remaining steps assume no type-1 or type-2 degeneracies are in the data set. The benign degeneracy signaled by Degen-type-3 set *true* can in fact be ignored and is dealt with below.

M2: (*Comment:* This step determines a set of (preliminary) illumination regions, and for each region determines its illuminated vertex set.)

begin $l \leftarrow 1$; $k \leftarrow 0$;
  while $\neg$ (LEFT$_k$) **do** $k \leftarrow k + 1$ **od**;
  $r \leftarrow A_k$;

(*Comment:* The point $x_k = p_r$ is the first left intercept in the list $x_0, x_1, \cdots, x_{2n-1}$.)

  for $\alpha \leftarrow 0$ **until** $2n - 1$ **step** 1 **do**
    begin $R_\alpha \leftarrow [x_\alpha, x_{\alpha+1}]$;

(*Comment:* R_α is an illumination region defined by endpoints x_α and $x_{\alpha+1}$ on bd K.)

      if $x_\alpha = x_{\alpha+1}$, **then** $V(R_\alpha) \leftarrow \{v_{l-1}, v_l, v_r, v_{r+1}\}$;
        else $V(R_\alpha) \leftarrow \{v_l, v_r\}$;

(*Comment:* $V(R_\alpha)$ is the illuminated vertex set associated with region R_α.)

      if LEFT$_{\alpha+1}$ **then** $r \leftarrow r + 1 \bmod n$;
        else $l \leftarrow l + 1 \bmod n$;
    end
end

(*Comment:* This step is essentially identical to step 4 of construction procedure 1 in Silio [1], and arguments justifying why this step correctly identifies and names the IR's can be found in [1]. This step takes time $O(n)$.)

M3: Merge the ordered list x_k and the vertices of K (as in step 5 of construction procedure 1 in [1]). Because a preliminary illumination region from step M2 may extend over more than one edge of K, break these regions into further subregions if they overlap with more than one edge of K, and associate with each subregion the illuminated vertex set of its parent region.

(*Comment:* This step takes time $O(n + m)$.)

end Procedure 1.

Application of the above algorithm to the illustration in Fig. 2 results in the illumination regions and corresponding illuminated vertex sets shown in the figure. Note that preliminary IR R_0 extends from x_0 to and including x_1, R_1 extends from x_1 to and including x_2, and so on with R_9 extending from x_9 to and including x_0. Note that R_3 is a single point region with Degen-type-3 set *true*. Sample parameters for step M1 are also shown in the figure, as space permits. Note that step M3 would break regions R_1, R_4, R_6, and R_8 into two subregions each, respectively. Each pair of subregions would inherit the IVS of its parent, and each would then be described as a single straight line segment in a single facet of K.

IV. Conclusion

We have presented an optimal algorithm for computing illumination regions and associated illuminated vertex sets for one convex polygon P contained in another convex polygon K. This $O(n + m)$ algorithm can be used to replace the less efficient construction procedure 1 in [1] for deciding simplex coverability of similarly situated convex polygons. After the set of illumination regions and illuminated vertex sets have been obtained using the optimal illumination region algorithm presented here, one can solve the simplex coverability problem by applying construction procedure 2 in Silio [1]. Note that line 3 in step 2 of construction procedure 2 in [1] contains a typographical error and incorrectly reads: "**while** NEXT1 $\neq$ *false* and NEXT2 $\neq$ *false*, **do**." This line should be corrected to read "**while** NEXT1 $\neq$ *false* or NEXT2 $\neq$ *false*, **do**"; thereby replacing the "and" with "or." With this correction in place construction procedure 2 in [1] proceeds in linear time to produce a collection of what are called coverable candidate sets of vertices of P with associated sets of illumination region combinations to be used as constraint sets. To determine the existence of a covering triangle, one solves systems of second order equations, also in linear time, subject to the finite list of constraint sets provided by construction procedure 2 in [1]. Hence by using the optimal illumination region algorithm presented here, coupled with construction procedure 2 in [1], an optimal simplex coverability algorithm for convex polygons results.

When applying these optimal algorithms to stochastic sequential machine covering problems, the input vertices may be randomly related sets of points; in which case, one must first apply an appropriate convex hull algorithm (e.g., [9], [10]) to find the hulls in time $O(n \log n + m \log m)$. One can then apply the $O(n + m)$ procedure described here first to compute and name illumination regions and then apply the $O(n + m)$ algorithm from [1] that uses the illumination regions to generate and solve constrained simultaneous equations to determine simplex coverability.

Acknowledgment

The authors are grateful to the referees for comments and suggestions on a prior version of the material presented here.

References

[1] C. B. Silio, Jr., "An efficient simplex coverability algorithm in E^2 with application to stochastic sequential machines," *IEEE Trans. Comput.*, vol. C-28, pp. 109–120, Feb. 1979.
[2] A. Aho, J. Hopcroft, and J. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley, 1974.
[3] M. I. Shamos and D. Hoey, "Closest-point problems," in *Proc. IEEE 16th Symp. Foundations Comput. Sci.*, 1975, pp. 151–162.
[4] G. Ott, "Reconsider the state minimization problem for stochastic finite state systems," in *1966 IEEE Conf. Rec., 7th Annu. Symp. Switching Automata Theory*, 1966, pp. 267–273.
[5] A. Paz, *Introduction to Probabilistic Automata*. New York: Academic, 1971.
[6] M. I. Shamos, "Computational geometry," Ph.D. dissertation, Dep. Comput. Sci., Yale Univ., New Haven, CT, 1977.
[7] D. T. Lee and F. P. Preparata, "The all nearest neighbor problem for convex polygons," *Inform. Proc. Lett.*, vol. 7, pp. 189–192, June 1978.
[8] ——, "An optimal algorithm for finding the kernel of a polygon," *J. Ass. Comput. Mach.*, vol. 26, pp. 415–421, July 1979.
[9] F. P. Preparata and S. J. Hong, "Convex hulls of finite sets of points in two and three dimensions," *Commun. Ass. Comput. Mach.*, vol. 20, pp. 87–93, Feb. 1977.
[10] R. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Inform. Proc. Lett.*, vol. 1, pp. 132–133, June 1972.

Bandwidth of Crossbar and Multiple-Bus Connections for Multiprocessors

TOMÁS LANG, MATEO VALERO, AND IGNACIO ALEGRE

***Abstract*—In this paper we compare the effective bandwidth in a multiprocessor with shared memory using as interconnection networks the crossbar**

Manuscript received August 4, 1981; revised December 31, 1981 and April 19, 1982.

T. Lang is with the Department of Computer Science, University of California, Los Angeles, CA 90024.
M. Valero is with the Facultat d'Informatica, Universitat Politecnica de Barcelona, Barcelona, Spain.
I. Alegre is with the Escola Tecnica Superior de Telecomunicacio, Universitat Politecnica de Barcelona, Barcelona, Spain.

0018-9340/82/1200-1227$00.75

or the multiple-bus. We consider a system with N processors and N memory modules, in which the processor requests to the memory modules are independent and uniformly distributed random variables. We consider two cases: in the first the processor makes another request immediately after a memory-service, and in the second there is some internal processing time.

The results of simulations show that the multiple-bus interconnection network with a number of buses slightly higher than $N/2$ produces a very small degradation with respect to the crossbar.

In addition, we propose an organization with partial buses that is more economical than the multiple-bus for the same effective bandwidth.

Index Terms—**Bus arbitration, memory bandwidth, multiple buses, multiprocessors, shared memory.**

I. Introduction

In many multiprocessor systems the shared memory is divided into independent modules so that each of these modules permits one access per cycle. The processors are connected to the memories by means of an interconnection network (Fig. 1).

Two types of operation of the system have been proposed: SIMD, in which all processors execute the same instruction on different data, and MIMD in which each processor executed a different instruction [1]. In this paper we are interested in the MIMD case.

Several interconnection networks have been proposed for these systems, such as the crossbar [2], single-bus [3]-[5], multiple-bus [6], shuffle-exchange [7], and others [8], [9]. Of these, the crossbar provides the largest potential bandwidth because there are no conflicts in the network. Nevertheless, it has a high cost, which is prohibitive for a large number of processors [10], [11] and does not allow graceful degradation. Consequently, other interconnection networks become attractive provided they do not significantly degrade the system performance. In this paper, we study the additional degradation introduced by the multiple-bus network.

There are many performance measures that can be used to evaluate such a system. The one most related to the final objective of the system is the amount of processing work done per unit of time. Nevertheless, this measure depends on too many factors, such as the type of programs executed and the characteristics of the processors, therefore, make it difficult to isolate the influence of a particular system parameter on performance. More manageable measures that have been used are: the average number of instructions executed per time unit, the average number of memory modules accessed per time unit, the average processor waiting time, etc. All of these measures are related and the one that has been most favored in previous studies is the effective memory bandwidth. We also use this performance measure.

In general, the effective bandwidth of such a multiprocessor depends on the distribution of the requests to the memory modules, the service time of the memory, the propagation delay and conflicts in the interconnection network, and the time between two consecutive requests by a processor.

Several performance studies have been reported for different interconnection networks and different model assumptions [2]-[7]. In particular, for the crossbar there exists an ample set of references [12]-[16] which describe exact and approximate mathematical models and simulations to calculate the effective memory bandwidth for different assumptions with respect to the distribution of the processor requests, the distribution of the service-time of memory, and the distribution of the time between requests. In [15] Baskett and Smith present models for a system with interleaved memory modules and several hypotheses and also results of simulations with real programs. They conclude that adequate results are obtained assuming that the processors are synchronized, that the requests to the memory have a uniform distribution, that the cycle time of memory is constant, and that the processing time has a binomial distribution. For these hypotheses Rau [16] has a good summary of existing models. We also use these hypotheses.

For these hypotheses and for the case in which a processor issues a new request as soon as the previous one is serviced, Table I indicates the effective bandwidth for 2, 4, 8, and 16 processors and memory modules. For $N = 2$, 4 and 8 results are exact [13]. For $N = 16$ the value is obtained from the approximate model proposed in [15], equation (14). We conclude that with the indicated hypotheses the potential bandwidth of the crossbar is not fully utilized due to memory conflicts. As a consequence of the high cost of the crossbar for large N, its bad fault tolerance features and the degradation due to conflicts, it seems convenient to consider other alternative networks.

In this paper we study the performance of the multiple-bus interconnection network. The N processors are connected to the M memory modules by means of $B \leq \min(N,M)$ buses. For $B < \min(N,M)$ the network produces a degradation in the bandwidth with respect to the corssbar, due to conflicts that occur in the network when the number of requests to different memory modules is greater than B. We are interested in evaluating this degradation as a function of B and N.

To evaluate this degradation we could use the exact or approximate models proposed in [6]. Nevertheless, the hypotheses used there are restrictive (exponential request and service times), and they are difficult to evaluate for the cases in which the number of processors, buses, and/or memory modules is large. For this reason, we use simulations which are sufficient to validate our conclusions. The results obtained indicate that for $B \simeq N/2$ (for $N = M$), the degradation with respect to the crossbar is approximately 5 percent.

We also performed simulations for the case in which the processor does not issue a new request as soon as the previous one is serviced. This represents the case in which there is some internal processing. Following our hypothesis that the processors are synchronized, and using the validation performed in [15], we assume that in each cycle a processor issues a request with a fixed probability p. As will be seen from the simulation results, for $p = 0.5$, the crossbar is very underutilized and therefore for the same degradation (of 5 percent with respect to the crossbar) fewer buses than for $p = 1$ are required.

As is discussed in the following sections, the multiple-bus network is less expensive than the crossbar. Nevertheless, for large N and B, its cost is still important due to arbitration time and complexity, and to the capacitive loads and drive requirements. To reduce further the network cost, we propose a network with partial buses, discussed in Section IV, in which the arbitration is simplified and the drive requirements reduced. This network produces a lower bandwidth than the multiple-bus, for the same number of buses. The bandwidth is determined by simulation, and we conclude that there are configurations with partial buses which produce roughly the same bandwidth as the multiple bus at a lower cost.

II. Multiple-Bus Organization and Comparison with the Crossbar

As was indicated in the introduction, we are interested in evaluating the performance of the multiple-bus interconnection network. In this case, the N processors and M memory modules are connected through B buses. Each processor is connected to all buses and each bus to all memory modules, so that a processor can access any memory module through any of the buses (Fig. 2).

The number of connections of the multiple-bus network is proportional to $B(N + M)$. The number of wires is proportional to B and each of the buses supports a capacitive load proportional to $M + N + K(M + N - 1)$. The value of K is dependent on the technology, and for present day tristate circuits, is much smaller than one; therefore, the capacitive load is proportional to $M + N$.

The multiple-bus network requires an arbiter to assign the buses to the outstanding requests. To assign the buses to the memory modules, an M-users B-servers arbiter is needed. This arbiter selects $\min(B,J)$ of the J memory modules with at least one outstanding request. Once a bus is granted to access a memory module, only one of the processors that demanded that memory must be chosen. This choice is implemented by an N-users 1-server type arbiter, since there are N demand inputs (each associated with a processor), and only one

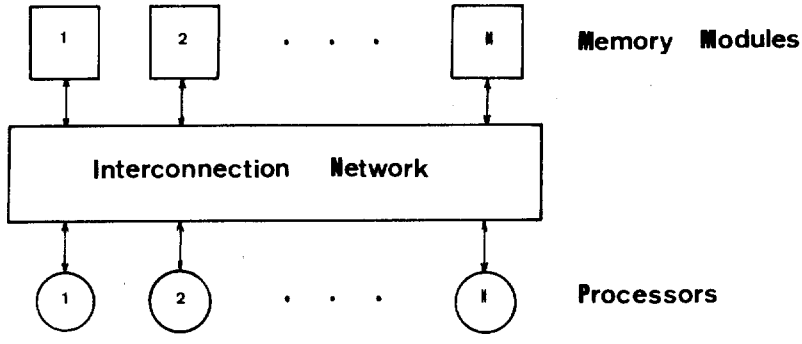

Fig. 1. Multiprocessors system.

TABLE I
EFFECTIVE BANDWIDTH FOR CROSSBAR ORGANIZATION

N	Eb
2	1.50
4	2.62
8	4.95
16	9.59

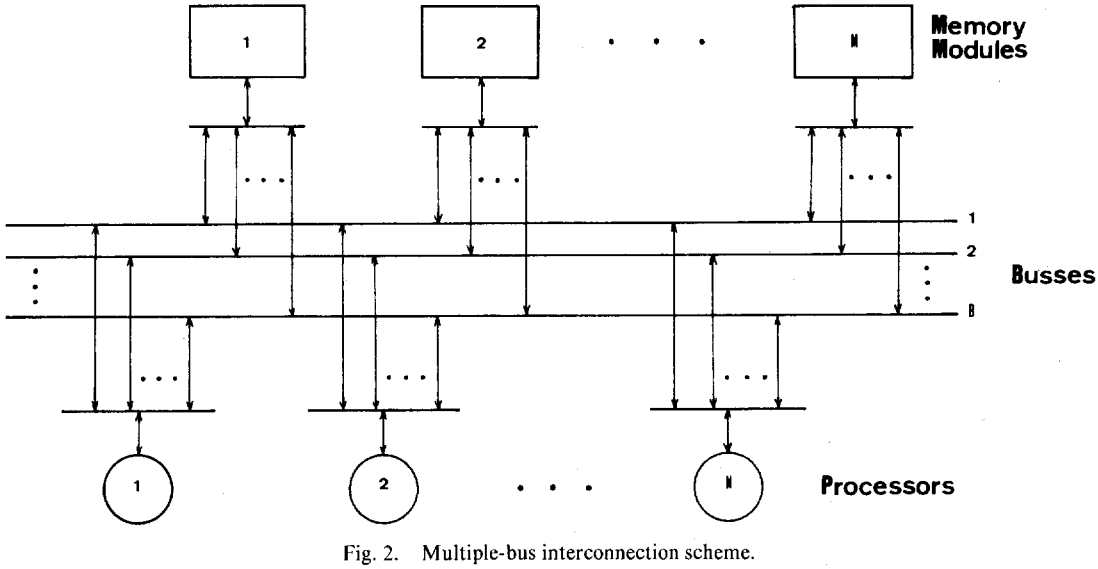

Fig. 2. Multiple-bus interconnection scheme.

can be granted. Therefore, the complete arbiter that the multiple bus network needs is built by one arbiter of the M-users B-servers type and M arbiters of the N-users 1-server type.

In [17] we presented the design of such an arbiter with a fair assignment policy. It assigns, in a cyclic fashion, the buses to memory modules that have at least one pending request and, for each module that obtains a bus, selects one among the processors that have a pending request for that module. We presented iterative designs with one and two levels of look-ahead, which produce arbiters that are modular and easy to integrate with present LSI technology. The arbitration time depends on the number of buses and memory modules and seems adequate, especially for a system with a not too large number of microprocessors. Several different designs of N-users 1-server type arbiters have already appeared in the literature [18]–[22].

The multiple-bus interconnection network is fault-tolerant because it can operate in a degraded mode after the failure of a subset of the buses.

The above mentioned characteristics are compared with those for the crossbar in Table II. The arbitration is simpler for the crossbar, since it only needs M N-users 1-server type arbiters, each of them controlling the access to a memory module. Also, the time such arbiter requires to decide which processors will access memory is less than the time the arbiter needs for the multiple-bus structure [17]. The crossbar is less fault-tolerant than the multiple-bus structure because a failure in one of the M buses disconnects completely one memory module.

III. SIMULATIONS

As mentioned in the introduction, we wish to determine the memory bandwidth for the multiple bus organization for different values of $N = M$ (in order to compare the results with the $N \times N$

TABLE II
COMPARISON OF CROSSBAR AND MULTIPLE-BUS

	Crossbar	Multiple-bus
Number of connections	N * M	B(N + M)
Load of busses	N	N + M
Number of wires	M	B
Arbiter	M of N-users 1-server type	1 of M-users, B-servers type plus M of N-users, 1-server type
Fault-tolerance and Expansion	poor	good

TABLE III
EFFECTIVE BANDWIDTH FOR MULTIPLE-BUS ORGANIZATION, WITH $p = 1$

Number of Busses	Number of Processors			
	4	8	12	16
1	1	1	1	1
2	1.97	2	2	2
3	2.55	2.99	3	3
4	2.62	3.93	4	4
5		4.62	4.99	5
6		4.90	5.93	6
7		4.94	6.68	6.98
8		4.95	7.12	7.92
9			7.27	8.72
10			7.28	9.27
11			7.30	9.53
12			7.30	9.61
13				9.63
14				9.63
15				9.63
16				9.63

crossbar) and a different number of buses ($1 \leq B \leq N$).

The hypotheses of the simulations are the following.

a) The processors are synchronized.

b) The processor requests are independent and uniformly distributed random variables.

c) The cycle time of all the memory modules is the same and constant.

d) A processor issues a new request in the next cycle after receiving memory service.

e) The propagation delays and arbitration times associated with the interconnection network are not included explicitly but may be thought of as forming part of the memory cycle.

f) In each cycle, the buses are assigned cyclically to the memory modules that have at least one outstanding request. For a module that receives a bus, a processor is selected at random from those with outstanding requests for that module.

With these hypotheses we performed a simulation using the technique of multiple independent repetitions. The relation defined by

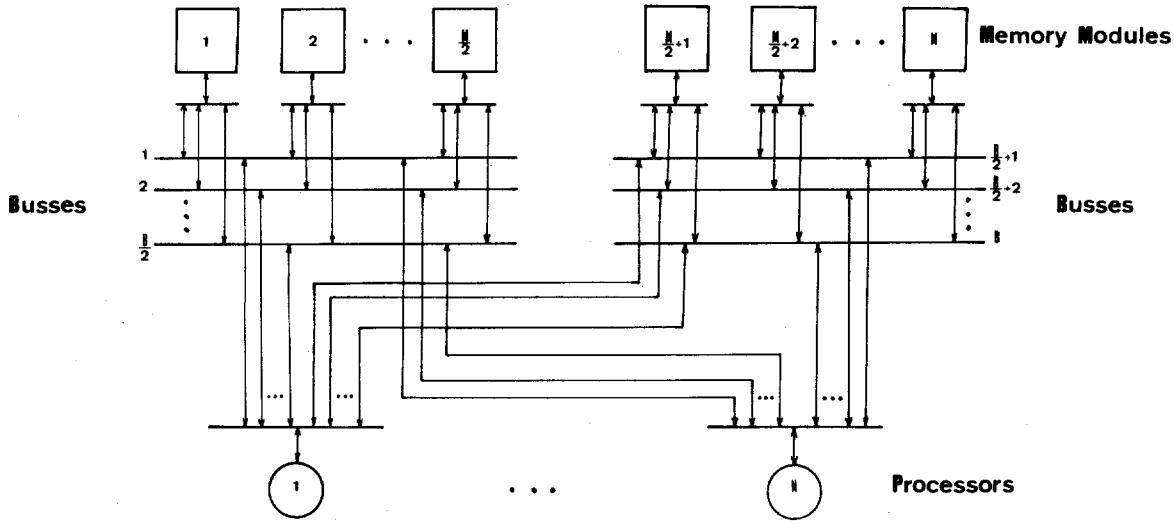

Fig. 3. Multiple-bus organization with partial buses and $g = 2$.

Lavenberg [23] was used to establish the number of simulation repetitions needed in terms of the defined confidence level. In our case, with a confidence level at 99 percent, we found that the number of repetitions needed was 5, obtaining as the amplitude of the confidence semiinterval the 6 percent of the mean. Each repetition of the simulation required a sample of 35 000 simulated cycles, of which the first 1000 were discarded as transient cycles.

To further validate these simulations, we compared them with known results. For $B = N$ (corresponding to the crossbar), the simulation results coincide with those determined using an exact mathematical model in [13] for $N = 4$ and 8, and with the value obtained using equation (8) in [16] for $N = 16$.

In view of the results of Table III we conclude that the multiple-bus organization with a number of buses $B \simeq N/2 + 1$ produces a bandwidth degradation of less than 5 percent with respect to the bandwidth obtained with the crossbar.

IV. Multiple Bus Organization with Partial Buses

The results obtained in the previous section show that the multiple-bus organization might be an attractive alternative to the crossbar. Nevertheless, the former structure still might be too costly for large N in some applications, due to the arbitration and drive requirements. This has led us to consider another network which is based on the multiple-bus, but has a lower cost.

This network consists also of B buses to connect the N processors to the M memory modules. Each of these buses is connected to all N processors, but only to a subset of M/g memory modules. That is, the memory modules are divided into g groups, and in each group, all memory modules are connected to the same B/g buses. Fig. 3 shows the case in which $g = 2$. In this type of network the number of connections is $B(N + M/g)$ and the load of each bus is proportional to $N + M/g$.

As can be seen, the number of connections and the loads are reduced with respect to the multiple-bus. Also, the partial-bus network requires g arbiters, but these arbiters are less complex and faster than those for the multiple-bus, because the arbitration time is a function of the number of buses and memory modules connected to each bus.

As was discussed in Section III, in the multiple-bus structure with B buses, the maximum possible bandwidth is not obtained when there are more than B memory modules with at least one request, because only B requests can be handled by the network. In the organization with partial buses there is an additional degradation due to the fact that only B/g buses are connected to each group of N/g memory modules. Therefore, if there are more than B/g memory modules with requests in a group, only B/g of them can be serviced.

We have performed simulations to evaluate this additional degradation using the same hypotheses employed in Section III. Some results for $g = 2$ and $N = 4$, 8, 12 and 16 are given in Table IV. The number of buses $(i + i)$ indicates that there are $2i$ buses divided into two groups of i buses each.

The comparison of the results of Tables III and IV shows the additional degradation produced by the network with partial buses with respect to the multiple-bus case.

Moreover, for the values N considered, the configuration with $(N/4 + 1, N/4 + 1)$ partial buses produces a higher bandwidth than the one produced by the multiple-bus structure with $N/2 + 1$ buses. As discussed before, the network cost and arbitration time are lower in the first case.

Of course, this concept of partial buses can be generalized to more than two groups, up to the case in which each group contains only one bus. In this latter case we would have an organization with B groups with partial single buses, which corresponds to the crossbar if $B = N$.

For a given number of buses B, as the number of groups is increased, the effective bandwidth decreases, but so does the cost and arbitration complexity. In a particular case, the selection of the number of buses and groups would depend upon the requirements of the system.

V. Case in Which the Processors Do Not Request Memory in All Cycles

In practical situations it might be possible that a processor does not request access to memory in every cycle. This could occur if the processor has a local memory and/or if some processor cycles are required for other than memory accesses. We have included this possibility in the model with the assumption that, in a given clock cycle, a processor requests access to memory with a probability $p < 1$.

In Table V we present simulation results for the multiple-bus network for $N = 4$, 8, 12 and 16 with $p = 0.5$ and $p = 1$ (for comparison). As might be expected, when $p = 0.5$ the bandwidth obtained for the crossbar (values for $B = N$ in Table V) is less than for $p = 1$, and therefore, the crossbar is even more underutilized. Moreover, the degradation of the multiple-bus with respect to the crossbar is smaller for $p = 0.5$ than for $p = 1$. For example, for the case $N = 16$ and $B = 8$, for $p = 1$, a degradation of 17.8 percent is obtained, while for $p = 0.5$ the degradation is only 2.1 percent. Therefore, the multiple-bus configuration is even more attractive for $p < 1$.

In Table VI we present simulation results obtained for the partial-bus organization. Again, it is evident that for $p < 1$ the degradation with $(N/2, N/2)$ buses is less than that for $p = 1$.

TABLE IV
EFFECTIVE BANDWIDTH FOR THE ORGANIZATION WITH PARTIAL BUSES AND $g = 2$

Number of Busses	Number of Processors			
	4	8	12	16
1 + 1	1.74	1.87	1.92	1.93
2 + 2	2.62	3.61	3.81	3.86
3 + 3		4.72	5.52	5.73
4 + 4		4.93	6.77	7.48
5 + 5			7.24	8.79
6 + 6			7.28	9.43
7 + 7				9.59
8 + 8				9.63

TABLE V
COMPARISON OF THE EFFECTIVE BANDWIDTH OF MULTIPLE-BUS WITH $p = 1$ AND $p = 0.5$

Number of Busses	Number of processors							
	4		8		12		16	
	p=1	p=.5	p=1	p=.5	p=1	p=.5	p=1	p=.5
1	1	1	1	1	1	1	1	1
2	1.97	1.65	2	2	2	2	2	2
3	2.55	1.77	3	2.87	3	3	3	3
4	2.62	1.77	3.93	3.33	4	3.95	4	4
5			4.62	3.45	4.99	4.67	5	4.98
6			4.90	3.47	5.93	5.03	6	5.85
7			4.94	3.47	6.68	5.13	6.98	6.43
8			4.95	3.47	7.12	5.16	7.92	6.70
9					7.27	5.16	8.72	6.82
10					7.28	5.16	9.27	6.83
11					7.30	5.16	9.53	6.83
12					7.30	5.16	9.61	6.83
13							9.63	6.84
14							9.63	6.84
15							9.63	6.84
16							9.63	6.84

TABLE VI
COMPARISON OF THE EFFECTIVE BANDWIDTH OF MULTIPLE-BUS WITH PARTIAL BUSES, WITH $p = 1$ AND $p = 0.5$

Number of Busses	Number of processors							
	4		8		12		16	
	p=1	p=.5	p=1	p=.5	p=1	p=.5	p=1	p=.5
1+1	1.74	1.50	1.87	1.83	1.92	1.90	1.93	1.93
2+2	2.62	1.77	3.61	3.11	3.81	3.64	3.86	3.79
3+3			4.72	3.44	5.52	4.76	5.73	5.43
4+4			4.93	3.47	6.77	5.10	7.48	6.42
5+5					7.24	5.16	8.79	6.77
6+6					7.28	5.16	9.43	6.83
7+7							9.59	6.84
8+8							9.63	6.85

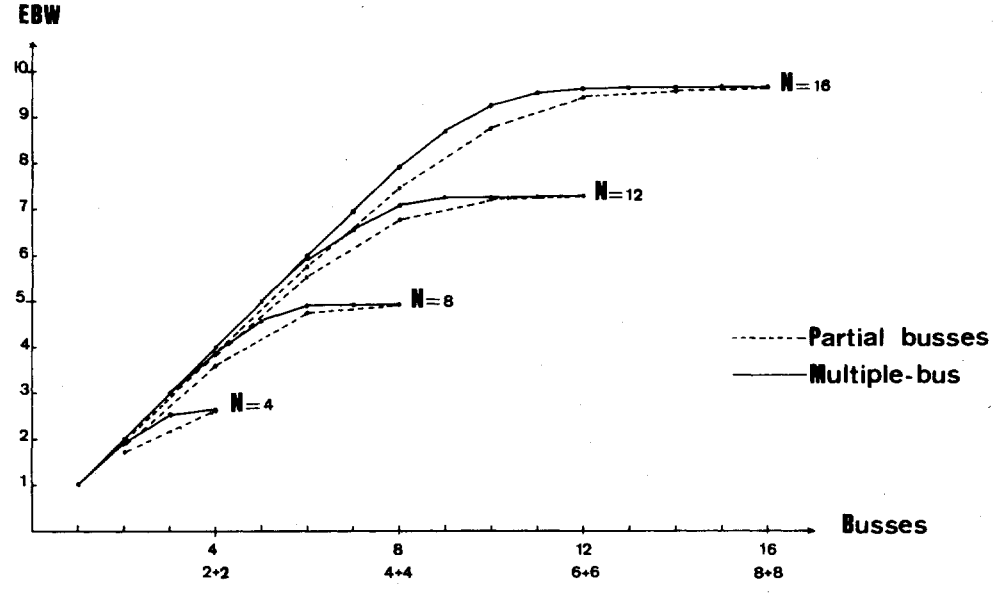

Fig. 4. Effective bandwidth with $p = 1$.

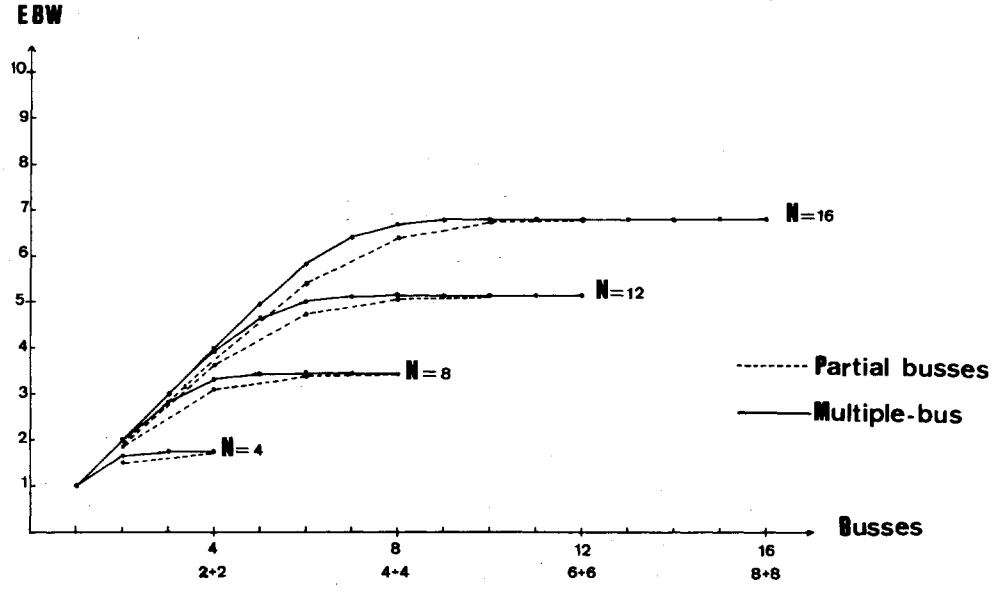

Fig. 5. Effective bandwidth with $p = 0.5$.

In Fig. 4 we show again the effective bandwidth values for $N = 4$, 8, 12 and 16 and $p = 1$, for the multiple-bus and multiple-bus with partial buses structures.

Similarly, Fig. 5 shows the values obtained when $p = 0.5$.

VI. Conclusions

In this paper we have compared the effective bandwidth of multiprocessors with shared memory using crossbar and multiple-bus interconnection networks. This work is motivated by the high cost and low fault tolerance of the crossbar.

We have assumed that the processors are synchronized, that the memory modules cycle time is constant, and that the processor requests are independent and uniformly distributed random variables. We conclude that with a number of buses slightly larger than $N/2$, the effective bandwidth of the multiple-bus organization is less than 5 percent smaller than that produced by the crossbar. In practical realizations that satisfy the model hypotheses, it would be better to use the multiple-bus structure because of its better cost and reliability characteristics.

For large N the cost of the multiple-bus structure can still be large, as a result of the number of connections, the loads and the complexity of the arbitration hardware. This finding has led us to propose the partial bus organization, which results in a similar bandwidth than the multiple-bus structure and at a lower cost.

Finally, we have simulated the case in which the processor requests memory with probability $p = 0.5$. This represents the case in which all clock cycles are not devoted to memory accesses. The results indicate that the crossbar is even more underutilized than for the case in which $p = 1$ and, therefore, that the multiple-bus and partial-bus organizations are even more attractive because the 5 percent performance degradation is obtained for a smaller number of buses.

References

[1] M. Flynn, "Some computer organizations and their effectiveness," *IEEE Trans. Comput.*, vol. C-21, Sept. 1972.

[2] W. A. Wulf and C. G. Bell, "C.mmp—A multi-mini-processor," in *Proc. Fall Joint Comput. Conf. AFIPS*, 1972, pp. 756–777.

[3] M. Ajmone *et al.*, "A study on processor-memory interconnection in multimicroprocessor system," *Alta Frequenza*, vol. L, no. 3, pp. 120–130, May–June 1981.

[4] S. Hoener and N. Roehder, "Efficiency of a multi-microprocessor system with time shared buses," *Euromicro*, pp. 35–42, 1977.

[5] R. J. Swan *et al.*, *CM*—A Modular Multimicroprocessor.* AFIPS, 1977, pp. 637–644.

[6] M. Ajmone and M. Gerla, "Markov models for multiple bus multipro-

cessor systems," U.C.L.A., Los Angeles, CA, Tech. Rep. CSD-810304, Feb. 1981.
[7] S. Thanawastien and V. P. Nelson, "Interference analysis of shuffle-exchange networks," *IEEE Trans. Comput.*, vol. C-30, pp. 545-556, Aug. 1981.
[8] H. Siegel *et al.*, "A survey of interconnection methods for reconfigurable parallel processing systems," in *Proc. NCC*, June 1979, pp. 529-542.
[9] S. Wu and M. T. Liu, "A cluster structure as an interconnection network for large multimicrocomputer systems," *IEEE Trans. Comput.*, vol. C-30, pp. 254-264, Apr. 1981.
[10] R. J. Swan, "The switching and addressing structure of an extensible multiprocessor: CM*," Ph.D. dissertation, Carnegie-Mellon Univ., Pittsburgh, PA, Aug. 1978.
[11] W. A. Wulf *et al. HYDRA/C.mmp. An Experimental Computer System.* New York: McGraw-Hill, 1981.
[12] W. D. Strecker, "Analysis of the instruction execution rate in certain computer structures," Ph.D. dissertation, Carnegie-Mellon Univ., Pittsburgh, PA, 1970.
[13] D. P. Bhandarkar "Analysis of memory interference in multiprocessors," *IEEE Trans. Comput.*, vol. C-24, pp. 897-908, Sept. 1975.
[14] C. M. Hoogendoorn, "A general model for memory interference in multiprocessors," *IEEE Trans. Comput.*, vol. C-26, pp. 998-1005, Oct. 1977.
[15] F. Baskett and A. Smith, "Interference in multiprocessor computer systems with interleaved memory," *Commun. Ass. Comput. Mach.*, vol. 19, pp. 327-334, June 1976.
[16] B. R. Rau, "Interleaved memory bandwidth in a model of a multiprocessor computer system," *IEEE Trans. Comput.*, vol. C-28, pp. 678-681, Sept. 1979.
[17] T. Lang and M. Valero, "*M*-users, *B*-servers arbiter for multiple-bus multiprocessor," *Microprocessing and Microprogramming, J. Euromicro*, to be published.
[18] W. Plummer, "Asynchronous arbiters," *IEEE Trans. Comput.*, vol. C-21, pp. 37-42, Jan. 1972.
[19] R. C. Pearce *et al.*, "Asynchronous arbiter module," *IEEE Trans. Comput.* pp. 931-932, Sept. 1975.
[20] K. Søe Højberg "One-step programmable arbiters for multiprocessors," *Comput. Design*, pp. 154-158, Apr. 1978.
[21] M. Courvoisier, "One-step *N*-user programmable arbiter," *Electron. Lett.*, vol. 15, pp. 430-432, July 1979.
[22] E. Petriu, "*N*-channel asynchronous arbiter resolves resource allocation conflicts," *Comput. Design*, pp. 126-132, Aug. 1980.
[23] S. Lavengerg *et al.*, "Sequential stopping rules for the regenerative method of simulation," *IBM J. Res. Develop.*, vol. 21, pp. 545-558, 1977.