A Multiprocessor Bus Design Model Validated by System Measurement

Thin-Fong Tsuei and Mary K. Vernon, *Member, IEEE*

Abstract— **In this paper we develop an accurate and efficient model of a commercial multiprocessor bus. Four important characteristics of the bus design are modeled: 1) asynchronous memory write operations, 2) in-order delivery of responses to processor read requests, 3) priority scheduling of memory responses, and 4) upper bounds on the number of outstanding processor requests. The represention of this complex behavior distinguishes the model from previous models of memory and bus interference in the literature. The model is a two-level hierarchical model employing both Markov chain and mean value analysis techniques for analyzing queueing networks. The model is shown to accurately predict *measured* system performance for two parallel program workloads that have different memory access characteristics. This also distinguishes this work from previous studies which have used system simulation to validate analytic models. The results in this paper provide evidence that analytic queueing models can be extremely accurate in spite of simplifying assumptions required for model tractability. Model estimates are compared against detailed simulation of the bus to investigate in more detail the likely source of small model inaccuracies. Finally, we illustrate the use of the analytical model for assessing system design tradeoffs.**

Index Terms— **Bus interference, hierarchical models, Markov chain analysis, mean value analysis, multiprocessors, performance evaluation, split transaction buses.**

I. Introduction

BUSES are used to interconnect processors and memory modules either in small scale multiprocessor systems or in subclusters of larger systems. A primary performance factor in bus based systems is the contention for bus and memory bandwidth. Previous studies have analyzed memory and bus interference assuming very simple bus protocols [8], [11], [15], [17], [18]. However, in practice buses often have complex features that impact performance. For example, decoupled transactions are often implemented in order to use bus bandwidth more efficiently. In this case, a processor relinquishes the bus while waiting for a response to a read request from memory or an IO device. The decoupling of transactions raises design issues such as response buffering and scheduling of bus operations. The goals of this paper are to develop an analytical model of a complex commercial bus and to validate this model against actual system measurement.

Manuscript received December 3, 1991; revised July 1 1992. This work was supported by the National Science Foundation under Grant DCR-8451405 and CDA-8920777, and by a grant from the AT&T Foundation.

T.-F. Tsuei is with the Advanced Technology Group, Apple Computer Inc., Cupertino, CA 95014.

M. K. Vernon is with the Computer Sciences Department, University of Wisconsin–Madison, Madison, WI 53706.

IEEE Log Number 9203547.

The purpose of the validation exercise is to test the hypothesis that the analytical model can yield results that are sufficiently accurate for evaluating system design tradeoffs in spite of typical simplifying assumptions that are required for model tractability.

We analyze the memory and bus interference in a commercial shared memory multiprocessor, namely the Sequent Symmetry/S-81 system. The S-81 bus employs a decoupled request-response protocol that 1) requires responses to come back on the bus in the same order the requests are issued on the bus, and 2) requires a processor read or write request to block when the respective specified limit on the number of outstanding requests has been exceeded. We show that an accurate and efficient analytic model can be developed in spite of these and two other features of the Symmetry system: 3) asynchronous write requests that can immediately follow processor read requests, and 4) priority scheduling on the bus for memory and cache responses to processor read requests.

We develop queueing network models that include a combination of hierarchical modeling and approximate mean value analysis to accommodate the above nonseparable characteristics of the system. Previously developed approximations are employed to model the asynchronous write operations and the priority scheduling on the bus. We develop new heuristics for the blocking of processor requests and out-of-order memory responses. Careful reasoning about possible system timing behavior simplifies the task in the case of the ordered memory responses,

Two models are developed. The first is a *response-blocking* model that includes all of the bus characteristics except the bounds on the number of outstanding requests. This model can be used to evaluate the performance impact of the response ordering requirement, as well as to assess the impact of removing bounds on the processor requests. The response-blocking model is then extended to a *complete* model that includes the blocking of processor requests due to bounds on the number of outstanding read and write requests.

The remainder of this paper is organized as follows. Section II contains an overview of the Sequent Symmetry system and our modeling approach. We develop our response-blocking model in Section III and our complete model in Section IV. In Section V, the models are validated against detailed simulation of the bus and against *measurements of the actual system* under two different (parallel program) workloads. Section VI illustrates the use of the model by exploring the impact of changes in certain aspects of the system design. Section VII contains the conclusions of this work.

II. Background

The Sequent S-81 is a single bus system in which up to 30 processors (each with a local cache) are connected to two shared memory modules and an I/O subsystem via a 10 MHz system bus. Below we describe the Sequent bus design, give an overview of our bus models, and review previous models of memory and bus interference.

A. The Sequent Symmetry S-81 Bus Design

We consider three types of transactions that can be initiated by the processors on the S-81 bus.[1] A read request (r) initiates a read operation in another cache or in one of the shared memory modules. When the read operation is complete the requested block of data will be returned on the bus. An invalidation request (iv) causes one or more other caches to invalidate their copies of the specified block of data. No data response is required for this transaction. An atomic read-write request (rw) initiates a read operation in another cache or one of the shared memory modules, as well as a write request to one of the shared memory modules. The read operation begins after the first cycle of the rw bus request. The write operation allows the processor to delete an entry in its cache to make room for the data requested by the read. The write operation completes asynchronously and write acknowledgments do not interfere with other requests or responses on the bus.

The main characteristics of the S-81 bus operation are [14]:

- Split transactions that decouple read requests and responses.
- Upper bounds on the number of outstanding processor requests: three reads, two writes, and/or one I/O request. A processor first arbitrates for the bus and then may block due to one of the limits on outstanding requests. Thus, a blocked processor request prevents other processors from acquiring the bus (even if they have a request that won't exceed any limit.)
- Read responses must be returned in the order that requests were issued on the bus. If a remote cache access or shared memory access completes but an earlier read request has not yet been satisfied, the responding memory or cache blocks until all earlier requests are satisfied before requesting the bus. This blocking does not prevent other transactions from using the bus.
- Nonpreemptive priority scheduling of bus operations. I/O responses have the highest priority, memory and cache responses to read requests have the next highest priority, and processor requests have the lowest priority. Processor requests are granted bus access in round robin order.

B. Performance Model Overview

Our analytical models of memory and bus interference for the S-81 bus design are based on the closed queueing network

[1] Other requests or combinations of requests can be issued by a processor, but (with the exception of I/O requests) are not observed in the execution of the parallel programs used in our validation experiments in Section V. I/O operations occur primarily at the beginning and end of the parallel programs used for model validation, which were excluded from the measurement intervals. The models can easily be extended to include other operations for examining performance under other workloads.

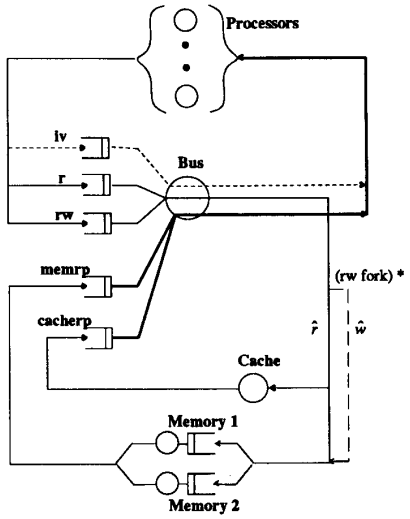

Fig. 1. Queueing network model for the S-81 MBI analysis.

of Fig. 1. This figure shows the routing of processor requests (iv, r, and rw) and memory and cache responses ($memrp$ and $carp$). In one *cycle*, a process executes for a random amount of time, makes a bus request, waits for the bus request to complete and, if the request is a read, waits for a response from shared memory or another cache. An invalidation request requires only a simple bus operation. A rw request forks into a read ($\hat{r}$) and a write ($\hat{w}$) request at the point indicated by (rw fork)* in Fig. 1. The asynchronous write request accesses one of the shared memory modules and then leaves the system.

A parameter of the model defines the probability that a read request is satisfied by another cache. Our models assume that all caches have equal probability of responding. Since each processor has at most one active remote memory read request and since the total number of outstanding read requests is bounded by three, the remote cache access time can be approximately modeled by a pure delay (no queueing) server. Two shared memory modules are modeled, each with equal probability of being accessed by a memory read or write operation. The service times at the bus, the memory modules and the remote cache are all deterministic. Blocking of processor requests and the ordering of the responses are not shown in the figure.

Processor execution times are represented by an infinite server center in the network. The bus is modeled as a multiple-class service center with priority scheduling of cache and memory responses. Round robin scheduling of the three processor requests classes is approximated by FCFS scheduling of these requests in the model.

C. Previous Work

The previous work most closely related to our study are the analytical models that focus on the general throughput characteristics of various bus and memory configurations [3], [8], [10], [11], [15]–[18], 21]. Modeling techniques used include Markov chains [3], [10], [15]–[18], [21], Markov

chains specified by Petri nets [11], and mean value analysis [8]. These models contain very simple workload parameters, such as frequency of remote memory requests and the probability distribution for which memory module is accessed. The models also assume very simple operation of the bus and memory subsystem. A recent study uses an analytical model to examine the performance implications of a few bus design features such as split transactions, and bus width (i.e., bus transaction latency) [6]. However, for the most part, complex features such those described above for the Sequent bus have not been modeled.

Another collection of related previous models are those that focus on the performance characteristics of various cache coherence protocols [2], [7], [19], [20], [25]–[27]. These models entail translating input parameters that characterize the data sharing behavior of programs into parameters that define the rates at which bus and memory operations are generated by the coherence protocol. System performance is then typically examined as a function of various program sharing characteristics. The coherence protocol models have also assumed simple operation of the bus and memory subsystem. For example, most of the models assume a circuit-switched bus in which read-response transactions are atomic [2], [7], [19], [20], [25], [26]. In this case, issues related to memory response buffering and scheduling do not arise. A comparative study of various snooping protocols that assumes a split-transaction bus [27] reaches very similar conclusions to a comparative study that assumes a circuit-switched bus [2], indicating that simple bus abstractions are probably appropriate for coherence protocol performance comparisons.

In contrast to the coherence protocol models, we develop a model of the fairly complex high-performance Sequent bus using simple workload parameters that define the frequency of various bus and memory transactions. In other words, we do not model the operation of any particular coherence protocol, but instead model the operation of a specific bus. For the particular design issues our model addresses (illustrated in Section VI), a more detailed bus model is required. Also in contrast to all of the previous work, we validate our analytical model against actual system measurement.

Since the Sequent system implements the Illinois coherence protocol [14], [20], the measured workload parameters in our validation experiments are a function of the programs running during the measurement interval and the Illinois coherence protocol. To analyze the same bus design for a workload with essentially no cache coherence operations or a system that implements a different invalidation-based coherence protocol (e.g., the Berkeley protocol [12]), only our workload parameter values would change. The workload parameters might be obtained by: 1) measuring an existing system as is done in this study, 2) measuring a workload of interest on a system that does not necessarily have the same detailed bus features, 3) trace simulation of a workload of interest (e.g., [6]), or 4) deriving the processor request frequencies from higher-level input parameters using a model of the appropriate coherence protocol. Parameters similar to the inputs we require are derived from a particular set of workload parameters that characterize data sharing behavior in [1], assuming an invalidation-based directory protocol with similar interconnection network requests to the Illinois protocol.

III. The Response-Blocking Model

Our response-blocking S-81 model is an approximate mean value analysis (MVA) model that estimates system performance when responses from memory modules and caches are returned in order, but there is no limit on the number of outstanding requests. Priority scheduling of bus operations and the spawning of asynchronous write requests are included in this model. The response-blocking model establishes the fundamental equations for the lower level submodel of the complete hierarchical S-81 model in Section IV. The term "response-blocking" refers to the fact that a memory module or cache will block when ready to return a response to a processor read request if an earlier processor request is still outstanding.

A. Overview of the Response-Blocking Model Approximations

The spawning of asynchronous write requests is modeled using the MVA approximations developed in [9]; whereas priority scheduling at the bus is based on the approximations developed in [4]. The read response ordering requirement, however, requires new heuristic extensions to the MVA equations.

Analysis of the S-81 system timing information reveals several reasonable assumptions that can be made to reduce the number of cases of response-blocking that are modeled. In particular, the remote cache read latency is significantly larger than the shared memory latency for a read operation,[2] whereas the memory read latency is approximately equal to the bus access time for a read request. These facts, together with the priority scheduling of cache and memory responses on the bus, imply that 1) a read access to one shared memory module rarely completes out of order with read accesses to the other shared memory module, and 2) a read access to a cache rarely completes before a previously issued read access to a shared memory module. Also, since the queueing of bus read requests at remote caches is negligible (as argued in Section II), two remote cache accesses almost always occur in the order they were issued. However, a memory response to a read request will block on the bus if the read request was issued soon after another read request that required a remote cache access. Thus, this is the only case of response blocking that we model. Although this greatly simplifies the calculations, the general approach we use could be extended to other cases of response blocking for other bus design parameters.

For a read request to a shared memory module, we calculate the probability that the response is delayed by a cache response to a prior request, and the conditional mean time the response is blocked given that blocking occurs. These quantities must be accurately estimated since they impact the memory blocking time as well as the delay until the processor receives a response. We will discuss the estimation of these quantities as we present the MVA equations in Section III-B.

[2]The precise values of these quantities is proprietary information that we are not authorized to reveal. In Section V we experiment with a range of values for these parameters.

TABLE I
NOTATION FOR RESPONSE-BLOCKING MODEL

Request Classes

- $j \in \{iv, rw, r, cacherep, memrp\}$ denotes the request class at the *bus*. iv, rw, and r are the invalidation, read/write and read requests from processors, respectively. $cacherep$ and $memrp$ are the cache and memory responses, respectively
- $k \in \{\hat{r}, \hat{w}\}$ denotes the request class at *memory*. $\hat{r}$ is the memory read generated by an r or rw bus request and $\hat{w}$ is a memory write generated by an rw bus request.

Input parameters

- τ denotes the mean processor interrequest time for the bus.
- f_j denotes the probability that a processor request is of type $j \in \{iv, rw, r\}$
- f_{ca} denotes the probability that a read request is responded by a cache.
- $t_{bus,j}$ denotes the bus access time for request class j.
- $t_{bus,rp}$ denotes the bus access time for a memory or cache response.
- $t_{mem,k}$ denotes the memory access time for request class k
- t_{ca} denotes the cache read time.

Model parameters

- $R(N)$ denotes the mean processor cycle time (execution plus bus access plus possible remote memory access) in a system with N processors.
- $R_j(N)$ denotes the mean time for a request $j \in \{iv, rw, r\}$ from the time it is generated until the request completes (iv) or until a response returns to the processor (r, rw)
- $W_{bus,j}(N)$ denotes the mean bus waiting time, not including service, for class j requests.
- $W_{mem,j}(N)$ denotes the mean waiting time before receiving service at the memory for class j requests.
- $Q_{bus,j}(N)$ and $U_{bus,j}(N)$ denote the mean queue length of class j requests at the bus, and the bus utilization by class j requests, respectively.
- $Q_{ca}(N)$ denotes the mean number of read requests at the caches.
- $P_{ca}(N)$ denotes the probability that a response from memory will be blocked by a response from a cache.
- $\Lambda_{bus,j}(N)$ denotes the bus throughput for request class j.
- $D_{mem,\hat{r}}(N)$ denotes the mean time spent in memory for a class $\hat{r}$ request, including the queueing, service and blocked time waiting for the response to return in order

B. The Response-Blocking MVA Equations

We present the MVA equations in a top-down fashion, using the notation given in Table I. The processor cycle time is presented first, followed by the equations for the components in the cycle time.

The mean processor cycle time is the sum of the execution time τ, and the weighted average of the mean response times, $R_j(N)$, for the three types of processor requests. Thus,

$$R(N) = \tau + \sum_{j=iv,r,rw} f_j R_j(N). \tag{1}$$

The mean response time for an invalidation request is simply:

$$R_{iv}(N) = W_{bus,iv}(N) + t_{bus,iv}. \tag{2}$$

The mean time for a read request to complete, $R_r(N)$, is the sum of the time the request waits for and uses the bus, plus the time until the read response returns. The read request has a probability f_{ca} of accessing another cache, and a probability $(1 - f_{ca})$ of accessing shared memory. $D_{mem,r}(N)$ includes the queueing and service time in memory, the mean blocking time due to response ordering, and the mean time the response waits for the bus. The bus access time for a cache or memory response is $t_{bus,rp}$. Thus,

$$\begin{aligned} R_r(N) =& W_{bus,r}(N) + t_{bus,r} + f_{ca} \times [t_{cs} + W_{bus,cacherp}(N)] \\ &+ (1 - f_{ca}) \times D_{mem,r}(N) + t_{bus,rp}. \end{aligned} \tag{3}$$

For a read followed by write request (rw), the cache or memory read operation occurs in parallel with the write operation on the bus, but is guaranteed to take at least as long, due to the times for these operations. We ignore the fact that responses to rw requests are more likely to find the bus busy (due to other requests that queued while the write operation occupied the bus). Thus,

$$R_{rw}(N) = R_r(N). \tag{4}$$

Mean Bus Waiting Time for Processor Requests ($W_{bus,iv}$, $W_{bus,r}$, $W_{bus,rw}$): Processor requests are served FCFS at the bus, whereas responses have higher priority. The bus waiting time for an iv, r or rw request is thus equal to the sum of the mean service times for requests and responses found in the bus queue when a request arrives, plus the service times for the responses that arrive during the request's queueing time. Since the bus service time is deterministic for all requests and responses, the mean residual service time of the operation in progress when a request arrives is one-half its total service time. Thus, the time to serve the requests and responses found in the queue by an arriving request is

$$\begin{aligned} \kappa = \sum_{j=iv,r,rw,memrp,cacherp} & [Q_{bus,j}(N-1) - U_{bus,j}(N-1)] \\ & \times t_{bus,j} + U_{bus,j}(N-1) \times t_{bus,j}/2 \end{aligned}$$

Applying the MVA approximation for a nonpreemptive priority service center in [4], new cache and memory responses arrive during the request queueing time at rates approximated by the respective bus throughputs, $\Lambda_{bus,cacherp}(N-1)$ and $\Lambda_{bus,memrp}(N-1)$. Each of these responses uses the bus for $t_{bus,rp}$ time, causing an additional request waiting time equal to $W_{bus,r}(N-1) \times [\Lambda_{bus,cacherp}(N-1) + \Lambda_{bus,memrp}(N-1)] \times t_{bus,rp} = W_{bus,r}(N) \times [U_{bus,cacherp}(N-1) + U_{bus,memrp}(N-1)]$. The minimum waiting time for requests is one bus cycle, since a request that arrives to an idle bus must wait for a bus arbitration cycle. Thus,

$$\begin{aligned} W_{bus,iv}(N) &= W_{bus,r}(N) = W_{bus,rw}(N) \\ &= \max\left(1.0, \frac{\kappa}{1 - [U_{bus,cacherp}(N-1) + U_{bus,memrp}(N-1)]}\right). \end{aligned} \tag{5}$$

Mean Bus Waiting Time for Responses ($W_{bus,memrp}$, $W_{bus,cacherp}$): The bus waiting time of a memory response depends on whether the response blocks due to the requirement for in-order delivery of responses to read requests. A *blocked* memory response waits in the memory until the blocking cache access is complete and then queues behind the cache response at the bus (see (8) below). The waiting time for the bus is thus equal to the time the cache response queues for and uses the

bus. A memory response that doesn't block will find very few responses ahead of it in the bus queue, since $t_{bus,r}$, $t_{mem,\hat{r}}$ and $t_{bus,rp}$ are all approximately equal. We thus assume these memory responses wait only for the operation currently on the bus to complete before they receive permission to use the bus,

$$W_{bus,memrp}(N) = P_{ca}(N) \times (W_{bus,cacherp}(N) + t_{bus,rp}) + (1 - P_{ca}(N)) \times \sum_j U_{bus,j}(N-1) \times t_{bus,j}/2. \quad (6)$$

A cache response, as for the nonblocked memory response, only waits for the bus operation in progress to complete before gaining access to the bus,

$$W_{bus,cacherp}(N) = \sum_{j=iv,r,rw,memrp,cacherp} U_{bus,j}(N-1) \times t_{bus,j}/2. \quad (7)$$

We have assumed that the probability that a response from one memory module is blocked by a response from the other memory module is negligible. Our estimation of the blocking time in $D_{mem,r}(N)$ may thus be slightly optimistic.

Mean Delay at Memory ($D_{mem,r}$): The mean memory delay $D_{mem,r}(N)$ for a read request, including the time the response waits for the bus, is a weighted average of the delay when the memory response is not blocked and the delay when it is blocked. The delay of a blocked memory read is estimated as the remaining service time in the cache ($t_{ca}/2$) plus the time the cache response waits for and uses the bus, assuming the queueing and service time at the memory is less than or equal to this value. The delay of a nonblocked memory read is the sum of its queueing and service times in memory, and the time the response waits for the bus.

$$D_{mem,\hat{r}}(N) = P_{ca}(N) \times (t_{ca}/2 + W_{bus,cacherp}(N) + t_{bus,rp}) + (1 - P_{ca}(N)) \times \left(W_{mem}(N) + t_{mem,\hat{r}} + \sum_{j=iv,r,rw,memrp,cacherp} U_{bus,j}(N-1) \times t_{bus,j}/2 \right). \quad (8)$$

Note that the mean bus waiting time for the memory response is included in the memory delay and is not added again in the equations for mean cycle time [(3) and (4)].

Probability that a Memory Response Blocks due to Response Ordering (P_{ca}): The probability that a memory response blocks due to the requirement for in-order delivery of responses is estimated at the probability that the request finds at least one cache read in progress when the read request arrives to the memory. The mean number of remote cache read requests seen by the arriving memory read request is approximately $Q_{ca}(N-1)$. The utilization of each cache is equal to $Q_{ca}(N-1)/N$. The probability that there is at least one cache read access in progress when a read request arrives to memory is therefore equal to $1 - [1 - (Q_{ca}(N-1)/N)]^N$. We further assume, and have observed that $Q_{ca}(N-1)$ is very small and hence this probability can be approximated by $Q_{ca}(N-1)$.[3] The probability $P_{ca}(N)$ is thus,

$$P_{ca}(N) = \min(Q_{ca}(N-1), 1.0). \quad (9)$$

Mean Memory Waiting Time (W_{mem}): A request arriving at the memory may find read and/or write requests already in the memory queue. The spawning of asynchronous write requests from rw processor requests is modeled by an independent stream of write requests arriving at a rate of $\Lambda_{bus,rw}(N)$, as proposed in [9].

The mean memory waiting time $W_{mem}(N)$ depends on the mean memory holding time for the read and write requests, denoted by $S_{mem,\hat{r}}(N)$ and $S_{mem,\hat{w}}$, respectively. Since a write request is not affected by the bus response ordering,

$$S_{mem,\hat{w}} = t_{mem,\hat{w}}. \quad (10a)$$

The mean memory holding time for read requests includes mean time the response is blocked due to in-order delivery of responses, plus the time spent waiting for the bus. Thus, $S_{mem,\hat{r}}(N)$ is the mean delay in the memory, $D_{mem,r}(N)$ [(8)], less the waiting time $W_{mem}(N)$ at the memory,

$$\begin{aligned} S_{mem,\hat{r}}(N) &= P_{ca}(N) \times (t_{ca}/2 + W_{bus,cacherp}(N) + t_{bus,rp} - W_{mem}(N)) \\ &\quad + (1 - P_{ca}(N)) \times (t_{mem,\hat{r}} + \sum_j U_{bus,j}(N-1) \times t_{bus,j}/2) \\ &= P_{ca}(N) \times (t_{ca}/2) + (1 - P_{ca}(N)) \times t_{mem,\hat{r}} \\ &\quad + W_{bus,memrp}(N) - P_{ca}(N) \times W_{mem}(N) \end{aligned} \quad (10b)$$

where $W_{bus,memrp}(N)$ was derived in (6).

Requests in memory are served FCFS. Assuming that $S_{mem,\hat{r}}$ is approximately deterministic, the memory waiting time is given by,

$$W_{mem}(N) = \sum_{k=\hat{r},\hat{w}} \{ [Q_{mem,k}(N-1) - U_{mem,k}(N-1)] \times S_{mem,k}(N) + U_{mem,k}(N-1) \times S_{mem,k}(N)/2 \}.$$

Since the fourth term in $S_{mem,\hat{r}}(N)$ is $P_{ca}(N) \times W_{mem}(N)$, we simplify the above equation by defining $S'_{mem,\hat{r}}$ to be the first three terms in equation (10b), and $S'_{mem,\hat{w}} = S_{mem,\hat{w}}$. The memory waiting time can thus be written as

[3]Note that this approximation is another source of optimism in our equations. However, Section V will show that the model is quite accurate for a set of measured model parameters. The more precise equation can be used if needed.

$$W_{mem}(N) = \sum_{k=\hat{r},\hat{w}} \Big\{ [Q_{mem,k}(N-1) - U_{mem,k}(N-1)] \times S'_{mem,k}(N) + U_{mem,k}(N-1) \times S'_{mem,k}(N)/2 \Big\} / \left(1 + P_{ca}(N) \times \left(Q_{mem,r}(N-1) - \frac{U_{mem,r}(N)}{2}\right)\right). \quad (11)$$

Throughputs, Utilizations and Queue Lengths: Using Little's Result, the overall throughput of the system is

$$\Lambda(N) = \frac{N}{R(N)}. \quad (12)$$

The throughput for processor requests, $j \in \{iv, r, rw\}$, is

$$\Lambda_{bus,j}(N) = f_j \times \Lambda(N), \quad (13)$$

and the throughput of cache and memory responses is:

$$\Lambda_{bus,cacherp}(N) = f_{ca} \times (\Lambda_{bus,r}(N) + \Lambda_{bus,rw}(N)), \quad (14)$$

and

$$\Lambda_{bus,memrp}(N) = (1 - f_{ca}) \times (\Lambda_{bus,r}(N) + \Lambda_{bus,rw}(N)). \quad (15)$$

Bus utilization and the mean bus queue length are, respectively,

$$U_{bus,j}(N) = \Lambda_{bus,j}(N) \times t_{bus,j}, \text{ and} \quad (16)$$

$$Q_{bus,j}(N) = \Lambda_{bus,j}(N) \times (t_{bus,j} + W_{bus,j}(N)). \quad (17)$$

The utilizations and queue lengths of memory requests are estimated from the respective throughputs. Since there are two memory modules, each accessed with equal probability,

$$U_{mem,\hat{r}}(N) = \Lambda_{bus,memrp}(N) \times S_{mem,\hat{r}}/2, \quad (18)$$

$$U_{mem,\hat{w}}(N) = \Lambda_{bus,rw}(N) \times t_{mem,\hat{w}}/2, \quad (19)$$

$$Q_{mem,\hat{r}}(N) = \Lambda_{bus,memrp}(N) \times (S_{mem,\hat{r}} + W_{mem}(N))/2, \quad (20)$$

and

$$Q_{mem,\hat{w}}(N) = \Lambda_{bus,rw}(N) \times (t_{mem,\hat{w}} + W_{mem}(N))/2. \quad (21)$$

Note that $U_{mem,r}(N)$ here is the probability that a memory is held by (not necessarily serving) a read request.

Similarly, the mean queue length at the cache is

$$Q_{ca}(N) = \Lambda_{bus,cacherp}(N) \times t_{ca}. \quad (22)$$

The model can be solved either by stepping from 1 to the system population (N), or by iteration using Schweitzer's approximation [22].

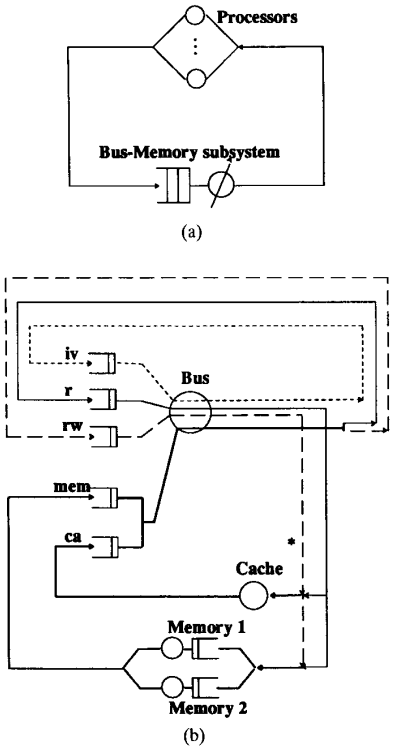

Fig. 2. (a) Higher level model: processor blocking (Markov chain analysis). (b) Lower level model: bus and memory subsystem (mean value analysis).

IV. The Complete (Full-Blocking) MBI Model

In this section we develop a two-level hierarchical model that extends the model of the previous section to include bounds on the numbers of outstanding processor requests that can be issued on the S-81 bus. The hierarchical model thus includes all of the characteristics of the Symmetry S-81 bus design outlined in Section II.

The higher level model [Fig. 2(a)] consists of the processors and a flow equivalent service center (FESC) representing the bus and memory subsystem. A processor request that blocks due to the bounds on outstanding read and write requests can be viewed as queueing at the FESC, waiting for a request of the appropriate type to depart the bus and memory subsystem. The lower level model [Fig. 2(b)] consists only of the bus and memory subsystem, and is similar to the response-blocking MVA model developed in Section III.

The separate limits on outstanding read and write processor requests requires a multiclass model and global balance solution at the higher level. The rate at which a request completes in the FESC is a function of its request type and the number and type of the other outstanding requests in the bus and memory subsystem. We solve the lower level submodel for all possible numbers of read, write, and invalidation requests, such that the bounds on the outstanding requests are not exceeded. For each of these feasible subsystem states, the mean throughput for each request type is estimated using approximate MVA equations similar to those developed in Section III. To solve the higher level submodel, we develop a continuous-parameter

Markov chain, where each state represents the number of each type of requests in each queue. The transition rates of the Markov chain are calculated from model input parameters and the calculated throughputs of the FESC.

We first discuss the details of the lower level submodel, followed by the details of the higher level Markov model.

A. The Bus and Memory Subsystem Model

The queueing network for this submodel [Fig. 2(b)] is obtained from the network in Fig. 1 by "shorting" the processors [13], so that requests now circulate only in the subnetwork. While the processor requests are in a single chain in the queueing network for the response-blocking model (Fig. 1), the requests circulate in three separate chains in Fig. 2(b). This is the principal distinction between the two models. The requests in the r (or rw) chain retain their chain identity as they visit the cache, memory, and bus response queues; this is not shown explicitly in the figure in order to simplify the diagram. Three chains are required in order to estimate the steady state throughput of each request type for a particular network population vector.

Let $N_{r,limit}$ and $N_{w,limit}$ denote the specified upper bounds on the number of outstanding read and read/write requests, respectively. The submodel is solved for each population vector (n_{iv}, n_r, n_{rw}) such that $n_{rw} \leq N_{w,limit}, n_{rw} + n_r \leq N_{r,limit}$, and $n_{iv} + n_{rw} + n_r \leq N$, where N is the population of the higher level model (i.e., N is the number of processors).

The MVA equations for the bus and memory subsystem for the complete MBI model are based on the same assumptions and reasoning about system behavior as in the response-blocking model. The notation and equations are given in the Appendix. The equation numbers for this model have a suffix B, and otherwise are numbered so that corresponding equations in the response-blocking model are easily identified. Note that the separation of the submodel population into three chains implies that the mean bus waiting time [(5B), (6B), (7B)], the mean memory waiting time [(11B)], and the mean memory delay [(8B)] are chain-dependent. That is, removing a customer from a specific chain removes only the service demands of the customer in that particular chain. This in turn implies that $R_r(\vec{N})$ and $R_{rw}(\vec{N})$ are not equal. Similarly, the mean bus and memory queue lengths, are affected by the chain dependent waiting times.

The equations in the Appendix are solved by starting from a total subsystem population (N_S) equal to 1, and successively increasing the population by one until the maximum population is reached, solving for the chain throughputs for each feasible distribution of the total population. The MVA equations can be solved for all populations because the limits, $N_{r,limit}$ and $N_{w,limit}$, which determine the maximum populations in the r and rw chains are typically small.

B. The Higher Level Submodel

The higher level submodel of the complete MBI model is a continuous time Markov chain that represents the state transitions in the queueing network of Fig. 2(a). A Markov chain is required by the separate bounds for different types of outstanding bus requests. The rates of completion of jobs in the FESC, which are dependent on the nonblocked subsystem population, are estimated from the lower level bus and memory submodel presented above.

Below we define the states and the state transition rates of the Markov chain. The calculation of processor cycle time and the bus utilization from the stationary state probabilities of the Markov chain, is discussed in Section IV-C.

1) State Definition: Since the completion rates of requests in the FESC depends on the number of active (i.e., nonblocked) requests in the bus and memory subsystem, the states of the Markov chain must include a specification of the blocked queue. Recall that the blocked queue at the FESC behaves as follows. A read or write request blocks when the number of outstanding requests of the same type (i.e., the number already active in the bus and memory subnetwork) equals the specified maximum value. Since a blocked read or write request blocks all later processor requests from accessing the bus, an invalidation request arriving at the bus blocks if the number of requests already blocked is greater than zero. When a read or write request completes in the subsystem, the read or read/write request at the head of the blocked queue may unblock, in which case subsequent blocked requests (in round-robin order, approximated by FCFS) are allowed to enter the subsystem until either the blocked queue empties or another request blocks.

The state of the Markov chain must contain sufficient information to predict 1) whether a processor request arriving at the FESC will block, and 2) how many blocked processor requests can enter the subsystem when a request completes service in the FESC. A sufficient representation includes the number and type of requests in the subsystem, as well as the number, the type and the (FCFS) *ordering* of the blocked processor requests. However, the Markov Chain can be simplified if we only represent the number of blocked processor requests (n_{blk}), and then compute the state probabilities for the blocked queue as needed from the number of blocked requests, the subsystem population vector, and the frequencies of the various request types. This state aggregation is approximate, but gives sufficiently accurate results in our experimental validations. The model is also simplified if we assume that a write request completes at approximately the same time as the read request generated by the same rw processor request (i.e., the number of outstanding writes in the subsystem is approximately equal to n_{rw}). This latter assumption is reasonable since $t_{bus,rp} \approx t_{mem,\hat{w}}$. With these assumptions, four parameters are sufficient to describe the state of the system:

$$\vec{S} = (n_{blk}; n_{iv}, n_{rw}, n_r)$$

where $|\vec{S}| = \sum_{i=blk,r,rw,iv} n_i \leq N$, $n_{rw} \leq N_{w,limit}$, and $(n_r + n_{rw}) \leq N_{r,limit}$. The number of processors executing locally can be obtained from this state definition and is equal to $(N - |\vec{S}|)$.

2) State Transition Rates: The Markov Chain model assumes that processor interrequest times and FESC service

TABLE II
HIGH-LEVEL MODEL STATE TRANSITIONS. (a) DEPARTURES FROM THE PROCESSOR QUEUE $\vec{S} = (n_{blk}; n_{iv}, n_{rw}, n_r)$, $N - |\vec{S}| > 0$. (b) DEPARTURES FROM THE FESC $\vec{S} = (n_{blk}; n_{iv}, n_{rw}, n_r)$, $n_{iv} + n_{rw} + n_r > 0$

Next State	Transition Rate	Restrictions
$\vec{S} + \vec{e}_{iv}$	$f_{iv}\mu$	$n_{blk} = 0$
$\vec{S} + \vec{e}_r$	$f_r\mu$	$n_{blk} = 0, n_r + n_{rw} < N_{r,limit}$
$\vec{S} + \vec{e}_{rw}$	$f_{rw}\mu$	$n_{blk} = 0, n_r + n_{rw} < N_{r,limit}$, $n_{rw} \leq N_{r,limit}$
$\vec{S} + \vec{e}_{blk}$	μ $(f_r + f_w)\mu$ $f_{rw}\mu$	$n_{blk} \geq 0$ $n_{blk} = 0, n_r + n_{rw} = N_{r,limit}$ $n_{blk} = 0, n_r + n_{rw} = N_{r,limit}$, $n_{rw} = N_{r,limit}$

(a)

Next State	Transition Rate	Restrictions
$\vec{S} - \vec{e}_{iv}$	$\Lambda_{iv}(\vec{S})$	$n_{iv} > 0$
$\vec{S} - \vec{e}_r$	$\Lambda_r(\vec{S})$	$n_{blk} = 0, n_r > 0$
$\vec{S} - \vec{e}_{rw}$	$\Lambda_{rw}(\vec{S})$	$n_{blk} = 0, n_{rw} > 0$
$\vec{S} - \vec{e}_d - (1+j)\vec{e}_{blk} + i\vec{e}_b + j\vec{e}_{iv}$	$\Lambda_d(\vec{S}) f_b^i f_{iv}^j (1 - f_{iv})^a$ $a = 0$ if $n_{blk} = 1 + j$ $a = 1$ otherwise	$b = $ 1st blocked job $d = $departed job $n_{blk} \geq 1 + j, j \geq 0$, $n_r + n_{rw} = N_{r,limit}$ $b \in \{r\}$ if $n_{rw} = N_{w,limit}$ and $d = r$ $b \in \{r, rw\}$ otherwise
$\vec{S} - (1+i+j)\vec{e}_{blk} + i\vec{e}_r + j\vec{e}_{iv}$	$\Lambda_{rw}(\vec{S}) X f_r^i f_{iv}^j (1 - f_{iv})^a$ $a = \min(1, n_{blk} - (1 + i + j))$ $X = $ state dependent factor	$n_{blk} \geq 1 + i + j, j \geq 0, n_{rw} = N_{w,limit}$ $n_r < N_{r,limit} - N_{w,limit}$ $N_{r,limit} - N_{w,limit} - n_r \geq i \geq 0$

(b)

times are exponentially distributed. Transition rates will be defined using the following notation:

- $\mu = 1/\tau$ denotes the rate of the exponential processor interrequest time,
- $\Lambda_j(\vec{S})$ denotes the FESC queue departure rate for class j in state $\vec{S}$ requests,
- $f_r{}' = f_r/(f_r + f_{rw})$,
- $f_{rw}{}' = f_{rw}/(f_r + f_{rw})$,
- $\vec{e}_c$ denotes a vector of the same form as $\vec{S}$, with $n_c = 1$, and $n_i = 0, i \neq c$.

Table II(a) and (b) gives the next state transition rates for an arbitrary state $\vec{S}$. In Table II(a), transitions for departures from the processor queue increment the count of the appropriate subsystem request type if there are no blocked requests and the new request will not exceed one of the upper bounds on outstanding bus requests. Otherwise the transition increments the count for the blocked request queue. The rates for these transitions are straightforward.

In Table II(b), the next state and transition rate for the completion of an iv request in the bus and memory subsystem are similarly straightforward, since no blocked jobs can be unblocked by this event.

Rates for transitions that are due to departures of r or rw requests from the FESC may require the calculation of various state probabilities of the blocked queue. In the simplest case, $n_{blk} = 0$, and the next state simply contains one less customer in the FESC. If $n_{blk} > 0$ and $(n_r + n_{rw}) = N_{r,limit}$, then the first blocked request is an r request with probability $f_r{}'$, and an rw request with probability $f_{rw}{}'$. The first blocked request, plus the next j consecutive invalidation requests in the blocked queue, will unblock. Each request in the blocked queue other than the first has probability f_{iv}, f_{rw}, and f_r of being a iv, rw or r request, respectively.

If $n_{blk} > 0$, $n_{rw} = N_{w,limit}$, and $n_r + n_{rw} < N_{r,limit}$, the first job in the blocked queue must be an rw request. In this case, a departing rw request from the subsystem is immediately replaced, and the next $i + j$ consecutive iv and/or r requests will also unblock, with the restriction that at most $i \leq N_{r,limit} - n_{rw} - n_r = N_{r,limit} - N_{w,limit} - n_r$ read (r) requests can unblock. The precise probabilities for these state transitions depends on the particular state of the blocked queue; i.e., all possible permutations of the consecutive requests in the blocked queue must be considered. We have used the symbol X in the transition rate to denote this state-dependent factor in the transition rates. Examples of the state transition rates for particular states are given in Table III.

C. Performance Estimates for the Complete Model

The performance metrics that we are interested in obtaining from the complete MBI model include the mean processor cycle time and the bus utilization. We compute these measures from the stationary state probability vector $(\boldsymbol{\pi})$ for the Markov chain, which we obtain using the Markov chain solution technique suggested in [23]. The mean number of executing processors, which we call processing power (P_{power}), and the utilization of each processor, which we call processor efficiency $(P_{efficiency})$ are then computed as follows:

$$P_{power}(N) = \sum_{\vec{S}} \pi(\vec{S}) \times (N - \vec{S}), \text{ and} \qquad (25B)$$

TABLE III
EXAMPLE STATE TRANSITION AND RATES.
(a) CURRENT STATE = (3; n_{iv}, 1, 2), (b) CURRENT STATE = (3; n_{iv}, 2, 0)

Next State	Transition Rate
(3; n_{iv}−1, 1, 2)	Λ_{iv}
(2; n_{iv}, 1, 2)	$\Lambda_{rw} f'_{rw}(1 - f_{iv})$
(1; n_{iv}+1, 1, 2)	$\Lambda_{rw} f'_{rw} f_{iv}(1 - f_{iv})$
(0; n_{iv}+2, 1, 2)	$\Lambda_{rw} f'_{rw} f^2_{iv}$
(2; n_{iv}, 0, 3)	$\Lambda_{rw} f'_{r}(1 - f_{iv})$
(1; n_{iv}+1, 0, 3)	$\Lambda_{rw} f'_{r} f_{iv}(1 - f_{iv})$
(0; n_{iv}+2, 0, 3)	$\Lambda_{rw} f'_{r} f^2_{iv}$
(2; n_{iv}, 2, 1)	$\Lambda_{r} f'_{rw}(1 - f_{iv})$
(1; n_{iv}+1, 2, 1)	$\Lambda_{rw} f'_{rw} f_{iv}(1 - f_{iv})$
(0; n_{iv}+2, 2, 1)	$\Lambda_{r} f'_{rw} f^2_{iv}$
(2; n_{iv}, 1, 2)	$\Lambda_{r} f'_{r}(1 - f_{iv})$
(1; n_{iv}+1, 1, 2)	$\Lambda_{r} f'_{r} f_{iv}(1 - f_{iv})$
(0; n_{iv}+2, 1, 2)	$\Lambda_{r} f'_{r} f^2_{iv}$

(a)

Next State	Transition Rate
(3; n_{iv}−1, 2, 0)	Λ_{iv}
(2; n_{iv}, 2, 0)	$\Lambda_{rw} f_{rw}$
(1; n_{iv}, 2, 1)	$\Lambda_{rw} f_{r}(1 - f_{iv})$
(0; n_{iv}+1, 2, 1)	$2\Lambda_{rw} f_{r} f_{iv}$
(1; n_{iv}+1, 2, 0)	$\Lambda_{rw} f_{iv} f_{rw}$
(0; n_{iv}+2, 2, 0)	$\Lambda_{rw} f^2_{iv}$

(b)

$$P_{efficiency}(N) = P_{power}/N. \tag{26B}$$

System throughput is calculated from the processor efficiency and the processor request rate $\mu = 1/\tau$,

$$\Lambda(N) = \mu \times P_{efficiency}(N). \tag{27B}$$

Finally, the mean processor cycle time is calculated from the throughput,

$$R(N) = N/\Lambda(N), \tag{28B}$$

and the bus utilization is obtained by multiplying the system throughput by the mean time that a processor uses the bus,

$$U_{bus}(N) = \Lambda(N) \times \left[f_r \times (t_{bus,r} + t_{bus,rp}) + f_{rw} \times (t_{bus,rw} + t_{bus,rp}) + f_{iv} \times t_{bus,iv}\right]. \tag{29B}$$

The estimated mean processor cycle time $R(N)$ for the complete model can be compared with the estimated mean cycle time for the response-blocking model [(1)], to study the performance impact of the bounds on outstanding bus requests.

V. MODEL VALIDATIONS

We validate the complete (or full-blocking, FB) model against measured performance of an actual 20-processor S-81 system for each of two particular parallel program workloads. Model input parameters were also determined by system measurement and/or system specifications for the S-81 bus. We find that the full-blocking (FB) model estimates of mean processor cycle time and bus utilization have at most 9% error, and are typically within 6% of the measured values. Response-blocking (RB) model estimates are also compared with measured performance and full-blocking model estimates. The comparison shows that for systems larger than 10 processors, it is important to model the bounds on the number of outstanding bus requests, as in the complete FB model.

To investigate the causes of the small inaccuracy in the FB model and to validate the RB model, we have developed a simulator of the S-81 system. This simulator precisely models the bus operation, and includes variable parameters for the bounds on the number of outstanding read and write bus requests. For each of the FB and RB models, we compare the predicted mean cycle time and bus utilization with simulation results. We find that the model predictions are always within 3% of the simulation estimates, indicating that the approximations in the model for features of the system that violate product form queueing network assumptions, are accurate for the workloads studied. As will be explained further below, we conclude that the inaccuracy of our analytic models is most likely due to our lack of knowledge of (possibly undocumented) features of the bus protocol not captured in the model.

A. Full-Blocking Model Validation Against System Measurement

The two parallel programs used to experimentally validate the models are called *Bicon* (parallel biconnectivity algorithm [24]) and *GE* (Gaussian Elimination with partial pivoting [5]). The model input parameter values measured for each of these programs as a function of the number of processors executing the parallel program, are given in Table IV. The input parameters, as well as the system performance metrics, were measured using a Northwest Instruments Microanalyst 2000 logic analyzer. Note that the probability that a read request is serviced by a cache ($(1 - f_{ca})$) is about 0.5 for the *GE* program, whereas this parameter varies between 0.1 and 0.3 for *bicon*. Thus, the performance of *GE* is more likely to be affected by the requirement for in-order delivery of read responses. The mean bus demand per processor request, which is dependent on the frequency and the service time for each request type, is approximately equal for *GE* and *Bicon*. However, the *Bicon* program has shorter mean interrequest times when the number of processors equal to or greater than 10, and hence has a higher rate of demand for the bus and memory in this case.

Table V shows the comparison of the predicted mean cycle time (R) for the FB and RB models with the measured values for the *Bicon* and *GE* programs. Similar comparison of the bus utilizations is given in Table VI. For both programs, when the number of processors is less than 10, the effect of limitation on the number of outstanding requests is small, and the results of both the response-blocking (RB) model and the full-blocking (FB) model are within 3–4% of the measured values. For 10 or more processors, the percent differences between the model results and the measured values increase. The difference for

TABLE IV
MEASURED WORKLOAD CHARACTERISTICS (a) *Bicon.* (b) *GE*

N	τ	f_r	f_{rw}	f_{iv}	f_{ca}
1	127.06	0.582	0.418	0.0000	0.0000
2	85.32	0.830	0.170	0.0004	0.1063
5	67.29	0.811	0.187	0.0019	0.3102
10	54.06	0.836	0.157	0.0062	0.2581
15	52.52	0.870	0.128	0.0014	0.2058
18	49.01	0.899	0.094	0.0073	0.1374

(a)

N	τ	f_r	f_{rw}	f_{iv}	f_{ca}
1	160.22	0.018	0.982	0.0000	0.0000
2	78.22	0.610	0.331	0.0590	0.5307
4	71.02	0.624	0.320	0.0565	0.4278
8	74.99	0.569	0.346	0.0856	0.4797
12	83.30	0.643	0.316	0.0405	0.5138
16	78.19	0.764	0.166	0.0704	0.5316

(a)

TABLE V
COMPARISON OF PREDICTED AND MEASURED MEAN CYCLE TIME. (a) *Bicon* (b) *GE*

N	Mean Cycle Time (R)				
	RB model	FB model	measured	%(RB-meas)	%(FB-meas)
2	93.28	93.29	93.31	0.0	0.0
5	77.57	77.68	78.40	−1.1	−0.9
10	65.45	66.74	71.18	−8.1	−6.2
15	66.96	70.38	76.16	−12.1	−7.6
18	66.77	71.56	76.19	−12.4	−6.1

(a)

N	Mean Cycle Time (R)				
	RB model	FB model	measured	%(RB-meas)	%(FB-meas)
2	89.10	89.11	87.84	1.4	1.4
4	82.16	81.64	84.12	−2.3	−2.9
8	86.33	86.95	89.95	−3.6	−2.9
12	96.23	97.78	102.35	−6.0	−4.5
16	92.10	95.50	103.96	−11.4	−8.1

(b)

the response-blocking model increases more rapidly, indicating that it is important to model the processor request bounds. The cycle time estimated from the full-blocking model can be as much as 30% (*GE* at $N = 32$) longer than that from the response-blocking model.

B. Model Validation Against Simulation

To investigate the causes of the discrepancies between the complete model results and the measured values, we simulated the bus design and compared the analytic, measured, and simulation results.

The simulator precisely mimics the operation of the bus, but represents cache accesses and the workload approximately. The features of the bus that are implemented precisely in the simulator include asynchronous execution of write requests at the shared memory, blocking in the memory due to in-order delivery of responses to read requests, priority scheduling of read responses on the bus, processor blocking due to bounds on the number of outstanding bus requests, and round-robin bus arbitration for the processor requests.

TABLE VI
COMPARISON OF PREDICTED AND MEASURED BUS UTILIZATION. (a) *Bicon.* (b) *GE*

N	Bus Utilization (U_{bus})				
	RB model	FB model	measured	%(RB-meas)	%(FB-meas)
2	0.075	0.075	0.075	0.0	0.0
5	0.029	0.229	0.227	0.9	0.9
10	0.523	0.518	0.486	7.6	6.6
15	0.757	0.721	0.666	13.7	8.3
18	0.881	0.822	0.772	14.1	6.5

(a)

N	Mean Cycle Time (R)				
	RB model	FB model	measured	%(RB-meas)	%(FB-meas)
2	0.087	0.087	0.088	−1.1	−1.1
4	0.188	0.188	0.183	2.7	2.7
8	0.358	0.356	0.345	3.8	3.2
12	0.482	0.475	0.454	6.2	4.6
16	0.583	0.562	0.517	−12.8	8.7

(b)

The deterministic bus access times, memory read and write latencies, and cache read latency are also precisely modeled in the simulator. However, cache accesses are assumed to incur no queueing delay in the simulator, as in the analytical models. We expect this to be reasonable since for the S-81 there are at most three read requests accessing the caches and shared memory at any given time, and for our workloads each request has 50% or higher probability of accessing main memory. Furthermore, with reasonable degree of uniformity in cache access probabilities and a bounded number of outstanding read requests, we would expect modeling error to decrease as system size increases if ignoring cache queueing delays were a source of significant error in the simulation or analytical models. This is not the case in Tables V and VI, nor in Table VII.

The simulator assumes exponential processor interrequest times and independent request probabilities as in the analytical models. The measured coefficient of variation of the processor interrequest times is less than 2. It is unlikely that any particular interrequest time distribution with such a low coefficient of variation would result in significantly different mean performance estimates than the exponential distribution. Thus, we don't expect these workload assumptions to introduce significant error in the simulation, and we expect the simulation model to agree very well with the measured system performance. We anticipated that the simulation would help us pinpoint more precisely which equations were the source of inaccuracy in our analytical models.

1) Full-Blocking Model Validation Against Simulation: Table VII compares the FB model estimates, simulation

TABLE VII
COMPARISON OF MEASURED, SIMULATION AND PREDICTED PERFORMANCE (FULL BLOCKING). (a) *Bicon* (b) *GE*

N	Mean Cycle Time (R)			Bus Utilization (U_{bus})		
	sim	%(sim-meas)	%(FB-sim)	sim	%(sim-meas)	%(FB-sim)
2	94.14	0.9	−0.9	0.074	−1.3	1.4
5	78.56	0.2	−1.1	0.226	−0.4	1.3
10	67.05	−5.8	−0.5	0.515	6.0	0.6
15	71.51	−6.1	−1.6	0.708	6.3	1.8
18	72.29	−5.1	−1.0	0.814	6.4	1.0

(a)

N	Mean Cycle Time (R)			Bus Utilization (U_{bus})		
	sim	%(sim-meas)	%(FB-sim)	sim	%(sim-meas)	%(FB-sim)
2	90.28	2.8	−1.3	0.086	−2.3	1.2
4	82.81	−1.6	−1.4	0.186	1.6	1.1
8	87.84	−1.9	−1.0	0.351	1.7	0.4
12	98.86	−3.4	−1.1	0.486	3.1	1.5
16	95.75	−7.9	−0.3	0.559	8.1	0.5

(b)

estimates, and actual system measurements of mean cycle time and bus utilization. Our first observation is that the FB model results are within 1–2% of the simulation results in all cases. Thus, the analytic approximations for in-order delivery of memory responses, priority scheduling of memory responses, asynchronous write requests, and bounded number of outstanding requests have not introduced significant error in the results and are not the cause of the discrepancy with measured performance. Second, in spite of the precise simulation of the bus design, the difference between the simulation results and the measured values is only slightly smaller than the difference between the FB model estimates and the measured values. As explained above, we believe the workload assumptions in the FB model and simulator are not likely to cause significant error in the reported measures. Thus, we hypothesize that the error in the FB model (as well as the simulator) is due to system details that we are not familiar with.[4] In any case, the discrepancies are small.

2) Response-Blocking Model Validation Against Simulation: Simulation results for the response-blocking model are obtained by setting the limits on the outstanding number of reads and writes in the simulator equal to the number of processors. Tables VIII(a) and VIII(b) show the RB model and simulation estimates of mean cycle time and bus utilization for the *Bicon* and *GE* program parameters. Results are shown for up to 32 processors, which is the maximum configuration of an S-81 system. Note that since we could not measure the input parameter values greater than 18 processors, we use the input parameter values for 18 (or 16) processors to produce the results for the larger configurations in the case of the *Bicon* (or *GE*) program.

The results in Table VIII show that the RB model agrees well with the simulation results for the two programs. Discrepancies are rarely and only marginally above 2%, Note that at 32 processors the bus utilization is close to 1 for both programs.

3) Validation of the Model Approximations for Response-Ordering: Recall that in both the FB and RB models, we approximately model the in-order delivery of responses to read requests on the bus, by analyzing the timing specifications for the S-81 system in which the remote cache read latency, t_{ca}, is approximately several times the memory read latency, $t_{mem,\hat{r}}$. To evaluate how the model might perform for other ratios of t_{ca} and $t_{mem,\hat{r}}$, we have compared the response-blocking model, in which out of order responses may occur more frequently, with simulation results for hypothetical cache latency values from $t_{ca} = t_{mem,\hat{r}}$ to $7.5 \times t_{mem,\hat{r}}$. For this range cache read latency values, the RB model estimates are within 7% of the simulation results.

VI. EVALUATING SYSTEM DESIGN TRADEOFFS

In this section, we illustrate the use of the FB and RB models for exploring system design issues and tradeoffs.

The timing specification for the S-81 system bus and memory, and the response priority bus scheduling are carefully designed so that if all bus read requests access the shared memory, responses will almost always return before the bounds on read and write requests are exceeded. The blocking of processor requests, as stated previously, occurs primarily due to the relatively long remote cache read latency. Furthermore, the response ordering requirement implies that memory response that would otherwise return quickly are delayed and further aggravate the processor blocking problem.

To evaluate the relative impact of the cache latencies and the outstanding request bounds on system performance, we compare the results of the FB and RB models for two cache latency values. We compare predicted mean cycle time in four cases: 1) full-blocking and S-81 t_{ca}, (FB, t_{ca}), 2) full-blocking and reduced t_{ca}, $(FB, t_{ca}/2)$, 3) response-blocking and S-81 t_{ca}, (RB, t_{ca}), 4) response-blocking and reduced

[4] Our understanding of the bus design comes from reading the bus circuit diagrams and from extensive discussions with system designers when our model was initially developed.

TABLE VIII
RESPONSE-BLOCKING MODEL VALIDATION RESULTS. (a) *Bicon*. (b) *GE*

N	Mean Cycle Time (R)			Bus Utilization (U_{bus})		
	RB model	sim	%(RB-sim)	RB model	sim	%(RB-sim)
2	93.28	94.14	−0.9	0.075	0.074	1.6
5	77.57	78.54	−1.0	0.229	0.226	1.4
10	65.45	66.51	−1.1	0.523	0.518	0.9
15	66.96	68.22	−1.3	0.757	0.742	2.1
18	66.77	67.04	−0.3	0.881	0.876	0.6
24	81.16	79.42	−2.2	0.966	0.987	2.1
32	106.21	104.66	+1.5	0.985	0.999	−1.5

(a)

N	Mean Cycle Time (R)			Bus Utilization (U_{bus})		
	RB model	sim	%(RB-sim)	RB model	sim	%(RB-sim)
2	89.10	90.28	−1.3	0.087	0.086	1.2
5	82.16	82.81	−0.8	0.188	0.186	1.1
8	86.33	87.71	−1.6	0.358	0.352	1.7
12	96.23	98.20	−2.0	0.482	0.472	2.1
16	92.10	93.21	−1.2	0.583	0.574	1.6
24	97.20	97.71	−0.5	0.829	0.822	0.7
32	111.62	109.85	+1.6	0.962	0.973	−1.1

(b)

t_{ca}, $(RB, t_{ca}/2)$. It is obvious that the best case will be the response-blocking system with reduced t_{ca} $(RB, t_{ca}/2)$, whereas the worst case will be system with blocking due to limits on the number of outstanding requests and the S-81 value of t_{ca} (FB, t_{ca}). However, the relative impact of varying each of the two parameters alone is difficult to predict. We view the particular parameter values for cache read latency and outstanding request limits in these experiments as values that bound an interesting design space.

To compare the four systems outlined above, we define a *baseline mean cycle time*, and compute the ratio of the system mean cycle time to the baseline cycle time. We call this ratio the *cycle time expansion factor*. The baseline mean cycle is defined to be the sum of the mean inter-request time, mean bus access time over all requests, and the mean remote memory or cache access time, assuming a remote cache read latency of $t_{ca}/2$. Since this baseline mean cycle time does not include any queueing or blocking of requests, the expansion factors will quantify how mean cycle time is inflated due to queueing, blocking and the increases in remote cache read latency. Figs. 3 and 4 show the expansion factors for the $Bicon$ and GE programs respectively.

Three observations can be made about the results in Fig. 3. First, the mean cycle time expansion factor for the (FB, t_{ca}) system is at most 20% greater than the expansion factor for the best case $(RB, t_{ca}/2)$ system. Second, of the two options for improving the performance of the (FB, t_{ca}) system, removing the bounds on the outstanding bus requests appears to be more beneficial. Finally, the 50% reduction in remote cache read latency has little impact on the response-blocking system. Since f_{ca} is low for the *Bicon* program, and since the system has no bounds on outstanding memory requests, memory queueing time dominates. In the full-blocking case, the reduction in remote cache read latency has a small but measurable impact on system performance because the opportunity for processor requests to block is reduced.

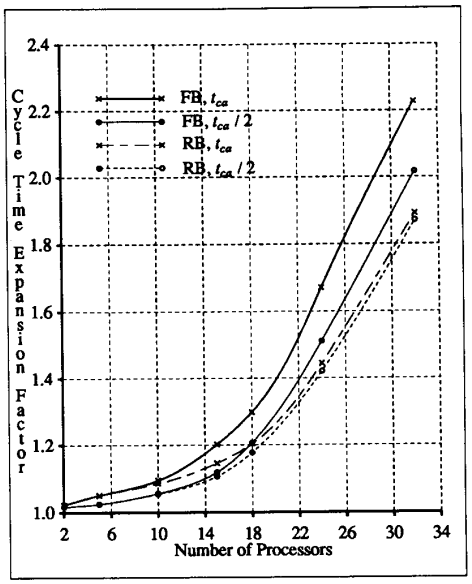

Fig. 3. Cycle time expansion factor for *Bicon*.

Fig. 4 shows that there are greater opportunities for improving the performance of the (FB, t_{ca}) system under the *GE* workload. Reducing the remote cache read latency by 50% and eliminating the bounds on outstanding processor requests appear to have similar potential for performance improvement.

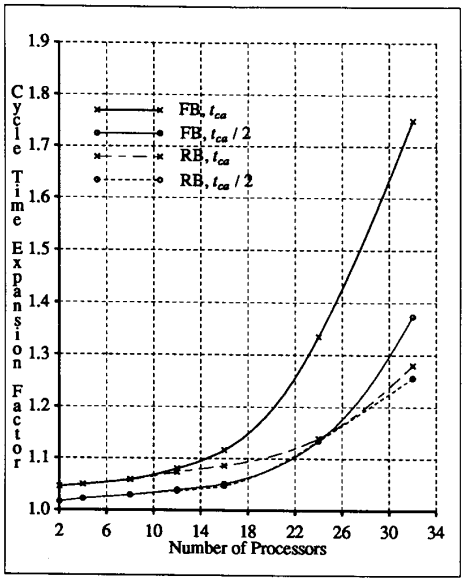

Fig. 4. Cycle time expansion factor for *GE*.

As in Fig. 3, the reduction in remote cache read latency results in little additional performance gain after bounds on the outstanding requests have been removed.

Considering the results of both Figs. 3 and 4, it appears that increasing the bounds on outstanding bus request has the potential greater benefit than reducing remote cache read latency.

VII. Conclusion

We have developed an efficient model to analyze the performance of the Sequent S-81 system bus design. The complex behavior of the S-81 bus included in the model distinguishes it from previous models of memory and bus interference in the literature. Furthermore, the upper bounds on the number of outstanding bus read and write requests, and the requirement for in-order delivery of responses to read requests, required new approximations in the analysis.

The performance estimates from the complete model were compared with measurements of an actual 20-processor system for two parallel program workloads with different memory access behavior. This also distinguishes this work from previous work which has used only simulation to validate the analytic models. Model estimates proved to be quite accurate (less than 9% error in all cases). Estimates from a model that ignores the bounds on outstanding requests were found to be less accurate in predicting measured system performance, indicating that it is important to model the bounds on the number of outstanding bus requests. Since performance estimates produced by a detailed bus simulator agreed more closely with our analytic model estimates than with measured system performance, we hypothesized that the small inaccuracy in the analytical model is due to a lack of knowledge of details of the S-81 implementation rather than to error in the analytical approximations.

The results in this paper provide evidence that complex system behavior can be accurately modeled using analytic queueing network solution techniques. In Section VI we illustrated the use of the models for assessing system design tradeoffs. The model can equally well be used to assess the sensitivity of system performance to workload parameter values. We believe that these techniques can be used during actual system design to facilitate rapid exploration of the design space. Workload parameter estimation is a difficult problem, as is choosing "representative" simulation workloads. One possible approach when designing a new system is to use measurement of a similar system, or a simulation model of the planned system, to determine approximate parameter values for a few real workloads, and then to use the analytical model to quickly and easily explore the impact of varying these parameter values.

Appendix

A. Additional Notation

A request in this submodel is distinguished by a chain identifier and a class identifier. The notation for chains and classes, and the parameters used in this model not given for the nonblocking model are given below.

Chains and Classes

- $c \in \{iv, rw, r\}$ denotes the *chain* that a request belongs to.
- $\vec{N}$ is the vector representing the population of each chain in the subsystem, $\vec{N} = (n_{iv}, n_{rw}, n_r)$.
- $(\vec{N} - \vec{e}_c)$ denotes a subsystem population with one few customer in chain c.
- N_c denotes the *maximum* population of chain c. n_c denotes a *feasible* population in chain c.
- $N_S = \sum_c n_c$ denotes the total population in the memory and bus subsystem, $N_S \leq \sum_c N_c$.
- $j \in \{iv, rw, r, cacherp, memrp\}$ denotes the *request class* at the *bus*. The subscript (c, j) represents a class j request in chain c.
- $k \in \{\hat{r}, \hat{w}\}$ denotes the *request class* at *memory*, and subscript (c, k) represents a class k request in chain c.

Note: the subscript j (or k) alone denotes to all class j requests (or class k requests) in all chains.

Model parameters

- $R_c(\vec{N})$ denotes the *mean cycle time* in the subsystem for chain c, when the subsystem population is $\vec{N}$. This time does not include the processor execution time.
- $D_{mem,(c,r)}(\vec{N})$ denotes the mean delay of a memory read in chain $c \in \{r, rw\}$.
- $W_{bus,(c,j)}(\vec{N})$ denotes the mean bus waiting time for a class j request in chain c.
- $W_{mem,c}(\vec{N})$ denotes the mean memory waiting time for job in chain $c \in \{r, rw\}$.
- $N_{w,limit}$ and $N_{r,limit}$ are the upper bounds on the number of outstanding write and read requests, respectively.

B. The Full-Blocking MVA Equations

Initial Values

$$Q_{bus,j}(\vec{0}) = 0,\ U_{bus,j}(\vec{0}) = 0, \qquad i \in \{iv, rw, r, cacherp, memrp\}$$
$$Q_{mem,(i,r)}(\vec{0}) = 0,\ U_{mem,(i,r)}(\vec{0}) = 0, \qquad i \in \{rw, r\}$$
$$Q_{mem,(rw,w)}(\vec{0}) = 0,\ U_{mem,(rw,w)}(\vec{0}) = 0$$
$$\Lambda_{bus,memrp}(\vec{0}) = 0,\ \Lambda_{bus,cacherp}(\vec{0}) = 0,\ Q_{ca}(\vec{0}) = 0.$$

Recursive Solution

For $N_S = 1$ *to* $\sum_c N_c$ **do**

For each state $\vec{N} = (n_{iv}, n_{rw}, n_r)$ where $\sum_c n_c = N_S$, $n_{rw} \le N_{w,limit}$, $n_r \le N_{r,limit}$ **do**

For each chain $c = iv, rw, r$ **do**

Bus waiting times:

$$W_{bus,(c,c)}(\vec{N}) = \max\Bigg(1.0, \sum_{j=i,rw,r,memrp,cacherp} \Big\{[Q_{bus,j}(\vec{N} - \vec{e}_c) - U_{bus,j}(\vec{N} - \vec{e}_c) \times t_{bus,j} + U_{bus,j}(\vec{N} - \vec{e}_c) \times t_{bus,j}/2\Big\} / \Big\{1 - [U_{bus,memrp}(\vec{N} - \vec{e}_c) + U_{bus,cacherp}(\vec{N} - \vec{e}_c)\Big\}\Bigg) \tag{5B}$$

$$W_{bus,(c,memrp)}(\vec{N}) = P_{ca,c}(\vec{N}) \times (W_{bus,(c,cacherp)}(\vec{N}) + t_{bus,rp}) + (1 - P_{ca,c}(\vec{N}) \times \sum_{j=iv,rw,r,memrp,cacherp} U_{bus,j}(\vec{N} - \vec{e}_c) \times t_{bus,j}/2 \tag{6B}$$

$$\text{where } P_{ca,c}(\vec{N}) = \min(Q_{ca}(\vec{N} - \vec{e}_c), 1.0) \tag{9B}$$

$$W_{bus,(c,cacherp)}(\vec{N}) = \sum_{j=iv,rw,r,memrp,cacherp} U_{bus,j}(\vec{N} - \vec{e}_c) \times t_{bus,j}/2. \tag{7A}$$

Memory delay and waiting time:

$$W_{mem,c}(\vec{N}) = \sum_k \sum_i \Big\{[Q_{mem,(i,k)}(\vec{N} - \vec{e}_c) - U_{mem,(i,k)}(\vec{N} - \vec{e}_c)\Big\} \times S'_{mem,(i,k)}(\vec{N}) + U_{mem,(i,k)}(\vec{N} - \vec{e}_c) \times S'_{mem,(i,k)}(\vec{N})/2\Big\} / \left(1 + P_{ca,c}(\vec{N}) \times \sum_{l=r,rw} \Big[Q_{mem,(l,\hat{r})}(\vec{N} - \vec{e}_c) - U_{mem,(l,\hat{r})}(\vec{N} - \vec{e}_c)/2\Big]\right) \tag{11B}$$

where $(k, i) = (\hat{r}, r), (\hat{r}, rw)$, and $(\hat{w}, rw)$ and

$$S'_{mem,(i,k)}(\vec{N}) = \begin{cases} P_{ca,c}(\vec{N}) \times (t_{ca}/2) + (1 - P_{ca,c}(\vec{N})) \times t_{mem,\hat{r}} + W_{bus,(i,memrp)}(\vec{N}) & k = \hat{r}, i = r, rw \\ t_{mem,\hat{w}} & k = \hat{w}, i = rw \end{cases}$$

$$D_{mem,(c,\hat{r})}(\vec{N}) = P_{ca,c}(\vec{N}) \times (t_{ca}/2 + W_{bus,(c,cacherp)}(\vec{N}) + t_{bus,rp}) + (1 - P_{ca,c}(\vec{N})) \times \left(W_{mem,c}(\vec{N}) + t_{mem,\hat{r}} \sum_j U_{bus,j}(\vec{N} - \vec{e}_c) \times t_{bus,j}/2\right) \tag{8B}$$

End(for each chain)

Cycle time for each chain,

$$R_{iv}(\vec{N}) = W_{bus,(iv,iv)}(\vec{N}) + t_{bus,iv} \tag{2B}$$

$$R_c(\vec{N}) = W_{bus,(c,c)}(\vec{N}) + t_{bus,\hat{r}} + f_{ca} \times [t_{ca} + W_{bus,(c,cacherp)}(\vec{N})] + (1 - f_{ca}) \times D_{mem,(c,\hat{r})}(\vec{N}) + t_{bus,rep}, \qquad c \in \{rw,r\} \tag{3B)\&(4B}$$

Throughputs, utilizations and queue lengths,

$$\Lambda_{bus,c}(\vec{N}) = n_c / R_c(\vec{N}) \qquad c \in \{iv, rw, r\} \tag{12B}$$

$$\Lambda_{bus,cacherp}(\vec{N}) = f_{ca} \times [\Lambda_{bus,r}(\vec{N}) + \Lambda_{bus,rw}(\vec{N})] \tag{14B}$$

$$\Lambda_{bus,memrp}(\vec{N}) = (1 - f_{ca}) \times [\Lambda_{bus,r}(\vec{N}) + \Lambda_{bus,rw}(\vec{N})] \tag{15B}$$

$$U_{bus,j}(\vec{N}) = \Lambda_j \times t_{bus,j} \tag{16B}$$

$$U_{bus}(\vec{N}) = \sum_{j=iv,rw,r,memrp,cacherp} U_{bus,j}$$

$$Q_{bus,j}(\vec{N}) = \Lambda_{bus,c}(\vec{N}) \times [t_{bus,c} + W_{bus,(c,c)}(\vec{N})], \qquad j \in \{iv, rw, r\} \tag{17B-1}$$

$$Q_{bus,carp}(\vec{N}) = f_{ca} \times \sum_{c=rw,r} [\Lambda_{bus,c}(\vec{N}) \times (t_{bus,rp} + W_{bus,(c,cacherp)}(\vec{N}))] \tag{17B-2}$$

$$Q_{bus,memrp}(\vec{N}) = (1 - f_{ca}) \times \sum_{c=rw,r} [\Lambda_{bus,c}(\vec{N}) \times (t_{bus,rp} + W_{bus,(c,memrp)}(\vec{N}))] \tag{17B-3}$$

$$U_{mem(c,\hat{r}}(\vec{N}) = (1 - f_{ca}) \times [\Lambda_{bus,c}(\vec{N}) \times S_{mem,(c,\hat{r})}(\vec{N})]/2 \tag{18B}$$

$$U_{mem,(rw,\hat{w})}(\vec{N}) = \Lambda_{bus,rw}(\vec{N}) \times t_{mem,\hat{w}}/2 \tag{19B}$$

$$Q_{mem,(c,\hat{r})}(\vec{N}) = (1 - f_{ca}) \times [\Lambda_{bus,c}(\vec{N}) \times S_{mem,(c,\hat{r})}(\vec{N}) + W_{mem,c}(\vec{N})]/2 \tag{20B}$$

$$Q_{mem,(rw,\hat{w})}(\vec{N}) = \Lambda_{bus,rw}(\vec{N}) \times [t_{mem,\hat{w}} + W_{mem,rw}(\vec{N})]/2 \tag{21B}$$

$$Q_{ca}(\vec{N}) = \Lambda_{bus,cacherp}(\vec{N}) \times t_{ca} \tag{22B}$$

End(for each state)
End(for each N_S)

ACKNOWLEDGEMENT

The authors wish to thank Sequent Computer Systems, Inc. for the loan of the Northwest Instruments Microanalyst 2000 Logic Analyzer. We also thank the Sequent Symmetry developers, particularly N. Wyse, T. Lovett, and B. Kasten, for comments of this manuscript and their assistance in deciphering the bus specifications and using the logic analyzer.

REFERENCES

[1] S. V. Adve, V. S. Adve, M. D. Hill, and M. K. Vernon, "Comparison of hardware and software cache coherence schemes," in *Proc. 18th Int. Symp. Comput. Architecture*, vol. 19, no. 3, May 1991, pp. 298–308.

[2] J. Archibald and J.-L. Baer, "Cache coherence protocols: Evaluation using a multiprocessor simulation model," *ACM Trans. Comput. Syst.*, vol. 4, no. 4, pp. 273–298, Nov. 1986.

[3] D. P. Bhandarkar, "Analysis of memory interference in multiprocessors," *IEEE Trans. Comput.*, vol. C-24, no. 9, pp. 897–908, Sept. 1975.

[4] R. M. Bryant, A. E. Krzesinski, M. S. Lakshmi, and K. M. Chandy, "The MVA priority approximation," *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 335–359, Nov. 1984.

[5] R. L. Burden and J. D. Faires, *Numerical Analysis*, 4th ed., PWS-Kent Publishing, 1989.

[6] M-C. Chiang and G. S. Sohi, "Evaluating design choices for shared bus multiprocessors in a throughput-oriented environment," *IEEE Trans. Comput.*, vol. 41, no. 3, pp. 297–317, Mar. 1992.

[7] M. Dubois and F. A. Briggs, "Effects of cache coherency in multiprocessors," *IEEE Trans. Comput.*, vol. C-31, no. 11, pp. 1083–1099, Nov. 1982.

[8] A. Goyal and T. Agerwala, "Performance analysis of future shared storage systems," *IBM J. Res. Develop.*, vol. 28, no. 1, pp. 95–108, Jan. 1984.

[9] P. Heidelberger and K. S. Trivedi, "Queueing network models for parallel processing with asynchronous tasks," *IEEE Trans. Comput.*, vol. C-31, pp. 1099–1108, Nov. 1982.

[10] C. M. Hoogendoorn, "A general model for memory interference in multiprocessors," *IEEE Trans. Comput.*, vol. C-26, pp. 990–1005, Oct. 1977.

[11] M. A. Holliday and M. K. Vernon, "Exact performance estimates for multiprocessor memory and bus interference," *IEEE Trans. Comput.*, vol. C-36, no. 1, pp. 76–85, Jan. 1987.

[12] R. Katz, S. Eggers, D. A. Wood, C. Perkins, and R. G. Sheldon, "Implementing a cache consistency protocol," in *Proc. 12th Int. Symp. Comput. Architecture*, Boston, MA, June 1985, pp. 276–283.

[13] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[14] T. Lovett and S. Thakkar, "The Symmetry multiprocessor system," in *Proc. Int. Conf. Parallel Processing*, Aug. 1988.

[15] M. A. Marsan, G. Balbo, and G. Conte, "Comparative performance analysis of single bus multiprocessor architectures," *IEEE Trans. Comput.*, vol. C-31, no. 12, pp. 1179–1191, Dec. 1982.

[16] M. A. Marsan and M. Gerla, "Markov models for multiple bus multiprocessorsystems," *IEEE Trans. Comput.*, vol. C-31, no. 3, pp. 239–248, Mar. 1982.

[17] M. A. Marsan, G. Balbo, G. Conte, and F. Gregoretti, "Modeling bus contention and memory interference in a multiprocessor system," *IEEE Trans. Comput.*, vol. C-32, no. 1, pp. 60–72, Jan. 1983.
[18] I. H. Onyuksel and K. B. Irani, "Markovian queueing network models for performance analysis of a single-bus multiprocessor system," *IEEE Trans. Comput.*, vol. 39, no. 7, pp. 975–980, July 1990.
[19] S. Owicki and A. Agarwal, "Evaluating the performance of software cache coherence," in *Proc. 3rd Int. Conf. Architectural Support for Programming Languages and Oper. Syst.*, Boston, MA, Apr. 1989, pp. 230–242.
[20] M. S. Papamarcos and J. H. Patel, "A low-overhead coherence solution for multiprocessors with private cache memories," in *Proc. 11th Int. Symp. Comput. Architecture*, Ann Arbor, MI, June 1984, pp. 348–354.
[21] R. B. Rau, "Interleaved memory bandwidth in a model of a multiprocessor computer system," *IEEE Trans. Comput.*, vol. C-28, no. 9, pp. 678–681, Sept. 1979.
[22] P. Schweitzer, "Approximate analysis of multiclass closed networks of queues," *J. ACM*, vol. 29, no. 2, Apr. 1981.
[23] W. J. Stewart, "A comparison of numerical techniques in Markov modeling," *Commun. ACM*, vol. 21, no. 2, Feb. 1978.
[24] R. Tarjan and U. Vishkin, "An efficient parallel biconnectivity algorithm," *SIAM J. Comput.*, vol. 14, no. 4, pp. 862–874, 1985.
[25] M. K. Vernon and M. A. Holliday, "Performance analysis of multiprocessor cache consistency protocols using generalized timed Petri nets," in *Proc. Performance '86 and ACM SIGMETRICS 1986 Joint Conf. Comput. Performance Modeling, Measurement, and Evaluation*, Raleigh, NC, May 1986, pp. 9–17.
[26] M. K. Vernon, E. D. Lazowska, and J. Zahorjan, "An accurate and efficient performance analysis technique for multiprocessor snooping cache-consistency protocols," in *Proc. 15th Int. Symp. Comput. Architecture*, Honolulu, HI, May 1988 pp. 308–315.
[27] Q. Yang, L. N. Bhuyan, and B.-C. Liu, "Analysis and comparison of cache coherence protocols for a packet-switched multiprocessor," *IEEE Trans. Comput.*, vol. 38, no. 8, pp. 1143–1153, Aug. 1989.

Thin-Fong Tsuei received the B.Eng. degree in electrical engineering from the University of Singapore, and the M.S. and Ph.D. degrees in computer science from the University of Wisconsin–Madison.

She is currently with the Advanced Technology Group at Apple Computer, Inc. Her research interests include performance modeling and analysis of computer systems, distributed systems, and multimedia workload characterization.

Dr. Tsuei is a member of the Association for Computing Machinery and the IEEE Computer Society.

Mary K. Vernon (S'82–M'82) received the B.S. degree with Departmental Honors in chemistry in 1975, and the M.S. and Ph.D. degrees in computer science in 1979 and 1983, from the University of California at Los Angeles.

In August 1983 she joined the Computer Science Department, University of Wisconsin–Madison, where she is currently an Associate Professor. Her research interests include techniques for performance modeling of parallel systems, and analysis of parallel architecture issues.

Dr. Vernon received an NSF Presidential Young Investigator Award in 1985 and an NSF Faculty Award for Women in Science and Engineering in 1991. She is a member of the Association for Computing Machinery and is currently an editor for the IEEE TRANSACTIONS ON SOFTWARE ENGINEERING.