A GENERAL MULTI-MICROPROCESSOR INTERCONNECTION MECHANISM FOR NON-NUMERIC PROCESSING

Hoo-min D. Toong
Svein O. Strommen*
Earl R. Goodrich II

February 1980

CISR No. 53
Sloan WP No. 1113-80

Center for Information Systems Research

Massachusetts Institute of Technology
Sloan School of Management
77 Massachusetts Avenue
Cambridge, Massachusetts, 02139

A GENERAL MULTI-MICROPROCESSOR INTERCONNECTION MECHANISM FOR NON-NUMERIC PROCESSING

Hoo-min D. Toong
Svein O. Strommen*
Earl R. Goodrich II

February 1980

CISR No. 53
Sloan WP No. 1113-80

Reprinted from Proceedings of Fifth Workshop on Computer Architecture for Non-Numeric Processing.

*Work performed while on leave from THE CHRISTIAN MICHELSEN INSTITUTE, Bergen, Norway.

ABSTRACT

MMPS interconnection problems are discussed in terms of a single time-shared bus. The performance drawbacks usually associated with the single bus alternative are attributed to the high bus utilization at the basic building block level. The Pended Transaction Bus protocol is presented as a general solution to such utilizations. Such a bus is developed to support more than 50 processors without severe contention. The basic protocol of the MC68000 as a current generation microprocessor is investigated, and shown ineffective for true multi-microprocessor systems.

1. INTRODUCTION

Gains in computer performance can be achieved through improvements at the circuit level (eg. faster circuitry), the basic building block level (eg. more powerful microprocessors), the building block interconnection level (eg. better computer system architecture), and the system software level (eg. more effective system software). Many of these points are studied in [1]. However, there are strong relative dependencies between the levels (see Figure 1), and a full system utilization of improvements at one level will usually require some associated modifications at the other levels. The absence of both necessary system interconnection signals and important system software instructions in modern building blocks are typical examples of situations where it is appropriate to make adjustments across level boundaries. Some of the dependencies and their associated modifications will be illustrated in this paper.

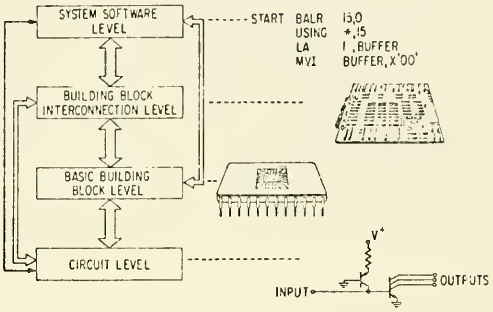

Figure 1
The four basic levels of potential Computer System Performance Improvement; i.e. circuit level, building block level, building block interconnection level, and system software level.

The recent LSI technology evolution has created significant improvements at the circuit level and the basic building block level. However, equivalent improvements have not yet taken place either at the building block interconnection level or at the system software level. Unfortunately, there is also still a lack of simple methods by which the necessary adjustments across the level boundaries can be handled in order to obtain optimum system utilization of the available technology; i.e., too much money is usually invested in solutions at the different levels when the actual changes are proposed.

73990?

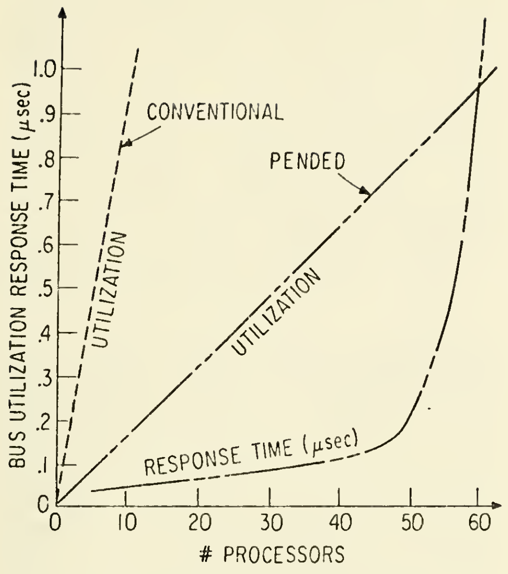

Figure 4
Average bus utilization and average bus response time as a function of the number of processors for a pended transaction based MMPS. The graph also includes the average bus utilization for a conventional single bus based MMPS. [15]

2.4 Memory and Processor Contentions

In a single bus based MMPS, any number of processors may simultaneously need information from the same memory module. In addition, there is likely to be an exponential-type distribution of the memory load [15]; i.e., certain areas of the memory will be in greater demand than other areas (see Figure 2). Since a memory module can only service one request at a time, this situation may result in a severe contention among the processors which are requesting the use of the highly demanded memory areas. Similarly, a set of memory modules may like to respond simultaneously to one particular processor. This creates a processor contention. The memory and processor contentions, which are often referred to as the "device busy" problem, may degrade the MMPS performance. This performance degradation will arise from processors awaiting crucial information from the highly demanded memory modules, and from the extra bus load which will be imposed by multiple requests for transmission to busy devices. Thus, the device busy problem must be dealt with in a single bus-based MMPS design.

The memory contention can be dealt with in two different ways. Danielsson and his colleagues [7] have suggested that the memory space should be divided into small modules. This will allow the memory requests to be spread out over a large number of independent modules, thus reducing the probability of getting simultaneous requests for the same module. Rather than using small memory modules, the problem can be solved using faster memory circuitry in the highly demanded areas. This concept must be accompanied by a memory content migration schema which must be based on continuous memory traffic statistics. The idea of using memory modules with different speed characteristics is analagous to the well-known cache concept; however, the shared-bus architecture is more flexible than a standard cache. Toong [15] has studied the memory speed part of this solution (assuming a stationary memory content), using an analytic model, and his results show promising effects.

No practical implementation of the above suggested solutions to the memory contention will eliminate the entire problem. The processors in the system will therefore continue to become unproductive when they must wait for crucial information from the memory. Note that only a portion of the delayed information will influence the processor productivity.

During the unproductive periods, the actual processors may waste bus bandwidth through repeated requests for the needed information. The waste of bus bandwidth can be reduced significantly by using input and output queues on all of the devices that are connected to the bus. This will allow all of the devices to transmit information on the bus even though the receiver should be busy; i.e., the information will be stored in the input queue until it can be processed. It will also permit the devices to keep on working even if they cannot get immediate access to the bus, i.e., the output information will be deposited in the output queue until it can be transmitted. In normal operation, the actual size of the queues, which is a system design parameter, is not likely to go beyond practical limits. According to Toong [13], who has studied both the memory speed and the queue solutions to the device busy problem, a 64-level queue will result in a queue overflow probability on the order of 10**-12, which is nearly zero for all practical purposes.

Design of a reliable single-bus-based MMPS, which would utilize queues of insufficient length to reduce the device busy problem, must incorporate mechanisms that will prevent queue overflow. A "queue-full" signal can be used to solve this problem. To avoid wasting bus bandwidth on queue-full conditions, it would be necessary to incorporate all the

The physical characteristics of the bus lines (i.e., speed, length, etc.) will definitely influence the bus performance. However, the effect of this matter is significant only for long busses (length > 1 ft.) and/or for critical speed requirements. For the purposes of this paper, the bus lines will be considered to be short enough and well conditioned so that they do not impose any significant data transfer constraints. The only remaining physical speed constraint, then, is the speed of the interface circuitry. This can be solved, for all practical purposes, by using high-speed, uniform logic at the bus interface (See Figure 3 for illustration).

The only remaining factor to consider, then, is the protocol used on the bus as a limit to data transfer rates. The high bus utilizations of the above mentioned processors is primarily a function of the master-slave based bus protocols that are used. Normally, the protocols are fixed at the basic building block design stage, and cannot be changed after the design is completed. Recent micro-coded processors, though, present the potential of being modified to optimize bus protocol without changing the basic processor. In general, the basic building block designers may use any bus protocol. The actual choice is always the result of a trade-off process, which, until recently, due to low-density devices, could not favor sophisticated bus protocols. Today, however, with VLSI technology at hand, it should be possible to implement low bus utilization protocols without any major penalties on the other system parameters.

Different versions of a special "split transaction" bus protocol, have been proposed by various authors as a general solution to the high bus utilization problem [3,6,7]. This type of protocol, which is illustrated in figure 3, splits the regular master-slave based transaction (Tstd) into two subtransactions (T1 and T2). These can take place disconnected in time as a transaction initiation part and a transaction completion part. Consequently, the bus will be free for other usages during the asynchronous wait interval (Mt). Obviously, this implies that a read transaction will utilize both of the subtransactions, whereas a write operation will utilize only the first subtransaction (T1).

The actual bus protocol implementation varies in two major areas. These are:

- centralized versus decentralized control, and
- synchronous versus asynchronous logic.

The trade-off process between centralized and decentralized control involves topics of reliability (fail-soft), modularity, and cost. The choice between synchronous and asynchronous logic is more likely to involve performance trade-offs. However, the differences between the various implementations has no major impact on performance, and a further discussion of these topics is not pertinent here.

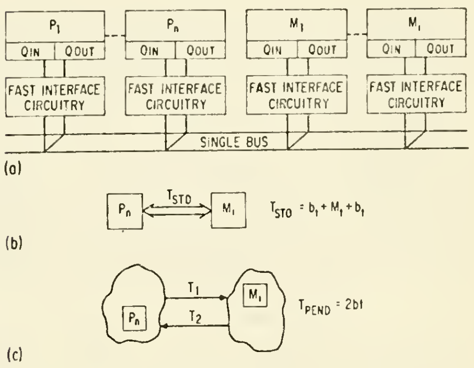

Figure 3
The Split Transaction Bus protocol concept. (a) The pended Transaction Bus protocol with I/O queues and fast interface circuitry, (b) the common Master/Slave transaction, and (c) the Pended Transaction concept. [15]

2.3 The Pended Transaction Bus Protocol

A split transaction bus protocol, which is called the "Pended Transaction Bus Protocol" (PTBP), is presently under investigation at M.I.T. This bus utilizes asynchronous logic and assumes infinite input and output queues on all of the devices that are using the bus. An analytic model [13] has been developed to study the average bus response time and the average queue length as a function of the number of processors (Pn). Figure 4 shows the average bus utilization and the average bus response time as a function of Pn. These results indicate that 50 processors can be connected to the single bus without any severe contention. For comparison reasons, the figure also shows the equivalent bus utilization for a conventional master/slave based MMPS.

A fully interlocked asynchronous version of the PTBP without queues has been implemented at MIT [3]. Danielsson, et. al. [7], and Lavington, et. al. [6], have both developed a synchronous split transaction based bus. However, none of these buses have been implemented with queues. The importance of queues will be illustrated later in this paper. Bus speeds between 60ns and 100ns have been achieved for the different buses. However, if these buses were implemented using today`s technology, they could all run at speeds of less than 30ns.

Performance has always been considered the only major drawback with the single bus. Interconnection techniques to overcome the performance problem have usually added to the system complexity, and thus corrupted the three aforementioned advantages. Two one-way paths and multiple two-way paths are among the most frequently suggested modifications in terms of performance gain. Note that these alternatives represent a bypass of the actual problem instead of a solution to it.

Danielsson, et. al. [7], have shown that the single bus represents only a minor performance degradation, when compared to a multiport system, as long as the bus is relatively much faster (a factor of 10x) than the devices that are using the bus. In other words, a low bus utilization at the building block level is necessary for successful operation of a single-bus-based MMPS. This fact is also expressed by Haagens [3]. The bus, as used herein, consists of the actual buslines, the immediate busline interface circuitry, and the bus protocol.

The above results imply that multiport based systems are probably the best alternative in applications where ultra-high bandwidths are required and where price is only a minor concern. On the other hand, however, the results also indicate that given the necessary low bus utilizations at the building block level, the single bus will probably become the superior overall interconnection mechanism for a major portion of the MMPS application spectrum.

2.2 The Bus Utilization Problem

Given the above promising prospects regarding the single time-shared bus as an MMPS interconnecton mechanism, the fundamental problem now becomes that of designing a new bus which can support such an MMPS architecture. Essentially, the single bus operates in a two-step cycle, as follows:

- devices with an active bus need compete for the bus in an arbitration process before the selected device is connected to the vacant bus, and
- information is transferred on the bus, which is released when the transfer is finished.

Depending on the actual implementation, the two steps may or may not be "overlapped" in time (i.e., the "next bus master" may or may not be selected in parallel with the "current bus usage"). From a bus throughput point of view, the first alternative is obviously the best. In fact, given this alternative, the bus allocation mechanism will not influence the bus throughput at all, if the entire bus arbitration process is at least as fast as the minimum bus information transfer process. On the other hand, it should be noted that a slow, non-overlapping bus arbitration process will waste a good portion of the available bus bandwidth.

The actual bus arbitration process, which involves decision algorithms based on the system priority structure, will not be discussed in this paper. Most of the problems related to the arbitration process are described in the literature by various authors [3,4,8,9,10,11]. In the remainder of this paper, the arbitration and the information transfer processes will be assumed to be overlapping. In addition, it is also assumed that the arbitration process is at least as fast as the minimum information transfer process. The bus utilization thus becomes solely a function of the information transfer process in the remainder of this paper.

Previous generations of microprocessors (eg., Intel 8080/85. Motorola M6800, and Zilog Z80) have very high bus utilization [12]. A set of bus utilization values for different processors is given in Table 1. As a result of the high bus utilization, these processors cannot successfully work together on a single bus without major modifications (i.e., between 1.5 and 2.5 non-modified processors will consume the entire bus capacity).

	8080*	8080**	6800	6502	9900	LSI-11
ADD8	82%	58%	92%	93%	55%	47%
SUB8	88%	58%	92%	93%	55%	47%
MULT8	71%	48%	66%	84%	39%	26%
DIV8	74%	50%	71%	80%	36%	20%
ADD16	82%	55%	92%	84%	55%	50%
SUB16	88%	58%	92%	84%	55%	50%
MULT16	74%	50%	75%	86%	39%	21%
DIV16	79%	53%	75%	83%	34%	20%
ADD32	81%	54%	92%	84%	48%	47%
SUB32	88%	58%	92%	84%	48%	47%
MULT32	87%	58%	82%	72%	37%	32%
DIV32	77%	52%	70%	77%	44%	37%
SORT8	77%	50%	57%	82%	45%	45%
SORT16	77%	52%	52%	80%	44%	44%
INTERRUPT	85%	57%	82%	60%	81%	46%

* 8080 uses bus during T1 (SYNCH) time.

** 8080 does not use bus during T1 (SYNCH) time.

Table 1
Processor bus utilizations for various benchmark programs [12].

queue status and device-ID signals into the arbitration process. However, this will increase the arbitration complexity, and the cost/performance potential of the single bus based MMPS will be severely degraded. Thus, it is not worth the effort to try to eliminate all of the additional bus utilization (especially since the potential gain is most likely insignificant in any reasonable size queue system). The problems associated with passing critical real-time parameters through the queues, which will impose an unpredictable delay in the response, must also be taken care of in the MMPS design. Both the "repeated request" problem and the problems associated with real-time parameters must be solved on an individual system basis.

3. A MODERN MICROPROCESSOR ARCHITECTURE.

It will be useful at this point to relate some of the previous ideas to a current microprocessor architecture and a vendor supplied MMPS interconnection scheme. The Motorola MC68000 will be used as a representative of the current microprocessor technology available for general use. The MC68000 is chosen primarily because of its microcoded nature and associated potential of being modified. Note that there is no major difference between the MC68000 and the other alternative 16 bit processors (i.e., Intel 8086 and Zilog Z8000) in terms of a single bus based MMPS.

Let us first review the architectural characteristics of the MC68000. At the chip level, this newest generation of microprocessors presents significant improvements over previous architectures. In particular, data paths to and from memory have been expanded to 16 bits, and the address ranges have been extended to allow larger memory systems to be directly accessed. All address calculations in the MC68000 are performed in 32 bits. However, at the present time only 24 bits are delivered outside the chip. The address is delivered in 4 separate spaces, thus yielding 64 megabyte addressability. All 32 bits could easily become available (given more pinout), and future versions of the processor will definitely be implemented with the full 32 bit address space. The MC68000 uses memory mapped I/O. This allows all memory referencing instructions to be used for I/O referencing as well, thus saving instructions at the expense of I/O protection at the instruction level. As a result, the I/O protection must be performed at the memory protection level. The large memory space virtually requires some form of management. The MC68000 uses an external memory management chip. This allows increased function by increasing silicon area. The management unit can relocate, check bounds, and function check all references to support very sophisticated memory mapping and protection facilities.

The supervisor/user mode separation in the MC68000 is a valuable feature. It allows for the protection of certain instructions and separate system/user stack pointers. The processor has an implementation of an "atomic" test and set operation, and a vectored interrupt structure for operating system support. The advanced instruction set includes capabilities for fully relocatable code, multiply and divide, and many high level language support mechanisms, including diverse addressing modes and regular structures.

The processor features a significant number of large general purpose registers, and can handle bits, 8 bit bytes, BCD digits, 16 bit words, and 32 bit long words as operands. There are also provisions for string handling as a stream of bytes, and future expansion room for floating point operations. Operational speed has also been improved. The shortest execution time (assuming sufficiently fast memory) is 500 ns.

Most of the above processor features would obviously be very useful in a multiple microprocessor environment. However, the problem of interconnection remains to be solved before the features will have any value in the area of MMPS.

3.1 Microprocessor Timing

The MC68000 bus utilization depends on the instruction and data fetching process. Actual bus loading numbers are not available at the present time. However, sufficient bus utilization estimates can be made. The MC68000 has three basic bus operations. These are:
(Each state is one half clock cycle, or 62.5 ns at 8 MHz.)
- -Read: 8 bus states plus wait state pairs as required
- -Write: 10 bus states plus wait state pairs as required
- -Read-Modify-Write: 18 bus states plus wait state pairs as required

A detailed description of the basic bus operations along with the timing diagrams is given in [14].

Most MC68000 instructions are memory fetch time bound. In addition, CPU and memory operations are carried out in parallel. Therefore, the processor is almost always on the bus doing a read or a write operation. In some addressing modes, requiring a full 32 bit add, the bus is idle for one basic machine cycle (i.e., 250ns at 8 Mhz). Some instructions, however, such as RESET, MULTIPLY, DIVIDE, SHIFT, and ROTATE are CPU intensive. These instructions, which are less frequently encountered, do not require any external accesses during execution. Overall, though, it is reasonable to assume that a continuous stream of reads and writes will

be performed by each processor under execution. The bus cycle description implies that a processor uses the bus for all but one of the states in each of the basic bus operations. If one were simply to connect two processors on a single time-shared bus, they could overlap bus dependent operations (execution) only during that single state. Thus, with respect to overall bus utilization, there is no major difference between the old Motorola M6800 and the new MC68000. Bus utilization values like those shown for the 16 bit processors (LSI-11 and TI9900) in Table 1 should be expected for the MC68000.

From the above description it can be seen that each processor inherently uses the bus it is connected to for a very large portion of the time. Any attempts to connect more than one processor on one such bus would not yield a significant increase in computation power. This is because each processor would have to be proportionately slowed down in order to get access to the bus. Obviously, the above conclusions are highly dependent on the instruction mix. However, the overall result is that, at best, 2 processors would profit on a single bus. That is far from a promising result.

3.2 Vendor Supplied Multiprocessing

The microprocessor manufacturers are aware of the high bus utilization problems, and they have devised plans to allow many of their CPUs to be connected together in a MMPS. The designs differ widely from vendor to vendor. Again, the actual analysis depends, to a large extent, on the bus usage of any single processor. In addition, it also depends on the amount of crosstalk traffic between processors and various memory and I/O elements, i.e., the total amount of data to be transmitted over a common communication system.

In the Motorola interconnection system, each processor is set up with a local bus with local memory and peripherals. In addition, there is a Global bus connecting all the local buses together through Bus Arbitration Modules (BAMs). The system is illustrated in Figure 5. Each processor is free to execute at full speed within its own local space until either it needs something in another local bus area, or some local information is requested from the BAM unit. Any access off the local bus onto the Global bus takes more time than a simple local access. Accesses from the Global bus back onto a local bus are done using a DMA operation. There are no strictly global, shared resources in the Motorola definition. The BAMs operate as memory manager devices. Each recognizes off-local accesses, and requests the global bus to get access to that resource, which is assumed to be on some other local bus. When granted that bus, it makes the request, and the processor system whose local bus contains the needed information is stopped and the data is passed back to the requester, who then relinquishes the global bus.

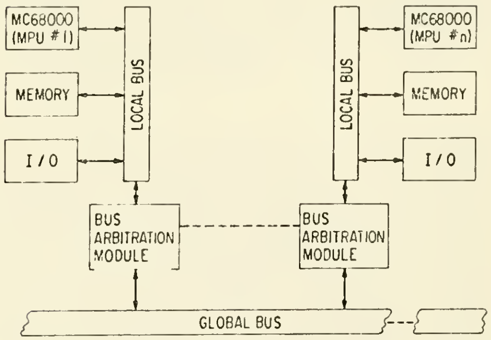

Figure 5
Multi-MC68000 system using Bus Arbitration Modules (BAMs).

It seems evident that the BAM solution is designed for low non-local access rates. The time required to acquire the necessary communication path (local-global-local) and transfer the data is relatively long. Furthermore, the global bus can only have one transaction going on at a time, ignoring all other requests until the bus is again free. Overall, this process is relatively simple, since there is no pending of operations or other dynamic considerations. The bus switches supply a complete communication connection and hold it as long as the entire transaction requires. A major difficulty with this scheme is that there is nothing to prevent several processors from making continuous accesses into one processor, effectively stopping that processor entirely. The priority on the global bus appears to be fixed, so the processor with the least priority may never get a global transaction completed.

An effective speedup could be achieved by using global memory attached to a dedicated BAM type unit, with no other processor local to it. In this case, nothing else would be contending for the local bus, and any request from the Global bus could be granted immediately without any arbitration. As soon as a processor got the Global bus, it could be routed directly to global memory, and the data could be returned fairly quickly. Global memory would then be truly shared, but access to it would be limited in rate by the global bus bandwidth, which, by definition, cannot be any higher than the data rate on a single local processor bus. Thus, a

processor executing a common routine in global memory could tie it up completely. As the processors are assigned a fixed priority, a complete lockout of the other requests would occur.

3.3 Split Transaction Multiprocessing

In the previous MC68000 designs, buses were used to supply complete communication links between one master and one slave device; the master would ask the slave to do something, and then wait until the slave responded. To allow more extensive sharing of resources, a higher speed communication mechanism is needed between devices. In this regard, it becomes desirable to connect several MC68000 processors to a split (especially Pended) transaction bus. One very important multiprocessing point must be brought out here. In a multiprocessing system, processor synchronization requires some form of an "atomic" test-and-set primitive to implement resource allocation. In the MC68000, this is performed via a read-modify-write operation, which is formed of connected read and write operations. In a split transaction bus, the first read cycle can be started, and other bus operations can occur while the first read is completing. If one of those operations happens to be to the current location being read by the test-and-set, then it can get to it between the test-and-set cycles if careful hardware measures are not taken, which would destroy the intended operation of the test-and-set operation. This problem is particularly evident in a Pended bus with queues, where the queue may buffer up many requests to many locations.

Virtually any processor can be connected to a split transaction bus (possibly through some external control logic), provided it creates some form of request signal (such as an address strobe), and has an asynchronous signal to denote transaction completion (wait or acknowledge). On processors where the split transaction protocol must be created from such external signals, the timings may degrade processor-to-memory performance figures. In the MC68000, the protocol can be created, but the timing is less than optimal. If the slaves are memory fast enough to supply the processor without wait states if connected directly, then a split transaction system will have to insert two to four wait states [14] into the processor read cycles due to the delays associated with proper address strobing and data acknowledge. The addition of the wait state pairs results in a net slow-down of up to 50 percent for each cycle of each processor. By far the worst difficulty, however, is that the MC68000 test-and-set operation simply makes a read operation followed by a write operation. It is impossible to tell, until the read is done, that a read-modify-write operation is in progress, which is too late to stop interference [14]. This problem can only be solved by a change in the protocol microcode to enable an output signal to give an earlier indicator. The only other choice is to ignore the test-and-set software instruction and to create the indivisible operation in hardware test-and-self-set flags.

4. SUMMARY

MMPS interconnection problems have been discussed in terms of the single time-shared bus. The performance drawbacks usually associated with the single bus alternative has been attributed to the high bus utilization at the basic building block level. The Pended Transaction Bus protocol has been presented as a general solution to the high bus utilization problem. It has been indicated that a properly designed single time-shared bus can support more than 50 processors without any severe contention.

The basic bus control protocol of the MC68000 has been investigated. It has been shown to effectively curtail significant multi-microprocessing because of high bus utilization. Although the low bus bandwidth is highly dependent on the instruction mix, severe throughput degradation results because of an inefficient protocol design at the processor level.

The vendor-supplied multiple microprocessor interconnection system has also been reviewed. A consensus view is that the global system bus can be easily saturated by the requests of a single high-priority processor. The proposed Motorola design is best aimed at low-volume global transactions where each processor relies heavily on its local resources. The Local/Global bus solution would be inappropriate and inefficient for a large (>5) number of processing elements, with high bus bandwidth requirements.

A pended transaction protocol is one approach for increasing the available bandwidth of such a time-shared bus. However, the process of adapting a standard, high bus usage microprocessor to such a Pended multiprocessor scheme still has several difficulties which give such an approach limited practicality. These problems include additional wait states in the processor cycle, and inability to guarantee proper operation during a read-modify-write sequence. Since the microprogrammed nature of the MC68000 allows modification of the addressing control structures, it would seem possible to properly implement the Pended Bus Protocol directly from the chip without radically altering the chip's internal structure. Practical multiprocessing capabilities could be implemented easily and directly.

The implications of such an improved system interconnection architecture for high bandwidth applications, such as real-time industrial control or computation-intensive tasks, are truly significant. Computer control functions such as multi-dimensional, multi-arm machine tool operations can now be accomplished in a modular, expandable fashion that can still accomodate current technology building blocks. Such techniques are also extendable to other application areas.

REFERENCES

[1]-Enslow, P.H. Jr., "Multiprocessor Architecture-A Survey," (invited paper) 1975 Segamore Computer Conference on Parallel Processing, pp. 63-70.

[2]-Enslow, P.H. Jr., ed., Comtre Corp., Multiprocessors and Parallel Processing, John Wiley, New York, (1974), 340+xii pp.

[3]-Haagens, Randolph B., "A Bus Structure for Multi-Microprocessing," MIT Department of Electrical Engineering and Computer Science S.M. thesis, submitted January 1978.

[4]-Thurber, K.J., and Masson, A.M., Distributed Processor Communication Architectures, Lexington Books, 1979.

[5]-Enslow, P.H., "Multiprocessor Organization - A Survey," ACM Computer Surveys, March 1977, pp. 103-129.

[6]-Lavington, S.H., Thomas, G., and Edwards, D.B.G., "The MU5 Exchange," 1974 Conference on Computer Systems and Technology, IEEE Conference Publication No. 121, pp. 219-225.

[7]-Danielsson, Per-Erik, and Gudmensson, Bjorn, "Time-Shared Memory-Processor Interface," 1975 Segamore Computer Conference on Parallel Processing, pp. 90-98.

[8]-Jenson, E.D., . Thurber, K.J., and Schnieder, A.M., "A Review of Systematic Methods in Distributed Processor Interconnection," International Communication Conference, 1976, 7.17-7.22

[9]-Chen, R.C., "Bus Communication Systems," Ph.D. Dissertation, Carnegie-Mellon University, 1974.

[10]-Anderson, G.A., and Jenson, E.D., "Computer Interconnection Structures: Taxonomy, Characteristics, and Examples," ACM Computing Surveys, December 1975.

[11]-Thurber, K.J., et. al., "A Systematic Approach to the Design of Digital Busing Structures," Proceedings of the FJCC, 1972, pp. 719-740.

[12]-Gerson, Ira, [MRG.I.BU.00] Internal Monograph, MIT, May, 1976.

[13]-Toong, Hoo-min D., and Gupta, Amar, "IMMPS-Interactive Multi-Microprocessor Performance System," MIT CISR Technical Report #6, December 1979.

[14]-Toong, Hoo-min D., Strommen, S., and Goodrich II, E. et. al., "Architectural Comparisons: MC68000, Z8000, 8086," MIT CISR Technical Report #5, December 1979.

[15]-Toong, Hoo-min D., work in progess, to be published.