DIGITAL SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES
DEPARTMENT OF ELECTRICAL ENGINEERING
STANFORO UNIVERSITY. STANFORD, CA 94305

SU-SEL 77-020 SU-326-P.39-22

PROGRAMBEHAVIOR ANDTHEPERFORMANCE OFINTERLEAVEDMEMORIES

by

B. Ramarishna Rau

May 1977

Technical Report No. 138

The work described herein was supported by the
U.S. Energy Research and Development Administration
under contract No. EY-76-S-03-0326-PA 39.
Computer time was made available by the
Stanford Linear Accelerator Center

SU-SEL 77-020 SU-326-P.39-22

PROGRAM BEHAVIOR AND THE PERFORMANCE OF INTERLEAVED MEMORIES

by

B. Ramarishna Rau

May 1977

Technical Report No. 138

DIGITAL SYSTEMS LABORATORY
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, CA 94305

The work described herein was supported by the U.S. Energy Research and Development Administration under contract No. EY-76-S-03-0326-PA 39. Computer time was made available by the Stanford Linear Accelerator Center.

SU-SEL 77-020 SU-326-P.39-22

DIGITAL SYSTEMS LABORATORY
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, CA 94305

Technical Report No. 138

PROGRAM BEHAVIOR AND THE PERFORMANCE OF INTERLEAVED MEMORIES

by

B. Ramakrishna Rau

May 1977

ABSTRACT

One of the major factors influencing the performance of an interleaved memory system is the behavior of the request sequence, but this is normally ignored. This report examines this issue. Using trace driven simulations it is shown that the commonly used assumption, that all requests are equally likely to be to any module, is not valid. The duality of memory interference with paging is noted and this suggests the use of the Least-Recently-Used Stack Model to model program behavior. Simulation shows that this model is quite successful. An accurate expression for the bandwidth is derived based upon this model.

The work described herein was supported by the U.S. Energy Research and Development Administration under contract No. EY-76-S-03-0326-PA 39. Computer time was made available by the Stanford Linear Accelerator Center.

1. Introduction

Overlapped, pipelined processors and multiprocessor systems have the capability to make multiple requests simultaneously. The use of a monolithic memory, which can process only one request at a time, would result in serious performance degradation, especially if the memory cycle time were large in comparison to the processor cycle time. A solution commonly employed is to partition the memory into a number of modules, each of which is capable of processing a request. Low-order interleaving is the most frequently used strategy, wherein all addresses with the same low-order bits are placed in the same module. Such an organization offers the potential for increased bandwidth by virtue of the parallelism present. In practice, however, not all of this parallelism is observed. The modules do not constitute a set of "identical servers" since each one contains a distinct set of memory locations. Accordingly, the situation can arise where all the requests are concentrated on a subset of the modules, causing these modules to be overworked while the remaining modules lie idle. The frequency and severity of such an occurrence will depend upon the properties of the program generating the requests. Our purpose is to investigate this aspect of program behavior and its impact upon the performance of an interleaved memory.

A complete analysis of the performance of an interleaved memory system must be based on a model which incorporates, at least, the following three factors:

1. the processor structure,
2. the memory structure, and
3. program behavior.

A description of the processor structure would include details of the number of processors and for each processor the speed of the processor, the rate at which it is capable of issuing requests and the number of pending requests that it is capable of buffering. The second factor accounts for the number of modules in the memory, the cycle time of each module and the width of each module (i.e., the number of bits made available on each module access). The third factor is the primary focus of interest here and once again includes at least three important points. Firstly, we must consider the timing behavior of the requests, i.e., the instants at which requests are made. This is influenced by the fraction of instructions which require operands and by whether certain instructions require multiple operands. A second aspect of program behavior is the order in which the modules are referenced, which we shall term the sequencing behavior. Lastly, one must consider the logical dependencies that exist between memory requests which might prevent one request being generated if a previous one is pending.

Most of the literature on interleaved memories has modelled the memory structure in a reasonable manner, The approach to the modelling of the processor structure partitions the literature into two groups. The work by Skinner and Asher [SKIN69], Strecker [STRE70], Bhandarkar [BHAN73], Baskett and Smith [BASK76], and Rau [RAU76], considers multiprocessor structures in which each processor is permitted to have a maximum of one outstanding request. Such a model describes, more or less accurately, the situation that exists in a multi-miniprocessor system such as C.mmp [WULF72]. The remaining work in the field is devoted to the overlapped, uniprocessor which is capable of having more than one outstanding request. The assumption here, generally, is that the processor is capable of having a limitless number of requests outstanding. The work exemplifying this line of thought is by Flores [FLOR64], Hellerman [HELL73], and Burnett and Coffman [BURN75]. In reality, the number of requests that a processor is capable of is finite and greater than one. This number is limited both by the logical dependencies that exist between requests (e.g., we cannot generate an operand request before the corresponding base register has been loaded) and by the amount of buffering and the degree of overlap in the processor.

2. Program Behavior.

An examination of the literature reveals very few attempts at modelling the behavior of the request stream. Chang, [CHAN75], recognizes the existence of dependencies between requests. But he does not make any attempt to model this. Rather, he uses the standard assumption of independent, equi-probable requests to analyze several different processor-memory configurations. Flores assumes that the instants at which requests are made constitute a Poisson process. The alternative seems to be to assume that a request is made every cycle unless the processor is blocked. This is the assumption used by Hellerman and by Burnett and Coffman. It shall be used by us as well. The results obtained may be interpreted, therefore, as constituting an upper bound to the bandwidth that would have actually been observed. Both the dependency structure and the timing behavior are irrelevant in models where the processor can make only one request at a time.

Almost equally neglected is the sequencing behavior of the request stream. By and large, each request is arbitrarily assumed to be equally likely to reference any module. This we shall term the Random Independent Reference Model (RIRM). The only exception to this is the work by Burnett and Coffman where the instruction and data request streams are separately characterized by their probability of proceeding sequentially. A non-sequential request is considered to be, with equal probability, for any of the remaining modules. They do not, unfortunately, validate their model against real reference strings.

Terman, [TERM76], has tested the validity of Burnett and Coffman's model of program behavior via trace driven simulations. As might be expected the model works well for the instruction stream which does, in fact, generate sequential runs of requests. But it performs very poorly in describing the data requests. Whereas, Terman's simulation does not correspond exactly to Burnett and Coffman's assumptions, there is reason to believe that the conclusions are valid and that their model is inadequate. We first note that sequentiality is not the primary issue by observing that the request sequences 12341234 and 13421342 perform equally well (where the numbers denote the module referenced). Furthermore, even if both the instruction stream and the data stream were extremely sequential, when they get merged as happens in real systems, the resulting stream would display little, if any, sequentiality. Of importance is how soon a module is re-referenced since this determines the delay that would be experienced in waiting for the module to free up. We need a model which can "remember" how long ago each module was referenced.

A Markov model is the most obvious candidate. To be fully effective it would have to be of an order at least as high as the cycle time of the module. So, if the cycle time of the module was T cycles, then the state of the model would have to be defined by the most recent T references. Such a model has two serious drawbacks. Firstly, the number of states is extremely large; if the degree of interleaving is M, then the number of states is M^T. This is the case if we are interested in modelling only the sequencing behavior. If we wish to model the timing of the requests too, we have $(M+1)$ possible events at each epoch -- M modules that can be referenced and the possibility of no request at all. This would increase the state space to $(M+1)^T$. The estimation of the parameters of this model and analysis using this model would be cumbersome. Secondly, the model would depend upon the cycle time of the modules. Ideally, a model of program behavior should be independent of the physical parameters of the memory.

The duality of the memory interference problem with the paging problem suggests a solution. In a paging situation, a reference to a page guarantees that the page will be retained for a certain period of time thereafter and so, if the next reference to the page occurs fairly soon, it will be a hit. In an interleaved memory, a reference to a module causes it to be unavailable for a certain length of time and if the next reference to it follows soon it will result in a delay. In either case, the clustering in time of references to the same page or module is the relevant property which is beneficial in one situation and detrimental in the other. Accordingly, it might be expected that the models of program behavior that have been found to be successful in the paging studies will be successful in the current context in modelling the sequencing behavior.

3. Modelling of Program Behavior.

Two models which have been found to be capable of capturing the clustering effect in paging studies are the Working Set Model (WSM) and the Least-Recently-Used Stack Model (LRUSM), [COFF73]. By default, the WSM assumes that the references to each page form a renewal process and that the renewal processes corresponding to the various pages are independent of one another. This leads to problems if we wish to generate a reference string using the WSM since, in fact, the renewal processes are not independent -- two modules cannot be referenced in the same cycle. The WSM requires that a large number of statistics be gathered. We need f(i), the probability of referencing a page after an interval i for i ranging from 1 to the maximum observed interval. The WSM characterizes the clustering behavior explicitly. However, in the memory interference context it is not as amenable to analysis as is the LRUSM. The LRUSM has the advantage that it requires fewer statistics and also that it permits the generation of a reference string unlike the WSM. This last property is of great value when we wish to estimate the accuracy of an analytical solution and wish to separate the error due to the analysis from the error inherent in the model.

Bearing this in mind, the LRUSM was selected as the model of program behavior to be used in our study of memory interference. The LRUSM statistics were gathered for degrees of interleaving ranging from 2 through 16 in the manner described by Mattson et al., [MATT70]. The degree of interleaving plays the same role in our measurements as does page size in Mattson's paper. The width of the modules was fixed at one doubleword (8 bytes).

The statistics were first used to test the RIRM assumption of randomness and independence between the successive requests made by the program. If this assumption were true, we should expect an equal probability of reference to each stack distance. In fact, it was found that an increased probability of reference existed for the modules at the top and at the bottom of the stack for all degrees of interleaving, thereby invalidating the RIRM (Table 1). The increased probability of referencing the top of the stack may be ascribed to consecutive references to the same word. For instance, the byte manipulating instructions in the 360 architecture cause the same word to be referenced repeatedly to access the individual bytes. The distribution of the intervals between consecutive references to the same module generally has a rather long tail. All the probability in this tail contributes to the probability of accessing the bottom of the stack causing it to be relatively high.

The next step in the investigation was to see how good the LRUSM was, both absolutely and relative to the RIRM. This was accomplished by constructing a simulator of the interleaved memory. The assumptions built into the simulator were:

1. The processor is capable of making one request every cycle and does so unless the current request is to a busy module. This model will provide us with an upper bound on the bandwidth that would actually be observed.
2. The conflicting request is held until the requested module is free at which point the processor continues to submit a request every cycle to the memory.

3. The cycle time of the memory is T cycles.
4. The module width is 8 bytes.
5. The bus is time division multiplexed between the modules, which operate asynchronously (as opposed to the models of Hellerman and Burnett and Coffman, where the modules operate in unison).
6. The dependency structure and the timing behavior of the request stream are neglected. Only the sequencing behavior is considered.

The simulator was driven by three types of request streams -- trace tapes, reference strings generated by the LRUSM and reference strings generated under the RIRM. This was repeated for each of five trace tapes and in each case 100,000 references were processed. The trace tapes used were:

043	-- Fortran execution
049	-- Cobol execution
050	-- Cobol compilation
051	-- Fortran compilation
052	-- Cobol Sort

Figs.105 display the results obtained. Each one is a plot of the reciprocal of the bandwidth as a function of the cycle time T. Each figure corresponds to a particular trace tape and permits comparison, for various degrees of interleaving, of the measurements obtained from the trace tape itself to the measurements obtained by generating reference strings using the LRUSM and the RIRM. The numbers on the righthand side indicate the degree of interleaving.

The plots show that, by and large, the LRUSM predicts the memory bandwidths quite accurately. It is least accurate for the trace tape 043, but even here it does much better than does the RIRM. Almost invariably, the LRUSM performs well, both absolutely as well as in comparison to the RIRM. We may conclude that the LRUSM is an adequate model in this context and that it captures most of the sequencing structure in the reference string that is relevant.

In general, the RIRM underestimates the bandwidth for low degrees of interleaving and overestimates it for high degrees of interleaving. It is conjectured that this is due to the looping structure of programs. When the degree of interleaving is less than the loop length, the modules will be referenced in an apparently sequential fashion, resulting in more bandwidth than if the requests were random. For high degrees of interleaving, the existence of a loop results in a subset of the modules being referenced repeatedly. In such a situation, the observed bandwidth is lower than would be obtained with a random reference pattern.

4. Analysis of the LRUSM

Our next objective is to develop an analytic method for predicting the bandwidth so as to avoid having to simulate the LRUSM. In this section we shall derive an expression for the bandwidth based on the LRUSM and determine its accuracy. It turns out that the LRUSM lends itself very conveniently to analysis. We shall find it helpful to make use of the notion of a packet. The reference string can be divided into packets, which are defined as the maximally long sub-strings such that no sub-string contains two references to the same module. Thus a packet is determined by starting with a particular reference and scanning the reference string until a reference is found which is to the same module as a previously scanned reference. All the scanned references, excluding the duplicate reference, are included in one packet. The duplicate reference terminates the current packet and initiates the next one. Fig.6 illustrates this procedure.

The significance of defining a packet in this manner is obvious if we let the memory cycle time, T, assume an extremely large value. We then find that the processor makes a burst of references until a memory conflict occurs. At this point the processor stops issuing requests for an interval of approximately T cycles, after which the modules rapidly become available. The processor then issues another burst of requests. Each burst of references corresponds exactly to a packet. If T is reduced by one cycle, the interval between every pair of adjacent packets is reduced by exactly one cycle too. The total time needed to complete a large number of requests, and thus, the average time per request, are linear functions of T for large values of T (Figs.1-5). The bandwidth expressed as the average number of requests per cycle is proportional to the reciprocal of T. As T approaches 1, however, the packets begin to run into one another. In other words, the bandwidth is limited by the rate at which the processor can issue requests, which is one per cycle.

Having discussed qualitatively the manner in which the bandwidth is affected by the memory cycle time, T, we can now proceed to derive an expression for the bandwidth. This is given by the average length of a packet divided by the average duration of a packet. The duration of a packet is defined as the interval between the start of that packet to the start of the next packet. In addition to T, this is affected by the position in the packet of the critical request, i.e., the request with which the first request in the next packet conflicts. This is illustrated in Fig.7.

For a degree of interleaving M, the LRUSM is defined by a set of M probabilities $\{p(1),p(2),...,p(M)\}$ where $p(i)$ is the probability of referencing the module at depth i in the LRU stack. Define $h(k) = p(1)+...+p(k)$ and $m(k)=1-h(k)$. In a paging environment these would be termed the hit ratio and miss ratio respectively. The average packet length is given by

$$\sum_{i=1}^{M} i.\text{Prob}[\text{packet length} = i].$$

In general,

$$\text{Prob[packet length} = i] = h(i)\cdot\prod_{j=1}^{i-1} m(j) \qquad 1 \le i \le M.$$

We shall derive this for the case $i=3$. The general derivation is obvious thereafter. A packet of length 3 requires three references to distinct modules followed by a reference to one of these three modules. The first reference of the packet may be to any module and, so, occurs with probability 1. The second reference may not be to the same module, which is now at the top of the LRU stack. The probability of the second reference being distinct from the first one is m(1). The third reference which may not be to either of the modules in the top two positions of the stack occurs with probability m(2). Finally, the fourth reference (which is not part of the packet) must be to one of the modules in the top three positions of the stack. The probability of this is h(3). Therefore, the probability of a packet of length 3 is $m(1).m(2).h(3)$. This reasoning may be extended for any i, $1 \le i \le M$.

The duration of a packet is affected by the position of the critical request. Let the r-th request in the packet be the critical one. Then for very large T, the duration is given by T+r-1 (Fig.8a). For very small T, the duration might be the packet length i (Fig.8b). In general, the duration is given by max(i,T+r-1). Translating this in terms of the LRU stack, we note that the r-th reference in a packet of length i will, at the start of the next packet, be at depth d=i-r+1 in the LRU stack. Therefore, r=i-d+1 and the duration of a packet is given by max(i,T+i-d). Given that the packet length is i, the average duration is

$$\sum_{j=1}^{i} P[d=j|\text{packet length} = i].\max(i,T+i-j)$$

$$= \sum_{j=1}^{i} \max(i,T+i-j).p(j)/h(i)$$

Therefore, the average unconditional duration is

$$\sum_{i=1}^{M} h(i). \sum_{k=1}^{i-1} m(k) \sum_{j=1}^{i} \max(i,T+i-j).p(j)/h(i)$$

$$= \sum_{i=1}^{M} \sum_{k=1}^{i-1} m(k) \sum_{j=1}^{i} p(j).\max(i,T+i-j)$$

Therefore, the bandwidth is given by

$$\frac{\displaystyle\sum_{i=1}^{M} ih(i) \sum_{k=1}^{i-1} m(k)}{\displaystyle\sum_{i=1}^{M} \sum_{k=1}^{i-1} m(k) \sum_{j=1}^{i} p(j).\max(i,T+i-j)}$$

Our analysis makes one assumption, viz., that once the first request in the packet has been made, the remaining requests can be made in consecutive cycles. The example in Fig.9 shows that this is not necessarily true. We have neglected this effect in our analysis, thereby introducing some error in our result.

To test the accuracy of the expression for bandwidth, the bandwidths predicted analytically from the LRUSM statistics were compared with the measurements obtained by driving the simulator with the reference string generated by the LRUSM. The maximum discrepancy was about 1.3%, but, by and large, the error was well below 1%. The expression itself is very accurate and any error present when compared to the trace driven result is inherent in the LRUSM and does not lie in the analysis.

5. Conclusion

We have considered the case of a high speed processor generating requests to an interleaved memory. Our primary interest was in studying the effect of program behavior upon the memory bandwidth. It was shown that the commonly used assumption that requests have an independent and equal probability of being to any module was invalid and could result in fairly large errors, We found that the LRUSM is a reasonably accurate model of the sequencing behavior of reference strings, and, based upon this model, we developed an accurate analytic expression for the memory bandwidth.

However, it must be noted that the LRUSM might well be totally inadequate in a different environment. In particular, if we consider an interleaved memory structure with a finite sized queueing space per module, we find that the LRUSM grossly overestimates the bandwidth that would be observed (Fig.10). One possible explanation for this lies in the inability of the LRUSM to accurately reflect the variance of the time between consecutive references to the same module. Table 2 lists the measured values of the variance for the trace tape 052 against the values obtained theoretically from the LRUSM for various degrees of interleaving. (The theoretical expression for the variance and its derivation are to be found in [RAU77]). In general, the LRUSM significantly underestimates the variance, which in turn would lead to overestimating the bandwidth. When queues are allowed to build up, references which are widely separated can interfere with one another. A model which hopes to capture this behavior would need to have a very long "memory". The LRUSM has an inaccurate memory for very long inter-reference intervals. However, in the case that we studied, where queueing on the modules is not permitted, the LRUSM is quite satisfactory.

Acknowledgements

The author wishes to thank the Stanford Linear Accelerator Center for making available their computing resources. The plotting of the graphs was greatly facilitated by the graphics packages developed by Bob Beach and Roger Chaffee.

References

BASK72 F.Baskett and A.J.Smith, "Interference in Multiprocessor Computer Systems with Interleaved Memory", CACM 19, 6, June 1976, 327-334.

BHAN73 D.P.Bhandarkar, "Analytic Models for Memory Interference in Multiprocessor Computer Systems", Ph.D. Thesis, Carnegie-Mellon U., Pittsburgh, PA., 1973.

BURN75 G.J.Burnett and E.G.Coffman, "Analysis of Interleaved Memory Systems using Blockage Buffers', CACM 18, 2, Feb. 1975, 91-95

CHAN75 D.Y.Chang, "Analysis and Design of Interleaved Memory Systems," Report No. UIUCDCS-R-75-747, Dept. of Computer Science, Univ. of Illinois, Aug. 1975.

COFF73 E.G.Coffman and P.J.Denning, "Operating System Theory", Prentice-Hall, 1973, 241-312.

FLOR64 I.Flores, "Derivation of a Waiting Time Factor for a Multiple-Bank Memory", JACM 11, 3, July 1964, 265-282.

HELL67 H.Hellerman, "Digital System Principles", McGraw-Hill, 1967, 228-229.

MATT70 R.L.Mattson, J.Gecsei, D.R.Slutz and I.L.Traiger, "Evaluation Techniques for Storage Hierarchies", IBM Sys. Jour., 2, 1970.

RAU76 B.R.Rau, "An 'Almost-Exact' Solution for the N-Processor, M-Memory Bandwidth Problem", Report No. 117, Digital Systems Lab., Stanford Univ., June 1976.

RAU77 B.R.Rau, "Properties and Applications of the Least-Recently-Used Stack Model of Program Behavior", Report No.139, Digital Systems Lab., Stanford Univ., May 1977.

SKIN69 C.Skinner and J.Asher, "Effect of Storage Contention on System Performance", IBM Sys. Jour., 8, 4, 1969, 319-333.

STRE70 W.D.Strecker, "Analysis of the Instruction Execution Rate in certain Computer Structures", Ph.D. Thesis, Carnegie-Mellon U., Pittsburgh, PA., 1970.

TERM76 F.W.Terman, "A Study of Interleaved Memory Systems by Trace Driven Simulation", Fourth Symposium on the Simulation of Computer Systems, Aug. 1976.

WULF72 W.Wulff and C.G.Bell, "C.mmp, a Multi-mini-processor", AFIPS Conf. Proc., Vol. 41, part II, 1972 FJCC, AFIPS Press, Montvale, NJ., 765-777.

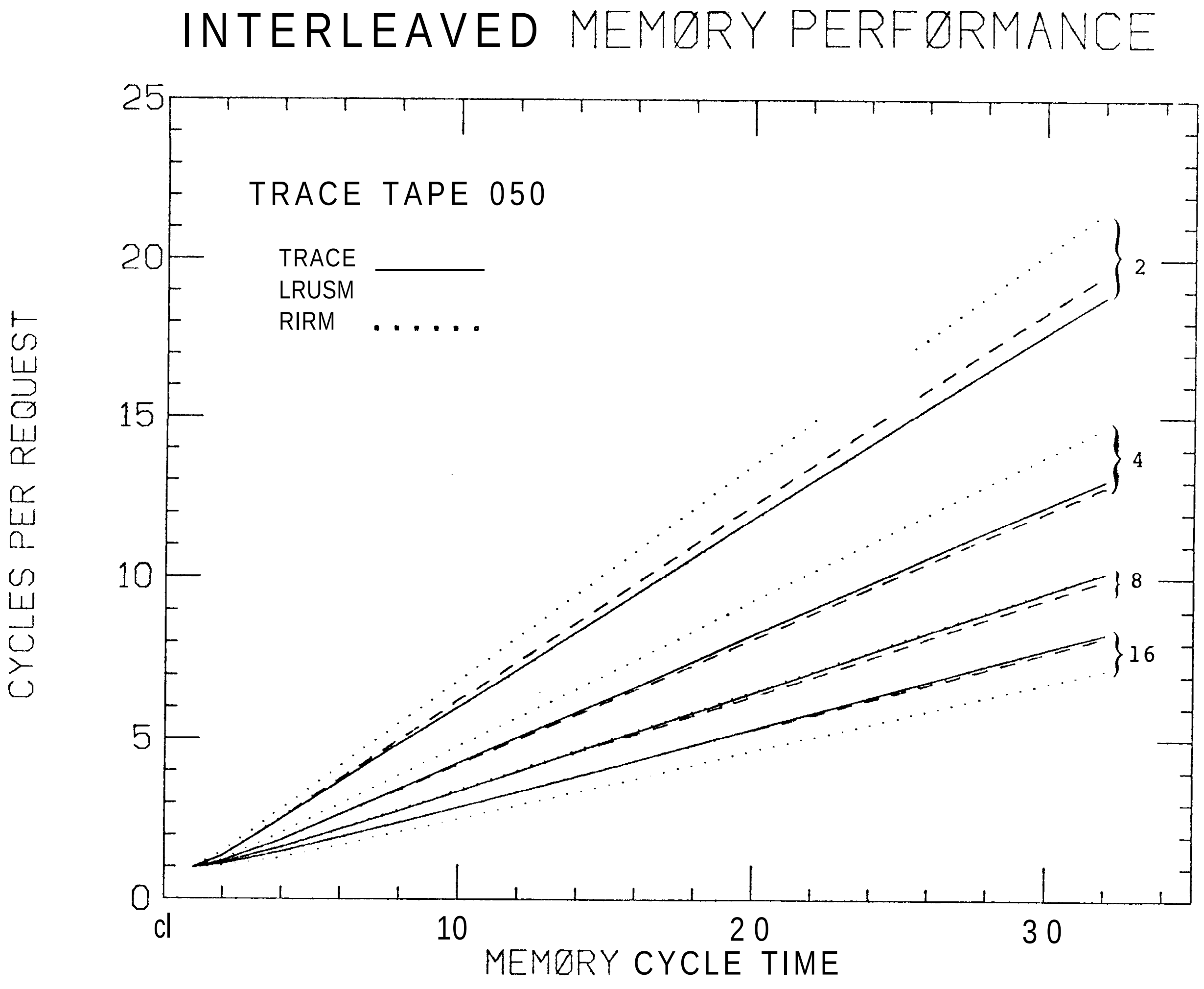
INTERLEAVED MEMØRY PERFØRMANCE
TRACE TAPE 050
TRACE
LRUSM
RIRM
CYCLES PER REQUEST
MEMØRY CYCLE TIME
0
5
10
15
20
25
cl
10
20
30
2
4
8
16

FIG.1

INTERLEAVED MEMØRY PERFØRMANCE

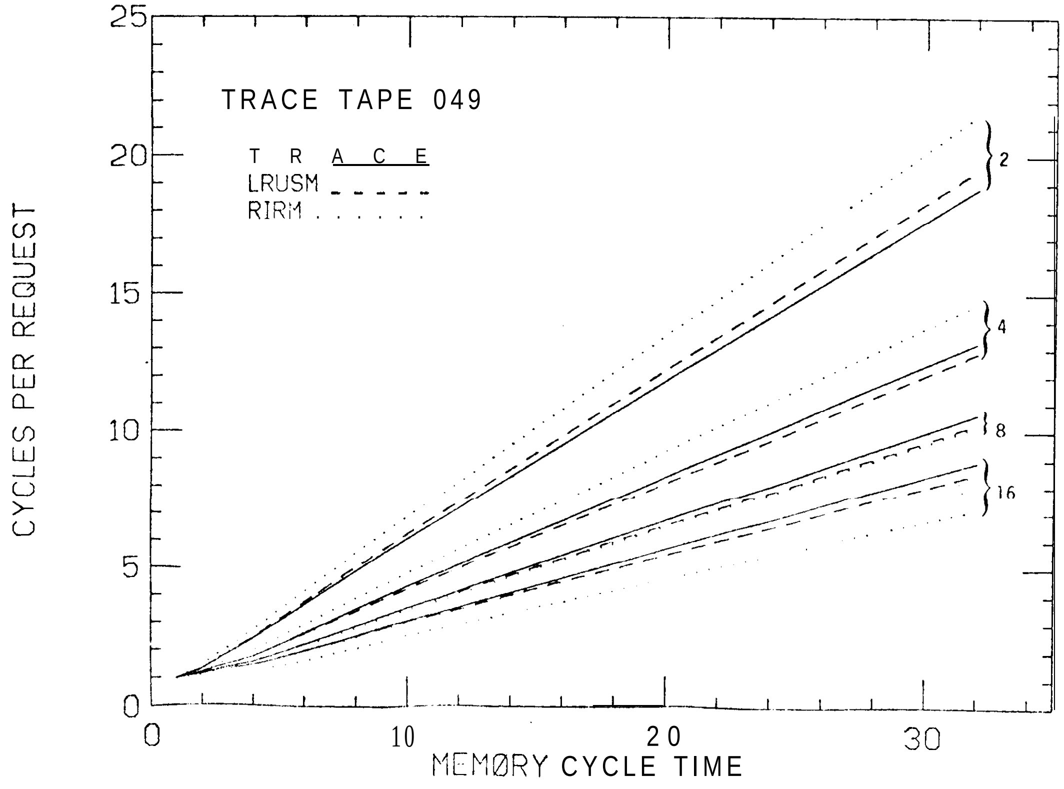

FIG.2

INTERLEAVED MEMØRY PERFØRMANCE

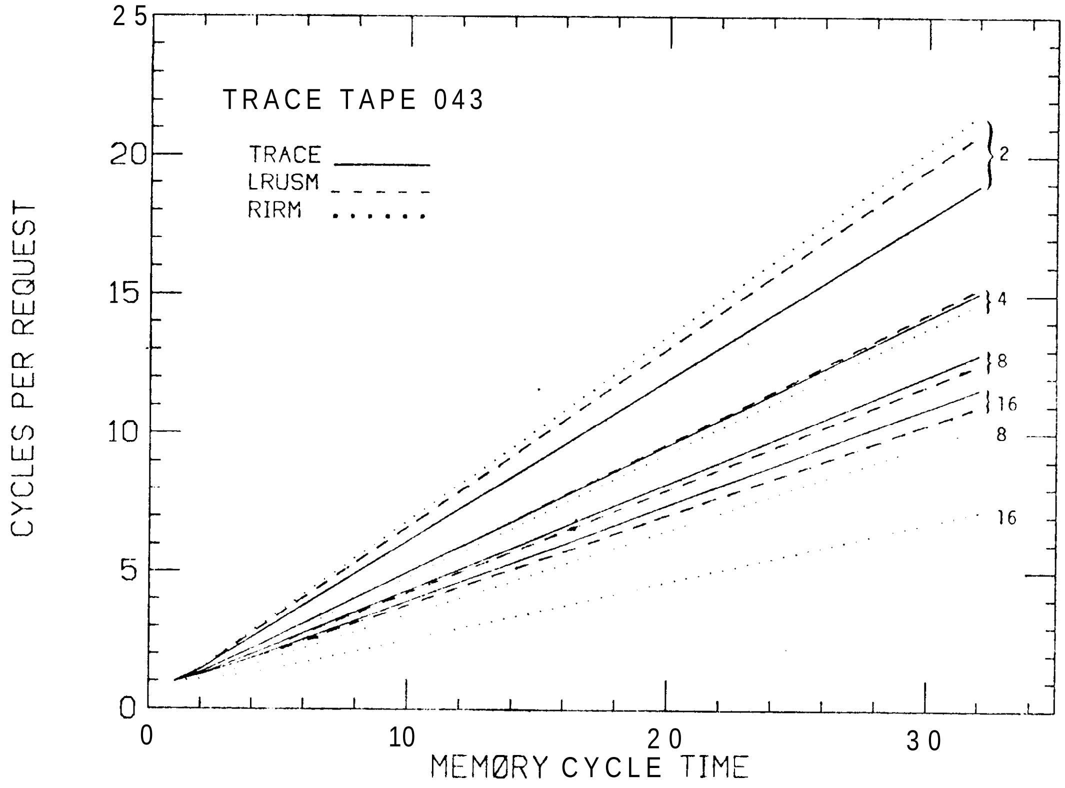

FIG.3

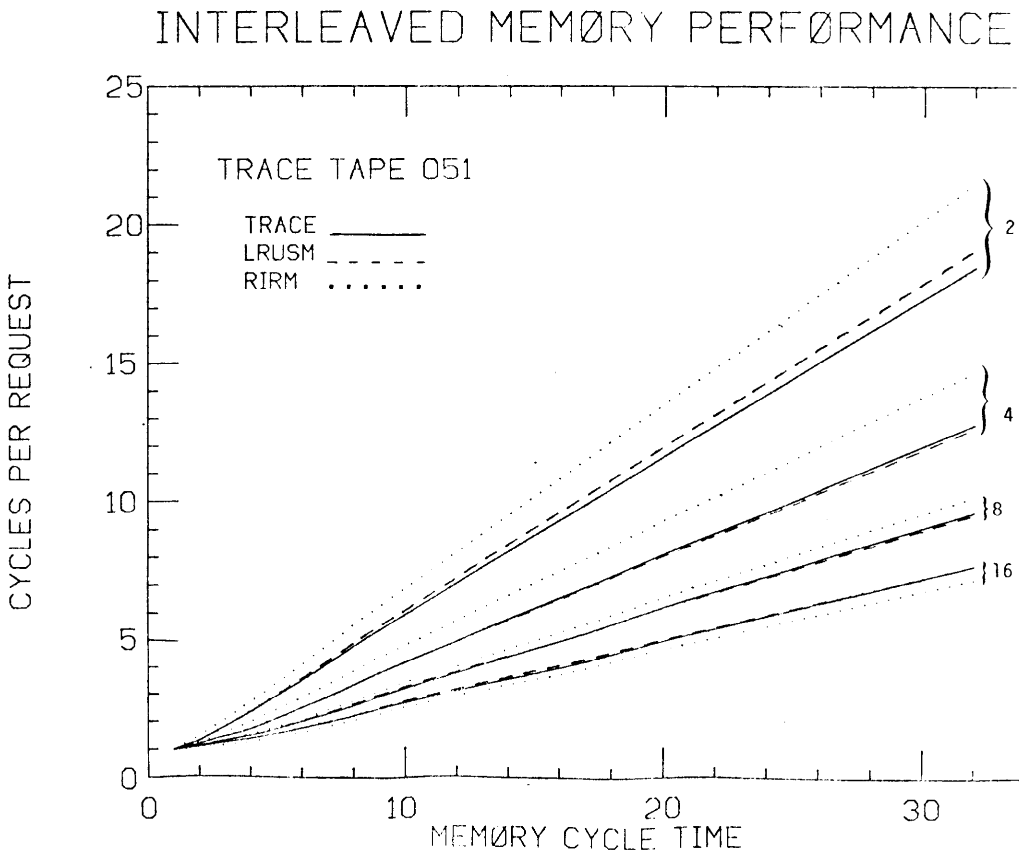
INTERLEAVED MEMØRY PERFØRMANCE
TRACE TAPE 051
TRACE
LRUSM
RIRM
2
4
8
16
CYCLES PER REQUEST
MEMØRY CYCLE TIME
0
5
10
15
20
25
0
10
20
30

FIG. 4

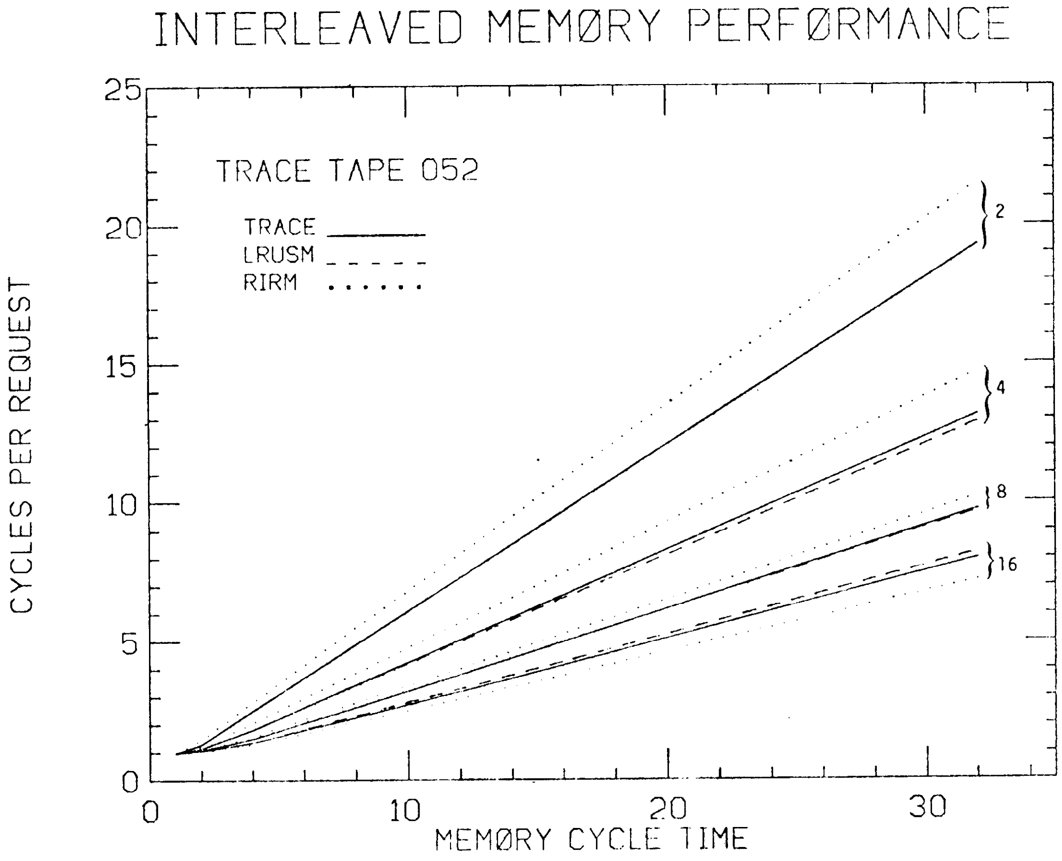
INTERLEAVED MEMØRY PERFØRMANCE
TRACE TAPE 052
TRACE
LRUSM
RIRM
2
4
8
16
CYCLES PER REQUEST
MEMØRY CYCLE TIME
0
5
10
15
20
25
0
10
20
30

FIG. 5

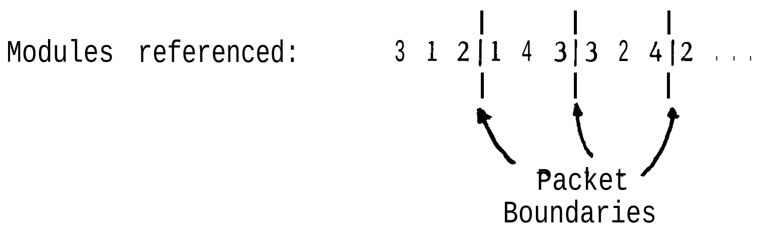

FIG.6 - Partitioning of the reference string into packets.

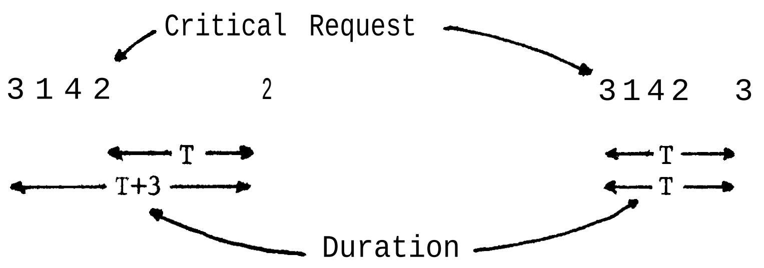

FIG.7 - Effect of the position of the critical request on duration

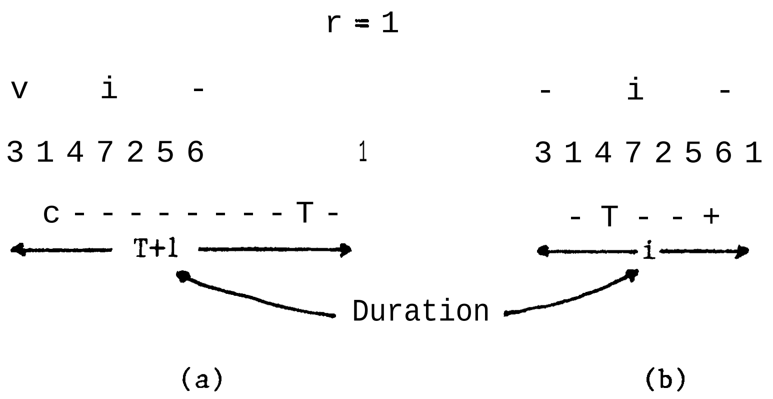

FIG.8 - Effect of the memory cycle time upon duration.

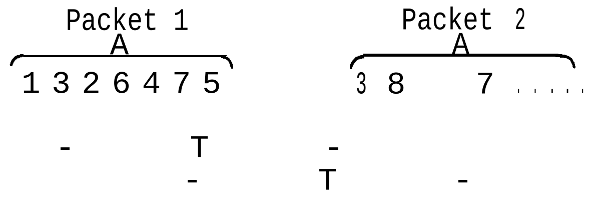

FIG.9 - Requests in the same packet may not be issued in consecutive cycles.

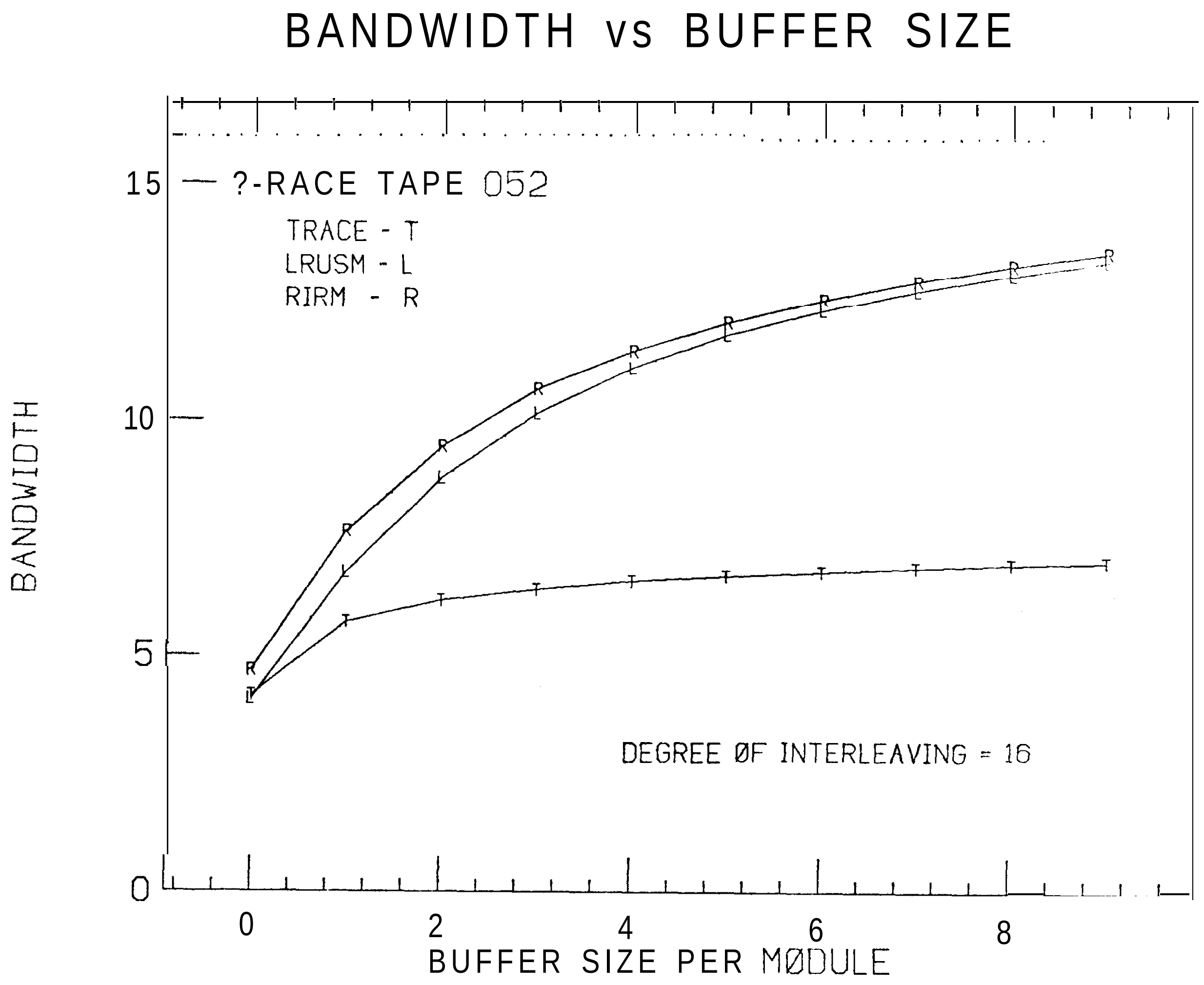
BANDWIDTH vs BUFFER SIZE
?-RACE TAPE 052
TRACE - T
LRUSM - L
RIRM - R
DEGREE ØF INTERLEAVING = 16
BANDWIDTH
BUFFER SIZE PER MØDULE
0
5
10
15
0
2
4
6
8

FIG.10

NUMBER OF MODULES M	TRACE	043	049	050	051	052
	STACK DEPTH	PROB	PROB	PROB	PROB	PROB
	d	p(d)	p(d)	p(d)	p(d)	p(d)
2	1	0.4489	0.3484	0.3508	0.3115	0.3229
	2	0.5511	0.6516	0.6492	0.6885	0.6771
4	1	0.3357	0.2218	0.2110	0.2079	0.1860
	2	0.1376	0.1236	0.1417	0.1296	0.1760
	3	0.1394	0.1397	0.1533	0.1439	0.2329
	4	0.3873	0.5149	0.4940	0.5186	0.4050
8	1	0.2760	0.1808	0.1534	0.1427	0.1079
	2	0.0901	0.0754	0.0871	0.0836	0.0879
	3	0.0662	0.0762	0.0985	0.0837	0.1862
	4	0.0666	0.0934	0.0807	0.0784	0.0834
	5	0.0680	0.0730	0.0890	0.0737	0.1022
	6	0.0592	0.0675	0.0575	0.0643	0.0881
	7	0.0538	0.0656	0.0751	0.0830	0.1106
	8	0.3201	0.3680	0.3587	0.3905	0.2335
16	1	0.2488	0.1513	0.1196	0.1167	0.0780
	2	0.0495	0.0476	0.0612	0.0531	0.0480
	3	0.0516	0.0371	0.0609	0.0422	0.1832
	4	0.0442	0.0600	0.0444	0.0406	0.0339
	5	0.0450	0.0373	0.0469	0.0371	0.0466
	6	0.0273	0.0378	0.0640	0.0346	0.0575
	7	0.0274	0.0312	0.0399	0.0298	0.0475
	8	0.0228	0.0431	0.0288	0.0401	0.0450
	9	0.0370	0.0362	0.0367	0.0452	0.0363
	10	0.0470	0.0331	0.0316	0.0426	0.0375
	11	0.0538	0.0478	0.0268	0.0491	0.0516
	12	0.0357	0.0568	0.0331	0.0421	0.0481
	13	0.0276	0.0564	0.0476	0.0574	0.0555
	14	0.0190	0.0283	0.0569	0.0376	0.0436
	15	0.0316	0.0404	0.0619	0.0541	0.0442
	16	0.2318	0.2557	0.2396	0.2776	0.1435

TABLE 1 - LRUSM Statistics

Trace Tape 052

Degree of Interleaving	Variance predicted by the LRUSM	Variance measured directly
2	0.9538	0.8684
4	6.5788	16.0911
8	39.8318	115.9912
16	224.3838	621.7017

TABLE 2 – Inaccuracy of the LRUSM in predicting the variance of the inter-reference interval.