BIT 33 (1993), 536–560.

A GENERAL PERFORMANCE ANALYSIS METHOD FOR UNIFORM MEMORY ARCHITECTURES

JONG-JENG CHEN, CHIAU-SHIN WANG and CHING-ROUNG CHOU

Institue of Computer Science and Information Engineering, National Chiao-Tung University, Hsin-Chu, Taiwan, R.O.C.

AT & T Bell Laboratories, IH6U-213 2000, N. Naperville Rd. Naperville, IL 60566, USA

Abstract.

The performance of a multiprocessor system greatly depends on the bandwidth of its memory architecture. In this paper, uniform memory architectures with various interconnection networks including crossbar, multiple-buses and generalized shuffle networks are studied. We propose a general method based on the Markov chain model by assuming that the blocked memory requests will be redistributed to the memory modules in the next memory cycle. This assumption results in an analysis with lower complexity where the number of states is linearly proportional to the number of processors. Moreover, it can provide excellent estimation on the system power and memory bandwidth for all three types of interconnection networks as compared with the simulation results in which the blocked memory requests are resubmitted to the same memory module. Comparisons also show that our method is more general and precise than most existing analysis methods. The method is further extended to estimate the performance of multiprocessor system with caches. The approximation results are also shown to be remarkably good.

CR Categories: D.4.1., I.2.8.

Keywords: multiprocessor system, interconnection networks, crossbar, multiple buses, generalized shuffle network, memory bandwidth, performance analysis, Markov chain.

1. Introduction.

With the advent of VLSI technologies, a great deal of attention has been paid to the design of multiprocessor systems to achieve higher computation power. However, the performance of a multiprocessor system greatly depends on the efficiency of its memory architecture. Memory architectures have been classified into two categories, *Nonuniform-Memory-Access* (NUMA) and *Uniform-Memory-Access* (UMA). As described in [1], NUMA systems are difficult to program because their performance is sensitive to the allocation of shared data structures and memory modules. In UMA systems, all shared memory is accessed through a common

Received November 1992. Revised April 1993.

interconnection network, so access time to any memory location is uniform across processors. This paper concentrates on the performance of UMA systems with various popular interconnection networks, such as crossbar, single bus, multiple buses, multistage interconnection networks and others.

Early multiprocessor systems were implemented using crossbars, which allowed conflict-free connections between processors and resources. The needs of decreasing hardware cost and increasing number of processors drive the trend of using simpler interconnection structures, such as shuffle/exchange networks and multiple-buses. Figure 1 shows a general model of multiprocessor system with UMA architecture, where the big rectangle box can be any type of those interconnections.

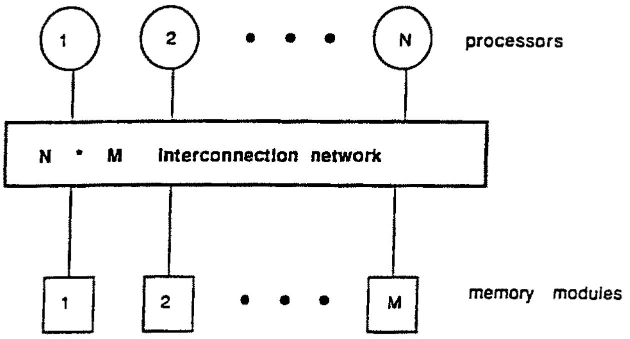

Fig. 1. A simple multiprocessor system.

On the other hand, modern multiprocessor systems often incorporate cache memory in each processor module in order to alleviate the data traffic through the interconnection networks. Such an architecture is conceptually depicted in Figure 2. The major drawback of the private cache is the data inconsistency problem, that is, the possibility of creating several copies of a single variable, where a copy is manipulated in a private memory independently of other copies, thus producing inconsistent values among those copies of the same variable. Such a problem can be solved through hardware or software interlocks and protocols. In the later analysis of UMA with cache memories, we shall assume an environment in which data consistency is not an essential problem and can be solved with implicit mechanisms embedded in the system.

In this paper, a general method which can be used for analyzing most of those interconnection networks in UMA architecture is presented. Crossbar, multiple buses and general shuffle networks (GSN) [2] are analyzed, where GSN is a very broad class of multistage interconnection networks. Some self-routing networks, such as Omega [3] and Delta [4] are its subclasses. It is shown that this method is not only more general but also very precise as compared with most existing analysis methods.

So far, there are three analytical approaches to analyzing the performance of the

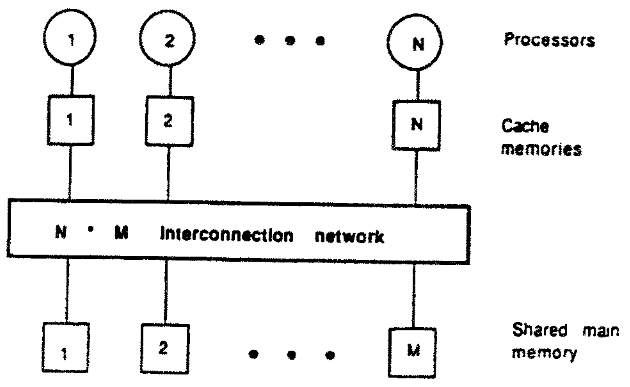

Fig. 2. A multiprocessor system with private caches.

UMA architecture with crossbar interconnection. The first one is a Markov chain analysis proposed by Bhandrakar [5]. It is very time consuming because of the enormous number of possible states. The second method was proposed by Strecker [6] for analyzing similar systems in which, if several processors happen to generate requests for accesssing the same memory, only one request will be accepted and all others will be missing. Although such an assumption may simplify the analysis, it will underestimate the memory bandwidth and thus the system power of a regular multiprocessor system. In [7], Yen et al. further modify the request rate to get an approximate memory bandwidth. However, this approach was applied to full crossbar interconnections only.

Since then, researches have been extended to the analysis of multiple-buses and multistage interconnection networks. Many researchers, such as Patel, Lang, Bhuyan, Das and Mudge have proposed quite a few interesting approaches [8–16]. However, some of them are not so general when analyzing all these three types of interconnection networks. Some of them may underestimate the bandwidth and the system power. We now present an alternative method which can be applied to estimate the performance of UMA interconnection networks more generally and/or precisely than most existing approaches.

Our analysis method applied to the three types of UMA interconnection networks is presented in Section 2. The approximate analysis of the UMA system with private caches is presented in Section 3. Section 4 shows some analysis results and comparisons with a few well-known methods. A brief conclusion is given in Section 5.

2. Analysis of interconnection networks.

In this section, we focus on the analysis of the architecture shown in Figure 1, in which the interconnection networks can be crossbar, multiple buses or GSN. Before analyzing such systems, a few concepts and assumptions are required to clarify our presentation.

First, *System Power* is defined as the average number of busy processors in a memory cycle. *Bandwidth* is the average number of busy memory modules in a cycle. *Waiting Time* is the average time since a processor generates a request until it gets the memory service. When we analyze the performance of such multiprocessor systems, these measures are used as fundamental indices. These three indices are closely related and given any one of them, the other two can be obtained. The less the waiting time is, the more the memory bandwidth can be utilized and thus the higher system power can be achieved. Throughout this paper, we will focus our interest on deriving the formulae for calculating these three performance indices for the systems consisting of various interconnection networks.

Our analysis is based on the technique of *Markov Chain*, which calculates the probability of each possible state in the state space. In our study, we use the *number of busy processors in the system* as the state of the system. The size of the state space is therefore linearly proportional to the size of the system. The following assumptions are made to simplify the state transitions in our analysis.

ASSUMPTION 1: The system has N processors and M memory modules. It will be referred to as $N \times M$ system.

ASSUMPTION 2: All memory modules have equal constant cycle time and their operations are synchronous. All processors are identical and they all generate their requests at the beginning of a memory cycle.

ASSUMPTION 3: The processors and memories are connected by crossbar, multiple buses or GSN, which allows every processor to have an access route to every memory module. The propagation delay and arbitration time in these interconnection networks are assumed to be ignorable. Alternatively, they may be considered as part of the memory cycle.

ASSUMPTION 4: From each memory module, only one word can be accessed at a time. If two or more processors simultaneously make requests to the same memory module, only one of these requests can be served in that memory cycle. The other processors will *redistribute* its request to the memory modules in the next memory cycle. (The reason of this assumption will be explained later.)

ASSUMPTION 5: The conflict resolutions among processors in accessing the memory modules, buses and switches in crossbar and GSN are unbiased, i.e., there is no processor having the priority to get any particular path from itself to some memory module.

ASSUMPTION 6: The requests are randomly distributed to the memory modules and the request generated by a processor is independent of the request generated by another processor.

ASSUMPTION 7: The request generated in a cycle is independent of the request generated in the previous cycle.

ASSUMPTION 8: If the processor is not waiting for memory service, the process of generating a request is a *Bernoulli* trial and we define p to be the request rate, i.e., the probability that a processor generates a memory request.

Among the assumptions above, it seems that only Assumption 4 is a little bit unrealistic because the requests blocked in a memory cycle are usually *resubmitted* to the same memory module instead of being *redistributed* to some module in the subsequent cycle. However, we shall show that this assumption may help to simplify the analysis but will cause little bias in the performance estimation. The simplification of the analysis will become clear in the derivation of the complete set of performance calculating formulae, and its unbiasedness will be illustrated by comparing its analysis results with the simulation data of the real situation without this assumption in the later sections. The analysis results will also be compared with those results obtained by using other analytic methods.

2.1. *Relations among performance indices.*

A processor in the multiprocessor systems is assumed to be always in one of the two states, busy or waiting. It is either busy in doing certain useful work, or idle and waiting for the memory service. If we investigate the activities of a processor under our assumptions, it is easy to see that the relations among the three performance indices are independent of whether the interconnection networks is crossbar, multiple buses or GSN.

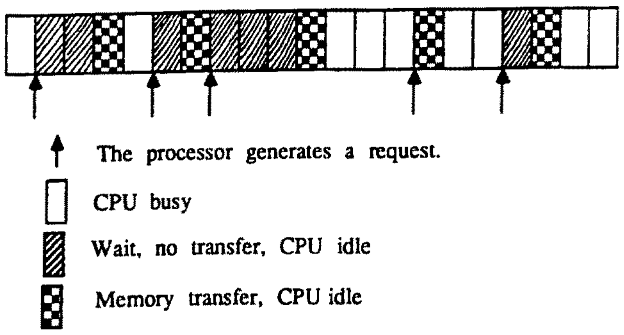
The processor generates a request.
CPU busy
Wait, no transfer, CPU idle
Memory transfer, CPU idle

Fig. 3. The activity of a single processor.

Consider Figure 3, which shows a possible sequence of activities of a single processor. When a processor is not waiting for the memory service, it may generate a request with probability p and the probability that the processor does local computation is thus $(1 - p)$ in a cycle. If the processor does k computations, then

$$p: (1 - p) = x: k. \tag{1}$$

Thus the average number x of memory requests generated by the processor in k computations is

$$x = kp/(1 - p). \tag{2}$$

Let ω denote the average waiting time (number of waiting cycle) per memory request described above, then the processor utilization is

$$\begin{aligned} U &= k/(k + k(w + 1)p/(1 - p)) \\ &= 1/(1 + (w + 1)p/(1 - p)). \end{aligned} \tag{3}$$

The system power can thus be calculated as follows:

$$P = N/(1 + (w + 1)p/(1 - p)) \tag{4}$$

and the memory bandwidth is

$$\begin{aligned} BW &= N(kp/(1 - p))/(k + k(w + 1)p/(1 - p)) \\ &= P(p/(1 - p)). \end{aligned} \tag{5}$$

From (3), (4), (5), if we can obtain any one of these performance indices, we actually know all the others.

2.2. *General scheme.*

Imagine that we are now analyzing a multiprocessor system where all blocked requests will be redistributed in the next memory cycle. With this assumption the Markov state can thus be represented by the number of requests in the beginning of a cycle. We define some notations as follows.

S_i	is the state when there are i requests in the beginning of a cycle.
$TRAN(i,j)$	is the probability that the state changes from S_i to S_j.
$THRU(i,j)$	is the probability that in the beginning of a cycle, there are i requests which pass through the interconnection network and j of them are accepted by memories.
$GEN(i,j)$	is the probability that when there are i processors which are not waiting for the memory service in the beginning of a cycle, j of them generate new requests.

From Assumption 8, a processor will generate a request according to the Bernoulli trial process. If there are i free processors, i.e., any of them can generate requests with probability p, then the process of generating requests is a binomial distribution with parameters i and p. Thus

$$GEN(i,j) = \begin{cases} 0 & \text{if } j > i \\ \binom{i}{j} p^j (1-p)^{(i-j)} & \text{otherwise.} \end{cases} \tag{6}$$

If we can get the value of $THRU(i,j)$ for all i and j, then

$$TRAN(i,j) = \begin{cases} GEN(N,j) & \text{if } i = 0 \\ \sum_{a=1}^{i} THRU(i,a) \times \\ \qquad GEN(N-i+a, j-i+a) & \text{otherwise.} \end{cases} \tag{7}$$

Thus we can establish the Markov state diagram as shown in Figure 4. This can be

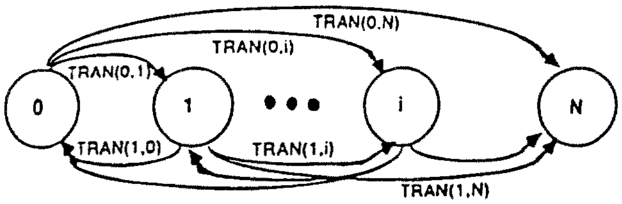

Fig. 4. The discrete Markov state diagram for multiprocessor system.

easily solved by the Gauss elimination method or any other similar methods. Let $pr(i)$ be the probability of state S_i. Then the system power is

$$P = \sum_{i=0}^{N} (N-i) pr(i). \tag{8}$$

Note that the number of states is exactly the number of processors. The solution of this *Markov Chain* is thus very easy to obtain and its complexity is only $O(N^3)$.

Now the major remaining problem is simply how to get the value of $THRU(i,j)$. Its solution, of course, depends on the type of the interconnection networks. The following sections provide the solution of $THRU(i,j)$ for the crossbar, multiple buses and GSN, respectively.

2.3. *Crossbar.*

The crossbar architecture is shown in Figure 5. Because there is no conflict in the crossbar, the problem of solving $THRU(i,j)$ is equivalent to the occupancy problem with i balls and M urns where M is the number of memory modules.

Thus

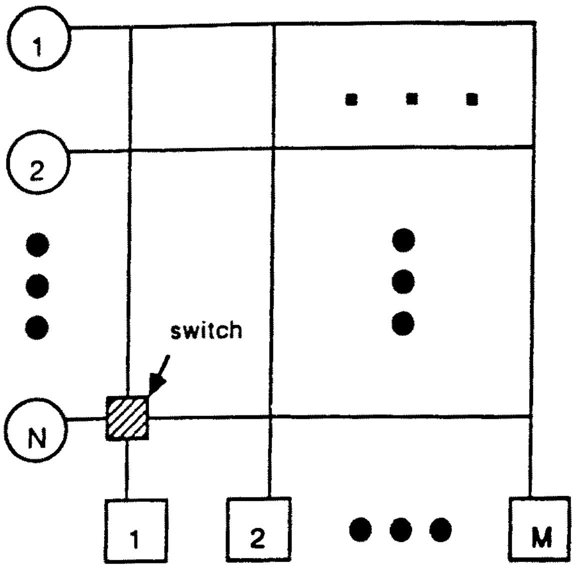

Fig. 5. The organization of crossbar interconnection.

$$(9) \qquad THRU(i,j) = \begin{cases} \binom{M}{j} \sum_{a=0}^{j-1} (-1)^a \binom{j}{a} \left(\frac{j-a}{M}\right)^i & \text{if } y \le i \text{ and } j \le M \\ 0 & \text{otherwise.} \end{cases}$$

2.4. *Multiple buses.*

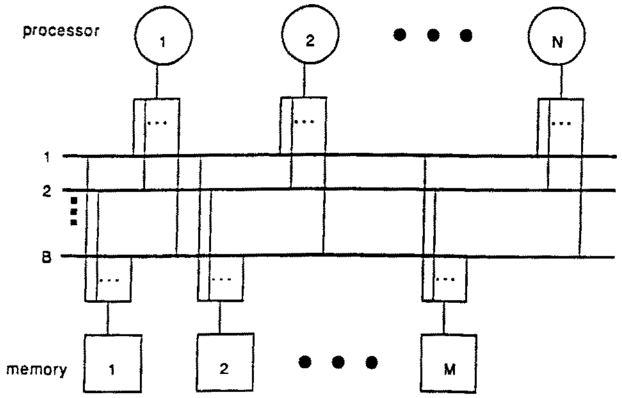

Fig. 6. An $N \times M \times B$ multiple bus network.

The architecture of a multiple-bus system is shown in Figure 6, and we refer to it as an $N \times M \times B$ system.

The problem of finding $THRU(i,j)$ is similar to the occupancy problem with only a small difference. If the number of memory modules requested by the processors is greater than B, then only B requests can be accepted by memories. Thus

$$(10)\quad THRU(i,j) = \begin{cases} \binom{M}{j}\sum_{a=0}^{j-1}\binom{j}{a}(-1)^a\left(\frac{j-a}{M}\right)^i & \text{if } j < B \\ \sum_{x=B}^{i}\left\{\binom{M}{x}\sum_{a=0}^{x-1}\binom{x}{a}(-1)^a\left(\frac{j-a}{M}\right)^i\right\} & \text{if } j = B \\ 0 & \text{otherwise.} \end{cases}$$

2.5. *General shuffle network* (*GSN*).

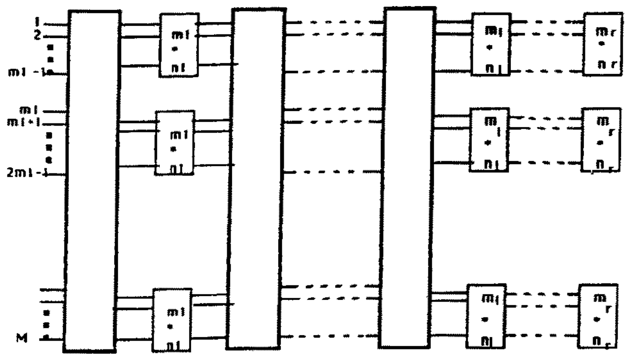

Fig. 7. An $M \times N$ generalized shuffle network where $M = m_1 \times m_2 \times \ldots \times m_r$ and $N = n_1 \times n_2 \times \ldots \times n_r$.

A generalized shuffle network (GSN) [2] is a broad class of networks. The M inputs connect to N outputs for any arbitrary values of M and N. Let M and N be the products of r-terms such as $M = m_1 \times m_2 \times \ldots \times m_r$ and $N = n_1 \times n_2 \times \ldots \times n_r$. An $M \times N$ GSN with M inputs and N outputs is an r-stage interconnection network as shown in Figure 7, consisting of a few crossbar switches of size $m_i \times n_i$ at the ith stage for all $1 \leq i \leq r$.

This stage of GSN can be considered to consist of independent crossbars. If more than one request at the input side of a crossbar attempts to reach the same link at the output side, then only one of these requests will be accepted, and the others are rejected by the crossbar. Let $R(m, n, x, y)$ be the probability that there are y requests *rejected* by a *single* crossbar given that there are x requests in the input side of the $m \times n$ crossbar. By taking $M = n$, $i = 1$ and $j = x - y$ in equation (9), it is easy to find that

$$(11)\quad R(m,n,x,y) = \begin{cases} \binom{n}{x-y}\sum_{a=0}^{x-y-1}\binom{x-y}{a}(-1)^a \times \binom{x-y-a}{m} \\ \qquad\qquad\qquad\qquad \text{if } x \le m \wedge y \le n \wedge y \le x \\ 0 \qquad\qquad\qquad\qquad \text{otherwise.} \end{cases}$$

Given h crossbar switches of size $m \times n$, each having b input requests, let $Q(h,m,n,b,d)$ be the probability that there are d *rejected* requests within the $b \times h$ requests to these crossbars. We can calculate this probability by solving recurrent equations as shown by the following lemma.

LEMMA 1. *There exists a recurrent relation to calculate $Q(h,m,n,b,d)$.*

PROOF. By the notation of generating function and equation (11) which is the case of a single crossbar, $Q(h,m,n,b,d)$ is the *coefficient* of x^d in the following expression

$$(12)\qquad \left(\sum_{i=0}^{b-1} R(m,n,b,i)x^i\right)^a.$$

We expand this to get the coefficient of x^d, and the result is

$$(13)\qquad Q(h,m,n,b,d) = \sum_{\eta=0}^{b-1} R(m,n,b,\eta)Q(h-1,m,n,b,d-\eta)$$

where the factor $R(m,n,b,\eta)$ in the right hand side of equation (13) can be considered as if the first crossbar rejected η requests and the factor $Q(h-1,m,n,b,d-\eta)$ as if the other $(h-1)$ crossbars rejected $(d-\eta)$ requests. It is trivial that $Q(h,m,n,b,d)$ reduces immediately to $R(m,n,b,d)$ when $h=1$, i.e., it only has one crossbar. Thus

$$(14)\quad Q(h,m,n,b,d) = \begin{cases} R(m,n,b,d) & \text{if } h=1 \\ \sum_{\eta=Max(0,b-n)}^{Min(d,b-1)} R(m,n,b,\eta) \times & \\ Q(h-1,m,n,d,d-\eta) & \text{if } h>1 \\ 0 & \text{otherwise.} \end{cases}$$ ■

Let $S(a,m,n,b,x,y)$ be the probability that, given that there are totally x requests in the input side of a crossbars with size $m \times n$, there are y requests reaching output links, and each individual crossbar has *at most b* input requests.

THEOREM 1. *There exists a recurrent relation to calculate $S(a,m,n,b,x,y)$.*

PROOF. By the definition, the Product rule and the Sum rule, $S(a,m,n,b,x,y) = \sum_\alpha \sum_\beta \text{prob}\{$[there are α crossbars having exactly b input requests] and [the α crossbars will totally reject β requests while any one of the α crossbars having exactly b input requests] and [given that there are totally $(x-\alpha b)$ requests in the input sides of the remaining $(a-\alpha)$ crossbars, any one of the $(a-\alpha)$ crossbars has *at most* $(b-1)$ input requests and $(y-\alpha b+\beta)$ requests of them can reach the output links in the $(a-\alpha)$ crossbars]$\}$.

By lemma 1,

$$S(a,m,n,b,x,y) = \sum_{\alpha}\sum_{\beta} \frac{\binom{a}{\alpha}\binom{m}{b}^{\alpha}\binom{m(a-\alpha)}{x-\alpha b}}{\binom{am}{x}}$$

$$\times\ Q(\alpha,m,n,b-1,\beta)$$

$$\times\ S(a-\alpha,m,n,b-1,x-\alpha b,y-\alpha b+\beta).$$

Because there are α crossbars having b input requests and the $(a-\alpha)$ remaining crossbars have at most $(b-1)$ input requests, therefore,

$$0 \le x-\alpha b \le (b-\alpha)(a-\alpha) \quad \text{and} \quad \alpha \ge 0.$$

This implies that

$$\text{Max}(0, a+x-ab) \le \alpha \le \lfloor x/b \rfloor.$$

For a single crossbar with b input requests, it would reject at most $(b-1)$ requests. Therefore for α crossbars each with b input requests, they totally reject at most $\alpha(b-1)$ requests. Therefore

$$0 \le \beta \le \alpha(b-1).$$

As a result,

$$(15)\quad S(a,m,n,b,x,y) = \begin{cases} 0 & \text{if } (b=1 \wedge x \ne y) \text{ or } (b=1 \wedge x=y \wedge x>a) \\ 1 & \text{if } x=y=0 \\ \sum_{\alpha=Max(0,\,a+x-ab)}^{\lfloor x/b\rfloor}\sum_{\beta=0}^{\alpha(b-1)} \binom{a}{\alpha}\binom{m}{b}^{\alpha}\Big/\binom{am}{x} & \\ \times\binom{m(a-\alpha)}{x-\alpha b} Q(\alpha,m,n,b,\beta) & \\ \times\ S(a-\alpha,m,n,b-1,\ x-\alpha b,y-\alpha b+\beta) & \text{otherwise} \\ \text{where } a,\ m,\ n,\ x,\ y \text{ are all nonnegative integers.} & \end{cases}$$

Let t^i_{ab} be the single-stage probability of having b requests surviving at the outputs of the ith stage of GSN, given a requests at the inputs of the ith stage. Let T^i be the single-stage matrix of the ith stage such that $T^i = [t^i_{ab}]$.

Applying Theorem 1, it is clear that

$$(16)\qquad t^i_{ab} = S(l,m,n,m,a,b)$$

where the ith stage of GSN consists of l crossbars with size of $m \times n$.

Again, let $g^i_{n_0 n_i}$ be the probability of having n_i requests surviving at the outputs of the ith stage, given n_0 initial requests at the inputs of the first stage, and let G^i be the i-stage matrix such that $G^i = [g^i_{n_0n_i}]'$ where $g^i_{n_0n_i}$ is found by applying (16) at each stage as follows.

1. The probability $g^1_{n_0n_i}$ of having n_1 outputs from the first stage, given that n_0 randomly generated inputs from the processors is given by (16).
2. Given n_1 inputs to the second stage, which are randomly distributed because we assume an unbiased conflict resolution policy at all stages, the probability of having n_2 outputs at the second stage is given by

$$\begin{aligned}(17)\qquad g^2_{n_0n_2} &= Prob[X_2 = n_2 \mid X_0 = n_0]\\ &= \sum_{n1=0}^{N} Prob[X_1 = n_1 \mid X_0 = n_0] Prob[X_2 = n_2 \mid X_1 = n_1]\\ &= \sum_{n_1=0}^{N} t^1_{n_0n_1} t^2_{n_1n_2}; \qquad 0 \le n_0, n_2 \le N\end{aligned}$$

   where N is the number of the processors and the random variable X_i, $i \ne 0$, is the number of the output links of the ith stage. The random variable X_0 is the number of requests generated by the processors.
3. If n_3 is the number of requests at the outputs of the third stage, we have similarly

$$(18)\qquad g^3_{n_0n_3} = \sum_{n_2=0}^{N} g^2_{n_0n_2} t^3_{n_2n_3}; \qquad 0 \le n_0, n_3 \le N.$$

We generalize this process to get the input-output relation of an r-stage GSN as

$$(19)\qquad g^r_{n_0n_r} = \sum_{n_{r-1}}^{N} g^{r-1}_{n_0n_{r-1}} t^r_{n_{r-1}n_r} \qquad 0 \le n_0, n_{r-1}, n_r \le N.$$

Clearly $g^r_{n_0n_r}$ is the r-stage probability; thus

$$(20)\qquad THRU(n_0, n_r) = g^r_{n_0n_r}.$$

Besides, let the r-stage matrix of these r-stage probabilities be denoted by G^r, i.e., $G^r = [g^r_{ab}]$. From (19), we have

$$(21)\qquad G^r = \prod_{i=1}^{r} T^i. \qquad \blacksquare$$

The value of $THRU(i,j)$ can thus be calculated for the crossbar, multiple-buses, and GSN according to equations (9), (10) and (20) respectively. As a result, the state transition probabilities and then the processor utilization, the system power as well as the memory bandwidth of the multiprocessor system can be easily obtained no matter which type of interconnection network is used.

3. Approximate analysis of multiprocessor systems with private caches.

In a multiprocessor system with private caches as shown in Figure 2, the memory transfer time is no longer a single memory cycle. At each cycle, a cache may make a request to the main memory with probability m which is considered as proportional to the miss ratio. After a period of delay, a data transfer takes place. Throughout the whole cache miss period, the processor remains idle. A processor in our system is thus either busy doing computation or idle waiting for a cache miss service.

The crossbar, multiple bus and GSN interconnections will again be studied here. These networks will be assumed to operate in the circuit switching mode. Once a miss occurs in a cache, the cache miss handling hardware would request a block transfer from a particular main memory module and the network established a path between the cache and the main memory modules. This path is held until the memory access is completed. The path cannot be preempted by any other requests coming from other cache modules. In this description it is assumed that a block would reside in a single memory module. However, each memory module itself may be interleaved with others to increase the bandwidth. The advantage of using circuit switching and storing each individual block in one memory module is the reduction in block transfer time. In the three interconnection networks, there is an initial delay in establishing a path due to arbitration, decoding, and setting the appropriate switches. Once the path is established the data can be transferred at a higher rate.

In reality, there may be various types of requests in a multiprocessor system with private caches. Different types may have different request rate and transfer time. For example, in the simple write through protocol, there may be a read request for a block transfer and a store request for a single word transfer. In our analysis, we assume that the interconnection network assigns equal priority to all different requests.

To analyze such systems, we adopted the approach used in [12] into our method. We apply our analysis with the modified request rate, which is derived from the original request rate of each type of the memory access and its transfer time, as seen by the interconnection network. The approximation is that t consecutive requests to a single memory module are decomposed into t separate requests which are randomly, independently and uniformly distributed over all modules. Although such a system condition is no longer equivalent to the original one, the offered traffic between the caches and the memory modules is not changed on the average.

Let us first examine a simple cache model in which each cache miss causes a period of waiting delay followed by t time units of data cache between a memory module and the cache. Consider Figure 8, which shows the situation of processor activities with respect to a few cache misses and waiting periods. Since in each processor cycle a cache miss may happen with probability m, there are on the average x faults for k units of useful computation, where $x:k = m:(1 - m)$. Thus

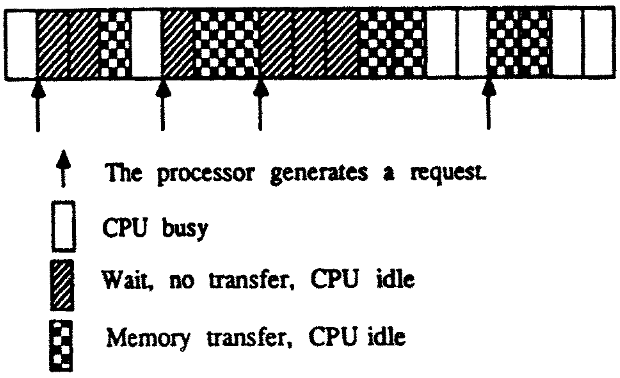
The processor generates a request.
CPU busy
Wait, no transfer, CPU idle
Memory transfer, CPU idle

Fig. 8. The activity of a single processor.

$$x = km/(1 - m). \tag{22}$$

Let t be the block transfer time, then it has totally $mkt/(1 - m)$ time units spent in block transfers within k units of useful computation.

Consider Figure 8 again, which shows the activity of a single processor. While a request is granted, the interconnection network maintains a path to a memory module for t time units. We can view this as t consecutive requests to the same module, each request requiring one time unit of service. Thus, from (22), it has $mkt/(1 - m)$ requests for unit service within k units of useful computation. Therefore, the request rate (for unit service) from a cache module as seen by the interconnection network is

$$m' = [mkt/(1 - m)]/[k + mkt/(1 - m)] = mt/(1 - m + mt). \tag{23}$$

In the result we can apply the methods discussed above with such modified request rate, m', to each of the corresponding types of interconnection network.

For further elaboration, suppose that there exists a particular multiprocessor system which has i different types of requests. We define m_a to be the request rate of the a-type request and t_a its transfer time.

Summarizing the activities of a single processor, we have

$$k : x_1 : x_2 : \ldots : x_i = (1 - \textstyle\sum_{\alpha=1}^{i} m_\alpha) : m_1 : m_2 : \ldots : m_i$$

where there are k time units of useful computation, and x_a is the average number of a-type requests. Thus

$$x_a = m_a k/(1 - \textstyle\sum_{\alpha=1}^{i} m_\alpha) = y_a k$$

where

$$y_a = m_a/(1 - \textstyle\sum_{\alpha=1}^{i} m_\alpha).$$

Then, we have the modified request rate

(24)
$$m' = \sum_{\alpha=1}^{i} y_\alpha t_\alpha k/(k + \sum_{\alpha=1}^{i} y_\alpha t_\alpha k)$$
$$= \sum_{\alpha=1}^{i} y_\alpha t_\alpha/(1 + \sum_{\alpha=1}^{i} y_\alpha t_\alpha)$$

which is the request rate from a cache module as seen by the interconnection network. We can similarly apply the same method using such a modified request rate for analyzing the target interconnection networks connecting multiple processors/caches to memories.

4. Analysis results and comparisons.

4.1. *Analysis and simulation results of interconnection networks without caches.*

We now show some analysis results and the corresponding simulation results. The simulation was written in C and all memory requests were generated and distributed randomly. The simulation conditions and constraints follow the same assumptions as in our analytical model except that all rejected requests are resubmitted to the same memory module instead of redistribution at the next cycle. We show that Assumption 4 in our model would not impact the accuracy of the analysis results. The simulation was run for 400,000 time units. Each result is the mean of all those sample values taken every 20,000 time units for a total of 20 samples during the simulation run.

From Tables 1, 2, 3, we can see that there is little difference between the computation and the simulation results. This is because a blocked request will eventually be submitted to a certain memory module. It makes little difference to consider it as a new request which is created and distributed to those memory modules. In a reasonable period of time the average traffic in the interconnection network and the workload of the memory modules thus stay the same as compared with the actual traffic where a blocked request will be resubmitted in the next cycle.

The memory bandwidths of crossbars, multiple buses and GSN are not listed because they can be obtained easily using Equation (5) based on Table 1, 2 and 3. For ease of comprehension, Figures 9 through 12 illustrate the trends of the analytical results of bandwidths obtained with our method. The corresponding values of some curves can be found in Table 10 through 12.

4.2. *Analysis and simulation results of interconnection networks with caches.*

Tables 4, 5, 6 list several results obtained by using our approximate analysis of a multiprocessor system with private caches and compare them with the simulation results. The results are compared over a wide range of parameters. The varied parameters are: request rate m from 1/128 to 1/2, block transfer time t from 1 to 64 units.

Table 1. *The system power of $N \times M$ crossbar interconnection under various request rates, the number of processors is N and the number of memory modules is M.*

	16 × 16		32 × 32	
rate	analytical	simulation	analytical	simulation
0.1	14.33	14.34	28.65	28.75
0.2	12.54	12.56	25.15	25.13
0.3	10.67	10.65	21.80	21.84
0.4	8.80	8.80	17.54	17.58
0.5	6.98	7.01	13.91	13.87
0.6	5.28	5.29	10.51	10.50
0.7	3.73	3.70	7.41	7.43
0.8	2.33	2.33	4.63	4.63
0.9	1.10	1.10	2.17	2.17
1.0	0.00	0.00	0.00	0.00

Table 2. *The system power of $N \times M \times B$ multiple bus interconnection, the number of processors is N, the number of memory modules is M and the number of buses is B.*

	32 × 32 × 16		32 × 32 × 8	
rate	analytical	simulation	analytical	simulation
0.1	28.65	28.70	28.65	28.66
0.2	25.05	25.02	24.87	24.93
0.3	21.30	21.38	18.42	18.37
0.4	17.53	17.52	11.99	11.98
0.5	13.79	13.77	7.99	7.99
0.6	10.12	10.16	5.34	5.35
0.7	6.76	6.76	3.43	3.43
0.8	3.99	4.00	2.00	2.00
0.9	1.77	1.77	0.88	0.99
1.0	0.00	0.00	0.00	0.00

Table 3. *The system power of GSN interconnection.*

	(4 × 4) × (4 × 4)		(8 × 4) × (4 × 8)	
rate	analytical	simulation	analytical	simulation
0.1	14.29	14.32	28.38	28.39
0.2	12.36	12.37	23.96	24.01
0.3	10.37	10.36	19.10	19.10
0.4	8.30	8.30	14.44	14.42
0.5	6.40	6.40	10.56	10.55
0.6	4.71	4.69	7.46	7.43
0.7	3.25	3.26	4.96	4.96
0.8	2.00	2.00	2.96	2.97
0.9	0.91	0.91	1.33	1.33
1.0	0.00	0.00	0.00	0.00

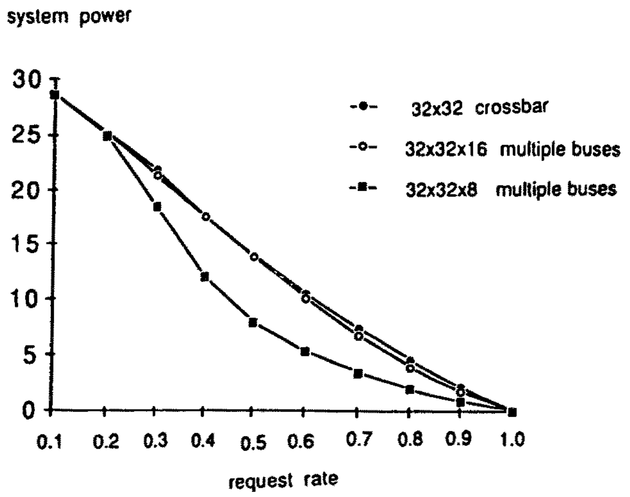

Fig. 9. The system powers of crossbar and multiple-bus interconnections.

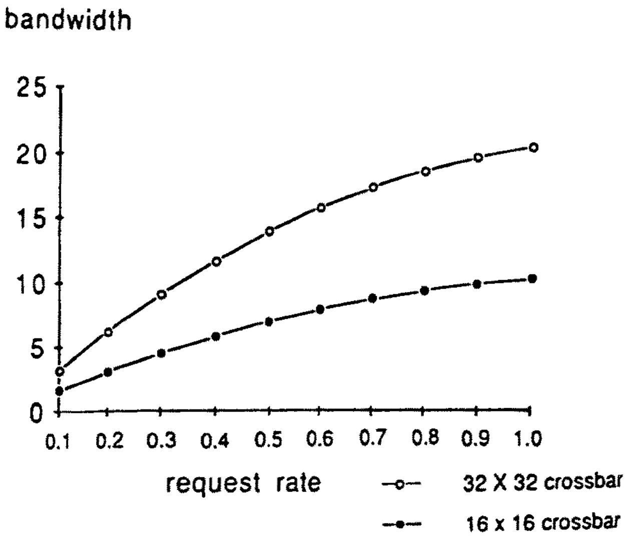

Fig. 10. The memory bandwidth of crossbar interconnection networks.

In these three tables, the system powers obtained from our approximate method are a little bit higher than from the actual simulation. When the block transfer time is small, there is little difference between the computation and the simulation results. The difference grows as the block transfer time increases. If we compute the percentage of difference relative to the simulation result, the highest relative difference is 2 to 4 percent, which occurs when the block transfer time is 64. For most cases, the difference is less than 1 percent.

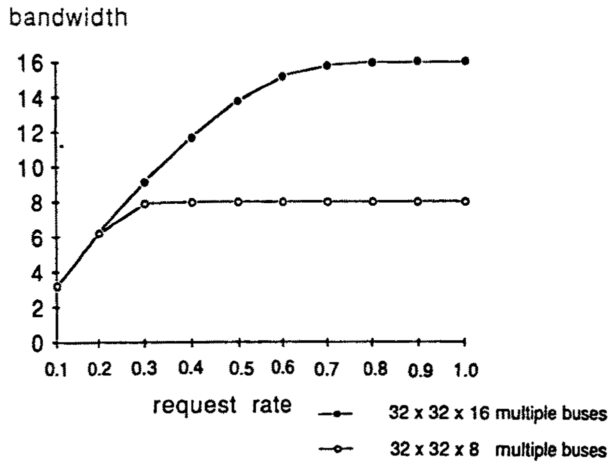

Fig. 11. The memory bandwidth of multiple-bus interconnection networks.

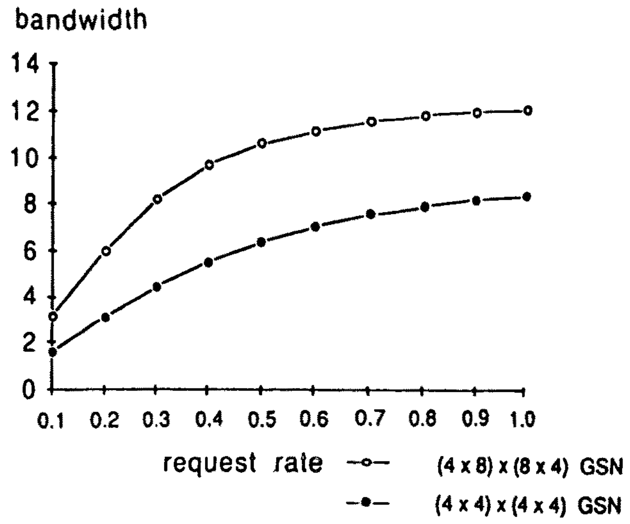

Fig. 12. The memory bandwidth of GSN interconnection networks.

As for the situation with multiple type of memory requests, we have applied the approximate method to the case with two types of request. One is word-request whose transfer time is one unit, and the other is block transfer. Tables 7, 8, and 9 compare the analytical and simulation results. The highest differences are 5 to 8 percent which occur when the transfer time is 16. For most cases, the differences are less than 2 percent.

Table 4. *The system power of crossbar interconnection with caches. The number of processors is* 32 *and the number of memory modules* 32. *The upper part of each cell is the analytical result, and the lower part is the simulation result.*

rate	Block transfer time						
	1	2	4	8	16	32	64
2^{-1}	13.91	8.41	4.63	2.43	1.24	0.63	0.31
	13.87	8.39	4.63	2.41	1.22	0.63	0.31
2^{-2}	23.20	17.64	11.45	6.62	3.56	1.84	0.94
	23.27	17.57	11.43	6.56	3.54	1.81	0.93
2^{-3}	27.77	24.23	18.90	12.75	7.54	4.11	2.14
	27.80	24.29	18.89	12.63	7.36	3.95	2.07
2^{-4}	29.94	28.02	24.66	19.50	13.34	7.98	4.37
	29.94	28.02	24.63	19.42	13.10	7.67	4.20
2^{-5}	30.98	30.00	28.15	24.86	19.78	13.63	8.20
	30.98	29.99	28.12	24.84	19.68	13.27	7.92
2^{-6}	31.50	31.00	30.04	28.21	24.96	19.91	13.77
	31.30	30.99	30.02	28.18	24.91	19.73	13.22
2^{-7}	31.75	31.50	30.99	30.05	28.24	25.00	19.98
	31.75	31.49	30.99	30.02	28.23	24.89	19.86

4.3. *Comparison of analysis results between our method and others.*

Our method is both general and precise enough to analyze the performance of any of crossbars, multiple buses and generalized shuffle networks. As shown in the last section, the analysis results are considerably precise as compared with the real simulation in which the blocked memory requests are resubmitted to the same module instead of redistribution at the next memory cycle. In order to demonstrate the superiority of our method, we present comparisons with other methods and their analysis results in this section.

Table 10 lists the bandwidths of the 32 $\times$ 32 crossbar under different request rates. It is easy to see that the discarded-request approach used in [6] yields a result which underestimates the bandwidth because all blocked requests due to memory conflicts are discarded. The results from both Yen's method [7] and our method are very close to the simulation results. However, Yen's method is applicable only to crossbar type interconnections and is thus more restricted than our method.

Table 11 lists the bandwidths of 32 $\times$ 32 $\times$ 16 and 32 $\times$ 32 $\times$ 8 multiple buses under different request rates. Das's method [11] always results in a lower bandwidth estimation. It is also because that the blocked memory requests are assumed to be discarded in Das's method. Das's method can be applied only to multiple bus

Table 5. The system power of multiple bus interconnection with caches. The number of processors is 32, *the number of memory modules* 32 *and the number of buses* 16. *The upper part of each cell is the analytical result, the lower is the simulation result.*

rate	Block transfer time						
	1	2	4	8	16	32	64
2^{-1}	13.79 13.77	7.83 7.71	3.99 3.93	2.00 1.93	0.99 0.97	0.50 0.49	0.25 0.25
2^{-2}	23.19 22.28	17.53 17.73	11.15 11.02	5.95 5.79	2.99 2.94	1.50 1.47	0.75 0.74
2^{-3}	27.77 27.75	24.23 24.29	18.90 18.87	12.55 12.39	6.89 6.76	3.49 3.40	1.75 1.71
2^{-4}	29.94 29.94	28.03 28.03	24.66 24.66	19.50 19.43	13.19 12.94	7.37 7.19	3.74 3.68
2^{-5}	30.99 30.98	30.01 30.00	28.15 28.11	24.86 24.82	19.78 19.67	13.49 13.39	7.60 7.35
2^{-6}	31.50 31.49	31.00 30.99	30.03 30.02	28.21 28.14	24.96 24.83	19.91 19.66	13.64 13.36
2^{-7}	31.75 31.75	31.50 31.49	31.01 31.00	30.05 30.04	28.23 28.21	25.06 24.94	19.81 19.72

interconnection networks. Our method is therefore more precise and general than Das's method. On the other hand, Mudge's method [14] is based on a *Semi-Markov Interference* model. It assumes that all rejected memory requests will be resubmitted to the same module in the next cycle. The analysis results of Mudge's method are very close to ours. Both are remarkably precise. The advantage of Mudge's method is that it can model variable connection time easily. However, its analysis applies only to multiple-bus systems and is therefore restricted, too.

Table 12 lists the bandwidth of $(4 \times 4) \times (4 \times 4)$ GSN interconnection networks. Our method also results in higher bandwidth which is more precise than Patel's method [8]. The reason is that blocked memory requests are ignored in Patel's method. Patel's method cannot be applied to the analysis of multiple buses and is therefore also restricted.

From the comparisons above, it is obvious that our method is more general than most previous methods. It provides a fairly efficient analysis and can still yield precise results which are close to the real situation where the blocked memory requests are resubmitted to the same module in the next cycle.

Table 6. *The system power of* (4 × 8) × (8 × 4) *GSN with caches. The upper part of each cell is the analytical result, the lower is the simulation result.*

rate	Block transfer time						
	1	2	4	8	16	32	64
2^{-1}	10.56 10.55	5.74 5.72	2.98 2.90	1.50 1.49	0.75 0.75	0.38 0.38	0.19 0.19
2^{-2}	21.55 22.58	14.44 14.39	8.28 8.22	4.37 4.36	2.23 2.22	1.13 1.12	0.57 0.57
2^{-3}	27.34 27.38	22.90 23.00	16.07 16.04	9.46 9.44	5.06 5.06	2.60 2.60	1.31 1.30
2^{-4}	29.84 29.87	27.65 27.67	23.46 23.42	16.80 16.79	10.02 10.04	5.40 5.41	2.78 2.77
2^{-5}	30.96 30.96	29.91 29.91	27.79 27.77	23.71 23.70	17.15 17.12	10.29 10.30	5.57 5.57
2^{-6}	31.49 31.49	30.98 30.98	29.94 19.94	27.86 27.85	23.84 23.81	17.32 17.31	10.43 10.40
2^{-7}	31.75 31.75	31.49 31.49	30.98 30.97	29.96 29.96	27.89 27.87	23.90 23.89	17.49 17.47

Table 7. *The system power of crossbar interconnection with caches and variable requesting types.*

				System power			
				16 × 16		32 × 32	
m_1	t_1	m_b	t_b	analysis	simulation	analysis	simulation
0.1	1	0.01	2	14.00	14.01	27.99	28.00
0.1	1	0.01	4	13.70	13.68	27.38	27.35
0.1	1	0.01	8	13.11	12.90	26.21	26.14
0.1	1	0.01	16	12.05	11.50	24.08	23.24
0.1	1	0.05	2	12.71	12.72	25.40	25.40
0.1	1	0.05	4	11.40	11.38	22.78	22.61
0.1	1	0.05	8	9.35	9.11	18.65	18.02
0.1	1	0.05	16	6.73	6.49	13.40	12.48
0.2	1	0.01	2	12.21	12.19	24.39	24.38
0.2	1	0.01	4	11.92	11.91	23.82	23.82
0.2	1	0.01	8	11.39	11.22	22.75	22.19
0.2	1	0.01	16	10.42	9.78	20.81	20.26
0.2	1	0.05	2	10.94	10.93	21.84	21.86
0.2	1	0.05	4	9.77	9.71	19.45	19.36
0.2	1	0.05	8	7.98	7.74	15.90	15.11
0.2	1	0.05	16	5.76	5.48	11.45	19.57

Table 8. *The system power of crossbar interconnection with caches and variable requesting types.*

				System power			
				16×16		32×32	
m_1	t_1	m_b	t_b	analysis	simulation	analysis	simulation
0.1	1	0.01	2	27.98	27.99	27.99	28.00
0.1	1	0.01	4	27.36	27.33	27.38	27.35
0.1	1	0.01	8	26.13	25.98	26.21	26.14
0.1	1	0.01	16	23.47	22.76	24.08	23.24
0.1	1	0.05	2	25.20	25.20	25.40	25.40
0.1	1	0.05	4	21.32	21.11	22.78	22.67
0.1	1	0.05	8	13.59	13.35	18.64	18.06
0.1	1	0.05	16	7.55	7.41	13.25	12.44
0.2	1	0.01	2	23.90	23.90	24.39	24.37
0.2	1	0.01	4	23.08	22.94	23.82	23.81
0.2	1	0.01	8	21.26	20.88	22.75	22.54
0.2	1	0.01	16	7.55	7.41	13.25	12.44
0.2	1	0.05	2	19.51	19.48	21.84	21.83
0.2	1	0.05	4	14.98	14.61	19.39	19.32
0.2	1	0.05	8	10.00	9.88	15.86	15.59
0.2	1	0.05	16	6.00	5.89	11.15	10.72

Table 9. *The system power of GSN interconnection with caches and variable requesting types.*

				System power			
				$(8 \times 4) \times (4 \times 8)$		$(4 \times 4) \times (4 \times 4)$	
m_1	t_1	m_b	t_b	analysis	simulation	analysis	simulation
0.1	1	0.01	2	27.60	27.62	13.94	13.94
0.1	1	0.01	4	26.87	26.71	13.62	13.62
0.1	1	0.01	8	25.43	25.22	12.99	12.87
0.1	1	0.01	16	22.71	21.86	11.84	11.21
0.1	1	0.05	2	24.40	24.39	12.56	12.54
0.1	1	0.05	4	21.02	20.97	11.13	11.01
0.1	1	0.05	8	14.17	13.94	8.89	8.46
0.1	1	0.05	16	8.57	8.39	5.82	5.53
0.2	1	0.01	2	23.17	23.20	12.01	12.12
0.2	1	0.01	4	22.37	22.16	11.70	11.69
0.2	1	0.01	8	20.98	20.21	10.96	10.67
0.2	1	0.01	16	18.47	17.34	9.01	8.57
0.2	1	0.05	2	19.80	19.75	10.63	10.72
0.2	1	0.05	4	16.80	16.23	9.35	9.22
0.2	1	0.05	8	12.11	11.64	6.72	6.54
0.2	1	0.05	16	6.93	6.46	4.28	4.14

Table 10. *The memory bandwidth of* 32 × 32 *crossbars. Comparison between our method, discard-request method and Yen's method.*

rate	32 × 32 crossbar		
	our method	discard-request	Yen's method
0.1	3.18	3.05	3.18
0.2	6.29	5.82	6.24
0.3	9.34	8.33	9.30
0.4	11.69	10.60	11.64
0.5	13.91	12.67	13.79
0.6	15.75	14.54	15.53
0.7	17.29	16.23	16.91
0.8	18.52	17.77	18.00
0.9	19.53	19.16	18.85

Table 11. *The memory bandwidth of* 32 × 32 × 16 *and* 32 × 32 × 8 *multiple buses. Comparison between our method. Das's method and Mudge's method.*

rate	32 × 32 × 16 multiple bus			32 × 32 × 8 multiple bus		
	our method	Das's	Mudge's	our method	Das's	Mudge's
0.1	3.17	3.05	3.18	3.17	3.05	3.18
0.2	6.26	5.82	6.26	6.22	5.62	6.18
0.3	9.13	8.33	9.13	7.89	7.18	7.86
0.4	11.69	10.58	11.66	7.98	7.79	7.99
0.5	13.79	12.51	13.65	7.99	7.96	7.99
0.6	15.18	14.00	14.91	8.00	7.99	8.00
0.7	15.77	15.00	15.52	8.00	8.00	8.00
0.8	15.96	15.56	15.78	8.00	8.00	8.00
0.9	15.96	15.82	15.89	8.00	8.00	8.00

Table 12. *The memory bandwidth of* (4 × 4) × (4 × 4) *GSN. Comparison between our method and Patel's method.*

rate	(4 × 4) × (4 × 4) GSN	
	our method	Patel's method
0.1	1.57	1.49
0.2	3.07	2.77
0.3	4.44	3.88
0.4	5.53	4.83
0.5	6.40	5.66
0.6	7.06	6.38
0.7	7.58	7.01
0.8	8.00	7.55
0.9	8.19	8.03

5. Conclusions.

We have presented a method to analyze the performance of interconnection networks in UMA architecture. The computation effort of this method is reasonably low and the number of Markov states in the analysis is linearly proportional to the number of processors in the system. It provides a nice estimation of the performance of various interconnection networks for multiprocessor systems. Comparing the analytical results with the results of simulation in a practical situation, the highest difference is less than 2 percent. Compared with a few well-known methods, our results are also shown to be more precise.

Another advantage of our method is its generality. The analytical model can be widely used in various interconnection networks. We have analyzed the crossbar, multiple buses and GSN interconnections. Since no restriction is made on the connection pattern between stages of GSN in our analytical model, this method can easily be extended to all multi-stage interconnection networks. The multi-stage interconnection networks may include Butterfly, Delta, Omega, Base-line, Reverse-exchange and others. For the UMA architecture with private caches, the difference between the analytical and the simulation results may increase as the length of the block transfer time increases. The largest difference is no more than 8 percent. For most cases, the differences are less than 2 percent. It shows that the accuracy of the approximate method is remarkably good.

However, our method does not apply to NUMA architecture. There is also some possible improvement which can be done in our model when analyzing the UMA architecture. A few assumptions could be modified to allow more practical situations in a real multiprocessor system. For example, the processors and the memories may not be synchronized by a unified clock. They may run at different speeds. The interconnection delays may not be negligible, especially for a large multistage interconnection network. The path routing in the network and the accessing in the memory module may also be performed at a different pace. Any kind of these modifications on our original assumptions will result in a more complicated analysis. Whether they may have a significant impact on the analysis results will require further research efforts.

REFERENCES

1. M. Dubois and S. Thakkar, *Cache architectures in tightly coupled multiprocessors*, IEEE Computer, June 1990, pp. 9–11.
2. L. N. Bhuyan and D. P. Agrawal, *Design and performance of generalized interconnection networks*, IEEE Transaction on Computer, vol. C-32, Dec. 1983, pp. 1081–1090.
3. D. H. Lawire, *Access and alignment of data in an array processor*, IEEE Transaction on Computer, vol. C-24, Dec. 1975, pp. 1145–1155.
4. H. J. Siegel, *Analysis techniques for multiprocessors*, IEEE Transaction on Computer, vol. C-29, Oct. 1980, pp. 771–780.

5. D. P. Bhandarkar, *Analysis of memory interference in multiprocessor*, IEEE Transaction on Computer, vol. C-24, Sept. 1975, pp. 897–908.
6. W. D. Strecker, *Analysis of the instruction execution rate in certain computer structures*, Ph.D. dissertation, Carnegie-Mellon University, 1970.
7. D. Yen, J. Patel and E. Davidson, *Memory interference in synchronous multiprocessor systems*, IEEE Transaction on Computer, vol. C-31, Nov. 1982, pp. 1116–1121.
8. J. H. Patel, *Performance of processor-memory interconnections for multiprocessors*, IEEE Transaction on Computer, vol. C-30, Oct. 1981, pp. 771–780.
9. T. Lang, M. Valero and I. Alegre, *Bandwidth of crossbar and multiple-bus connection of multiprocessors*, IEEE Transaction on Computer, vol. C-31, Dec. 1982, pp. 1227–1234.
10. L. N. Bhuyan, *An analysis of processor-memory interconnection networks*, IEEE Transaction on Computer, vol. C-34, Mar. 1985, pp. 279–283.
11. C. R. Das and L. N. Bhuyan, *Bandwidth availability of multiple-bus multiprocessors*, IEEE Transaction on Computer, vol. C-34, Oct. 1985, pp. 918–926.
12. J. H. Patel, *Analysis of multiprocessors with private cache memories*, IEEE Transaction on Computer, vol. C-31, April 1982, pp. 296–304.
13. T. N. Mudge, J. P. Hayes, G. D. Buzzard and D. C. Winsor, *Analysis of multiple-bus interconnection networks*, Proceedings of IEEE 1984 International Conference on Parallel Processing, Aug. 1984, pp. 228–232.
14. T. N. Mudge and H. B. Al-Sadoun, *A Semi-Markov model for the performance of multiple-bus systems*, IEEE Transaction on Computer, vol. C-34, Oct. 1985, pp. 934–942.
15. J. P. Sheu and W. T. Chen, *Performance analysis of multiple bus interconnection networks with hierarchical requesting model*, Proceeding of 8th IEEE Conference on Distributed Computing Systems, June 1988, pp. 138–144.
16. Q. Yang and S. G. Zaky, *Communication performance in multiple-bus system*, IEEE Transaction on Computer, vol. C-37, July 1988, pp. 848–853.