A RESTRUCTURABLE INTEGRATED CIRCUIT FOR IMPLEMENTING PROGRAMMABLE DIGITAL SYSTEMS

ROB BUDZINSKI
JOHN LINN
SATISH THATTE

TEXAS INSTRUMENTS INCORPORATED
DALLAS, TEXAS

This work is funded in part by the Defense Advanced Research Projects Agency under contract No. MDA 903-79-C-0433.

ABSTRACT

The Restructurable Integrated Circuit, a highly flexible and programmable multimicrocomputer is presented. The goal of this integrated circuit is to apply the large number of gates that are available on a custom designed VLSI IC to the design of a highly flexible integrated circuit. The main application of the Restructurable Integrated Circuit (RIC) will be the implementation of digital system hardware through programmation of a RIC. The flexibility provided within the RIC includes: user definable micro language, user programmable assembly language, user programmable microcode, dynamic coordination of multiple internal processors, coordination of processors on multiple RICs, internal memory use as caches or as a member of a virtual memory hierarchy, general topology for interchip communication and external data paths, and a user definable interrupt mechanism. By providing this high degree of flexibility, the cost and reliability advantages of high volume production can be accrued, while providing performance comparable to a custom VLSI IC.

1. INTRODUCTION

The concept of a highly programmable, restructurable VLSI integrated circuit is an extremely important step towards achieving the maximum impact from VLSI. Not only do programmability and flexibility provide new creative opportunities for the system designer, but they help overcome two major obstacles to pervasive use of VLSI, as well.

Design cost and design cycle time are critical barriers to implementing VLSI systems. Typically a state-of-the-art LSI custom design consisting of 5K gates and 5K bits of read-only memory costs approximately $500K and takes about 18 months to design and layout. This works out to about $100 per gate in the design. Even if, over the next five years, design costs are reduced by an order of magnitude to $10 per gate, a typical state-of-the-art VLSI custom design in this time frame, consisting of 50K gates and 50K bits of read-only memory, will still cost $500K to design. A very flexible restructurable VLSI chip that can be structured to provide a wide range of capabilities with state-of-the-art performance presents a viable alternative to custom designs in low volume applications.

Reliability, testing and maintenance considerations are extremely important in complex VLSI systems. When reflected in the cost of service calls or returned products, poor reliability can contribute more to system lifetime cost than the initial manufacturing cost. Traditionally, reliability has improved with each increase in the level of integration. However, reliability benefits from accumulated learning, as shown in Figure 1. A generic programmable chip, in this case the Texas Instruments TMS-1000 microcomputer, increases in

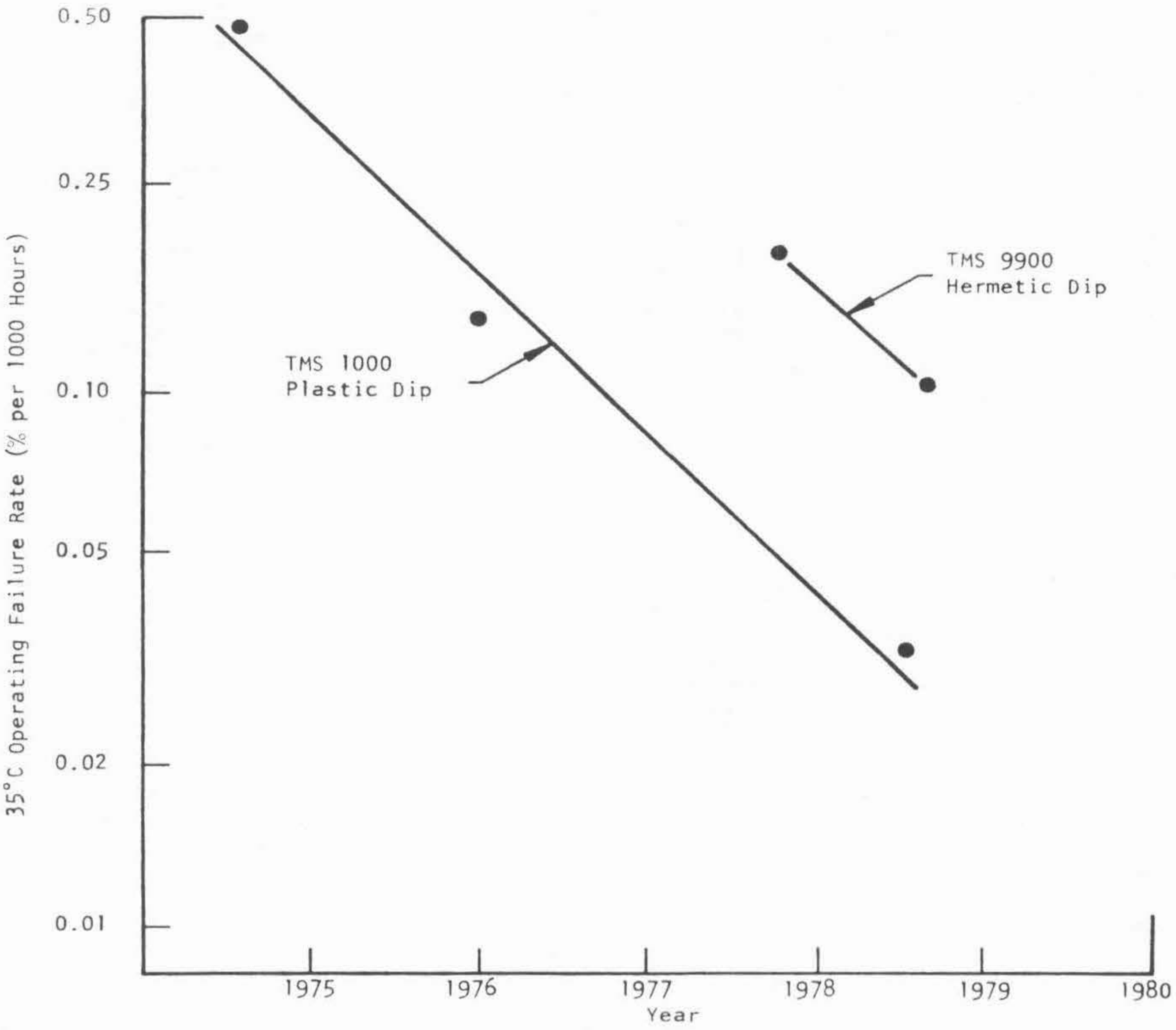

FIGURE 1. Microprocessor Failure Rate History

reliability as more are manufactured. Although any specific programmation of the chip may not be made in any significant volume, each programmation still benefits from the reliability improvements of the generic device. Custom designs, even using an identical process, do not benefit significantly from the high volume reliability learning of other chips. A restructurable VLSI circuit will provide a high degree of reliability learning, even though the volume of most programmation types may be small.

The RIC is a semi-custom IC similar in purpose to gate arrays or master-slices. The gate array approach allows a logic diagram to be translated into silicon through software. The software creates an interconnect pattern between the gates so that the logic diagram is implemented in silicon. The gate array approach is very flexible, but the gate array approach does not provide any special structure for implementing programmable digital systems. The RIC approach uses the large number of gates on a VLSI IC to build a chip which is highly flexible for implementing programmable digital systems. The RIC approach differs from gate arrays in that the RIC has the vast majority of silicon committed to a specific design. The flexibility of the RIC is achieved through the design of a programmable mechanism for controlling the hardware resources on the chip. The block diagram of the Restructurable IC is shown in figure 2.

The RIC is a multimicrocomputer containing four 16 bit processors called Microprogrammable Slices (MPSs). The MPS resources can be controlled at two basic levels. One level is the coordination of MPSs. The four MPSs can be dynamically configured at run time into any combination of three fundamental structures. One structure is the lockstep in which two or more MPSs are structured to form a wider word computer. This structure is formed by directing the same microinstruction stream to all of the MPSs in the lockstep and structuring the arithmetic status, carry chain, and shift/rotate linkage to configure the MPSs into a wider word computer. The second fundamental structure is independent MPSs. In this structure each MPS has its own microinstruction stream. A set of array processors can be structured in the independent configuration by directing the same microinstruction stream to the MPSs without any coordination of arithmetic signals between MPSs. The third fundamental structure is pipelined MPSs. Each MPS forms a stage in the pipeline. The microinstruction streams are different for each stage. An internal data bus within the RIC provides for simultaneous sending and receiving of data between adjacent stages (MPSs) in the pipeline.

The other basic level for controlling MPSs is language interpretation. The language interpretation structure is programmable at two levels. One level is a definable vertical microcode and/or assembly code. The other level of programming is a PLA which interprets the vertical microcode through finite state machines. The vertical microcode and/or assembly code are defined through the contents of the PLA. Microcode can be contained in the on chip ROM or in RAM. MPSs become user microprogrammable by allowing microcode to be contained in the RAM.

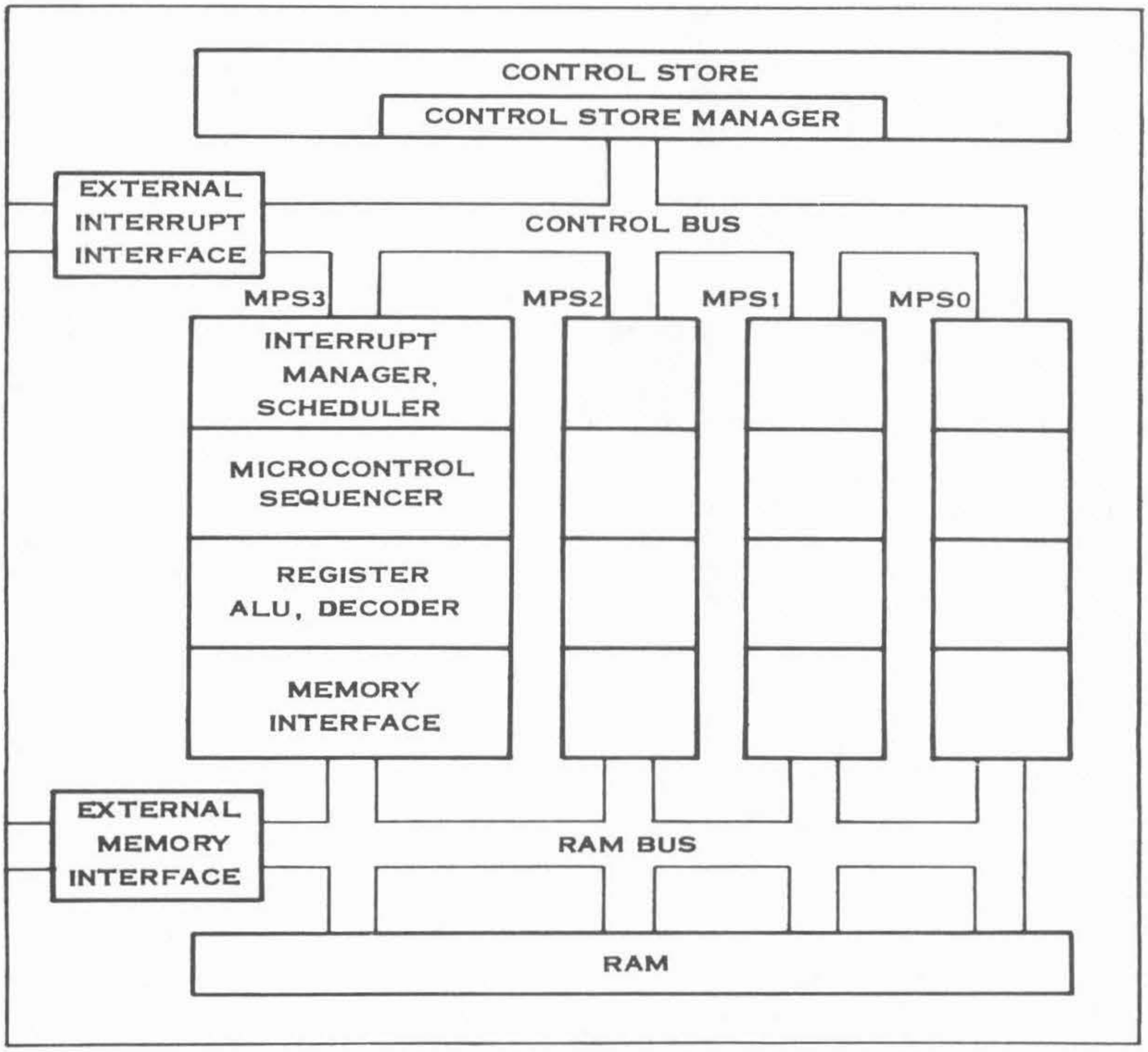

FIGURE 2. Restructurable IC Block Diagram

The RIC also provides for coordination of MPSs contained across multiple RICs. The concept of external coordination of microprocessors has been developed in (1,2). The coordination of MPSs on multiple chips is accomplished with status ports. The status port contains four signals for ALU status, a carry chain, a shift/rotate linkage and a synchronization signal. The external status port interconnection to internal MPS structures is programmable. The use of the RIC status ports provide for implementing computers with word widths greater than 64 bits. Also the status ports provide for flexibility in implementing pipelines with multiple RICs. The external interface of the RIC provides the capability for coordinating independent processors implemented on multiple RICs.

The RIC external interface is composed of three sections: a data port section, a status port section and an interrupt port section. There are two 16 bit wide data ports. The 16 lines are bidirectional and carry addresses and data. Each status port has a general purpose arbitration mechanism. One of three arbitration modes can be used: round robin, master-slave, or a general arbitration method implemented by external hardware. These arbitration methods allow a great amount of flexibility in the communication between RICs, memories, and I/O devices. The interrupt port on the RIC uses the same arbitration methods as the data port. Thus the interrupt port has great flexibility in the topology of the interrupt network. The interrupt port has flexibility in determining the information protocol. The interrupt port can be used for a range of applications from conventional receive only vectored interrupts up to a user defined interrupt driven interchip communication.

The internal memory system of the RIC supports the internal structures of MPSs. There are four memory modules. The memory modules can be accessed in parallel when each MPS accesses its own memory module. Also, the internal RAM bus allows the sharing of the memory modules among the MPSs. The internal RAM bus also can be structured to support pipelined MPSs. In the pipeline structure, each MPS can send and receive data simultaneously. The internal memory system has a capability for memory mapping. Memory mapping allows the internal RAM to be used as a cache or a member of a virtual memory hierarchy.

In the following section, the restructuring capability for the internal coordination of the multiple processors contained in a RIC is discussed. This will be followed by a discussion of configurations of multiple RICs. Then the processing element in the RIC will be described followed by a description of the internal RAM system and the external interface.

2. SINGLE-CHIP CONFIGURATIONS

The MPSs of a single RIC can be configured into several modes. The three basic MPS structures are: independent processors, locksteped processors and pipelined processors. The configuration used is determined by directing the microinstruction stream. In the independent and pipelined configurations, each MPS receives its own microinstruction stream. In the lockstep configuration, all MPSs in the lockstep receive the same instruction stream. The various MPS structures are discussed below.

In the independent mode there are up to four independent instruction streams on a RIC chip. They operate on four different data streams. The data streams can be completely independent, or they can communicate with one another by passing messages through memory (on-chip or off-chip RAM). Figure 3 illustrates this configuration.

The internal lockstep mode uses a single micro instruction stream to control multiple MPSs. This mode of operation allows a wide word (as wide as 64 bits when all MPSs have the same instruction stream) machine to be designed. Operation in this mode is similar to today's bit-sliced microcomputers (3,4) and it is shown in Figure 4.

The pipeline mode uses multiple instruction streams and multiple data streams. Each MPS implements one stage of the pipeline. The data normally flows unidirectionally between neighboring MPSs. For example, as shown in Figure 5, MPS 3 can be programmed to prefetch the machine instructions from the off-chip main memory, MPS 2 decodes the machine instruction, MPS 1 is programmed to perform address computation and fetches operands from the main memory, and MPS 0 is programmed to do computation specified by machine instructions.

In addition to the structures discussed above, various combinations of these configurations are possible within a single RIC chip. For example, MPS 0 and 1 form one internal lockstep, and MPSs 2 and 3 form another lockstep. The lockstep of MPSs 2 and 3 can emulate the Central Processing Unit (CPU) of a 32 bit machine, while the lockstep of MPSs 0 and 1 can be programmed as a graphics processor, making the system suitable for a high bandwidth graphics application.

3. Multi-chip Structures

Multi-chip configurations are used to achieve improved functionality and performance beyond that which is possible with a single-chip configuration. For example, a multi-chip structure may employ one or more RICs as the CPU of a machine, another RIC as an I/O processor, and another RIC as a floating point processor. In this section various multi-chip structures are described.

An external lockstep connects two or more MPSs, each on a different RIC, to

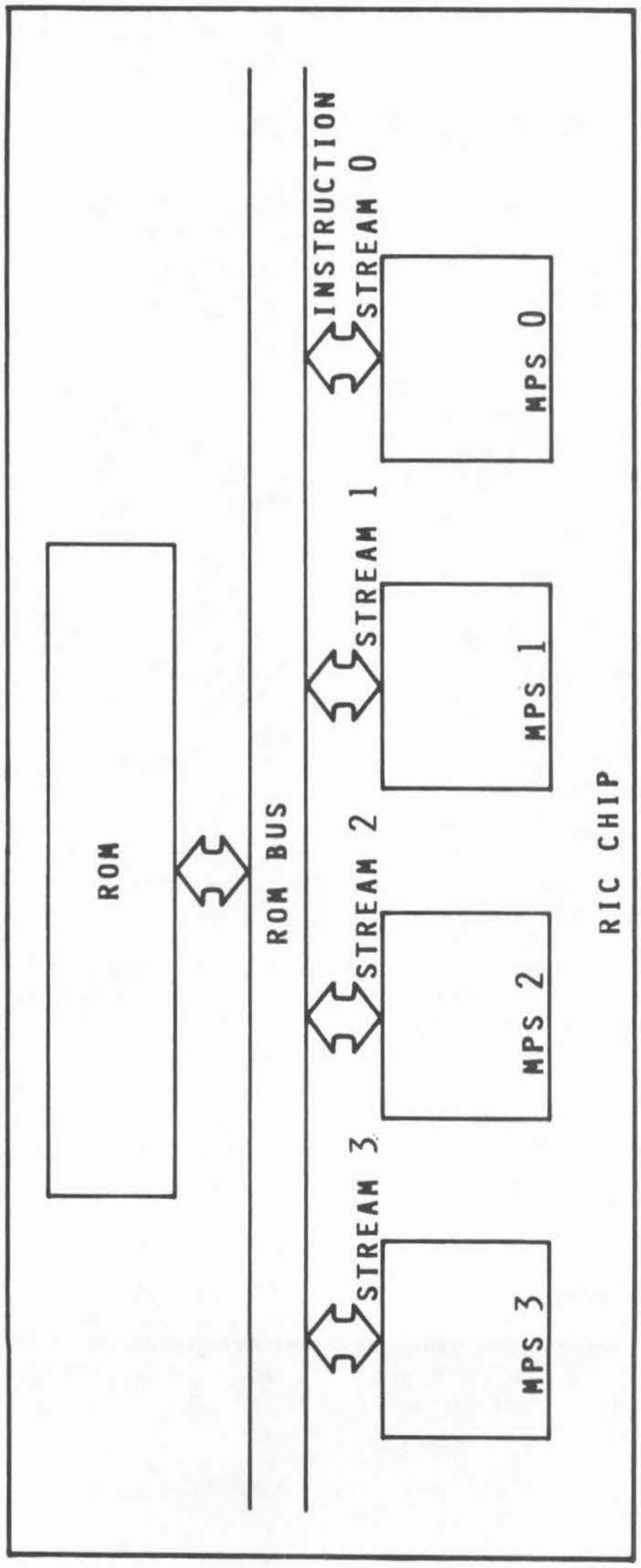

FIGURE 3 RIC INDEPENDENT MODE

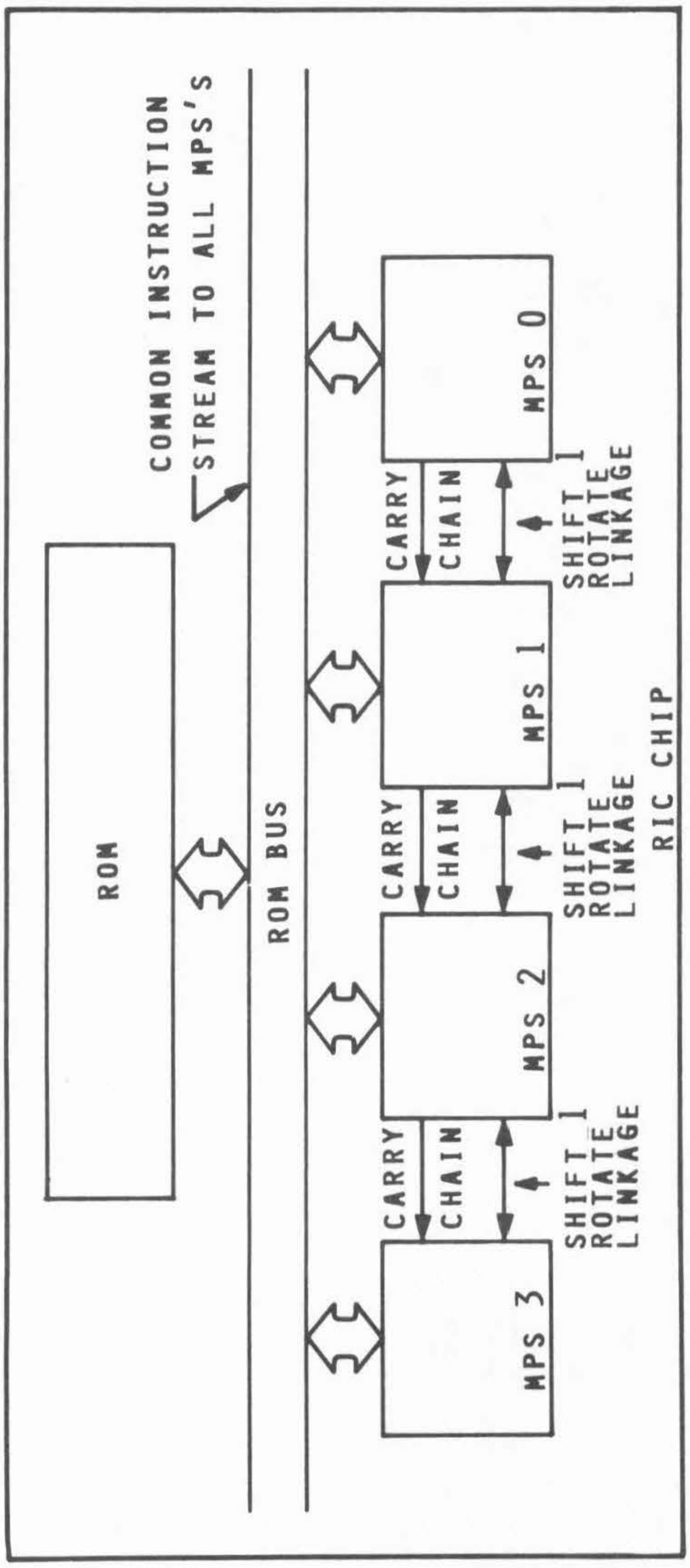

FIGURE 4 RIC INTERNAL LOCKSTEP MODE

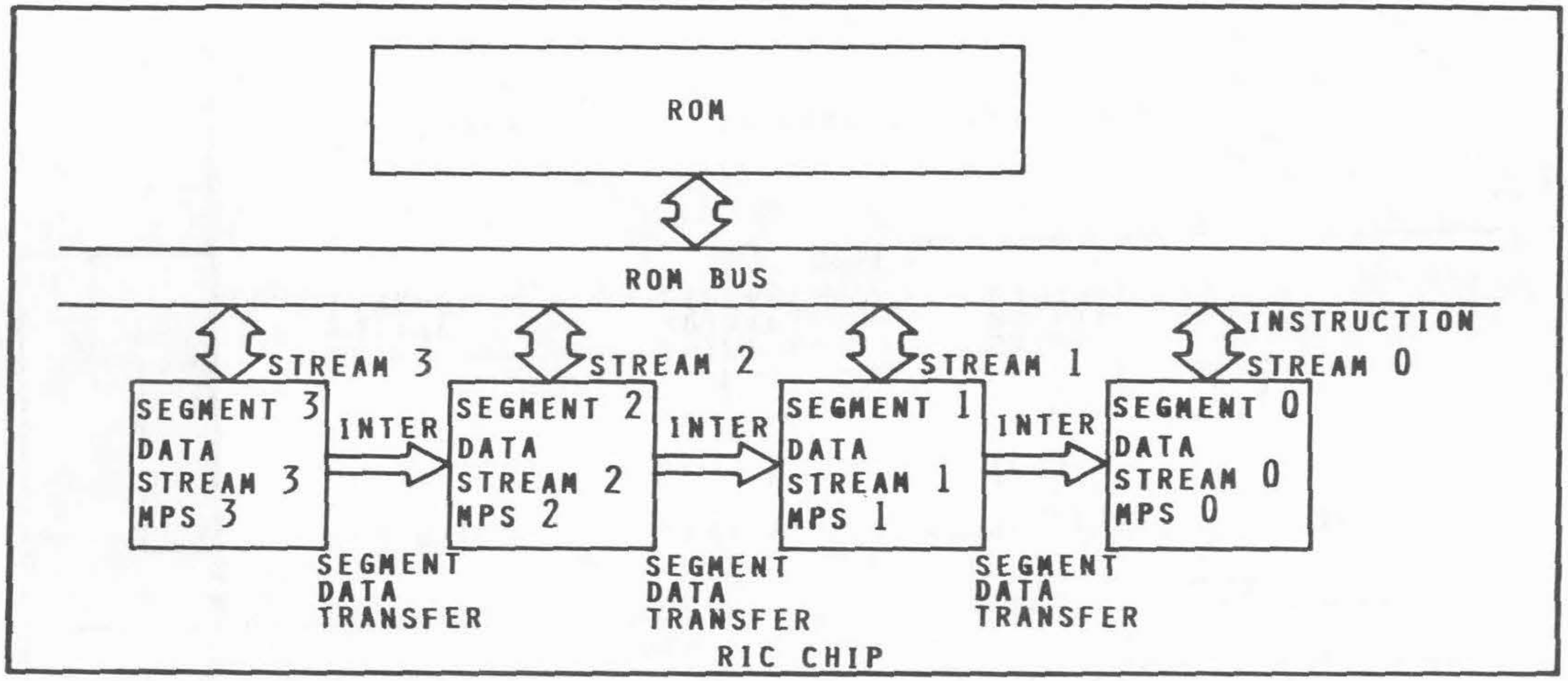

FIGURE 5 RIC PIPELINE MODE

form a lockstep. An external lockstep is illustrated in Figure 6. On each cycle, each MPS in the lockstep executes the same microinstruction, but operates on its own data stream. Since each MPS resides on a different chip each MPS has to fetch its own microinstructions. The status connection synchronizes externally locksteped MPSs so that they are executing instructions in unison. The status connection also contains ALU result status, a carry linkage, and a shift and rotate linkage.

The hybrid lockstep structure is a lockstep in which MPSs are locksteped together within one RIC as well as locksteped externally to MPSs on one or more other RICs. An example of a hybrid lockstep is given in Figure 7.

Two basic types of pipelines can be made with multichip structures. One type, the internal lockstep pipeline has each stage of the pipe formed with an internal lockstep of MPSs. This mode can be used for pipeline widths of up to 64 bits. If the pipe is required to be more than 64 bits wide, each stage of the pipe is formed with a hybrid lockstep. The second type of pipeline structure, the external lockstep pipeline, forms each stage of the pipeline with an external lockstep of MPSs.

In addition to the above structures, the RIC is designed so that combinations of the various internal and external configurations can be combined among various RICs.

4. MicroProgrammable Slice Design

The MPS is the processing element of the RIC. Each MPS contains six major blocks: the data path (computation hardware), the PLA for interpreting instructions for controlling the data path, the ROM address sequencer, the interrupt manager, the scheduler, and the programmable interconnect.

The data path in a MPS is 16 bits wide. It contains a dual port register file of sixteen 16-bit wide registers. The data path contains a high performance ALU. Two registers in the register file can be accessed from two 16 bit wide buses simultaneously. The data path also has a hardware unit to Shift, Extract, and Rotate data called the SERU. In addition to the usual shift and rotate operations in the data path, the SERU can be used to extract fields of a machine instruction being emulated and pass the extracted fields as parameters to the PLA generating control signals for the data path. This PLA also generates signals to coordinate the ROM sequencer. The ROM sequencer is used to generate addresses for the ROM. The sequencer provides for loop control, subroutine calls, branches, and repeating the execution of an instruction.

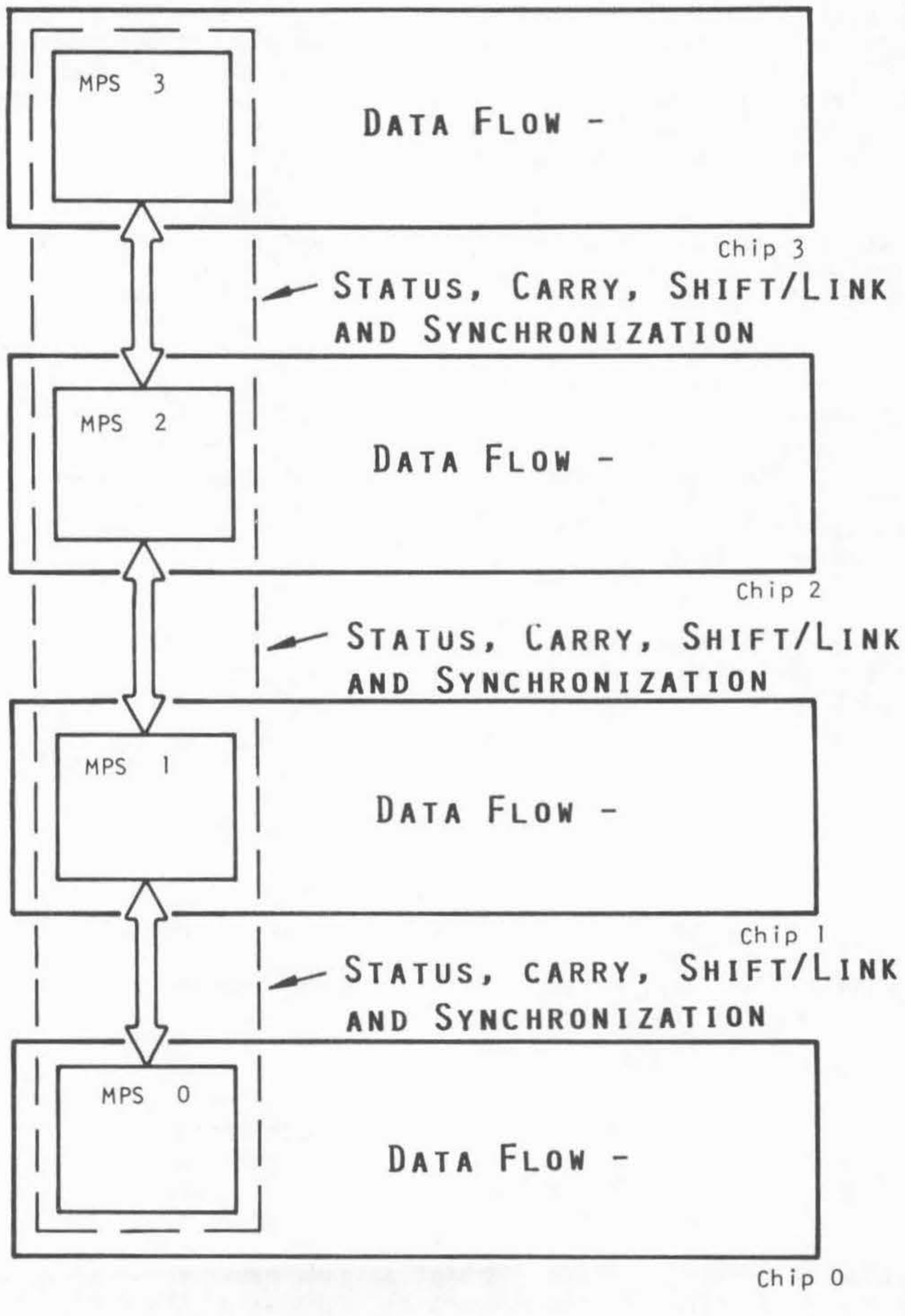
MPS 3
DATA FLOW -
Chip 3
STATUS, CARRY, SHIFT/LINK AND SYNCHRONIZATION
MPS 2
DATA FLOW -
Chip 2
STATUS, CARRY, SHIFT/LINK AND SYNCHRONIZATION
MPS 1
DATA FLOW -
Chip 1
STATUS, CARRY, SHIFT/LINK AND SYNCHRONIZATION
MPS 0
DATA FLOW -
Chip 0

FIGURE 6. External Lockstep

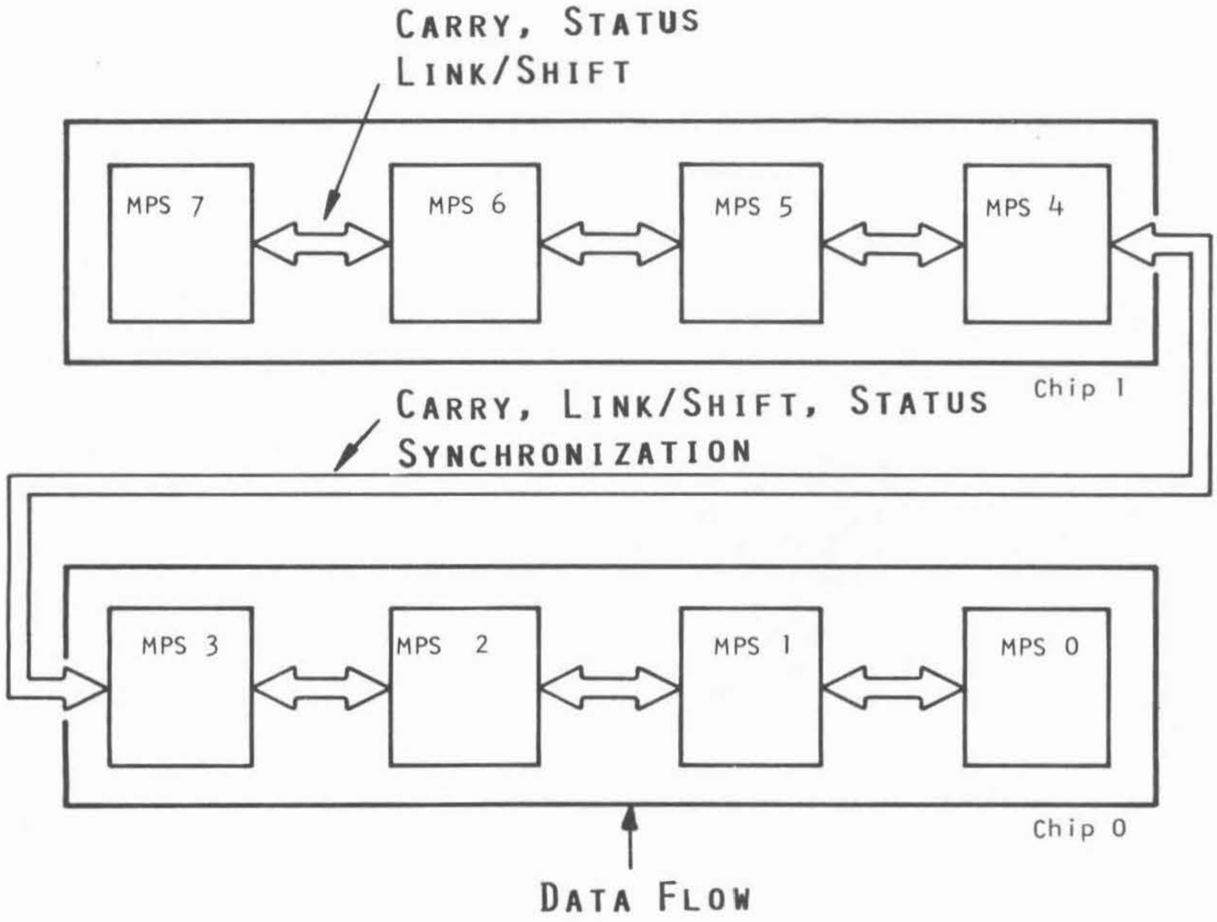

FIGURE 7. Hybrid Lockstep Forming a 128 Bit Lockstep

The MPS is made restructurable to an architecture through four techniques:

1 ROM programming

2 PLA code stored in each MPS

3 Interrupts at the microcode level

4 Programmable interconnect

The MPS is designed to interpret vertical microcode or machine code (assembly code) using the PLA. Microcode and low complexity machine code instructions are interpreted directly through the PLA. More complex machine code instructions are interpreted in terms of microinstructions or microroutines. The instructions that are interpreted by an MPS can be contained either in the on-chip ROM, in the on-chip RAM or in an external RAM. This feature provides for user microprograms to be contained in either ROM or RAM. The ROM can contain system microprograms for a variety of tasks: control programs for interrupts, internal memory management, self testing, initiating internal MPS structures, and initiating external RIC structures. The ROM can also contain microprograms for interpreting machine languages. A RIC can interpret multiple microcode languages and/or multiple machine languages. A PLA within an MPS is programmed to interpret a particular language. This PLA can be programmed to interpret more than one language, depending upon the languages. Since there are four MPSs on a RIC, at least four different languages could be interpreted. It is expected that a RIC will interpret existing languages (emulation) as well as interpret languages created for a particular application. For example, a language could be created for: instruction prefetch, instruction decode, address calculation, self testing, memory management, or any other computation task.

The information in the ROM is accessible to all MPSs through a shared ROM bus. A centralized ROM is used because this method allows easy code sharing and maximum flexibility in the amount of code that can be dedicated to an MPS compared to a separate ROM for each MPS. The bus is arbitrated in a round robin scheduling discipline. When an MPS issues a ROM access, the ROM manager buffers the tag(s) associated with the sending MPS. The ROM manager also routes the microinstructions to the appropriate MPS(s) using the tag(s). Multiple tags are sent by an MPS when it is in an internal lockstep mode, and the same microinstruction is routed by the ROM manager to all MPSs involved in the lockstep. The ROM bus is also used to send interrupts.

There are two basic categories of interrupts: internal and external interrupts. Internal interrupts are sent between MPSs within a single RIC. External interrupts are sent between an MPS on one RIC to one or more MPSs on one or more other RICs. Each MPS has an interrupt manager. The interrupt manager sends and receives both internal and external interrupts. The interrupt manager gains control of the ROM bus to send an interrupt. The interrupt manager sends the following information: identification of the interrupt source MPS, the destination MPS(s), the priority of the interrupt and run time information. An internal interrupt is sent to multiple MPSs to initiate a lockstep process or a pipeline process. The receivers of an interrupt respond as to whether the interrupt is to be immediately acted upon or not. An external interrupt is sent to the external interrupt manager. The external interrupt manager uses the priority of the interrupt to access a message block which is sent to external RICs and or other interruptable devices.

A process within the RIC is initiated by an interrupt. A process is defined as an instruction stream. An instruction stream is composed of microinstructions, macroinstructions or a combination of the two. Each process has a priority associated with it. Within a RIC there are 256 priority levels. The priority of a process and the priority of the interrupt which initiates this process have the same value. When an MPS receives an interrupt, its interrupt manager compares the received interrupt priority with that of the currently executing process. If the interrupt manager determines that the interrupt priority exceeds the current process priority, the interrupt manager signals that this interrupt process will cause a context switch. Otherwise, the interrupt manager indicates that the interrupt process priority is of lower priority. This type of feedback from the interrupt receiver to the sender is needed in the case of multiple receivers. If only a subset of the receivers can perform a context switch, MPSs would be idled unnecessarily while waiting for other MPSs to finish their higher priority process. If only a subset of the multiple receivers of an interrupt can perform a context switch, the interrupt is withdrawn and sent again later. This scheme prevents deadlock and unnecessary idling of resources.

If an interrupt is sent to a single receiver and the interrupt is of lower priority than the current process, the interrupt is buffered by the receiver MPSs scheduler. The scheduler buffers interrupts by priority in a 256 bit shift register. When a process is active, the scheduler scans through the shift register to find the process with the next highest priority. When the current process is finished or timed out, the scheduler uses the priority of the next highest priority process to access a table which contains a pointer to the process's context.

Programmable interconnect is used for routing the carry chain, the shift/rotate linkage, and the ALU result status flags. The routing of these signals depends upon the single or multi-chip structure being used. In the following, only the programmable carry chain is discussed. The carry-chain is unidirectional in nature flowing from MPS 0 to MPS 3, and then looping back to MPS 0. The carry routing logic is also responsible for handling carry in and carry out signals in the external and hybrid lockstep modes. The routing logic is expected to be implemented with pass transistors and is set up in a particular mode at the beginning of a structure by appropriate signals from the PLA, and remains set up that way until the next restructuring. For example, in the independent mode where all four MPSs are working as four independent processors, the routing logic isolates the carry chain into four independent segments. In the internal lockstep mode the routing logic establishes a separate carry chain for each lockstep on the chip. Figure 8a shows the carry chain when MPS 1 and 2 are working in a lockstep, and MPS 0 and 3 are working as independent processors (one's complement arithmetic is used requiring the end around carry). Figure 8b shows the carry chain when all four MPSs are involved in four different external locksteps. Programmation of the shift/rotate linkage and the ALU result status signals are similar. The ALU status signals differ slightly in that these signals from locksteped MPSs are individually connected to a bus using a wired-AND configuration.

5. INTERNAL RAM

The internal RAM of the RIC is organized as four independent memory modules that are byte addressable. An NMOS RIC with a minimum geometry feature of one micron (lambda equals .5 micron) could contain about 16-32K bytes of dynamic RAM. This RAM would occupy about one-third to one-half of the chip area. The internal RAM subsystem of the RIC includes four independent memory modules and a data bus interconnecting the RAM to the four MPSs. The data bus is designed to support the three basic internal structures of MPSs: independent, lockstep and pipeline. The memory subsystem also contains a memory mapper to automatically direct memory accesses to internal locations if the data is resident internally or to external locations otherwise. The memory subsystem is illustrated in Figure 9.

The data bus supports four concurrent accesses to memory, provided there is no interference between processors and memory. This bus allows a direct path from each MPS to its own memory module. When each MPS accesses its own memory module, then four simultaneous memory accesses can occur. If MPSs access memory modules other than their own, these memory accesses may result in memory interference, since a shared bus is used and multiple MPSs may access the same module resulting in queued memory requests. As shown in Figure 9, each memory module has a Memory Scheduling Unit (MSU) and a Bus Control Unit (BCU). When an MPS accesses its own memory module, it is directly connected through its BCU to its MSU. The MSU indicates whether there are pending memory requests or not. If there are no pending memory requests, the access occurs immediately. If there are accesses pending, the MSU queues a tag indicating the MPS which requested memory service. An MSU queues an MPS request with a first come first served scheduling discipline. When the MPSs request reaches the head of the queue, the MSU signals this to the MPS. The MPS reissues its request and the memory access is performed immediately. When an MPS accesses a memory module other than its own, the BCUs are configured to make the connecting bus a shared bus, as shown in Figure 10a. The MPS first waits for access to the shared bus. The shared bus is scheduled by a round robin by demand discipline where the first MPS or memory module gets access to the bus in round robin order. After an MPS gains access to the bus, it sends the memory information and a destination tag indicating the destination memory

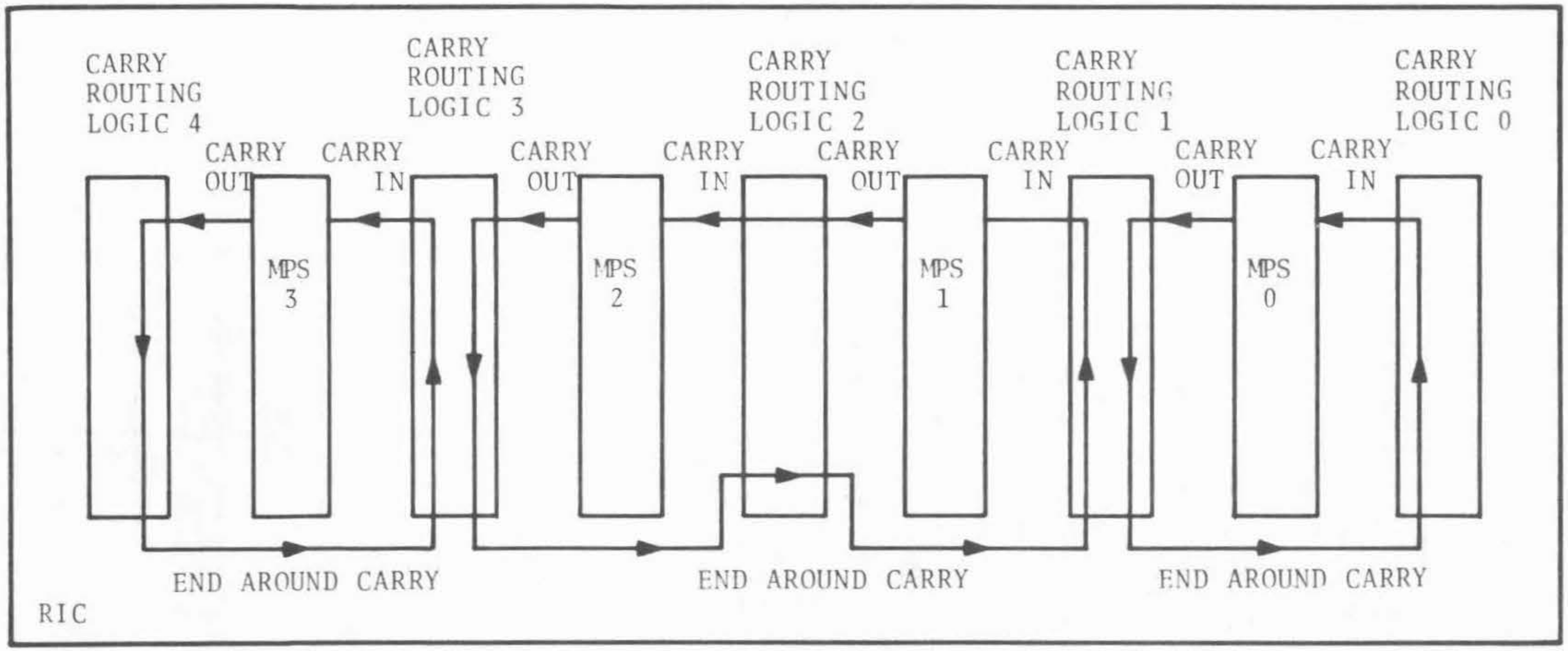

FIGURE 8A. Carry chain configuration when MPS 1 and MPS 2 are working in an internal lockstep and MPS 0 and MPS 3 are working as independent processors.

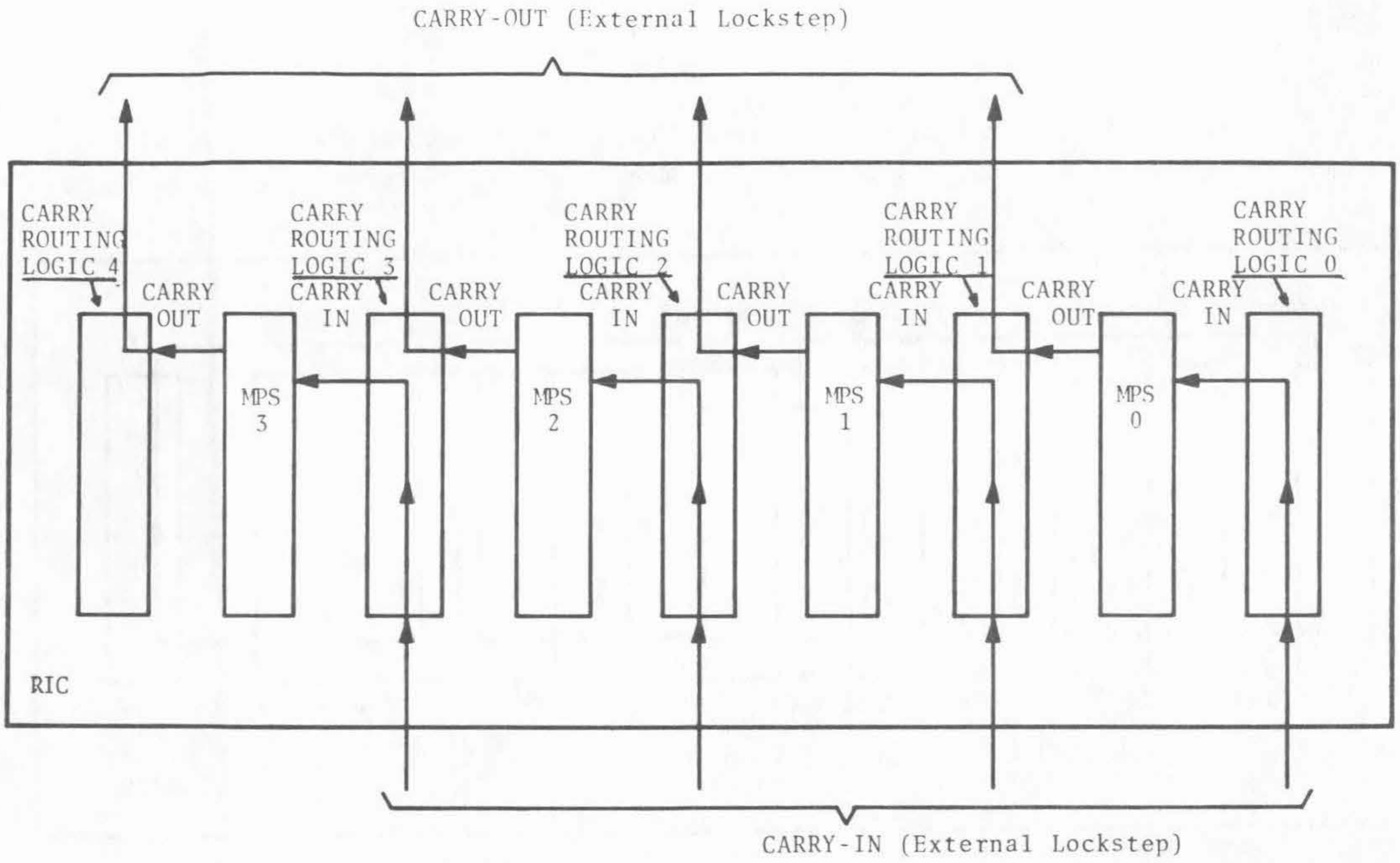

FIGURE 8B. Carry chain configuration when all 4 MPS's are working as independent processors.

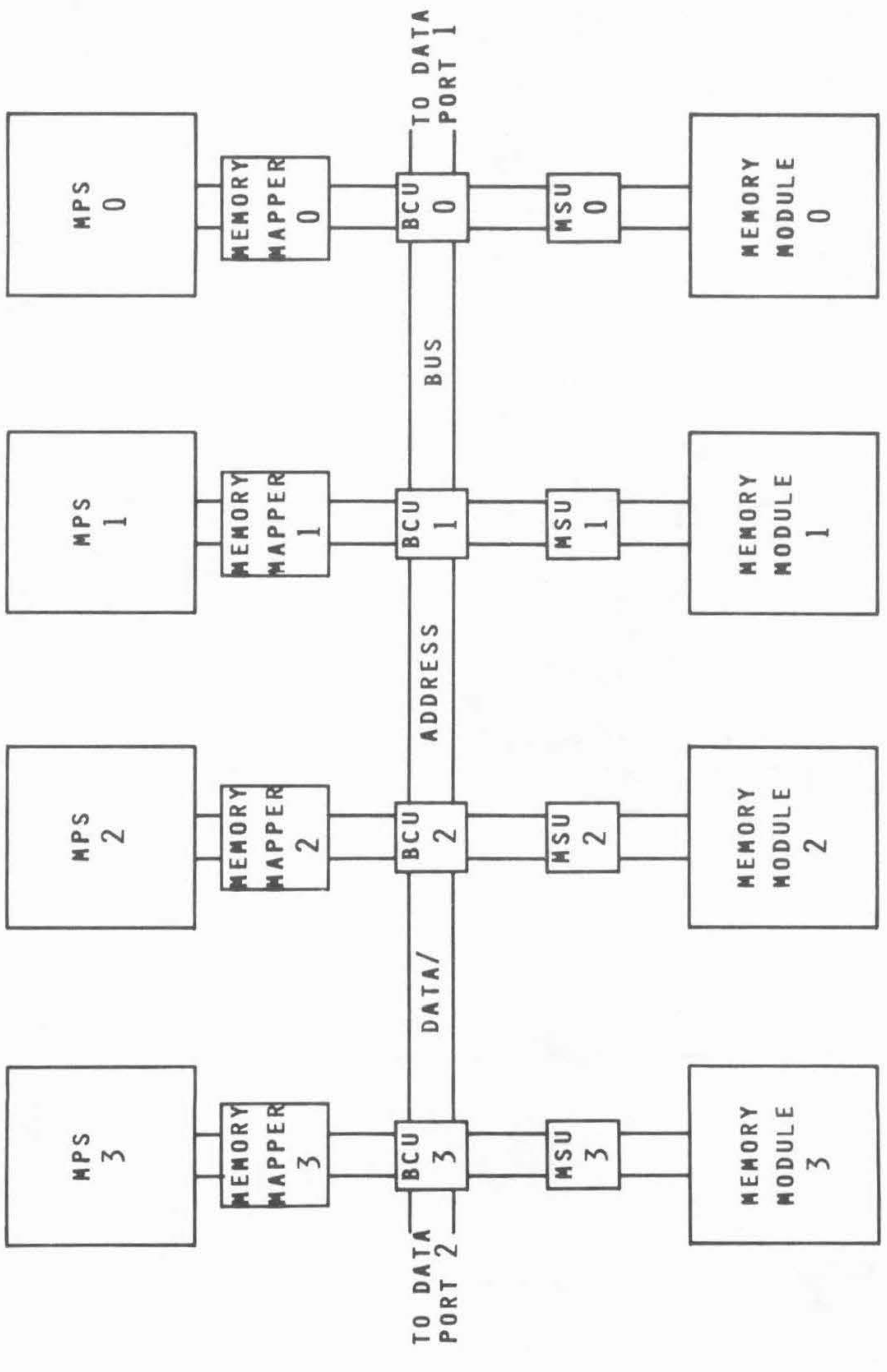

FIGURE 9 RIC RAM SYSTEM

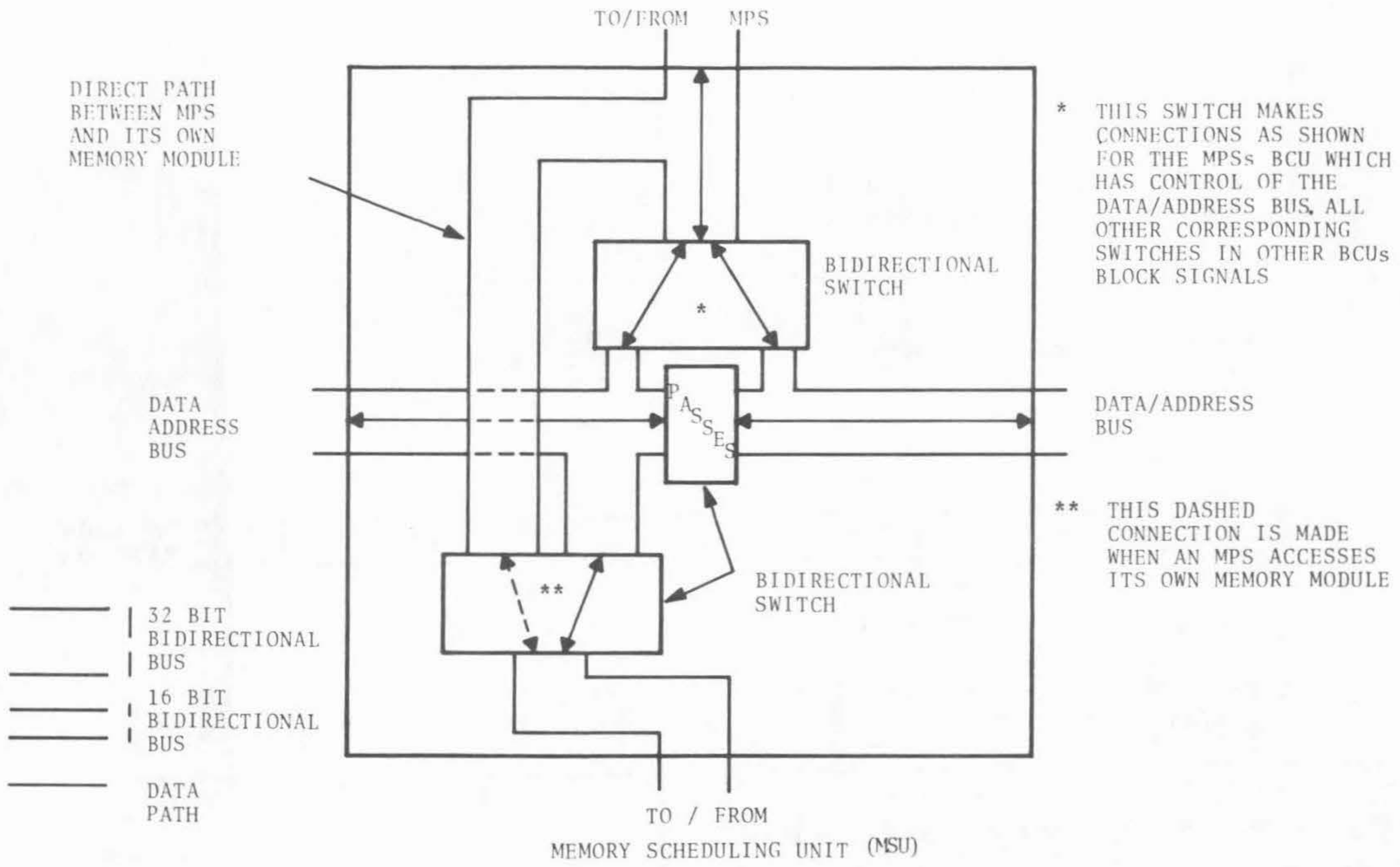
TO/FROM MPS
DIRECT PATH BETWEEN MPS AND ITS OWN MEMORY MODULE
* THIS SWITCH MAKES CONNECTIONS AS SHOWN FOR THE MPSs BCU WHICH HAS CONTROL OF THE DATA/ADDRESS BUS. ALL OTHER CORRESPONDING SWITCHES IN OTHER BCUs BLOCK SIGNALS
BIDIRECTIONAL SWITCH
*
DATA ADDRESS BUS
PASSES
DATA/ADDRESS BUS
** THIS DASHED CONNECTION IS MADE WHEN AN MPS ACCESSES ITS OWN MEMORY MODULE
**
BIDIRECTIONAL SWITCH
32 BIT BIDIRECTIONAL BUS
16 BIT BIDIRECTIONAL BUS
DATA PATH
TO / FROM
MEMORY SCHEDULING UNIT (MSU)

FIGURE 10A. BCU Configuration For Shared Data/Address Mode

module. The destination module sends the memory requests queue position on a separate bus. If 00 is sent the memory request is being processed immediately. Otherwise the two bit number indicates the number of pending requests before this current one. Any memory module can have at most four memory requests pending since an MPS can only have one memory request pending at a time. Each MPS has circuitry to monitor the bus. When the memory module that has an MPSs memory access pending completes a memory access, that MPS decrements the number of pending requests by one. When an MPS decrements this number to be zero, it means that its request is at the head of the queue at the memory module where this MPSs memory request is pending. When this MPS gains control of the shared bus, it reissues its request and the request is processed immediately. The operations described above support the memory accesses made by independent MPSs, locksteped MPSs and pipelined MPSs.

The RIC memory system supports locksteped and pipelined MPSs. Locksteped MPSs can make simultaneous requests to their own memory modules. It is possible that locksteped MPSs will not receive their request for memory service simultaneously because the queue length at one MPSs memory module could be different than the queue length at another locksteped MPSs queue. Locksteped MPSs are synchronized to avoid this problem. When locksteped MPSs make a memory request, a wired-AND line which connects all MPSs in the lockstep is pulled low by each MPS. After each MPS has had its memory request serviced, it discontinues pulling this line down. When the last MPS has its memory request finished the line will rise to a logic one, indicating that the lockstep process can continue. Also locksteped processors can access memory modules other than their own. In this case each locksteped MPS would issue its request when it got access to the bus. The locksteped memory access would be synchronized as above.

In addition to accessing memory, pipelined modes also use the data bus to send data between other MPSs in the pipe. Figure 10b shows the BCU configuration for pipelined data transfers. For this BCU configuration, the data bus is segmented to allow all adjacent MPSs in the pipe to transfer data in parallel, including transfers to MPSs on different RICs.

Each memory module is addressed with a 16 bit address. This allows for eventual growth of up to 64K bytes of directly addressable space for each of four MPSs. However an MPS supports two types of addresses: 16 and 32 bits. Sixteen bit addresses are used to directly access an MPSs own memory module. Thirty-two bit addresses are used to access other memory modules or external memory. In the case of accessing other memory modules, the most significant 14 bits are a tag indicating that the address is for an internal memory module. The next two significant bits select one of four memory modules. The remaining 16 bits point to an address in an internal memory module. If a 32 bit address does not point to an internal memory module directly it can either be an external address, or it can be a mapped address. A 32 bit address is mapped or external depending upon MPS control. If the address is designated to be an external address, it is sent to the external memory interface for processing. Otherwise, it is sent to the memory mapper. The memory mapper

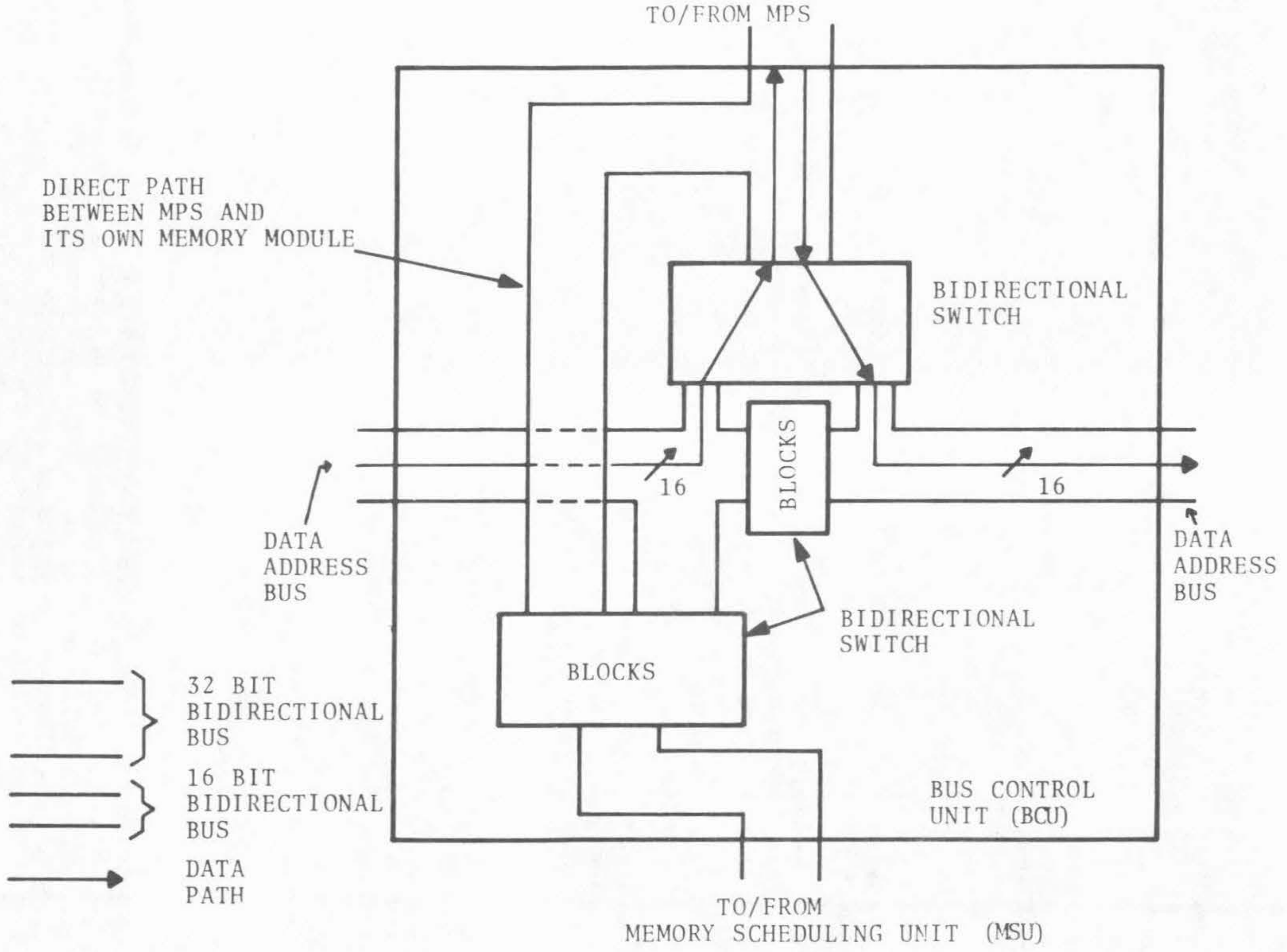

FIGURE 10B. BCU configuration for the pipeline data transfer mode.

uses an associative search to determine if the address is internal or external. If it is internal the associated internal address is sent to internal memory. If the address is external it is sent to the external memory interface.

6. EXTERNAL INTERFACE

The external interface for the RIC is designed to support multiple RIC configurations, interchip communication and data path communication between system memory and system I/O. Two versions of pin assignment are planned. An 82 pin version has two 16-bit data/address ports. A 114 pin version is the same as the 82 pin version except that it has two 32 bit data/address ports. The 82 pin RIC is discussed below.

In Figure 11 the RIC pin assignment is illustrated. There are five types of pin functions for the RIC: data/address, control, interrupt, status, and power/clock. The number of pins dedicated to each function group is listed in Table I.

Table I

FUNCTION	NUMBER OF PINS
data/address (2 ports)	50
control	2
interrupt	8
status (2 ports)	18
power/clock	4
	82

Pin Assignment By Groups

6.1. DATA PORT

The 82 pin version of the RIC has two 16-bit data/address ports. Each port has 16 bidirectional lines for carrying data and addresses. Associated with each port is a pair of handshake signals for gaining control of a shared

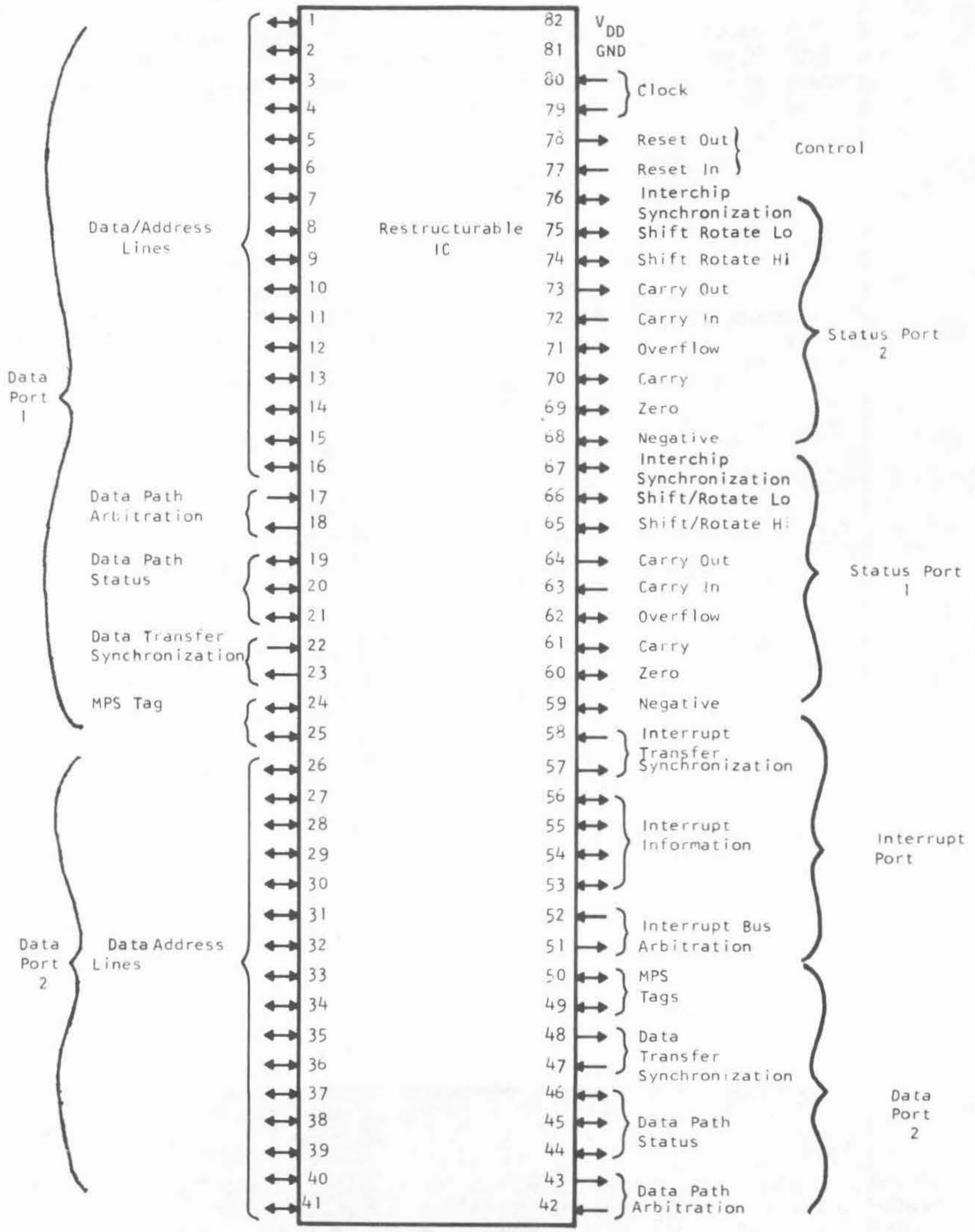

FIGURE 11. A Restructable IC Pin Assignment

resources. The arbitration method for the shared resources is either round robin by demand, master-slave, or determined by external circuitry. The round robin and master-slave arbitration methods support a shared bus, while the external arbitration circuitry supports a network with a general topology. Each port also has a pair of signals for synchronizing the sending and receiving of data and addresses on the bus. Also, each port has a bidirectional set of three signals to indicate bus status. Four types of read/write operations are indicated by these three signals. The four operations are: access a user specified RIC, access system RAM, access system I/O, and access the resource whose destination address is sent at the beginning of the access. Finally each port has a bidirectional pair of MPS tag identifiers. The MPS tag identifiers are used to indicate the source MPS at the sender and/or the destination MPS at the receiver. The two data/address ports are independent. However, these two ports can be combined into one port by internally performing the same operation to both ports concurrently, and externally treating the two ports as one port.

6.2. STATUS PORT

There are two identical status ports. The status port's main function is to provide the signals to lockstep two MPSs on different RICs. Status port 1 can be used to lockstep MPS 0, MPS 1 or a lockstep of MPSs 1 and 0 to external MPSs. Status port 2 can be used to lockstep MPS 2, MPS 3, or internal locksteps including MPS 2 and/or MPS 3 to external MPSs. The use of the status ports is illustrated in Figure 6. A 64 bit wide external lockstep is formed with 4 MPSs on four RICs in figure 6. There are four pin functions in each status port: ALU result status, carry linkage, shift/rotate linkage, and MPS synchronization.

There are four ALU result status pins. These are: the Negative result status, N; the Zero result status, Z; the Carry result status, C; and the oVerflow result status, V. These four signals connect to a bus using a wired-AND configuration. This bus connects all externally locksteped status ports. These signals are encoded to indicate up to one of 16 ALU result outcomes. The carry linkage is a carry-in signal and a carry-out signal. The carry-out signal of one RIC is connected to the carry-in signal of the next most significant RIC. The shift/rotate linkage is used to perform shift operations between externally locksteped MPSs. The shift/rotate hi signal of a RIC is connected to the next most significant RICs shift/rotate lo signal. The shift/rotate hi signal of the most significant RIC is connected to the shift/rotate lo signal of the least significant RIC to provide the shift/rotate linkage. The MPS synchronization pin ensures that externally locksteped MPSs are executing the same instruction in phase. Without synchronization, MPSs in an external lockstep can get out of phase because other MPSs on a RIC may be operating independently of the externally locksteped MPSs. Thus the time to fetch a microinstruction may vary among the RICs containing locksteped MPSs. The MPS synchronization pin serves as a flag to indicate that each MPS has finished the previous instruction and has fetched the next microinstruction and is ready to execute it. The MPS synchronization pins are wired together in a wired AND configuration. When all externally locksteped MPSs are ready to execute the next instruction, the MPS synchronization line will be high. If one or more MPSs is not ready, the line will be pulled low. When the MPS synchronization line is high, execution begins on the next clock cycle. (All RICs with MPSs in a common external lockstep must use the same system clock.) Shortly after execution begins, the MPS synchronization line is pulled low and it stays low until all MPSs are ready to execute the next instruction.

6.3. INTERRUPT PORT

The interrupt port serves two purposes. The first purpose is to receive and process interrupts in a manner similar to conventional microcomputers and microprocessors. The second is to provide for interchip communication. The interrupt concept has been generalized to include the capability to send interrupts to other receivers, providing for interchip communication. The purpose of interchip communication is to coordinate RICs to a task, to initiate a task, and to transfer information. The interchip communication system is used to transmit commands and/or small amounts of data. The bulk data part of an information transfer is communicated between memories. For example, a disc read operation is initiated by using the interrupt port of a RIC to send commands to a disc controller. The data transfer is accomplished on a separate data path between the disc system and the memory system. The interrupt port contains pins for arbitration, information and data transfer synchronization.

The interrupt port of the RIC has 8 pins. Two of the pins are used for arbitration of shared resources used during the sending of an interrupt. The same three arbitration modes used for the data ports are also used for the interrupt port: round robin, master-slave, or a general arbitration method.

Four pins of the interrupt port are dedicated to data transfer. The data protocol has minimal specification with maximal user definition. In the interchip communication mode, the first information sent on these pins is an address. The length of the address is designated by the user. When an interrupt is sent, all chips on a common interrupt bus receive the address and store it. The status signals indicate whether the information lines carry address or data. The receiver buffers the address portion as long as the status indicates address bits are being sent. After the destination address has been sent, each receiver uses the address to access a bit in the chips RAM to determine if this chip is an intended receiver of the interrupt. In the conventional interrupt scheme, the first information sent is the interrupt level. The remaining two pins of the communication port are used for interrupt bus status. The four status values are: sending address, sending data, data/address nibble received, and interrupt information transfer completed.

An interrupt is sent by first gaining control of the interrupt bus. After gaining control of the bus, the interrupt data is sent. The amount of data that is sent is determined by the user. The hooks have been provided to send an optional destination address of variable length, a variable length data portion, and an optional source address of varying length as the components of the information sent during an interrupt. The interrupt information is buffered at the destination by the RICs external interrupt manager. The external interrupt manager interrupts the destination MPS and passes it the length of the message and a pointer to the interrupt message block.

6.4. CONTROL LINES

There are two control lines, the Reset In (RI) and Reset Out (RO). The RI and RO signals from all RICs are connected together. RI signal is active high. When the RI signal is raised to a 1, the RICs begin to initialize themselves for operation. The RO signals are wired together in a wired AND configuration. When a RIC has completed the initialization operation, the RO signal which had been pulled low is allowed to float. When all RICs have completed initialization, the RO signal will be high indicating that the system has finished initialization.

6.5. POWER/CLOCK

The RIC will use two power pins: +3 volts and ground. The RIC will use an on-chip clock generator. This will allow a crystal to be placed across the two clock inputs, or an external clock can replace the crystal.

7. CONCLUSION

The RIC has been described at a high level and many details have not been included. The major goal of the RIC is the achievement of a highly flexible part that can be used to achieve a wide variety of specific hardware designs through programmation. The designed-in flexibility of the RIC provides for this programmation. The flexibility within the RIC includes: user definable micro language and assembly language, user programmable microcode, dynamic coordination of multiple internal processors, coordination of processors on multiple RICs, internal memory that can be used either as a caches or as an element of a virtual memory hierarchy, general topology for interchip communication and external data paths and a user definable interrupt mechanism.

REFERENCES

1 R. G. Arnold, and E. W. Page, A hierarchical, restructurable multi-microprocessor architecture, 3rd Annual Symposium of Computer Architecture, pp 40-45. 1976

2 S. I. Kartashev and S. P. Kartashev, A multicomputer system with dynamic architecture, IEEE Trans. Comput., vol. C-28, pp. 704-720, October, 1979.

3 Bipolar Microcomputer Components Data Book, Texas Instruments Inc.

4 The Am2900 Family Data Book, Advanced Micro Devices Inc.