A MODEL OF INTERFERENCE

IN A SHARED RESOURCE MULTIPROCESSOR*

John E. Jensen and Jean-Loup Baer
Department of Computer Science
University of Washington
Seattle, Washington

Abstract

This paper presents a generalized model of tightly-coupled multiprocessor systems which is then simplified to form a stochastic model for the study of interference. Analysis is performed on the resource contention which is characteristic of such systems in order to find a measure of system performance. After reviewing the problem of memory interference, the analysis is extended to contention in other individual resources, then combined to form a model for the interacting effects of contention in systems where processors contend for several shared resources.

1. Introduction

Recent design proposals and realizations [4,10,11] have included multiprocessors in attempts to meet the expanding demand for high-performance systems. A solution to the need for improved efficiency lies in the distribution, duplication and sharing of hardware resources. Unfortunately this leads to situations in which a given unit may receive several simultaneous requests for service (e.g. a memory module). The result is degraded performance, or interference, measured by comparing actual machine performance to the ideal case for which there is no contention. This paper presents a generalized model of tightly-coupled multiprocessors with highly shared computing resources. Analysis is then performed on the resource contention in order to find a measure of system performance.

The best known contention problem is when processors and IO controllers interfere in their access to main storage. Analytic models with exact solutions exist for two processor systems [8] via Markov chain methods, but the general case becomes too complex, precluding a precise solution. For a solution in closed-form one has to introduce simplifying assumptions in order to prevent the analysis from becoming unwieldy. A series of models have been introduced in which a prototype instruction is assumed and its execution rate (IER) analyzed for a variety of multiprocessor types [9]. Closed-form solutions are obtained for IER in terms of parameters which relate typical design characteristics of the memories and processors. In addition, cache memories may be introduced to the processor-memory interface [3]. In this paper, we extend previous formulas [9] to include cache memories, and then propose a more general one for systems in which processors contend for several resource classes as well as primary memory.

2. The Machine Model

2.1 A General Shared Resource Multiprocessor

Figure 1 shows a general model of a shared resource multiprocessor (SRM) in PMS notation [2]. The example was chosen purely for ease in description and conservation of space, with the design of more specific configurations being one of the objectives of the model. 8 central processors P.c share 16 modules of primary memory M.p through a central processor-memory switch S.mp. Each P.c possesses some local memory M.c and a set of mapping registers D.map which define its access to main memory.

The P.c's have no arithmetic power, performing only load, store and branch instructions. Other instructions are merely fetched and decoded, while operands are sent to a shared set of pipelined execution units D.e via a common request bus L.req. The P.c's are arranged into 2 time-shared rings by S.ring, which creates a maximum overlap of computing with a minimum bandwidth required in the request bus [4]. Input-output is initiated to IO controllers in the same way as a request for a D.e. When an "IO" instruction is executed, a request is sent to an appropriate K.io and the P.c is allowed to continue. IO-completion interrupts cause the appropriate P.c to be interrupted [5].

A special controller K.sched is provided for assigning new tasks to P.c's, with two options for flexibility in the scheduling mechanism. In the "floating control" scheme [5,7] P.c's perform their own scheduling under the control of K.sched. Under "fixed control", K.sched serves each request by returning the entry point of the new task in memory, while a dedicated processor P.sched (with associated M.a for the scheduling tables) constantly supplies K.sched with the next task to be assigned for execution.

2.2 Examples of the Model

The generality and versitility of the model may be illustrated by examining some current designs. C.mmp [11] is a set of 16 asynchronously executing PDP-11's (each with local memory) which access main memory through D.map's and S.mp. The ring structure and D.e's are missing since each P.c has its own complete processing capability. C.mmp's IO system is similar to its memory system in that the P.c's are connected to busses supporting the IO controllers by the S.kp switch. Scheduling is handled by the operating system without any additional hardware.

Figure 2 is a conception of Texas Instruments' ASC [10]. A single P.c feeds instructions to 4 high-speed pipelined D.e's which consume streams of vector operands under the control of registers found in M.c. The most interesting feature is the "peripheral processor" which performs the control and data-management functions for the ASC, and is actually a ring of "virtual processors" (P.v).

Figure 3 emphasizes the ring structure aspects by modeling Flynn's SRM [4]. It has 4 rings of 8 P.c's, and uses L.req and D.e's as in the model. The P.c's have no D.map or S.mp, but access memory through buffers. Cache memory M.c is associated with each ring. No mention is made of IO, and scheduling is done under program control through a standard fork-join construct.

2.3 The Simplified Machine Model

The model described thus far requires too much detail to be studied at the instruction level, hence we capture some of its generality into a more manageable form in Figure 4. Centrally located is S.mp which provides access by the P.c's and K.io's to the M.p modules. The specialized scheduling processor P.s (with memory M.a) makes all policy decisions regarding the activation of user and operating system tasks as well as allocating the system's resources. IO consists of three subsystems, representing the common IO speeds anticipated.

*This research was supported by NSF grant GJ-41164.

The multiprocessing resources consist of synchronized processor rings (3 in the figure) with a set of independent pipe-lined D.e's which are capable of performing all arithmetic functions (except divides) with the same latency. Each ring consists of skeleton P.c's and corresponding M.c's connected by a processor-ring switch S.p. The purpose of the time-multiplexed switch [4] is to select the P.c that is to be considered "active" during each time-slice of the ring, and to coordinate all communication between the P.c's and the D.e's and the remainder of the system.

The instruction units use an instruction set which is patterned after the SRM [4]. Each of the 8 skeleton P.c's begins its instruction-fetch sequence one minor cycle behind its predecessor on the ring. In one major cycle each P.c will prepare one instruction for execution to take place during the subsequent one. A 60ns minor cycle is assumed [1,10], resulting in a 480ns major cycle which provides ample time (120ns) for finding operands in an implicit cache. In the case where an access to main memory is required ("miss" on the cache), 600ns should be more than sufficient to perform the transfer (120ns plus one major-cycle delay) and still maintain the synchronous timing of the processor ring.

3. The Resource Contention Model

3.1 The Memory Interference Problem

In this section, we introduce an analytic model for general resource contention used to estimate the losses due to interference between processors requesting identical resources. We begin by examining memory interference (the request by more than one P.c for the same M.p module) using expected values for the number and types of instructions executed. The combined effects of the hardware speed and memory conflicts are characterized by a single entity, the instruction execution rate (IER), for which we calculate and estimate.

The P.c's and M.p's are viewed as a stochastic service system in which the M.p's represent m servers, each capable of serving one of k P.c's. Each server handles only those requests directed toward it, serving them in order of arrival and queuing those occurring when it is busy. The M.p's are characterized by a constant service time (access time) followed by an interval of unavailability (rewrite time) before subsequent requests can be serviced. P.c's are characterized by the amount of elapsed time between the completion of service on one memory request and the generation of the next one.

The problem is made more tractable with a few simplifying assumptions. Although processor behavior varies with different instruction types, the probability distribution of instructions, the average frequency of memory requests, and the average time required to execute one instruction can be determined. The access pattern of each processor is assumed to be random, and no distinction is made between read and write requests. We simplify further by considering each instruction to be a series of instances of a "unit instruction" consisting of one memory access followed by a fixed (mean) interval of processor activity.

3.2 An Analytic Model for Memory Contention

In Strecker's formulas for the "unit execution rate" [9], the execution sequence is considered as a Markov process, consisting of a series of "unit instructions", from which we may calculate the rate of memory service. (The principle parameters are defined in Table 1.) The unit instruction begins when an address is received by one of the m modules of M.p at S.mp. Ta is the time required for the memory control to set up the switch and for data to be delivered. Tw is the time required for the module to recover and become ready for the next request. Tp begins for each of the k active P.c's when it receives data from an M.p, extending through the computation until the P.c has a new data address. The "computation" done in this "unit instruction" may be an instruction decode, an (indirect) address computation, or the actual execution of a machine instruction. Several of these unit instructions comprise one complete machine instruction.

The unit execution rate (UER) is the number of unit instructions executed per unit time. In terms of the service times Sm and Sp [9]

$$UER = m * [1 - (1-Pm/m)^k] / Sm$$

such that

$$Pm = 1 - (m/k) * (Sp/Sm) * [1 - (1-Pm/m)^k].$$

The analysis is split into three cases (bases upon the relationship between Tp and Tw) which may be combined to form composite equations for the service times as

$$Sp = Tp\theta Tw \text{ and } Sm = Ta+Tw - (Tw\theta Tp) * (1-Pm/m)^k$$

(where $a\theta b = a-b$ if $a>b$, and $a\theta b = 0$ if $a\leq b$). The complete equation for the unit execution rate is then

$$UER = \frac{m * [1 - (1-Pm/m)^k]}{Ta+Tw - (Tw\theta Tp) * (1-Pm/m)^k}$$

where

$$Pm = 1 - \frac{(m/k) * (Tp\theta Tw) * [1 - (1-Pm/m)^k]}{Ta+Tw - (Tw\theta Tp) * (1-Pm/m)^k}.$$

In order to solve the Pm equation, we examine the two cases $Tp\leq Tw$ and $Tp>Tw$. In the first case Pm=1 and we are done. In the second case the denominator simplifies to Ta+Tw, resulting in a k-th order polynomial in Pm. Since the two sides of the equation are monotonic in opposite directions on the interval [0,1], for a given set of parameters we may solve for Pm in this interval and obtain the UER from the first equation above.

We extend this model by associating with each P.c a cache memory with access time Tb and "hit ratio" Pb. The effect of this addition is that with probability Pb, the memory request will be satisfied in the cache (hence no M.p service) while with probability 1-Pb, a normal memory cycle will be required. For the case where $Tp\geq Tw$ it has been shown [3] that

$$Sp = Pb*(Tp+Tb) + (1-Pb)*(Tp-Tw)$$

and

$$Sm = Pb*(\ \ 0\ \ ) + (1-Pb)*(Ta+Tw)$$

such that Pm equals

$$1 - \frac{m*[Pb*(Tp+Tb)+(1-Pb)*(Tp-Tw)] * [1-(1-Pm/m)^k]}{k * (1-Pb) * (Ta+Tw)}.$$

This new Pm equation has a single solution in the interval [0,1] as in the previous case. We may generalize this formulation to include the case where $Tp<Tw$ [6], but the memory being considered in this model is relatively fast, so the case $Tp\geq Tw$ is sufficient, yielding

$$UER = \frac{m * [1 - (1-Pm/m)^k]}{(1-Pb) * (Ta+Tw)}$$

where Pm is determined from the above formula.

3.3 Modeling Multiple-Resource Systems

Previously, a unit instruction was defined in terms of memory access frequency, with all other aspects of the instruction being considered as "processor activity", or Tp. Using the same analysis as above we can determine the effects of multiprocessor contention for other shared resources by extending the notion of a unit instruction to represent one "access" to a resource of any given class (e.g. pipelined D.e's) followed by the average processing time between requests for that resource class. The period of time comprising one unit instruction will, in all cases but for M.p, include several machine instructions. For example requests for floating-point multiplies occur in approximately

13% of the instructions for a scientific mix [6], such that one unit instruction for the multiply resource becomes 1/0.13 times the length of one machine instruction.

When main memory is considered as the sole contendable resource, the IER of a system is computed by first estimating the UER of memory, then dividing by the number of memory references per instruction. The UER of memory is computed using Strecker's approximation which assumes an otherwise constant P.c processing time. A similar set of assumptions will allow the UER to be calculated for the floating-point multiply units (or any other resource), given that some fixed value can be derived for the remaining "processor activity" between requests for the multiply units (cf. section 3.4). The IER can then be calculated by dividing by the frequency of multiply instructions.

In order to model the UER of other resources, the parameters used in the contention model must be generalized. Table 1 defines the set of resource contention parameters a-z which will be used in the remainder of this paper. The correspondence in parameter names for the memory interference example is given in the table and is illustrated here functionally.

UER (k,m,Tp,Ta,Tw,Tb,Pb) = h(k,m,t,a,w,b,p)

Table 2 illustrates typical figures for these parameters applied to a variety of harware resources.

With the introduction of pipelined D.e's, the number of stages v in the pipelines becomes of importance. So far we have assumed that all k P.c's actively contend for the system's resources at all times such that UER=h(k,...). In our machine model, however, the P.c's are intentionally arranged into time-phased rings of v P.c's each, so that they only contend with corresponding P.c's from other rings on the same time-slot, increasing the IER of the system. If the system contains k P.c's which are all active, then there are v separate contentions (one per time-slice on the processor ring) among goups of k/v P.c's. In this situation (for a single-resource system) UER=v*h(k/v,...) such that

$$IER = v * h(k/v,m,t,a,w,b,p) / f.$$

Suppose now that some P.c's are idle such that k is less than the total number of P.c's in the system. The approximation above is optimistic in that it assumes the k active P.c's to be optimally distributed over the v time-slots. In particular, if k<v, it computes the IER to be better than optimal! The invalidating factor is that not all v time-slots necessarily contain active processors. If we assume the k active P.c's to be randomly distributed, then c, the expected number of currently active time-slots, may be determined as was the expected number of busy memories:

$$c = v * [1 - (1-1/v)^k]$$

and hence

$$IER = c * h(k/c,m,t,a,w,b,p) / f.$$

3.4 The Model for Combined Resources

We have shown how the UER of each resource class may be determined, from which we calculate the performance measure IER=UER/f. In order to combine the analyses of the individual resources, we normalize this measure to the number of processors by the "processor execution rate" PER=IER/k. We also define the "effective execution rate" EER=IER(k)/IER(1) which measures the performance in terms of the number of effective processors, and the "multiprocessor efficiency" EFF=EER/k, which gives a direct measure of the degradation caused by contention in the system.

We now combine the analyses of the individual resources to form a model for the interacting effects of contention. Consider a system of k processors with n resource classes, each characterized by a set of parameters {m,v,a,w,b,p} (e.g. Table 2). We calculate the UER for each resource class i (assuming that we know t_i, the average time between the completion of service and the generation of the next request for resource i) by substituting the appropriate parameters into

$$h_i = h(z_i,m_i,t_i,a_i,w_i,b_i,p_i).$$

Allowing the unknowns z_i and t_i, an equation for L, the expected length of one complete machine instruction, may be obtained from L=1/PER in terms of the UER of the i-th resource:

$$1/L = (h_i/f_i) / z_i$$

with z_i, the average number of processors in contention for resource i, being computed as

$$z_i = k/c_i \text{ where } c_i = v_i * [1 - (1-1/v_i)^k].$$

The remaining unknown t_i was defined earlier (for systems with $t_i \geq w_i$ such that one unit instruction for class i has length t_i+a_i. (We have assumed for simplicity that $p_i=0$. Otherwise a_i may be replaced by the appropriate expression in a_i, b_i and p_i.) However, t_i is not a function solely of the i-th resource (as assumed earlier), but rather of the execution rates of the n-1 other resources. Thus the equation above contains two unknowns, L and t_i. In order to eleminate t_i, we repeat the above equation for the n resource classes and add a constraint to form a system of n+1 equations in n+1 unknowns $\{t_1,t_2,...,t_n,L\}$. The constraint is that L must be the sum of the access times per instruction of each of the n shared resources, plus the service time of the non-shared resources in the skeleton processor (time required to decode, index, and issue instructions).

To obtain an equation for this constraint, consider the example of Figure 5. Shown is a six-instruction sequence for a system with three resource classes: two M.p modules, an add and a multiply unit. (We assume an access time of 3 minor cycles and a rewrite time of 2 minor cycles for M.p, for a major cycle time of 8 minor cycles.) The time occupied by communication a_i between the processor and each resource is shown by solid lines in the figure, with dashed lines representing the other activities w_i. Occasional delays d_i, represented by dotted-lines, are caused when the requested resource is busy serving requests from another processor (e.g. the first multiply is delayed 1 major cycle). Requests to functional units are sent on the last minor cycle of the instruction, with the result available exactly one major cycle later (cf. a_2's and a_3's and their associated w_2's and w_3's). The skeleton processor looks only one instruction ahead and hence need not worry about potential register conflicts. This was also subsumed in our concept of a unit instruction.

The individual times may be summed in order to form a constraint on the length of each machine instruction, as demonstrated in Table 3. The total elapsed time for one unit instruction on resource i is

$$t_i + a_i + d_i,$$

where d_i is the average delay due to contention for the i-th resource. (Thus the table entries for t_i may be found by subtracting a_i and d_i from the total time elapsed). We use this expression to determine an expected value for the length of one complete machine instruction L in terms of the i-th resource

$$L = (t_i + a_i + d_i) * f_i.$$

The i-th resource occupies time $(a_i + d_i) * f_i$ out of each instruction, which may be solved from the equation

above to yield

$$(a_i + d_i) * f_i = L - t_i * f_i.$$

If we let Lo be the time required per machine instruction by the skeleton processor, we have as our constraint equation

$$L = Lo + \sum_{i=1}^{n} \{ L - t_i * f_i \}.$$

This completes our system of equations, which has a unique solution that may be determined numerically. The knowledge of L implies that of PER as defined previously and hence that of IER. The analytical solutions thus achieved are in accordance with the results from simulation presented in Table 4. The example system in Table 2 was simulated, with resource request frequencies determined by random draws from four typical instruction mixes [6]. The resulting instruction lengths are compared with the contention-free instruction lengths computed by ignoring time lost waiting for resources.

4. Summary and Conclusions

A general model of a large, tightly-coupled multiprocessor system has been introduced and shown to be capable of representing several recent design proposals and realizations. It was then reduced to a more specific model of a shared-resource multiprocessor for use in an analytical study of resource contention. By examining first the problem of interference in main memory, we have been able to abstract previous results [3,9] to find closed-form formulas for the effects of contention in any individual resource, on the assumption that the behavior of the system with respect to all of its other resources is known. Furthermore, we have combined the analyses of the separate resources to form a more complete model when processors contend for several resource classes simultaneously.

Solving for this model yields a unique solution which allows a prediction of performance and degradation in multiple-resource systems. Several hypothetical systems have been parameterized through the model, and the iterative numerical solution has converged to the correct processor execution rate in each case. The performance estimates measured by this analysis have been shown to be reasonable by simulation at the instruction level, and it is anticipated that future simulations of systems will make use of this result to account for hardware resource contention while retaining a high-level view of the systems being modeled.

References

[1] Anderson, D. W., Sparacio, F. J. and Tomasulo, R. M. "The IBM System/360 Model 91: Machine Philosophy and Instruction Handling" IBM J. of R. & D. 11:1 (Jan. 1967), pp. 8-24.

[2] Bell, C. G. and Newell, A. Computer Structures: Readings and Examples McGraw-Hill, New York, N. Y., 1971.

[3] Bhandarkar, D. P. "Analytic Models for Memory Interference in Multiprocessor Computer Systems" Ph.D. Dissertation, Carnegie-Mellon University, Sept. 1973.

[4] Flynn, M. J. and Podvin, A. "An Unconventional Computer Architecture: Shared Resource Multiprocessing" Computer 5:2 (March-Apr. 1972), pp. 20-28.

[5] Gountanis, R. J. and Viss, N. L. "A Method of Processor Selection for Interrupt Handling in a Multiprocessor System" Proc. IEEE 54:12 (Dec. 1966) pp. 1812-1819.

[6] Jensen, J. E. "Dynamic Task Scheduling in a Shared Resource Multiprocessor" Ph.D. Dissertation, University of Washington (in preparation).

[7] Pariser, J. J. "Multiprocessing With Floating Executive Control" IEEE Int. Conv. Record, 1965, pp. 266-275.

[8] Skinner, C. E. and Asher, J. R. "Effects of Storage Contention on System Performance" IBM Systems J. 8:4 (1969), pp. 319-333.

[9] Strecker, W. D. "Analysis of the Instruction Rate in Certain Computer Structures" Ph.D. Dissertation, Carnegie-Mellon University, June 1970.

[10] Watson, W. J. "The TI ASC -- A Highly Modular and Flexible Super Computer Architecture" Proc. AFIPS 1972 F.J.C.C., pp. 221-228.

[11] Wulf, W. A. and Bell, C. G. "C.mmp -- A Multi-Mini-Processor" Proc. AFIPS 1972 F.J.C.C., pp. 765-777.

Table 1 - Contention Model Terminology

Ta	effective access time of M.p (service time)
Tw	effective rewrite time of M.p (recovery time)
Tp	average time between the completion of service on one memory request and the generation of the next request by P.c
Tb	cycle time of fast buffer memory
Sm	time required by M.p to service one request
Sp	time beyond memory cycle required by P.c to prepare the next request
Pb	probability of finding the request in buffer
Pm	probability that a request is queued at an M.p
a	service (access) time of each resource (Ta)
b	buffer speed for each resource (Tb)
c	number of processor-ring time-slots containing requests for each resource
d	delay time caused by contention at each resource
f	frequency of use for each resource (ratio of requests per number of machine instructions)
h	the contention function (UER)
i	index to the various resource classes
k	number of active processors (those to which tasks are currently assigned)
L	length of one machine instruction (inverse of IER on one processor)
m	number of resource units in each resource class
n	number of resource classes
p	probability of using buffer for each resource (Pb)
t	time between completion of service on one request for each resource and the generation of the next request for that resource (Tp)
v	number of stages in the functional-unit pipelines for each resource (coincides with the number of time-slices in the processor rings)
w	recovery (rewrite) time for each resource (Tw)
z	average number of processors in contention for each resource

Table 2 - Example of Resource Parameters

Resource Class	m	v	a	w	b	p
main memory	16	1	600ns	120ns	60ns	0.9
integer add	1	8	0	480ns	-	0
floating add	2	8	0	480ns	-	0
multiply unit	2	8	0	480ns	-	0
divide unit	2	8	0	1920ns	-	0
logical unit	1	8	0	480ns	-	0
shift unit	1	8	0	480ns	-	0
IO controller	4	1	0	12ms	-	0
scheduler	1	1	480ns	6μs	480ns	0.9

Table 4 - Comparison With Simulation Results

(average instruction length in nanoseconds)

Instruction Mix	Analytic Result	Simulation	Contention-Free
Floating Point	483	477	401
FORTRAN I/O	650	659	369
XPL/S Compiler	439	443	371
SIMTRAN Simulation	532	535	377

Table 3 - Timing Summary for the Computation A*B-C*D+E

	Total Occurrences			Length of Occurrence			Total Time Consumed			
	a_i	w_i	d_i	a_i	w_i	d_i	a_i	w_i	d_i	t_i
memory units	11	11	5	3	2	-	33	22	21	26
add unit	2	2	1	1	8	8	2	16	8	70
multiply unit	2	2	1	1	8	8	2	16	8	70

Total Decode Time (Lo)	6
Total Access Time a_i	37
Total Delay Time d_i	37
Total Time Elapsed	80
Number of Instructions	6
Average Instruction Length	13.3

Figure 3 - The Model Applied to Flynn's SRM

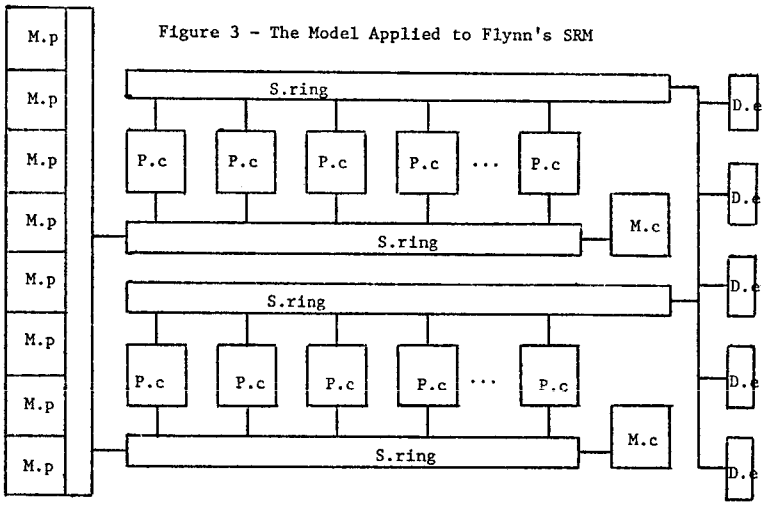

Figure 4 - The Simplified Machine Model

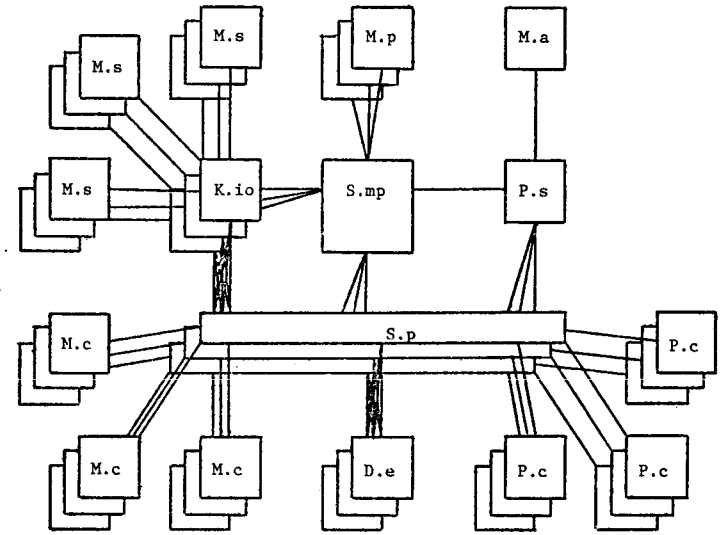

Figure 1 - A General Shared Resource Multiprocessor

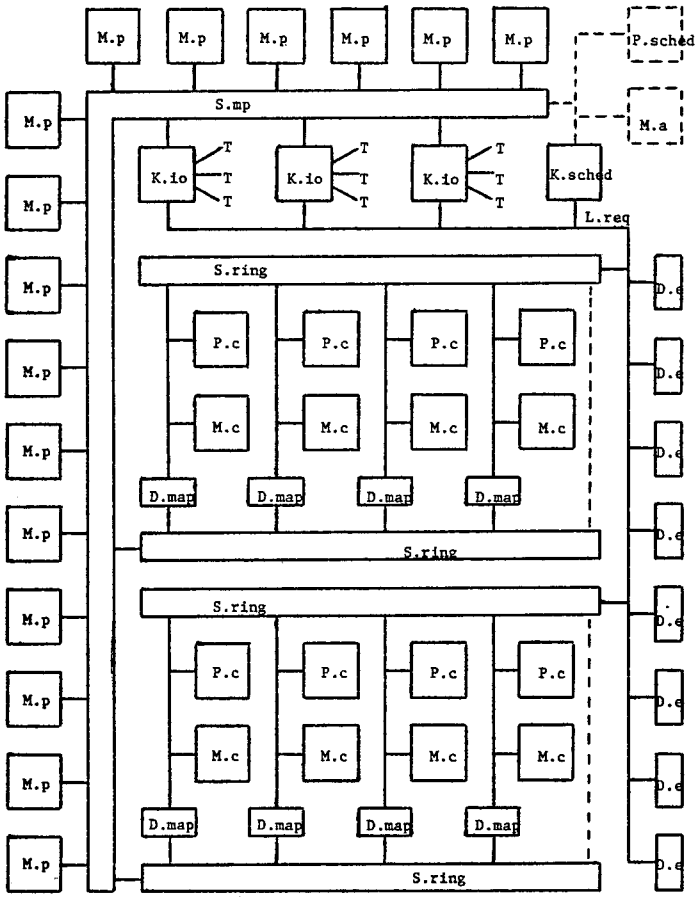

Figure 2 - The Model Applied to TI's ASC

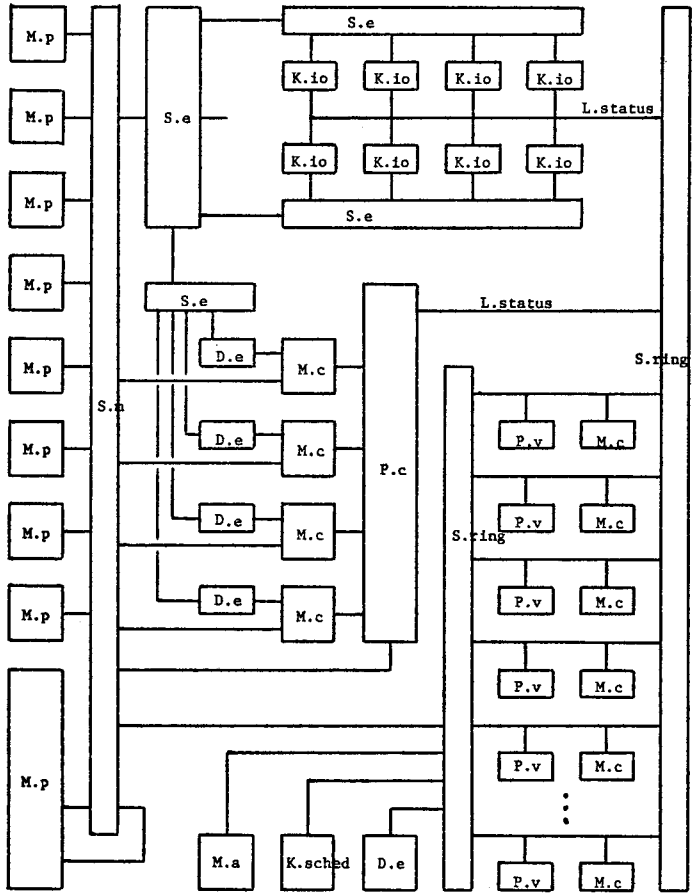

Figure 5 - Instruction Sequencing for the Computation A*B-C*D+E