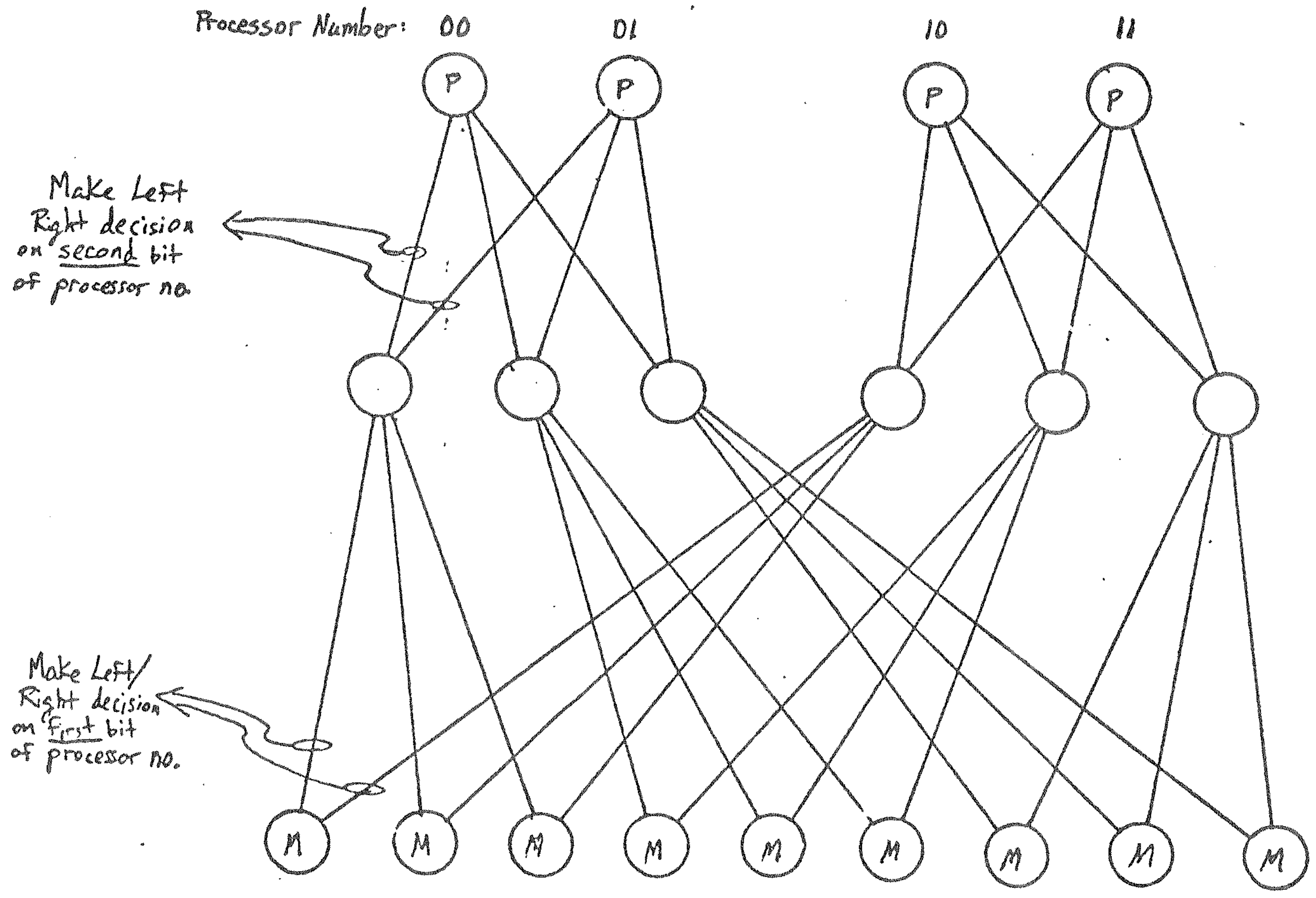
Thus the buffer serves as a go-between for the synchronous filling by the processor and the asynchronous emptying onto the switch network (by asynchronous we mean that the total time to empty the buffer is unpredictable, whereas the synchronous loading of the buffer can be accomplished in a fixed amount of time).

Basically, then, the buffer/transmitter is a first-in-first-out buffer along with control and interface circuitry. To the switch network, the circuit looks just like another switch module. In this way, all switches can be built identically and all the advantages of standardization and mass-production can be realized.

2.2 Packet Switching Circuit

This circuit resides in all switch modules except the lowest level. The function of this circuit is to arbitrate between packets attempting to enter from below (memory side), buffer the packet data, and use the packets destination specification to guide the packet towards its destination processor.

The first byte of the packet contains the destination processor number. At each switch node, a decision has to be made as to which of the two possible upward routes must be taken in order to get to the destination. The regularity and symmetry of the banyan network makes all switches at a particular


Processor Number:
00
01
10
11
P
M
Make Left Right decision on second bit of processor no.
Make Left/ Right decision on first bit of processor no.

P=processors; M=memories
NOTE: To get to any given processor from a given memory module, take processor number; starting at memory, go upwards to left if first bit of processor number is 0; go right if it is one. At next level, repeat but use second bit of processor number. Diagram shown is a simplification of actual TRAC system.

FIGURE 2.1 Example of packet routing

level look alike as far as upward travel to the processors. In other words, the left-right decision sequence to get to a given processor is the same regardless of which memory module the packet starts at. To take advantage of this, the processor destination number is just a binary encoding of the left-right sequence. At each switch, the binary value of a particular bit position (depending on the level of the switch) of this processor number is examined and used to determine the upward direction. All switches at a particular level use the same bit position. Figure 2.1 illustrates this graphically for a simplified network.

2.3 Packet Buffer/receiver

The packet buffer/receiver circuits reside in each processor module. This circuit collects a packet from the switch network and then signals the processor of its arrival. Means for subsequently reading the buffer by the processor are provided.

The buffer/receiver is very similar in operation to the buffer/transmitter. It is basically a first-in-first-out buffer capable of buffering an

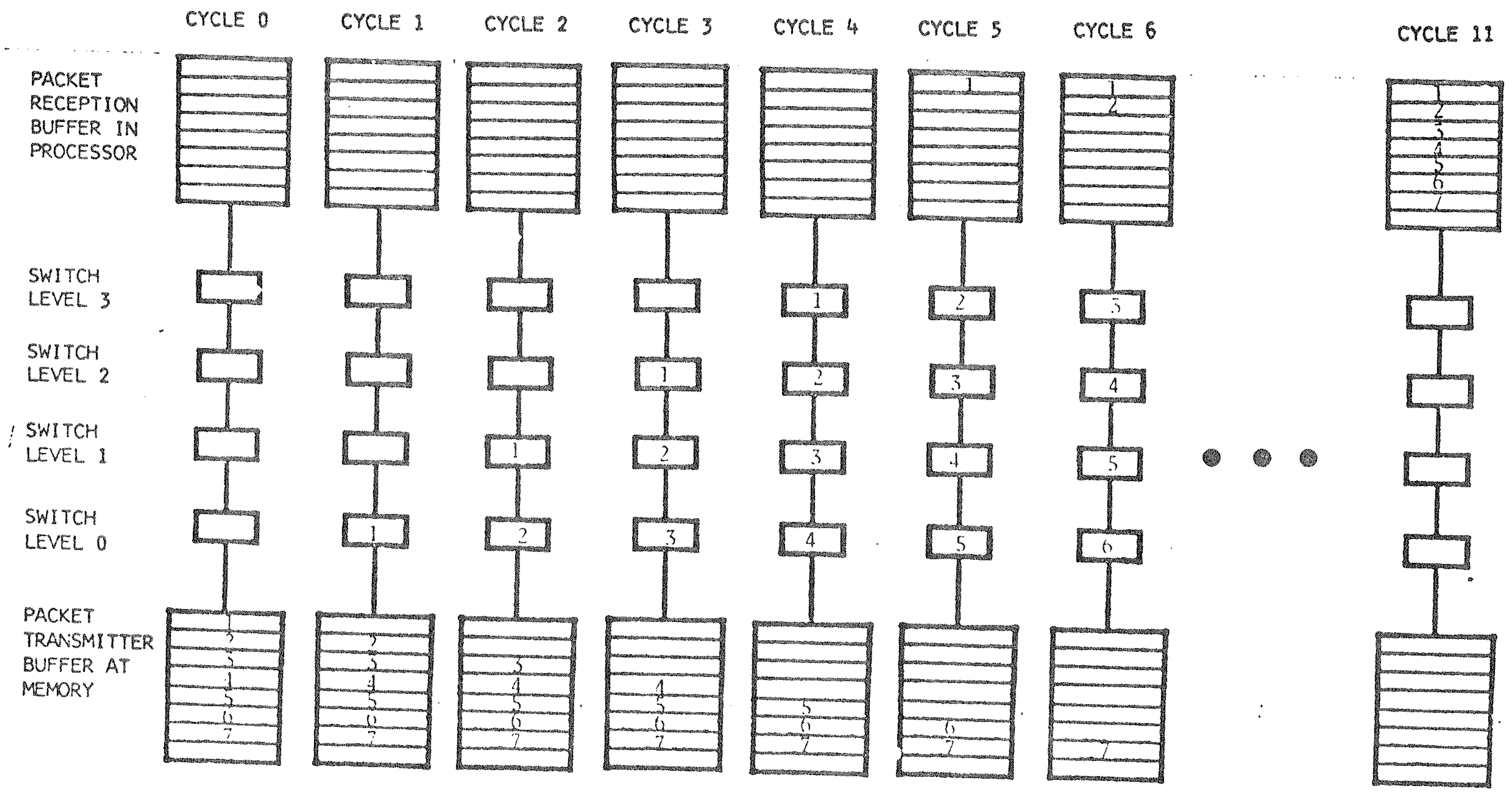


FIGURE 2.2 EXAMPLE OF PACKET MOVEMENT

entire packet if necessary, along with circuitry interfacing the switch with the processor. To the switch network, the buffer/receiver looks just like another switch module; to the processor, it provides the proper type of signals necessary to inform it of packet arrivals, and a means to read the packet data out of the buffer.

2.4 Packet Movement

As previously mentioned, packets in TRAC are 7 bytes long, the first byte of which is the destination processor number or direction byte. The banyan data bus is eight bits wide and the time within the TRAC memory cycle alloted to packet movement allows only one byte to be transferred safely from one switch node to the next in a given clock cycle phase. Therefore, packet bytes are moved serially upward from memory towards the processors. Each switch module can buffer one byte of a mapping packet and one byte of an interrupting packet.

Figure 2.2 shows movement of packets graphically. The switches are shown in the drawing with one input and one output; in TRAC, there are 3 packet inputs and 2 possible outputs. The next section shows how inputs are prioritized and outputs are selected based on the packet direction byte.

The figure makes it clear that TRAC packets are somewhat different than the usual concept of packets. They can be thought of as "packet trains" in that the bytes of the packet may be spread out over many switch nodes, yet the bytes are coupled together by the mechanism defined in Section 3.2.

Note that by the geometry of the banyan network, the path from a particular memory module to a particular processor module is unique. This might be thought to imply a poor fail-soft performance. The saving solution is that a processor can change the memory module that sends the packet among the memories in a data tree to achieve a reasonable fail-soft capability.

2.5 Packet Buffering

From Figure 2.2 it can be seen that packets originate in a 7-byte buffer at the memory end and make their way byte by byte through the switch network and finally into a 7-byte buffer at the processor. Each processor actually has two buffers, one for mapping packets and one for interrupting packets; likewise, each switch has a pair of one-byte buffers. Thus any processor can simultaneously receive the two types of packets. The memory modules, on the other hand, have only one 7-byte buffer which may be designated during configuration or at execution as either a mapping or interrupting packet buffer. This was done because most processors will have control of more than one memory module anyway, and it is advantageous to distribute the source locations of packets. The hardware for the packet buffer/transmitter is also substantially simplified. One memory module is designated as a mapping packet source and another is designated as an interrupting packet source. If a processor has only one memory in its data tree and it desires capability of sending both types of packets, the mode of the memory module can be changed dynamically through software to send either type of packet, though both types of packets could not be sent simultaneously.

Read/write overlap has been built into both the receive and transmit buffers. In the case of the transmit buffer at the memory end, after the first byte has been buffered up by the processor, negotiation logic immediately begins attempting to "empty" the buffer onto the packet switching network. The buffer load takes seven consecutive clock cycles. If no blockages occur in the switch network, the buffer will empty simultaneously as fast as they are loaded; at completion of the load, then, it may be possible for the processor to immediately load a second packet. Of course, in the worst case the entire packet may sit in the buffer for some period of time before being emptied onto the switch network.

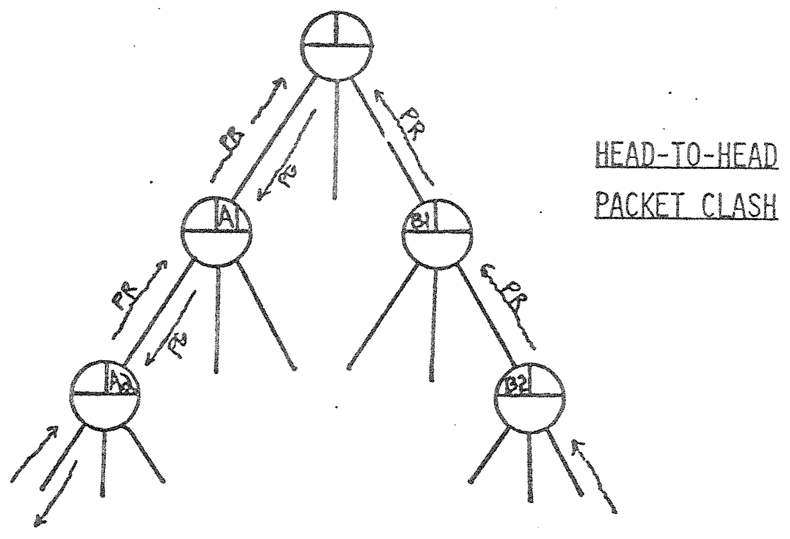
Similarly, the receive buffer at the processor end allows overlap of filling from the switch network and emptying by the receiving processor. Once again, in the best case the bytes of the packet will be read out by the processor as fast as they are coming in from the switch. Thus, assuming switch traffic is light, this buffer scheme allows a potential high throughput of packet data. Section 2.7 describes this further.

More detailed information on the buffering scheme will be given in the next section.

2.6 Packet Conflicts

Two packet trains may try to gain access to the same switch node (or receive buffer) simultaneously. This may occur in a "head-to-head" situation or a "head-to-body" situation; both are illustrated in Figure 2.3. In the head-to-head case, a priority circuit in the switch grants access to one of the packets. The three inputs to the switch have fixed priority in this respect. The other packet waits in its position for 8 cycles, which is the time it takes for the selected packet to move through and clear out of the switch; the waiting packet then gains access to the switch and begins to move again. The analogy of the switch network to a railroad switchyard works well.

In the head-to-body situation, the packet which is farther along the switch network is the one given priority; it proceeds to snake its way up the switch network until it moves clear of the switch in contention. Then the stalled packet is allowed to resume movement, assuming another packet of higher priority does not immediately attempt to gain access to the switch.



PACKETS A1-A2-A3-A4-A5-A6-A7
B1-B2-B3-B4-B5-B6-B7

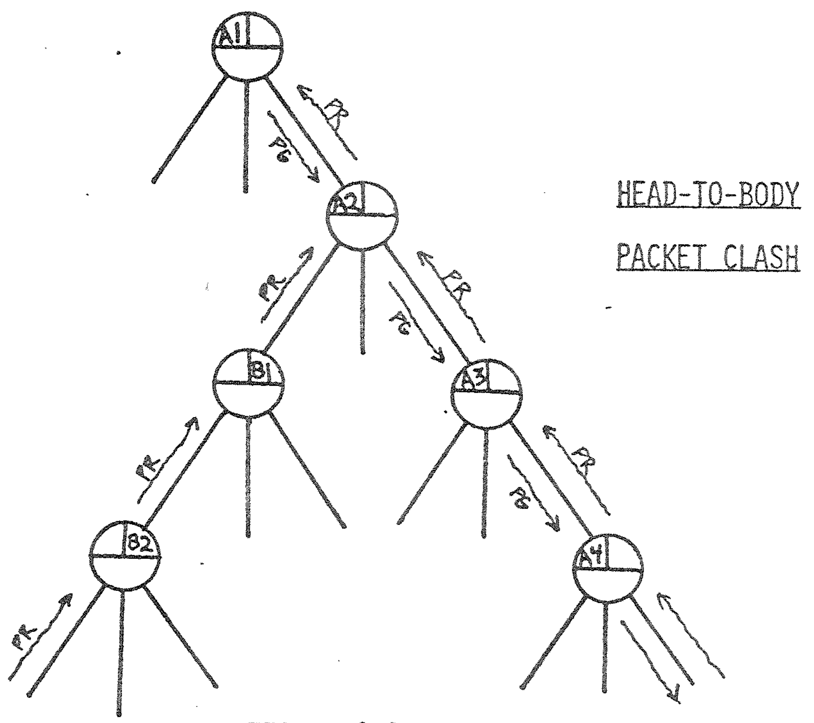


FIGURE 2.3 PACKET CONTENTION

The three-way head on collision situation is resolved similarly to the two-way situation. The packet on the highest priority line is passed through, then the one on the second-highest priority line is passed through, and finally the third packet is passed through.

2.7 Packet Timing

The moment the first byte of a packet is placed into the transmitter/buffer, it begins to arbitrate for entry into the switch network. If no conflicts arise during the upward movement of the packet, the first byte will arrive in the receiver/buffer exactly 5 cycles later; that is, one cycle for every switch node in the path.

At this time, the receiving processor can read the bytes out of the buffer. Six cycles later, the last byte of the packet train arrives in the receiver/buffer. Despite the initial delay of 5 cycles, at full bore the packet transmission rate approaches one byte per clock cycle (actually, the overhead of the direction byte plus a "dead" byte between packets reduces the maximum throughput to .75

bytes per clock cycle). Delays in packet arrivals due to network blockage are inevitable in heavy traffic conditions, and will cause a delay in packet arrival equal to 8 cycles for every blockage that stopped the packet.

Another consideration is the precedence of packets. If a processor P1 sends two packets from the same memory module in succession to a single processor P2, the packets are guaranteed to arrive in the same order they were sent, since they both must follow the same route. However, if two processors P1 and P2 simultaneously send packets to a common destination P3, nothing can be said about the arrival order since the paths are different and blockages can occur along either or both paths.

In the next section, operational details of each of the three packet support circuits are given.

3.Ø PACKET CIRCUITS

The details of packet subsystem operation are given in this section. The buffer/receiver circuit will be described first, then the packet switching circuit and finally the buffer/transmitter circuit.

3.1 Packet Buffer/Receiver Circuit

A block diagram of the buffer/receiver is shown in Figure 3.1. Signals at the top interface the logic with the rest of the processor; the bottom signals connect to the three switch modules below. Here is a description of the interface signals:

1. MPA- Mapping Packet Arrival; sent to the processor, it is asserted upon introduction of the first byte of a mapping packet into its buffer.

2. IPA- Interrupting Packet Arrival; sent to the processor, it is asserted upon introduction of the first byte of an interrupting packet into its buffer.

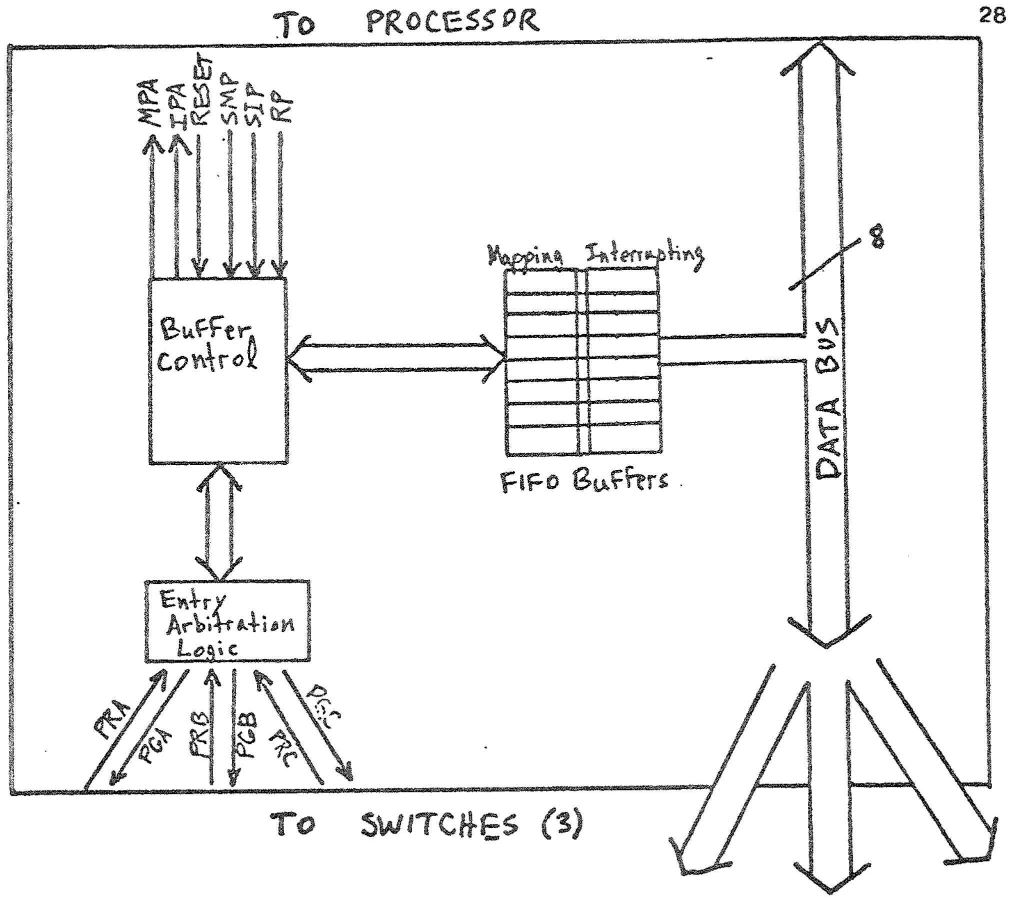


FIGURE 3.1

PACKET BUFFER/RECEIVER

3. SMP- Set Mapping Packet; controlled by the processor, this signal sets the control logic up to read a mapping packet from the buffer into the processor.

4. SIP- Set Interrupting Packet; controlled by the processor, this signal sets the control logic up to read an interrupting packet from the buffer into the processor.

5. RP- Read Packet; controlled by the processor, this signal strobes the next byte of the previously selected type of packet (via SMP or SIP) from the buffer onto the data bus for reading by the processor.

6. PRA,PRB,PRC- Packet Request; sent to the logic from one of the switches beneath it in order to request entry of a packet byte into the proper buffer.

7. PGA,PGB,PGC- Packet Grant; returned to the requesting switch in reply to a PR signal, granting entry into the proper buffer.

A typical scenario of how these signals are used will illustrate operation of the circuitry. This example is for handling mapping packets. Interrupting packets are handled similarly except SMP becomes SIP, MPA becomes IPA, phase Ø becomes phase 1 and phase 3 becomes phase 4.

1. PRA is asserted by the switch in phase 3, indicating that the first byte of a packet is at the switch and would like to enter the buffer/receiver. If the mapping packet buffer is empty, PGA is returned, giving the switch the go-ahead.

2. The following phase Ø, the byte enters the buffer. MPA is asserted to the processor.

3. When the processor is free to read the buffer contents, SMP is asserted.

4. During any phases except phases Ø and 1, the processor asserts RP; the first byte of the mapping packet buffer is placed on the data bus for reading by the processor.

5. Meanwhile, during each phase 3, PRA is sent by the switch and PGA is returned. During each phase Ø, subsequent bytes of the mapping packet are placed on the data bus and read into the buffer.

6. Each time the processor asserts RP, the next byte from the mapping packet buffer is placed on the data bus for reading by the processor.

7. The PRA/PGA/packet-load is repeated 8 times in all (the last byte is a dead byte).

8. The RP/data-read is done 8 times by the processor; on the eigth time, the buffer has been flushed.

9. A final SMP is asserted by the processor; this clears the receiver control circuits and it is then ready for the next mapping packet to come along.

Note that loads of the mapping and interrupting packet buffers can occur simultaneously, but the processor must read one entire buffer before it can read the other buffer. This was done to aid in avoiding stalled packet trains on the switch network which would tend to degrade the overall packet throughput.

A provision in the circuit is made for the case of a dead (disabled) processor. If the processor "broke" bit is set, the circuit will automatically accept any packets that attempt to enter the buffer. This is to prevent a packet traffic jam from ensuing, which would degrade (if not stop completely) overall packet communication. For this same reason, interrupting packets generate non-maskable interrupts.

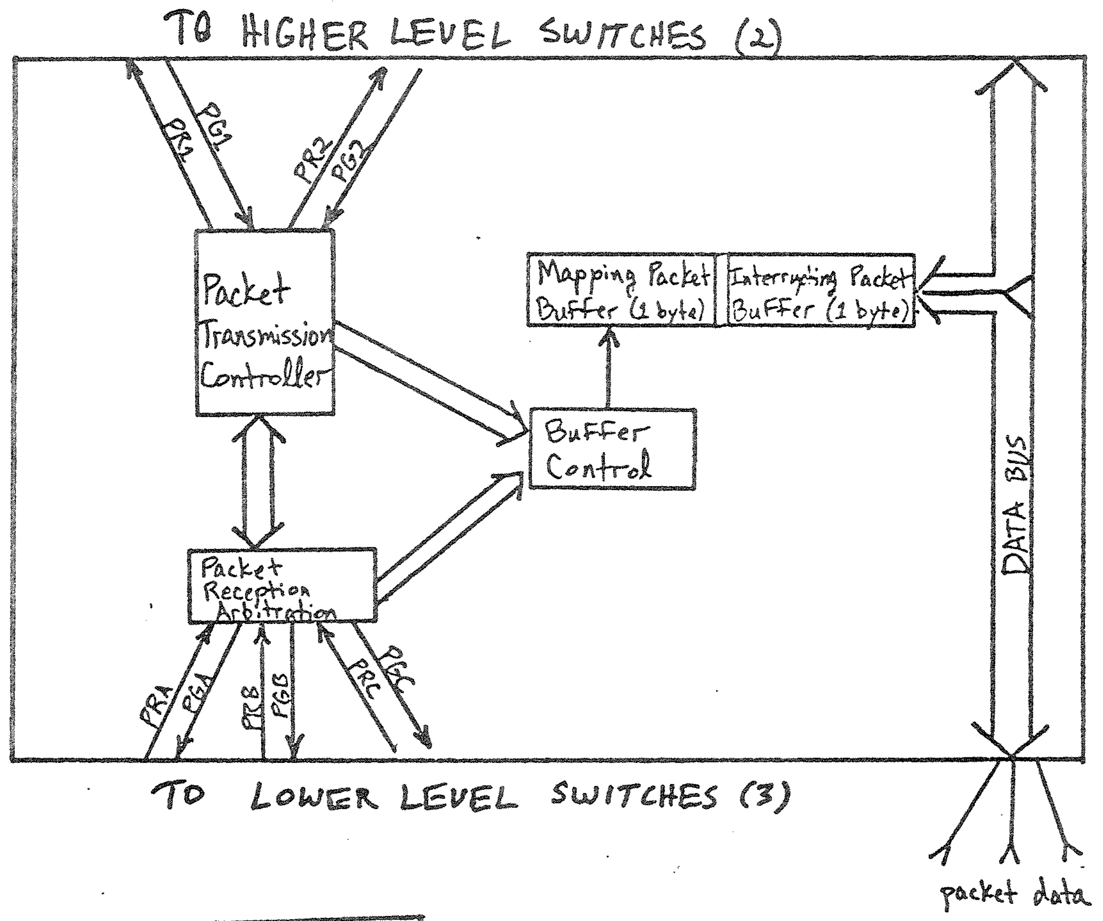
The actual circuit on the processor board is shown in Appendix A.

3.2 Packet Switching Circuitry

A block diagram of this circuit is shown in Figure 3.2. Signals at the top of the figure go to corresponding signals at the next higher switch level (2 of them); the signals at the bottom go to corresponding signals on switches at the next lower level (3 of them).

In order to synchronize the levels of the switch network, the backplane of each switch provides 3 signals: NEG (-otiate), DIR (-ection), and END. As shown in Figure 3.3, these signals are distributed across levels of switches. After each major cycle of the TRAC master clock, the signals each shift up by one level. From the viewpoint of a particular switch, then, one cycle is spent in NEG mode, the next in DIR mode, the next 5 in blank modes, and then the next in END mode, after which the pattern repeats. The use of these signals is described below.

The phases of the clock used for the various packet related functions are described here:



TO HIGHER LEVEL SWITCHES (2)
PR1
PG1
PR2
PG2
Packet Transmission Controller
Mapping Packet Buffer (1 byte)
Interrupting Packet Buffer (1 byte)
Buffer Control
DATA BUS
Packet Reception Arbitration
PRA
PGA
PRB
PGB
PRC
PGC
TO LOWER LEVEL SWITCHES (3)
packet data

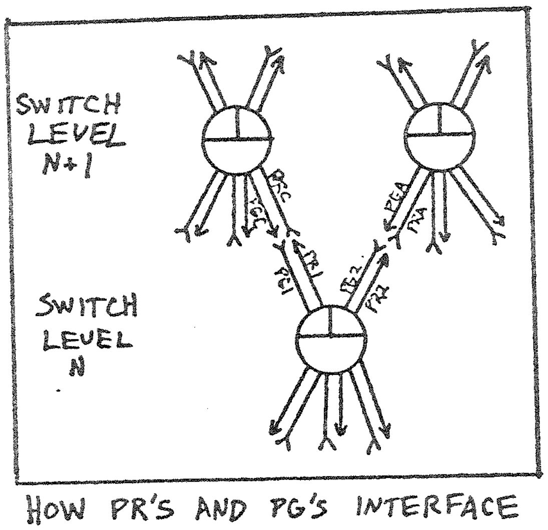


FIGURE 3.2

PACKET SWITCHING

	CYCLE 0	CYCLE 1	CYCLE 2	CYCLE 3	CYCLE 4	CYCLE 5	CYCLE 6	CYCLE 7	CYC 8
SWITCH LEVEL 3	---	---	---	END	NEG	DIR	---	---	---
SWITCH LEVEL 2	---	---	END	NEG	DIR	---	---	---	---
SWITCH LEVEL 1	---	END	NEG	DIR	---	---	---	---	---
SWITCH LEVEL 0	END	NEG	DIR	---	---	---	---	---	END

FIGURE 3.3

BACKPLANE SIGNALS

phase Ø-movement of mapping packet data over the data bus;

phase 1-movement of interrupting packet data over the data bus;

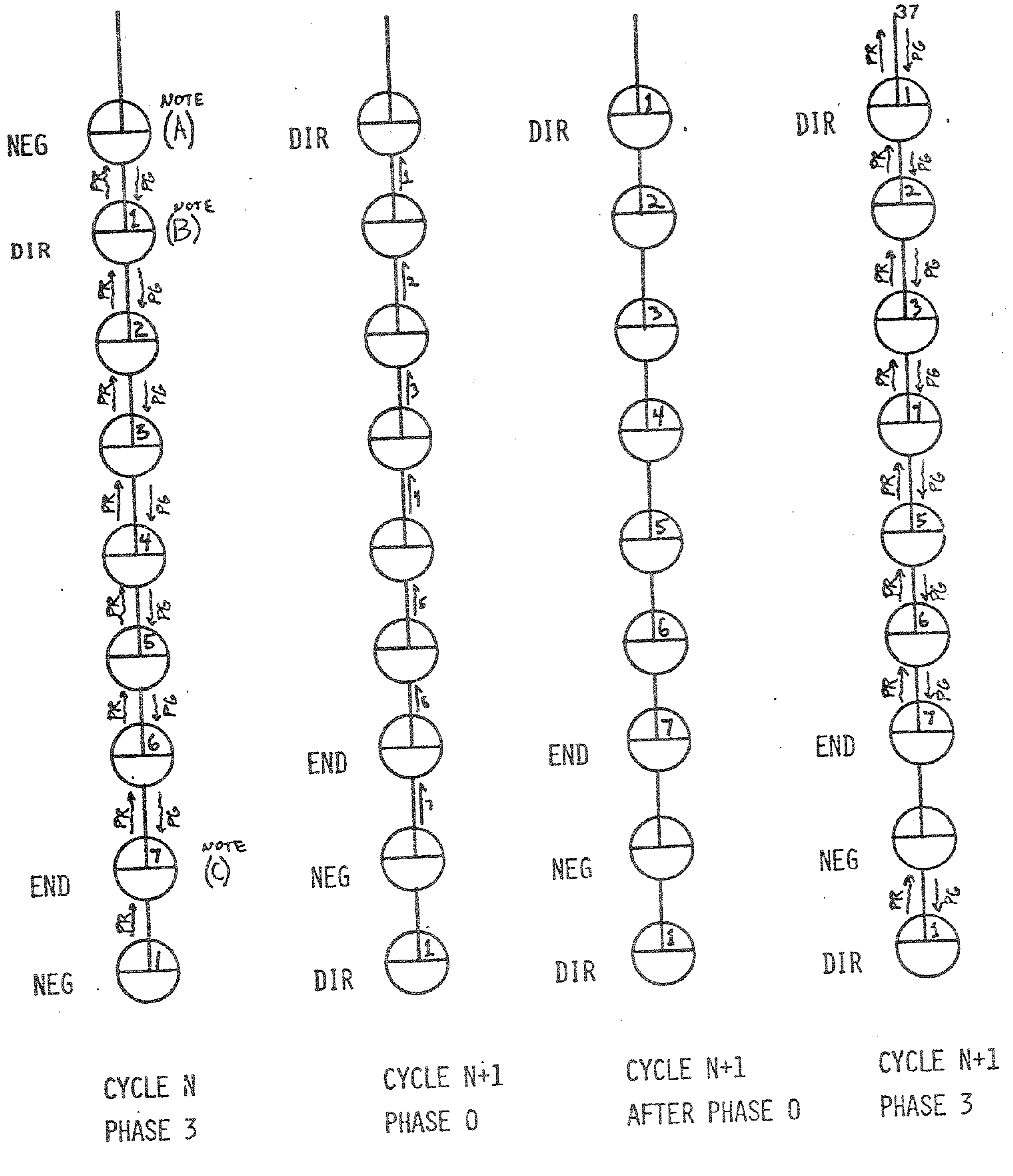
phase 3-arbitration for mapping packets using PR and PG;

phase 4-arbitration for interrupting packets using PR and PG;

During phases 3 and 4, the data bus is not used by the packet circuitry. Figure 3.4 illustrates an example of mapping packet movement. Interrupting packet movement is analogous except phase Ø becomes phase 1, and phase 3 becomes phase 4.

The following points can be noted from the Figure 3.4:

1. (see (A) in the Figure 3.4) When a switch has NEG asserted and its buffer is empty, and it receives a PR from below, that byte is a header of a packet. A PG will always be returned.



NOTES: CIRCLES REPRESENT SWITCH NODES;
UPPER QUARTERS REPRESENT LEFT AND RIGHT BYTE BUFFERS;
ONLY RELEVENT LINKS ARE SHOWN;
MORE LEVELS ARE SHOWN THAN IN ACTUAL COMPUTER FOR ILLUSTRATION.

FIGURE 3.4 PACKET SWITCHING SIGNALS

2. (see (B) in figure) When a switch has DIR asserted, the byte, if any, in its buffer is the header byte and therefore is used to determine which of the two upward paths is to be taken next.

3. (see (C) in the figure) When the END signal is asserted, the byte, if any, in its buffer is known to be the last byte of a packet; thus the PG signal is not propagated downwards beyond this switch node.

During phase 3, all switches that have packet bytes in their buffers generate packet requests (PR). Packet grants (PG) are returned by the switch above if 1) The switch above is empty and in a NEG state (that is, can accept a header), or 2) The switch above is not empty but has received a PG from the switch above it (the exception to this is if the switch is in an END state, in which case it never issues a PG).

Thus it is seen that packet grants originate at the head of a packet and propagate downwards through each switch containing a byte of the packet.

This propagation continues until a switch in the END state is encountered, where the PG propagtion terminates. Thus each switch taking part in holding the packet is informed that in the subsequent phase Ø, they will pass data up to the next higher level, and accept data from the lower level. Note that if the head of a packet becomes blocked by another packet, the lack of a PG is similarly propagated downwards, and each byte holds still. Figure 2.2 shows some examples of PG propagation.

From a system-wide point of view, then, packets are "pulled up" the banyan network by the NEG signal, which is continuously cycling upwards. When two packets being pulled up try to gain access to the same switch, one of the packets waits while the other one is pulled through and clear of the switch. The next NEG to come along, eight cycles later, grabs the waiting packet and commences to pull it up.

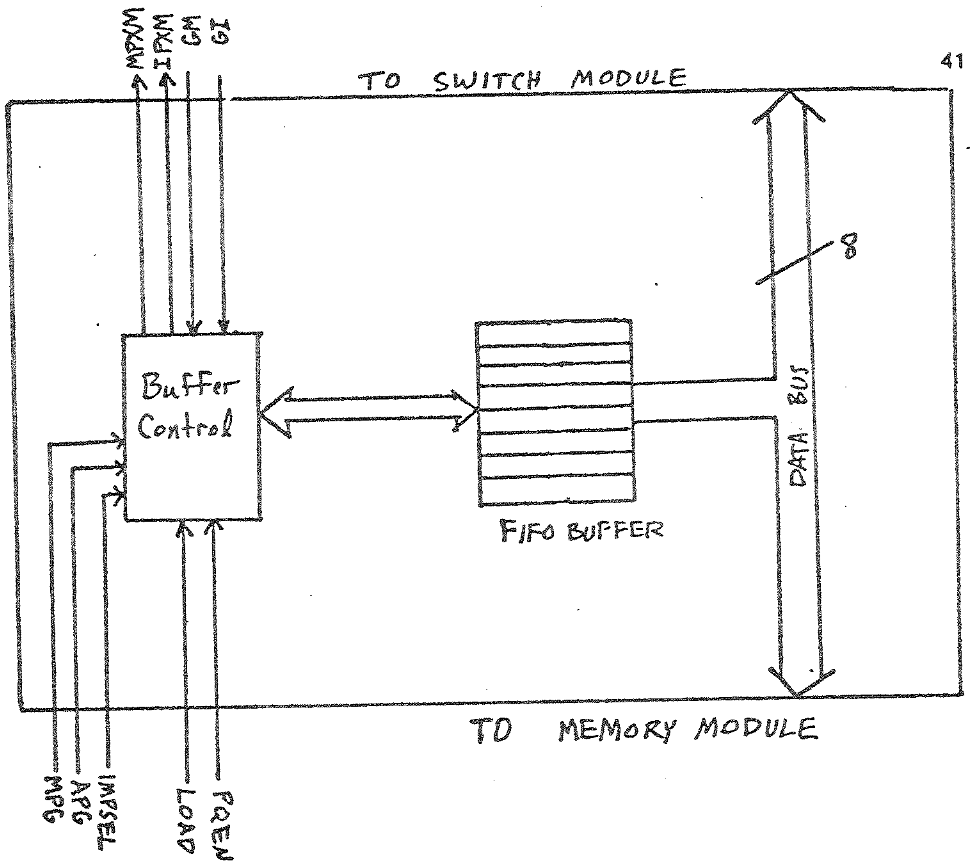
Note that there are actually two buffers in each switch, one for the left upward link and one for the right (so between mapping and interrupting packets there are 4 buffers in all). This was done to simplify the control circuitry in that the tri-state output from each of the sides can then be independently enabled in order to steer the packet upwards. When a byte enters the switch it is latched into both buffers. When the byte moves on upwards, it is output only from the correct buffer (left or right).

The actual circuit used for packet switching is shown in Appendix A.

3.3 Packet Buffer/Transmitter Circuitry

A block diagram of this circuit is shown in Figure 3.5. The buffer/transmitter is very similar to the buffer/receiver at the processor end of the system in that the buffer serves to interface a processor desiring to transmit a packet with the switch network; however, the loading and emptying of the buffer are reversed; here, the processor loads the buffer with a packet, and the packet is subsequently emptied onto the switch network.

The following signals are used by the packet buffer/transmitter circuit:



<u>FIGURE 3.5</u> PACKET BUFFER/TRANSMITTER

1. MPG- Mapping Packet Generator; asserted by logic on the memory board, this signal flags the buffer/transmitter as being dedicated to sending mapping packets.

2. APG- Interrupting Packet Generator; asserted by logic on the memory board, this signal flags the buffer/transmitter as being dedicated to sending interrupting packets.

3. IMPSEL- Interrupting/Mapping Packet Generator; when the circuit starts actively handling packets, this signal from the memory board must be set to 1 or Ø for sending mapping or interrupting packets respectively.

4. PQEN- Packet Query Enable; when asserted by the memory module (and ultimately by the processor whose data bus the memory module is on), this signal causes the upper order bit of the switch data bus to be a Ø or a 1 depending on whether the buffer is completely empty or not; used for buffer querying by the processor.

5. MPXM- The equivalent of a switch module's "PR" signal; basically, a request to send a mapping packet onto the switch network.

6. IPXM- The same as MPXM, but for interrupting packets.

7. GM- The equivalent of a switch module's "PG" signal; a grant from the switch module above the memory module giving the go-ahead for buffer emptying.

8. GI- The same as GM, but for interrupting packets.

9. LOAD- Generated by the memory module; when asserted, loading of the packet buffer by the processor can occur.

The following is the sequence used by the processor to send a packet (assume the packet is a mapping packet; interrupting packets are handled similarly):

1. At time of task initialization, MPG should be set at the memory module selected for mapping packet generation; likewise, the module selected for interrupting packet transmission should have APG set.

2. The processor causes PQEN to be asserted and reads the switch data bus; if the upper order bit is a 1, the processor must wait till the packet transmit buffer becomes empty (the upper order bit goes to Ø).

3. IMPSEL is set to Ø or 1 for sending an interrupting or mapping packet respectively.

4. LOAD is asserted by the sending processor. The first byte of the packet to be sent is placed on the switch data bus. Loading of the byte into the buffer takes place at the enu of phase 4.

5. On each subsequent cycle, the processor places succeeding bytes of the packet on the switch bus.

6. Meanwhile, the packet buffer/transmitter circuit attempts to empty the contents of the buffer onto

the packet switching network. MPXM is sent to the switch module adjacent to the memory module, during phase Ø. If the switch module can accept the data, GM is returned; as long as this occurs, succeeding packet bytes are read from the buffer during phase Ø and loaded onto the packet switching network on their way towards the destination processor.

7. If GM is not returned, MPXM is held true until it is; in any event, after all 7 bytes of the buffer are transferred onto the packet switching network, the buffer has become empty and the circuit resets itself.

Section 2.5 described how packet buffer/transmitter modules can be configured as either mapping or interrupting modules. This configuration is done by asserting one of MPG or APG in that module. In a given processor's data tree, only one memory module should have MPG asserted and only one (not the same one) should have APG asserted. If the processor has only one memory module in its data tree, assertion of MPG and APG can be time-multiplexed in order to alternately send mapping packets and interrupting packets. Changes in the MPG/APG settings must occur, however, only when the transmit buffer is empty.

The actual circuitry used for packet switching is shown in Appendix A.

4.Ø CONCLUSION

We have seen that packets provide a powerful communication media for data movement within a task, between tasks, and between the TRAC operating system and tasks (or jobs). The right to send packets is irrevocable and thus provides a degree of assurance of communication capability, as opposed to shared memories and other forms of data transfer which require software assignment of resources.

The packet support subsystems are designed to work together to provide fast, reliable channels between senders and receivers. Packet movement through the switch network takes advantage of the idle clock cycle phases on the data busses to move packet data without degrading the performance of the rest of the system operations. The inclusion of packet switching in TRAC, then, is a no-cost addition as far as machine speed is concerned.

Packets also exhibit the desirable characteristic that throughput of packet data degrades very gracefully as the size of the banyan network is increased. They take full advantage of the inherent parallelism provided by the TRAC architecture.

Some key features of the implementation are:

1. Packets move on the switch network during times the data busses are idle;

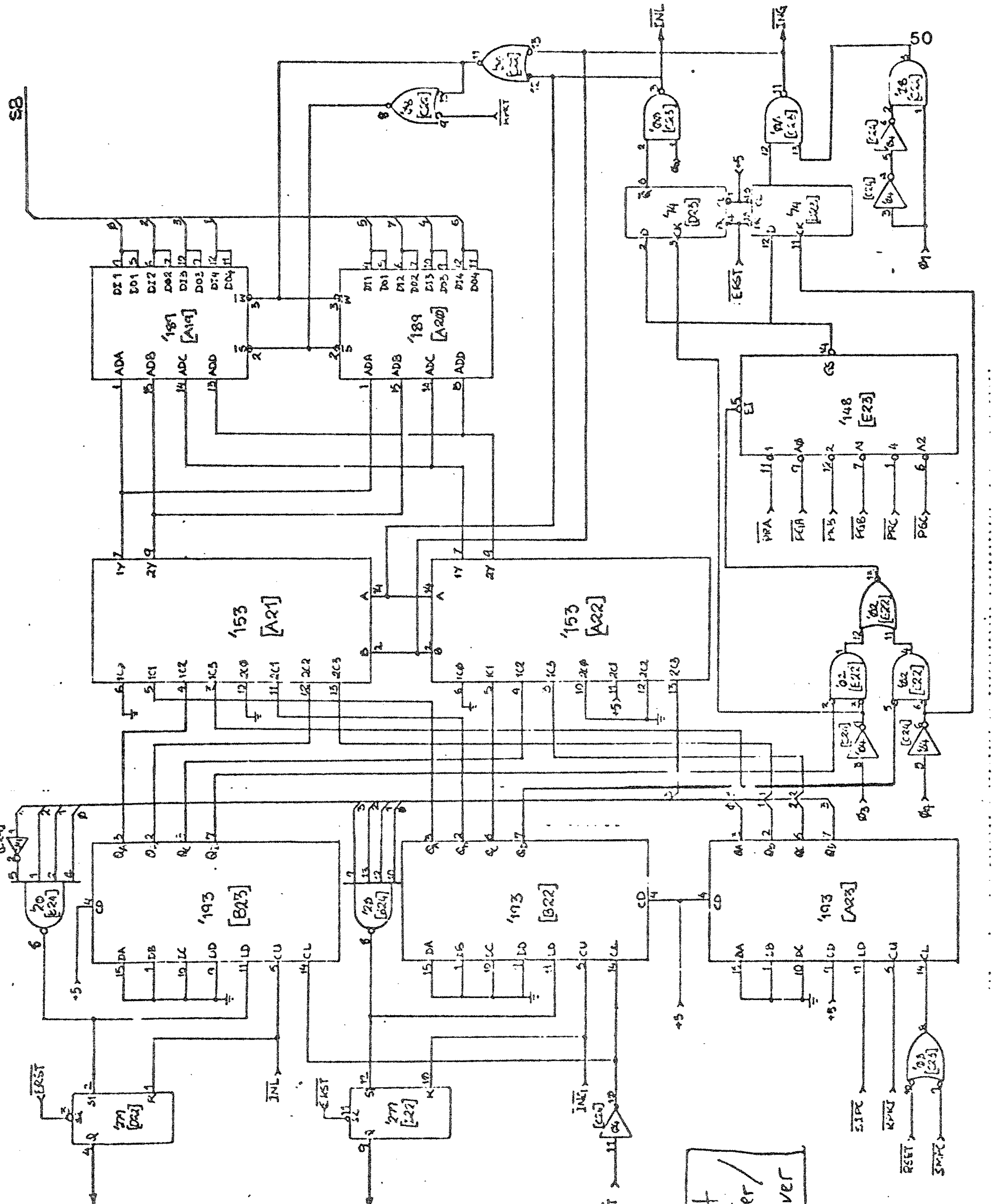
2. Buffering at the source and destination allow packets to move over the switch network at their own rates, without bogging down sender or receiver; read/write overlap in the buffers allow potentially high throughput;

3. Packets take advantage of symmetry in the banyan network to use the direction byte of the packet in steering the packet to the correct destination processor, regardless of the source location;

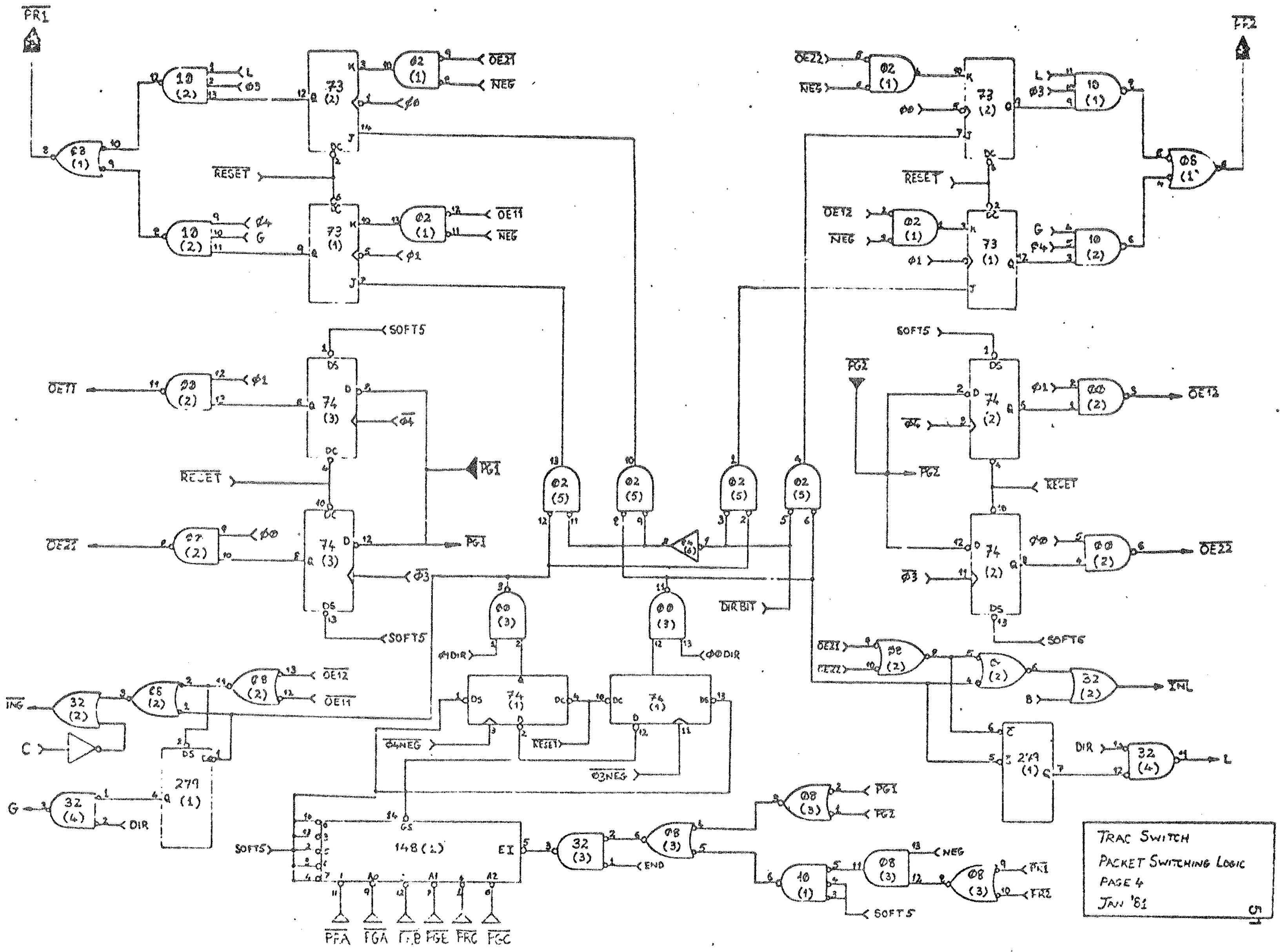
4. High throughput is achieved by using a "shift-register" approach in which packets are broken up into managable pieces, shipped through the switch network, then reassembled at their destination.

APPENDIX A

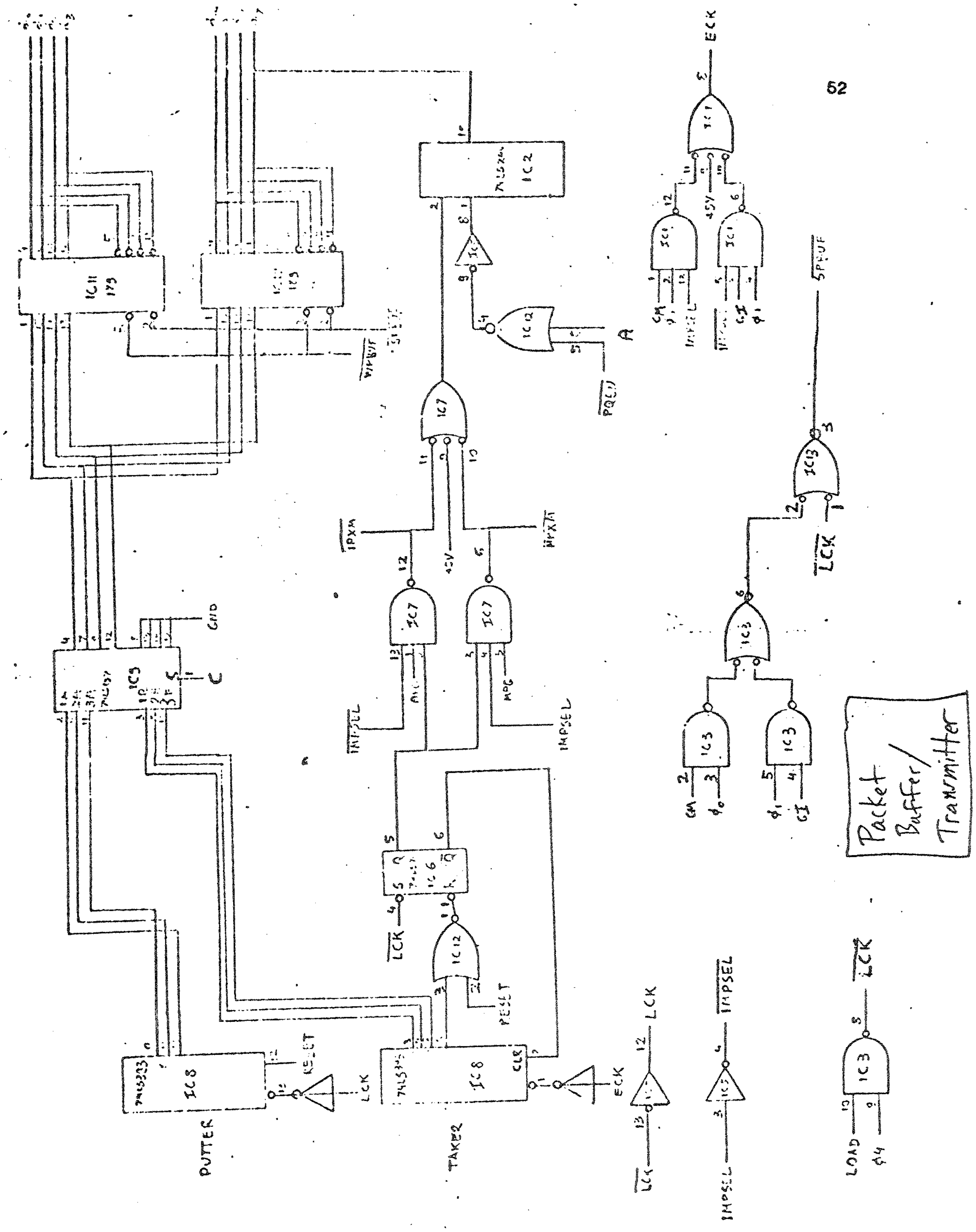
Packet Circuit Schematics



Packet Buffer/ Receiver



TRAC SWITCH
PACKET SWITCHING LOGIC
PAGE 4
JAN '81


Packet Buffer/Transmitter
PUTTER
TAKER
74LS393
IC 8
IC11
IC 5
IC 7
IC 6
IC 12
IC 13
IC 3
IC 1
IC 2
74LS74
RESET
LCK
ECK
IMPSEL
LOAD
GND
A
C
5PFUF

BIBLIOGRAPHY

1. Dujari,G.,Horne,P.,"A Preliminary Report on the Node Circuit for Banyan Switch of the Texas Reconfigurable Array Computer,"Technical Report TRAC-9, University of Texas at Austin, January, 1979.

2. Lipovski,G. J.,"The Architecture of the Banyan Switch for TRAC,"Technical Report TRAC-7, University of Texas at Austin, January, 1979.

3. Tripathi,A.R.,Lipovski,G.J.,"Packet Switching in Banyan Networks,"Technical Report TRAC-14, University of Texas at Austin, January, 1979.

4. Premkumar,U.V.,"TRAC: Principles of Operation," Technical Report TRAC-3, University of Texas at Austin, January, 1979.

VITA

Matthew C. Sejnowski was born in Cleveland, Ohio, on November 22, 1955, the son of Joseph F. Sejnowski and Theresa C. Sejnowski. After graduating from St. Joseph High School in Cleveland in 1972, he enterred Case Western Reserve University in Cleveland, one year of which he worked part-time at General Electric Corp. as a programmer. He received his Bachelor of Science degree in Mathematics at CWRU in 1976, and went to work for Texas Instruments Inc. as a programmer. While there, he held positions of systems engineer and marketting systems engineer as well. In May of 1979 he went on leave of absence from Texas Instruments in order to attend the Graduate School of the University of Texas.

Permanent address: 611 W. 31 1/2 Street
Austin, Texas 78705

This report was typed by the author.