ANALYSIS OF THE PASM CONTROL SYSTEM MEMORY HIERARCHY

David Lee Tuomenoksa
Howard Jay Siegel

Purdue University
School of Electrical Engineering
West Lafayette, IN 47907

Abstract - Many proposed large-scale parallel processing systems (e.g., PASM) can operate in multiple-SIMD mode. The multiple control units in such a system share a common secondary storage for programs. The control units use paging to transfer programs to their primary memories. One design problem is determining the optimal service rate for the secondary storage, where the "optimal" is characterized by maximum processor utilization. The problem is approached by developing a queueing network model for the PASM control system memory hierarchy. Based on assumed values for parameters which characterize the expected task environment, an optimal service rate is derived from the model. The values of the parameters in the model can be varied to determine the impact these changes would have on system performance. Simulation results verifying various aspects of the model are presented. The results are shown to apply to the general model for a multiple-SIMD machine.

I. Introduction

A multiple-SIMD system (e.g., [9]) is a parallel processing system which can be dynamically reconfigured to form one or more independent SIMD (single instruction stream - multiple data stream) [5] machines of varying sizes. Handling the memory management problem for the multiple control units is an issue which must be considered in the design of multiple-SIMD systems. One possible solution to the problem is the use of virtual memory [1]. If virtual memory is used in a multiple-SIMD system with common secondary storage for the multiple control units it is necessary to determine the optimal page request service rate for the secondary storage. The optimal is characterized by maximum utilization of the processors.

PASM is a multimicrocomputer system being designed at Purdue University for a variety of image processing and pattern recognition problems [10]. It is the use of PASM in the multiple-SIMD mode of operation which motivates this study. In this paper a queueing network model is developed for the memory hierarchy of the multiple control units in PASM and is analyzed to determine the optimal page request service rate for the secondary storage. The optimal service rate for the secondary storage can be determined from the average system page request rate using heuristics for serial multiprogrammed systems as a guideline. The average system page request rate can be determined by making assumptions about the task environment (e.g., number of processing elements which tasks require and the estimated execution time of the tasks). The system idle time which results from the multiple control units waiting for page requests to be serviced is determined for the case where the system page request rate deviates from the average rate which was used to determine the optimal secondary storage service rate. The values of the parameters in the model can be varied to determine the impact these changes would have on system performance. Simulation results verifying various aspects of the model are presented. The model and analysis is related to a general model for a multiple-SIMD machine.

Section II is an overview of the PASM multimicrocomputer system. Terminology is defined in Section III. In Section IV a queueing network model for the PASM control system memory hierarchy is developed. The average system page request rate for PASM is determined in Section V. In Section VI the optimal service rate for the secondary storage is determined. Operational analysis [3] is used to determine the average idle time for the multiple control units in Section VII. Simulation results are presented in Section VIII. In Section IX the analysis is related to the general model for a multiple-SIMD machine.

II. PASM Overview

PASM, a *pa*rtitionable *S*IMD/*M*IMD machine, is a large-scale dynamically reconfigurable multiprocessor system [10]. It is a special purpose system being designed to exploit the parallelism of image processing and pattern recognition tasks. PASM can be partitioned to operate as many independent SIMD and/or MIMD (multiple instruction stream - multiple data stream) machines of varying sizes. *PASMOS* is the operating system for PASM.

A block diagram of the basic components of PASM is given in Figure II.1. The *Parallel Computation Unit* contains $N = 2^n$ processors, N memory modules, and an interconnection network (see Figure II.2). The *Parallel Computation Unit processors* are microprocessors that perform the actual SIMD and MIMD computations. The *Parallel Computation Unit memory modules* are

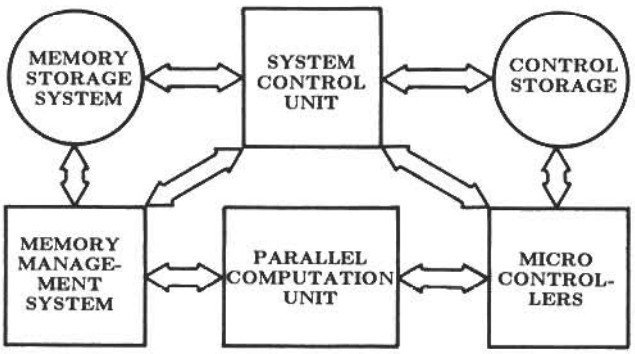

Figure II.1: Block diagram overview of PASM.

This research was supported by the Air Force Office of Scientific Research, Air Force Systems Command, USAF, under grant number AFOSR-78-3581 and by a Purdue University Graduate Fellowship. The United States Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation hereon.

0190-3918/82/0000/0363$00.75 © 1982 IEEE

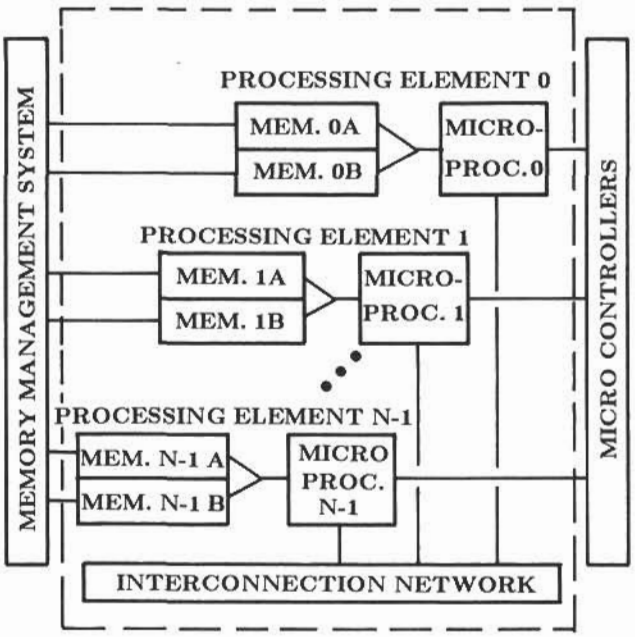

Figure II.2: PASM Parallel Computation Unit.

used by the Parallel Computation Unit processors for data storage in SIMD mode and both data and instruction storage in MIMD mode. The *interconnection network* provides a means of communication among the Parallel Computation Unit processors and memory modules. The *System Control Unit* is a conventional machine, such as a PDP-11, and is responsible for the overall coordination of the activities of the other components of PASM.

The *Memory Storage System* provides secondary storage space for the Parallel Computation Unit data files in SIMD mode, and for both the Parallel Computation Unit data and program files in MIMD mode. The *Memory Management System* controls the transferring of files between the Memory Storage System and the Parallel Computation Unit memory modules. It employs a set of cooperating dedicated microprocessors. Multiple storage devices are used in the Memory Storage System to allow parallel data transfer.

The *Micro Controllers (MCs)* are a set of microprocessors which act as the control units for the Parallel Computation Unit processors in SIMD mode and orchestrate the activities of the Parallel Computation Unit processors in MIMD mode. There are $Q = 2^q$ MCs. Each MC controls N/Q Parallel Computation Unit processors [7]. A virtual SIMD machine (partition) of size RN/Q, where $R = 2^r$ and $1 \leq r \leq q$, is obtained by loading R MC memory modules with the same instructions simultaneously. Similarly, a virtual MIMD machine of size RN/Q is obtained by combining the efforts of the Parallel Computation Unit processors and R MCs. Q is therefore the maximum number of partitions allowable, and N/Q is the size of the smallest partition. Possible values of N and Q are 1024 and 16, respectively.

Each MC processor is attached to a memory module which consists of a pair of memory units. The second memory unit may be used to load the initial pages of the next task while the current task is executing instructions from the first memory unit. In this analysis the steady-state condition is considered, i.e., the effect of preloading is ignored. Since a task which is executing uses only one memory unit, the paging analysis dose not consider the double-buffering. In SIMD mode, each MC fetches instructions from its memory module, executing the control flow instructions (e.g., branches) and broadcasting the data processing instructions to its Parallel Computation Unit processors. In MIMD mode the MCs may be used to help coordinate the activities of their Parallel Computation Unit processors.

SIMD programs are stored in the *Control Storage* which is the secondary storage for the MCs (see Figure II.1). The loading of SIMD programs from the Control Storage into the MC memory units is controlled by the System Control Unit and *Control Storage Controller*. The Control Storage Controller is a dedicated microprocessor which manages the Control Storage file system. When large SIMD tasks are run, i.e., SIMD tasks which require more than N/Q processors, more than one MC executes the same set of instructions. Therefore each of the MC memory units must be loaded with the same set of instructions. The fastest way to load several MC memory units with the same set of instructions is to load all of the memory units at the same time. This can be accomplished by connecting the Control Storage to all the MC memory units via the MC Memory System Switch. A block diagram of the control system memory hierarchy is given in Figure II.3. The MC Memory System Switch is controlled by the Control Storage Controller. All interaction between the Control Storage and the System Control Unit is done through the Control Storage Controller. (In [10] an enhanced scheme for connecting the MC processors to the MC

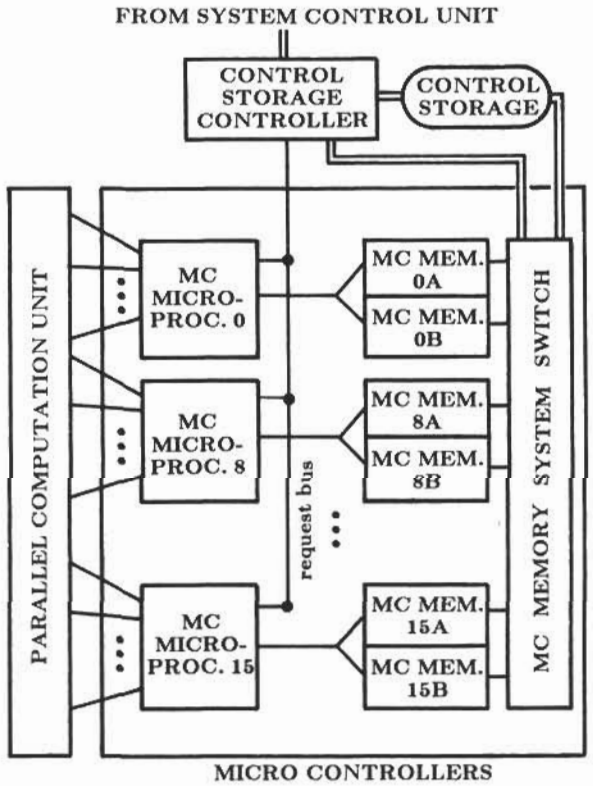

Figure II.3: Overview of PASM control system memory hierarchy for Q=16.

memory modules is also considered. The analysis in this paper also applies directly to that scheme.)

For some applications of PASM it is possible that the SIMD programs may be too large to fit into the primary memory (memory unit) of a given MC. Virtual memory may be used to give the programmer the illusion that the primary memory is much larger than in reality. There are two methods for implementing virtual memory: paging and segmentation [1]. In this paper, paging is considered. To implement paging as a part of the PASMOS operating system, the system must provide a translation mechanism to map the virtual address, which is used by the programmer, to a physical address, which is used by the system. In PASM, the translation is done by the MCs. When the page is not in the MCs primary memory (memory unit), it has a page fault. When an MC has a page fault, the MC sends a request on the request bus (see Figure II.3) to the Control Storage Controller which then services the request by locating the page in the Control Storage and sending it to the appropriate MC memory units through the MC Memory System Switch.

Consider the case where an SIMD task requires more than one MC. When a page fault occurs for the task, all of the MCs which are executing the task have a page fault. Since the faulted page is the same for all of the MCs which are executing the task, the page may be broadcast to all of the MC memory units simultaneously through the MC Memory System Switch. Hence, only one of the MCs must report the page fault to the Control Storage Controller, i.e., only one page request is generated. The MC which reports the fault can always be the same (e.g., logical 0 in the virtual SIMD machine) or may vary from one page fault to the next.

When PASM is operating as a number of independent virtual SIMD machines of varying sizes, the MCs are in effect a virtual MIMD machine. The secondary storage to this MIMD machine is the Control Storage. The model for the control system memory hierarchy is developed in Section IV.

III. Terminology

In this section the terminology which is used in the analysis is defined. *Virtual time* is defined to be the time that a processor is executing a task not including the time that it is idle waiting for page faults to be serviced or the time which a task is not assigned to it. *Real time* includes all time. The *real page fault rate,* referred to as the "page fault rate," is defined to be the rate at which page faults occur over real time, i.e., the number of page faults for the processor divided by the real time. The *virtual page fault rate* is defined to be the rate at which page faults occur over virtual time, i.e., the number of page faults for the processor divided by the virtual time. The *real page request rate*, referred to as the "page request rate," and *virtual page request rate* are the rates which pages are requested from the secondary storage over real time and virtual time, respectively.

For PASM, the virtual page request rate for MC_i is ν_i. The MC utilization is the fraction of time an MC is executing. The utilization of MC_i is U_i. The (real) page request rate for MC_i is $\lambda_i = U_i\nu_i$. The (real) system page request rate is the combined page request rates of all the MCs and is denoted by λ_{sys}.

IV. Model

In this section a queueing network model is developed for the PASM control system memory hierarchy. The interaction between the MCs and the Control Storage can be modeled by the two-station cyclic network in Figure IV.1 [3]. The MC subsystem contains the Q MC processor-memory unit pairs. Since only one of the MCs in a group of MCs executing a task makes page requests, there are only T MCs making requests where T is the number of tasks executing. Hence, the network is closed with T customers. The average time between page requests for each of the MCs making page requests is $1/\Lambda$, where Λ is assumed to be the virtual page fault rate for all tasks. The page request rate of the MC subsystem is the system page request rate λ_{sys}.

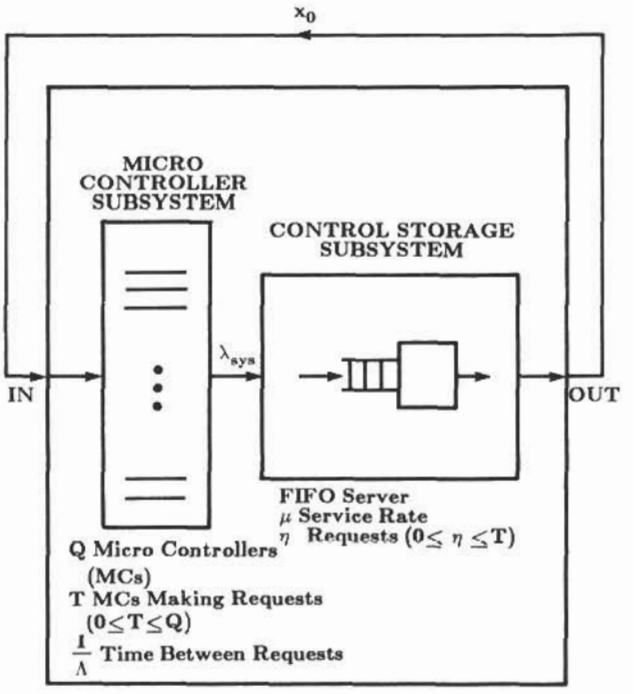

Figure IV.1: Two-station cyclic network which describes the interaction between the MCs and the Control Storage. The combined page request rates of the MCs is λ_{sys} and the throughput of the network is X_0.

The Control Storage subsystem services the page requests made by the MC subsystem. The service rate of the Control Storage subsystem is μ. The service queue at the Control Storage uses a FIFO queueing discipline. The number of requests in the Control Storage subsystem at a given time is η, where $0 \leq \eta \leq T$. The throughput of the network is X_0.

V. System Page Request Rate

In this section the queueing network model is analyzed to determine the average system page request rate. If all of the MCs are executing a different task (Q tasks executing), then the virtual page request rate for each of the MCs is Λ, i.e., $\nu_i = \Lambda$, where $0 \leq i < Q$. Therefore, the system page request rate is:

$$\lambda_{sys} = \sum_{i=0}^{Q-1}\lambda_i = \sum_{i=0}^{Q-1}U_i\nu_i = \sum_{i=0}^{Q-1}U_i\Lambda = \Lambda\sum_{i=0}^{Q-1}U_i,$$

Using the simplifying assumption that $U_i = U_{mc}$, a constant MC utilization, where $0 \leq i < Q$, then $\lambda_{sys} = QU_{mc}\Lambda$.

However, in the case of PASM, there are not usually Q independent tasks executing. For example, if an SIMD task of size RN/Q is being executed by the Parallel Computation Unit, the same instruction stream is being used by each of the R MCs which are controlling the task. The virtual page request rate to the Control Storage by the R MCs can be reduced from $R\Lambda$ to Λ by having one MC make the page requests and having the Control Storage broadcast the page to all R MC memory units simultaneously through the MC Memory System Switch (see Figure II.3).

To determine the actual average system page request rate it is necessary to determine the average number of independent instruction streams or tasks being executed by the MCs. This discussion will be limited to the execution of SIMD tasks. There are q+1 different sizes of SIMD tasks which can be controlled by the Q MCs, where $q = \log_2 Q$. A task may require 2^i MCs, where $0 \leq i \leq q$. Let the probability that an SIMD task requiring 2^i MCs is created be P_i. Let $\bar{E}_i$ be the average execution time for tasks which require 2^i MCs. The average value of the processor-time product for a task which requires 2^i MCs is defined as the product of the average execution time and the number of MCs required, $\bar{E}_i\, 2^i$. The processor-time product can then be used to weight the P_i distribution to determine R_i, the probability that a task requiring 2^i MCs will be executing on any MC which has a task assigned to it. R_i is defined as:

$$R_i = \frac{P_i \bar{E}_i 2^i}{\sum_{j=0}^{q} P_j \bar{E}_j 2^j} .$$

The average execution time parameters may be varied based on system use experience. For this analysis it is assumed that tasks of all MC requirements have equal execution time, so $\bar{E}_i = \bar{E}$. Therefore,

$$R_i = \frac{P_i \bar{E} 2^i}{\sum_{j=0}^{q} P_j \bar{E} 2^j} = \frac{P_i 2^i}{\sum_{j=0}^{q} P_j 2^j} .$$

In the analysis in this paper, a PASM with 16 MCs is assumed, i.e., q=4. For this analysis it is also assumed that distribution of the number of MCs required by a task is uniform, i.e., $P_i = 1/5$, where $0 \leq i \leq 4$. Once again, this assumption can be varied based on system use experience. The probability that a task requiring 2^i MCs is executing on a given assigned MC is $R_i = 2^i/31$, where $0 \leq i \leq 4$. The following theorem uses the above result for the probability that a task requiring 2^i MCs will be executing on any given assigned MC to determine the average system page request rate.

Theorem 1: The average system page request rate, $\bar{\lambda}_{sys}$, is:

$$\bar{\lambda}_{sys} = QU_{mc}\Lambda \sum_{i=0}^{q} \frac{1}{2^i} R_i ,$$

where Λ is the virtual task page fault rate, U_{mc} is the MC utilization for all MCs, and R_i is the probability that a task which requires 2^i MCs will be executing on any given assigned MC.

Proof: Consider an SIMD task which requires 2^i MCs. The set of MCs which is executing this task will be denoted by S_i. The virtual page fault rate for the task is Λ. When a page fault occurs, only one of the MCs in the set S_i reports the fault to the Control Storage Controller. If MC_j, $j \epsilon S_i$, is reporting the page faults, $\nu_j = \Lambda$ and $\nu_k = 0$ for all $k \epsilon S_i$ and $k \neq j$. Therefore, from the set S_i of 2^i MCs, only one page request is generated for each task page fault. Thus,

$$\sum_{k \epsilon S_i} \nu_k = \Lambda .$$

The average virtual page request rate for MC_j, where $j \epsilon S_i$ is defined as:

$$\mathrm{Avg}[\nu_j \mid j \epsilon S_i] = \frac{1}{2^i} \sum_{k \epsilon S_i} \nu_k = \frac{1}{2^i} \Lambda .$$

The notation Avg[x] denotes the average value of x. The average of the virtual MC page request rates, $\bar{\nu}$, can then be calculated by taking the average value of ν_j over all possible task sizes. Hence,

$$\bar{\nu} = \mathrm{Avg}[\nu_j] = \sum_{i=0}^{q} R_i \, \mathrm{Avg}[\nu_j \mid j \epsilon S_i] = \sum_{i=0}^{q} R_i \frac{\Lambda}{2^i} .$$

The system page request rate, λ_{sys}, is defined as:

$$\lambda_{sys} = \sum_{j=0}^{Q-1} \lambda_j = \sum_{j=0}^{Q-1} U_j \nu_j .$$

Assuming that the utilization for all of the MCs is the same, i.e., $U_j = U_{mc}$, where $0 \leq j \leq Q$, the average value of the system page request rate, $\bar{\lambda}_{sys}$, is:

$$\bar{\lambda}_{sys} = \mathrm{Avg}[\lambda_{sys}] = \mathrm{Avg}[\sum_{j=0}^{Q-1} U_j \nu_j]$$

$$= \sum_{j=0}^{Q-1} \mathrm{Avg}[U_j \nu_j] = \sum_{j=0}^{Q-1} U_{mc} \bar{\nu} = QU_{mc}\bar{\nu} .$$

Substituting in the equation for $\bar{\nu}$,

$$\bar{\lambda}_{sys} = QU_{mc} \sum_{i=0}^{q} R_i \frac{\Lambda}{2^i} = QU_{mc}\Lambda \sum_{i=0}^{q} \frac{1}{2^i} R_i ,$$

which is the desired result.

□

It is noted that the system page request rate is dependent on Q, the number of Micro Controllers and is independent of N, the number of Parallel Computation Unit processors. Theorem 1 is generalized to account for the fact that the task virtual page fault rate Λ may vary for tasks requiring different numbers of MCs in the following corollary.

Corollary 1: The average system page request rate, $\bar{\lambda}_{sys}$, is:

$$\bar{\lambda}_{sys} = QU_{mc} \sum_{i=0}^{q} \frac{\Lambda_i}{2^i} R_i ,$$

where Λ_i is the virtual page fault rate for the instruction stream of a task which requires 2^i MCs.
Proof: Follows directly from the proof of Theorem 1. □

When an MC is not executing, it is either waiting for a page request to be serviced by the Control Storage or it does not have a task assigned to it to execute. The *virtual utilization* is the utilization of the MC while it has a task assigned to it and is denoted by U'_{mc}. The *assignment ratio* is the fraction of time an MC has a task assigned to it. If $\overline{A}$ is the average MC assignment ratio, then $U_{mc} = \overline{A}\, U'_{mc}$. Note that if the MCs are always assigned tasks, then $U_{mc} = U'_{mc}$. The (real) page fault rate for a task may now be defined as $U'_{mc}\Lambda$ since the virtual utilization only accounts for the time that a task is assigned to a group of MCs.

The multitasking level, T, is defined to be the number of tasks which are executing on the system at a given time. The average system page request rate may be defined in terms of $\overline{T}$, the average multitasking level, and $U'_{mc}\Lambda$, the (real) task page fault rate, to be: $\overline{\lambda}_{sys} = \overline{T}U'_{mc}\Lambda$. Combining this with the result of Theorem 1, the average multitasking level is determined to be:

$$\overline{T} = \frac{\overline{\lambda}_{sys}}{U'_{mc}\Lambda} = \frac{\overline{A}\,\overline{\lambda}_{sys}}{U_{mc}\,\Lambda} = \overline{A}\sum_{i=0}^{q} R_i\,\frac{Q}{2^i}\,.$$

Hence, instead of Q tasks executing, the average multitasking level is $\overline{T}$, which for the uniform distribution case with all MC assigned tasks ($\overline{A} = 1$) would be: 80/31 and the average system page request rate is:

$$\overline{\lambda}_{sys} = \overline{T}U'_{mc}\Lambda = \frac{80}{31}\,U'_{mc}\Lambda = 2.58\,U'_{mc}\Lambda\,,$$

where $U'_{mc}\Lambda$ is the task page fault rate.

In conclusion, in the case where all 16 MCs are executing tasks it might be expected that the average system page request rate would $16U'_{mc}\Lambda$, where $U'_{mc}\Lambda$ is the page fault rate for the task running on each MC. In this section it has been determined that the average page request rate for the system is only $2.58U'_{mc}\Lambda$ when there is a uniform distribution of task sizes. Hence, the average system page request rate is only 16.1 per cent of what might be expected when all 16 MCs are executing tasks. The worst case system page fault rate is $16U'_{mc}\Lambda$ which occurs when each MC is executing an independent task. On the other hand, when all MCs are executing the same task, the system page fault rate is $U'_{mc}\Lambda$. The average multitasking levels for a variety of P_i distributions are given in Table V.1.

Table V.1: Average multitasking level, $\overline{T}$, for a variety of task size distributions. The task size is the number of MCs a task requires. P_i is the probability that a task which requires 2^i MCs is created.

P_0	P_1	P_2	P_3	P_4	$\overline{T}$
0.20	0.20	0.20	0.20	0.20	2.58
0.00	0.25	0.25	0.25	0.25	2.13
0.00	0.00	0.33	0.33	0.33	1.72
0.00	0.00	0.00	0.50	0.50	1.33
0.00	0.00	0.00	0.00	1.00	1.00
1.00	0.00	0.00	0.00	0.00	16.00
0.50	0.50	0.00	0.00	0.00	10.67
0.33	0.33	0.33	0.00	0.00	6.86
0.25	0.25	0.25	0.25	0.00	3.75
0.23	0.23	0.23	0.23	0.08	3.38
0.50	0.00	0.00	0.00	0.50	1.88
0.00	0.50	0.00	0.00	0.50	1.78
0.00	0.00	0.50	0.00	0.50	1.60
0.00	0.00	0.00	0.50	0.50	1.14
0.00	0.00	0.50	0.50	0.00	2.67
0.00	0.00	0.00	1.00	0.00	2.00
0.00	0.00	1.00	0.00	0.00	4.00
0.00	1.00	0.00	0.00	0.00	8.00
0.00	0.33	0.33	0.33	0.00	3.34
0.10	0.25	0.30	0.25	0.10	2.96

VI. Optimal Control Storage Service Rate

Criterion for optimal memory management in multiprogrammed systems have been given in [2,4,8]. The optimum is characterized by maximal system service rate, and in turn by maximal processor utilization and minimal response time [4]. One such criterion is the 50% criterion. The *50% criterion* for optimal memory management states that in a multiprogrammed system with page request rate λ, the use of the CPU is "optimized" when the disk service rate $\mu = 2\lambda$ so that the disk is 50% utilized [8]. So for $\mu \leq 2\lambda$, as μ is increased, the system service rate is increased significantly, and for $\mu > 2\lambda$, as μ is increased, the system service rate does not increase significantly. Thus, $\mu = 2\lambda$ is considered "optimal."

For the class of systems studied in [8], it was determined that the utilization of the secondary storage which resulted in optimal memory management was 50%. In order to determine the appropriateness of the 50% criterion for the PASM MC secondary storage, MC utilization was used as a performance measure. Figure VI.1 is a graph of the MC utilization as a function of the Control Storage utilization which was generated from simulation data (details of simulation are in [11]). There are three optimal values for the Control Storage utilization in PASM: 32.5%, 50%, and 62.5%. They are all considered optimal since the increase in MC utilization resulting from a small decrease in Control Storage utilization is much less the the decrease in the MC utilization resulting from a small increase in the Control

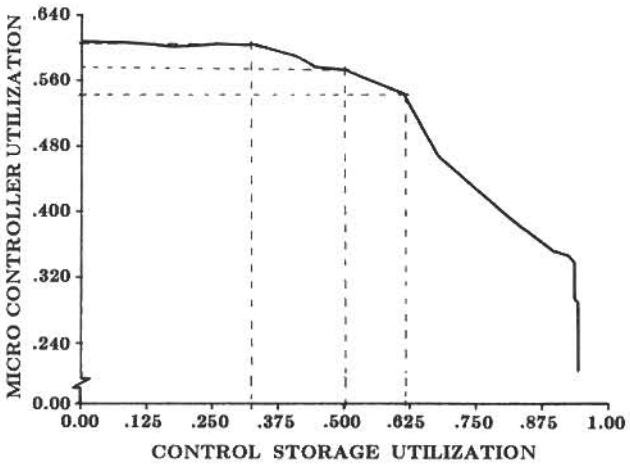

Figure VI.1: MC utilization as a function of Control Storage Utilization.

Storage utilization. The selection of the optimal Control Storage utilization to be used may depend on factors such as desired speed and cost of available secondary storage devices (e.g., disks).

It is noted that even when the Control Storage utilization is 0% (all page faults are serviced instantaneously), the MC utilization in Figure VI.1 is not 100%. This is due to other factors which impact the MC utilization besides the Control Storage utilization, such as availability of tasks to be scheduled [12] and fragmentation of MCs (i.e., available MCs do not form allowable group). For example, the MC utilization in Figure VI.1 could be increased if the task interarrival rate was increased, but its shape would remain similar.

To apply the optimal result to PASM, the Control Storage service rate μ must be selected so that $U_{cs}\,\mu = \bar{\lambda}_{sys} = \bar{T}\,U'_{mc}\,\Lambda$, where U_{cs} is the Control Storage utilization. Hence, $\mu = (\bar{T}\,U'_{mc}\,\Lambda) / U_{cs}$. If the optimal Control Storage utilization of 50% is selected, $\mu = 2\,\bar{T}\,U'_{mc}\,\Lambda$. In the case where there is a uniform distribution of the sizes of tasks created (derived in the previous sections), $\mu = 5.16\,U'_{mc}\,\Lambda$. If the virtual MC utilization is assumed to be one, then $\mu = 5.16\,\Lambda$. In actuality, U'_{mc} would be less than one, and a value other than one could be used here. Therefore, based on the assumption of a uniform distribution of the sizes of tasks created, the Control Storage service rate should be set to 5.16 times the virtual task page fault rate.

VII. Micro Controller Idle Time

Since the MCs in PASM are not multiprogrammed, there is not another task for an MC to execute while it is waiting for a page request from its current task to be serviced by the Control Storage. In this section, operational analysis is used to determine the MC idle time which results from an MC waiting for a page request to be serviced by the Control Storage. Note that this does not include the time that the MC is idle while it does not have a task assigned to it. This derivation makes use of Little's Law and is similar to that of the "Interactive Response Time Formula" for a terminal system in [3]. Let $\bar{m}$ be the mean queue length for a device (including the request which is being serviced); let X_0 be the throughput of the device; and let R be the accumulated time at that device per request (time spent by the request in the queue of the device while waiting for service plus the service time of the device). Then Little's Law [3] is: $\bar{m} = X_0\,R$.

Let I denote the average MC idle time and Z denote the average time interval between when an MC resumes execution after a page request is serviced and when its next page fault occurs. Hence, Z is the average execution time or busy time between page faults for a given MC. Each task is executed by a group of one or more MCs. Since each MC can have at most one instruction stream associated with it, the system has a finite customer population [6] (i.e., there is a finite number of page requests waiting to be serviced by the Control Storage at any given time since each MC cannot have another page fault while it is waiting for its current request to be serviced).

A task repeats "busy-idle cycles" while it has a group of MCs assigned to it. A *busy-idle cycle* has two phases: the busy phase, when the group of MCs which is assigned to the task is executing, and the idle-phase, when the group of MCs is waiting for a page fault to be serviced. The mean time for a task to complete one busy-idle cycle on a group of MCs is I + Z. Note that Z is the average time spent in the MC subsystem and I is the average time spent in the Control Storage subsystem during each busy-idle cycle (see Figure IV.1). Since all tasks which are assigned to MCs are repeating busy-idle cycles, $\bar{m}$ is equal to $\bar{T}$, the average multitasking level. Let X_0 be the throughput of the Control Storage. Applying Little's Law, $\bar{T} = X_0(I + Z)$.

The throughput, X_0, is the product of the utilization, U_{cs}, and the service rate, μ. So the throughput of the Control Storage is $U_{cs}\mu$. The average execution time between page faults, Z, is $1/\nu$ where ν is the MC virtual page fault rate. Therefore, the average MC idle time is:

$$I = \frac{\bar{T}}{X_0} - Z = \frac{\bar{T}}{U_{cs}\mu} - \frac{1}{\nu}\,.$$

This maps to the "Interactive Response Time Formula" in [3] by letting the MC busy time correspond to the user think time, the MC idle time correspond to the user wait time, the number of tasks executing (i.e., multitasking level) corresponds to the number of terminals, and the Control Storage throughput corresponds to the throughput of the central server.

Suppose the results for the optimal service rate μ from the previous section are used. Then the Control Storage service rate $\mu = 5.16\Lambda$ and

$$I = \frac{\bar{T}}{U_{cs}5.16\Lambda} - \frac{1}{\Lambda} = \frac{\bar{T} - U_{cs}5.16}{U_{cs}5.16\Lambda}\,.$$

Next the worst case situation is considered. In the worst case the Control Storage is completely utilized, so $U_{cs} = 1$ and

$$I = \frac{\bar{T} - 5.16}{5.16\Lambda}\,.$$

If the Control Storage is completely utilized, the system page fault rate λ_{sys} must be greater than or equal to the Control Storage service rate μ, so $\lambda_{sys} \geq \mu$. Hence, based on the 50% criterion and the assumptions used to select the Control Storage service rate μ, $\lambda_{sys} \geq 5.16\Lambda$ and $\bar{T} \geq 5.16$. If during some time interval there are 16 tasks executing (i.e., a multitasking level of 16), the MC idle time during that time interval would be:

$$I = \frac{16 - 5.16}{5.16\Lambda} = \frac{10.84}{5.16\Lambda} = 2.1\,\frac{1}{\Lambda}\,.$$

The time which an MC is busy executing a task is the time between page faults, $1/\Lambda$. The time which an MC is waiting for a page request to be serviced is its idle time, I. During a time interval when there are 16 tasks executing, the fraction of time which a given MC would be idle is:

$$\frac{I}{I + (1/\Lambda)} = \frac{2.1(1/\Lambda)}{3.1(1/\Lambda)} = 0.677\,.$$

So, if during some interval of time the average multitasking level was 16 (i.e., worst case level), the MCs would be idle 67.7% that time interval. Based on the simplifying assumptions in Section V used to compute the optimal Control Storage service rate μ, the probability of the worst case occurring is less than 0.1%. Note that if the probability had been greater, it would have impacted the calculation for the average system page fault rate $\bar{\lambda}_{sys}$ which would have resulted in a faster

Control Storage service rate μ.

VIII. Simulation Results

In this section results from the PASMOS simulator [11] are given. The simulator has been run with a variety of average execution times and distributions for the number of MCs a task requires. The results of all runs agree with the analytical result of Section V. As an example, consider the following simulation run where a random number generator was used to produce a uniform distribution (i.e., $P_i = 0.2$) for the number of MCs a task requires and the expected execution time for the tasks was fifteen seconds. The simulation ran for twenty thousand simulation seconds and over two thousand tasks were executed. The resulting distribution for the number of MCs required by a task for the simulation was: $P_0 = 0.191$, $P_1 = 0.204$, $P_2 = 0.198$, $P_3 = 0.208$, and $P_4 = 0.199$. The resulting average execution time for a task which requires 2^i MCs for the simulation was: $\bar{E}_0 = 15.006$, $\bar{E}_1 = 15.518$, $\bar{E}_2 = 14.215$, $\bar{E}_3 = 14.715$, and $\bar{E}_4 = 15.869$. The average MC assignment ratio $\bar{A}$ was 0.606 and the average multitasking level $\bar{T}$ was 1.531 streams. (Further details of the simulator are beyond the scope of this paper and are given in [11]. A description of the task scheduling algorithm which was used by the simulator is given in [12].)

Using the equation from Section V for R_i, the probability that a task which requires 2^i MCs is executing on a given assigned MC, with the $\bar{E}_i$s and P_is from the simulation, the results are found to be: $R_0 = 0.030$, $R_1 = 0.066$, $R_2 = 0.118$, $R_3 = 0.256$, and $R_4 = 0.529$. Substituting the MC utilization and the R_is into the equation for the average multitasking level:

$$\bar{T} = \bar{A}\sum_{i=0}^{q} R_i \frac{Q}{2^i} = (0.606)\sum_{i=0}^{4} R_i \frac{16}{2^i} = 1.530 ,$$

the average multitaking level in found to be 1.530.

Hence, by using the analytical method of Section V with the system characteristics from the simulation it has been determined that the average number of independent instruction streams is 1.530. The simulation results give the average number of instruction streams to be 1.531. Therefore, the simulation results support the analysis in Section V.

The simulator may also be used to confirm the worst case MC idle time result from Section VII. The Control Storage service rate and task page fault rate were selected so that $\mu = 5.16\Lambda$. To create the worst case situation, the distribution of the number of MCs required by a given task was adjusted so that all tasks would require one MC. The average execution time was adjusted so that the assignment ratio for all MCs would be one and the average number of tasks executing, $\bar{T}$, would approach sixteen. The resulting average multitasking level was 16.0; the Control Storage utilization, U_{cs}, was 1.0; and the average MC was idle for 67.5% of the simulation time. This result agrees with the expected result from the analysis in Section VII, where in the worst case (i.e., the multitasking level is 16) the average MC was idle 67.7% of the time. Again it is noted that the probability that this worst case would occur is very small.

IX. Relation to the General Multiple-SIMD Model

A general model of a multiple-SIMD system is shown in Figure IX.1. There is a pool of control units with a common secondary storage, a pool of processing elements, a switch which is used to connect a control unit to a group of processing elements, and an interconnection network for communication among the processing elements. In the case of PASM, the switch is fixed in that each processing element is connected to exactly one control unit (MC), and large machines are created by combining control units (MCs). Other MSIMD systems, such as MAP [9], use a crossbar type of switch to assign the processing elements to the control units. All of the control units are not always used. When PASM is executing an SIMD task only one of the MCs which is executing the task is used to make requests for pages.

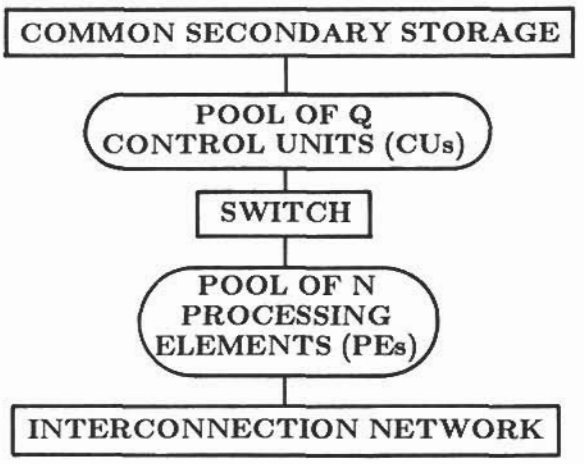

Figure IX.1: A general model of a multiple-SIMD machine.

The optimal service rate analysis may be applied to the general case by letting the used control units correspond to the MCs which are making the requests and the unused control units correspond to the MCs which are executing tasks but not making page requests. Thus the analysis applies to the general case where the SIMD machines have a power of two processing elements. The power of two constraint may also be eased to allow any size SIMD machine.

X. Conclusion

In this paper a queueing network has been analyzed to determine the "optimal" page request service rate for the Control Storage of the PASM multimicrocomputer system. It has been shown that the optimal service rate for the PASM Control Storage is much lower than might be expected. Two possible methods for varying the Control Storage service rate include varying the number or type of disks, or changing the method of storing pages on the disks. Simplifying assumptions were made about average execution time, task page fault rate, the distribution of the number of MCs which a task requires, etc. Based on experience any or all of these assumptions can be changed to reflect actual or expected system characteristics. Operational analysis has been used to determine the MC idle time which results from MCs having to wait for page requests to be serviced. Simulation results have been given which support the analytical result for the average number of independent instruction streams and the worst case MC idle time.

This study can also be applied to the use of PASM in the MIMD mode of operation or in a combination of

the MIMD and SIMD modes. When PASM is operating as a number of virtual MIMD machines of varying sizes, the MCs may be used to help coordinate the activities of the Parallel Computation Unit processors. The coordination activity of MIMD mode requires the MCs to execute significantly fewer instructions than in the control activity of SIMD mode. Hence, the page request rate is significantly lower for MIMD mode than for SIMD mode. Since it is expected that in MIMD mode each MC will have its own instruction stream, it can be treated as one SIMD partition, and incorporated into the run-time statistics.

In summary, a model has been developed for the PASM control system memory hierarchy. Using any combination of system feature assumptions and actual system characteristics (from experience), the model can be used to determine the "optimal" service rate for the Control Storage. Furthermore, using the model, values for the parameters which characterize the expected task environment and secondary storage service rate can be varied to determine the impact on MC utilization. The model can be adapted for use in any multiple-SIMD machine with common secondary storage for the multiple control units.

Acknowledgment

The authors would like to thank Prof. Peter J. Denning, Prof. Dorothy E. Denning, George B. Adams III, and Robert J. McMillen for their comments and suggestions.

References

[1] P. J. Denning, "Virtual memory," *Computing Surveys*, vol. 2, pp. 153-189, Sep. 1970.

[2] P. J. Denning, "Working sets past and present," *IEEE Trans. Soft. Engr.*, vol. SE-6, pp. 64-84, Jan. 1980.

[3] P. J. Denning and J. P. Buzen, "The operational analysis of queueing network models," *Computing Surveys*, vol. 10, pp. 225-262, Sep. 1978.

[4] P. J. Denning and K. C. Kahn, "An L=S criterion for optimal multiprogramming," *Int'l. Symp. Comp. Performance, Modeling, Measurement, and Evaluation*, ACM, Mar. 1976, pp. 219-229.

[5] M. J. Flynn, "Very high-speed computer systems," *Proc. IEEE*, vol. 54, pp. 1901-1909, Dec. 1966.

[6] L. Kleinrock, *Queueing Systems, Vol. 1: Theory*, John Wiley and Sons, Inc., New York, 1975.

[7] J. T. Kuehn, H. J. Siegel, and P. D. Hallenbeck, "Design and simulation of an MC68000-based multimicroprocessor system," *1982 Int'l. Conf. Parallel Processing*, Aug. 1982, to appear.

[8] J. Leroudier and D. Potier, "Principles of optimality for multiprogramming," *Int'l. Symp. Comp. Performance, Modeling, Measurement, and Evaluation*, ACM, Mar. 1976, pp. 211-218.

[9] G. J. Nutt, "Microprocessor implementation of a parallel processor," *4th Symp. Comp. Architecture*, Mar. 1977, pp. 147-152.

[10] H. J. Siegel, L. J. Siegel, F. C. Kemmerer, P. T. Mueller, Jr., H. E. Smalley, Jr., and S. D. Smith, "PASM: a partitionable SIMD/MIMD system for image processing and pattern recognition," *IEEE Trans. Comp.*, vol. C-20, pp. 934-947, Dec. 1981.

[11] D. L. Tuomenoksa, *Design and Analysis of an Operating System for a Reconfigurable Multimicrocomputer System*, Ph.D. Dissertation, Purdue University School of Electrical Engineering, in preparation.

[12] D. L. Tuomenoksa and H. J. Siegel, "Analysis of multiple-queue task scheduling algorithms for multiple-SIMD machines," *3rd Int'l. Conf. Distributed Computing Systems*, Oct. 1982, to appear.