AD-787 008

INTERFERENCE IN MULTIPROCESSOR COMPUTER SYSTEMS WITH INTERLEAVED MEMORY

Forest Baskett, et al

Stanford University

Prepared for:

Army Electronics Command
Office of Naval Research
National Science Foundation

August 1974

AD 787008

BIBLIOGRAPHIC DATA SHEET	1. Report No. Technical Report No. 90	2.	3. Recipient's Accession No.
4. Title and Subtitle Interference in Multiprocessor Computer Systems with Interleaved Memory			5. Report Date August 1974
			6.
7. Author(s) Forest Baskett and Alan Jay Smith			8. Performing Organization Rept. No. STAN-CS-74-450
9. Performing Organization Name and Address Digital Systems Laboratory Stanford University Stanford, CA 94305			10. Project/Task/Work Unit No. 6930 and 6940
			11. Contract/Grant No. NSF GJ 35720 JSEP N-00014-67-A-0112-0044
12. Sponsoring Organization Name and Address National Science Foundation Washington, D. C. 20550 & Joint Services Electronics Program			13. Type of Report & Period Covered technical
			14.

15. Supplementary Notes

16. Abstracts

We analyze the memory interference caused by several processors simultaneously using several memory modules. We compute exact results for a simple model of such a system. We derive the limiting value for the relative degree of memory interference as the system size increases. The model of the limiting behavior of the system yields approximate results for the simple model and also suggests that the results are valid for a much larger class of models including those more nearly like real systems than the simple model. We test the assumptions and results of the simple model against some measurements of program behavior and simulations of systems using memory references from real programs. The model results provide a very good indication of the performance we should expect from real systems of this type.

17. Key Words and Document Analysis. 17a. Descriptors

Memory,
Memory Interference
Multi-processing
Interleaved Memory
Trace Driven Simulation

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement Approved for public release; distribution unlimited	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 44
	20. Security Class (This Page) UNCLASSIFIED	22. Price $3.75/2.25

283049

INTERFERENCE IN MULTIPROCESSOR COMPUTER SYSTEMS WITH INTERLEAVED MEMORY

1

by

Forest Baskett
Alan Jay Smith*

August 1974

Technical Report No. 90

Digital Systems Laboratory
Stanford University
Stanford, California

*The author's current address is: Department of Electrical Engineering and Computer Science, University of California, Berkeley, California 94720.

This work was partially supported by the Joint Services Electronics Program, contract N-00014-67-A-0112-0044; National Science Foundation Grant GJ 35720, and by Mr. Smith's National Science Foundation Fellowship.

INTERFERENCE IN MULTIPROCESSOR COMPUTER SYSTEMS WITH INTERLEAVED MEMORY

Forest Baskett and Alan Jay Smith*
Computer Science Department
Stanford University, Stanford, California 94305

ABSTRACT

We analyze the memory interference caused by several processors simultaneously using several memory modules. We compute exact results for a simple model of such a system. We derive the limiting value for the relative degree of memory interference as the system size increases. The model of the limiting behavior of the system yields approximate results for the simple model and also suggests that the results are valid for a much larger class of models including those more nearly like real systems than the simple model. We test the assumptions and results of the simple model against some measurements of program behavior and simulations of systems using memory references from real programs. The model results provide a very good indication of the performance we should expect from real systems of this type.

KEY WORDS: Memory, Memory Interference, Multi-Processing, Interleaved Memory, Trace Driven Simulation

COMPUTING REVIEWS CATEGORIES: 4.32, 6.21, 6.34, 8.1

---

* The author's current address is: Department of Electrical Engineering and Computer Science, University of California, Berkeley, California.

This work was partially supported by the Joint Services Electronics Program contract N-00014-67-A-0112-0044, National Science Foundation Grant GJ 35720, and by Mr. Smith's National Science Foundation Fellowship.

I. INTRODUCTION

Computer systems with multiple, independent memory modules and multiple, independent processors have been available for a number of years. Recent proposals for systems with a large number of primary processors as well as a large number of memory modules [Bell, 1972; Flynn, 1972], give added importance to the general question of the amount of memory interference caused by independent processors. In a system with N CPUs and M memory modules, independent programs may make simultaneous requests to the same memory module and interference will occur. If the memory modules are selected by low order bits of the addresses, as in typical interleaving, the memory interference could be severe. Our aim is to develop an abstract model of the operation of a multiple processor, multiple memory module system, determine the degree of memory interference in that model, and to assess the degree to which results from the model would correspond to actual system behavior.

II. SIMPLE MODEL

A simple model of N processors using M interleaved memory modules is the following Markov chain. All processors and memories are synchronized. At the beginning of each memory cycle, all the processors whose memory request from the previous memory cycle were satisfied make a new memory request. The request of each processor is directed to a particular memory module chosen at random, with all memory modules being equally likely to be chosen. Several requests may be for the same memory module. Each module will service exactly one request during the memory cycle if it has any requests before it. Any remaining requests are held for future memory cycles.

It should be clear that we are considering a system that is bound by the speed of its memory; that is, each processor always has a request waiting for the memory as soon as the memory is able to accept it. The system in question is also synchronous; all N processors make requests at the same time, and receive their data at the same time. Although such an organization will exactly describe only a minority of current or proposed multiprocessor systems (e.g., the Honeywell 645 has a slower processor cycle time than memory cycle time, and does not run its processors in synchrony [Sekino, 1972 and private communications]), it will be shown that one can use the results in this paper to obtain very close approximations to the true degree of memory interference in other systems.

We can define a complete set of states as follows: a 2N-tuple $(m_1, k_1, m_2, k_2, \ldots m_N, k_N)$ where m_i is the memory at which the reference from processor i is directed, and k_i is its position in the queue at that memory. Since all processors are assumed to have the same reference pattern (uniform random over the set of integers $1 \ldots M$), and since we are interested solely in the overall rate of memory access, not in the relative performance of one processor over another, both the queuing discipline and the position of each processor in each memory queue become irrelevant. Nor is it important to specify which processor is queued for which memory, rather we need specify only how many processors are queued for each memory. The state can now be represented as an M-tuple $K = (k_1 \ldots k_M)$, where k_i is the number of requests queued for memory i.

The number of such states $K = (k_1 \ldots k_M)$ is the same as the number of ways of distributing N balls (processors) into M boxes (memories) [Feller, 1966] $\binom{n+m-1}{m-1}$. At the end of a memory cycle, (before new memory requests are issued), the state of the system is represented by $H = (h_1 \ldots h_M)$ where $h_i = k_i - 1$ if $k_i > 0$, 0 otherwise. Another state $G = (g_1 \ldots g_M)$ is reachable in one step from $(k_1 \ldots k_M)$ if $g_i \geq h_i$ for all i. If $d_i = k_i - h_i$, and $x = \sum_1^M d_i$, then $P(K,G)$, the probability of transition from state K to state G is

$$P(K,G) = \frac{x!}{d_1!\, d_2!\, d_3! \ldots d_M!} \left(\frac{1}{M}\right)^x$$

as discussed by Bhandakar and Fuller [1973].

It should be clear without proof that this stochastic system is a Markov chain, since the choice of next state is affected only by the current state, is aperiodic, since from any state a transition to itself in one step is possible, and is irreducible, since any state can reach any other state in a finite number of steps.

It is possible to reduce the number of states again without losing useful information. Because the memory modules are identical, it is not necessary to associate queue lengths with specific memory modules. For example, a state (3,2,1,1) is equivalent to a state (1,2,1,3). We can thus deal with equivalence classes, where, for example, (2,1,1) would represent the states (1,1,2), (2,1,1) and (1,2,1). The number of states is now equal to the number of ways to partition N objects into M groups, where one or more of the groups may be empty. There is no closed form expression for this number, but for large N, M ($M \geq N$), it is asymptotic to

$$\frac{1}{4\pi\sqrt{3}} \exp \pi \left(\frac{\sqrt{2N}}{3} \right)$$

[Beckenbach, 1964], which grows much less quickly for large M, N than does $\binom{N+M-1}{M-1}$.

Baskett has devised a method of enumerating all of these (equivalence class) states and calculating the transition probabilities in an efficient manner. This calculation has been performed and described elsewhere [Chewning, 1973; Fuller and

Bhandarkar, 1973]; the results for a number of cases are displayed in Figures 1, 2, and 3.

It is easy to solve either the $N \times 2$ or the $2 \times M$ cases for all N or M.

(a) 2 Processors, M memories

$$\rho = 2 - \frac{1}{M}$$

(b) N Processors, 2 Memories

$$\rho = 2 - \frac{1}{N}$$

All attempts to find a simple closed form solution for the system described with finite M, N ($M > 2$, $N > 2$) have failed, and the authors are of the opinion that such a solution is unlikely. Two items lend support to this opinion -- the failure of the decomposition method (see Section IV), and the disproval of the following conjecture -- "that the throughput of an $N \times M$ (N processor, M memory) system is the same as the throughput of an $M \times N$ (M processor, N memory) system." This conjecture holds for either $M = 2$ or $N = 2$, for $M = N$, and in the limit as $N \to \infty$, $M \to \infty$, with M/N constant. It fails to hold, however, for the next most complicated case, the 3×4 system.

It should be noted that the model above is similar to a model proposed and partially analyzed by Skinner and Asher [1969]. They use a larger collection of states; since they are interested in which processor's memory request is satisfied, the model contains tie breaking probabilities in the case of memory usage conflict. This more detailed model only allows them to solve a very small number of cases, unfortunately. Other authors that have considered the memory interference problem (although with somewhat different models) are Budnick and Kuck [1971] and Flores [1964].

III. ASYMPTOTIC RESULTS

A. The Simple Model

As $N \to \infty$, $M \to \infty$, $N/M \to L$, a constant, there exists an exact solution. We can view each memory as the server in a queue, which operates in discrete time. At the end of each time interval, one service is completed if any customer is waiting for service (i.e., if any processor is waiting for the contents of a location in this memory). At the same time zero or more customers arrive. Let L_i be the equilibrium probability of being in state i, i.e., with i customers in the queue, and let p_{ij} be the probability of having j customers in the queue, given i customers there during the previous interval. Then, in equilibrium

$$L_j = \sum_{i=0}^{\infty} L_i p_{ij} \tag{1}$$

For very large M and N, p_{ij} becomes a function of $j-i$. Let A_i be the probability of i customers arriving at the end of an interval.

Then

$$p_{0j} = A_j$$

and

$$p_{ij} = A_{j-i+1}, \quad j \geq i - 1$$

$$p_{ij} = 0, \quad j < i - 1 .$$

Thus

$$L_j = \sum_{i=1}^{j+1} L_i A_{j-i+1} + L_0 A_j$$

Define

$$A(Z) = \sum_{j=0}^{\infty} A_j Z^j \quad , \quad L(Z) = \sum_{j=0}^{\infty} L_j Z^j .$$

as the generating functions of the arrivals (per interval) and the queue length, respectively. We then have

$$L(Z) = \sum_{j=0}^{\infty} \sum_{i=1}^{j+1} L_i A_{j-i+1} Z^j + L_0 A(Z)$$

By interchanging the order of summation and simplifying, we obtain

$$L(Z) = \frac{L_0 A(Z)(Z - 1)}{Z - A(Z)} \qquad (2)$$

By differentiating this expression with respect to Z , and evaluating at $Z = 1$, we obtain $\bar{L}$, the mean queue length.

$$L(Z) = L_0 A(Z)(Z-1)\big(Z-A(Z)\big)^{-1} \qquad (3)$$

$$\begin{aligned} \frac{d}{dZ} L(Z) &= L_0 A'(Z)(Z-1)\big(Z-A(Z)\big)^{-1} \\ &+ L_0 A(Z)\big(Z-A(Z)\big)^{-1} \\ &- L_0 A(Z)(Z-1)\big(Z-A(Z)\big)^{-2}\big(1-A'(Z)\big) \ . \qquad (4) \end{aligned}$$

As this expression is undefined at $Z = 1$, we collect terms and then apply L'Hôpital's rule twice.

We obtain

$$\bar{L} = L_0 \left[\frac{A''(1)+2A'(1)-2A'^2(1)}{2-4A'(1)+2A'^2(1)} \right] . \qquad (5)$$

A reader familiar with queueing theory will have noticed that the equations above are exactly the same as those used in obtaining results for the M/G/1 (Poisson arrival, general service time, single server) queue [Cox and Smith, 1961] . We can thus replace L_0 , the fraction of time that the queue is empty, by $(1 - \rho)$, where ρ is the symbol for the utilization of a server.

When N and M are finite, and the number of busy servers (memories) is known (and equal to b), the arrivals are binomially distributed as

$$P\{d \text{ arrivals}\} = \underset{0 \le d \le b}{\binom{b}{d}} \left(\frac{1}{M}\right)^d \left(1 - \frac{1}{M}\right)^{b-d}$$

When N , $M \to \infty$, the number of busy servers will become a constant fraction of the total number, equal to ρ . The distribution of the number of arrivals will become Poisson in the limit [Feller, 1966], i.e.,

$$P[i \text{ arrivals}] = \frac{\rho^i e^{-\rho t}}{i!} \qquad (6)$$

and the generating function of this distribution is

$$A(Z) = e^{\rho(Z-1)} \quad .$$

We can obtain an expression for $\bar{L}$ either directly from equation (5) or by referring to any queueing theory text [Saaty, 1961], which gives

$$\bar{L} = \rho + \frac{\rho^2}{2(1 - \rho)} \quad , \qquad (7)$$

From the structure of the problem, we know $\bar{L}$, the mean queue length (!) , it is simply N/M; since every processor is queued at exactly one memory, there are N customers among M servers. Thus

$$\frac{N}{M} = \rho + \frac{\rho^2}{2(1 - \rho)} \quad .$$

Solving for ρ , we obtain

$$\rho = \left(1 + \frac{N}{M}\right) - \left(\left(\frac{N}{M}\right)^2 + 1\right)^{1/2} \qquad (8)$$

The fraction of time then that a given memory is idle (due to interference between processors which are queued elsewhere) is:

$$\text{idle time} = 1 - \rho = \left(\left(\frac{N}{M}\right)^2 + 1\right)^{1/2} - \frac{N}{M} \quad .$$

We will show later that this asymptotic result is quite accurate even for systems in Which N and M are as small as 8 or 16. We will also shown in Section IV that a slight modification of this expression will yield even more accurate expressions for finite N and M.

B. Alternate Models

It is interesting to note that we have obtained exactly the same results as for an M/D/1 (Poisson arrivals, constant service time) queue which differs from the current problem in two important respects:

(a) arrivals are Poisson distributed, but arrive continuously, not in batches at the start of an interval, and

(b) service starts at exactly the time of an arrival to an empty queue, not at the beginning of an interval.

This model is <u>almost</u> equivalent to that of a paging drum with one sector and an infinite source Poisson arrival process. The difference is that the drum model has arrivals occurring continuously in time, rather than at discrete intervals, although services are synchronized in time and constant in duration. This drum model has been analyzed [Coffman, 1969; Skinner, 1967], and the result, using our previous notation is

$$\bar{L} = \rho\left(\frac{3}{2}\right) + \frac{\rho^2}{2(1-\rho)}$$

Letting

$$\bar{L} = \frac{N}{M} \quad \text{as before,}$$

$$\rho = \left[3 + 2\frac{N}{M} - \sqrt{\left(3 + \frac{2N}{M}\right)^2 - 16\frac{N}{M}}\right] \Big/ 4 \qquad (9)$$

Another variant of this model is to view it as a queueing network. Customers (processors) are served at a server (memory) and then branch with equal probability to one of the other servers.

Since every server is identical to every other, every branching probability is the same (uniform over all servers), and all customers are identical, in two special cases we can obtain a result. If either the service times are exponential (with mean 1 as before) [Jackson, 1963; Gordon and Newell, 1967] or constant, with LCFS preemptive service [Muntz and Baskett, 1972], we have

$$\bar{L} = \frac{\rho}{1 - \rho} = \frac{N}{M}$$

$$\rho = \frac{N/M}{1 + N/M} \qquad (10)$$

This approach has also been examined in Bhandarkar and Fuller [1973] for finite numbers of customers. As should be clear, as N , $M \to \infty$, each server in the queuing network becomes independent of every other; thus if each server is FCFS with constant service time, the previous M/D/1 queue result holds.

In Table 1 the results for a few simple examples using formulas 8, 9 and 10 are shown.

TABLE 1

MEMORY UTILIZATION (ρ) FOR $M \to \infty$, $N \to \infty$,

$N/M = L$, A CONSTANT

L	OUR MODEL M/D/1 MODEL	DRUM MODEL	QUEUEING NETWORK MODEL
1/2	.40455	.29289	.3333
1	.58579	.5	.5
1.5	.6972	.63397	.6
2	.7639	.7192	.6666

IV. APPROXIMATIONS

Because it is extremely time consuming in terms of computer time, as well as in programming effort, to calculate exact values for our simple model for finite M and N, it was considered desirable to find a useful approximation. A number of different approaches are considered and are discussed below; the binomial approximation is found to work well in all cases and is displayed in several tables and illustrations.

A. Balls and Boxes

The simplest and most obvious approximation to the simple model discussed in Sections II and III was used by Strecker [1970]. In his model the probability that K processors are queued for a specific memory is

$$P\{k = K\} = \binom{N}{K}\left(\frac{1}{M}\right)^{K}\left(1 - \frac{1}{M}\right)^{N-K} \qquad (11)$$

This is equivalent to the probability of a box having K balls, given that N balls are distributed randomly among M boxes. The probability that a memory is busy is just

$$1 - P\{K = 0\} = 1 - \left(1 - \frac{1}{M}\right)^{N} = \rho$$

The value obtained here is consistently low compared to the correct solution, and it is not hard to see why. Initially, the processor requests are described as indicated. In the next cycle, those processors serviced during the first cycle generate requests randomly, but those requests may find substantial queues in front of them. Thus longer queues tend to build up, lowering the overall utilization

B. The Decomposition Approximation

It was suspected that the model under consideration could be decomposed and analyzed piecemeal in some manner. Specifically, we tried analyzing a single (memory) queue, based on the assumption that all customers, (totaling K) not queued at the memory in question were distributed among the other $(M - 1)$ memories with precisely the same distribution that would occur at equilibrium in a $K \times (M - 1)$ system. Results for some $N \times 3$ systems were calculated in this manner and are compared with the exact results in Table 2. The numbers are very close, but not exact. These results are only marginally easier to obtain than the exact results. The failure of the system to decompose in this manner does add weight (as mentioned above) to our feeling that a closed form solution to this problem (if any such exists) will be very complex.

C. The Binomial Approximation

Earlier, in equation 5, we obtained an expression for the mean queue length $\bar{L}$, as a function of the moments of the arrival distribution, $A(n)$.

$$\bar{L} = L'(Z)\Big|_{Z=1} = L_0 \frac{A''(1) + 2A'(1) - 2A'^2(1)}{2\big(1 - A'(1)\big)^2} \qquad (12)$$

TABLE 2

DECOMPOSITION APPROXIMATION IN THE N PROCESSOR, 3 MEMORY SYSTEM

PROCESSORS	**FRACTIONAL MEMORY IDLE TIME**	
	DECOMPOSITION APPROXIMATION	**SIMPLE MODEL**
2	.4444	.4444
3	.3162	.3175
4	.2426	.2433
5	.1962	.1966
6	.1644	.1647
7	.1415	.1416
8	.12406	.1242

$A'(1)$ is simply the mean number of arrivals per interval, or ρ . $L_0 = 1 - \rho$ and $\bar{L} = N/M$ as before. Thus

$$\frac{N}{M} = \frac{A'(1) + 2A'(1) - 2A'^2(1)}{2\left(1 - A'(1)\right)^2} \qquad (13)$$

If we assume that the arrivals are binomially distributed (instead of Poisson distributed), i.e.,

$$P\{K \text{ arrivals}\} = A_K = \binom{N}{K} p^K (1 - p)^{N-K}$$

with

$$P(Z) = (1 - p + pZ)^N \text{ , and mean } np = \rho$$

then

$$A''(1) = n(n - 1)p^2 \quad \text{and} \quad A'(1) = np \quad .$$

p here is simply $1/M$, the probability of an arrival entering the queue in question. Solving for $np = \rho$, the mean arrival rate, we obtain

$$\rho = np = \frac{-\left(p - \frac{2N}{M} - 2\right) - \sqrt{\left(p - \frac{2N}{M} - 2\right)^2 - 8\,\frac{N}{M}}}{2} \qquad (14)$$

It is possible to extend the use of the binomial approximation beyond the simple model. In the simple model each CPU issues a memory request immediately on the completion of service by the memory of its previous request. Consider instead a CPU which, after receiving the desired information, "thinks" for a period of time with mean T and arbitrary distribution. The binomial approximation can be used for this model also, provided we are able to obtain an expression for the mean queue length at the memories, which is no longer N/M .

The mean cycle time of a customer (processor request) in the system will be $F(L) + 1 + T$, where T is the mean think time of the processor, 1 is the memory service time, and $F(L)$ is the mean queue length observed by a customer arriving at a memory, given that L is the equilibrium queue length as measured at an arbitrary time. Note that $F(L)$ is not necessarily equal to L , since the arrival of a customer is not independent of the state of the system. For example, an arriving customer will never see more than $N - 1$ customers in the queue ahead of him, although measured at arbitrary times, the queue length will at times be equal to N .

An expression for L then 's

$$L = \frac{F(L) + 1}{F(L) + 1 + T} \cdot \frac{N}{M} \quad . \tag{15}$$

In general, an expression for $F(L)$ is very difficult to obtain, and as a simple approximation $F(L)$ will be set equal to $\frac{N-1}{N} L$. Then

$$L = \frac{-\left(1 + T - \frac{N-1}{M}\right) + \sqrt{\left(1 + T - \frac{N-1}{M}\right)^2 + 4\left(\frac{N-1}{M}\right)}}{2\left(\frac{N-1}{N}\right)} \tag{16}$$

and

$$\rho = \frac{-\left(\frac{1}{M} - 2L - 2\right) - \sqrt{\left(\frac{1}{M} - 2L - 2\right)^2 - 8L}}{2} \qquad (17)$$

Comparisons of the binomial approximation to measured trace driven simulation results and/or simple model results are shown in Figures 1, 2, 3 and 4, and also in Table 3. As can be seen in these figures and tables, formulas (14) and (17) provide very accurate estimates of the interference we can expect in a real multiprocessor system with interleaved memory.

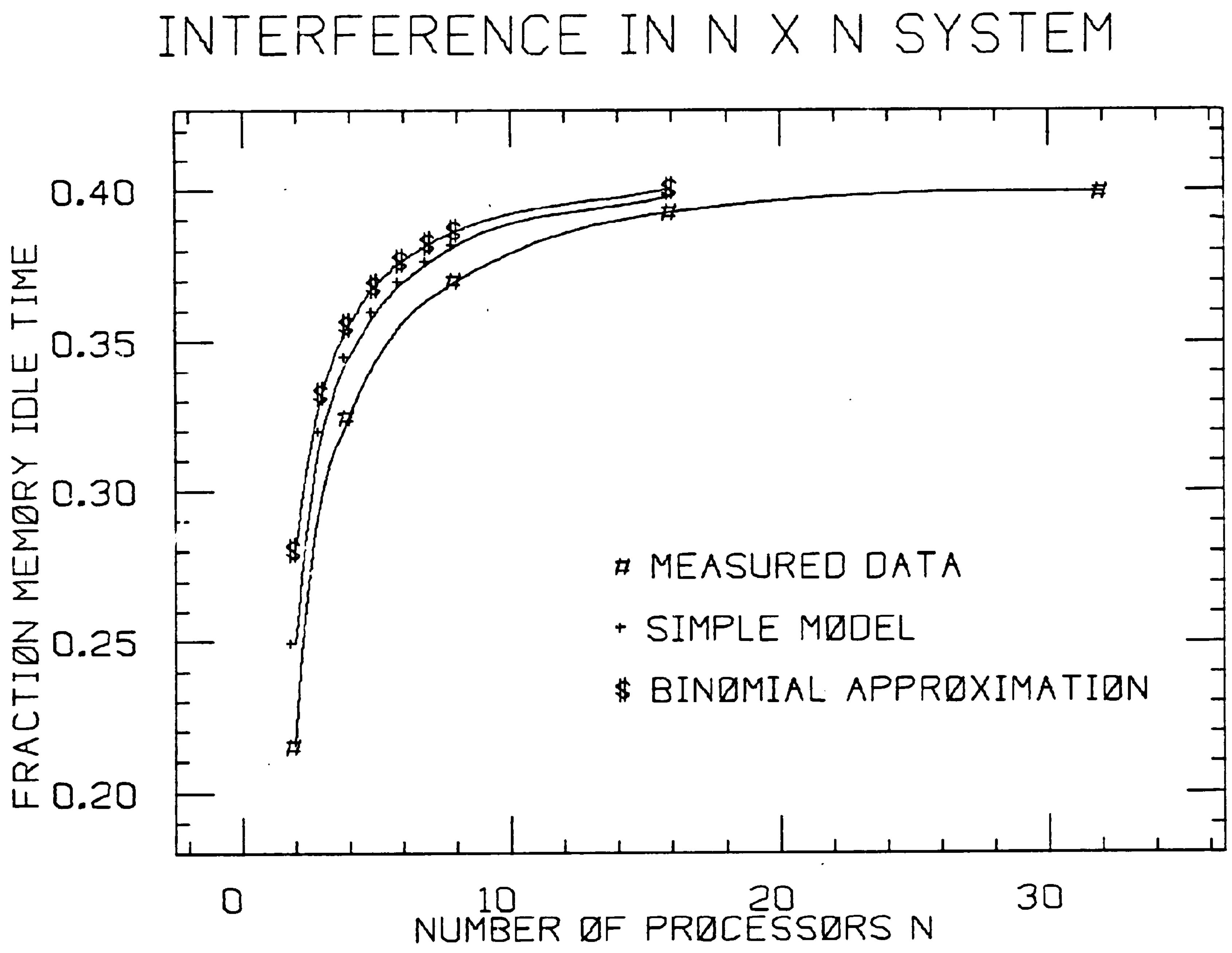

INTERFERENCE IN N X N SYSTEM
FRACTIØN MEMØRY IDLE TIME
0.40
0.35
0.30
0.25
0.20
NUMBER ØF PRØCESSØRS N
0
10
20
30
# MEASURED DATA
+ SIMPLE MØDEL
$ BINØMIAL APPRØXIMATIØN

FIGURE 1

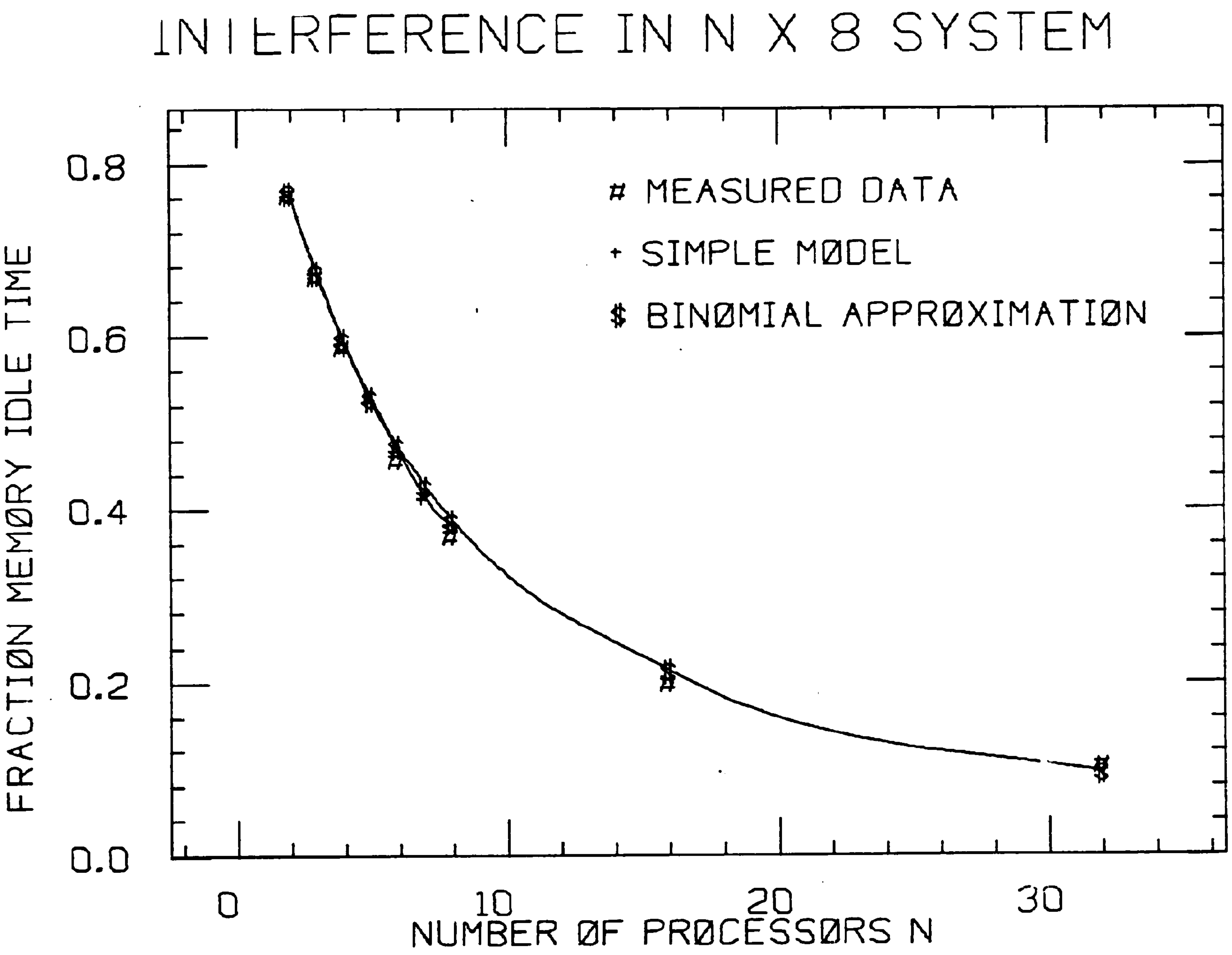

INTERFERENCE IN N X 8 SYSTEM
# MEASURED DATA
+ SIMPLE MØDEL
$ BINØMIAL APPRØXIMATIØN
FRACTIØN MEMØRY IDLE TIME
NUMBER ØF PRØCESSØRS N
0.0
0.2
0.4
0.6
0.8
0
10
20
30

FIGURE 2

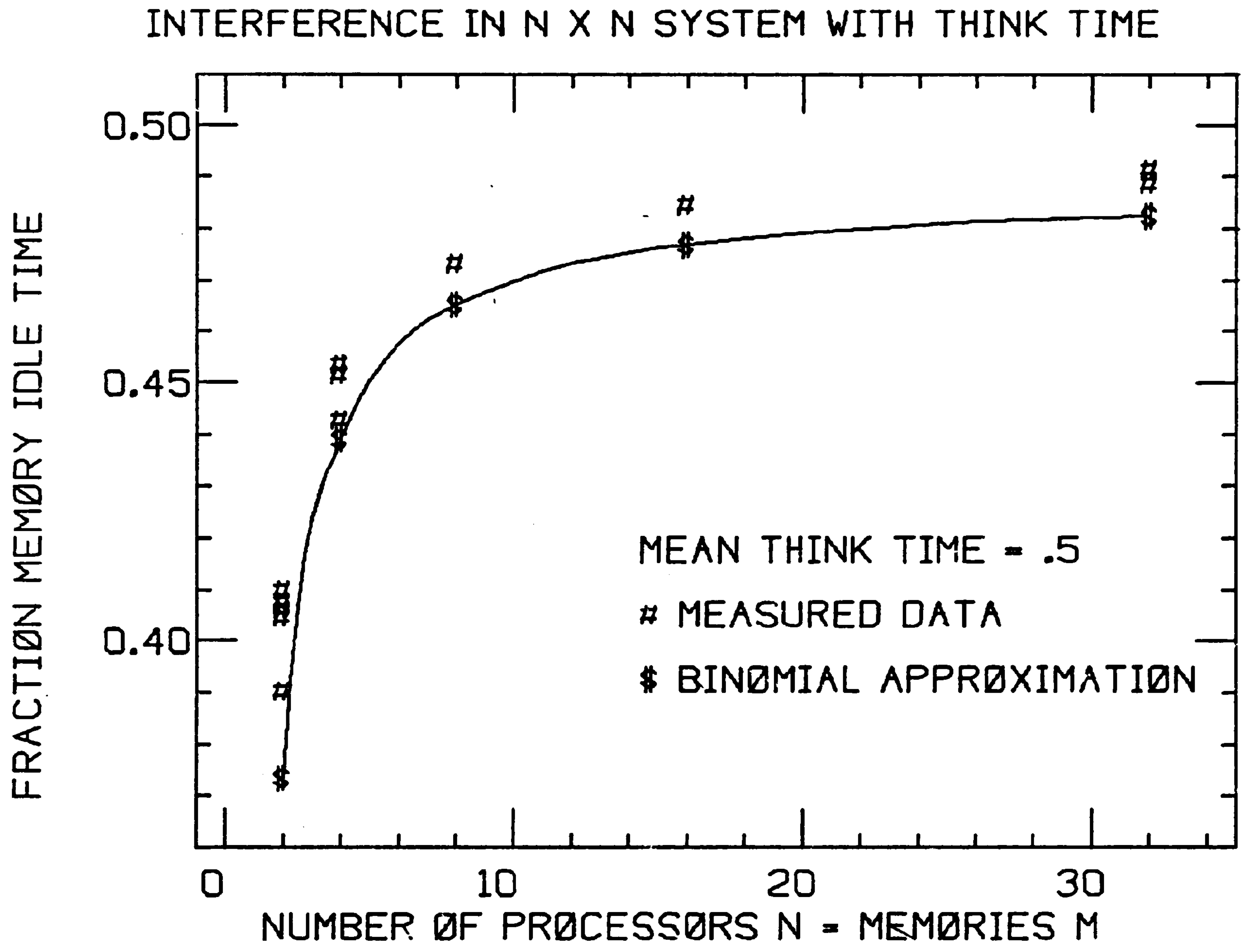
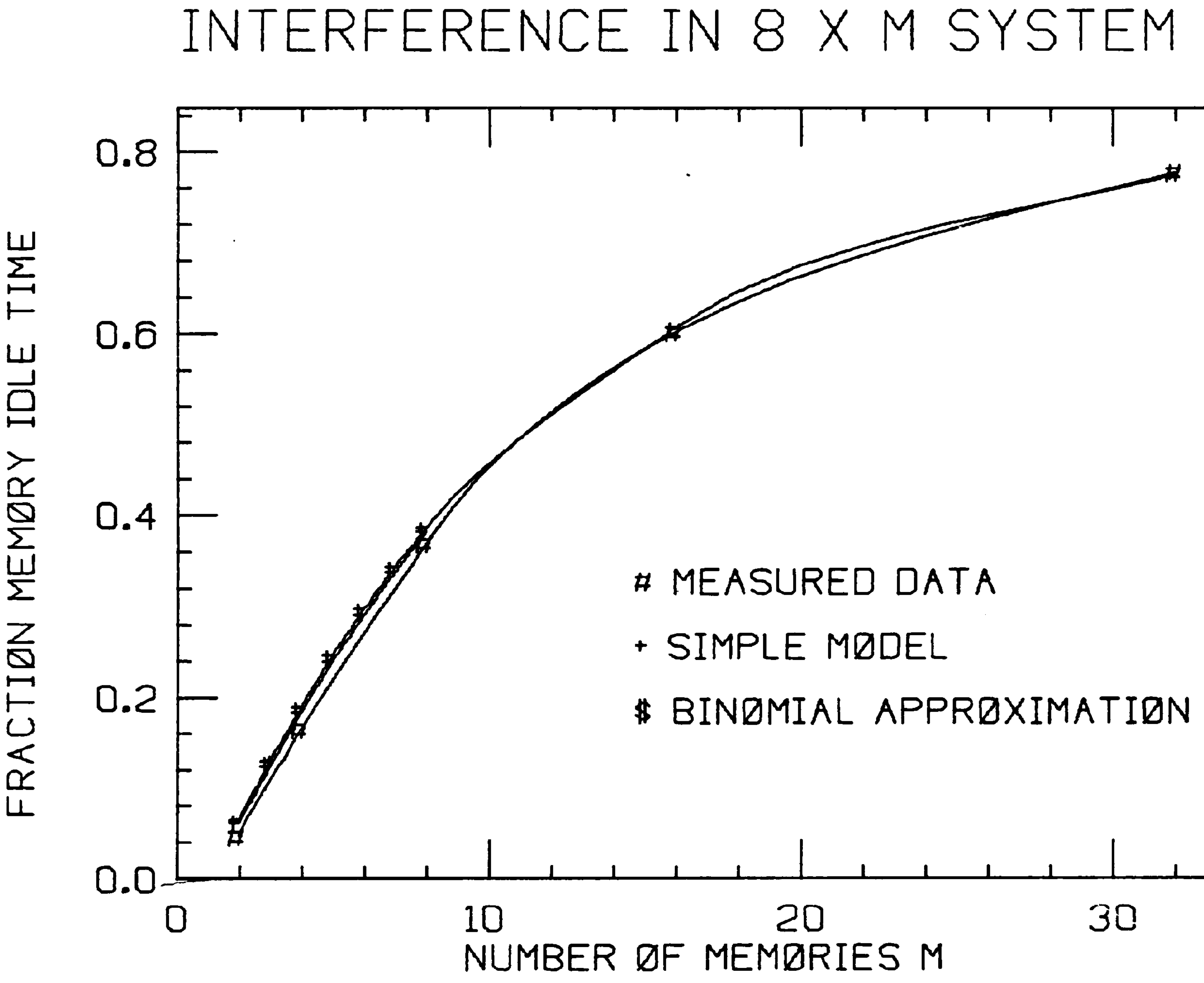

INTERFERENCE IN N X N SYSTEM WITH THINK TIME
FRACTIØN MEMØRY IDLE TIME
0.50
0.45
0.40
0
10
20
30
NUMBER ØF PRØCESSØRS N = MEMØRIES M
MEAN THINK TIME = .5
# MEASURED DATA
$ BINØMIAL APPRØXIMATIØN

FIGURE 3



INTERFERENCE IN 8 X M SYSTEM
FRACTIØN MEMØRY IDLE TIME
0.0
0.2
0.4
0.6
0.8
0
10
20
30
NUMBER ØF MEMØRIES M
# MEASURED DATA
+ SIMPLE MØDEL
$ BINØMIAL APPRØXIMATIØN

FIGURE 4

TABLE 3

INTERFERENCE WITH CPU'S USING THINK TIME

		% MEMORY IDLE			THINK TIME
CPU'S	MEMORIES	MEASURED	BINOM. APPROX.	MEAN	DISTRIBUTION
2	2	.4081	.3738	.5	Exponential
2	2	.3906	.3738	.5	Constant
2	2	.4109	.3738	.5	Hyper Exponential (CV=4)
2	2	.4079	.3738	.5	Erlang 2
2	2	.4065	.3738	.5	Erlang 3
2	2	.4050	.3738	.5	Erlang 4
4	4	.4519	.4392	.5	Exponential
4	4	.4431	.4392	.5	Constant
4	4	.4540	.4392	.5	Hyper Exponential (CV=4)
4	4	.4505	.4392	.5	Erlang 2
4	4	.4506	.4392	.5	Erlang 3
4	4	.4487	.4392	.5	Erlang 4
8	8	.4734	.4654	.5	Exponential
16	16	.4848	.4771	.5	Exponential
32	32	.4921	.4883	.5	Exponential
32	32	.4905	.4883	.5	Constant
32	32	.4923	.4883	.5	Hyper Exponential (CV=4)
32	32	.4908	.4883	.5	Erlang 2
32	32	.4888	.4883	.5	Erlang 3
32	32	.4916	.4883	.5	Erlang 4
4	4	.3255	.3560	.01	Exponential
4	4	.3479	.3727	.1	Exponential
4	4	.3777	.3895	.2	Exponential
4	4	.4269	.4230	.4	Exponential
4	4	.5160	.4876	.8	Exponential
4	4	.5530	.5176	1	Exponential
4	4	.6265	.5842	1.5	Exponential
4	4	.6823	.6385	2	Exponential

V. VALIDATION OF RESULTS, A TRACE DRIVEN SIMULATION

In order to test both the assumptions upon which our model is based and the results we have predicted, four different memory address traces were analyzed. Each of four different programs were interpreted, and a tape of the memory addresses referenced was produced. The programs interpreted include WATFIV, a Watfiv compiler, WATEX, the execution of a "typical" scientific computation program compiled under Watfiv, APL, the execution of a plotting program in APL, and FFT1, a fast Fourier transform program written in Fortran. The memory was assumed to be interleaved, and to be 64 bits (8 bytes) wide in each module, thus if the least significant bit of the address is referred to as bit 0, then bits 3-5 would give the module number in the case of an eight way interleaved memory.

To simulate an N processor system, N different sections of the same trace were used for each processor. In order to compensate for a non-uniformity within a trace of the modules most favored (receiving the largest number of memory requests) a linear offset was added to each address in a given section of each trace. Thus, the section of the trace belonging to processor i would have all the module numbers k translated to $k + i \bmod M$. As can be seen in Table 4, the interference observed when no offset was used was generally considerable higher than with an offset. A random (uniform [1,M]) offset (instead of linear) was also used, and as can be seen also in Table 4, when the number of modules M

TABLE 4

MEASURED INTERFERENCE USING DIFFERENT OFFSETS

NUMBER OF PROCESSORS	NUMBER OF MEMORIES	PERCENT IDLE IDLE	OFFSET
2	2	.2156	None
2	2	.2158	Linear
4	4	.3260	None
4	4	.3274	Random
4	4	.3252	Linear
8	8	.3721	None
8	8	.3703	Linear
8	8	.3724	Random
16	16	.4019	None
16	16	.3932	Linear
16	16	.3928	Random
32	32	.4405	None
32	32	.3996	Linear
32	32	.4017	Random

was large enough, the effect was indistinguishable from that of the linear offset. In Table 5 some results when two different traces were used instead of just one are shown, and no significant difference between the single and multiple trace simulations is evident. The remainder of the simulation results used one trace and a linear offset.

Figures 1, 2, and 3 show the results of trace driven simulations of our simple model structure, using the WATFIV trace. As can be seen, both the exact solution to the simple model and the binomial approximation are very close to the simulated measured results. In Table 6 the fraction of memory idle time in the N x N system is tabulated for simulations of each of the four memory address traces. There is no noticeable difference observed between the different traces; for this reason unless otherwise noted, all simulation results were obtained using the WATFIV trace. With some confidence we feel that the results are generally applicable.

Table 3 and Figure 4 show the results of trace driven simulations when the processor uses "think time". Six different distributions of think time were used, exponential, hyperexponential with coefficient of variation 4.4, constant, and Erlang with parameters 2, 3 and 4. The binomial approximation is again quite accurate, and the results seem to be insensitive to the distribution of think time.

TABLE 5

MEASURED INTERFERENCE IN SPECIAL CASES

NUMBER OF PROCESSORS	NUMBER OF MEMORIES	PERCENT IDLE TIME	SITUATION
2	2	.2156	No offset
2	2	.2141	No offset, exact same trace on each processor
2	2	.2178	No offset, different traces on each processor
2	2	.2180	Linear offset, different traces on each processor
4	4	.3300	No offset, exact same trace on all processors
4	4	.3263	Linear offset, two different traces
4	4	.3166	Random offset, same trace
8	8	.3593	Random offset, same trace
8	8	.3941	No offset, same trace on all processors

TABLE 6

MEMORY INTERFERENCE IN SYMMETRIC N x N SYSTEM

		MEMORY IDLE TIME			
		PROGRAM TRACE			
PROCESSORS	MEMORIES	WATFIV	WATEX	APL	FFT1
2	2	.2158	.2331	.2163	.2137
4	4	.3252	.3362	.3287	.3329
8	8	.3703	.3809	.3712	.3664
16	16	.3932	.3972	.4119	.3832
32	32	.3996	.4074	.4048	.3947

One of the basic assumptions in our simple model was that the memory contention effect could be accurately modelled by assuming that the memory reference strings generated by each processor were uniformly random over the set of memory modules and that each reference was independent of the preceding one(s). From the results discussed above (Figures 1-4 and Tables 3, 4, 5 and 6), this can be seen to be an adequate assumption, but it was found worthwhile to test this assumption further. A number of simulations were run using "sequential" processors. These processors generated entirely sequential memory reference strings, i.e., 1,2,3,4,5,6,7,8,9 etc. This, in a four way interleaved memory system would become memory modules 1,2,3,4,1,2,3,4,1 ... . Although it was expected that the interference would decrease substantially due to the sequential processor(s) becoming synchronized, it is evident from Table 7 that this decrease is rather small. These sequential processors operate in much the same manner as input/output devices such as disks and drums which produce just such sequential reference patterns; I/O devices seldom constitute a very significant fraction of the load on the memory, however. Clearly, then, the memory interference is not very sensitive to the degree of sequentiality in the memory module traces.

Other authors, such as Burnett and Coffman [1973] and Burnett [1970] have analyzed the memory interference problem by explicitly considering the degree of sequentiality in memory address traces. In order to measure the degree of sequentiality that did in fact

TABLE 7

INTERFERENCE WITH CPU'S AND SEQUENTIAL PROCESSORS

CPU'S MEMORIES	SEQUENTIAL PROCESSORS	% MEMORY IDLE
2	0	.2158
2	1	.1700
4	0	.3252
4	1	.3058
4	2	.2680
4	3	.2296
8	0	.3703
8	1	.3605
8	2	.3533
8	3	.3419
8	4	.2904
8	6	.2414

occur, additional analysis was performed on the memory address traces. If in the memory address trace, module i was referenced, and then next module j, a transition of $j - i \bmod M$ was recorded. For all j, $j \neq i+1 \bmod M$, references were found to occur reasonably evenly. In Table 8 the fraction of transition $j = i + 1 \bmod 32$ is shown, and it is clearly larger than any other transitions. The degree of sequentiality varied substantially among different programs, though, and no general figure can be stated.

The autocorrelation coefficient [Cox and Lewis, 1966] of the memory module trace was computed as a further test of sequentiality and patterns. It was found that whatever patterns that exist within each trace are not common between the traces. First order autocorrelations of up to .35 were found. This is further confirmed by the results displayed in Figures 5 and 6 where Fourier transforms of the memory module traces for WATFIV and FFT1 are shown. Clear and strong patterns are evident in the FFT program, which has a tight loop structure; that of WATFIV shows a different set of patterns, which are less prominent.

One further simulation was performed. Two, four, and eight processors were run, each with exactly the same trace. With a linear offset, clearly there is zero interference; with random and no offsets, the effects are comparable to either using different portions of the same trace or different traces, as can be seen in Table 5.

TABLE 8

TRACE	Probability that if reference j is to module i then reference j+1 is to module i+1 mod 32 .
WATFIV	.2650
WATEX	.1245
APL	.2815
FFT1	.1376
FFT2	.1349

FØURIER TRANSFØRM ØF MØDULE REFERENCE STRING

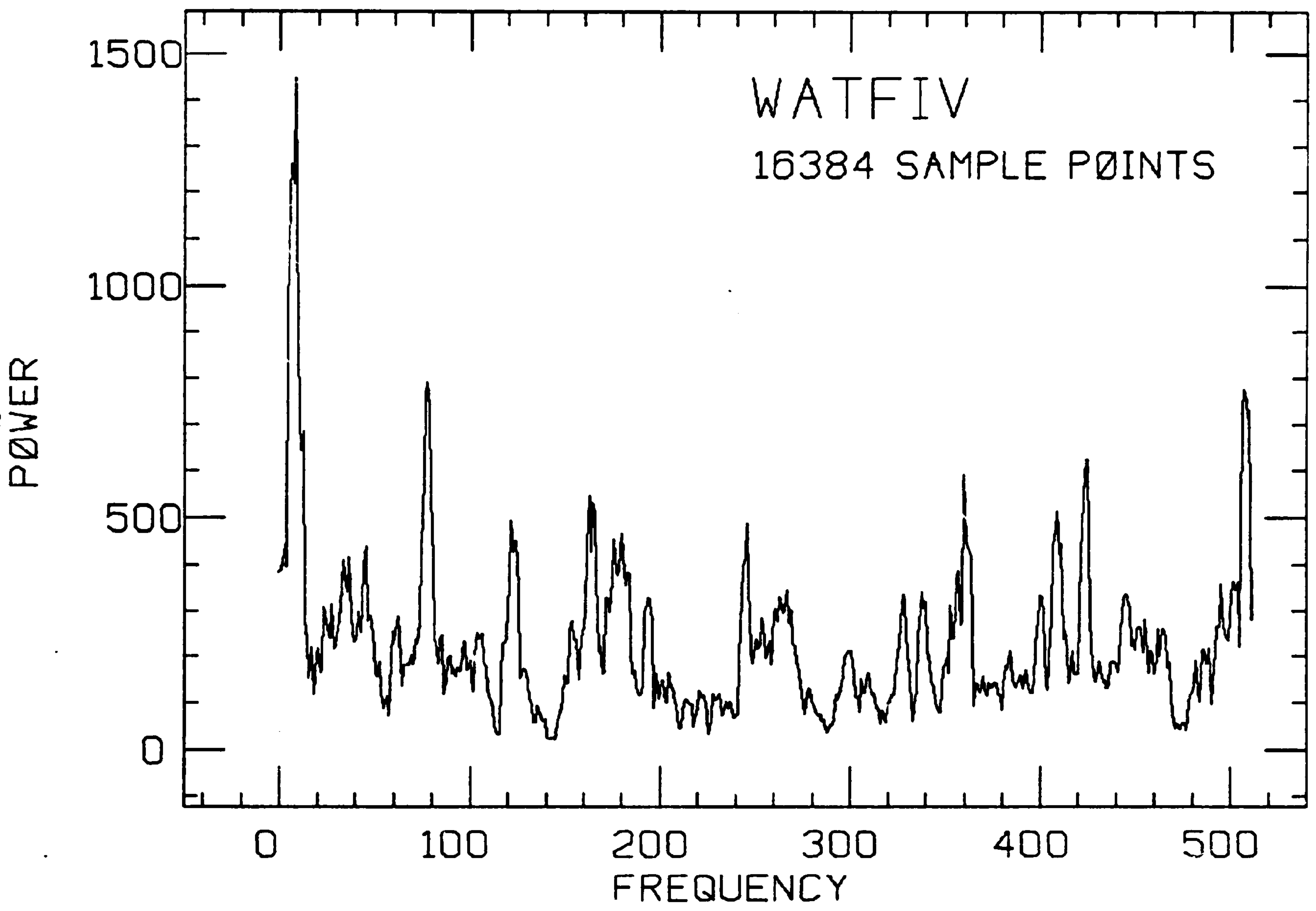


FIGURE 5

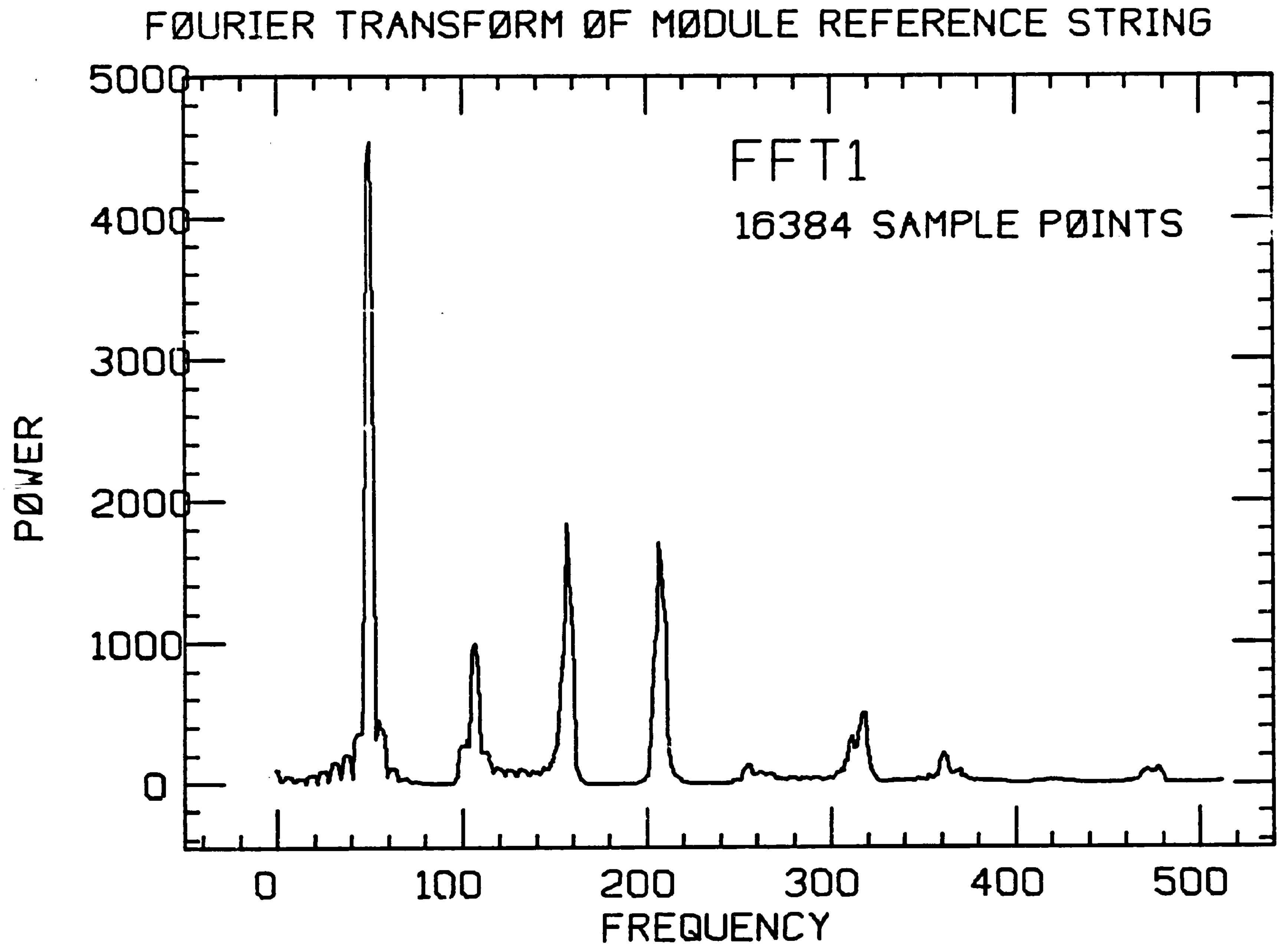

FØURIER TRANSFØRM ØF MØDULE REFERENCE STRING
FFT1
16384 SAMPLE PØINTS
PØWER
FREQUENCY
0
1000
2000
3000
4000
5000
0
100
200
300
400
500

FIGURE 6

VI. IMPLICATIONS AND CONCLUSIONS

We have proposed a simple model for a multiprocessor system with interleaved memory, which when modified using the binomial approximation can adequately represent many real computer systems. We have demonstrated with trace driven simulations that our analytic results are quite accurate and relatively insensitive to departures from our model. Such interference has been observed in real systems, such as Multics [Sekino, 1972] which running on the GE 645 lost about 5% of its memory cycles to interference, and the Honeywell 6000 series [private communication]. One major manufacturer uses as a rule of thumb that additional processors are to be considered as only .9 in processing power due to memory interference. Both of these results are compatible with our model.

Many modern computer systems use a cache memory [Conti, 1969], which is not shared between processors. One would expect that the address string arriving at the shared memory would have lost almost all of its sequentiality, and that our model would further improve in accuracy.

An issue not addressed here is the problem of memory lockout due to contention for serially but not concurrently accessible system tables. This problem has been considered briefly by Madnick [1968] and could easily be a problem of substantial magnitude if adequate steps are not taken to guard against it. Multics reports about the same degree of interference from table lockout as from memory conflict [Sekino, 1972].

Also not considered here is the possibility of organizing the contents of memory in some other manner than interleaving so as to minimize interference.

ACKNOWLEDGEMENTS

The authors would like to thank Leonard Shustek for generating the program address traces analyzed in this paper Roger Chafee and Robert Beach of the Stanford Linear Accelerator Center for the graphics system that made possible the illustrations in this chapter, and Talley Kenyon for her skillful typing of the manuscript.

BIBLIOGRAPHY

Beckenbach, E. (Ed.). Applied Combinatorial Mathematics, John Wiley and Sons, New York, (1964).

Bell, C. G. C.mmp, a multi-mini-processor, Proc. FJCC, (1972), 765-777.

Bhandarkar, D. P. and Fuller, S. H. A survey of techniques for analyzing memory interference in multi-processor systems, Carnegie-Mellon University Technical Report, Pittsburgh, Pennsylvania, (April, 1973).

Budnik, Paul and Kuck, David. The organization and use of parallel memories, IEEE Trans. on Comp., C-20, no. 12, (December, 1971), 1566-1569.

Burnett, Gerald J. Performance analysis of interleaved memory systems, Ph.D. Dissertation, Princeton University, Princeton, New Jersey, 1970.

Burnett, G. J. and Coffman, E. G., Jr. A combinatorial problem related to interleaved memory systems, J. of the ACM, 20, nol, no. 1, (January, 1973), 39-45.

Chewning, D. Multiprocessor memory interference, unpublished report, Department of Electrical Engineering, Stanford University, Stanford, California, (April, 1973).

Coffman, E. G., Jr. Analysis of a drum input/output queue under scheduled operation in a staged computer system, J. of the ACM, 16, no. 1, (January, 1968), 73-90.

Coffman, E. G., Jr., Burnett, G. J., and Snowdon, R. A. On the performance of interleaved memories with multiple word bandwidths, IEEE Trans. on Comp., C-20, no. 12, (December, 1971), 1566-1569.

Conti, C. J. Concepts for buffer storage, IEEE Computer Group News, (March, 1969), 9-13.

Cox, D. R. and Lewis, P. A. W. The Statistical Analysis of Series of Events, Methuen and Col, London, (1966).

Cox, D. R. and Smith, W. L. Queues, Chapman and Hall, London, (1961).

Feller, W. An Introduction to Probability Theory and its Applications, Volume I, John Wiley and Sons, New York, (1968).

Flores, Ivan. Derivation of a waiting time factor for a multiple bank memory, JACM, 11, no. 3 (July, 1964), 265-282.

Flynn, Michael J. Some computer organizations and their effectiveness, IEEE Trans. on Comp. C-21, no. 9, (September, 1972), 948-960.

Gordon, W. J. and Newell, G. S. Closed queueing systems with exponential servers, Operations Research, 15, (1967), 254-265.

Jackson, J. R. Jobshop-like queueing systems, Management Science, 10, no. 1, (October, 1963), 131-142.

Madnick, S. Multiprocessor software lockout, Proc. 1968 ACM National Conf., (1968), 19-24.

Muntz, Richard R. and Baskett, Forest. Open, closed, and mixed networks of queues with different classes of customers, Stanford Electronics Laboratories Technical Report No. 33, Stanford University, Stanford, California, (August, 1972).

Saaty, T. L. Elements of Queueing Theory, McGraw-Hill, New York, (1961).

Sekino, A., Performance evaluation of multiprogrammed time-shared computer systems, Ph.D. Dissertation, Project MAC Document MAC TR-103, Massachusetts Institute of Technology, Cambridge, Massachusetts, (September, 1972).

Singleton, Richard C. On computing the fast Fourier transform, CACM, 10, no. 10, (October, 1967), 647-654.

Skinner, C. E. Priority queueing systems with server-walking time, Operations Research, 15, no. 2, (1967), 2[illegible]85.

Skinner, C. E. and Asher, J. Effects of storage contention on system performance, IBM Sys. J., no. 4, (1969), 319-333.

Strecker, W. D. Analysis of the instruction execution rate in certain computer structures, Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, Pennsylvania, (1970).