Humoud, Humoud B.

A STUDY IN MEMORY INTERFERENCE MODELS

*The University of Michigan* PH.D. 1985

PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark __√__.

1. Glossy photographs or pages _____
2. Colored illustrations, paper or print _____
3. Photographs with dark background _____
4. Illustrations are poor copy _____
5. Pages with black marks, not original copy _____
6. Print shows through as there is text on both sides of page _____
7. Indistinct, broken or small print on several pages ✓
8. Print exceeds margin requirements _____
9. Tightly bound copy with print lost in spine _____
10. Computer printout pages with indistinct print _____
11. Page(s) _____ lacking when material received, and not available from school or author.
12. Page(s) _____ seem to be missing in numbering only as text follows.
13. Two pages numbered _____. Text follows.
14. Curling and wrinkled pages _____
15. Other_____

# A STUDY IN MEMORY INTERFERENCE MODELS

by
Humoud B. Humoud

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
( Electrical Engineering )
in The University of Michigan
1985

Doctoral Committee:

Associate Professor Trevor N. Mudge, Chairman
Professor Daniel E. Atkins
Professor John P. Hayes
Associate Professor Toby J. Teorey
Associate Professor Anthony C. Woo

To my parents and my wife

# ACKNOWLEDGMENT

I would like to express my gratitude and sincere appreciation to my thesis adviser, Professor Trevor Mudge, for his guidance, interest, encouragement, and friendship. His advice and constructive criticism have undoubtedly improved the quality of this work. I am also grateful to Professors Daniel Atkins, John Hayes, Toby Teorey, and Anthony Woo for serving on my thesis committee. Special thanks are due to Professor Daniel Atkins for his advice and assistance.

I would like to extend my gratitude to Miss Betty Cummings for her careful reading of the manuscript of this thesis. Her help and suggestions are very much appreciated.

The staff and the student members of the Computing Research Lab, past and present, made my stay in the East Engineering Building enjoyable and memorable for which I thank all of them. Their friendship and kindness were particularly helpful in cheering me up on the dismal days when nothing seemed to be going right.

I am indebted to University of Kuwait for their partial financial support for me during the course of my study. Their support is very much appreciated.

I would like to express my deep thanks and gratitude to my family. Their moral support and encouragement kept me going all this time. I would like to extend my sincere thanks to my second family, Al-Ali, to whom I am indebted for the rest of my life.

Finally, I would like to thank the person who shared with me every second of this study. I would like to thank my wife Khalidah. Her patience, understanding, and encouragement made this thesis possible. I dedicate this work to her and my parents.

## TABLE OF CONTENTS

## LIST OF FIGURES

Figure

## LIST OF TABLES

Table

# CHAPTER I

# INTRODUCTION

## 1.1. Statement of the Problem

The introduction of multiprocessor systems was prompted by two major developments: the decreasing cost of the hardware and the need for higher performance. Although a multiprocessor can supply the user with a number of desirable features, such as high performance and fault tolerance, it also introduces a number of new problems not associated with uniprocessors. These enigmas, such as the complexity of the design and the interference between the processing elements over common resources, are some of the major challenges in multiprocessor design. Therefore, designers of multiprocessor systems need some techniques to help them in multiprocessor system synthesis--a typical design synthesis procedure is depicted in Figure 1.1. A major step in the procedure is approximating the actual behavior of the multiprocessor system stochastically. In other words, the system designers will simulate approximately the actual behavior of a multiprocessor by employing a stochastic process. Thereafter, the stochastic description of the multiprocessor system will be analyzed using simulations or analytical models. Such analysis will produce quantitative measures of the multiprocessor system performance.

Simulation has two main advantages over the analytical model: first, the accuracy of the result and second, the simplicity of the derivation. On the other hand, simulation is typically more expensive and does not provide the designer with an insight into the underlying mechanism. Such insight into the problem should not be underestimated: a reasonably accurate analytical model provides the designer with quantitative information about the multiprocessor system as

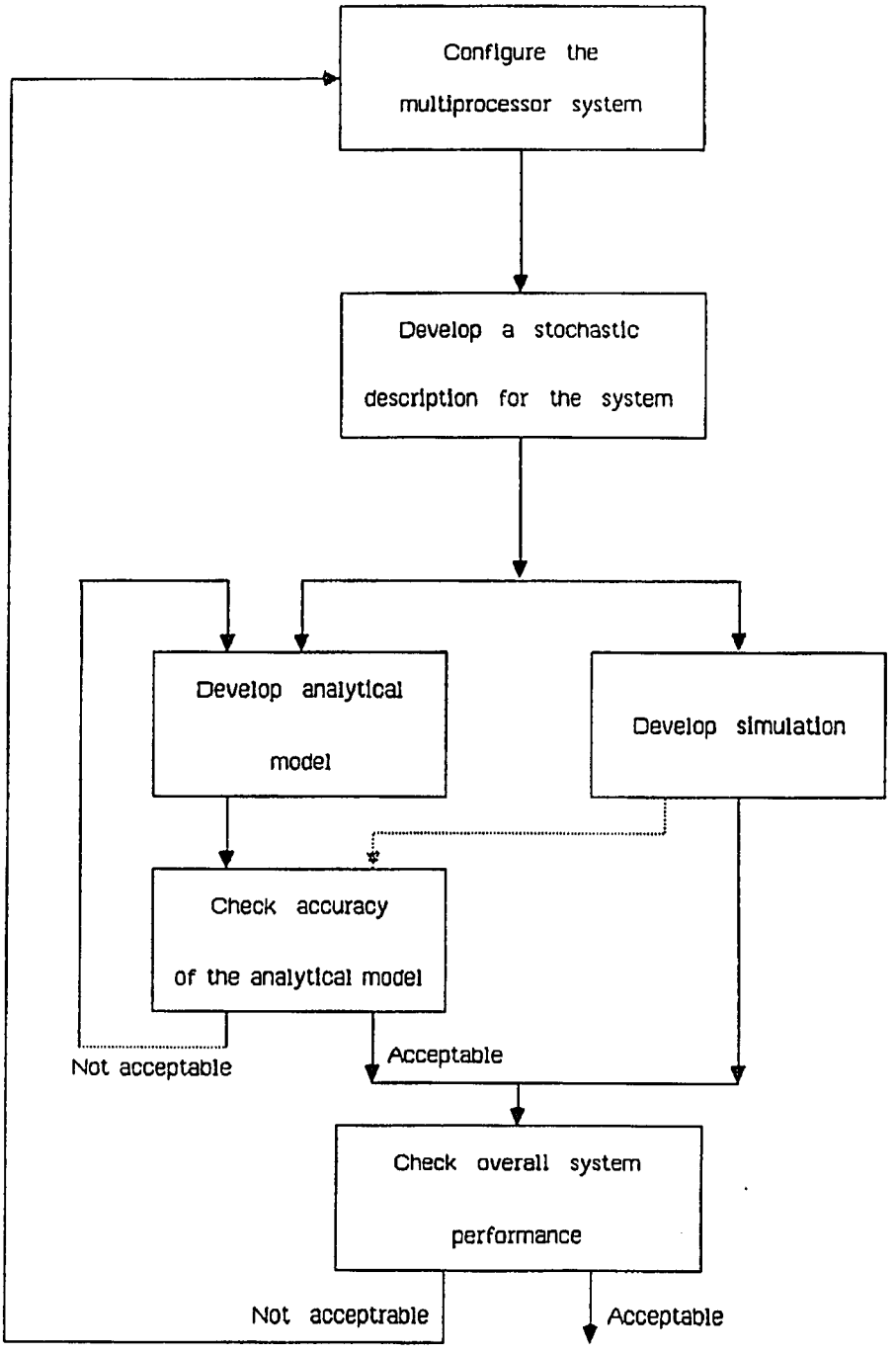



**Figure 1.1 A typical design synthesis procedure.**

well as a qualitative understanding that is usually unobtainable from simulation. For example, the analytical model will show the effect of the high moments of some random variable over the system performance while the simulation will not be able to determine such a relationship.

The accuracy of the analytical model could be improved at the expense of the model complexity. A very detailed and complex model might produce accurate results to the designer, but its cost might exceed the cost of simulation. Hence, a tradeoff between the model accuracy and the model cost must be adopted. The prime interest of this study is in the analytical modeling area. The problems associated with modeling of a tightly coupled multiprocessor system and the effects of the different system parameters on system performance are studied.

## 1.2. The Thesis

The overall performance of a multiprocessor system depends on the intrinsic performance characteristics of its constituents, as well as on interactions between them. This study focuses on a tightly coupled multiprocessor system, typified by a C.mmp multiprocessor system (see [WuB72] for details). The system of interest depicted in Figure 1.2 is a synchronous multiprocessor system which has a number of processing elements and memory modules. The processing elements share the memory modules through an interconnection network.

A processing element can simply be a CPU with or without its own cache memory or some special purpose processor, e.g., I/O channel or DMA controller. A processing element could issue an access request to any memory module and this request will be serviced by the destined memory module whenever it is possible. The actual behavior of the processing element can be approximated by a stochastic process that has been tested in a number of studies by comparing the behavior of the stochastic process to trace-driven simulation results (see [BaS76, Hoo77] for details). The processing elements cannot communicate with each other directly, although they communicate indirectly through the shared modules. Therefore, each processing element will behave autonomously by executing a program stored in its local memory.

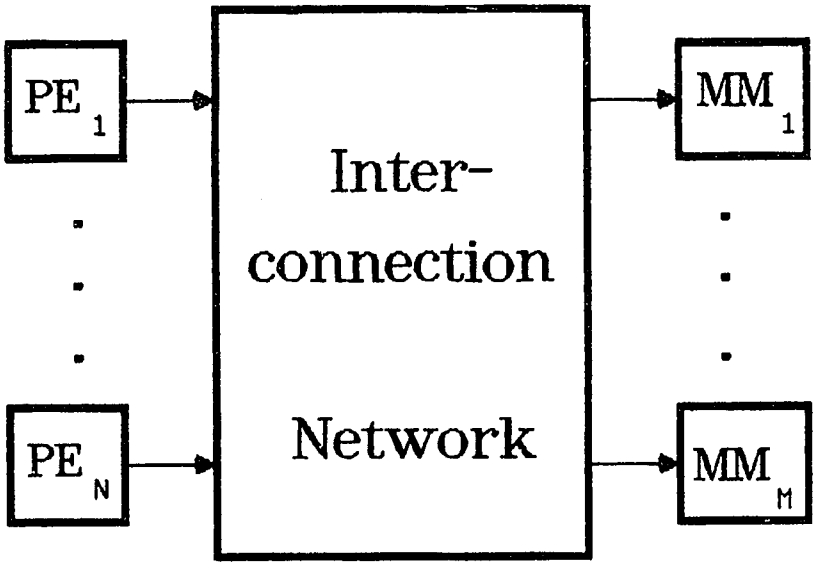



**Figure 1.2 The multiprocessor system.**

The memory modules are system resources that respond to requests for service generated by the processing elements. These modules could be: RAM modules, disk units, or even some special purpose processors that receive requests for service and respond with an acknowledge signal when done. The memory modules in the system, that could be identical or distinct, will behave independently of each other. The interconnection network is the media through which the processing elements will be connected to the memory modules. In this study two types of connection networks are considered: crossbar and multiple-bus connections.

The stochastic process that describes the different behaviors of the independent processing elements is usually intractable computationally. Furthermore, considering the behavior of a

processing element in the absence of the effects of the other processing elements is not accurate due to the coupling between the processing elements over the common resources. Therefore, the analytical model must adopt some further simplifying assumptions. These assumptions fulfill two main provisions: first, the resultant model is tractable and second, the resultant model will produce results "similar" to the original stochastic process.

This study is divided into three major parts. First, an outline of the stochastic process that approximates the actual behavior of the multiprocessor system is presented. Second, an analytical model is proposed to analyze the stochastic process. The analytical model supplies the system designer with a number of quantitative measures of the system performance that help in the synthesis procedure. Nevertheless, the model might adopt some further simplifying assumptions in order to be tractable. Third, the results of the model are compared to the results obtained through simulation of the stochastic process outlined earlier. The major objective of this comparison is to test the simplifying assumptions adopted in the analytical model. It is noted that the simulation was supplied by artificially generated address sequences.

The thesis of this dissertation is that the overall performance of the multiprocessor system can be deduced from the individual performances of the processing elements. Each processing element is considered independently from the other processing elements; nevertheless, the processing element coupling with other processing elements is factored into its individual performance description.

## 1.3. The Organization

The dissertation is organized as follows:

- In Chapter II the analytical models that have been published in the literature are presented. A taxonomy of these models is proposed.

- In Chapter III the stochastic process that approximates the actual behavior of the multiprocessor system is presented. The system operation assumptions are outlined and the model input parameters are discussed. Furthermore, the performance measures, which are the quantitative outputs of the analytical model are defined.

- In Chapter IV the analytical model is developed. The model, termed the SMI model, is discussed. Three special cases of the general assumptions outlined in Chapter III are discussed. These cases are used to motivate the model and to demonstrate its capabilities. The cases are: the uniform case, the mailbox case, and the favorite module case.

- In Chapter V some hypothetical cases obtained in an effort to verify and validate the SMI model are described. In these cases the results from the simulations and the SMI model are compared and the relative percentage error between them is calculated. Furthermore, these results are used to obtain quantitative relationships among the different parameters of the multiprocessor system.

- In Chapter VI a multiprocessor system with cache memories is discussed to illustrate the usage of the SMI model in the design phase of such systems. Different implementations of the system are discussed and the SMI model is used to analyze the behavior of these different implementations. Furthermore, this example motivates and justifies the assumption of variable connection time between a processing element and a memory module. Simulation results are used to confirm the accuracy of the SMI model.

- In Chapter VII a multiprocessor system with multiple-bus network is discussed. The SMI model was modified to accommodate this extension. The effects of the bus contention on the system performance are illustrated.

- In Chapter VIII the concluding remarks are presented, the basic contributions of this dissertation are summarized, and some areas for further study are suggested.

# CHAPTER II

# BACKGROUND AND LITERATURE SURVEY

## 2.1. Introduction

The purpose of this chapter is to introduce the different memory interference models that have been reported in the literature and to motivate the work described in the following chapters. A taxonomy of these models is proposed in this chapter; the taxonomy is outlined in Figure 2.1. All of these models deal with a system similar to the one depicted in Figure 1.2. The multiprocessor system has $N$ processing elements connected to $M$ memory modules through an interconnection network. The processing elements may access any memory module. Two types of memory conflict, or memory interference, can occur.[1] Type one conflicts arise when one or more processing elements attempt to access a busy memory module. In this situation the processing elements wait until the memory module becomes idle before they reattempt an access. Type two conflicts arise when several processing elements attempt to access an idle memory module simultaneously. In this situation one of the processing elements is selected, according to a predefined selection strategy, to access the memory module while the other processing elements wait until the selected processing element is done. Therefore, both types of conflicts have a negative effect on the overall performance of the multiprocessor system. In the next two sections, the continuous time and the discrete time memory interference models are discussed.

[1] The models presented in this chapter do not consider conflicts that may arise in the interconnection network due to connectivity limitations, i.e., a virtual crossbar interconnection network is assumed unless it is specified otherwise.

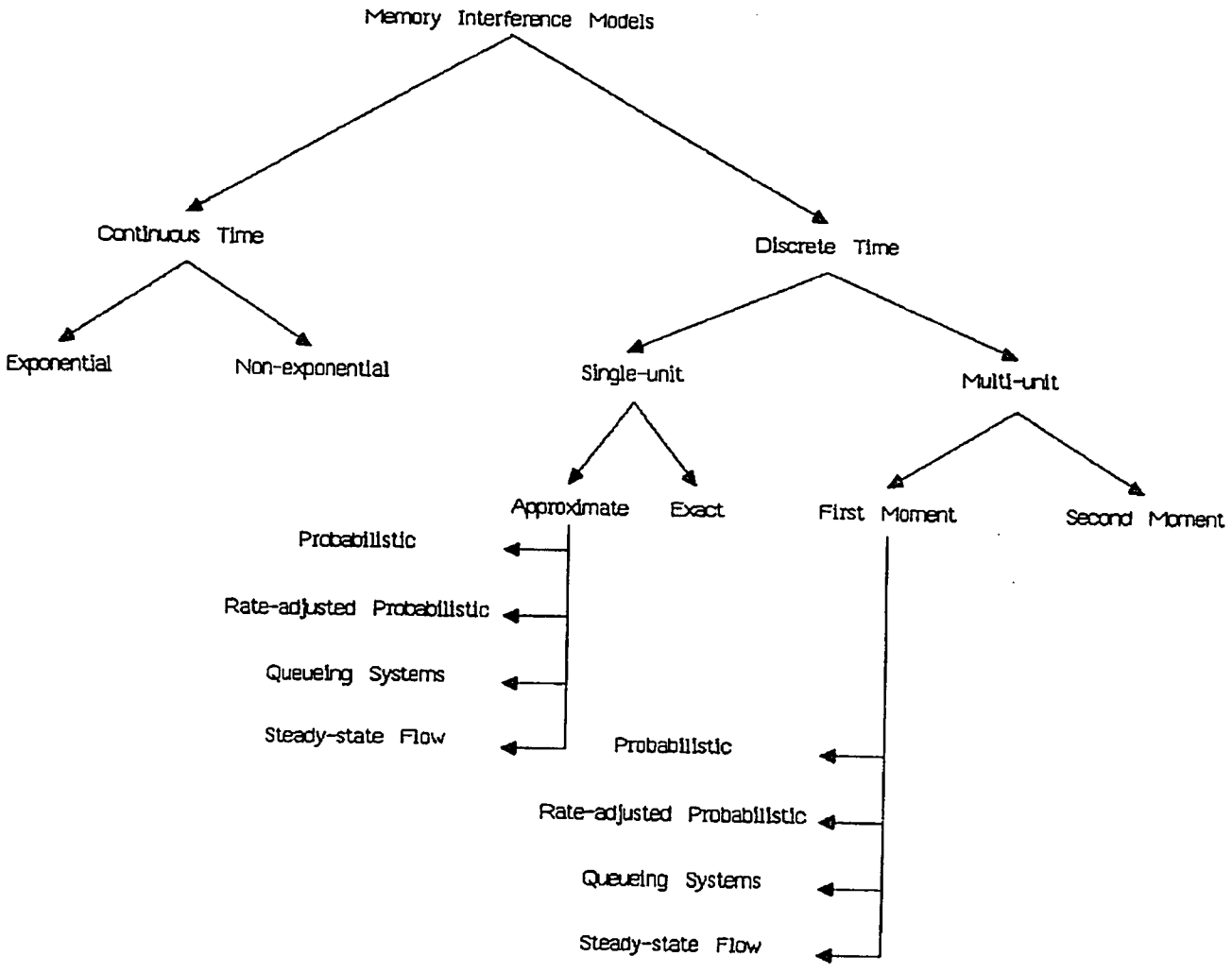



Figure 2.1 Taxonomy of memory interference models.

## 2.2. Continuous Time Models

In continuous time models the actual behavior of the multiprocessor system are approximated by a stochastic process as follows. A processing element spends a random amount of time executing some internal tasks without any reference to the memory modules; this period is referred to as the "thinking time." At the end of its thinking time, a request is placed to access a memory module. The destination of the request will be distributed uniformly between the $M$ memory modules. If a memory conflict of type one occurs, the request will join the queue of the destined module. Otherwise, the processing element will access that particular memory module. In the continuous time models, a type two memory conflict cannot occur because simultaneous events cannot occur, i.e., a type two memory conflict occurs with probability zero. The connection time between a memory module and a processing element is referred to as the "access time." At the end of the access time, a memory module will randomly select one of the requests in its queue, if any, and service it. The processing element will not attempt any useful work while its access request is in the queue of a memory module. Each processing element will have at most one pending request.

It can be seen from the previous description of the multiprocessor system operation that the development of an analytical model will be a laborious task. Therefore, some simplifying assumptions must be adopted by the analytical model in order to facilitate the development of the analytical model.

In this taxonomy, the continuous time models were divided into two types, namely: exponential and nonexponential models. This grouping depends on the simplifying assumptions adopted by these models.

### 2.2.1. Exponential Models

In the exponential models the simplifying assumption is that the access time and the thinking time are assumed to be exponentially distributed random variables. This assumption drastically simplifies the development of the analytical model. The first model was reported in

[BhF73]. In the study, a continuous time Markov chain was developed to analyze the multiprocessor system performance. The simplifying assumptions were that the think time of the processing elements is zero, and the connection time between the processing element and the memory module is distributed exponentially. The state in this continuous time Markov chain is an M-tuple $( k_1, k_2, \cdots, k_M )$, where $k_j$ is the number of the processing elements in the $j^{th}$ memory module. The memory bandwidth was expressed in a closed form as shown below:

$$BW = \frac{N\ M}{N + M - 1} .$$

This expression has a number of interesting properties: the expression is symmetric in $N$ and $M$; if $N = M$ and $N \rightarrow \infty$, then $BW \rightarrow ( N/2 )$. The second observation implies that the law of diminishing returns does not apply: no matter how many processing elements are used, one half of them are expected to be active.

The second model was reported in [MaG82]. In this study a continuous time Markov chain was developed to analyze the multiprocessor system performance. The study in [MaG82] further assumes that the interconnection network can be either a crossbar or multiple bus. Another study, reported in [OnI83], minimized the state space of the continuous time Markov chain, of [MaG82], by lumping equivalent states together. In both of these studies, the continuous time Markov chain was solved in order to obtain the processing power of the system. The processing power was defined as the expected number of the processing elements that are in the thinking period at steady state, i.e., the expected number of processing elements that are executing some internal tasks. Furthermore, the study in [OnI83] showed that decreasing the number of busses in a crossbar network by a factor of two produces a negligible degradation on the system performance in most cases. The study, reported in [MaG82], concluded that the robustness of the model was found to be very good for light system loads.

The main advantage of this approximation is the simplification of the analytical model. On the other hand, it has a number of disadvantages. First, the approximation yields invalid results in high load systems, see [BhF73] and [MaG82]. Second, the restriction of approximating the con-

nection time as an exponentially distributed random variable does not allow one to gauge accurately or to gain insight into the effect of a situation where this approximation does not hold.

### 2.2.2. Non-Exponential Models

The nonexponential model is reported in [Mak84]. In this model both the think time and the access time were assumed to have general distribution. The study in [Mak84] assumed that the interconnection network is a crossbar. A semi-Markov process was developed to model the program that is executed in the processing element. The model produces quantitative measures for the different components, e.g., utilization of the processing elements, the queueing waiting time, and the queue length. The main difference between this model and the other memory interference models is the way in which the behavior of the processing element is obtained. The model of [Mak84] obtains the behavior of the processing element from the program flow chart, i.e., a high-level description of the processing element activities, while the memory interference models obtain the processing element behavior from the program trace, i.e., a low-level description of the processing element activities. Nevertheless, the model reported in [Mak84] was used as a memory interference model. It was shown in [Mak84] that the model will produce an acceptable result for the utilization measures, but the other measures will have a reasonably large error component. The reason for such an error can be attributed to two factors: first, [Mak84] did not include the synchronous events due to the system synchronization and second, [Mak84] used an FCFS service discipline while the memory interference models assume a random service priority discipline. The first factor has a larger effect on the result than the second.

Most of the models proposed in the literature are discrete time models rather than continuous time models. Since the tightly coupled multiprocessor system is a discrete time system rather than a continuous time system.

## 2.3. Discrete Time Models

The literature contains a number of discrete time memory interference models for the multiprocessor system, depicted in Figure 1.2. The models in this section assume that the multiprocessor is synchronized with a system clock whose period is referred to as the "system cycle" or, where context allows, simply the "cycle." In these studies system operations are approximated by a stochastic process as follows. At the beginning of the system cycle a processing element, that has no pending request, makes a request to access a memory module with probability $r$. The memory module is chosen at random from the set of memory modules with probability $1/M$. If a type two memory conflict occurs at the memory module, that memory selects, with equal probability, one of the conflicting processing elements to access it. The processing elements that are not selected reattempt to access the same memory in the next system cycle. This retry will generally occur in the presence of new requests for access. The connection between the processing element and the memory module lasts for a number of system cycles; at the end of this connection time the processing element releases the memory module. A processing element will have at most one pending request waiting for access at any time. The behavior of the processing elements is considered to be independent and statistically identical. Hence, the thinking time is approximated as a geometrically distributed random variable.

The connection time between the processing element and the memory module will be used to categorize the discrete time models. Therefore, the discrete time models could be single-unit connection models or multiple-unit connection models. The single-unit connection models assume that the connection time between the processing element and the memory module lasts for one cycle which is the time to transfer a single word, while the multi-unit connection models assume that the connection time between the processing element and the memory module lasts for a number of cycles which is the time needed to transfer a block of words, e.g., a cache line or a page. The two types are discussed throughly in the next sections. Most of these models reported the memory bandwidth as the measure of the multiprocessor system performance. The memory bandwidth can be defined as the average number of memory modules which are connected to pro-

cessing elements at steady state. Other measures were reported in some of these studies, but will not be included in this discussion.

### 2.3.1. Single-unit Connection Models

In single-unit connection models, the connection between the processing element and the memory module will last for one system cycle. Therefore, only type two memory conflicts take place under the assumptions outlined in section 2.3. There are a number of studies that attempted to solve for the memory bandwidth exactly, while other models assume further simplifying assumptions in order to approximate the system performance.

#### 2.3.1.1. Exact Models

From the preceding description, the stochastic process, that approximates the multiprocessor system behavior, can be modeled using a discrete time Markov chain. One of the early models, reported in [SkA69], attempted to use a discrete time Markov chain. The main drawback of the solution proposed in [SkA69] was the unmanageable large state space of the Markov chain. The study assumes that the processing elements are not identical and every processing element has its own request pattern. Furthermore, each memory module has its own access priority assignment to resolve memory conflicts. Nevertheless, even if the assumptions outlined in section 2.3 were adopted, a moderate size multiprocessor system will need a large number of states in the Markov chain.

Another study, reported in [BhF73], proposed a discrete time Markov chain to solve the case where $r = 1$, i.e., the thinking time for the processing element is zero. The state of the Markov chain is defined by an M-tuple $(k_1, k_2, \cdots, k_M)$, where $k_j$ is the number of the processing elements that requested the $j^{th}$ memory module and $\sum_{j=1}^{M} k_j = N$. Therefore, the number of the states in the Markov chain is equal to $\binom{N+M-1}{M-1}$, i.e., the number of ways $N$ balls can be

assigned to $M$ bins. However, since all the processing elements behave identically, a number of states can be lumped together due to its equivalence. Thus, the reduced states are given by the different ways in which the number $N$ can be partitioned into $M$ parts. The number of such partitions (for $N \leq M$) is asymptotic to $(1/4\pi\sqrt{3})\, e^{-\pi\sqrt{2N/3}}$. Nevertheless, the state space grows rapidly with the size of the multiprocessor system. For a moderate sized system, a relatively large Markov chain must be solved.

The complexity of these models leads to the introduction of approximate models. In the approximate models further simplifying assumptions must be adopted by the models. These assumptions must fulfill two conditions. First, the resultant model is tractable computationally and second, the further simplifying assumptions must not change the system behavior beyond some small tolerance. In the next section an outline of these models and their assumptions are presented.

#### 2.3.1.2. Approximate Models

The approximate models develop equations for the memory bandwidth, $BW$, the probability that a memory request is accepted, $P_a$, and in some cases processor utilization, $U_p$. In [YPD82] a classification was presented for these models according to the approach used in their formulation. The classes are: probabilistic models, rate-adjusted probabilistic models, queueing system models, and steady-state flow models.

##### 2.3.1.2.1. Probabilistic Models

The probabilistic models simplify the analysis by assuming that a memory request from a processor will be discarded if it is denied memory access as a result of memory interference. At the next system cycle a new and unrelated request will be made by the processor with probability $r$. Examples of these models are reported in a number of studies, see [Str70, Rav72, Bha75, BrD77, Pat81, MuM82, BhL83].

The first study, reported in [Str70], assumed that $r = 1$. The results presented in [Str70] can be deduced as follows. The probability that a particular processing element requests a particular memory module is $1/M$. Therefore, the probability the processing element did not request a particular memory module is $( 1 - 1/M )$. Hence, the probability that none of the $N$ processing elements requested a particular memory module is $( 1 - 1/M )^N$. Therefore, the probability that some processing elements requested a particular memory module is $1 - ( 1 - 1/M )^N$, i.e., the probability a memory module is busy. The memory bandwidth can be obtained from the following equation:

$$BW = M \left[ 1 - ( 1 - 1/M )^N \right] . \tag{2.1}$$

Another model, reported in [Rav72], used combinatorial arguments to obtain the memory bandwidth of the multiprocessor system. The memory bandwidth was expressed as follows:

$$BW = \sum_{k=1}^{J} \frac{k \; k ! \; S ( N , k ) \binom{M}{k}}{M^k} , \tag{2.2}$$

where $J = min(N,M)$ and $S ( N , k )$ is the Stirling number of the second type, see [Lin68]. A study, reported in [CKL77], showed that equations (2.1) and (2.2) are equivalent.

The study, reported in [BrD77], used a similar argument to analyze a system of parallel-pipelined processors. In this system the memory modules are organized in a matrix form called L-M memory organization. The system cycle is assumed to be the pipeline segment unit time. Thus, the memory cycle consists of a number of segment unit cycles. A Markov chain was developed to model the status of one memory line and the memory bandwidth was deduced from the Markov chain.

As mentioned earlier, the study reported in [BhF73] proposed an exact solution for the case $r = 1$ using a discrete time Markov chain. The solution of the Markov chain is intractable for moderate sized systems. Nevertheless, it was concluded in that study that the memory bandwidth in this case is almost symmetric in $N$ and $M$. Furthermore, equation (2.1) is more accurate for the cases where $M > N$ than for the cases where $M < N$. To enhance the accuracy of equa-

tion (2.1), a study in [Bha75] suggested the following equation for the memory bandwidth:

$$BW = K \left[ 1 - \left( 1 - 1/K \right)^{J} \right] ,$$

where $K = \max(N,M)$ and $J = min(N,M)$.

None of the preceding studies were extended to cover the cases where $r < 1$. A study, reported in [Pat81], compared the performance of the delta network vs. the performance of the crossbar network where $r < 1$. In that study, the probability that a particular processing element would request a particular memory module is $r/M$ in lieu of $1/M$. Therefore, the memory bandwidth can be expressed as follows:

$$BW = M \left[ 1 - \left( 1 - r/M \right)^{N} \right] . \tag{2.3}$$

The same result was also reported in [MuM82, BhL83]. In both of these studies, the result was extended to cover the case were the processing elements are not identical.

### 2.3.1.2.2. Rate-adjusted Probabilistic Models

The rate-adjusted probabilistic models simplify the analysis by assuming that a rejected memory request will be resubmitted in the next cycle as a new request, i.e., it is not necessarily that the rejected request will be submitted to the original requested module. Obviously, this assumption will overestimate the memory bandwidth and underestimate the queue length in front of the module. A typical cycle of a processing element is shown in Figure 2.2. From the cycle, the adjusted request rate, $R$, can be calculated as follows:

$$\begin{aligned} R &= \frac{\textit{the number of the submissions of the request until accepted}}{\textit{the period of the cycle}} \\ &= \frac{1/P_a}{(1/r - 1) + 1/P_a} \\ &= \frac{1}{1 + P_a (1/r - 1)} \end{aligned} \tag{2.4}$$

Furthermore, the probability of acceptance, $P_a$, can be defined as the fraction of requests accepted per cycle. Therefore, it can be obtained from the following equation:

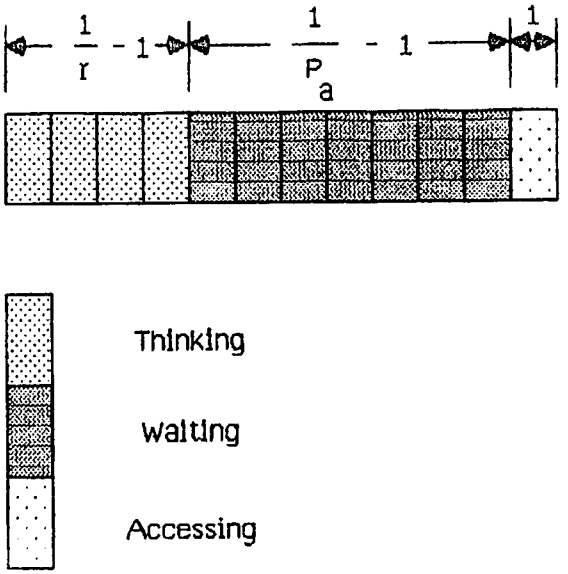



**Figure 2.2 A typical cycle of a processing element.**

$$P_a = \frac{\textit{the number of requests accepted}}{\textit{the total number of requests submitted}} \tag{2.5}$$

$$= \frac{BW}{N\,R} = \frac{M}{N\,R}\left[1 - \left(1 - R/M\right)^N\right] .$$

Equations (2.4) and (2.5) can be solved by fixed point iteration. Thereafter, the memory bandwidth can be expressed as follows:

$$BW = M\left[1 - \left(1 - R/M\right)^N\right] . \tag{2.6}$$

The model, reported in [Hoo77], furnished a similar result for the uniform case. The study was extended to cover nonidentical processing elements and it also covers cases where the thinking time is not geometrically distributed. Another study, reported in [EmD78], derived this result in the context of a shared control store for a multistream processor. The study evaluates perfor-

mance by allowing requests with deadlines wherein no performance degradation results from rejected requests provided they are accepted within some deadline number of cycles, as appropriate in a prefetch context.

#### 2.3.1.2.3. Queueing System Models

The queueing system models view the memory modules as service stations and processor requests as customers. In [BaS76] a binomial approximation is used to solve these queueing systems. The assumption in this class is that arrivals to the queue are binomially distributed. The study in [BaS76] obtained an expression that is asymptotically exact (as either $M$ and/or $N$ tend to infinity) for the system memory bandwidth in the case where $r = 1$. The memory bandwidth in this case can be expressed as follows:

$$BW = M + N - \frac{1}{2} - \sqrt{\left( M + N - \frac{1}{2} \right)^2 - 2MN} \quad . \tag{2.7}$$

The study of [BaS76] was extended to cover the cases where thinking time is more than zero, i.e., $r < 1$. The study concluded that the memory bandwidth of the multiprocessor system will be affected primarily by the first moment of the think time but will be relatively insensitive to higher moments of the distribution. Such conclusions were deduced after examining the results of trace driven simulations in which six different distribution functions were used to characterize the think time. The resulting bandwidth is given by the following equation:

$$BW = \frac{M}{2} \left[ 2 + 2L - \frac{1}{M} - \sqrt{(2 + 2L - 1/M)^2 - 8L} \right] \quad , \tag{2.8}$$

where $L$, the approximate mean queue length, is given by:

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

and $\overline{T}$ is the average think time. Therefore, if the think time is geometrically distributed, $\overline{T}$ can be expressed as follows:

$$\overline{T} = \frac{1}{r} - 1 \quad .$$

The expression for $L$ is obtained by approximating the length of the queue seen by an arriving customer to $(1-(1/N))L$. This linear approximation usually gives good results for highly utilized systems but works poorly for systems with low $r$. An improvement on this approximation is given in [Rau79] for the case where $r = 1.0$. The improvement relies on a decomposition approximation suggested in [BaS76]. The decomposition approximation stated that all customers (totaling K) not queued at the memory module in question were distributed among the other $M-1$ modules with precisely the same distribution that would occur at equilibrium in a $K \times (M-1)$ system. The memory bandwidth can be expressed as follows:

$$BW = \frac{\sum_{i=0}^{J-1} 2^i \binom{M-1}{i} \binom{N-1}{i}}{\sum_{i=0}^{J-1} \frac{2^i}{i+1} \binom{M-1}{i} \binom{N-1}{i}} \quad , \tag{2.9}$$

where $J = min(N,M)$.

An asymptotic solution can be obtained from equation (2.9), i.e., solutions for the cases $M$ and/or $N$ tend to infinity. In this case the memory bandwidth is insensitive to the addition of one or more memory modules to the multiprocessor system. The introduction of this approximation will derive equation (2.7) from equation (2.9) as is shown in [Rau79].

The last model in this class was reported in [YeF80]. The model in that study allows multi-unit connection time. Hence, it is discussed in detail in the next section. Nevertheless, the model can be used as a single-unit connection model which allows arbitrarily think time. The memory bandwidth can be expressed as follows:

$$BW = \frac{(2M(\overline{T}+1)+2N-1) - \sqrt{(2M(\overline{T}+1)+2N-1)^2 - 8MN(1+2\overline{T})}}{2(2\overline{T}+1)} \quad , \tag{2.10}$$

where $\overline{T}$ is the expected think time, i.e., $\overline{T} = (1/r) - 1$. Equation (2.10) can be reduced to

equation (2.7) if $\overline{T} = 0$. The study of [YeF80] reported that this model gives better results than the model of [BaS76].

#### 2.3.1.2.4. Steady State Flow Models

The steady-state flow models are proposed in [YPD82]. The idea behind this proposal is that the flow of active processors requests to the memory modules will equal the flow of satisfied requests from the memory modules. An active processor is a processor that does not have a pending request. The bandwidth obtained from the flow model is given by the following equations:

$$BW = N\ U_p\ r \tag{2.11}$$

and

$$BW = M \left[ 1 - \left( 1 - \frac{U_p\ r}{M} \right)^N \left\{ 1 - \frac{1}{M} \left[ 1 - \left( 1 - \frac{1-U_p}{M} \right)^N \right] \right\}^M \right] \tag{2.12}$$

Equations (2.11) and (2.12) can be solved by iteration using Newton's method. In [YPD82] it is shown that the steady-state flow model has a smaller maximum error and a smaller average mean-square error than the other models over the whole range of $r$, i.e., $0 \leq r \leq 1$. This model was extended to cover the case when the processing elements are not identical.

### 2.3.2. Multi-unit Connection Models

In the multi-unit connection models, the connection time between the processing element and the memory module will last for several system cycles. Hence, both types of memory conflicts occurs under this assumption. The exact solution of these models is intractable computationally for the same reason outlined in Section 2.3.1.1. Therefore, the models must adopt some further simplifying assumptions in order to make the development of the analytical models feasible. The common factor between most of the models in this category is that they were developed in order to study multiprocessor cache systems. In these systems the cache is placed on the processing element while the main memory is shared between the processing elements and resides on the

memory modules. The connection between the processing element and the memory module will last until a cache line is transferred.

The models in this category can be classified into two classes; namely first-moment models and second-moment models. The difference between these two classes is that the first-moment models use the expected value of the access time, i.e., the first moment of the access time, while the second-moment models use the first and second moments of the connection time. Thus, the first-moment models do not take the access distribution function into account.

#### 2.3.2.1. First Moment Models

Most of the first-moment models have been motivated by cache memory studies. The access time is viewed as the time needed to transfer a cache line between the cache memory and the main memory. This transfer is not necessarily deterministic, but the models in this class assume that the coefficient of variation, [2] $C_v$, of the access time is small. Therefore, the access time is nearly deterministic, and the average value of the access time is sufficient for the model computations.

The approach used to analyze the models in this class is similar to the one used to analyze the approximate models in the single-unit connection time category. One way of viewing the models in this class is that they are extensions of the models presented in section 2.3.1.2. Hence, a similar classification is used here to classify these models. The four types of models in this class are: probabilistic models, rate-adjusted probabilistic models, queueing system models, and steady-state flow models. These four types are discussed in the following sections.

---

[2] The coefficient of variation of a random variable is defined as the ratio between the standard deviation of the random variable and its expected value.

#### 2.3.2.1.1. Probabilistic Models

A study, reported in [YPD83], considered a tightly coupled multiprocessor system in which the main memory and cache memory are shared between the processing elements. The study assumed that each one of the $N$ processing elements is a pipelined processor of order $s$. The memory modules were organized in $L-M$ matrix organization in which each line has a number of shared cache modules and main memory modules. The analytical model is oriented toward developing the probability of acceptance, $P_a$, for a typical shared-cache memory request. Since the multiprocessor system organization in [YPD83] differs from the system depicted in Figure 1.2, a brief description of the multiprocessor operation assumptions are presented as follows. A processing element will place a request to a cache module in every segment time unit (STU), i.e., the basic unit of the multiprocessor system. The memory request will be directed, with equal probability, to one of the cache modules on any memory line. If the destined cache module accepts the request, it will be busy for $c$ STUs, i.e., the cache module cycle time. At the end of the cache module cycle time, the module will become idle if the request was a hit; otherwise, a block transfer will take place between that cache module and a main memory module connected to the same line. It is assumed that all requests in process within busy cache modules on a line will be aborted when an earlier request causes a cache miss on that line at the end of its cache memory cycle. The cache access conflict occurs when a request attempts to access a busy line or module or when two or more simultaneous cache memory requests attempt to access the same line. In order to simplify the analysis of this multiprocessor system, the study in [YPD83] assumed that the rejected requests will be discarded.

The study in [YPD83] models the state of a memory line by a discrete time Markov chain. The transition probabilities of the Markov chain are functions of three variables: the number of the cache modules per line, $m$; the hit ratio, $h$; and the probability that a particular line has been requested, $q$. The first two variables, $m$ and $h$, are input parameters to the model. However, the third variable, $q$, can be calculated using the following equation:

$$q = 1 - \left( 1 - 1/l \right)^N \quad , \tag{2.13}$$

where $l$ is the number of lines in the $L-M$ memory organization. The similarity between equation (2.13) and equation (2.1) is the result of discarding the rejected requests. A solution of the Markov chain is outlined in [YPD83]. The term $P_a$ will be derived from the Markov chain steady-state probabilities. The Markov chain state space will be large for long memory cycle time or long block transfer time.

#### 2.3.2.1.2. Rate-adjusted Probabilistic Models

The first rate-adjusted probabilistic model was reported in [Pat82] for a simple cache organization. The multiprocessor system of [Pat82] assumed that every processing element has its own private cache memory. In case of a cache fault a request will be issued by the cache memory, i.e., by the processing element, to one of the main memory modules—the memory modules represent the shared main memory. When the cache obtains the connection with the requested module, a cache line will be transferred through the interconnection network.

The stochastic process that approximates the actual behavior of the multiprocessor system was outlined earlier in Section 2.2. The request rate, $r$, is actually less than the miss ratio because not every system cycle is a memory reference. In other words, the miss ratio is the probability that a memory reference caused a cache fault, while the request rate, $r$, is the probability that the processor requests a memory location in its cache and that request caused a cache fault. The average access time needed to transfer the cache line between the private cache and the shared main memory is $\overline{C}$ cycles. (The study of [Pat82] used different notations from the ones presented here. The reason for the change is to keep the notations consistent in the dissertation.) In order to obtain a tractable solution for the stochastic process outlined previously, a further simplifying assumption must be adopted by the model. The simplifying assumption of [Pat82] was that a rejected request will be replaced by a new and independent request that will be resubmitted in the next cycle. In most of the multiprocessor systems that have cache memories, processor utilization, $U_p$, was defined as the fraction of time the processing element is not accessing or waiting to access the main memory as a result of a cache fault. The memory bandwidth in [Pat82] was

expressed in two different equations as follows:

$$BW = N\ \overline{C}\ r\ U_p$$

and

$$BW = M\left[1 - \left(1 - \frac{1 - U_p}{M}\right)^N\right] \quad . \tag{2.14}$$

These equations can be solved by iteration, and the memory bandwidth, $BW$, can be obtained. The preceding model was extended to cover other cache organizations; namely: write-through, buffered-write back, and load through. A discussion of these extensions is presented in Chapter VI.

Another model of this class was reported in [BrD83]. The study of [BrD83] assumed that the memory organization is L-M matrix. Similar to the study of [Pat82], this study uses only the first moment of the connection time. However, this study demonstrated that the connection between the processor and the memory module is variable in spite of the fact that a fixed size block is being transferred between them. This is due to the cache coherency checks that exist in a realistic cache system. However, the study in [BrD83] noted that their proposed model produces good estimates only when the coefficient of variation of the connection time is kept small (less than 0.5).

#### 2.3.2.1.3. Queueing System Models

This class contains the queueing system model reported in [YeF80]. The study of [YeF80] assumes a multiprocessor system similar to the system depicted in Figure 1.2. Each processing element has a processor and a private cache memory. The operation assumptions of the multiprocessor system are as follows. A processor will perform some internal tasks for a number of cycles that has an average value of $\overline{T}$ cycles. Then the processor will issue a request for memory access. With probability $h$, the request will be a cache hit and the processor will access its private cache memory for one cycle. Otherwise, the request will be a cache miss and a line transfer will take

place between the processing element and one of the shared main memory modules, that has been selected equiprobably. The line transfer will last for $\overline{C}$ cycles.

The study of [YeF80] approximates the multiprocessor system as a closed queueing system that has $N$ customers, one queue with infinite servers each with a deterministic service time of one cycle (to represent the private cache memories), and $M$ queues each with a deterministic service time of $\overline{C}$ cycles (to represent the shared main memory modules). The arrival process to these queues has been approximated as a binomial process. This approximation was used in [BaS76]. The model expresses the memory bandwidth, $BW$, as follows:

$$BW = \frac{X - \sqrt{X^2 - 8MN\left[\overline{C}^2 + \frac{2\overline{C}(\overline{T}+h)}{1-h}\right]}}{2\left[\overline{C}^2(1-h) + 2\overline{C}(\overline{T}+h)\right]} , \tag{2.15}$$

where

$$X = (2M+2N-1)\,\overline{C} + \frac{2M\,(\,\overline{T}+h\,)}{1-h} .$$

Furthermore, the study proposed two bounds for the memory bandwidth, $BW_{low}$ and $BW_{up}$, as follows:

$$BW_{low} = \frac{1}{\overline{T} + h + \overline{C}\,(1-h)}$$

and

$$BW_{up} = \frac{M}{\overline{C}\,(1-h)} .$$

These two bounds could be used to calculate the saturation point, $N^*$. The saturation point, $N^*$, can be defined as the number of processing elements beyond which a queue somewhere in the shared memory of $M$ modules is certain to be observed, see [DeB78] for details. The saturation point, $N^*$, is expressed as follows:

$$N^* = \frac{BW_{up}}{BW_{low}} = \frac{M\,[\,\overline{T} + h + \overline{C}\,(1-h)\,]}{\overline{C}\,(1-h)} .$$

#### 2.3.2.1.4. Steady-State Flow Models

This class contains one steady-state flow model reported in [MuA84]. The model is termed the equivalent rate model and model assumes a multiprocessor system similar to the system depicted in Figure 1.2. It assumes that the thinking period of the processor is geometrically distributed with parameter $r$. The processor places a request to one of the memory modules equiprobably after its thinking period. The connection time between the processor and the memory module has a mean value of $\overline{C}$ cycles. The equivalent rate model will approximate the behavior of this multiprocessor system by transforming it to a unit-connection multiprocessor system. This can be done by assuming that the connection time of $\overline{C}$ cycles is actually $\overline{C}$ connection times of one cycle each. Hence, the thinking time of the processor will decrease and the new parameter, $r_{eq}$, of the new geometric thinking time can be expressed as follows:

$$r_{eq} = \frac{\overline{C}}{(\frac{1}{r} - 1) + \overline{C}} \quad .$$

Thereafter, the equivalent multiprocessor system that has a request rate of $r_{eq}$ and a deterministic connection time of value one can be approximated using the steady-state flow model. Hence, the value $r_{eq}$ will be substituted in equations (2.11) and (2.12) to obtain the memory bandwidth, $BW$.

### 2.3.2.2. Second Moment Models

As mentioned earlier, the study of [BrD83] demonstrated the need to include higher moments of the connection time in the memory interference model. The first model to include higher moments of the connection time is the Markov chain (MC) model reported in [MuA84]. Because of the significance of the MC model in the development of the model proposed in this dissertation, this model will be discussed in detail. The MC model assumes a multiprocessor system similar to the system depicted in Figure 1.2. It assumes that the thinking period of the processor is geometrically distributed with parameter $r$. The processor places a request to one of the

memory modules equiprobably after its thinking period. The connection time between the processor and the memory module can be characterized as a discrete random variable, $X$, which has a known probability mass function, $f(x)$. The MC model adopts a decomposition approximation approach to analyze the behavior of the multiprocessor system. In this approximation the discrete time Markov chain that describes the exact behavior of the multiprocessor system is decomposed into $N$ identical Markov chains, each describes the behavior of one of the processors. The coupling between these $N$ independent chains is embedded in the calculations of the transition probabilities between the states. The Markov chain that describes the behavior of one of the processors is depicted in Figure 2.3. Solution of the Markov chains produces the following three equations:

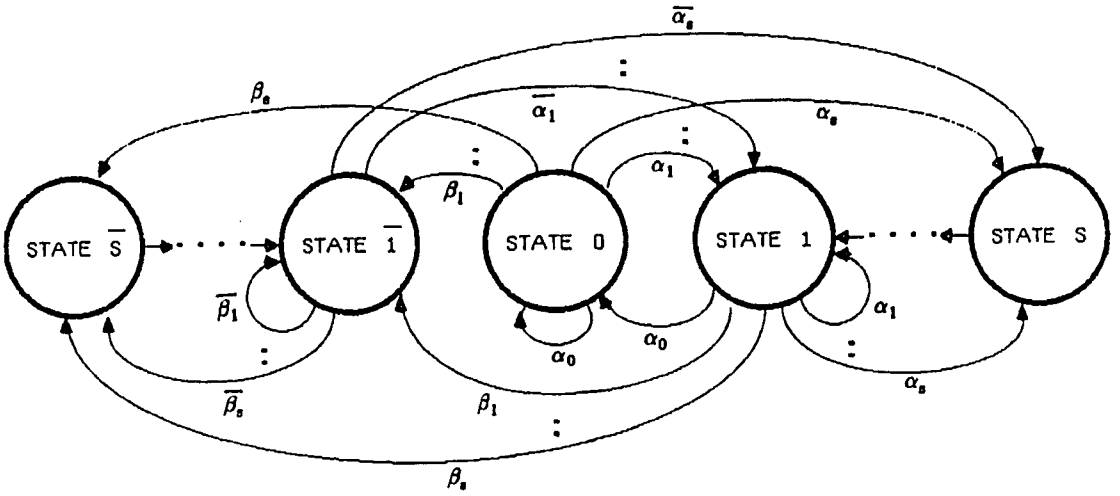



**Figure 2.3 The Markov chain that describes one processor.**

$$P_{win} = \frac{M}{N\,R}\left[1-\left(1-R/M\right)^{N}\right] ,$$

$$B = \frac{(\bar{X}-1)\,P_{win}\,R}{1+\frac{N-1}{M}\,(\bar{X}-1)\,P_{win}\,R}$$

and

$$R = \frac{1}{\left(1-\frac{N-1}{M}\,B\right)\left[\bar{X}+(1/r-1)\,P_{win}+\frac{(N-1)\,P_{win}\,R}{M}\,\frac{(\bar{X^2}-\bar{X})}{2}\right]} .$$

These three equations are solved by iterating on the value of $R$ (the initial value of $R$ is $r$). A fixed point iteration is sufficient in this case. It is noted that the full distribution of the discrete random variable, $X$, is not needed. However, the first two moments of $X$, i.e., $\bar{X}$ and $\bar{X^2}$, are needed. The study of [MuA84] demonstrated that the performance of the multiprocessor system degrades as the variation in the random variable, $X$, increases. In other words, the bandwidth of the multiprocessor system $BW$ decreases as the second moment of the connection time $\bar{X^2}$ increases.

The previous models mainly used the utilization of the processors or the memory modules as the performance measures of the multiprocessor system. Some of the performance measures, such as the queueing time and queue length, that are essentials in certain cases have been ignored by most of these memory interference models. The calculation, or prediction, of such measures was one of the major motivations of this research. For instance, the need to obtain an approximation of the queueing time is important in calculating the execution time of a job by one of the processors given its memory request trace.

# CHAPTER III

# SYSTEM ASSUMPTIONS AND PERFORMANCE MEASURES

## 3.1. Introduction

The system of interest in this study, depicted in Figure 1.2, is a synchronized multiprocessor system that has $N$ processing elements connected to $M$ memory modules through an interconnection network. The network is assumed to be a crossbar,[1] i.e., no connectivity limitation is assumed. Therefore, the processing elements may access any memory module. Two types of memory conflict, or memory interference, can occur. Type one conflicts arise when one or more processing elements attempt to access a busy memory module. Type two conflicts arise when several processing elements attempt to access an idle memory module simultaneously. Both types of conflicts have a negative effect on the overall performance of the multiprocessor system.

Before proceeding to the system operation assumptions and the performance measures, some notations that will be used throughout this dissertation are presented. A processing element in the multiprocessor system may be in any of three states: *thinking*, when it is working on an internal task with no memory request outstanding; *accessing*, when it is connected to a memory module; and *waiting*, when it is waiting in the queue of a memory module for that memory to become available. The memory module can be in any of two states: *busy*, when a processing element is connected to it and *idle*, when there is no processing element connected to it. The $i^{th}$

[1] The crossbar is replaced by a multiple-bus connection in Chapter VII to study the conflict due to connectivity limitations.

processing element is denoted by $PE_i$; the $j^{th}$ memory module is denoted by $MM_j$; a discrete random variable is denoted by its name with a ~ above it, e.g., the discrete random variable $Z$ is denoted by $\tilde{Z}$; the cumulative distribution function (CDF) of $\tilde{Z}$ is denoted by $Z(x)$, i.e., $Z(x) = Pr[\tilde{Z} \leq x]$; the probability mass function (pmf) of $\tilde{Z}$, is denoted by $z(x)$, i.e., $z(x) = Pr[\tilde{Z} = x]$; the mean value of $\tilde{Z}$ is denoted by $\bar{Z}$; and the $n^{th}$ moment of $\tilde{Z}$ is denoted by $\overline{Z^n}$.

### 3.2. System Operation Assumptions

The multiprocessor system operation is characterized by the following assumptions:

I. The behavior of the $PE$s can be modeled as stochastic processes.

II. The $PE$s think for an integer number of system cycles. The thinking period of $PE_i$ is characterized by a discrete independent random variable, $\tilde{T}_i$, where $1 \leq i \leq N$.

III. Each $PE$ will submit a memory request after its thinking period; requests originating from the same processing element are independent of each other. The destination memory module of the request originating from $PE_i$ will be determined by a discrete independent random variable, $\tilde{D}_i$, where $1 \leq i \leq N$.

IV. When the first type of memory conflict occurs, the memory module equiprobably selects one of the conflicting processing elements to gain access. The blocked processing element(s) wait until the connection is completed and then resubmit their requests to the same memory module.

V. When the second type of memory conflict occurs, the blocked processing element(s) wait until the connection is completed and then resubmit their requests to the same memory module.

VI. The connection time between the $i^{th}$ processing element, $PE_i$, and the $j^{th}$ memory module, $MM_j$, is characterized by a discrete independent random variable, $\tilde{C}_{i,j}$, where $1 \leq i \leq N$ and $1 \leq j \leq M$.

Empirical evidence reported in [Bha75, BaS76, Hoo77] supports the preceding assumptions in the case where the $PE$s are identical, $\tilde{T}$ is geometrically distributed, $\tilde{D}$ is uniformly distributed between 1 and $M$, and $\tilde{C}$ is deterministic with a value of one. Further work reported in [Mak84] supports these assumptions.

In order to obtain numerical information from the SMI model developed later, the values of $M$, $N$, $\overline{T}$, $\overline{C}$ and $\overline{C^2}$ must be obtained through measurements or by hypothesis. These quantities can be regarded as input parameters of the SMI model; knowledge of the full distributions of $\tilde{T}$ and $\tilde{C}$ is not necessary for solving the SMI model.

### 3.3. Performance Measures

A number of performance measures can be derived from the analytical model. These measures are: the memory bandwidth, $BW$; the $i^{th}$ processing element utilization, $PU_i$; the $j^{th}$ memory module utilization, $MU_j$; the average queue length of the $j^{th}$ memory module queue, $L_j$; and the average waiting time experienced by the $i^{th}$ processing element in the $j^{th}$ memory module queue, $W_{ij}$.

- The memory bandwidth, $BW$, is defined as the average number of busy memory modules when the multiprocessor system reaches steady state. This is the same as the average number of accessing processing elements when the multiprocessor system reaches steady state. Hence, $BW$ can be expressed as follows:

$$BW = \lim_{t \to \infty} \left[ \sum_{i=1}^{N} Pr \left[ PE_i \text{ is accessing at time } t \right] \right] .$$

- The $i^{th}$ processing element utilization, $PU_i$, is the probability that the $i^{th}$ processing element is thinking or accessing a memory module when the multiprocessor system reaches steady state. Hence, $PU_i$ can be expressed as follows:

$$\begin{aligned} PU_i &= \lim_{t \to \infty} Pr \left[ PE_i \text{ is thinking or accessing at time } t \right] \\ &= 1 - \lim_{t \to \infty} Pr \left[ PE_i \text{ is waiting at time } t \right] . \end{aligned}$$

Some of the memory interference models, [Pat82] and [BrD83], that were motivated by cache studies of multiprocessor systems define $PU_i$ as the probability that the $i^{th}$ processing element is thinking when the multiprocessor system reaches steady state. In this study, it is considered that memory accessing contributes to the progress of the computation and is

therefore counted as useful work. The alternative quantity can be derived readily from the SMI model if it is required.

- The $j^{th}$ memory module utilization, $MU_j$, is the probability that the $j^{th}$ memory module is busy when the multiprocessor system reaches steady state. This is the same as the probability that any processing element is accessing the $j^{th}$ memory module when the multiprocessor system reaches steady state. Hence, $MU_j$ can be expressed as follows:

$$MU_j = \lim_{t \to \infty} \sum_{i=1}^{N} Pr\ [\ PE_i \text{ is accessing } MM_j \text{ at time } t\ ]$$

The memory bandwidth, $BW$, can be expressed in terms of the memory utilizations as follows:

$$BW = \sum_{j=1}^{M} MU_j$$

- The average queue length at the $j^{th}$ memory module, $L_j$, can be defined as the expected number of processing elements waiting to access the $j^{th}$ memory module. Hence, $L_j$ can be expressed as follows:

$$L_j = \sum_{i=1}^{N} \lim_{t \to \infty} Pr\ [\ PE_i \text{ is waiting to access } MM_j\ ]$$

- The average waiting time experienced by the $i^{th}$ processing element while waiting to access the $j^{th}$ memory module is denoted by $W_{i,j}$.

It is shown in Chapter IV that the last two measures fall out naturally from the analytical model.

# CHAPTER IV

# MODEL DESCRIPTION

## 4.1. Introduction

In this chapter an analytical model is developed in order to study a multiprocessor system that adopts the operation assumptions outlined in Chapter III. An exact Markov chain model is outlined in Section 4.2. However, it is shown that this model is intractable because of the size of its state space. This difficulty demonstrates the need for approximate analytical models that adopt further simplifying assumptions. The main purpose of this adoption is to make the model tractable without changing the overall behavior of the system "significantly."

The SMI model, that is developed in this chapter, is an approximate model for the multiprocessor system characterized in Chapter III. The model is presented first for three special cases—that are termed: uniform case, mailbox case and favorite case. Thereafter, the general SMI model is presented as an approximate model to analyze the multiprocessor system characterized in Chapter III.

## 4.2. The Exact Model

As was previously stated in Section 2.3.1.1, a discrete time Markov chain can be developed to analyze the behavior of the multiprocessor system outlined in Chapter III. However, the size of the state space of this Markov chain will be extremely unmanageable. To illustrate this point the following example is considered:

A multiprocessor system that has the same operation assumptions as the ones presented in Section 3.1 is considered. The system has two $PE$s and one $MM$. The thinking time of the first $PE$, $PE_1$, is uniformly distributed between one and four cycles, while the thinking time of the second $PE$, $PE_2$, is equiprobably to be zero or one cycle. The connection time between $PE_1$ and $MM$ is uniformly distributed between one and four cycles. The connection time between $PE_2$ and $MM$ is deterministic with value equal to two cycles. Table 4.1 shows the list of the states of the Markov chain that describes exactly the behavior of the multiprocessor system. In this example the Markov chain has 29 states. The state of the Markov chain is defined by a 2-tuple, ($k_1$,

| State of $PE_1$ | State of $PE_2$ | The set of MC states |
|---|---|---|
| Thinking | Thinking | (1,1) ; (2,1) ; (3,1) ; (4,1) |
| Thinking | Accessing | (1,+1) ; (1,+2) ; (2,+1) ; (2,+2)<br>(3,+1) ; (3,+2) ; (4,+1) ; (4,+2) |
| Accessing | Thinking | (+1,1) ; (+2,1) ; (+3,1) ; (+4,1) |
| Accessing | Waiting | (+1,-1) ; (+2,-1) ; (+2,-2) ; (+3,-1)<br>(+3,-2) ; (+3,-3) ; (+4,-1) ; (+4,-2)<br>(+4,-3) ; (+4,-4) |
| Waiting | Accessing | (-1,+1) ; (-1,+2) ; (-2,+2) |

**Table 4.1 The list of the states of the Markov chain of the example.**

$k_2$), where $k_j$ can be defined as follows:

$$k_j = \begin{cases} +i & PE_j \text{ has been accessing MM for the last } i \text{ cycles.} \\ -i & PE_j \text{ has been waiting for MM for the last } i \text{ cycles.} \\ i & PE_j \text{ has been thinking for the last } i \text{ cycles.} \end{cases}$$

The transition probabilities of the Markov chain can be calculated from the system parameters given here. It is noted that this Markov chain is not the smallest chain that can describe the exact behavior of the multiprocessor system outlined previously. However, the state space of any Markov chain that describes a moderate-size multiprocessor system will be large and unmanageable. ∎

In general, the state of the Markov chain is defined by a $2N$-tuple, $(k_1, l_1, \cdots, k_N, l_N)$, where the pair $k_j, l_j$ are defined as follows:

$$k_j, l_j = \begin{cases} s, +i & PE_j \text{ has been accessing } MM_s \text{ for the last } i \text{ cycles.} \\ s, -i & PE_j \text{ has been waiting for } MM_s \text{ for the last } i \text{ cycles.} \\ 0, i & PE_j \text{ has been thinking for the last } i \text{ cycles.} \end{cases}$$

It is noted that this description of the Markov chain may not be the optimum description. However, any other equivalent description has unmanageable state space. Furthermore, it can be seen that the size of the Markov chain is a function of the following three parameters: the size of the multiprocessor system, the maximum thinking time of a processing element, and the maximum accessing time.

For moderate-size systems the use of approximate models is inevitable. The approximation is introduced to the model by assuming further simplifying assumptions. These assumptions should reduce the complexity of the model drastically without changing the overall performance significantly.

## 4.3. The Approximate Model

In this study an approximate model, termed the semi-Markov interference (SMI) model, is introduced. The SMI model adopts simplifying assumptions similar to the ones presented in

[MuA84]. In that work the Markov chain that describes the model exactly is replaced by $N$ identical Markov chains. Each of these chains describes the behavior of a $PE$. In other words, the enormous Markov chain is decoupled into $N$ separate moderate-size Markov chains. The coupling between the $N$ chains appears in the transition probabilities between the states within each chain. Solving the model requires only one of the chains to be considered which dramatically reduces the solution complexity. Moreover, because the chains are coupled, independence of $PE$s does not have to be assumed, resulting in a more realistic model (assumption Ia does not imply independence). The number of states in the model of [MuA84] can still grow large, in some cases, because it depends on the number of discrete values $\tilde{T}$ and $\tilde{C}$ can assume. This can be avoided, resulting in a further simplification, by replacing the Markov chains by semi-Markov processes.

A detailed discussion of semi-Markov processes (SMPs) is found in [Cin75, HeS82, Ros70]. Briefly, a semi-Markov process (SMP) is a stochastic process that can be in any one of $K$ states $1, 2, \ldots, K$. Each time it enters state $i$ it remains there for a random amount of time (the sojourn time) having mean $\eta_i$ and then makes a transition into state $j$ with probability $p_{i,j}$. As a special case, a discrete time Markov chain is an SMP with a deterministic sojourn time of value one. If the SMP has an irreducible embedded Markov chain that consists of ergodic states, then the limiting probability of being in state $i$, denoted by $P_i$, can be expressed as follows:

$$P_i = \frac{\pi_i \eta_i}{\sum_{j=1}^{K} \pi_j \eta_j} \quad , \tag{4.1}$$

where $\pi_i$ is the limiting probability of state $i$ in the embedded Markov chain. All the SMPs that appear in this dissertation have irreducible embedded Markov chains with ergodic states, therefore, equation (4.1) will always be applicable. The rate of leaving state $i$, $\lambda_i$, is defined as the reciprocal of the average time elapsed between two consecutive departures from state $i$. The rate can be obtained using the following equation:

$$\lambda_i = \frac{P_i}{\eta_i} = \frac{\pi_i}{\sum_{j=1}^{K} \pi_j \eta_j} \quad . \tag{4.2}$$

Since the average sojourn time in any one of the states of the SMPs that appear in this study is at least one system cycle, then $\lambda_i$ falls in the range [0,1] and it is possible to view $\lambda_i$ as the probability of leaving state $i$ at the beginning of a system cycle.

Before moving on to describe the SMI model, we shall present a theorem about the average residual accessing time of a busy module is presented. The residual accessing time is defined as the amount of time remained until the accessing processor finishes its accessing period. This theorem is important in the development of the SMI model.

**Theorem 1:**

The average residual accessing time between $PE_i$ and $MM_j$ seen by a requesting $PE$ at the beginning of a system cycle is expressed as $( \overline{C^2}_{i,j} - \overline{C}_{i,j} ) / 2 ( \overline{C}_{i,j} - 1 )$, where $\overline{C}_{i,j}$ and $\overline{C^2}_{i,j}$ are the first and second moments of the connection time, $\tilde{C}_{i,j}$, respectively.

**Proof:**

The definition of the probability mass function of the random variable, $\tilde{C}_{i,j}$, states that:

$$c_{i,j} ( k ) = Pr [ \tilde{C}_{i,j} = k ] .$$

Two events, $A_k$ and $B$, are used in this proof. These events are defined as follows: $A_k$ is the event that a requesting $PE$ will see a residual accessing time of $k$ cycles between $PE_i$ and $MM_j$; $B$ is the event that a requesting $PE$ will find $MM_j$ busy. Therefore, the average residual accessing time of a busy memory, $MM_j$, seen by a requesting $PE$ can be expressed as follows:

$$Average\ residual\ time = \eta_{i,j} = \sum_{k=1}^{S-1} k\ Pr [ A_k | B ] ,$$

where $S$ is the maximum connection time between a $PE_i$ and an $MM_j$. Since $A_k$ and $B$ are dependent events, Bayes' rule can be used to obtain the following equation:

$$Pr [ A_k | B ] = \frac{Pr [ A_k \cap B ]}{Pr [ B ]} .$$

The term $Pr\ [\ A_k \cap B\ ]$ is determined as follows:

$$Pr\ [\ A_k \cap B\ ] = \sum_{l=1}^{S-k} Pr\ [\ \textit{the accessing } PE_i \textit{ submitted } l \textit{ cycles ago to } MM_j \textit{ an accessing request of } k+l \textit{ cycles}\ ]$$

$$= \sum_{l=1}^{S-k} c_{i,j}\ (k+l) \ = \ \sum_{l=k+1}^{S} c_{i,j}\ (l) \quad .$$

while the term $Pr\ [\ B\ ]$ is determined as follows:

$$Pr\ [\ B\ ] = \sum_{l=1}^{S-1} Pr\ [\ \textit{the accessing } PE_i \textit{ obtained the connection with } MM_j \textit{ } l \textit{ cycles ago for at least } l+1 \textit{ cycles}\ ]$$

$$= \sum_{l=1}^{S-1} \sum_{k=l+1}^{S} c_{i,j}\ (k) \ = \ \sum_{l=1}^{S} (\ l-1\ )\ c_{i,j}\ (l) \quad .$$

Therefore, $\eta_{i,j}$ can be expressed as follows:

$$\eta_{i,j} \ = \ \sum_{k=1}^{S-1} i\ \frac{\sum_{l=k+1}^{S} c_{i,j}\ (l)}{\sum_{l=1}^{S} (l-1)\ c_{i,j}\ (l)} \ = \ \frac{\sum_{k=1}^{S-1} k \sum_{l=k+1}^{S} c_{i,j}\ (l)}{\sum_{l=1}^{S} (l-1)\ c_{i,j}\ (l)} \ = \ \frac{\sum_{k=1}^{S} \frac{k\ (k-1)}{2}\ c_{i,j}\ (k)}{\sum_{l=1}^{S} (l-1)\ c_{i,j}\ (l)}$$

$$= \frac{\overline{C^2}_{i,j} - \overline{C}_{i,j}}{2\ (\ \overline{C}_{i,j} - 1\ )} \qquad Q.E.D. \quad \blacksquare$$

In addition to this theorem, three probabilistic terms that will be used throughout the SMI model development are defined. These terms are: $R_{i,j}$, $WIN_{i,j}$ and $BUSY_{i,j}$. They are defined as follows:

**Definition 1:**

The term $R_{i,j}$ is defined as the probability that $PE_i$ makes a request to access $MM_j$ at the beginning of a system cycle. Hence, it is the probability that one of two events occurs at the beginning of a system cycle. The first event is that $PE_i$ will direct a new request to $MM_j$. The second event is that $PE_i$ will resubmit a previously blocked request to access $MM_j$. The term $R_{i,j}$ can be viewed as the request rate of $PE_i$ to $MM_j$.

**Definition 2:**

The term $WIN_{ij}$ is the probability that $MM_j$ selects the request of $PE_i$ over other requests (if there are any) at the beginning of a system cycle, given that $MM_j$ is idle at that time.

**Definition 3:**

The term $BUSY_{ij}$ is defined as the probability that $PE_i$ finds $MM_j$ busy at the beginning of a system cycle. Therefore, one of the other $(N-1)$ PEs is accessing $MM_j$ and is not on the point of releasing it.

To simplify the development of the SMI model, the SMI model is presented in three cases that make simpler operation assumptions than the ones outlined in Chapter III. These cases are termed: the uniform case, the mailbox case, and the favorite module case. Thereafter, the general SMI model is presented. The general SMI model assumes that the operation assumptions are the ones outlined in Chapter III.

### 4.3.1. Uniform Case

In the uniform case, the multiprocessor system operation is characterized by the following assumptions:

Iu. The behavior of the $PE$s can be modeled as identical stochastic processes.

IIu. The $PE$s think for an integer number of system cycles. The thinking period of any $PE$ is characterized by a discrete independent random variable, $\tilde{T}$.

IIIu. Each $PE$ will submit a memory request after its thinking period; requests originating from the same processing element are independent of each other. The destination of the request originating from any $PE$ will be uniformly distributed between the $M$ modules.

IVu. When the first type of memory conflict occurs, the memory module equiprobably selects one of the conflicting processing elements to gain access. The blocked processing element(s) wait until the connection is completed and then resubmit their requests to the same memory module.

Vu. When the second type of memory conflict occurs, the blocked processing element(s) wait until the connection is completed and then resubmit their requests to the same memory module.

VIu. The connection time between processing element and any memory module is characterized by a discrete independent random variable, $\tilde{C}$.

Appending the letter "u" to the assumption number and the equation number in this section was done for convenience. Furthermore, the subscripts to the variables are not used in this case due to the symmetry between the *PE*s and the *MM*s.

As mentioned previously, the SMI model uses an SMP to approximate the behavior of a *PE* that functions according to the system operation assumptions outlined previously. Therefore, $N$ identical SMPs will approximate the behavior of the multiprocessor system. The SMP in this case is depicted in Figure 4.1. The states of the SMP denote the different states of any *PE*, and they can be partitioned to four disjoint subsets.[1] The first subset is the *thinking subset*, $S^{th} = \{0\}$. The process enters state 0 and remains there for a duration of time with mean value $\eta_0$, equivalent to the thinking time of the *PE* (see Figure 4.1). A memory request is modeled by the SMP leaving state 0. The destination state depends on the state of the requested *MM*. The second subset is the *accessing subset*, $S^{ac} = \{1\}$. The process enters state 1 and remains there for a duration of time with mean value $\eta_1$, equivalent to the connection time between the *PE* and any *MM*. From state 1 the process returns to state 0, i.e., the *PE* resumes thinking after it has completed its memory access. The third subset is the *full waiting subset*, $S^{fw} = \{2\}$. The process enters state 2 when the *PE* requests an idle *MM* simultaneously with at least one other request and the *PE* fails to be selected by the module, i.e., a type one conflict occurs and another *PE* is selected to have access to the *MM*. In this case the *PE* must wait for the full duration of the connection time between the *MM* and the selected *PE*; this duration has a mean value of $\eta_2$. The original *PE* will retry to access the same *MM* when the selected *PE* releases the module. If it succeeds, the process enters state 1, otherwise the process reenters state

[1] For the moment these subsets are singletons. Later generalizations increase their cardinality.

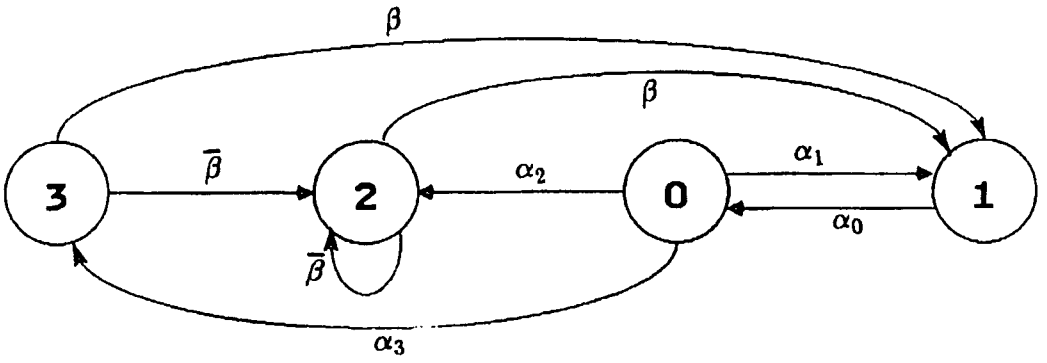



**Figure 4.1 SMP that describes PE behavior with the uniform case assumptions.**

2. The fourth subset is the ***residual waiting subset***, $S^{rw} = \{3\}$. The process enters state 3 when the *PE* requests a busy *MM*, i.e., when a type two conflict occurs. The *PE* must wait for the remaining (residual) connection time before retrying to access the *MM*; the mean value for the residual time is $\eta_3$. The process then enters state 1 if the *PE* succeeds in accessing the *MM*, or it enters state 2 if it fails to obtain a connection. Clearly, the SMP description does not include which module the *PE* is accessing or which module the *PE* is waiting to access. This does not represent an approximation of the behavior of the *PE* because of the symmetry in the uniform case. In nonsymmetric cases, as is shown later, the SMP must represent this information. The underlying approximation of the SMI model is in describing any *PE* behavior independently from the other *PE*s while compensating for the coupling between the behaviors of the *PE*s in the transition probabilities between the states of the SMP (the coupling results from the *PE*s sharing the *MM*s).

In order to derive numerical information from the SMI model, the values of $N$, $M$, the first moment of $\tilde{T}$, and the first two moments of $\tilde{C}$ must be obtained through measurement or, if it is

considered satisfactory, by hypothesis. These quantities can be regarded as the input parameters to the model. These parameters are defined as follows:

$$
\begin{aligned}
N &\triangleq \textit{the number of } PEs, \\
M &\triangleq \textit{the number of } MMs, \\
\overline{T} &\triangleq \textit{the first moment of } TT, \\
\overline{C} &\triangleq \textit{the first moment of } \tilde{C}
\end{aligned}
$$

and

$$
\overline{C^2} \triangleq \textit{the second moment of } \tilde{C} \ .
$$

The average sojourn times of the different states of the SMP can be obtained from the parameters of the model as follows:

$$
\eta_j = \begin{cases} \overline{T} & j = 0 \\ \overline{C} & j = 1 \\ \overline{C} & j = 2 \\ \dfrac{\overline{C^2} - \overline{C}}{2(\overline{C} - 1)} & j = 3 \end{cases} \tag{4.3u}
$$

The average sojourn times in states 0, 1 and 2 arise directly from the definition of these states. The average sojourn time in state 3 is obtained by using Theorem 1.

The terms $R$, $WIN$ and $BUSY$, see definitions 1 through 3, are not subscripted in this case due to the symmetry of $PE$s and $MM$s. From definition 1, the term $R$ can be computed as the probability of leaving state 0 or 2 or 3 to access a particular $MM$. Therefore, $R$ is defined as follows:

$$
\begin{aligned}
R &\triangleq R_{ij} \qquad \forall\ i, j \\
&= \frac{1}{M} \left( \lambda_0 + \lambda_2 + \lambda_3 \right) \ .
\end{aligned} \tag{4.4u}
$$

The term $WIN$ is derived by the following argument: the probability that a $PE$ will not request a particular $MM$ is $1 - R$; the probability that none of the $N$ $PE$s request that $MM$ is $\left( 1 - R \right)^N$; the probability that a particular $MM$ is requested by at least one of the $PE$s is $\left[ 1 - \left( 1 - R \right)^N \right]$; and the expected number of $PE$s that requested that $MM$ at the beginning

of a system cycle is $NR$. Therefore, $WIN$ can be defined as follows:

$$\begin{aligned} WIN &\triangleq WIN_{i,j} \qquad \forall\ i, j \\ &= \frac{1}{N\,R}\left[1-\left(1-R\right)^{N}\right] . \end{aligned} \tag{4.5u}$$

Finally, the term $BUSY$ is defined as the probability that a $PE$ finds a particular $MM$ busy at the beginning of a system cycle. Therefore, one of the other $(N-1)$ PEs is accessing that $MM$ and is not on the point of releasing it. In other words, the requesting $PE$ experienced a memory conflict of type one. Hence, $BUSY$ is the probability that one of $(N-1)$ PE's is accessing a particular $MM$ and the accessing $PE$ is not on the point of releasing the $MM$. Thus, $BUSY$ can be defined as follows:

$$\begin{aligned} BUSY &\triangleq BUSY_{i,j} \qquad \forall\ i, j \\ &= \frac{N-1}{M}\left(P_1-\lambda_1\right) = \frac{N-1}{M}\left(\bar{C}-1\right)\lambda_1 . \end{aligned} \tag{4.6u}$$

The last step follows from equation (4.2).

The transition probabilities between the states of the SMP can be derived as the following functions of $BUSY$ and $WIN$:

$$\alpha_j = \begin{cases} 1 & j=0 \\ (1-BUSY)\,WIN & j=1 \\ (1-BUSY)(1-WIN) & j=2 \\ BUSY & j=3 \end{cases} \tag{4.7u}$$

$$\beta = (1-BUSY)\,WIN$$

$$\bar{\beta} = 1-\beta$$

The derivation proceeds as follows. When the process, shown in Figure 4.1, leaves the thinking state it enters any one of the other states: it enters the accessing state with probability $\alpha_1$ if the $MM$ is idle and the $PE$'s request is selected; it enters the full waiting state with probability $\alpha_2$ if the $MM$ is idle and the $PE$'s request fails to be selected; or it enters the residual waiting state with probability $\alpha_3$ if the $MM$ is busy. The process always enters the thinking state after it leaves the accessing state ($\alpha_0 = 1$). The process leaves the residual waiting state or the full wait-

ing state to enter the accessing state with probability $\beta$ if the requested $MM$ is idle and the $PE$'s request is selected; otherwise it will enter the full waiting state with probability $\bar{\beta}$. (Although $\alpha_1$ and $\beta$ are equal they are distinguished in preparation for the general case discussed later in this chapter.)

The embedded Markov chain can be solved and the $\pi$s can be represented as functions of the transition probabilities, i.e., of $BUSY$ and $WIN$. Then, from equation (4.2) the term $\lambda_1$ may be expressed as a function of $R$ and $WIN$. When the term $\lambda_1$ is substituted into equation (4.6u) the term $BUSY$ can be expressed as follows:

$$BUSY = \frac{(N-1)(\bar{C}-1)\,WIN\;R}{1+(N-1)(\bar{C}-1)\,WIN\;R} \quad . \tag{4.8u}$$

The SMP limiting probabilities can be derived by substituting the limiting probabilities of the embedded Markov chain ( $\pi$ s) into equation (4.1). Therefore, the SMP limiting probabilities can be expressed as functions of $R$ and the transition probabilities as shown below:

$$P_j = \begin{cases} \eta_0 M\;\beta\,R & j=0 \\ \eta_1 M\;\beta\,R & j=1 \\ \left\{\alpha_2 + \dfrac{(\bar{\beta})^2}{\beta}\right\}\eta_2 M\;\beta\,R & j=2 \\ \alpha_3\,\eta_3 M\;\beta\,R & j=3 \end{cases} \tag{4.9u}$$

It can be seen from these equations that there is a set of nonlinear equations to be solved. The nonlinearity is introduced because the transition probabilities are defined as functions of the limiting probabilities of the SMP; meanwhile, the limiting probabilities are defined as functions of the transition probabilities. An iterative algorithm can be used to solve these equations. The algorithm will iterate on the value of $R$ and then the performance measures of the system can be derived. The algorithm breaks down as follows:

1. The average sojourn times of the states are calculated using equation (4.3u).
2. An initial value is chosen for $R$ in the range $0 < R < 1$ (in this experiment $R = 1/M$ was used).

3. The terms *WIN* and *BUSY* are calculated using equations (4.5u) and (4.8u) respectively.

4. The transition probabilities are calculated using equation (4.7u).

5. A new value for $R$ is calculated by summing the four equations of equation (4.9u) to one. Then $R$ can be expressed as follows:

$$R = \frac{1}{\left[ \eta_0 + \left( 1 + \alpha_2 + \frac{(\bar{\beta})^2}{\beta} \right) \eta_1 + \alpha_3 \eta_3 \right] M \beta} .$$

6. Steps 3 through 5 are repeated until $R$ has the desired accuracy.[2]

The solution for $R$ may be used to calculate the limiting probabilities of the states using equation (4.9u). These can, in turn, be used to calculate the performance measures of Chapter III as follows:

$$\begin{aligned} BW &= N P_1 , \\ PU_i &= P_0 + P_1 , \\ MU_j &= \frac{N}{M} P_1 , \\ L_j &= \frac{N}{M} ( P_2 + P_3 ) \end{aligned}$$

and

$$W_{ij} = \eta_2 \left( \frac{1}{\beta} - ( 1 + BUSY ) \right) + \eta_3 BUSY ,$$

where $1 \leq i \leq N$ and $1 \leq j \leq M$. The last equation is the only one that does not follow directly from the definition of the states of Figure 4.1. It can be derived by calculating the expected value of $\tilde{W}_{ij}$ in the usual way from the pmf of $\tilde{W}_{ij}$. The pmf of $\tilde{W}_{ij}$ can be expressed as follows:

$$\begin{aligned} Pr [ \tilde{W}_{ij} = 0 ] &= \alpha_1 , \\ Pr [ \tilde{W}_{ij} = k \eta_2 ] &= \alpha_2 ( \bar{\beta} )^{k-1} \beta \qquad k \geq 1 \end{aligned}$$

and

$$Pr [ \tilde{W}_{ij} = (\eta_3 + k \eta_2) ] = \alpha_3 ( \bar{\beta} )^k \beta \qquad k \geq 0 .$$

---

[2] This is a simple fixed-point iteration scheme. Higher-order iteration schemes could be used but were found unnecessary in the experiments discussed in this dissertation. Eight iterations were usually sufficient.

The derivation of these equations proceeds as follows. The probability that the process moves from state 0 to state 1 without waiting is $\alpha_1$. The probability that the process moves from state 0 to state 1 and makes $k$ visits to state 2 is $\alpha_2 ( \overline{\beta} )^{k-1} \beta$, where $k \geq 1$. The probability the process moves from state 0 to state 1 and makes one visit to state 3 and $k$ visits to state 2 is $\alpha_3 ( \overline{\beta} )^k \beta$, where $k \geq 0$. These three cases exhaust all the possible values for $\tilde{W}_{ij}$.

As a final note it can be shown that, when $\overline{C} = \overline{C^2} = 1.0$, the SMI model reduces to the model of [Hoo77], i.e., the SMI model is reduced to equations (2.4), (2.5) and (2.6). In other words, the SMI model can be considered to be a generalization of the rate adjusted models.

### 4.3.2. Mailbox Case

In the mailbox case, the *PE*s favor a particular *MM* over the other *MM*s. This particular *MM* is referred to as the mailbox module. The first memory module, $MM_1$, is the mailbox module in this study. Hence, the multiprocessor system operation is characterized by the following assumptions:

Im. The behavior of the *PE*s can be modeled as identical stochastic processes.

IIm. The *PE*s think for an integer number of system cycles. The thinking period of any *PE* is characterized by a discrete independent random variable, $\tilde{T}$.

IIIm. Each *PE* will submit a memory request after its thinking period; requests originating from the same processing element are independent of each other. The destination memory module of the request originating from any *PE* will be determined by a discrete independent random variable, $\tilde{D}$. The request originated from any *PE* will be destined to $MM_1$ with probability $x$ or to any other *MM* with probability $(1-x)/(M-1)$.

IVm. When the first type of memory conflict occurs, the memory module equiprobably selects one of the conflicting processing elements to gain access. The blocked processing element(s) wait until the connection is completed and then resubmit their requests to the same memory module.

Vm. When the second type of memory conflict occurs, the blocked processing element(s) wait until the connection is completed and then resubmit their requests to the same memory module.

VIm. The connection time between any processing element and the mailbox module, $MM_1$, is characterized by a discrete independent random variable, $\tilde{C}_1$, while the connection time

between the processing element and non-mailbox module is characterized by a discrete independent random variable, $\tilde{C}_2$.

Appending the letter "m" to the assumption number and equation number in this section was done for convenience.

The SMP, shown in Figure 4.2, is used to describe the behavior of a *PE* in this case. The main difference between this SMP and the SMP in the uniform case is that in this case the SMP description must differentiate between the mailbox module and the other modules. In this case the state subsets are: $S^{th} = \{0\}$, $S^{ac} = \{1,2\}$, $S^{tw} = \{3,4\}$ and $S^{rw} = \{5,6\}$; states 1, 3 and 5 denote the mailbox module states while states 2, 4 and 6 denote the other (M-1) modules. The parameters that will be used in the SMI model, in this case, are as follows:

$\overline{T} \triangleq$ *first moment of* $\tilde{T}$ ,

$\overline{C}_1 \triangleq$ *first moment of connection time between any PE* and $MM_1$ ,

$\overline{C}_2 \triangleq$ *first moment of connection timebetween any PE* and *any MM other than* $MM_1$ ,

$\overline{C^2}_1 \triangleq$ *second moment of connection time between any PE* and $MM_1$ ,

$\overline{C^2}_2 \triangleq$ *second moment of connection time between any PE* and *any MM other than* $MM_1$

and

$$a_j \triangleq Pr\,[\,\tilde{D} = j\,] = \begin{cases} x & j = 1 \\ \dfrac{1-x}{M-1} & j \neq 1 \end{cases}$$

The SMP shown in Figure 4.2 can be solved using the same procedure outlined in the uniform case. The average sojourn time in the states are expressed as follows:

$$\eta_j = \begin{cases} \overline{T} & j = 0 \\ \overline{C}_1 & j = 1, 3 \\ \overline{C}_2 & j = 2, 4 \\ \dfrac{\overline{C^2}_1 - \overline{C}_1}{2(\overline{C}_1 - 1)} & j = 5 \\ \dfrac{\overline{C^2}_2 - \overline{C}_2}{2(\overline{C}_2 - 1)} & j = 6 \end{cases} \qquad (4.3m)$$

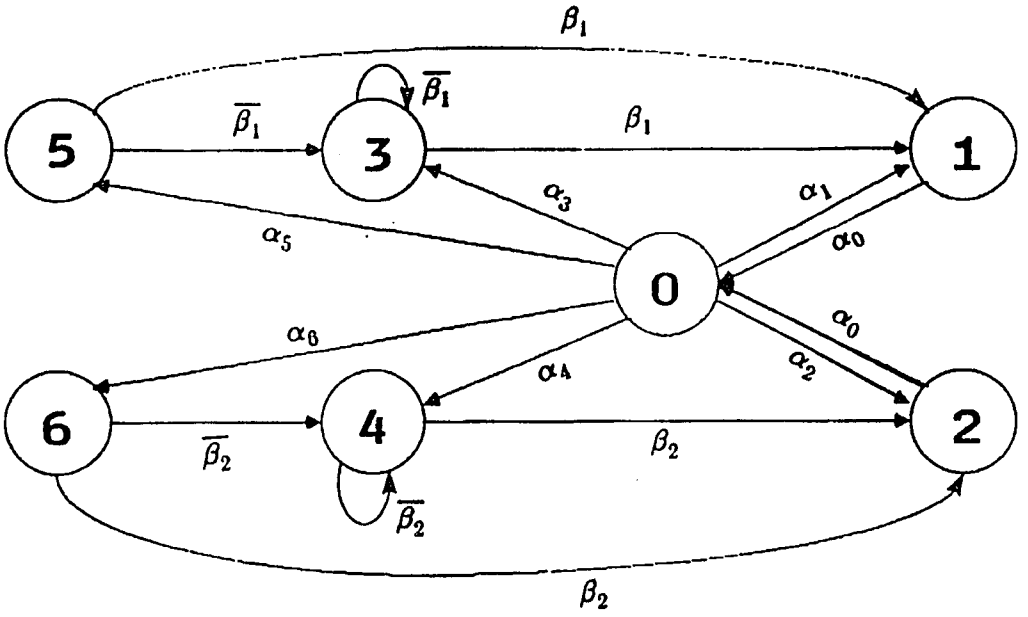



**Figure 4.2 The SMP that describes PE behavior in the mailbox case.**

The terms $R$, $WIN$ and $BUSY$ must be subscripted depending on the type of the requested memory module, i.e., whether the module is the mailbox module or not. These terms can be defined as follows:

$$R_1 \triangleq R_{ij} \qquad \forall\ i \text{ and } j = 1 \ ,$$

$$R_2 \triangleq R_{ij} \qquad \forall\ i \text{ and } j \neq 1 \ ,$$

$$WIN_1 \triangleq WIN_{ij} \qquad \forall\ i \text{ and } j = 1 \ ,$$

$$WIN_2 \triangleq WIN_{ij} \qquad \forall\ i \text{ and } j \neq 1 \ ,$$

$$BUSY_1 \triangleq BUSY_{ij} \qquad \forall\ i \text{ and } j = 1$$

and

$$BUSY_2 \triangleq BUSY_{ij} \qquad \forall\ i \text{ and } j \neq 1 \ .$$

These terms can be expressed as follows:

$$R_j = \begin{cases} x\,\lambda_0 + \lambda_3 + \lambda_5 & j = 1 \\ \frac{1}{M-1}\left[(1-x)\,\lambda_0 + \lambda_4 + \lambda_6\right] & j = 2 \end{cases} \quad (4.4m)$$

$$WIN_j = \frac{1}{N\,R_j}\left[1 - (1 - R_j)^N\right] \quad (4.5m)$$

$$BUSY_j = \begin{cases} (N-1)\,(P_1 - \lambda_1) & j = 1 \\ \left(\frac{N-1}{M-1}\right)(P_2 - \lambda_2) & j = 2 \end{cases} \quad (4.6m)$$

The transition probabilities of the SMP are expressed as functions of $BUSY$ and $WIN$ as follows:

$$\alpha_j = \begin{cases} 1 & j = 0 \\ x\,(1 - BUSY_1)\,WIN_1 & j = 1 \\ (1-x)\,(1 - BUSY_2)\,WIN_2 & j = 2 \\ x\,(1 - BUSY_1)\,(1 - WIN_1) & j = 3 \\ (1-x)\,(1 - BUSY_2)\,(1 - WIN_2) & j = 4 \\ x\,BUSY_1 & j = 5 \\ (1-x)\,BUSY_2 & j = 6 \end{cases} \quad (4.7m)$$

$$\beta_j = (1 - BUSY_j)\,WIN_j \qquad j = 1, 2$$

$$\overline{\beta_j} = 1 - \beta_j \qquad j = 1, 2$$

The embedded Markov chain of Figure 4.2 can be solved and the $\pi$s can be represented as functions of the transition probabilities, i.e., of $WIN_j$ and $BUSY_j$ where $j = 1, 2$. Thereafter, from equation (4.2) the term $\lambda_j$ may be expressed as a function of $R_j$ and $WIN_j$, where $j = 1, 2$. Substituting the term $\lambda_j$ in equation (4.6m) the term $BUSY_j$ can be expressed as follows:

$$BUSY_j = \frac{(N-1)\,(\overline{C}_j - 1)\,WIN_j\,R_j}{1 + (N-1)\,(\overline{C}_j - 1)\,WIN_j\,R_j} \quad , \quad (4.8m)$$

where $j = 1, 2$.

The SMP limiting probabilities can be obtained by substituting the limiting probabilities of the embedded Markov chain ($\pi$s) into equation (4.1). Hence, the SMP limiting probabilities can be expressed as functions of $R_1$, $R_2$ and the transition probabilities as follows:

$$P_j = \begin{cases} \frac{\eta_0}{x}\,\beta_1 R_1 & j = 0 \\ \eta_1\,\beta_1 R_1 & j = 1 \\ (M-1)\,\eta_2\,\beta_2 R_2 & j = 2 \\ \frac{\alpha_3 + \overline{\beta_1}\,\alpha_5}{x}\,\eta_3 R_1 & j = 3 \\ (M-1)\,\frac{\alpha_4 + \overline{\beta_2}\,\alpha_6}{1-x}\,\eta_4 R_2 & j = 4 \\ \frac{\alpha_5}{x}\,\eta_5\,\beta_1 R_1 & j = 5 \\ (M-1)\,\frac{\alpha_6}{1-x}\,\eta_6\,\beta_2 R_2 & j = 6 \end{cases} \qquad \textbf{(4.9m)}$$

To solve the mailbox case, a simple iterative algorithm can be utilized similar to the one used in the uniform case. In this case the algorithm iterates on the variables $R_1$ and $R_2$. The performance measures can be derived from the limiting probabilities, $P_i$s, as follows, where $1 \leq i \leq N$ and $1 \leq j \leq M$:

$$BW = N\,(P_1 + P_2)$$

$$PU_i = P_0 + P_1 + P_2$$

$$MU_j = \begin{cases} N\,P_1 & j = 1 \\ \frac{N}{M-1}\,P_2 & j \neq 1 \end{cases}$$

$$L_j = \begin{cases} N\,(P_3 + P_5) & j = 1 \\ \frac{N}{M-1}\,(P_4 + P_6) & j \neq 1 \end{cases}$$

$$W_{ij} = \begin{cases} \eta_3 \left( \frac{1}{\beta_1} - (1 + BUSY_1) \right) + \eta_5\,BUSY_1 & j = 1 \\ \eta_4 \left( \frac{1}{\beta_2} - (1 + BUSY_2) \right) + \eta_6\,BUSY_2 & j \neq 1 \end{cases}$$

### 4.3.3. Favorite Module Case

In the favorite module case, a multiprocessor system in which every *PE* favors a particular *MM* over the other *MM*s is considered. Hence, the multiprocessor system operation is

characterized by the following assumptions:

If. The behavior of the $PE$s can be modeled as stochastic processes.

IIf. The $PE$s think for an integer number of system cycles. The thinking period of any $PE$ is characterized by a discrete independent random variable, $\tilde{T}$.

IIIf. Each $PE$ will submit a memory request after its thinking period; requests originating from the same processing element are independent of each other. The destination memory module of the request originating from $PE_i$ will be determined by a discrete independent random variable, $\tilde{D}_i$. The request originated from $PE_i$, where $1 \leq i \leq N$, will be destined to $MM_i$ (the favorite memory module) with probability $x$ or to any other $MM$ with probability $(1-x)/(M-1)$.

IVf. When the first type of memory conflict occurs, the memory module equiprobably selects one of the conflicting processing elements to gain access. The blocked processing element(s) wait until the connection is completed and then they resubmit their requests to the same memory module.

Vf. When the second type of memory conflict occurs, the blocked processing element(s) wait until the connection is completed and then resubmit their requests to the same memory module.

VIf. The connection time between the processing element and its favorite module is characterized by a discrete independent random variable, $\tilde{C}_1$, while the connection time between the processing element and the nonfavorite module is characterized by a discrete independent random variable, $\tilde{C}_2$.

Appending the letter "f" to the assumption number and equation number in this section was done for convenience.

The SMP, depicted in Figure 4.3, is used to describe the behavior of a $PE$ in this case. It is noted that Figure 4.3a describes the behavior of the $PE$ that has a favorite memory module, while Figure 4.3b describes the behavior of the $PE$ that has no favorite memory module--this will happen if $N > M$. Hence, this case is a combination of the previous two cases. It is noted that the states of the SMP shown in Figure 4.3b use the same numbering system as the states in the SMP shown in Figure 4.3a. Similar to the SMPs presented earlier, the state subsets are: $S^{th} = \{0\}$, $S^{ac} = \{1,2\}$, $S^{tw} = \{3,4\}$ and $S^{rw} = \{5,6\}$; states 1, 3 and 5 denote the favorite module states, while states 2, 4 and 6 denote the other $(M-1)$ modules. Hence, the parameters that will be used in the SMI model in this case are as follows:

$\overline{T} \triangleq$ *first moment of* $\tilde{T}$ ,

$\overline{C}_1 \triangleq$ *first moment of connection time between* $PE_i$ and $MM_i$ ,

$\overline{C}_2 \triangleq$ *first moment of connection timebetween* $PE_i$ and *any MM other than* $MM_i$ ,

$\overline{C^2}_1 \triangleq$ *second moment of connection time between* $PE_i$ and $MM_i$ ,

$\overline{C^2}_2 \triangleq$ *second moment of connection time between* $PE_i$ and *any MM other than* $MM_i$

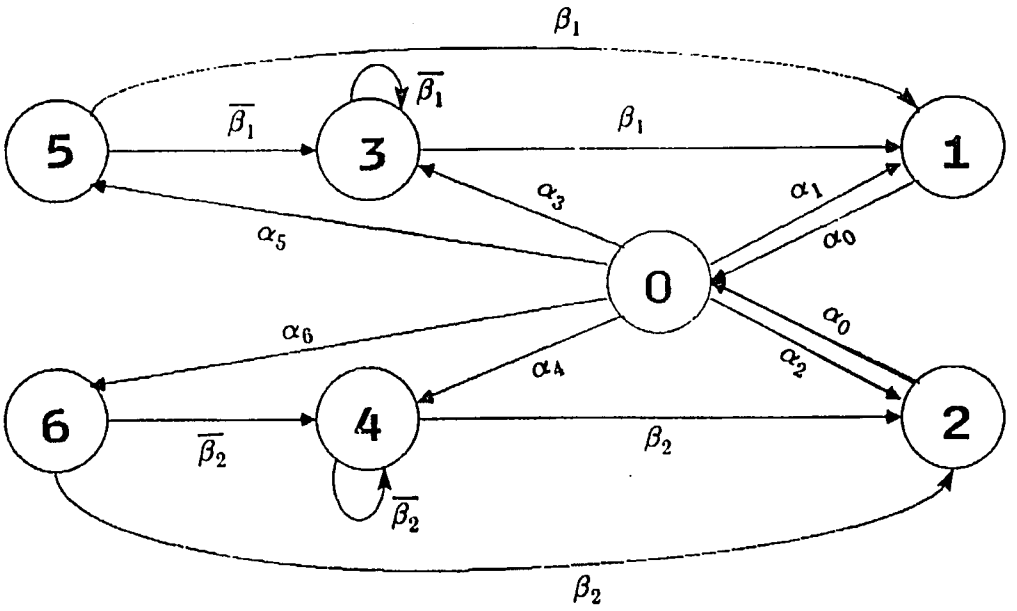
and

$$a_{ij} = Pr[\tilde{D}_i = j] = \begin{cases} x & j = i \text{ and } i \leq M \\ \frac{1-x}{M-1} & j \neq i \text{ and } i \leq M \\ \frac{1}{M} & i > M \end{cases}$$

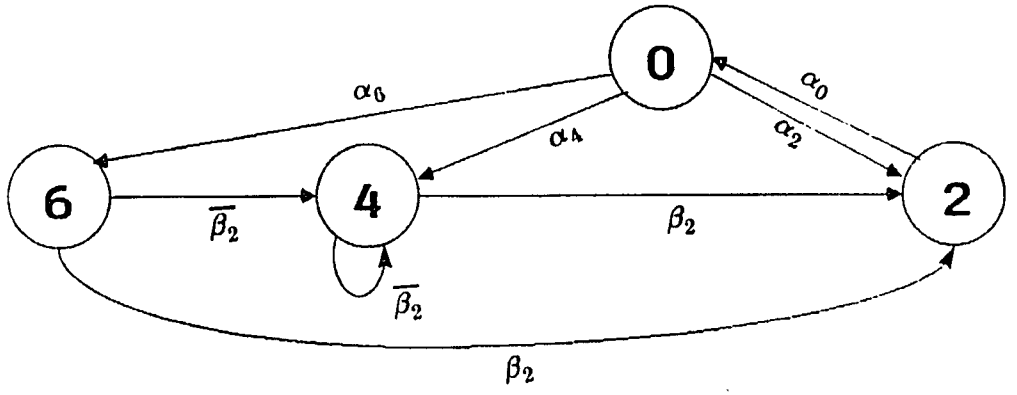
The average sojourn times in the states of the SMP can be expressed as follows:

$$\eta_j = \begin{cases} \overline{T} & j = 0 \\ \overline{C}_1 & j = 1 \\ \overline{C}_2 & j = 2 \\ \overline{C}_2 & j = 3 \\ \dfrac{R_1 \overline{C}_1 + (N-2) R_2 \overline{C}_2}{R_1 + (N-2) R_2} & j = 4 \\ \dfrac{\overline{C^2}_2 - \overline{C}_2}{2(\overline{C}_2 - 1)} & j = 5 \\ \dfrac{R_1}{R_1 + (N-2) R_2} \dfrac{\overline{C^2}_1 - \overline{C}_1}{2(\overline{C}_1 - 1)} + \dfrac{(N-2) R_2}{R_1 + (N-2) R_2} \dfrac{\overline{C^2}_2 - \overline{C}_2}{2(\overline{C}_2 - 1)} & j = 6 \end{cases} \quad (4.3f)$$

The derivation of the average sojourn times in states 0, 1, and 2 are straightforward. However, the derivations of the average sojourn times in the other states are somewhat tricky. If a process is in state 3 or 5, then the *PE* is waiting to access its own favorite module. Therefore, the connection time of the accessing *PE* is distributed as the random variable $\tilde{C}_2$. Hence, the average full connection time between the module and that accessing *PE* is $\overline{C}_2$ and the average residual connection time is $(\overline{C^2}_2 - \overline{C}_2)/2(\overline{C}_2 - 1)$. If the process is in state 4 or 6, then the *PE* is waiting to access a nonfavorite module. Therefore, it must be determined whether the accessing *PE* is favoring that module or not in order to derive the average sojourn times in states 4 and 6. The determination of the accessing *PE* is as follows. The total arrival rate to that particular module, excluding the *PE* in state 4 or 6, is $R_1 + (N-2) R_2$. The arrival rate due to the *PE* that




(a)




(b)

**Figure 4.3 The SMP that describes PE behavior in the favorite module case.**

favors that module is $R_1$. The arrival rate due to a $PE$ that does not favor that module is $R_2$, there are $N-2$ of those. Therefore, the accessing $PE$ is the one that favors that particular module with probability $R_1 / \{R_1 + (N-2)R_2\}$ and the accessing $PE$ is the one that does not favor that particular module with probability $(N-2)R_2 / \{R_1 + (N-2)R_2\}$. From this the average sojourn times of states 4 and 6 can be developed.

The size of the multiprocessor system in this case is important in the development of the SMI model. Therefore, there are three different situations in this case; namely, the distributing system, where $N < M$; the square system, where $N = M$; and the concentrating system, where $N > M$. In the distributing system each $PE$ will have a favorite module and $N-M$ modules will not be favored by any $PE$. In the square system each $PE$ will have a favorite module and every module will be favored by a $PE$. However, in the concentrating system only $M$ $PE$s will have favorite modules while the other $N-M$ $PE$s will not favor any modules. The square system is presented in this dissertation, the other two systems can be developed using the same procedure.

In this section it is assumed that $N = M$. Hence, the SMP depicted in Figure 4.3a is used to describe the behavior of the $PE$. The terms $R$, $WIN$ and $BUSY$ must be subscripted depending on the type of the requested module, i.e., whether the module is a favorite module or not. These terms can be defined as follows:

$$
\begin{aligned}
R_1 &\triangleq R_{i,j} && \forall\ i \text{ and } j = i\ , \\
R_2 &\triangleq R_{i,j} && \forall\ i \text{ and } j \neq i\ , \\
WIN_1 &\triangleq WIN_{i,j} && \forall\ i \text{ and } j = i\ , \\
WIN_2 &\triangleq WIN_{i,j} && \forall\ i \text{ and } j \neq i\ , \\
BUSY_1 &\triangleq BUSY_{i,j} && \forall\ i \text{ and } j = i
\end{aligned}
$$

and

$$
BUSY_2 \triangleq BUSY_{i,j} \qquad \forall\ i \text{ and } j \neq i\ .
$$

These terms can be expressed as follows:

$$R_j = \begin{cases} x\,\lambda_0 + \lambda_3 + \lambda_5 & j = 1 \\ \frac{1}{M-1}\left[(1-x)\lambda_0 + \lambda_4 + \lambda_6\right] & j = 2 \end{cases} \tag{4.4f}$$

$$WIN_j = \begin{cases} \frac{1}{N R_2}\left[1 - (1-R_2)^N\right] & j = 1 \\ \frac{R_1}{N R_2^2}\left[1 - (1-R_2)^N\right] + \frac{R_2 - R_1}{(N-1) R_2^2}\left[1 - (1-R_2)^{N-1}\right] & j = 2 \end{cases} \tag{4.5f}$$

$$BUSY_j = \begin{cases} (N-1)\,(P_1 - \lambda_1) & j = 1 \\ \left(\frac{N-1}{M-1}\right)(P_2 - \lambda_2) & j = 2 \end{cases} \tag{4.6f}$$

The transition probabilities of the SMP, shown in Figure 4.3a, are as follows:

$$\alpha_j = \begin{cases} 1 & j = 0 \\ x\,(1 - BUSY_1)\,WIN_1 & j = 1 \\ (1-x)(1 - BUSY_2)\,WIN_2 & j = 2 \\ x\,(1 - BUSY_1)(1 - WIN_1) & j = 3 \\ (1-x)(1 - BUSY_2)(1 - WIN_2) & j = 4 \\ x\,BUSY_1 & j = 5 \\ (1-x)\,BUSY_2 & j = 6 \end{cases} \tag{4.7f}$$

$$\beta_j = (1 - BUSY_j)\,WIN_j \qquad j = 1, 2$$

$$\overline{\beta_j} = 1 - \beta_j \qquad j = 1, 2$$

The embedded Markov chain of Figure 4.3 can be solved and the $\pi$s can be represented as functions of the transition probabilities, i.e., of $WIN_j$ and $BUSY_j$ where $j = 1, 2$. Thereafter, from equation (4.2) the term $\lambda_j$ may be expressed as a function of $R_j$ and $WIN_j$, where $j = 1, 2$. In here the term $\lambda_j$ is substituted into equation (4.6f), the term $BUSY_j$ can be expressed as follows:

$$BUSY_j = \begin{cases} \dfrac{\frac{1-x}{x}\,(\,\overline{C}_2 - 1\,)\,WIN_1\,R_1}{1 + \frac{1-x}{x}\,(\,\overline{C}_2 - 1\,)\,WIN_1\,R_1} & j = 1 \\ \dfrac{\left[\,\frac{x}{1-x}\,(M-1)\,(\,\overline{C}_1 - 1\,) + (N-2)\,(\,\overline{C}_2 - 1\,)\right]\,WIN_2\,R_2}{1 + \left[\,\frac{x}{1-x}\,(M-1)\,(\,\overline{C}_1 - 1\,) + (N-2)\,(\,\overline{C}_2 - 1\,)\right]\,WIN_2\,R_2} & j = 2 \end{cases} \tag{4.8f}$$

The SMP limiting probabilities can be obtained by solving the embedded Markov chain and then substituting the results into equation (4.1). The SMP limiting probabilities can be expressed as functions of $R_1$, $R_2$ and the transition probabilities as follows:

$$P_j = \begin{cases} \dfrac{\eta_0}{x}\,\beta_1\,R_1 & j = 0 \\ \eta_1\,\beta_1\,R_1 & j = 1 \\ (\,M-1\,)\,\eta_2\,\beta_2\,R_2 & j = 2 \\ \dfrac{\alpha_3 + \overline{\beta_1}\,\alpha_5}{x}\,\eta_3\,R_1 & j = 3 \\ (M-1)\,\dfrac{\alpha_4 + \overline{\beta_2}\,\alpha_6}{1-x}\,\eta_4\,R_2 & j = 4 \\ \dfrac{\alpha_5}{x}\,\eta_5\,\beta_1\,R_1 & j = 5 \\ (M-1)\,\dfrac{\alpha_6}{1-x}\,\eta_6\,\beta_2\,R_2 & j = 6 \end{cases} \tag{4.9f}$$

To solve the favorite module case, a simple iterative algorithm similar to the one used in the uniform case is used. In this case the algorithm iterates on the variables $R_1$ and $R_2$. The performance measures can be derived from the limiting probabilities, $P_i$s, as follows, where $1 \leq i,j \leq N$:

$$BW = N\,(P_1 + P_2)$$

$$PU_i = P_0 + P_1 + P_2$$

$$MU_j = P_1 + P_2$$

$$L_j = P_3 + P_4 + P_5 + P_6$$

$$W_{i,j} = \begin{cases} \eta_3 \left( \frac{1}{\beta_1} - (1 + BUSY_1) \right) + \eta_5\, BUSY_1 & j = i \\ \eta_4 \left( \frac{1}{\beta_2} - (1 + BUSY_2) \right) + \eta_6\, BUSY_2 & j \neq i \end{cases}$$

### 4.3.4. General Case

In the general case the system behavior is characterized by the operation assumptions outlined in Chapter III. Hence, the $PE$s are not necessarily identical and the random variables $\tilde{T}_i$, $\tilde{D}_i$ and $\tilde{C}_{i,j}$ may have any distribution. However, the full distribution of $\tilde{T}_i$, $\tilde{D}_i$ and $\tilde{C}_{i,j}$ are not needed by the SMI model. The input parameters of the SMI model are as follows:

$$\overline{T}_i \triangleq \textit{the first moment of } \tilde{T}_i \ ,$$

$$\overline{C}_{i,j} \triangleq \textit{the first moment of } \tilde{C}_{i,j} \ ,$$

$$\overline{C^2}_{i,j} \triangleq \textit{the second moment of } \tilde{C}_{i,j}$$

and

$$a_{i,j} \triangleq Pr\,[\,\tilde{D}_i = j\,] \ .$$

The SMP, shown in Figure 4.4, describes the behavior of $PE_i$, where $1 \leq i \leq N$. As described earlier, the SMI model uses these $N$ SMPs to approximate the behavior of the multiprocessor system. The states of the SMP, in this case, are partitioned into four subsets: $S_i^{th} = \{0\}$, $S_i^{ac} = \{1, \ldots, M\}$, $S_i^{fw} = \{M+1, \ldots, 2M\}$, and $S_i^{rw} = \{2M+1, \ldots, 3M\}$. The process enters state 0 and remains there for a duration of time with mean value $\eta_{i,0}$, equivalent to the thinking time of $PE_i$. A memory request, made by $PE_i$, is modeled by the SMP leaving state 0. The destination state depends on the state of the requested $MM$. For instance, if the requested module is

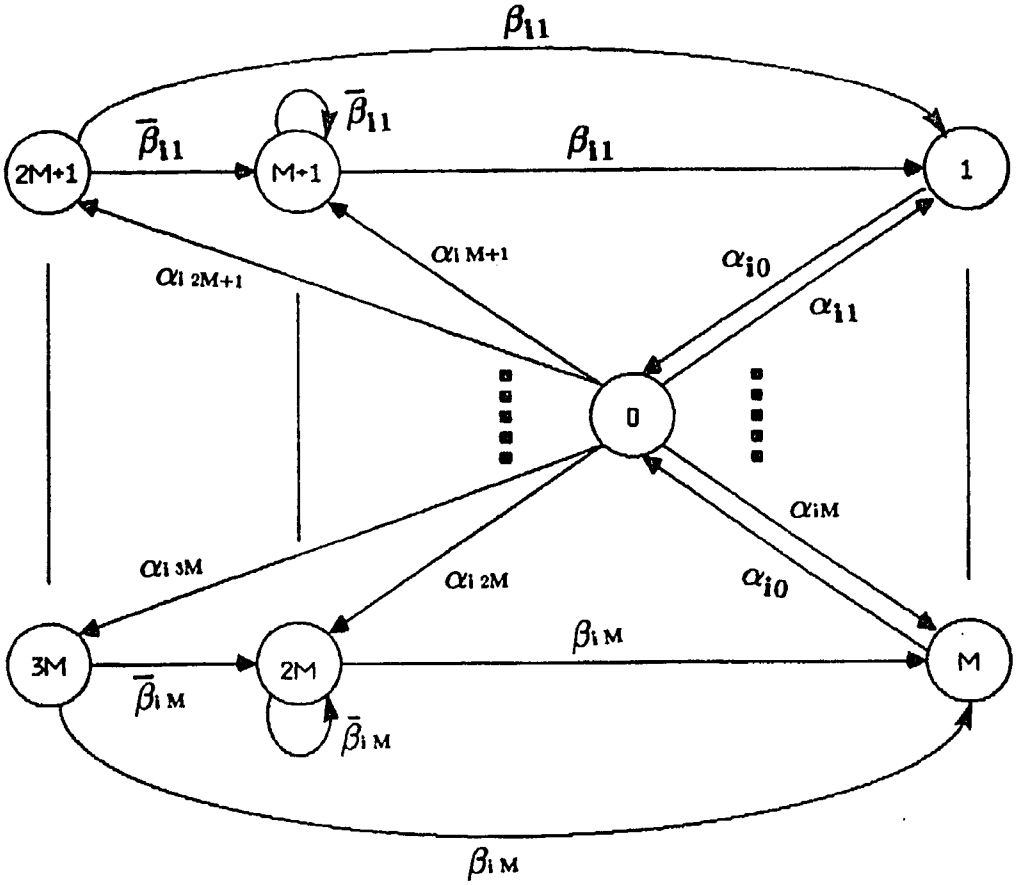



**Figure 4.4 The SMP that describes PE behavior in the general case.**

$MM_j$, where $1 \leq j \leq M$, three situations may take place. The first situation is that the process enters state $j$ if $PE_i$ obtains the connection with $MM_j$; the process remains there for a duration of time with mean value $\eta_{i,j}$ equivalent to the connection time between $PE_i$ and $MM_j$. From state $j$ the process returns to state 0, i.e., $PE_i$ resumes thinking after it has completed its memory access. The second situation is that the process leaves state 0 for state $M+j$ if $PE_i$ requested $MM_j$ simultaneously with at least one other $PE$ and $PE_i$ fails to be selected by $MM_j$, i.e., a type one conflict occurs and another $PE$ is selected to have access to $MM_j$. In this case $PE_i$ must wait for the full duration of the connection time between the selected $PE$ and $MM_j$; this duration has a mean value of $\eta_{i,\,j+M}$. The processing element $PE_i$ will retry to access the

module $MM_j$ when that module is released. If $PE_i$ succeeds, the process enters state $j$, otherwise the process reenters state $M+j$. The third situation is that the process leaves state 0 for state $2M+j$ if $PE_i$ requests $MM_j$ when it is busy, i.e., a type two conflict occurs. The processing element $PE_i$ must wait for the residual connection time before retrying to access $MM_j$; the mean value for the residual connection time is $\eta_{i,\,2M+j}$ and it is obtained through the result of Theorem 1. The process then enters state $j$ if $PE_i$ succeeds in accessing $MM_j$, or it enters state $M+j$ if it fails to obtain the connection.

Under the uniform case assumptions, the symmetry between the memory modules allows lumping of the equivalent states of the SMP of Figure 4.4 without loosing any information from the model. The same technique is used in the mailbox and the favorite memory module cases to produce the SMPs shown in Figures 4.3 and 4.4, respectively. Hence, in the uniform case resubmitting a rejected request to any module is not an approximation by itself, as reported in [YPD82].

The solution in the general case follows the lines of earlier cases. The average sojourn times in the states are as follows:

$$\eta_{ij} = \begin{cases} \overline{T}_i & j \in S_i^{th} \\ \overline{C}_{ij} & j \in S_i^{ac} \\ \displaystyle\sum_{\substack{k=1 \\ k \neq i}}^{N} \frac{R_{k\,j-M}}{\displaystyle\sum_{\substack{l=1 \\ l \neq i}}^{N} R_{l\,j-M}} \; \overline{C}_{k\,j-M} & j \in S_i^{fw} \\ \displaystyle\sum_{\substack{k=1 \\ k \neq i}}^{N} \frac{\left( P_{k\,j-2M} - \lambda_{k\,j-2M} \right)}{\displaystyle\sum_{\substack{l=1 \\ l \neq i}}^{N} \left( P_{l\,j-2M} - \lambda_{l\,j-2M} \right)} \; \frac{\overline{C^2}_{k\,j-2M} - \overline{C}_{k\,j-2M}}{2\left( \overline{C}_{k\,j-2M} - 1 \right)} & j \in S_i^{rw} \end{cases} \tag{4.3}$$

The first subscript indicates the $PE$ and the second, indicates the state. The calculations of the average sojourn time in the states of the thinking and accessing subsets, $S_i^{th}$ and $S_i^{ac}$, are straightforward. However, the calculation of the average sojourn time in the other states needs more elaboration. To calculate the average sojourn time of a state in the full waiting subsets, $S_i^{fw}$, we will condition on the processing element which got the connection when a memory conflict of type one

occurs. Hence, to reexamine the expression for $\eta_{i,j}$, where $j \in S_i^{rw}$ and $1 \leq i \leq N$, the term $\left( R_{i,j} \;/\; \sum_{\substack{l=1 \\ l \neq i}}^{N} R_{l,j-M} \right)$ is the probability that $PE_k$ obtains the connection to $MM_j$, i.e., the fraction of the request rate of $PE_k$ to the total request rate for $MM_j$ excluding $PE_i$ request rate. The second term $\overline{C}_{kj}$ is the average connection time between $PE_k$ and $MM_j$. In the same way, to calculate the average sojourn time of a state in the residual waiting subset, $S^{rw_i}$, we will condition on the processing element which is accessing $MM_j$ when the memory conflict of type two occurs. The above expression for $\eta_{i,j}$, where $j \in S_i^{rw}$ and $1 \leq i \leq N$, has two terms: the first is the probability that $PE_k$ is accessing $MM_j$ given that $MM_j$ is being accessed by some $PE$ other than $PE_i$; the second term is the average residual time of the connection between $PE_k$ and $MM_j$ obtained from Theorem 1.

The terms $R$, $WIN$ and $BUSY$ also require subscripts: the first indicates the $PE$ and the second indicates the $MM$. These terms can be expressed as follows:

$$R_{i,j} = a_{i,j}\,\lambda_{i,0} + \lambda_{i,j+M} + \lambda_{i,j+2M} \tag{4.4}$$

and

$$WIN_{i,j} = \sum_{k=1}^{N} \frac{1}{k}\,\Psi_{i,j,k} \quad , \tag{4.5}$$

where

$$\Psi_{i,j,k} = \sum_{l=1}^{\binom{N-1}{k-1}} \prod_{\substack{h=1 \\ h \neq i}}^{N} w_{i,j,k,l}(h) \quad ,$$

and

$$w_{i,j,k,l}(h) = \begin{cases} R_{hj} & \text{if } PE_i,\ PE_h \text{ and } (k-2) \text{ other } PE\text{'s request } MM_j \text{ in the } l^{th} \text{ case} \\ 1 - R_{hj} & \text{otherwise} \end{cases}$$

$$BUSY_{i,j} = \sum_{\substack{k=1 \\ k \neq i}}^{N} \left( P_{kj} - \lambda_{kj} \right) = \sum_{\substack{k=1 \\ k \neq i}}^{N} \left( \overline{C}_{kj} - 1 \right) \lambda_{kj} \quad . \tag{4.6}$$

The transition probabilities between the states of the SMP can be defined as follows:

$$\alpha_{i,j} = \begin{cases} 1 & j \in S_i^{th} \\ a_{i,j}\ (1 - BUSY_{i,j})\ WIN_{i,j} & j \in S_i^{ac} \\ a_{i,j-M}\ (1 - BUSY_{i,j-M})\ (1 - WIN_{i,j-M}) & j \in S_i^{fw} \\ a_{i,j-2M}\ BUSY_{i,j-2M} & j \in S_i^{rw} \end{cases} \tag{4.7}$$

$$\beta_{i,j} = WIN_{i,j}\ (1 - BUSY_{i,j}) \qquad j \in S_i^{ac}$$

$$\bar{\beta}_{i,j} = 1 - \beta_{i,j} \qquad j \in S_i^{ac}$$

The embedded Markov chain can be solved and the $\pi$s can be represented as functions of transition probabilities. Then, from equation (4.2), $\lambda_{kj}$ may be defined as a function of $R$ and the transition probabilities. Therefore, using equation (4.6) the term $BUSY_{i,j}$ can be expressed as follows:

$$BUSY_{i,j} = \sum_{\substack{k=1 \\ k \neq i}}^{N} \left( \bar{C}_{kj} - 1 \right) \beta_{kj}\ R_{kj} \tag{4.8}$$

Furthermore, the SMP limiting probabilities can be expressed as functions of $R$ and the transition probabilities as shown below:

$$P_{i,j} = \begin{cases} \eta_{i,j} \sum_{k=1}^{M} \beta_{ik}\ R_{ik} & j \in S_i^{th} \\ \eta_{i,j}\ \beta_{i,j}\ R_{i,j} & j \in S_i^{ac} \\ \left( \alpha_{i,j} + a_{i,j-M} \dfrac{\left(\bar{\beta}_{i,j-M}\right)^2}{\beta_{i,j-M}} \right) \dfrac{\eta_{i,j}\ \beta_{i,j-M}}{a_{i,j-M}}\ R_{i,j-M} & j \in S_i^{fw} \\ \dfrac{\alpha_{i,j}\ \eta_{i,j}}{a_{i,j-2M}}\ \beta_{i,j-2M}\ R_{i,j-2M} & j \in S_i^{rw} \end{cases} \tag{4.9}$$

To solve the general case, an iterative algorithm can be used similar to that proposed for the uniform case. In this case the algorithm iterates on the set of variables $R_{i,j}$, where $1 \leq i \leq N$ and $1 \leq j \leq M$. The performance measures can be derived from the SMPs limiting probabilities as follows:

$$BW = \sum_{i=1}^{N} \sum_{j=1}^{M} P_{ij}$$

$$PU_i = 1 - \sum_{j=M+1}^{3M} P_{ij}$$

$$MU_j = \sum_{i=1}^{N} P_{ij}$$

$$L_j = \sum_{i=1}^{N} \left( P_{i,j+M} + P_{i,j+2M} \right)$$

$$W_{ij} = \eta_{i,M+j} \left( \frac{1}{\beta_{ij}} - (1 + BUSY_{ij}) \right) + \eta_{i,2M+j} \, BUSY_{ij}$$

# CHAPTER V

# SIMULATION RESULTS

## 5.1. Introduction

In this chapter some hypothetical examples obtained in an effort to verify and validate the SMI model are described. In these examples the results from the simulations are compared with the results from the SMI model and the relative percentage error between them is calculated. Furthermore, these results are used to obtain quantitative relationships between the different parameters of the model. For instance, the relation between the second moment of the connection time and the queueing time can be concluded. The simulation was supplied by artificially generated address sequences. The simulation programs were written with the SIMSCRIPT II.5 simulation language. The simulation programs were run twice each time for at least 40,000 cycles. The simulation results are gathered every 2000 cycles and then averaged at the end of the simulation. The result of the simulation are compared with the results of the model by calculating the relative percentage error, $\%Error$. The term $\%Error$ is defined as follows:

$$\%Error = \frac{measure\ from\ the\ model - measure\ from\ the\ simulation}{measure\ from\ the\ simulation} \times 100 \quad .$$

In the simulation, different distributions of the think time were used to demonstrate the fact that the higher moment of the think time will not have significant contributions to the performance measures, see [BaS76] for discussion.

A multiprocessor system of size 8 × 8 or 32 × 32 is assumed in these examples. The input parameters of the model will be altered and their impact on the multiprocessor system behavior will be watched. In the next sections, the results obtained in four different cases are presented. These cases are: the uniform case, the mailbox case, the favorite module case, and the general case. The examples in the following sections are not exhaustive tests of the model. Nevertheless, they demonstrate the model accuracy under different operation assumptions.

## 5.2. The Uniform Case

In this section, four examples of a multiprocessor system whose behavior is characterized by the uniform case assumptions, i.e., assumptions Iu through VIu outlined in Section 4.3.1 are studied. The first two examples assume that the system is an 8 × 8 multiprocessor system, while the other two examples assume that the system is a 32 × 32 multiprocessor system. The analysis of the two systems with different sizes enables the effect of the size of the system on the performance measures to be examined.

**Example 1.1:**

In this example a multiprocessor system that has eight $PE$s and eight $MM$s is considered. The multiprocessor system operation can be characterized by the uniform case assumptions. The average connection time between any $PE$ and an $MM$, $\overline{C}$, is equal to four cycles. In this example, the average think time, $\overline{T}$, is varied from zero to ten cycles. Moreover, the constant of variation of the connection time between any $PE$ and the $MM$ will be varied from zero to two. Figure 5.1 shows the simulation results of the performance measures: $BW$, $PU_i$, $L_j$, and $W_{ij}$ as functions of $\overline{T}$ and $C_v$. The memory module utilization, $MU_j$, is not included with these performance measures because it is easy to obtain from the memory bandwidth. Due to symmetry in the system $MU_j = BW/M$, where $1 \leq j \leq M$.

Figure 5.1(a) shows that increasing the variation in the connection time between the $PE$ and the $MM$ will decrease the memory bandwidth, $BW$, of the multiprocessor system. Figure

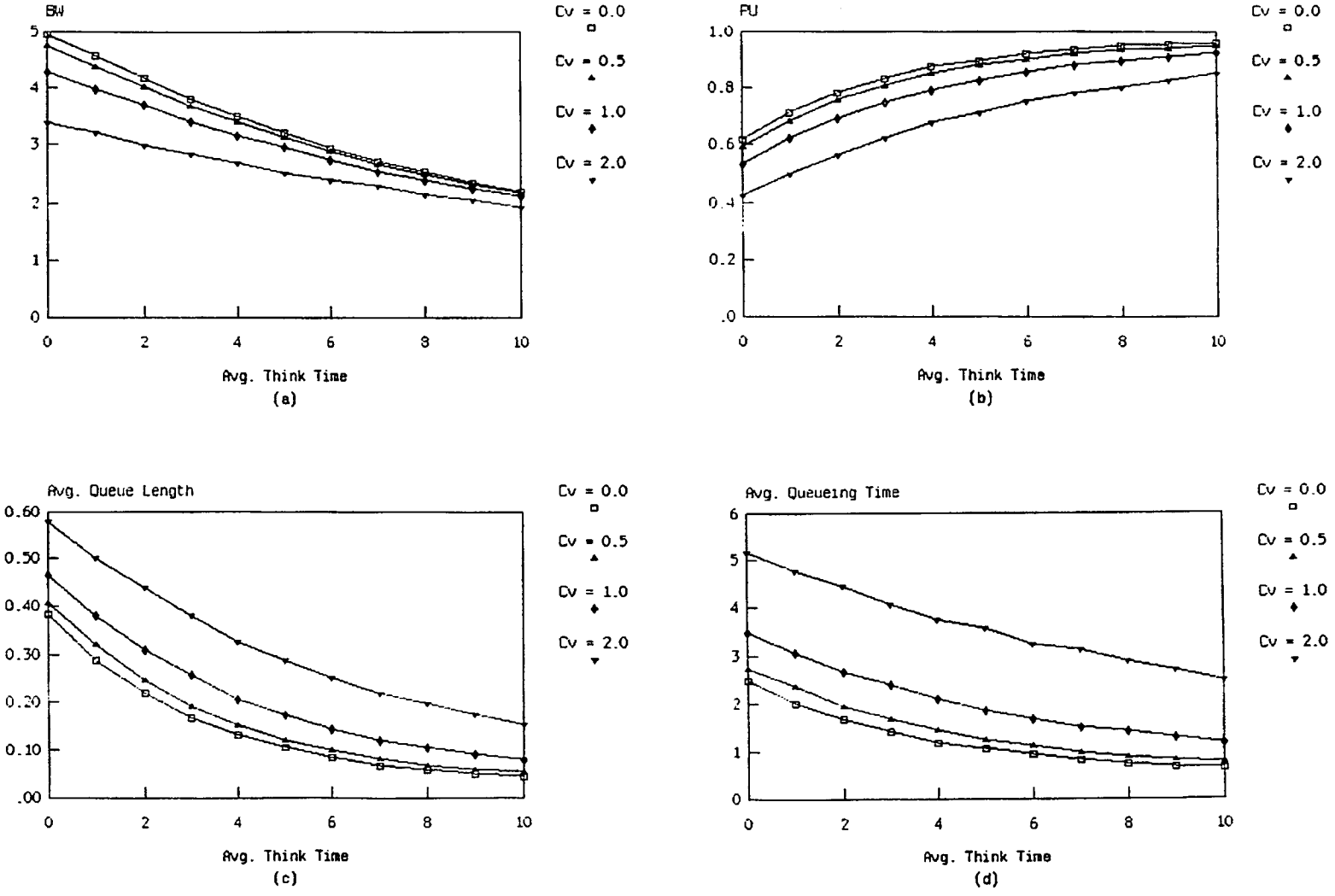



**Figure 5.1 The simulation results of Example 1.1.**

5.1(b) illustrates the same relationship between the variation in the connection time and the processor utilization, $PU_i$. However, Figures 5.1(c) and 5.1(d) show that increasing the variation in the connection time yields an increase in the average queue length, $L_j$, and in the average waiting time in the queue, $W_{ij}$. Moreover, it can be seen that the average waiting time in the queue is the most sensitive measure to the variation in the connection time. For instance, the waiting time of any $PE$ when $C_v = 2.0$ is approximately twice its waiting time when $C_v = 0$. This relation between the variation in the connection time and the different performance measures are similar to the behavior of an M/G/1 queue. In the M/G/1 queue the average queue length can be expressed as follows:

$$L_q = \lambda^2 \frac{(\bar{S})^2 + \sigma_S^2}{2(1 - \lambda \bar{S})} \quad .$$

and the average queueing time in the queue can be expressed as follows:

$$W_q = \frac{L_q}{\lambda} \quad ,$$

where $\lambda$ is the rate of the arrival to the queue, $\bar{S}$ is the average service time in the queue, and $\sigma_S^2$ is the variance of the service time. For more discussion on the M/G/1 queue see [GrH74] and [Kle75].

Figure 5.2 illustrates the relative percentage error, $\%Error$, between the simulation results of Figure 5.1 and the results obtained from the SMI model. It can be seen that the utilization measures, i.e., $BW$ and $PU$, were predicted more accurately than the average queue length or the average queueing time. Furthermore, the utilization measures were underestimated for low $C_v$s and overestimated for high $C_v$s. On the other hand, the average queue length and the average queueing time were overestimated for low $C_v$s and underestimated for high $C_v$s.

Figure 5.3 shows the relative percentage error between the simulation results of Figures 5.1(a,b) and the results obtained from the equivalent rate (ER) model of [MuA84]. The comparisons between Figures 5.2(a,b) and 5.3(a,b) illustrates the advantage of the second moment multi-unit connection models, see Section 2.3.2.2, over the first moment multi-unit connection models,

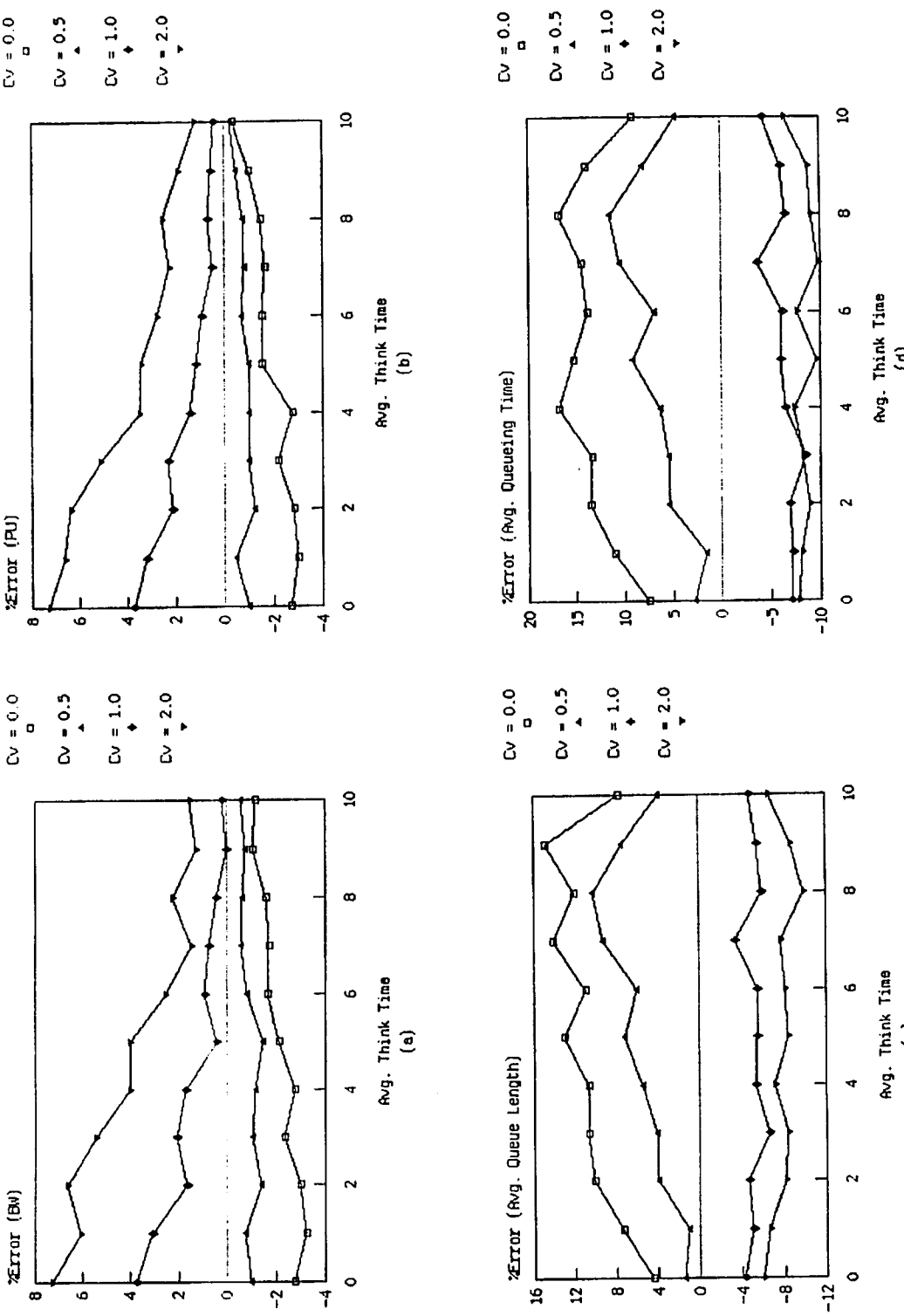



**Figure 5.2 The relative percentage error of Example 1.1 of the SMI model.**

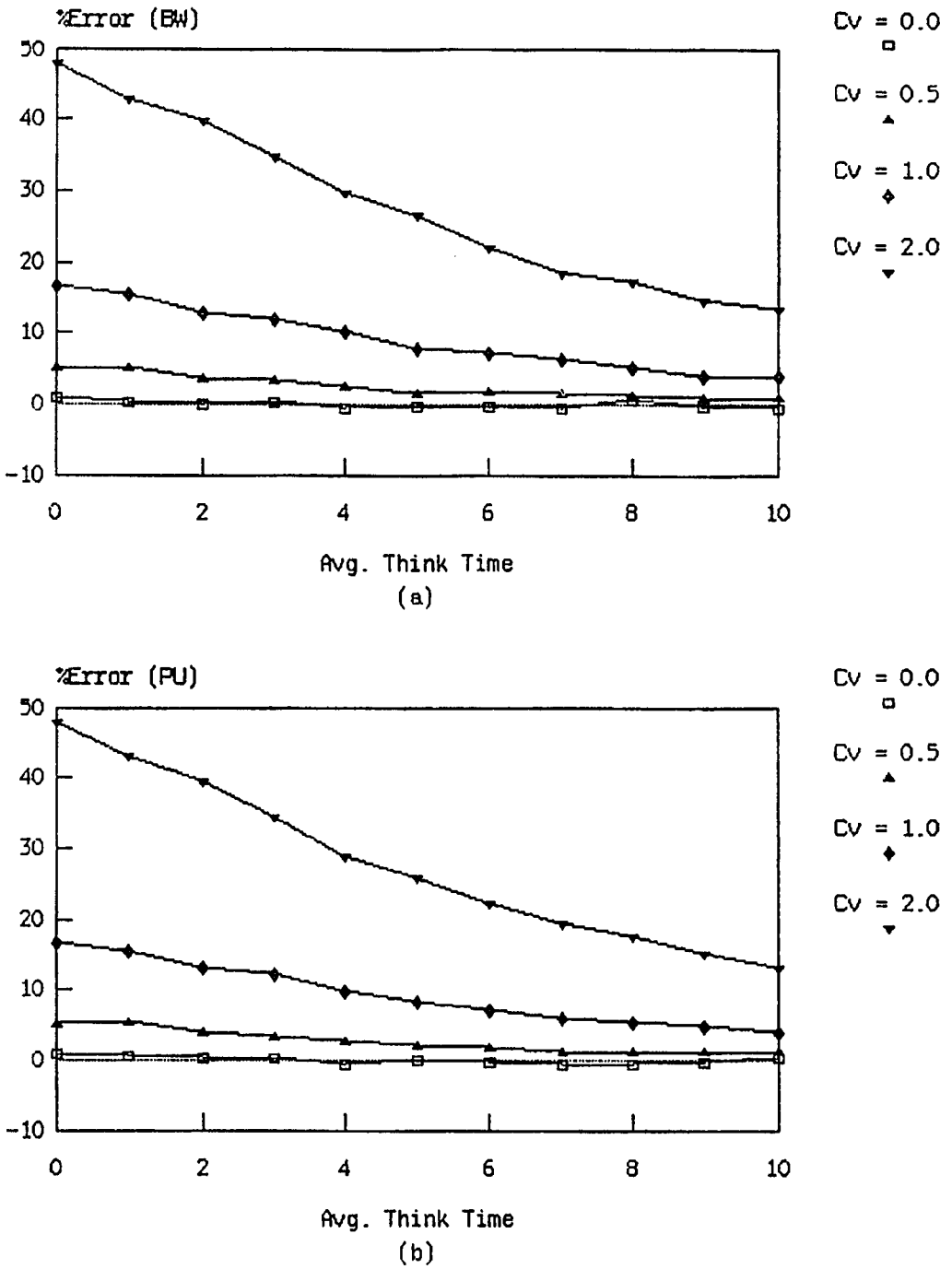



**Figure 5.3 The relative percentage error of Example 1.1 of the ER model.**

see Section 2.3.2.1. Ignoring the second moment of the connection time in the ER model yields a large relative percentage error, particularly in the cases where the variation in the connection time is high. Nevertheless, the ER model produces "accurate" results when the connection time is deterministic, i.e., $C_v = 0$.

It is clear from the results of this example that it is better to keep the connection between the $PE$ and the $MM$ to be deterministic. However, introducing a "little" variation to the connection time ($C_v \leq 0.5$) for certain applications will degrade slightly the behavior of the system. The behavior will be degraded rapidly as more variation is introduced to the system. ∎

**Example 1.2:**

This example reconsiders the multiprocessor system of Example 1.1. However, in this example it is assumed that the average connection time between any $PE$ and the $MM$, $\overline{C}$, is equal to eight cycles. This change enables us to study the effect of the first moment of the connection time between the $PE$ and the $MM$. Similar to Example 1.1, the performance measures of the system are studied as functions of $\overline{T}$ and $C_v$.

Figure 5.4 shows the simulation results of the multiprocessor system. The relationships between the different performance measures and $\overline{T}$ or $C_v$ are similar to the relationships deduced from Figure 5.1 of the previous example. However, comparing the simulation results of Figures 5.4 and 5.1 illustrates the effect of the first moment of the connection time on the different performance measures. This effect can be summarized as follows. First, the memory bandwidth, $BW$, increases as the average connection time decreases. This is particularly true when the average think time is more than zero. In the case that the average think time is zero, the memory bandwidth will not be affected by any changes in the first moment of the connection time. Second, the processor utilization, $PU_i$, decreases as the average connection time increases particularly when the average think time is more than zero. Third, the average queue length increases as the average connection time increases particularly when the average think time is more than zero. Fourth, the average queueing time increases as the average connection time increases regardless of

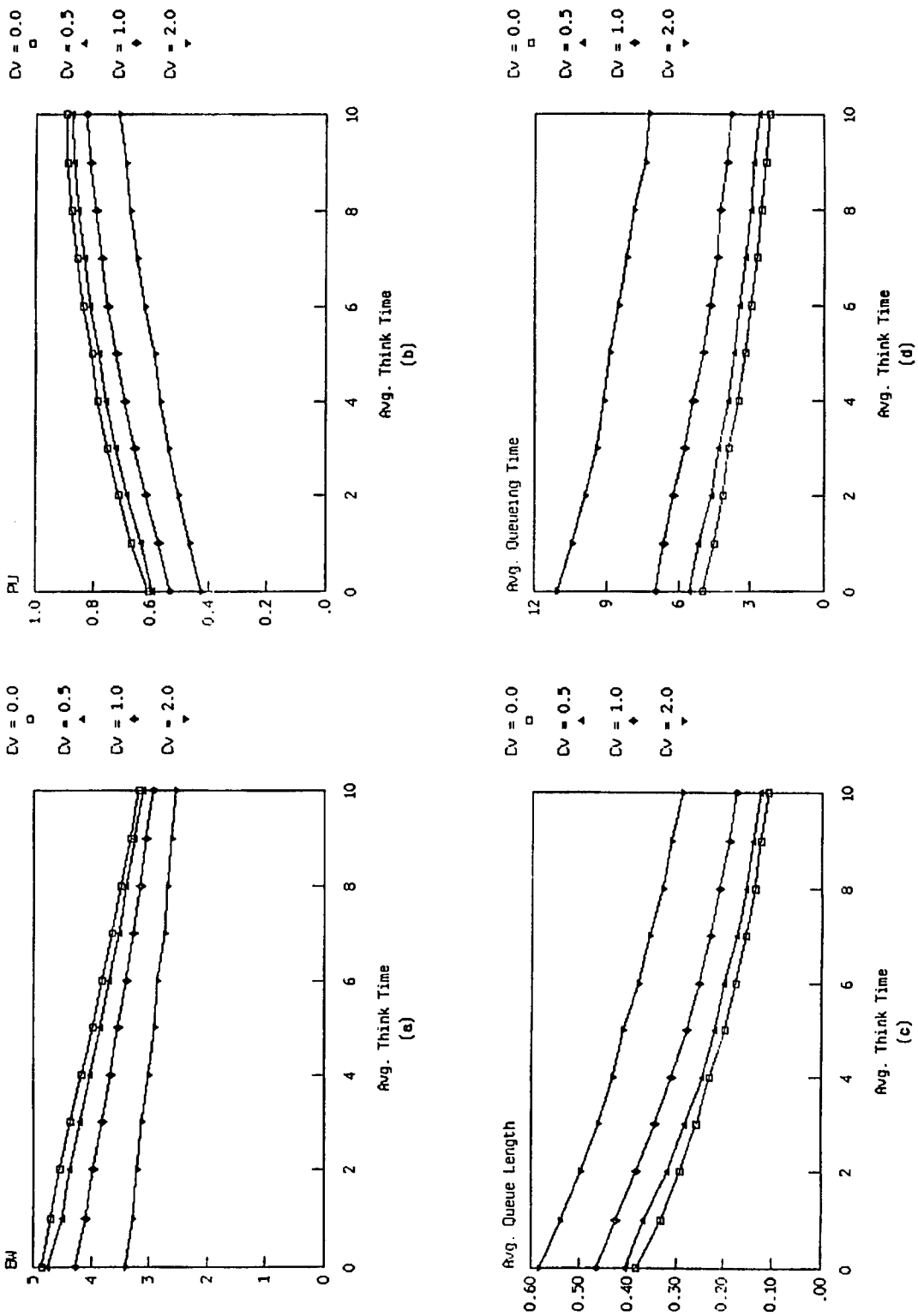



**Figure 5.4 The simulation results of Example 1.2.**

the average think time. Actually, the average queueing time is the most sensitive measure to the changes in the average connection time. The comparison between Figures 5.1(d) and 5.4(d) indicates that doubling the average connection time between any *PE* and an *MM* almost doubles the average queueing time of the *PE*. The effect of the average connection time on the average queue length and the average queueing time is similar to the effect of the average service time on the average queue length and the average queueing time in an M/G/1 queue.

Figure 5.5 illustrates the relative percentage error of these performance measures of the SMI model. The accuracy of the SMI model is similar to the accuracy obtained in the last example. Figure 5.6 shows the relative percentage error of the ER model. This example again illustrates the importance of including the second moment of the connection time in the memory interference model in order to produce an "accurate" prediction of the behavior of the multiprocessor system. ∎

It is clear from these two examples that all the performance measures of the multiprocessor system must be examined before stating anything about the system performance. For instance, increasing the average connection time between the processing element and the memory module will upgrade the performance of the multiprocessor system if only the memory bandwidth is considered as the performance measure. However, increasing the average connection time will degrade the performance of the system if the processor utilization, the average queue length, and the average queueing time are considered as the performance measures of the system.

In the next two examples, a $32 \times 32$ multiprocessor system operating under the same assumptions as the previous two examples is considered. By examining the results of these four examples, the effect of the size of the multiprocessor system on the performance measures of the system can be deduced.

**Example 1.3:**

In this example a multiprocessor system that has 32 *PE*s and 32 *MM*s is considered. The multiprocessor operations can be characterized by the uniform case assumptions. The average

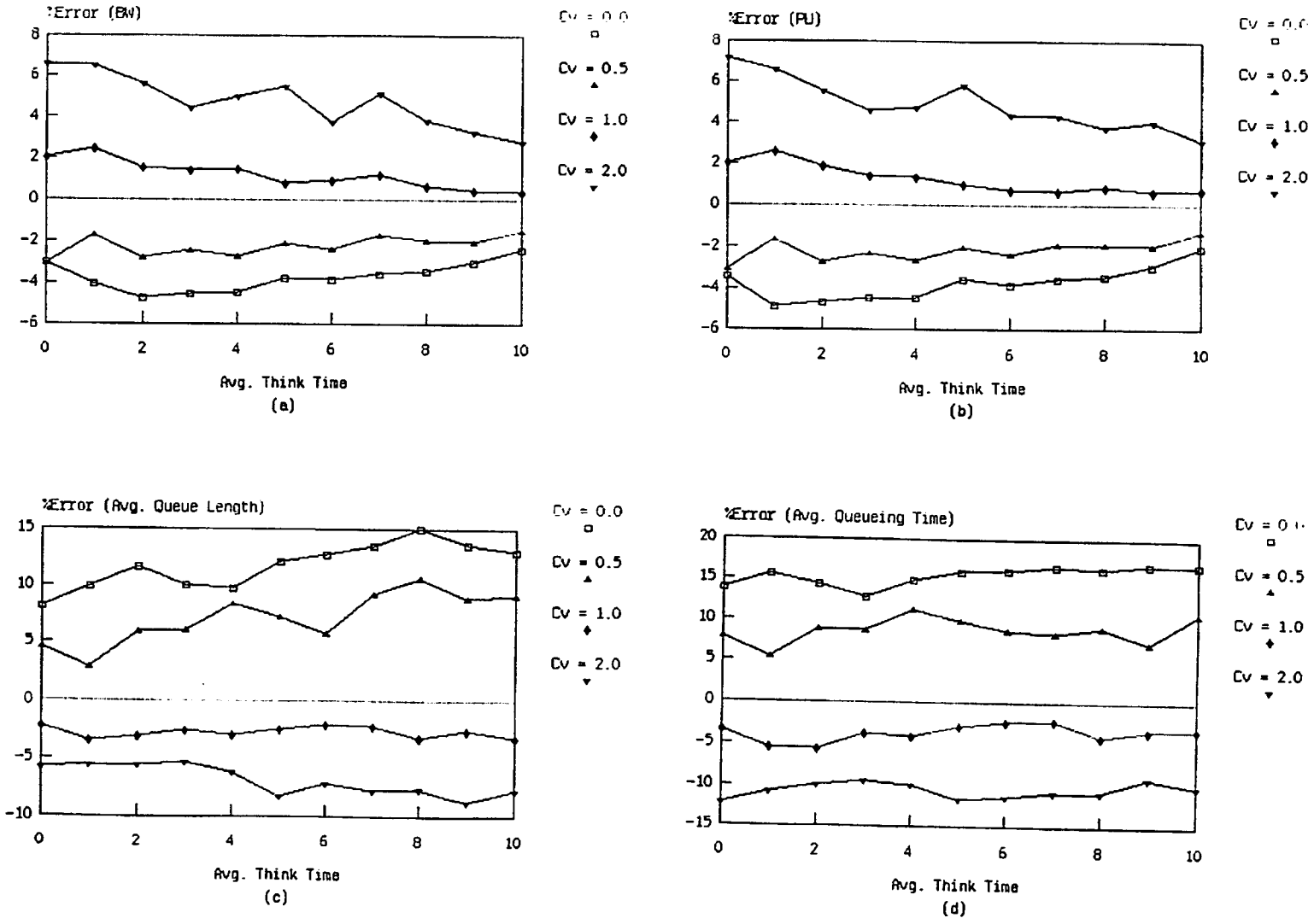



**Figure 5.6 The relative percentage error of Example 1.2 of the ER model.**

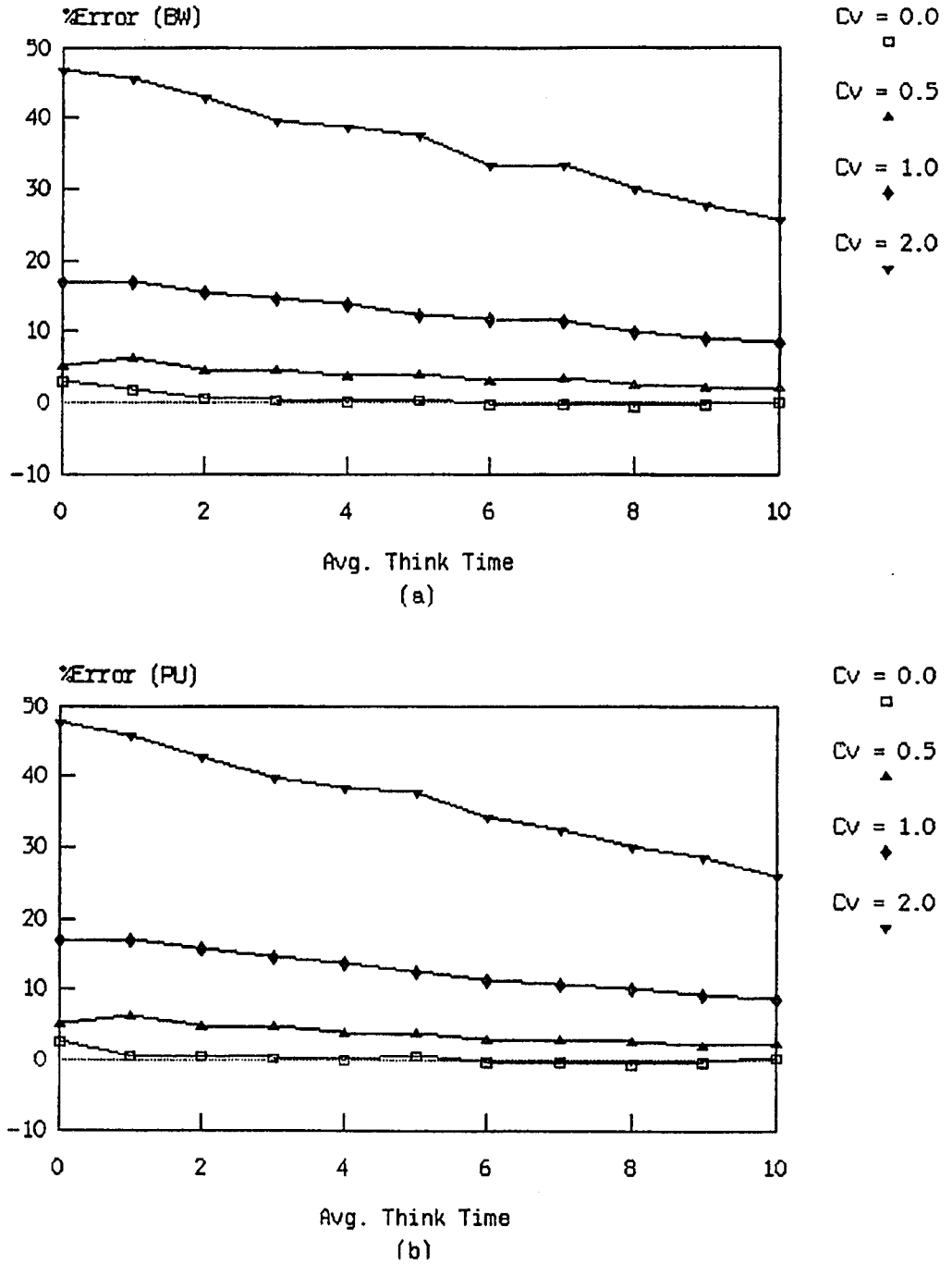



**Figure 5.6 The relative percentage error of Example 1.2 of the ER model.**

connection time between any $PE$ and the $MM$, $\overline{C}$, is equal to four cycles. Similar to the previous two examples, the average think time, $\overline{T}$, is varied from zero to ten cycles. Moreover, the constant of variation of the connection time, $C_v$, is varied from zero to two.

Figure 5.7 shows the simulation results of the performance measures: $BW$, $PU_i$, $L_j$, and $W_{ij}$ as functions of $\overline{T}$ and $C_v$. Comparing these results with the ones presented in Figure 5.1 shows that the only measure that was affected is the memory bandwidth, while the other measures did not change significantly. This is because the multiprocessor system stayed as a square system ($N = M$). However, if the size of the system is changed such that the multiprocessor system becomes a concentrator ($N > M$) or a distributor ($N < M$), the performance measures of the system will be affected definitely. For instance, in a concentrator system the average queue length and the average queueing time will be greater than their values in the square system under the same operating conditions.

Figure 5.8 shows the relative percentage error of the results obtained by the SMI model. The accuracy of the model in this example is similar to the previous two examples. Hence, the change in the size of the multiprocessor system had no effect on the accuracy of the model.

Figure 5.9 shows the relative percentage error of the results obtained by the ER model. Similar to the previous examples, it is clear that the exclusion of the second moment of the connection time will produce erroneous results. ∎

**Example 1.4:**

This is the last example of the uniform case examples. In this example the $32 \times 32$ multiprocessor system of Example 1.3 is reconsidered. However, in this example it is assumed that the average connection time is equal to eight cycles. Here again the simulation results are presented as functions of $\overline{T}$ and $C_v$.

Figure 5.10 shows the simulation results of the performance measures. Comparing the results of Figure 5.10 with those of Figure 5.4 indicates that the memory bandwidth is the only performance measure that changed significantly as a result of changing the size of the multipro-

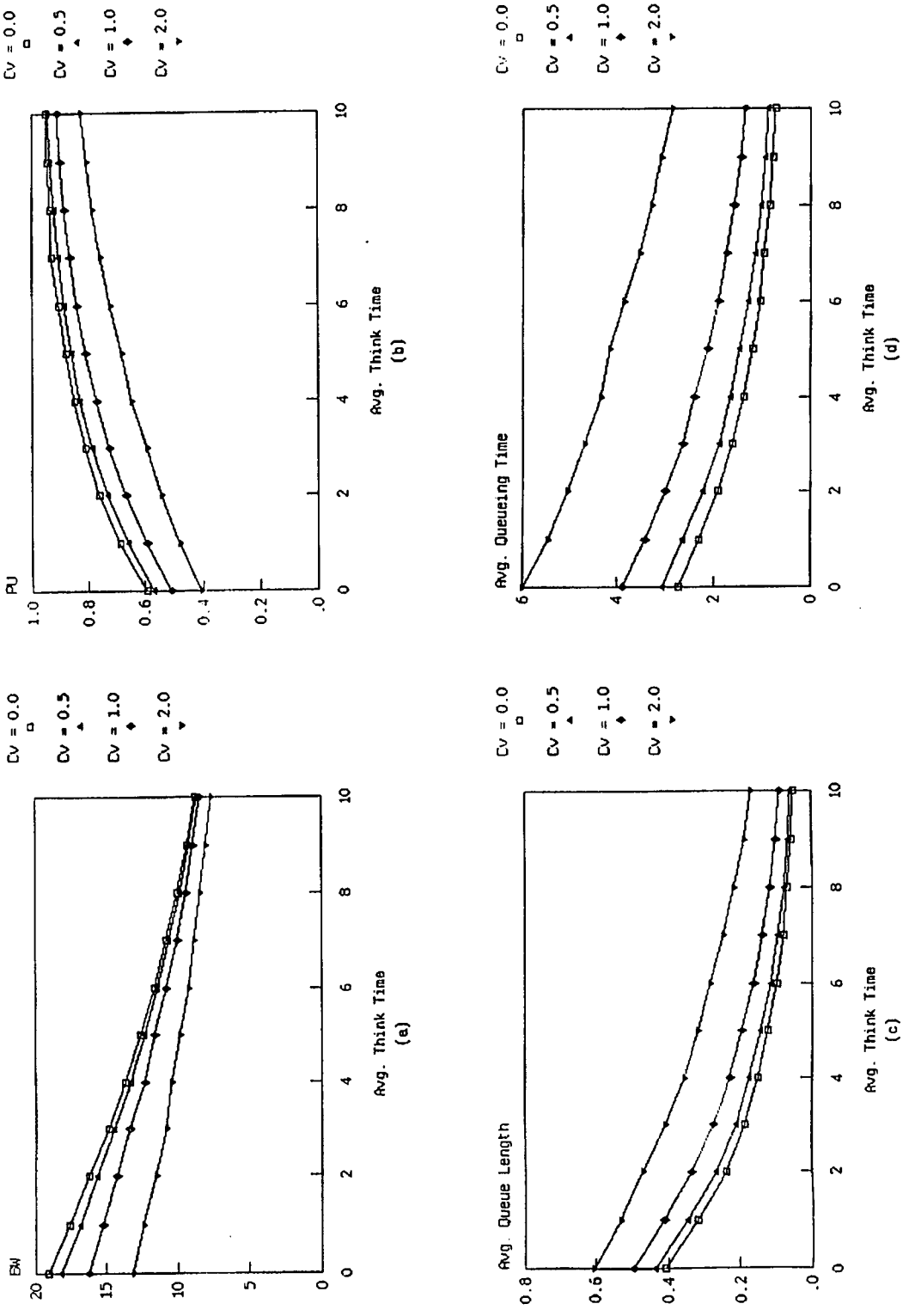



**Figure 5.7 The simulation results of Example 1.3.**

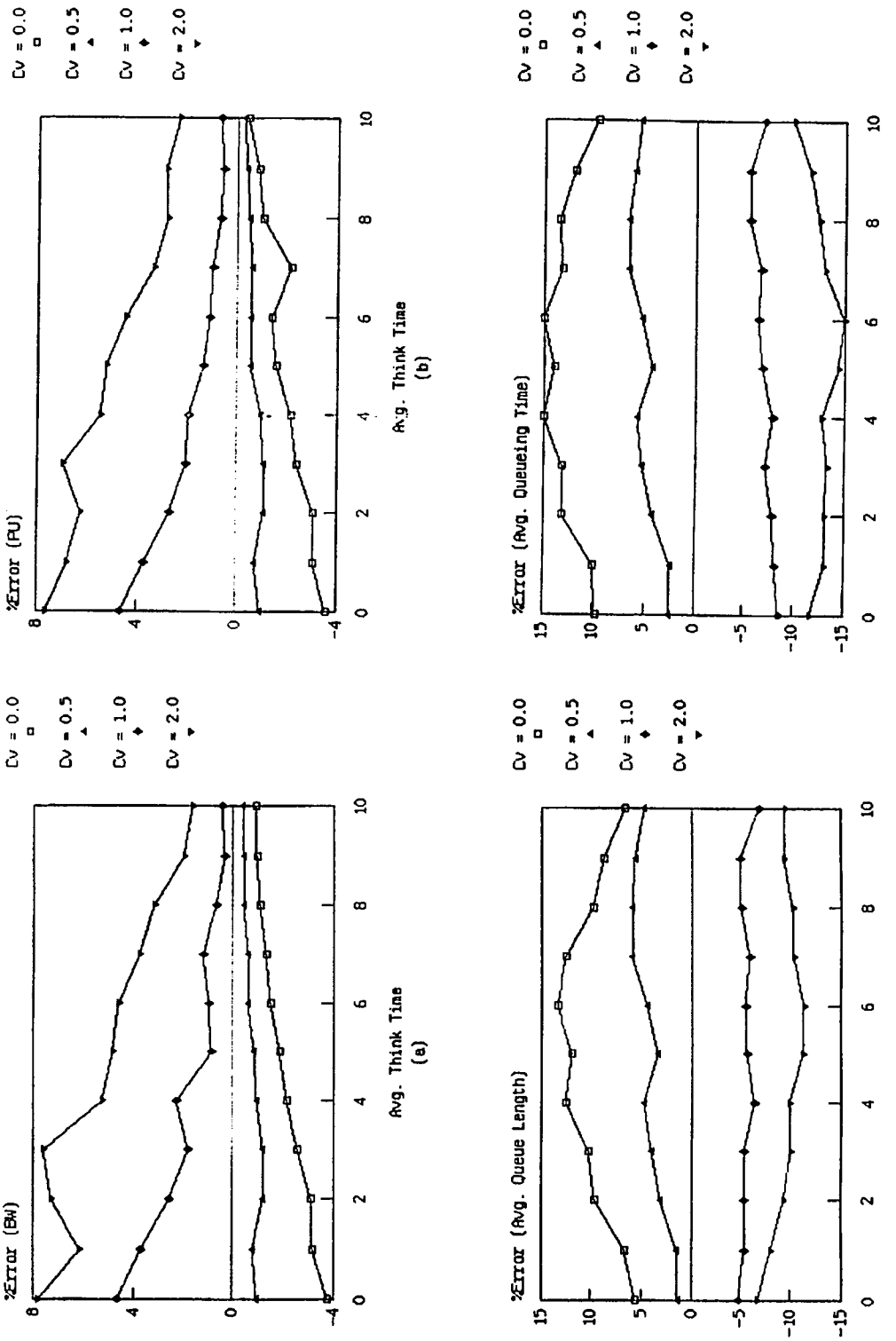



**Figure 5.8 The relative percentage error of Example 1.3 of the SMI model.**

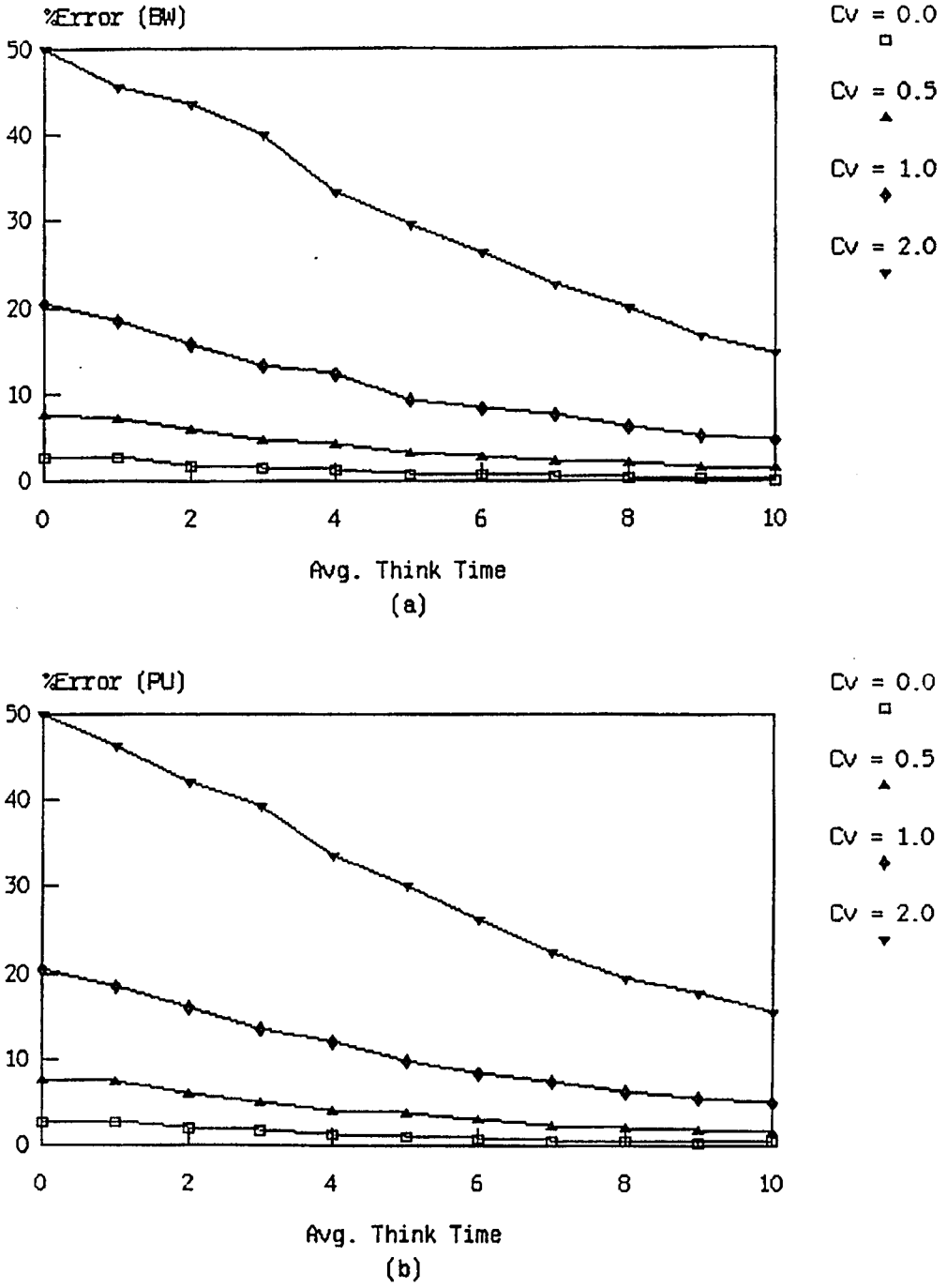



**Figure 5.9 The relative percentage error of Example 1.3 of the ER model.**

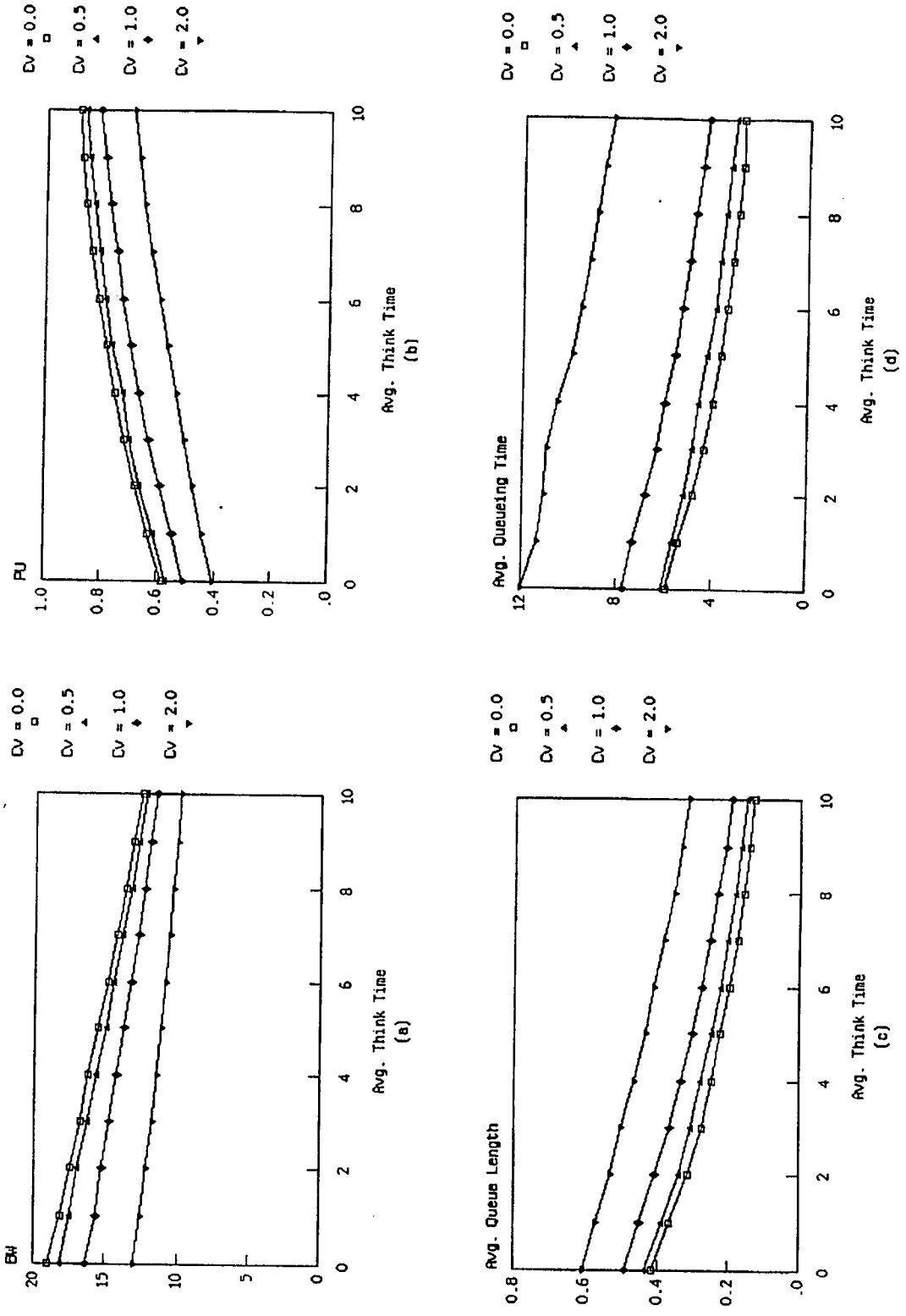



**Figure 5.10 The simulation results of Example 1.4.**

cessor system. Comparing the results of Figure 5.10 with those of Figure 5.7 indicates that the performance measures were changed as a result of increasing the average connection time, particularly when the average think time is more than zero.

Figure 5.11 shows that the accuracy of the SMI model in this example is similar to that in the previous examples. Figure 5.12 shows that the accuracy of the ER model in this example is similar to the accuracy obtained in the previous examples. ∎

## 5.3. The Mailbox Case

In this section, three examples of a multiprocessor system are studied whose behavior is characterized by the mailbox case assumptions, i.e., assumptions Im through VIm outlined in Section 4.3.2. The three examples assume that the system is a $32 \times 32$ multiprocessor system. In the first example, the effect of varying the average think time and the constant of variation on the different performance measures of the system are examined. In the second example, the average connection time between any $PE$ and the mailbox module will be changed to examine the effect of this change on the performance of the system. In the third example, the effect of varying the probability of requesting the mailbox module, $x$, on the performance of the system is examined.

The mailbox case assumptions will force the square multiprocessor system to behave like a concentrator rather than a square system. In other words, the mailbox case assumptions will yield a large coupling between the processing elements because they tend to go through the same memory module. Considering such assumptions will provide good test examples for the SMI model. It is noted that the SMI model decouples the processing elements then tries to compensate for that in the transition probabilities between the states.

**Example 2.1:**

In this example a multiprocessor system that has 32 $PE$s and 32 $MM$s is considered. The multiprocessor system operations can be characterized by the mailbox case assumptions. In this

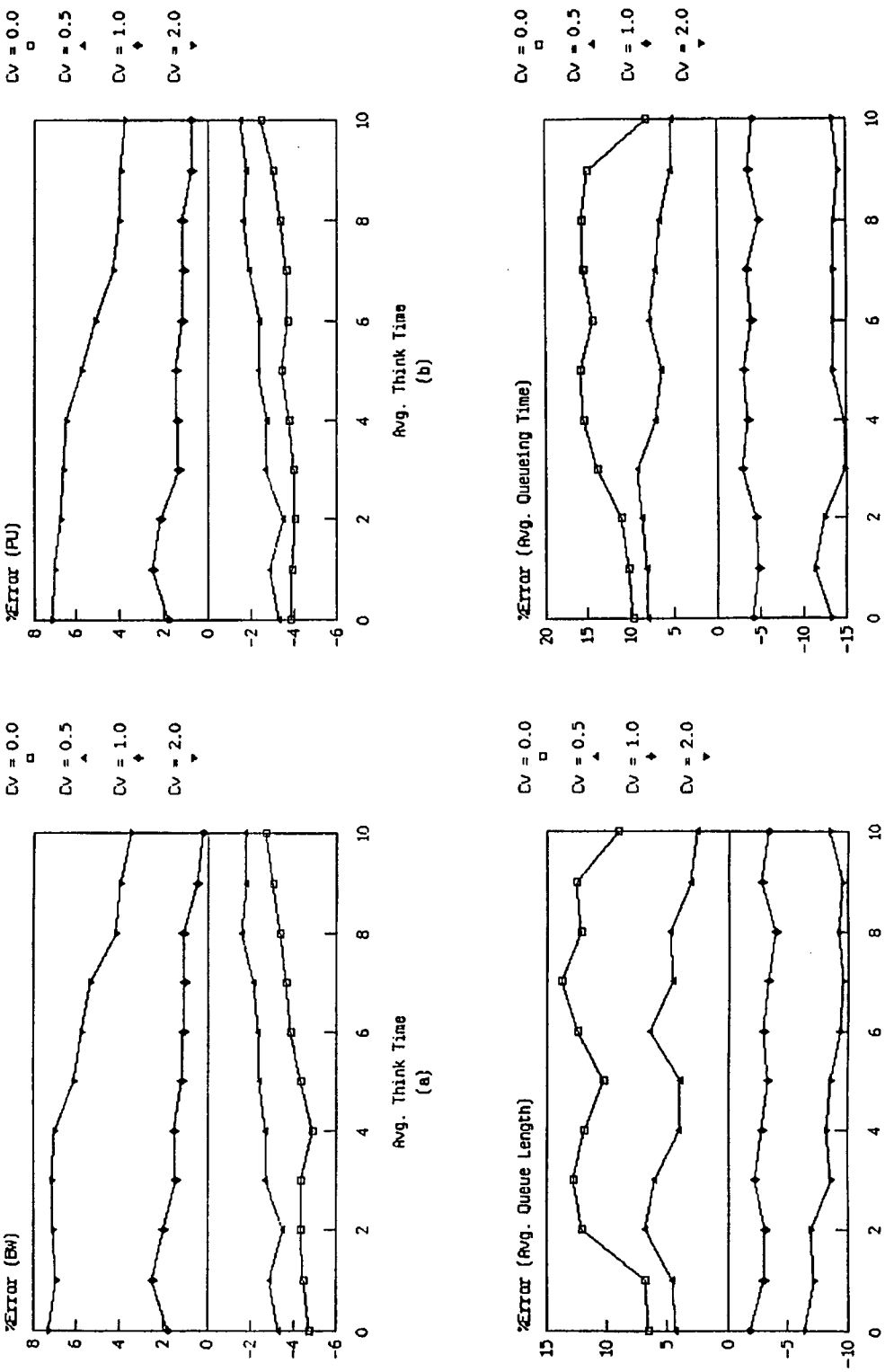



**Figure 5.11 The relative percentage error of Example 1.4 of the SMI model.**

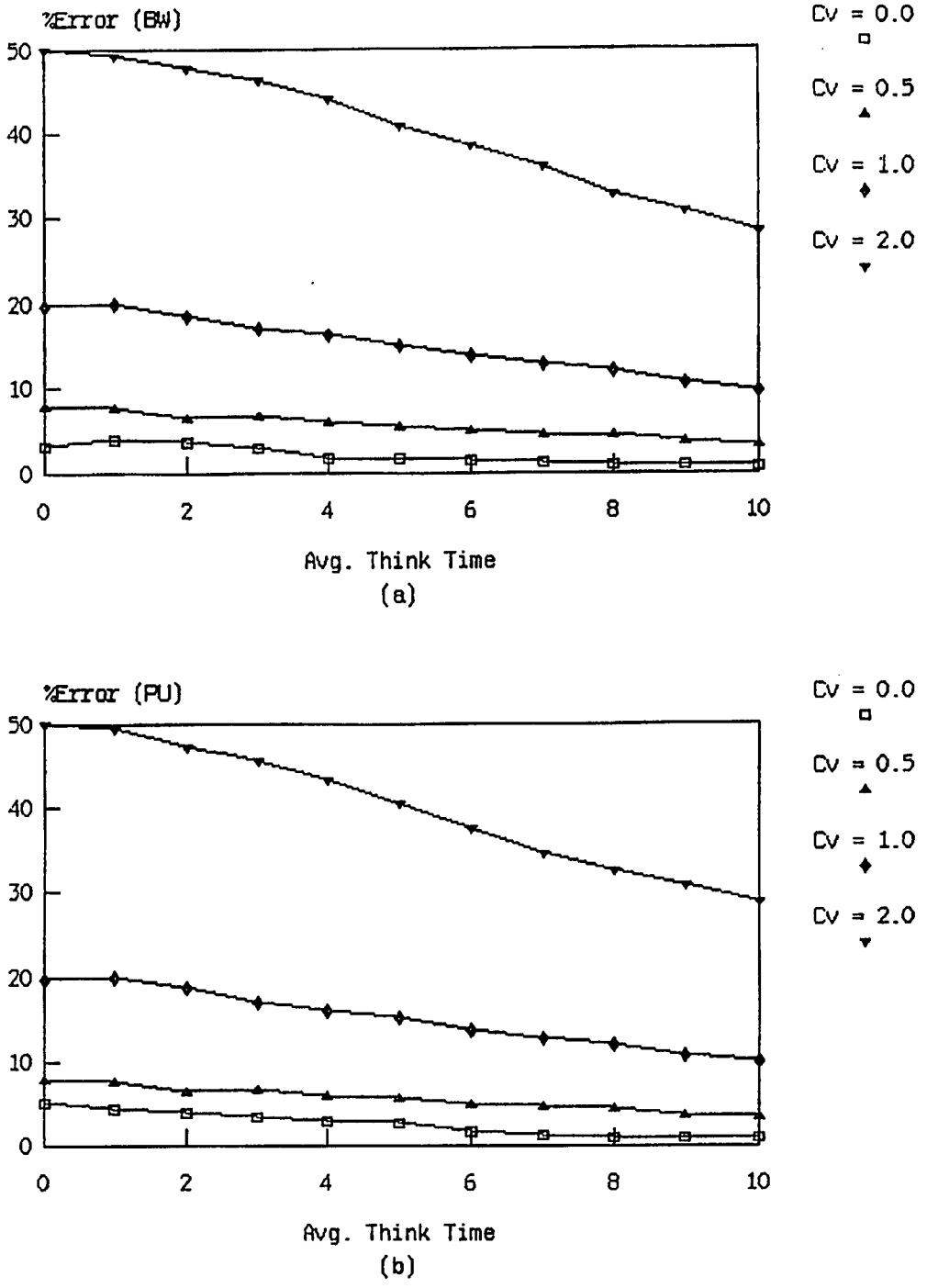



**Figure 5.12 The relative percentage error of Example 1.4 of the ER model.**

example, it is assumed that the probability that any $PE$ requests the mailbox module, $x$, is 0.07. The average connection time between any $PE$ and the mailbox module, $\overline{C}_1$, is four cycles. The average connection time between any $PE$ and a non-mailbox module ($MM_2$ through $MM_{32}$), $\overline{C}_2$, is four cycles. Similar to the examples in the previous section, the average think time, $\overline{T}$, is varied from zero to ten cycles. Moreover, the constant of variation of the connection time, $C_v$, is varied from zero to two. It is noted that the variation in the connection time is assumed to be the same whether the module is a mailbox or not.

Figure 5.13 shows the simulation results of the utilization measures: $BW$, $PU_1$, $MU_1$, and $MU_j$ (where $2 \leq j \leq 32$) as functions of $\overline{T}$ and $C_v$. The memory bandwidth and the processor utilization show similar behavior as in the uniform case. However, comparing Figures 5.13(a,b) and 5.7(a,b) indicates that when the average think time is small ( $\leq 4$ ), the bandwidth and the processor utilization in the mailbox case are less than their values in the uniform case because at these operating points the mailbox module behaves as a bottleneck to the system which, in turn, degrades the system performance. Nevertheless, when the average think time is large ( $> 4$ ), the bandwidth and the processor utilization in the mailbox case are similar to their values in the uniform case because at these operating points the long think time causes the bottleneck effect of the mailbox to be small. Figures 5.13(c,d) show the mailbox module utilization and the non-mailbox module utilization, respectively. The mailbox module is fully utilized particularly when $\overline{T}$ is small. It can be noticed that in all of the tested operating points the mailbox module was utilized at least 60% of the time and the non-mailbox module was utilized, at most, 44% of the time. Figure 5.14 shows the average queue length and the average queueing time in this example. In the case of the mailbox module, see Figure 5.14(a,c), the average queue length and the average queueing time of the mailbox module will decrease as the $C_v$ of the connection time increases and the average think time is small ( $\leq 4$ ). However, for large $\overline{T}$ ( $> 4$ ), both the average queue length and the average queueing time will increase slightly as $C_v$ increases. Therefore, it can be concluded that when $\overline{T}$ is small the mailbox module acts as a bottleneck to the multiprocessor system. This will result in a degradation in the system performance, particularly if compared with

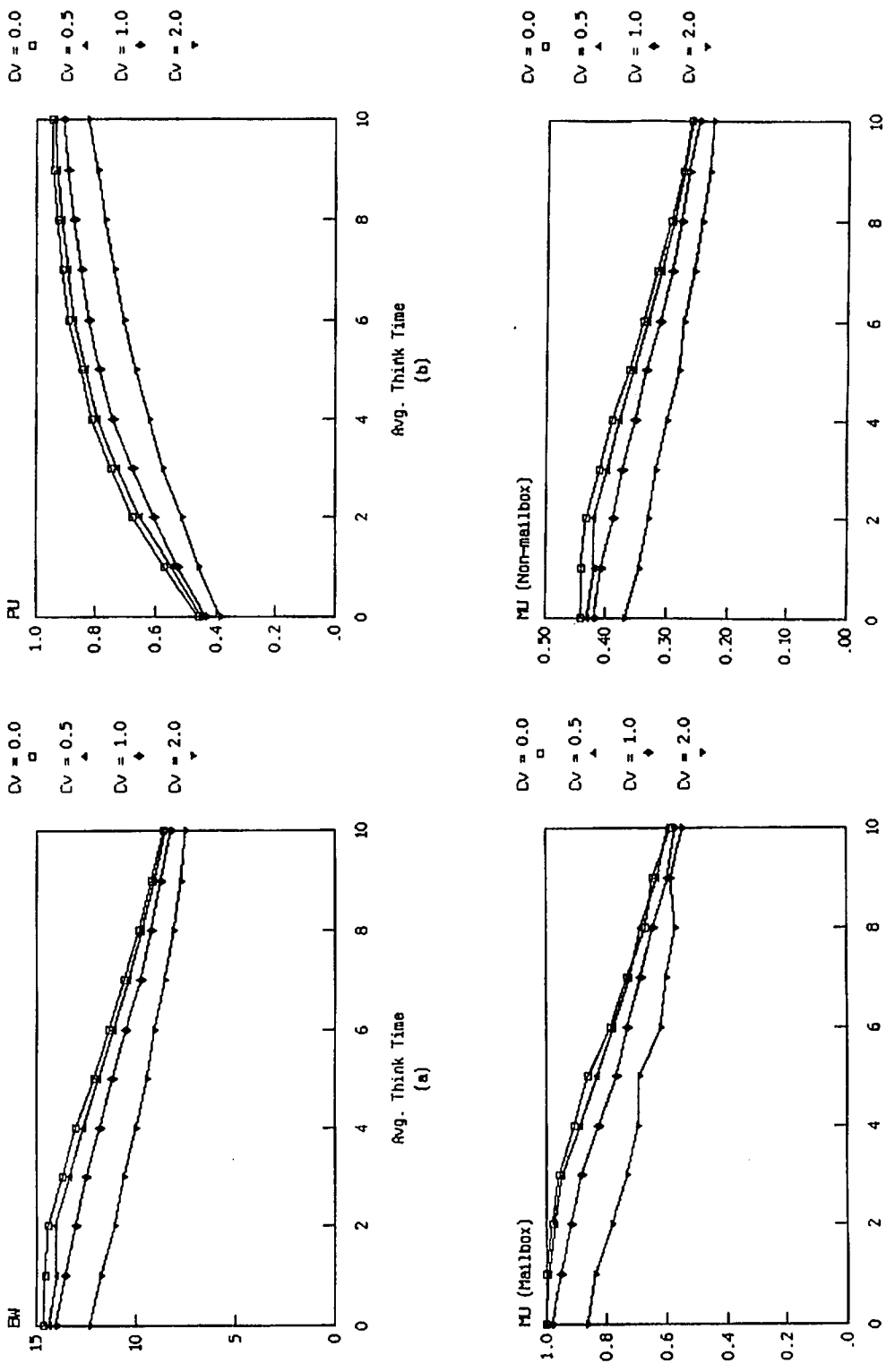



**Figure 5.13 The simulation results (utilisation measures) of Example 2.1.**

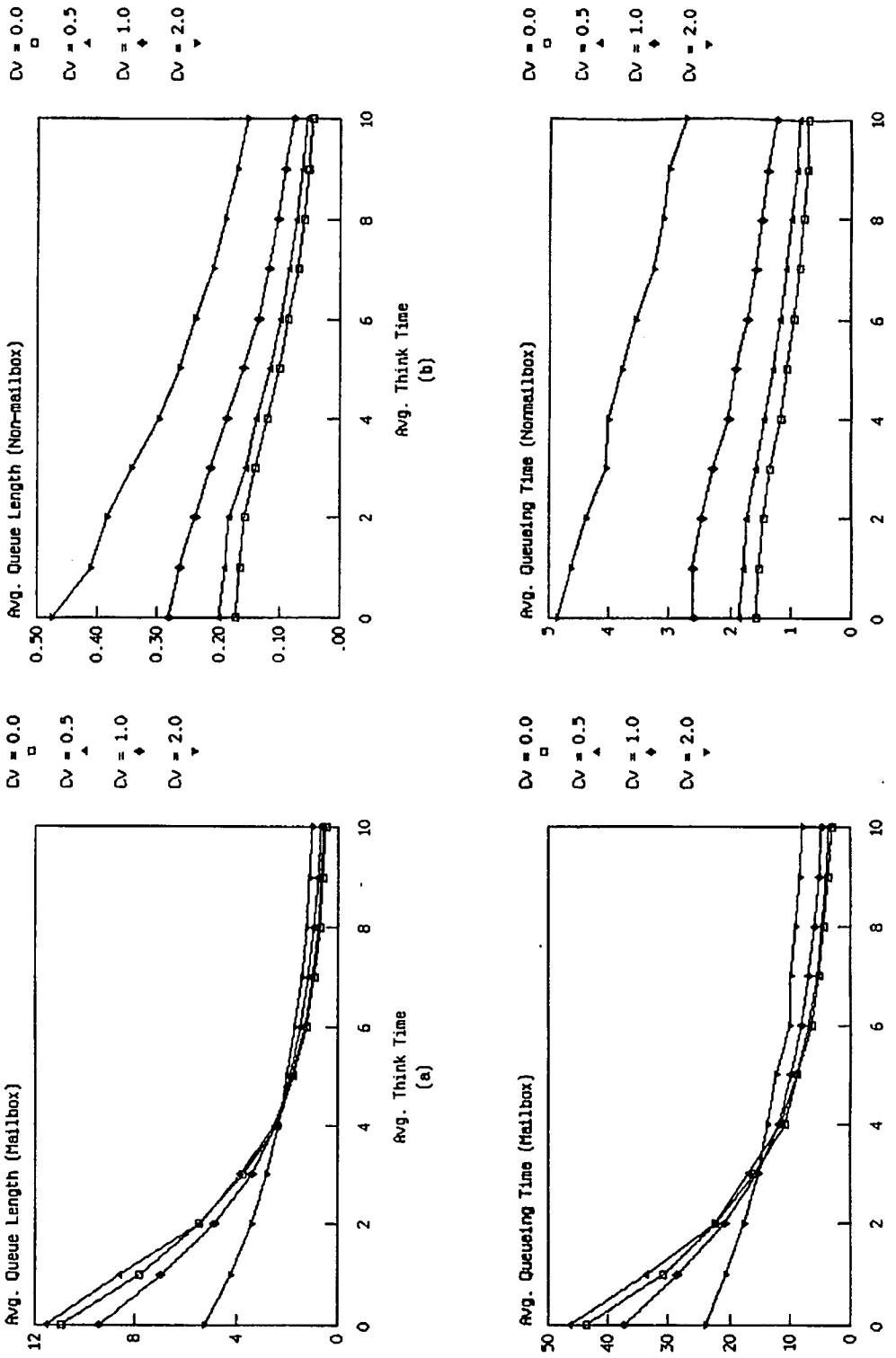



**Figure 5.14 The simulation results (queue measures) of Example 2.1.**

the system performance under the uniform case assumptions.

Figures 5.15 and 5.16 show the relative percentage error produced by the SMI model for the performance measures shown in Figures 5.13 and 5.14. It can be seen that the accuracy of the SMI model is similar to its accuracy under the uniform case assumptions in spite of the large coupling between the processing elements in this case, particularly when the average think time is small. ∎

**Example 2.2:**

This example reconsiders the multiprocessor system of Example 2.1. However, in this example it is assumed that the average connection time between any *PE* and the mailbox module, $\overline{C}_1$, is equal to eight cycles. The other system parameters have the same values as in Example 2.1. Comparing the results of this example with those of Example 2.1 enables us to study the effect of $\overline{C}_1$ on the performance of the system. Similar to the previous examples, the performance measures of the system are studied as functions of $\overline{T}$ and $C_v$.

Figure 5.17 shows the utilization measures of the system in this example. Varying the $C_v$ of the connection time from zero to two yields approximately about 1 to 2% changes in the utilization measures of the system. Hence, the only result shown is when $C_v = 0$. Furthermore, it can be seen that the memory bandwidth is almost constant around eight. The utilization of the mailbox module stayed at 1.0 regardless of the values of $\overline{T}$ or $C_v$. For this reason the memory utilization measure was not shown in Figure 5.17. Comparing the results of Figures 5.17 and 5.13, indicates that the effect of changing $\overline{C}_1$ on the utilization measures are as follows. First, the memory bandwidth decreases with the increase of $\overline{C}_1$. Second, the processor utilization also decreased with the increase of $\overline{C}_1$. The reason for this is that increasing $\overline{C}_1$ will cause the processing elements to stay longer in the mailbox module which is the bottleneck of the system. Therefore, increasing $\overline{C}_1$ is one way to intensify the effect of the mailbox as a bottleneck.

Figure 5.18 shows the queue measures of the multiprocessor system in this example. Changing the constant of variation of the connection time, $C_v$, from zero to two yields a small change,

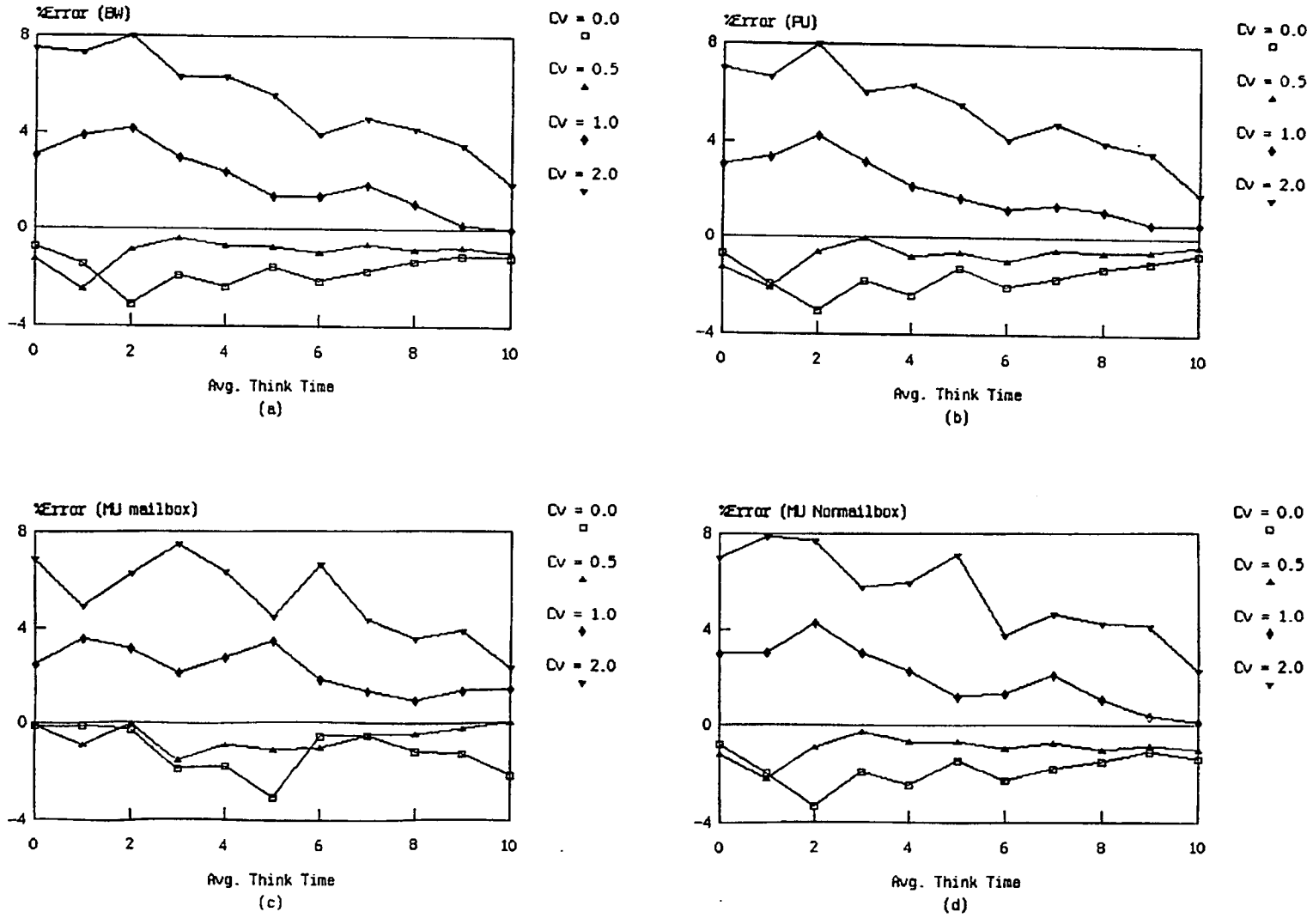



**Figure 5.15 The relative percentage error (utilization measures) of Example 2.1.**

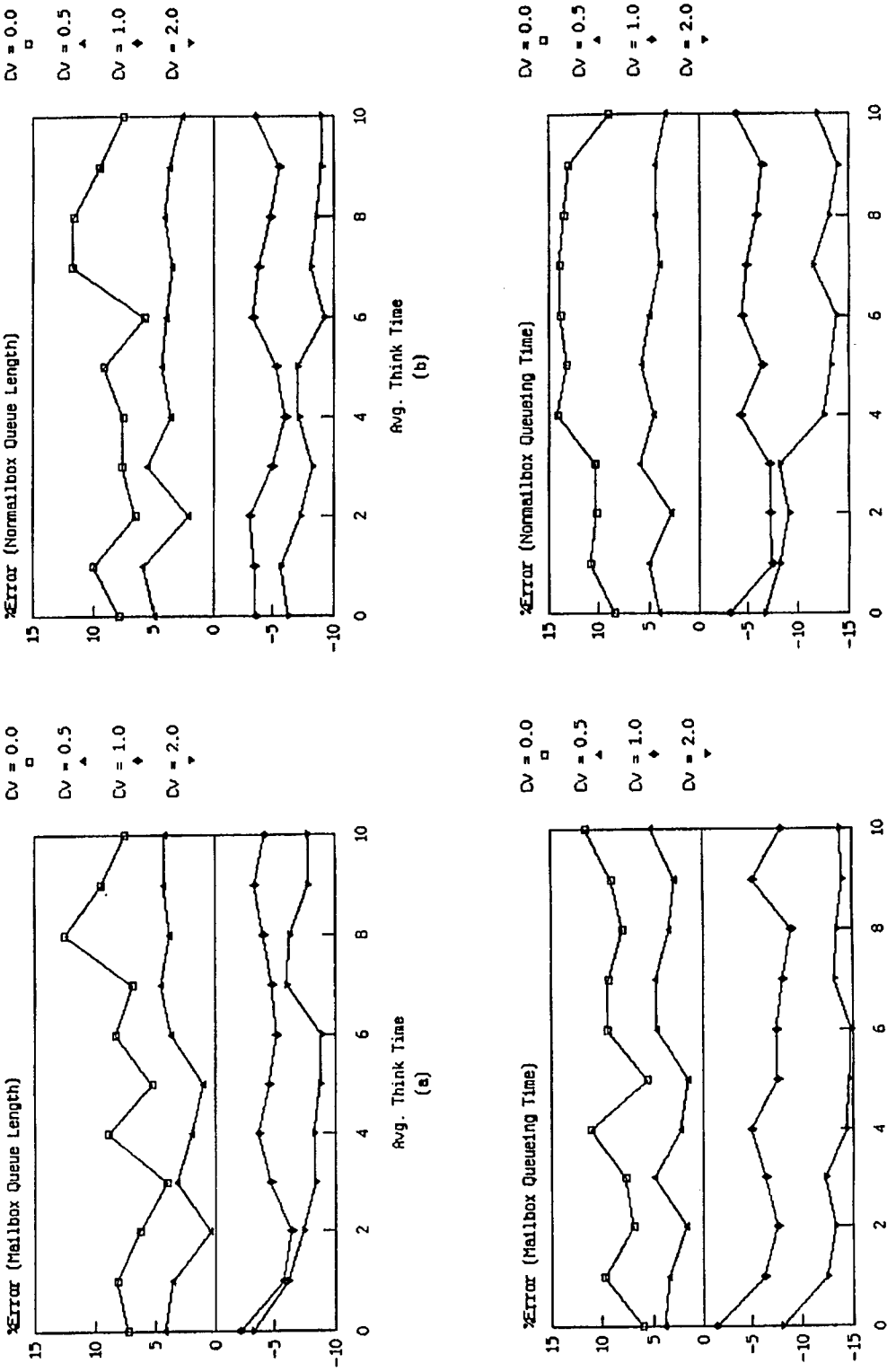



**Figure 5.16 The relative percentage error (queue measures) of Example 2.1.**

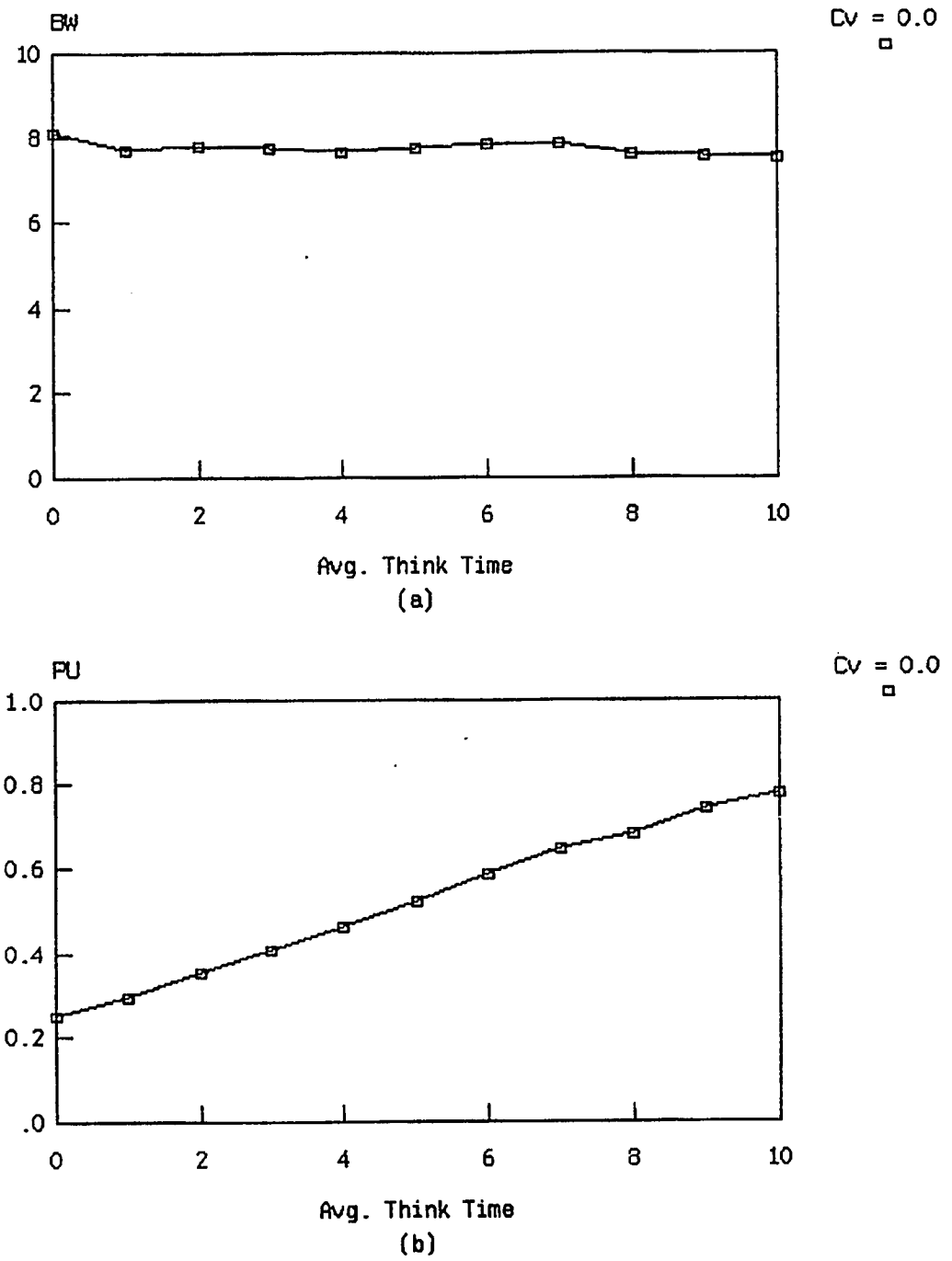



**Figure 5.17 The simulation results (utilization measures) of Example 2.2.**

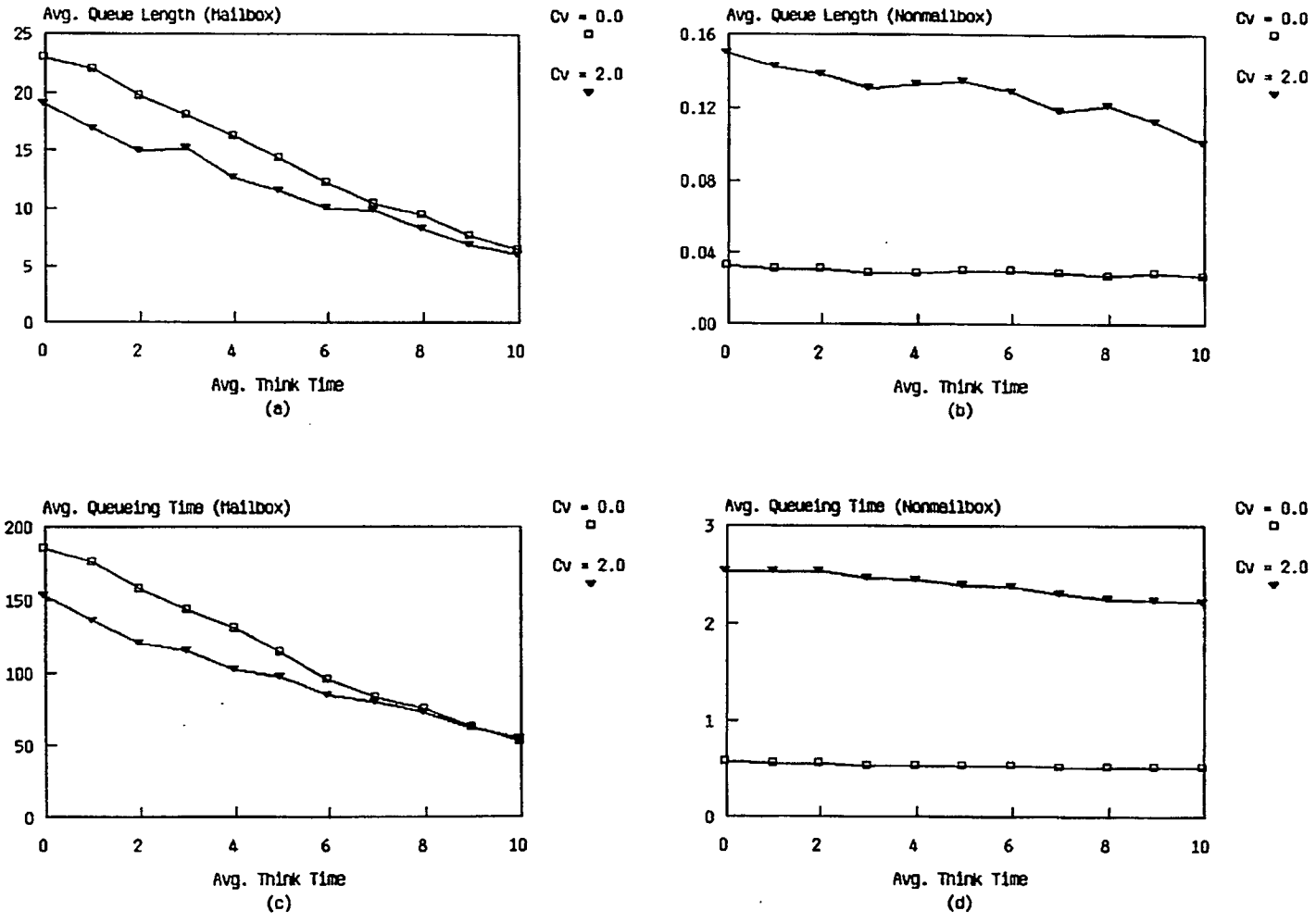



**Figure 5.18 The simulation results (queue measures) of Example 2.2.**

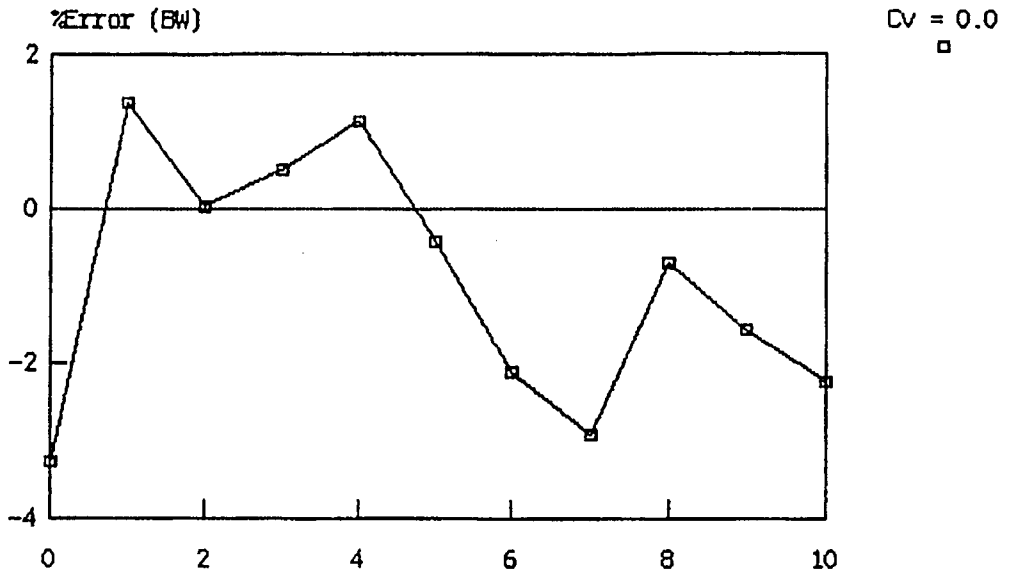
particularly in the average queue length of the mailbox module and the average queueing time in the mailbox module. Hence, only the results when $C_v = 0$ and $C_v = 2$ are displayed. Examining the results of Figure 5.18 indicates that the mailbox queue length and queueing time are much greater than the non-mailbox measures. Furthermore, the mailbox measures are greater than when $\overline{C}_1$ was four cycles, see Figure 5.14. This confirms the previous conclusion that the effect of the mailbox as a bottleneck is intensified after increasing $\overline{C}_1$. The average queue length and the average queueing time of the mailbox module decreases as $C_v$ increases. Since there are 31 non-mailbox modules in the system, it is correct to assert that the average queue length and the average queueing time of the multiprocessor system in this example will increase as $C_v$ increases.

Figures 5.19 and 5.20 illustrate the relative percentage error of the performance measures of the SMI model shown in Figures 5.17 and 5.18. The accuracy of the model is similar to the accuracy obtained in the previous examples, in spite of a great deal of coupling between the processing elements through the mailbox module. ∎

**Example 2.3:**

This example reconsiders the multiprocessor system of Example 2.1. However, in this example it is assumed that the probability that a $PE$ requests the mailbox module, $x$, is equal to 0.225. The other system parameters have the same values as in Example 2.1. Comparing the results of this example and those of Example 2.1 will enable us to study the effect of varying the parameter, $x$. It is clear that by increasing $x$ the bottleneck effect of the mailbox will be intensified. For example, if $x$ approaches one, then the multiprocessor system is actually a $32 \times 1$ system. Therefore, increasing $x$ will increase the coupling between the processing elements of the system and that can be used as a good example to test the accuracy of the SMI model.

When the results of the simulations of this example were examined, it was noticed that most of the utilization measures were not affected by $\overline{T}$ or $C_v$. They are summarized as follows: The memory bandwidth was 4.5 ($\pm$ 0.1) regardless of the value of $\overline{T}$ or $C_v$. The memory utilization of the mailbox module was 1.0 regardless of the value of $\overline{T}$ or $C_v$. The memory utilization of the



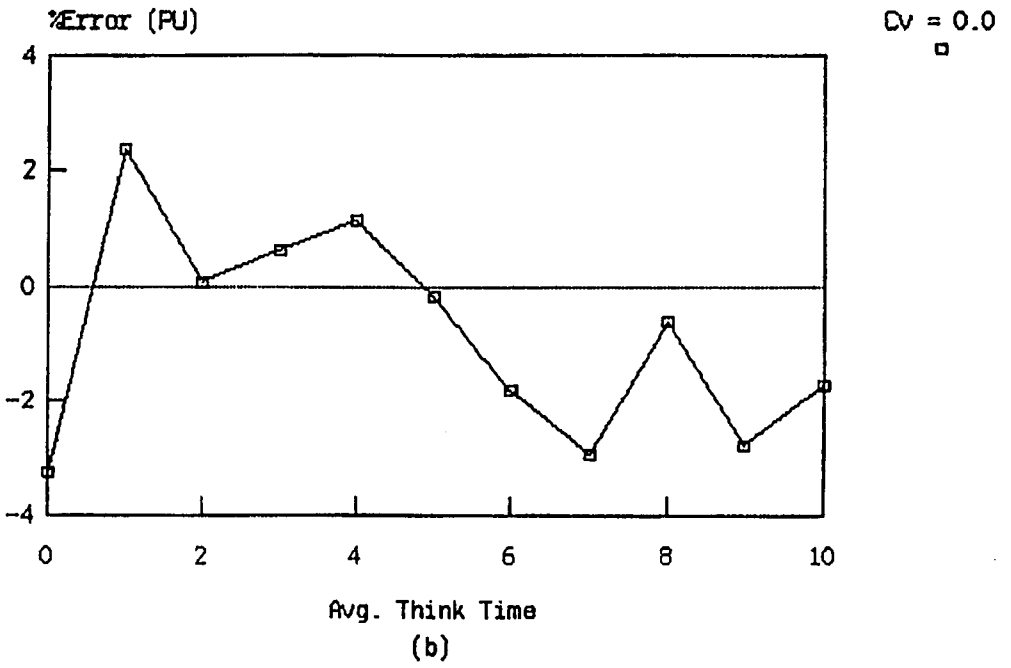




**Figure 5.10 The relative percentage error (utilisation measures) of Example 2.2.**

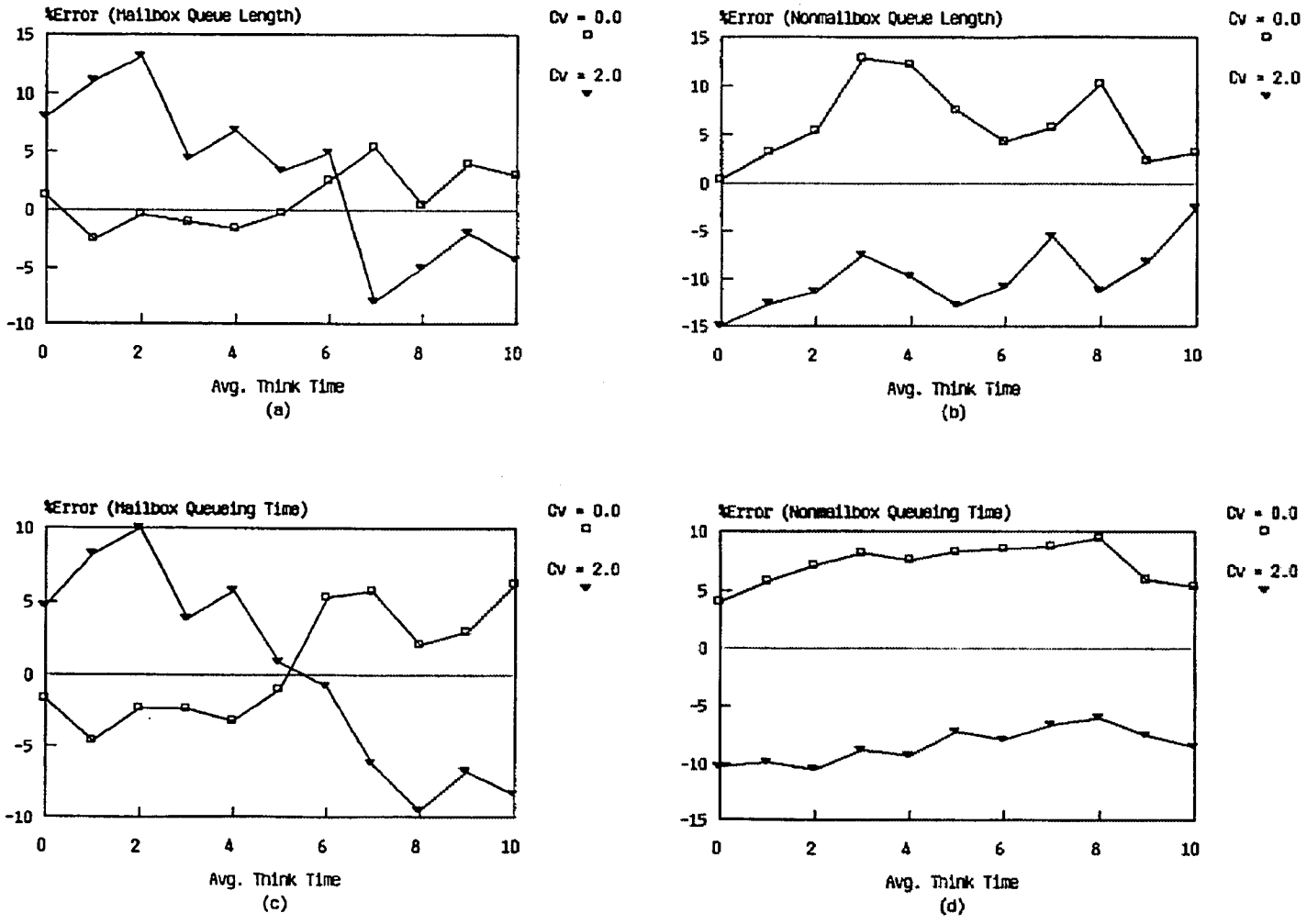



**Figure 5.20 The relative percentage error (queue measures) of Example 2.2.**

nonmailbox module was 0.112 (± 0.002) regardless of the value of $\overline{T}$ or $C_v$. However, the processor utilization, $PU_i$, is dependent on the value of $\overline{T}$ by definition. For example, at $\overline{T} = 0$ $PU_i$ has the value 0.142 and at $\overline{T} = 10$ $PU_i$ has the value 0.486 regardless of the value of $C_v$. From these results it can be seen that increasing the parameter $x$ intensified the effect of the mailbox module as a bottleneck to the system. Comparing these results with the utilization results of Example 3.1 yields the following observations. Increasing the parameter $x$ from 0.07 to 0.225 results in a sharp decrease in the memory bandwidth, the processor utilization, and the memory utilization of the nonmailbox module. Moreover, the increase in the parameter $x$ yields an increase in the mailbox module utilization. The last observation shows the intensifying effect of the mailbox module as a bottleneck to the multiprocessor system.

Figure 5.21 shows the queue measures of this example. Changing the constant of variation of the connection time, $C_v$, from zero to two yields a small change in these measures, particularly in the mailbox average queue length and the mailbox average queueing time. Hence, only the results where $C_v = 0$ and $C_v = 2$ are displayed. The average queue length and the average queueing time of the mailbox module decrease as the average think time increases. When Figures 5.21(a,c) and 5.14(a,c) are compared, it can be observed that the increase of $x$ from 0.07 to 0.225 more than doubles the average queue length and the average queueing time of the mailbox module. The average queue length and the average queueing time of the nonmailbox module remains almost constant regardless of the value of $\overline{T}$. However, both of these values increase as a result of increasing $C_v$. When Figures 5.21(b,d) and 5.14(b,d) are compared, it can be observed that the increase of $x$ from 0.07 to 0.225 yields a drastic decrease in the average queue length and the average queueing time of the nonmailbox module. This observation is a result of intensifying the effect of the mailbox as a bottleneck to the multiprocessor system.

The relative percentage error of the utilization measures of the SMI model were within ±4%. The relative percentage error of the queue measures are shown in Figure 5.22. In spite of the great degree of coupling between the processing elements, the SMI model produced some what accurate results for this example. ∎

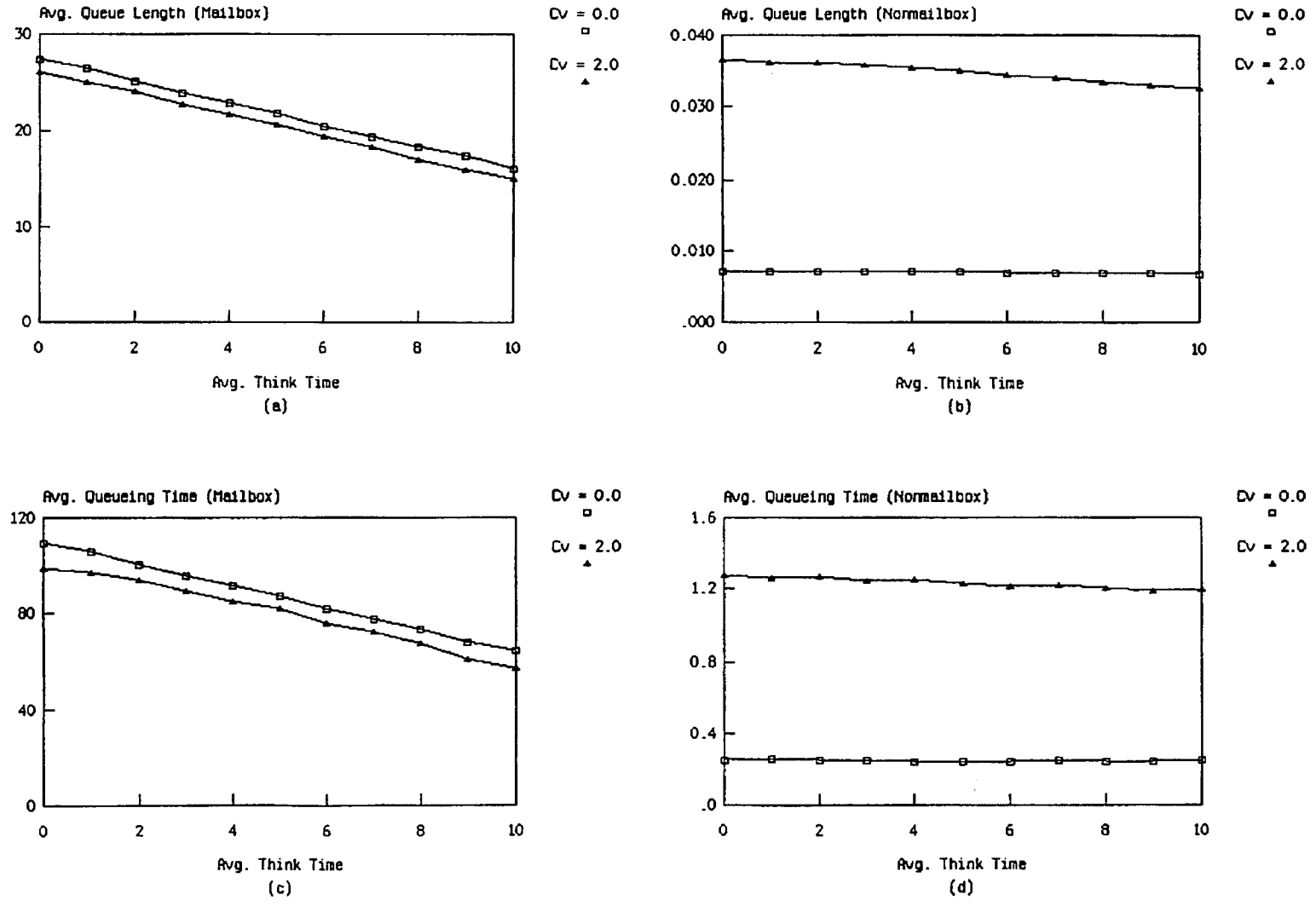



**Figure 5.21 The simulation results (queue measures) of Example 2.3.**

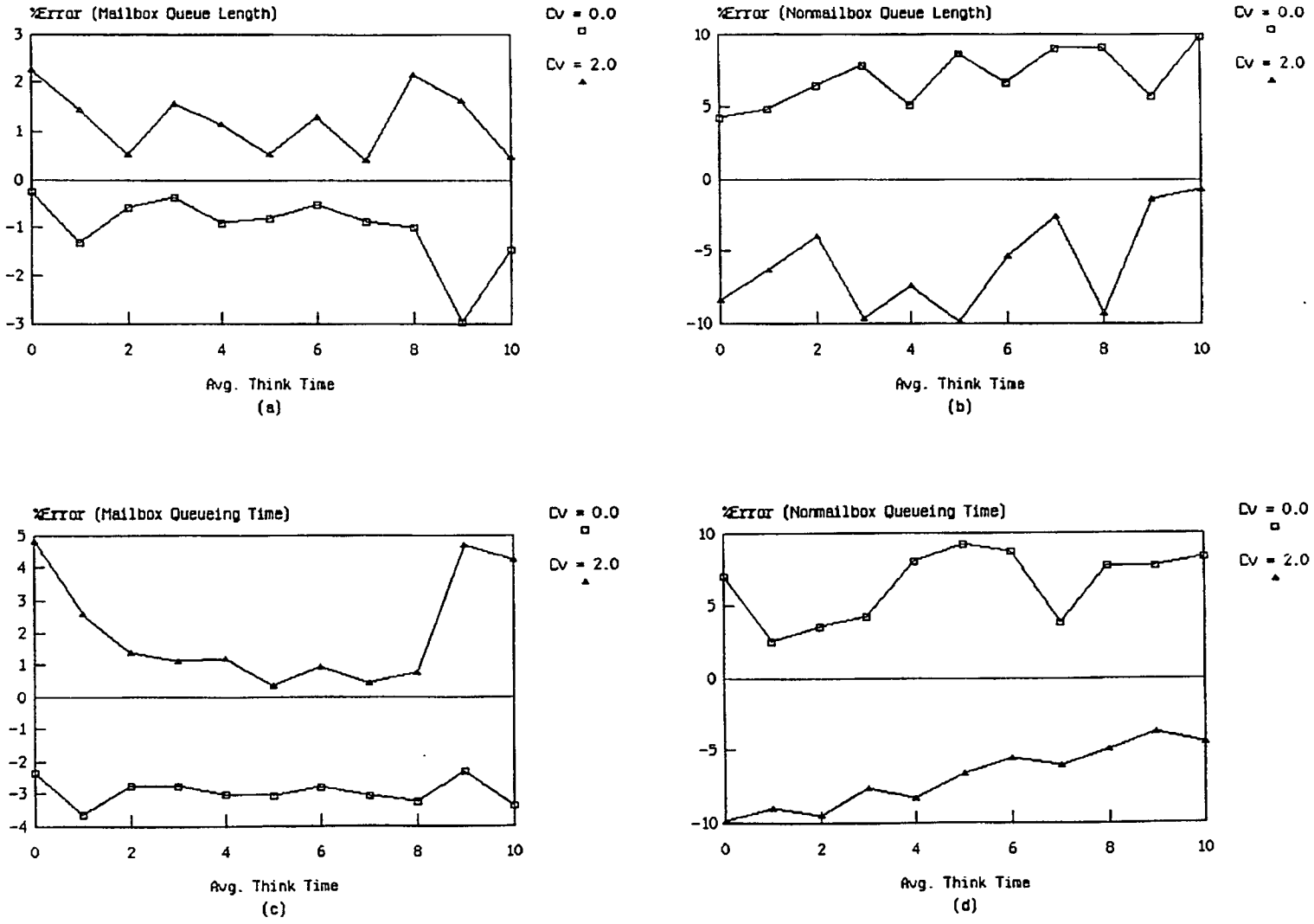



**Figure 5.22 The relative percentage error (queue measures) of Example 2.3.**

## 5.4. The Favorite Module Case

In this section, three examples of a multiprocessor system whose behavior is characterized by the favorite module case assumptions, i.e., assumptions If through VIf outlined in Section 4.3.3 are studied. The three examples assume that the system is a 32 × 32 multiprocessor system. In the first example, the effect of varying the average think time, $\overline{T}$, and the constant of variation of the connection time, $C_v$, on the different performance measures of the system are examined. In the second example the average connection time between any processing element and its favorite module, $\overline{C}_1$, will be changed to examine the effect of this parameter on the performance of the system. In the third example, the effect of varying the probability of requesting the favorite module, $x$, on the performance of the system will be examined.

The favorite module case assumptions will force some reduction in the coupling between the processing elements and hence the performance of the system will be upgraded. Under the favorite module assumptions the memory interference will decrease as the probability of requesting the favorite module increases. For this reason, the multiprocessor system will be preferred under these assumptions in order to exploit the system resources in an optimum way.

**Example 3.1:**

In this example a multiprocessor system will be considered that has 32 *PE*s and 32 *MM*s. The multiprocessor system operation can be characterized by the favorite module case assumptions. In this example it is assumed that the probability that any *PE* requests its own favorite module, $x$, is 0.07. The average connection time between any *PE* and its favorite module, $\overline{C}_1$, is four cycles. The average connection time between any *PE* and a nonfavorite module, $\overline{C}_2$, is four cycles. Similar to the previous examples, the average think time, $\overline{T}$, is varied from zero to ten cycles. Moreover, the constant of variation of the connection time, $C_v$, is varied from zero to two. The variation in the connection time is assumed to be the same whether the module is favorite or not.

Figure 5.23 shows the simulation results of the memory bandwidth, the processor utilization, and the average queue length as functions of $\overline{T}$ and $C_v$. The memory utilization is not included because it can be calculated easily from the memory bandwidth, i.e., $MU_j = BW/M$. The three measures shown in Figure 5.23 are similar to the measures of the uniform case shown in Figure 5.7. The reason for this is that the value of $x$ is not large enough in this example to produce a large difference from the uniform case, hence the multiprocessor system behavior is similar to the system behavior under the uniform case assumptions. Figure 5.24 shows the average queueing time experienced by a $PE$ in its favorite module queue and its nonfavorite module queue. It is clear that the average queue time in the favorite module is smaller than the average queueing time in the nonfavorite module. The reason for this is that the rate of arrival of a $PE$ to its favorite module is higher than the rate of arrival to a nonfavorite module. Nevertheless, the difference between the two queueing times is small when the variation in the connection time is small, and the difference is moderate when the variation in the connection time is large. Figures 5.23 and 5.24 show that when $C_v$ is increased $BW$ and $PU_i$ will decrease, but $L_j$ and $W_{i,j}$ will increase.

Figures 5.25 and 5.26 show the relative percentage error of the measures of Figures 5.23 and 5.24. It can be seen that the accuracy of the SMI model is similar to its accuracy in the previous examples. This confirms that the SMI model accuracy has not been affected by the operation assumptions of the system. ∎

**Example 3.2:**

This example reconsiders the multiprocessor system of example 3.1. However, in this example it is assumed that the average connection time between any $PE$ and its favorite module, $\overline{C}_1$, is eight cycles. The other system parameters have the same values as in Example 3.1. Comparing the results of this example with those of Example 3.1 enables us to study the effect of $\overline{C}_1$ on the performance of the system. Similar to the previous examples, the performance measures of the system will be studied as functions of $\overline{T}$ and $C_v$.

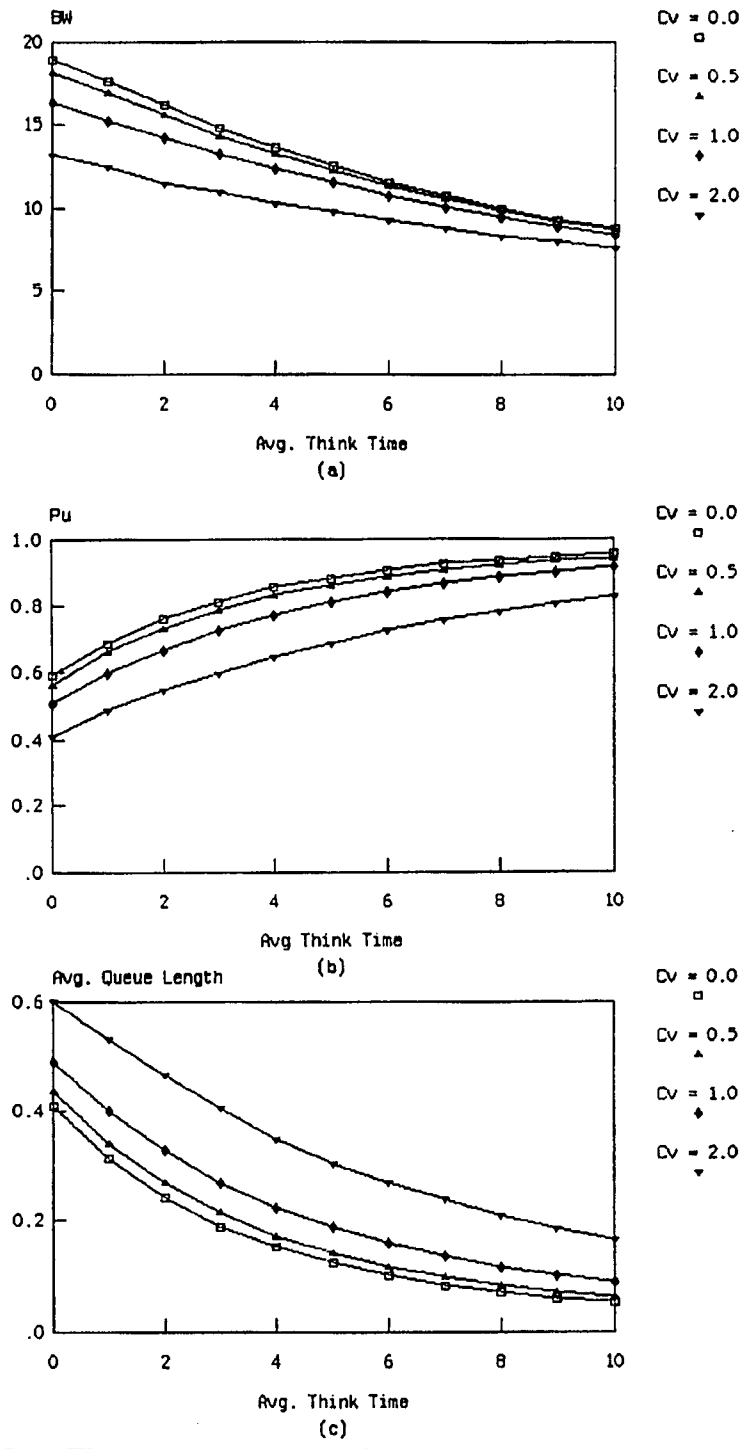



**Figure 5.23 The simulation results ($BW$, $PU_i$, $L_j$) of Example 3.1.**

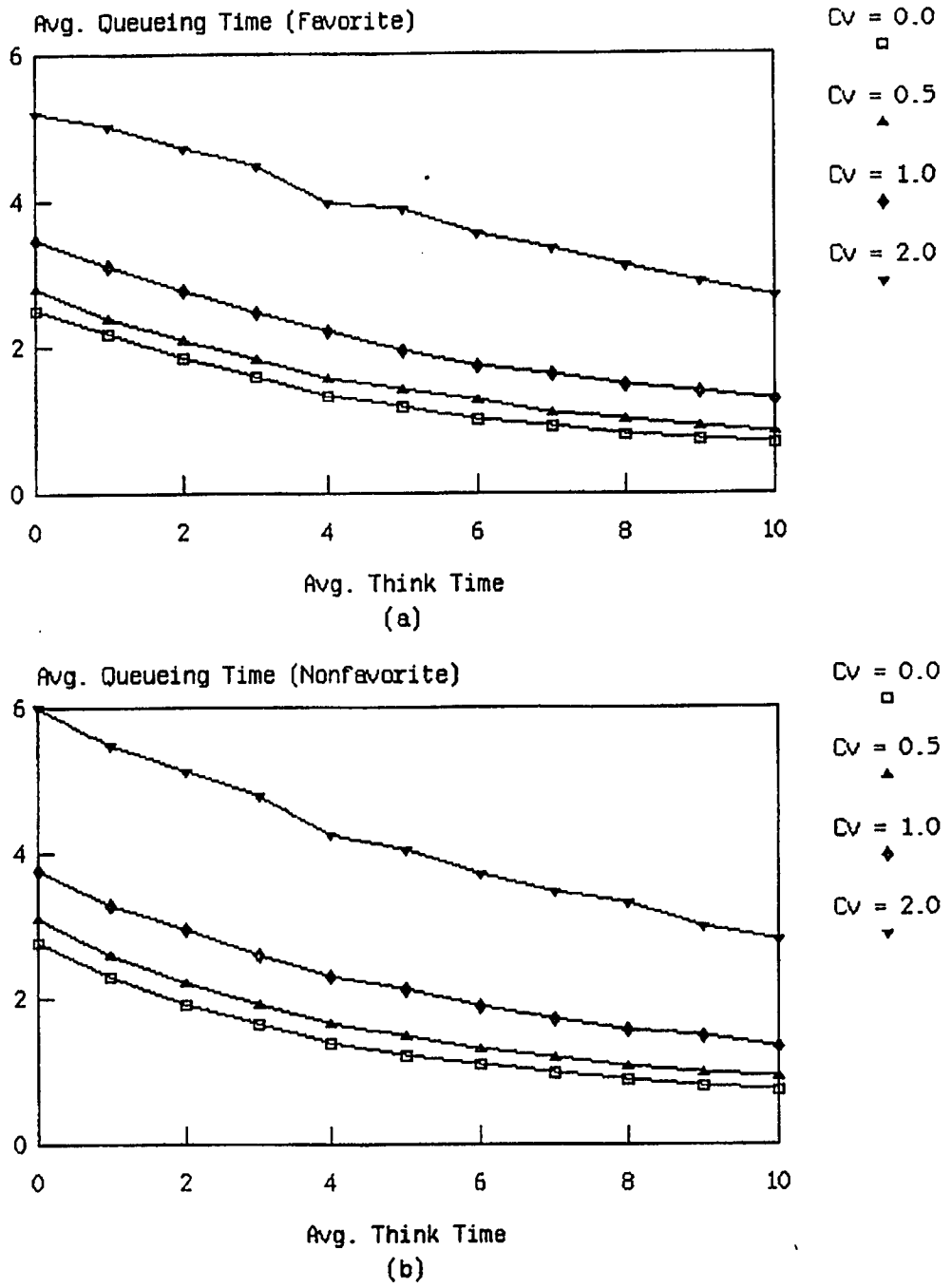



**Figure 5.24 The simulation results ($W_{i,j}$) of Example 3.1.**

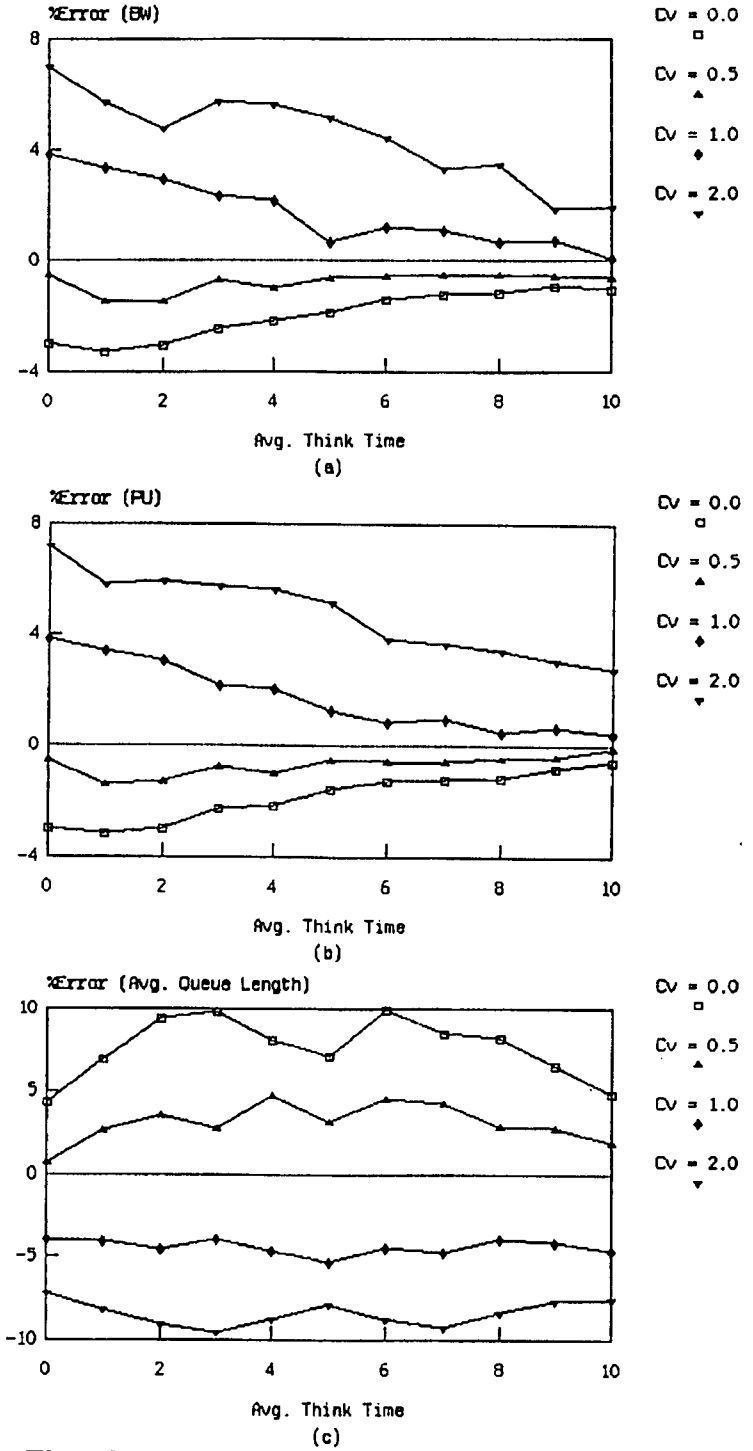



**Figure 5.25 The relative percentage error ($BW$, $PU_i$, $L_i$) of Example 3.1.**

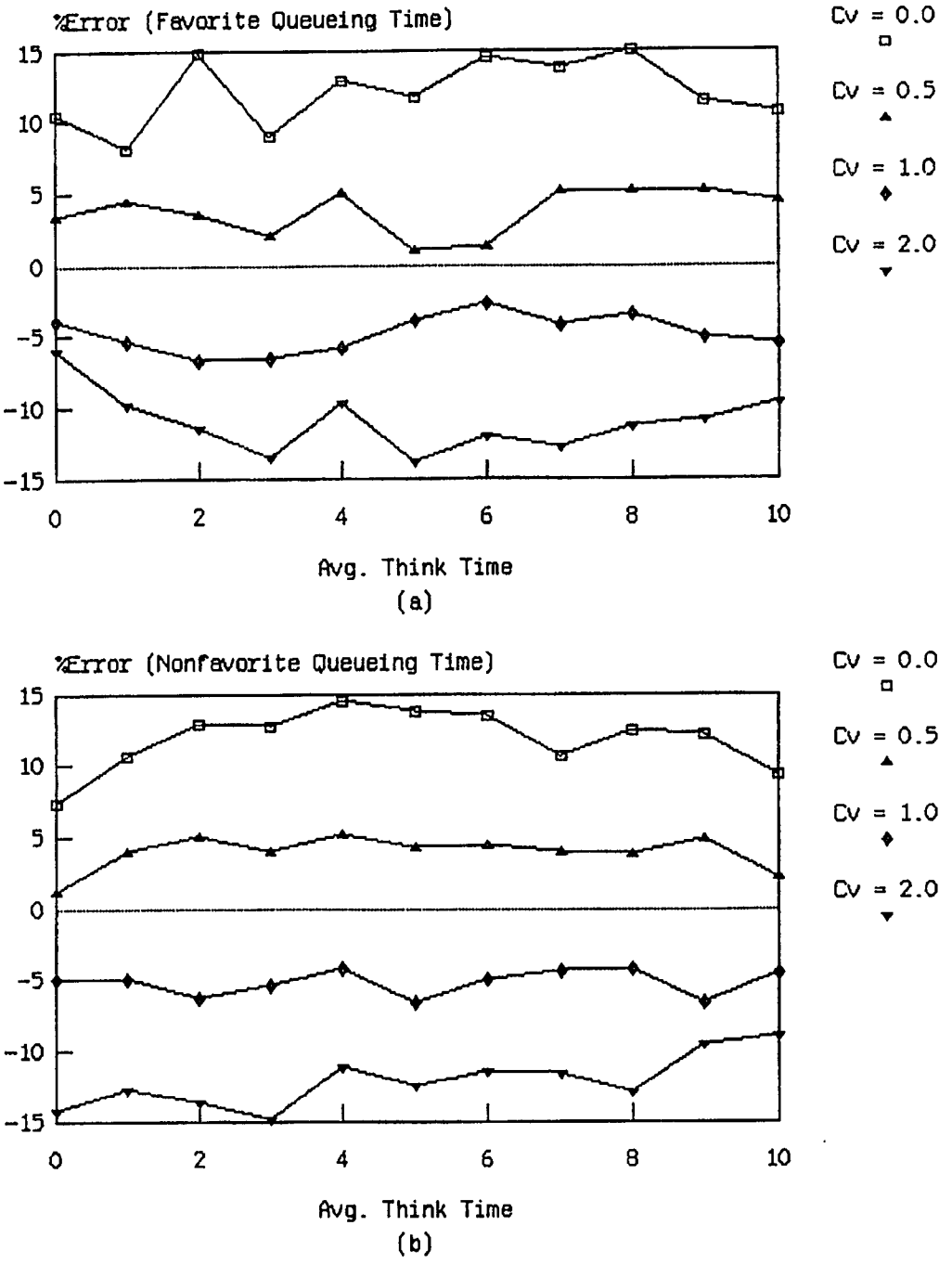



**Figure 5.26 The relative percentage error ($W_{1,j}$) of Example 3.1.**

The simulation results obtained in this example are similar to the results obtained in Example 3.1. Furthermore, the accuracy of the SMI model in this example is similar to its accuracy in Example 3.1. Therefore, the results of Example 3.2 are not presented. The main conclusion from this example is that increasing parameter $\overline{C}_1$ from four to eight cycles did not have a large impact on the system performance. Changing $\overline{C}_1$ from four to eight did not change the average connection time between any *PE* and an *MM* significantly. Hence, the multiprocessor system behavior did not experience any difference as a result of increasing $\overline{C}_1$. ■

**Example 3.3:**

This example reconsiders the multiprocessor system of Example 3.1. However, in this example it is assumed that the probability that a *PE* requests its favorite module, $x$, is equal to 0.225. The other system parameters have the same values as in example 3.1. Comparing the results of this example and those of Example 3.1 enables us to study the effect of varying the parameter, $x$. It is clear that by increasing $x$ the multiprocessor system will experience less memory interference and a better overall performance. For instance if $x$ approaches one, then the multiprocessor system is actually an interference-free system. The bandwidth will be equal to the maximum value, processor utilization will be one, the queue length will be zero, and the queueing time will be zero. Furthermore, there will be no coupling between the *PE*s.

The simulation results obtained in this example are similar to the results obtained in Example 3.1. However, the average queueing time in the favorite module is less than 20 to 25% of its value in Example 3.1. It is clear that increasing $x$ from 0.07 to 0.225 did not have a major effect on the behavior of the multiprocessor system. The only impact it had was to decrease the average queueing time in the favorite module. However, increasing the parameter, $x$, to other large values definitely will affect the other measures of the performance. The accuracy of the SMI model is similar to the accuracy in Example 3.1. Therefore, the results of Example 3.3 are not included here. ■

## 5.5. The General Case

In this section, two examples of a multiprocessor system whose behavior is characterized by the general case assumptions, i.e., assumptions I through VI outlined in Section 3.2 are studied. The first example assumes that the system is a 3 × 2 multiprocessor system. The second example assumes that the system is an 8 × 8 multiprocessor system. These two examples illustrate the usage of the SMI model in the general case.

**Example 4.1:**

A multiprocessor system that manages a large data base is considered. The system consists of two identical *PE*s with private cache, two identical logical buffers, and an I/O channel used for DMA between the logical buffers and a fixed head disk. The two identical *PE*s, $PE_1$ and $PE_2$, will request the first logical buffer with probability $x$ and the second logical buffer with probability $(1-x)$. The I/O channel, $PE_3$, requests the first logical buffer with probability $y$ and the second logical buffer with probability $(1-y)$.

In this example the problem of deciding which is the better of two mappings from the logical buffers to the physical memory modules is considered. In the first case, the logical buffers are mapped exactly into the physical memory modules. In the second case, each logical buffer is divided in half and each half is placed in a physical memory module. The two mappings are illustrated in Figure 5.27. It is noticed in the second case the $a_{ij} = 0.5$ $(i=1,2,3;\ j=1,2)$ since $x/2 + (1-x)/2 = 0.5$, etc. In both cases the connection time between the *PE*s and the *MM*s is relatively short and equal to the time needed to transfer a line between the private cache and the *MM*. Due to the coherency checks employed by the system, the connection time between the identical *PE*s and the *MM*s is variable. The connection time between the I/O channel and the *MM*s is relatively long. This connection time has two components: the rotational delay and the data transfer time. The rotational delay can be characterized by a random variable that is uniformly distributed between zero and the length of time needed by the disk to make a full rotation. The data transfer time can be characterized as a deterministic random variable. The transfer

time is the time needed to transfer a fixed block, e.g., a cylinder or a track, between the disk and the *MM*. Therefore, the connection time between the I/O channel and the *MM* can be characterized as a random variable with low $C_v$.

The SMI model is used to analyze the two cases mentioned here given the following operation conditions:

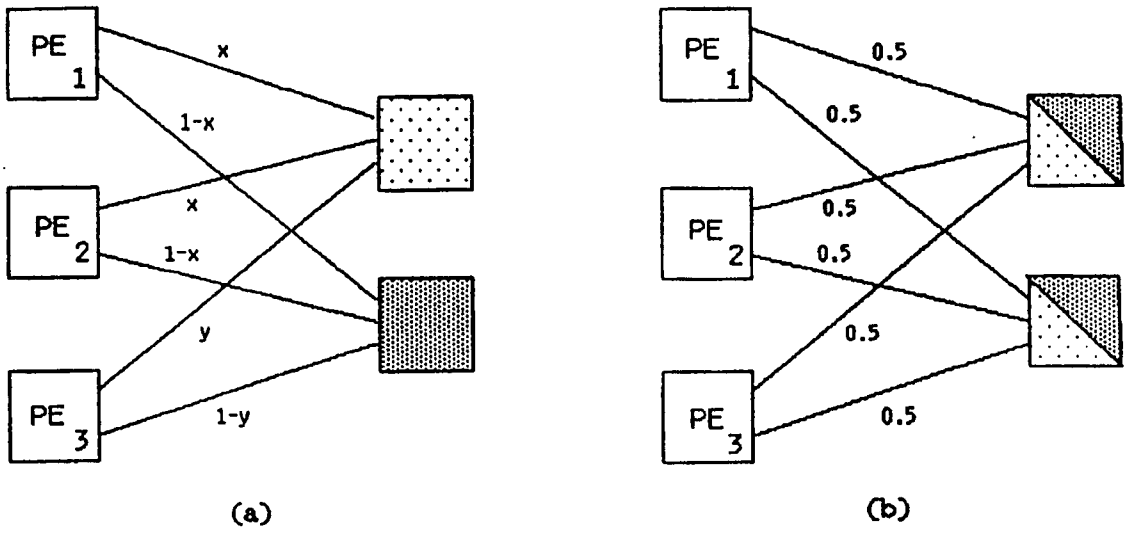



**Figure 5.27 The two different mappings of Example 4.1.**

$$\overline{T}_1 = \overline{T}_2 = 4.0 \ ,$$
$$\overline{T}_3 = 12.0 \ ,$$
$$\overline{C}_{11} = \overline{C}_{12} = 4.0 \ ,$$
$$\overline{C}_{21} = \overline{C}_{22} = 4.0 \ ,$$
$$\overline{C^2}_{11} = \overline{C^2}_{12} = 26.24 \ ,$$
$$\overline{C^2}_{21} = \overline{C^2}_{22} = 26.24 \ ,$$
$$\tilde{C}_{31} = \tilde{C}_{32} = \textit{UNIFORM}\ (14,18) \ ,$$
$$x = 0.9$$

and

$$y = 0.0 \ ,$$

where the distribution *UNIFORM* (14,18) is a uniform distribution between 14 and 18, i.e., data transfers take 14 cycles and the rotational delay takes between zero and four cycles.

Table 5.1 shows the results obtained from the simulations and from the SMI model. The SMI model agrees closely with the simulations and, in practice, would be used because it requires far less computation. As expected, the mapping of Case 1 yields better performance in most categories. The only exceptions being $W_{12}$ and $W_{22}$. However, the overall waiting time for $PE_i$ is given by:

$$W_i = \sum_{j=1}^{2} a_{ij} \ W_{ij}$$

and $a_{12}, a_{22} = 0.1$ in Case 1. Thus, the overall waiting times for $PE_1$ and $PE_2$ are also less in Case 1. ∎

**Example 4.2:**

In this example a multiprocessor system that has eight nonidentical *PE*s and eight nonidentical *MM*s is considered. Some of the system parameters have been chosen in random to check the robustness of the SMI model. The random variables, $\tilde{D}_i$s, that determine the destination of requests have the same distribution as in the mailbox case. In other words the requests will be directed to $MM_1$ with probability $x$ and to any other *MM* with probability $(1-x)/(M-1)$.

| Measure | Case # 1 | | Case # 2 | |
|---|---|---|---|---|
| | Simulation | %Error | Simulation | %Error |
| $BW$ | 1.39415 | -2.28 | 1.20084 | -1.0 |
| $PU_1$ | 0.8282 | -2.75 | 0.66214 | -3.33 |
| $PU_2$ | 0.83041 | -3.01 | 0.65778 | -2.69 |
| $PU_3$ | 0.98917 | 0.29 | 0.93723 | 2.47 |
| $MU_1$ | 0.74607 | -3.64 | 0.60502 | -1.75 |
| $MU_2$ | 0.64685 | -0.53 | 0.59582 | -0.23 |
| $L_1$ | 0.21997 | -3.51 | 0.38157 | -0.48 |
| $L_2$ | 0.12223 | 3.48 | 0.36931 | 2.82 |
| $W_{11}$ | 1.25232 | -1.59 | 4.15107 | 3.56 |
| $W_{12}$ | 5.4316 | 7.18 | 3.9695 | 8.23 |
| $W_{21}$ | 1.21373 | 1.54 | 4.15194 | 3.54 |
| $W_{22}$ | 5.54553 | 4.98 | 3.9923 | 7.68 |
| $W_{31}$ | 0.0 | 0.0 | 1.94831 | -9.98 |
| $W_{32}$ | 0.30673 | 6.09 | 1.87394 | -6.41 |

**Table 5.1 Comparisons between the two cases of Example 4.1.**

| The Pair $PE_i$ and $MM_j$ | Connection Time Distribution Functions | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $i,j$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1,1 | 0.5 | 0.0 | 0.3 | 0.0 | 0.1 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 |
| 1,2 | 0.0 | 0.2 | 0.2 | 0.1 | 0.1 | 0.3 | 0.1 | 0.0 | 0.0 | 0.0 |
| 1,3 | 0.0 | 0.0 | 0.5 | 0.0 | 0.0 | 0.3 | 0.1 | 0.1 | 0.0 | 0.0 |
| 1,4 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| 1,5 | 0.2 | 0.0 | 0.4 | 0.0 | 0.1 | 0.0 | 0.0 | 0.3 | 0.0 | 0.0 |
| 1,6 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 1,7 | 0.3 | 0.3 | 0.0 | 0.3 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 | 0.0 |
| 1,8 | 0.0 | 0.1 | 0.0 | 0.1 | 0.0 | 0.0 | 0.1 | 0.1 | 0.2 | 0.4 |
| 2,1 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.5 |
| 2,2 | 0.1 | 0.2 | 0.0 | 0.1 | 0.0 | 0.1 | 0.0 | 0.2 | 0.1 | 0.2 |
| 2,3 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2,4 | 0.0 | 0.0 | 0.3 | 0.0 | 0.2 | 0.1 | 0.1 | 0.3 | 0.0 | 0.0 |
| 2,5 | 0.0 | 0.4 | 0.0 | 0.2 | 0.0 | 0.2 | 0.0 | 0.0 | 0.2 | 0.0 |
| 2,6 | 0.0 | 0.1 | 0.2 | 0.0 | 0.2 | 0.0 | 0.2 | 0.0 | 0.0 | 0.3 |
| 2,7 | 0.0 | 0.0 | 0.1 | 0.3 | 0.1 | 0.0 | 0.2 | 0.1 | 0.2 | 0.0 |
| 2,8 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| 3,1 | 0.0 | 0.0 | 0.4 | 0.0 | 0.2 | 0.3 | 0.0 | 0.1 | 0.0 | 0.0 |
| 3,2 | 0.0 | 0.2 | 0.1 | 0.0 | 0.3 | 0.2 | 0.1 | 0.0 | 0.1 | 0.0 |
| 3,3 | 0.1 | 0.0 | 0.1 | 0.0 | 0.2 | 0.3 | 0.0 | 0.0 | 0.0 | 0.3 |
| 3,4 | 0.0 | 0.1 | 0.0 | 0.0 | 0.1 | 0.2 | 0.0 | 0.2 | 0.3 | 0.1 |
| 3,5 | 0.2 | 0.0 | 0.2 | 0.0 | 0.2 | 0.0 | 0.2 | 0.0 | 0.2 | 0.0 |
| 3,6 | 0.0 | 0.3 | 0.1 | 0.3 | 0.0 | 0.0 | 0.1 | 0.1 | 0.0 | 0.1 |
| 3,7 | 0.3 | 0.1 | 0.0 | 0.2 | 0.0 | 0.0 | 0.0 | 0.2 | 0.1 | 0.1 |
| 3,8 | 0.6 | 0.0 | 0.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.2 |
| 4,1 | 0.2 | 0.1 | 0.0 | 0.1 | 0.0 | 0.2 | 0.1 | 0.0 | 0.0 | 0.3 |
| 4,2 | 0.0 | 0.0 | 0.4 | 0.0 | 0.4 | 0.0 | 0.0 | 0.1 | 0.1 | 0.0 |
| 4,3 | 0.1 | 0.0 | 0.0 | 0.3 | 0.0 | 0.0 | 0.2 | 0.0 | 0.2 | 0.2 |
| 4,4 | 0.0 | 0.1 | 0.6 | 0.0 | 0.1 | 0.1 | 0.0 | 0.1 | 0.0 | 0.0 |
| 4,5 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| 4,6 | 0.2 | 0.0 | 0.2 | 0.0 | 0.2 | 0.0 | 0.0 | 0.2 | 0.0 | 0.2 |
| 4,7 | 0.0 | 0.2 | 0.0 | 0.2 | 0.0 | 0.2 | 0.2 | 0.0 | 0.2 | 0.0 |
| 4,8 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |

**Table 5.2 The Distributions of the connection times of example 4.2.**

| The Pair $PE_i$ and $MM_j$ | Connection Time Distribution Functions | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $i,j$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 5,1 | 0.2 | 0.1 | 0.1 | 0.1 | 0.2 | 0.1 | 0.1 | 0.1 | 0.0 | 0.0 |
| 5,2 | 0.1 | 0.0 | 0.3 | 0.0 | 0.0 | 0.4 | 0.0 | 0.0 | 0.1 | 0.1 |
| 5,3 | 0.0 | 0.2 | 0.0 | 0.4 | 0.1 | 0.0 | 0.2 | 0.1 | 0.0 | 0.0 |
| 5,4 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 5,5 | 0.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.2 | 0.0 | 0.0 | 0.0 | 0.6 |
| 5,6 | 0.0 | 0.4 | 0.0 | 0.2 | 0.0 | 0.0 | 0.0 | 0.4 | 0.0 | 0.0 |
| 5,7 | 0.0 | 0.0 | 0.3 | 0.0 | 0.4 | 0.0 | 0.3 | 0.0 | 0.0 | 0.0 |
| 5,8 | 0.0 | 0.1 | 0.0 | 0.1 | 0.0 | 0.2 | 0.0 | 0.2 | 0.0 | 0.4 |
| 6,1 | 0.1 | 0.0 | 0.4 | 0.0 | 0.0 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 |
| 6,2 | 0.0 | 0.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.7 | 0.0 | 0.1 |
| 6,3 | 0.0 | 0.0 | 0.3 | 0.0 | 0.2 | 0.0 | 0.0 | 0.0 | 0.5 | 0.0 |
| 6,4 | 0.1 | 0.0 | 0.0 | 0.1 | 0.0 | 0.1 | 0.1 | 0.2 | 0.2 | 0.2 |
| 6,5 | 0.0 | 0.3 | 0.0 | 0.2 | 0.3 | 0.0 | 0.2 | 0.0 | 0.0 | 0.0 |
| 6,6 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 6,7 | 0.4 | 0.0 | 0.2 | 0.0 | 0.1 | 0.1 | 0.1 | 0.0 | 0.1 | 0.0 |
| 6,8 | 0.0 | 0.3 | 0.0 | 0.3 | 0.0 | 0.3 | 0.0 | 0.0 | 0.0 | 0.1 |
| 7,1 | 0.4 | 0.0 | 0.4 | 0.0 | 0.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 7,2 | 0.0 | 0.0 | 0.0 | 0.2 | 0.0 | 0.0 | 0.4 | 0.0 | 0.0 | 0.4 |
| 7,3 | 0.0 | 0.1 | 0.3 | 0.0 | 0.6 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 7,4 | 0.1 | 0.2 | 0.3 | 0.4 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 7,5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.3 | 0.0 | 0.3 | 0.3 | 0.1 |
| 7,6 | 0.1 | 0.1 | 0.0 | 0.1 | 0.1 | 0.2 | 0.2 | 0.1 | 0.1 | 0.0 |
| 7,7 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.3 | 0.0 | 0.3 | 0.3 | 0.1 |
| 7,8 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| 8,1 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 8,2 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 8,3 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 8,4 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 8,5 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 8,6 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 8,7 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 8,8 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |

**Table 5.2** continued

The think time can be described as follows: the even number *PE*s have a deterministic think time of zero cycles; the think time of the odd number *PE*s is uniformly distributed between zero and four cycles. The distribution of the connection time between any *PE* and an *MM* has been chosen randomly. The full distribution of the different connection times are shown in Table 5.2 where the row represents the *PE-MM* pair and the column number represents the length of the connection time.

The performance measures obtained from the simulations in this case are shown in Table 5.3 with the relative percentage error of the SMI model. It can be seen that the model accuracy is similar to the accuracy demonstrated in the previous examples. In spite of the generality and robustness of this example, the SMI model produced good results for the multiprocessor system behavior. ∎

These cases by no means imply that a similar percentage error will be obtained if any other multiprocessor systems are studied using the SMI model. However, the previous systems indicate that the SMI model works sufficiently well for modeling the memory interference problem in a multiprocessor system.

| Measure | Simulation | Model | %Error |
|---|---|---|---|
| $BW$ | 4.259 | 4.222 | -0.87 |
| $PU_1$ | 0.557 | 0.545 | -2.15 |
| $PU_2$ | 0.479 | 0.484 | 1.04 |
| $PU_3$ | 0.598 | 0.595 | -0.5 |
| $PU_4$ | 0.516 | 0.523 | 1.36 |
| $PU_5$ | 0.620 | 0.609 | -1.77 |
| $PU_6$ | 0.483 | 0.493 | 2.07 |
| $PU_7$ | 0.600 | 0.581 | -3.17 |
| $PU_8$ | 0.407 | 0.392 | -3.69 |
| $MU_1$ | 0.785 | 0.793 | 1.02 |
| $MU_2$ | 0.473 | 0.474 | 0.21 |
| $MU_3$ | 0.430 | 0.414 | -3.72 |
| $MU_4$ | 0.463 | 0.466 | 0.65 |
| $MU_5$ | 0.510 | 0.499 | -2.16 |
| $MU_6$ | 0.532 | 0.516 | -3.01 |
| $MU_7$ | 0.483 | 0.480 | -0.62 |
| $MU_8$ | 0.583 | 0.580 | -0.51 |
| $L_1$ | 1.206 | 1.203 | -0.25 |
| $L_2$ | 0.224 | 0.233 | 4.02 |
| $L_3$ | 0.167 | 0.163 | -2.40 |
| $L_4$ | 0.203 | 0.215 | 5.91 |
| $L_5$ | 0.270 | 0.267 | -1.11 |
| $L_6$ | 0.286 | 0.288 | 0.7 |
| $L_7$ | 0.242 | 0.241 | -0.41 |
| $L_8$ | 0.399 | 0.426 | 6.77 |

**Table 5.3 The results of example 4.2.**

| Measure | Simulat. | Model | %Error | Measure | Simulat. | Model | %Error |
|---|---|---|---|---|---|---|---|
| $W_{11}$ | 5.590 | 6.361 | 13.79 | $W_{51}$ | 5.212 | 5.186 | -0.5 |
| $W_{12}$ | 2.622 | 2.689 | 2.56 | $W_{52}$ | 2.386 | 2.523 | 5.74 |
| $W_{13}$ | 1.857 | 1.744 | -6.09 | $W_{53}$ | 1.613 | 1.780 | 10.35 |
| $W_{14}$ | 2.247 | 2.228 | -0.85 | $W_{54}$ | 2.174 | 2.406 | 10.67 |
| $W_{15}$ | 2.808 | 3.157 | 12.43 | $W_{55}$ | 2.537 | 2.496 | -1.62 |
| $W_{16}$ | 3.277 | 3.053 | -6.84 | $W_{56}$ | 3.264 | 3.338 | 2.27 |
| $W_{17}$ | 2.718 | 3.065 | 12.77 | $W_{57}$ | 2.734 | 2.707 | -0.99 |
| $W_{18}$ | 4.280 | 4.066 | -5.0 | $W_{58}$ | 3.960 | 4.312 | 8.89 |
| $W_{21}$ | 5.264 | 4.661 | -11.46 | $W_{61}$ | 6.242 | 5.585 | -10.53 |
| $W_{22}$ | 2.389 | 2.508 | 4.98 | $W_{62}$ | 2.727 | 2.408 | -11.7 |
| $W_{23}$ | 2.070 | 2.283 | 10.29 | $W_{63}$ | 1.714 | 1.658 | -3.27 |
| $W_{24}$ | 2.373 | 2.390 | 0.72 | $W_{64}$ | 2.210 | 2.163 | -2.13 |
| $W_{25}$ | 3.263 | 3.232 | -0.95 | $W_{65}$ | 3.498 | 3.328 | -4.86 |
| $W_{26}$ | 3.304 | 3.178 | -3.81 | $W_{66}$ | 2.838 | 2.918 | 2.82 |
| $W_{27}$ | 2.703 | 2.637 | -2.44 | $W_{67}$ | 2.790 | 3.018 | 8.17 |
| $W_{28}$ | 4.611 | 5.108 | 10.78 | $W_{68}$ | 4.912 | 5.412 | 10.18 |
| $W_{31}$ | 5.371 | 4.811 | -10.43 | $W_{71}$ | 5.912 | 6.364 | 7.65 |
| $W_{32}$ | 2.441 | 2.583 | 5.82 | $W_{72}$ | 2.138 | 2.114 | -1.12 |
| $W_{33}$ | 1.613 | 1.527 | -5.33 | $W_{73}$ | 1.744 | 1.881 | 7.86 |
| $W_{34}$ | 1.920 | 1.956 | 1.88 | $W_{74}$ | 2.523 | 2.763 | 9.51 |
| $W_{35}$ | 3.230 | 3.016 | -6.63 | $W_{75}$ | 2.556 | 2.468 | -3.44 |
| $W_{36}$ | 2.985 | 3.372 | 12.96 | $W_{76}$ | 2.969 | 3.192 | 7.51 |
| $W_{37}$ | 2.678 | 2.674 | -0.15 | $W_{77}$ | 2.356 | 2.159 | -8.36 |
| $W_{38}$ | 5.393 | 5.657 | 4.9 | $W_{78}$ | 4.577 | 4.916 | 7.41 |
| $W_{41}$ | 5.397 | 4.802 | -11.02 | $W_{81}$ | 6.950 | 7.739 | 11.35 |
| $W_{42}$ | 2.480 | 2.750 | 10.89 | $W_{82}$ | 2.927 | 3.196 | 9.19 |
| $W_{43}$ | 1.715 | 1.647 | -3.97 | $W_{83}$ | 2.241 | 2.059 | -8.12 |
| $W_{44}$ | 2.519 | 2.675 | 6.19 | $W_{84}$ | 2.539 | 2.641 | 4.02 |
| $W_{45}$ | 2.939 | 3.104 | 5.61 | $W_{85}$ | 3.211 | 3.173 | -1.18 |
| $W_{46}$ | 3.766 | 3.371 | -10.49 | $W_{86}$ | 2.977 | 3.232 | 8.57 |
| $W_{47}$ | 2.981 | 2.735 | -8.25 | $W_{87}$ | 2.649 | 2.465 | -6.95 |
| $W_{48}$ | 4.471 | 4.065 | -9.08 | $W_{88}$ | 4.286 | 4.418 | 3.08 |

**Table 5.3** continued

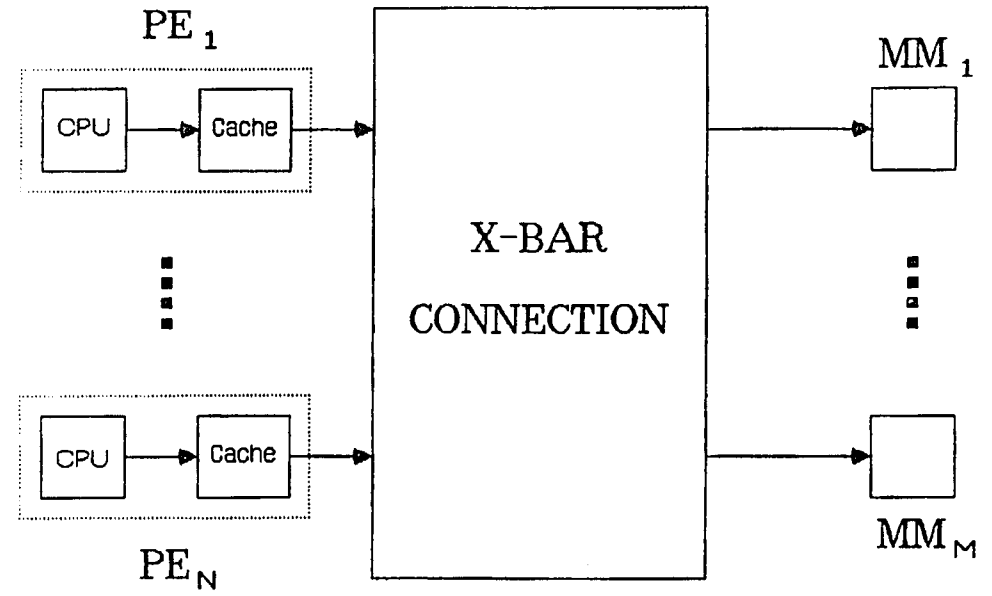
# CHAPTER VI

# CACHE MODEL EXAMPLE

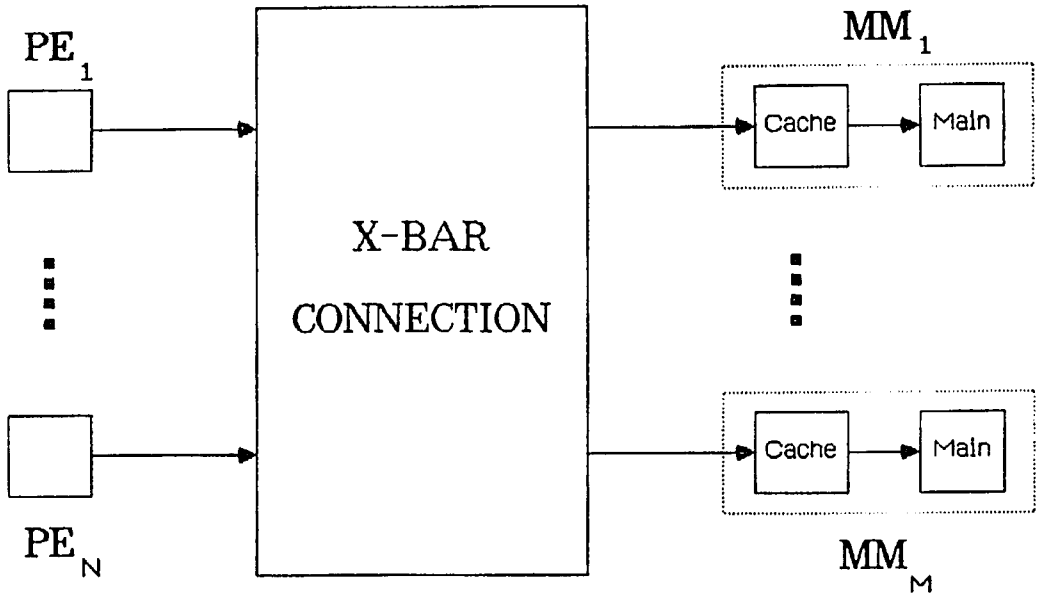
## 6.1. Introduction

In this chapter a multiprocessor system with cache memories to illustrate the usage of the SMI model in the design phase of such systems is discussed. Different implementations of the system are discussed and the SMI model is used to analyze the behavior of these different implementations. Furthermore, this example motivates and justifies the assumption of a variable connection time between a processing element and a memory module. Simulation results are used to confirm the accuracy of the SMI model.

A number of studies, see [Lip68, Mea70, KaW73, Str76, Smi78a, Rao78, Cla83], demonstrated the high performance and the cost effectiveness of the cache memories in the uniprocessor case. A detailed survey of cache memories is presented in [Smi82]. Cache memories can be defined as small, high-speed buffer memories used in modern computer systems to hold temporarily those sections of the main memory that have the likelihood or the potential to be used by the processor at that time. Information located in the cache memory may be accessed in much less time than that located in the main memory. For example, in a typical large, high-speed computer the main memory can be accessed in 300 to 600 nanoseconds; on the other hand, the cache can be accessed in 50 to 100 nanoseconds. The fixed size unit of information to be transferred between the cache memory and the main memory is referred to, in this chapter, as the block. Hence, the block resembles conceptually the page between the main memory and the secondary memory. It is noted that some studies refer to this quantum as the cache line, see [Smi82].




(a)




(b)

**Figure 6.1 The multiprocessor system with cache memories.**

In a multiprocessor system the cache memories can be placed in either side of the interconnection network, as shown in Figure 6.1. The first system, shown in Figure 6.1(a), is a multiprocessor system with private cache memories. The main advantage of this design is the high transfer rate between the processor and its private cache, since both can be placed on the same chip using VLSI technology. The transfer rate on the chip is much higher than the transfer rate across the chip boundaries. However, the main drawback of this design is the memory coherency problem, i.e., the existence of more than one copy of a block in different cache memories. The second system, shown in Figure 6.1(b), is a multiprocessor system with shared cache memories. This system will not have a cache coherency problem, since there will be only one copy of the block at any time. Furthermore, this design provides high cache utilization due to its dynamic space sharing. However, the cache access interference will degrade the performance of the multiprocessor system. In the next sections, both of these systems are studied in detail.

## 6.2. Private Cache System

The multiprocessor system, depicted in Figure 6.1(a), has a two-level memory hierarchy. The first level of memory is a private cache memory, while the second level is a shared main memory. As mentioned earlier the main drawback of this design is the cache coherency problem. Before describing the multiprocessor system shown in Figure 6.1(a), three major issues in private cache memories are introduced and discussed: first, the cache coherency solutions; second, policies of updating the main memory; third, the cache memory organization.

Three approaches are proposed in the literature to solve the cache coherency problem. These approaches are: dynamic, static, and quasi-dynamic solutions. The dynamic solution checks at run-time the presence of the block referenced by a processor in other caches, hence, a central directory and global and local flags must be maintained dynamically. Furthermore, each block will be identified as shared (read-only) or private (only one copy allowed in the private caches at any time). This solution was proposed in [Tan76, CeF78], and [DuB82] studied the effects of enforcing this policy on the system performance. The major drawback of this approach is its cost

and the overhead created by the access conflicts to the central directory. The second approach is the static solution in which each page in the shared memory is tagged as cachable or noncachable. Hence, only shared writeable data will be tagged as noncachable, see [BrD83]. The multiprocessor system C.mmp, see [WLH81], used a similar approach by storing only the "read-only" data in the cache. The third approach is the quasi-dynamic solution proposed in [BrD83]. In this approach all the blocks are tagged as cachable initially. A block that is cached by one processor and is referenced by another processor for a write operation becomes tagged as a shared writeable block. Such a block will be tagged noncachable after updating. Accesses to noncachable data are made on a word-by-word basis. However, in order to speedup access to noncachable data, the shared memory can be partitioned into cachable and noncachable data spaces where the noncachable data are stored in faster memory modules.

Another aspect of private cache memories are the policies to update the main memory. Some of these policies are: write-through (store-through), buffered-write-back, load-through, and write-back-write-allocate, see [KaW73]. These four policies are summarized as follows: If the write-through policy is adopted, a processor attempts to write a word in a cache and the word is stored in the main memory, regardless whether the corresponding block is present in the cache or not. In the case of the buffered-write-back policy on a cache miss the replaced block will be stored in a high-speed buffer. Thereafter, the new block will be loaded from the main memory and the replaced block will be stored in the main memory from the high-speed buffer. The third policy is the load-through policy. On a cache miss for a read reference, the desired word is loaded directly into a CPU register from the main memory after which the whole block containing that word is read into the cache module. The motive for this policy is to overlap the CPU execution time and the transfer time of a block; it can be combined with any one of the previous two policies for the case of a write miss. The last policy is the write-back-write-allocate policy in which the replaced page in the cache will be overwritten by the new page from the main memory only if the replaced page has not been modified, otherwise the replaced page will be written in the main memory before the new page is moved into the cache. It is noted that the probability that the replaced

block has been modified depends on the program behavior and the cache organization. Nevertheless, it is usually larger than the probability that a particular reference is a write, see [Smi79].

The last issue to be discussed is the cache memory organization (or placement algorithm). The two organizations that are of interest are fully associative mapping and set associative mapping. A set consists of a number of blocks and it is stored in one memory module. Fully associative mapping assumes that the replaced block and the new block are not necessarily from the same set. However, set associative mapping assumes that the replaced block and the new block belong to the same set. Another major aspect in the cache memory is the replacement policy. This policy is used to select the cache block that will be replaced by the new block, see [Smi78b] for extensive discussion on this subject. However, the replacement policy is not a major issue in the modeling of the system, hence it is not discussed here.

Most of the cache performance studies [BrD81a, BrD81b, Pat82] ignored the issue of cache coherency and assume an environment in which data consistency is not a problem. A recent study in [BrD83] examined a multiprocessor system with private caches and used a static or quasi-dynamic solution for the cache coherency. This study used a two-dimensional memory, called the L-M memory organization, for the shared main memory. However, it used only the first moment of the connection time between the private cache module and the shared main memory module. This model produces "good" results if the variation in the connection time is small, i.e., $C_v$ of the connection time is less than 0.5. The results of Chapter V demonstrate that the performance measures will not change drastically by changing $C_v$ from 0 to 0.5, however, changing $C_v$ to higher values yields a significant change in the performance measures. Therefore, the use of the second moment of the connection time should eliminate this deficiency in the model.

In this example a multiprocessor system with private caches as shown in Figure 6.1(a) are studied. Each main memory module is divided logically into two spaces, cachable and noncachable spaces. It is noted that each module consists of two physical modules sharing the same line, and the faster memory will be used to store the noncachable data. The system adopts the static or quasi-dynamic solution for the cache coherency problem. The SMI model does not differentiate

between these two solutions. The policy used to update the main memory is the write-back-write-allocate policy. The set associative approach is used in the cache organization, nevertheless, the impact on the SMI model if the fully associative approach is used is discussed.

The operations of the multiprocessor system can be described as follows. The system has $N$ identical processors, each has a private cache, connected to $M$ main memory modules via a crossbar interconnection network. The system is synchronized by a master clock. At the beginning of any cycle the processor will make a reference to the memory with probability $r$. The memory reference can be made to the cache or to the non-cachable space in the shared main memory. A cache hit will happen with probability $h$. In case of a cache miss the private cache will try to retrieve the requested block from the shared main memory. Hence, the cache will request a connection with the main memory module that has the needed block. The sets of the blocks will be randomly distributed between the main memory modules, therefore, in case of a cache miss the private cache will request a particular module with probability $1/M$. The probability that the replaced block has been modified is $w$. The cache block will need $S$ cycles to transfer between the private cache and the shared main memory. Therefore, if the replaced block has not been modified, the connection between the private cache module and the shared main memory module will last for $S$ cycles, however, if the replaced block has been modified, the connection will last for $2S$ cycles. The probability that the processor reference a noncachable reference is $x$. In this case the processor will request a connection to one of the shared memory modules; the probability it will request a particular memory module is $1/M$. The connection between the processor and the memory module will last for one cycle in order to transfer one word from the noncachable space in the shared main memory to the private cache memory.

It can be seen that the SMI model under the uniform assumptions can be used to analyze the multiprocessor system described in the previous paragraph. The model input parameters can be defined as follows:

$$\overline{T} = \frac{1}{r\,[\,x + (1-x)(1-h)\,]} \quad ,$$

$$\overline{C} = \frac{x}{x + (1-x)(1-h)} + \frac{(1-x)(1-h)}{x + (1-x)(1-h)} \left[\, S\,(1-w) + 2S\,w \,\right]$$

and

$$\overline{C^2} = \frac{x}{x + (1-x)(1-h)} + \frac{(1-x)(1-h)}{x + (1-x)(1-h)} \left[\, S^2(1-w) + 4S^2\,w \,\right] \quad .$$

The average thinking time has been calculated as the average value of a geometric random variable in which the probability of success is the probability that a reference is made to noncachable data or the reference is made to cachable data but it was a cache miss, i.e., the probability of success is $r\,\left[x + (1-x)(1-h).\right]$ The average connection time has been calculated from the probability mass function of the connection time. The probability that the connection time will last for one cycle is the probability that the processor made a reference to the noncachable space of the shared main memory given that the request to access the main memory has been made by the processor, i.e., the probability is $x/\left[\, x + (1-x)(1-h)\,\right]$. The probability that the connection time will last $S$ cycles is the probability that the reference made by the processor causes a cache miss and the replaced block has not been modified, i.e., the probability is $(1-w)(1-x)(1-h)/\left[\, x + (1-x)(1-h)\,\right]$. The probability that the connection time will last $2\,S$ cycles is the probability that the reference made by the processor causes a cache miss and the replaced block has been modified, i.e., the probability is $w\,(1-x)(1-h)/\left[\, x + (1-x)(1-h)\,\right]$. Furthermore, from this probability mass function the second moment of the connection time can be calculated.

To check the validity of the model in this case, the following system is used as a case study. The system has 16 processors, each has its own private cache memory; the system also has a shared main memory distributed between $M$ memory modules. Furthermore, the multiprocessor system has the following parameters:

$$r = 0.4, \quad h = 0.95, \quad w = 0.3, \quad x = 0.1 \quad .$$

Table 6.1 shows the simulation results of the performance measures in the cases where $M = 4, 8, 16,$ *or* $32$ and $S = 4$ *or* $8$. The relative percentage error, $\%Error$, of the

| M | S | RESULTS | | |
|---|---|---|---|---|
| | | Measure | Simulation | %Error |
| 4 | 4 | *BW* | 1.7686 | 0.38 |
| | | *PU* | 0.8205 | 0.83 |
| | | *L* | 0.2674 | -7.12 |
| | | *W* | 1.4068 | -8.03 |
| | 8 | *BW* | 2.3528 | 2.84 |
| | | *PU* | 0.6499 | 2.42 |
| | | *L* | 0.8022 | -8.67 |
| | | *W* | 5.2403 | -9.46 |
| | 16 | *BW* | 2.7894 | 3.86 |
| | | *PU* | 0.4185 | 4.42 |
| | | *L* | 1.6518 | -7.5 |
| | | *W* | 17.09823 | -11.84 |
| | 32 | *BW* | 2.9375 | 5.99 |
| | | *PU* | 0.23423 | 5.32 |
| | | *L* | 2.3467 | -4.77 |
| | | *W* | 42.56964 | -8.27 |
| 16 | 4 | *BW* | 1.8631 | -0.11 |
| | | *PU* | 0.87 | 0.07 |
| | | *L* | 0.0136 | -3.61 |
| | | *W* | 0.2688 | -3.62 |
| | 8 | *BW* | 2.8344 | 0.18 |
| | | *PU* | 0.7792 | 0.25 |
| | | *L* | 0.0437 | -5.13 |
| | | *W* | 0.9659 | -5.37 |
| | 16 | *BW* | 4.0878 | 1.83 |
| | | *PU* | 0.6233 | 0.72 |
| | | *L* | 0.1212 | -7.58 |
| | | *W* | 3.3516 | -8.24 |
| | 32 | *BW* | 5.31154 | 2.84 |
| | | *PU* | 0.4215 | 2.68 |
| | | *L* | 0.2465 | -8.4 |
| | | *W* | 10.077 | -10.74 |

**Table 6.1 The simulation results from the private cache case study.**

performance measures is shown also in Table 6.1. It is clear that the SMI model produces accurate results for these operating points. It is note that the processor utilization, $PU_i$, is defined as the probability that the $PE_i$ is thinking only, because in this case when the processor is accessing, it is doing a useless job since it is waiting for its cache to be updated. This definition of $PU_i$ is used in this chapter.

In the preceding discussion it is assumed that the multiprocessor system uses the set associative policy to organize the cache memory. The fully associative policy can also be studied using the SMI model. However, the semi-Markov process (SMP) of the SMI model must be modified in order to accommodate the new policy. The modified SMP is shown in Figure 6.2. The main difference between the SMP of Figure 6.2 and the SMP of the SMI model under the uniform case is that the SMP of Figure 6.2 must accommodate the fact that the connection time might be done with two different modules in the case where the replaced block has been modified. In that case

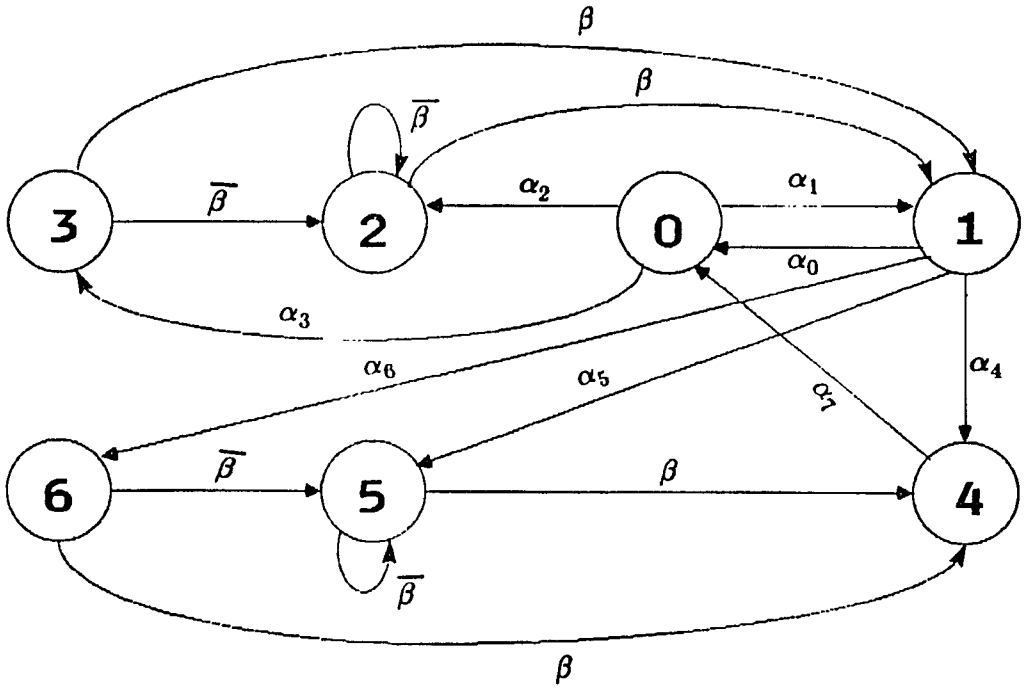



**Figure 6.2 The SMP of the processor with fully associative policy.**

the processor first writes the replaced module back in one of the modules then it reads the new block from the other module. Hence, the process enters the three new states (4, 5, and 6) in the case of reading the new block after writing back, to the main memory, the modified replaced block. In other words, the process leaves state 1 for state 0 if the request was for a noncachable data or the replaced block has not been modified. Otherwise, the process enters state 4, 5, or 6 depending on the state of the module that has the new block. The average sojourn times of the SMP states, in this case, are as follows:

$$
\begin{aligned}
\eta_0 &= \overline{T} \quad , \\
\eta_1 &= \frac{x}{x + (1-x)(1-h)} + \frac{(1-x)(1-h)}{x + (1-x)(1-h)} S \quad , \\
\eta_2 &= \eta_1 \quad , \\
\eta_3 &= \frac{S}{2} \quad , \\
\eta_4 &= S \quad , \\
\eta_5 &= \eta_2
\end{aligned}
$$

and

$$
\eta_6 = \eta_3 \quad .
$$

The transition probabilities between the states are as follows:

$$
\alpha_j = \begin{cases} 1 - w(1-x) & j=0 \\ WIN\ (1-BUSY) & j=1 \\ (1-WIN)\ (1-BUSY) & j=2 \\ BUSY & j=3 \\ w\ (1-x)\ WIN\ (1-BUSY) & j=4 \\ w\ (1-x)\ (1-WIN)\ (1-BUSY) & j=5 \\ w\ (1-x)\ BUSY & j=6 \\ 1 & j=7 \end{cases}
$$

$$
\begin{aligned}
\beta &= WIN\ (1-BUSY) \\
\overline{\beta} &= 1 - \beta
\end{aligned}
$$

The four subsets of the state space are: $S_i^{th} = \{0\}$, $S_i^{ac} = \{1,4\}$, $S_i^{/w} = \{2,5\}$, and $S_i^{rw} = \{3,6\}$. This SMP can be solved using the same technique as that used for the SMI model. The solution of the model in this case is straightforward and the accuracy of the result should be similar to the accuracy of Table 6.1.

## 6.3. Shared Cache System

As mentioned earlier, the private cache memory has two resident disadvantages. First, the cache coherency problem. Second, the existence of multiple copies of sharded system resources, e.g., the operating system routines, in the private cache memories if more than one processor reference these resources. The shared cache memory is an alternative approach that was investigated by [YPD83] and [Yeh81]. This approach has some resident advantages over the private cache memory approach. Cache coherency problem can be solved without any overhead penalty or cost. Furthermore, this approach supplies a dynamic space sharing, hence, the cache available for one processor is the total cache in the multiprocessor system. Therefore, the cache hit ratio is high. A third advantage of the shared cache memory is that the shared information will have only one copy in the cache. On the other hand, the shared cache memory has some major disadvantages. The major problem is the access conflict problem that will degrade the performance of the multiprocessor system. Moreover, in the shared cache memory approach the rate of data transfer between the processor and the cache is less than the rate in the case of the private cache. The reason for this is that the private cache can be placed on the same chip with the processor, however, in the shared cache approach this implementation is not possible.

The studies reported in [YPD83] and [Yeh81] studied a multiprocessor system with private cache memories. The system is depicted in Figure 6.1(b). However, these studies assume that the processor is a pipelined processor and the share cache memory is organized as an L-M memory organization. The studies concluded that the shared cache systems may perform better than the private cache systems if the shared cache systems provide a higher hit ratio, $h$, than private cache systems. The penalty of access interference can be reduced with a reasonable choice of system parameters.

The operations of the multiprocessor system in this case can be described as follows. The system has $N$ identical processors connected to $M$ shared cache memory modules via a crossbar interconnection network. Each of the cache modules is connected to a main memory module. The system is synchronized by a master clock. At the beginning of any cycle the processor will make a

reference to the cache memory with probability $r$. A cache hit will happen with probability $h$. In case of a cache miss, the processor will request a connection with the cache memory module that has the needed block. The sets of the blocks will be randomly distributed between the cache memory modules, therefore, in case of a cache miss the processor will request a particular module with probability $1/M$. The probability that the replaced block has been modified is $w$. The cache block will need $S$ cycles to transfer between the cache memory module and the main memory module. Therefore, if the replaced block has not been modified, the connection between the processor and the shared cache memory module will last for $S$ cycles, however, if the replaced block has been modified, the connection will last for $2S$ cycles. In case of a cache hit, the connection will last for one cycle which is the time needed to transfer that particular reference from the shared cache module and the processor.

It can be seen that the SMI model under the uniform assumptions can be used to analyze the multiprocessor system described in the previous paragraph. The model input parameters can be defined as follows:

$$\overline{T} = \frac{1}{r} - 1 \quad ,$$

$$\overline{C} = h + (1-h)\,w\,2S + (1-h)(1-w)S = h + (1-h)(1+w)S$$

and

$$\overline{C^2} = h + (1-h)(1+3w)S^2 \quad .$$

The average thinking time has been calculated as the average value of a geometric random variable in which the probability of success is the probability that a reference is made, i.e., the probability of success is $r$. It is noted that the probability that the think time equal zero cycles is $r$. The average connection time has been calculated from the probability mass function of the connection time. The probability that the connection time will last for one cycle is the probability that the processor made a cache hit, i.e., the probability is $h$. The probability that the connection time will last for $S$ cycles is the probability that the reference made by the processor causes a cache miss and the replaced block has not been modified, i.e., the probability is $(1-h)(1-w)$.

The probability that the connection time will last for $2S$ cycles is the probability that the reference made by the processor causes a cache miss and the replaced block has been modified, i.e., the probability is $w(1-h)$. Furthermore, from this probability mass function the second moment of the connection time can be calculated.

To check the validity of the model in this case, the following system is used as a case study. The system has 16 processors and a shared cache memory distributed between $M$ memory modules. Furthermore, the multiprocessor system has the following parameters:

$$r = 0.4, \quad h = 0.97, \quad w = 0.3 \quad .$$

Table 6.2 shows the simulation results of the performance measures in the cases where $M = 4, 8, 16, or\ 32$ and $S = 4\ or\ 8$. The relative percentage error, $\%Error$, of the performance measures is also shown in Table 6.2. It is clear that the SMI model produces accurate results for these operating points.

| M | S | RESULTS | | |
|---|---|---|---|---|
| | | Measure | Simulation | %Error |
| 4 | 4 | *BW* | 3.4567 | 7.9 |
| | | *PU* | 0.2881 | 7.77 |
| | | *L* | 2.0638 | -11.55 |
| | | *W* | 2.59815 | -15.15 |
| | 8 | *BW* | 3.31986 | 7.98 |
| | | *PU* | 0.2426 | 8.07 |
| | | *L* | 2.3277 | -11.71 |
| | | *W* | 3.5335 | -16.8 |
| | 16 | *BW* | 2.95748 | 8.06 |
| | | *PU* | 0.174 | 8.01 |
| | | *L* | 2.68381 | -8.74 |
| | | *W* | 5.7889 | -15.59 |
| | 32 | *BW* | 2.4643 | 8.6 |
| | | *PU* | 0.1042 | 8.43 |
| | | *L* | 3.02518 | -4.85 |
| | | *W* | 11.326 | -15.75 |
| 16 | 4 | *BW* | 5.9205 | 1.6 |
| | | *PU* | 0.4909 | 2.01 |
| | | *L* | 0.139 | -11.32 |
| | | *W* | 0.424 | -12.92 |
| | 8 | *BW* | 5.6773 | 3.47 |
| | | *PU* | 0.4143 | 3.69 |
| | | *L* | 0.2309 | -11.95 |
| | | *W* | 0.8346 | -14.95 |
| | 16 | *BW* | 5.0942 | 5.61 |
| | | *PU* | 0.3004 | 5.33 |
| | | *L* | 0.3812 | -8.89 |
| | | *W* | 1.9024 | -13.46 |
| | 32 | *BW* | 4.5335 | 3.8 |
| | | *PU* | 0.1901 | 4.67 |
| | | *L* | 0.5266 | -3.73 |
| | | *W* | 4.1456 | -7.78 |

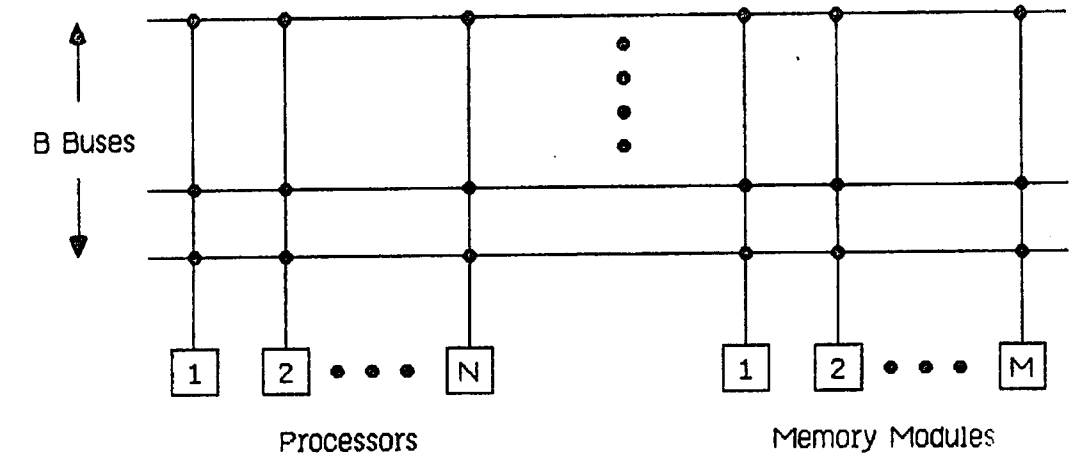
**Table 6.2 The simulation results from the shared cache case study.**

# CHAPTER VII

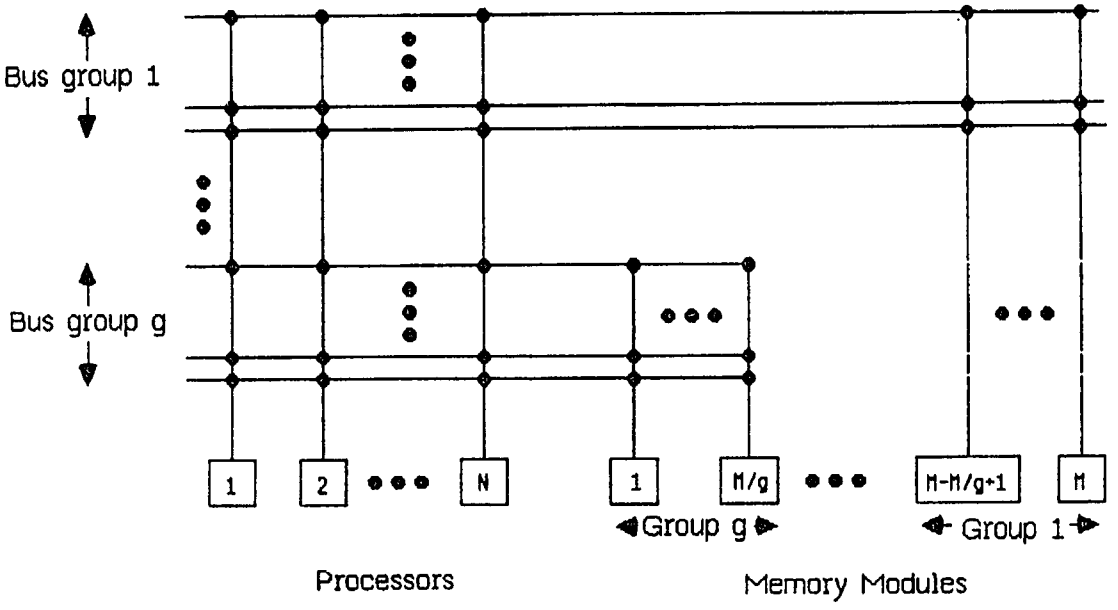
# MODEL EXTENSION TO MULTIPLE-BUS SYSTEM

## 7.1. Introduction

In this dissertation it was assumed that the interconnection network between the processing elements and the memory modules is a crossbar network. However, several interconnection networks were proposed in the literature for the multiprocessor system depicted in Figure 1.2, such as single-bus, multiple-bus, and shuffle-exchange. Of these, the crossbar provides the largest potential bandwidth because there is no conflict within the crossbar. However, the bandwidth obtained is not as high as the potential bandwidth due to the memory interference problem; this fact is obvious after examining the results of the previous two chapters. Furthermore, the crossbar has some inherent problems, such as poor fault tolerance, high cost, and special implementation requirements. Due to these disadvantages the crossbar has not been very attractive in the implementation of multiprocessor systems. A more attractive network is the multiple-bus. Figure 7.1 shows typical multiprocessor systems in which $B$ buses are used to interconnect $N$ processing elements with $M$ memory modules [where $B \leq \min(N, M)$]. The multiprocessor systems of Figure 7.1 are referred to as $N \times M \times B$ systems in this dissertation. The multiple-bus network has a number of desirable features. First, the multiple-bus has a good fault tolerance, for example, if one of the buses has a fault the network will still perform, however, its performance will be degraded. Meanwhile, the crossbar will lose the connection to one of the memory modules if it loses a line. Second, a study reported in [LVA82] demonstrated--using simulation--that the multiple-bus network with $B \approx \min(N, M) / 2$ produces bandwidth similar to those obtained




(a)




(b)

**Figure 7.1 Multiprocessor systems with multiple-bus network.**

using the crossbar network. Third, the multiple-bus network is modular, i.e., it allows easy incremental expansion of the number of the processing elements and the memory modules in the system. Fourth, the buses can be configured in different ways to provide a wide range of trade-offs between bandwidth, connection cost, and reliability.

In this chapter the SMI model is expanded to cover the case where a multiple-bus is used as the interconnection network in the multiprocessor system. In this case a third type of conflict is introduced into the system. This type occurs when a processing element tries to access an idle memory module but all the buses are unavailable. In this situation the processing element will wait until one of the buses becomes available and then it tries again. The study in [LVA82] proposed an arbitration scheme to resolve access conflicts. The scheme is a two-stage arbitration scheme. In the first stage, the conflict due to the memory modules will be resolved by $M$ arbiters of the $N$-users 1-server type; each of these arbiters selects equiprobably one of the processing elements that have outstanding requests to a particular memory module. In the second stage, the conflict due to the buses will be resolved by one arbiter of the $M$-users $B$-servers type; this arbiter will assign the memory requests selected in the first stage to the available buses. The designs of such arbiters are presented in [LaV82]. The arbiter makes the assignment in a cyclic fashion, i.e., on a round-robin basis.

Several performance studies–for a synchronized $N \times M \times B$ system–were reported in the literature, see [LVA82, LVF83, VLL83, MHB84, Bhu84, GoA84, MHB85]. In these studies system operation is approximated by a stochastic process as follows: At the beginning of the system cycle a processing element selects one of the memory modules with equal probability and makes a request to access that module with probability $r$. If more than one request is made to the same memory module, the $N$-users 1-server arbiter will choose one of the processing elements at random. The $M$-users $B$-servers arbiter will select at most $B$ processing elements, from those that were selected in the first stage of arbitration, to be connected with the memory modules. All of the processing elements that fail to be selected by the arbiters will retry in the next cycle. A processing element has at most one pending request waiting for access at any time. The behavior of

the processing elements is considered to be independent but statistically identical. A processing element that obtains a connection to a memory module at the beginning of the system cycle will release that module at the end of the cycle.

One of the early studies to examine the performance of a multiple-bus system was reported in [LVA82]. In this study the simulation approach was used to determine the network bandwidth characteristics of two configurations, complete bus and partial bus. In the complete bus configuration, which is depicted in Figure 7.1(a), every processing element and every memory module is connected to every bus. In the partial bus configuration, which is depicted in Figure 7.1(b), every processing element is connected to every bus, however, each memory module need only be connected to a subset of the buses. The study showed that the complete bus configuration will attain the same bandwidth as the crossbar [with $B \approx \min ( N , M ) / 2$] and it will have a higher fault tolerance than the crossbar. The partial bus configuration will achieve the same bandwidth as the complete bus, however, it will have a lower cost and less fault tolerance. Another study reported in [LVF83] proposed six different configurations of the multiple-bus network. The study performs comparisons between these configurations in terms of their connection cost, arbitration cost, reliability, and expandability. The study demonstrated a wide range of trade-offs between bandwidth, connection cost, and reliability.

Since the behavior of the multiprocessor system with a complete multiple-bus network, as described previously, can be modeled by an intractable analytical model–see the discussion in Section 2.3.1.1 about the exact models–some simplifying assumptions must be adopted by the analytical model in order to simplify the analysis. The models reported in the literature can be divided into two classes according to their simplifying assumptions. The two classes are: the probabilistic models and the rate-adjusted probabilistic models.

The probabilistic models adopt the assumption that the rejected requests will be discarded, see Section 2.3.1.2.1. The derivation of the model is as follows: The term $q$ is defined as the probability that a particular memory module has been requested by some processing element. It is expressed as follows:

$$q = 1 - \left( 1 - r/M \right)^N \quad . \tag{7.1}$$

The probability that $i$ memory modules have been requested is defined as $f(i)$, and it is expressed as follows:

$$f(i) = \binom{M}{i} q^i \left( 1 - q \right)^{M-i} \quad . \tag{7.2}$$

Hence, the bandwidth $BW$ can be expressed as function of $f(i)$s as follows:

$$BW = \sum_{i=1}^{M} \min(i, B) \; f(i) \quad . \tag{7.3}$$

This model was reported in [MHB84]. Furthermore, that study extended this model to cover the case where a partial multiple-bus is used. However, the derivation of equation (7.2) assumes that the events describing the states of the different memory modules are independent events. Since this assumption is not correct, some other models reported in [VLL83, Bhu84, MHB85] avoided such assumptions. These models defined the probability that exactly $i$ memory modules have been requested as $h(i)$. The term $h(i)$ is derived as the probability that at most $i$ memory modules have been requested by the processing elements minus the probability that exactly $j$ of these $i$ modules have been requested, where $0 \leq j < i$. Hence, the term $h(i)$ is expressed as follows:

$$h(i) = \begin{cases} \left( 1 - r \right)^N & i = 0 \\ \binom{M}{i} \left( 1 - r + ir/M \right)^N - \sum_{j=0}^{i-1} \binom{M-j}{i-j} h(j) & i > 0 \end{cases} \tag{7.4}$$

The proof of the equivalence of the three models is reported in [MHB85]. Similar to equation (7.3), the bandwidth $BW$ can be obtained as follows:

$$BW = \sum_{i=1}^{M} \min(i, B) \, h(i) \quad . \tag{7.5}$$

The second class is the rate-adjusted models. These models assume that the rejected request will be submitted again as a new request in the next cycle with probability one, see Section

2.3.1.2.2. The derivation of the model is as follows. The model assumes that the adjusted probability of request is equal to $\alpha$. Hence, the term $\alpha$ will replace the term $r$ in equations (7.2) or (7.4) to obtain $f(i)$ or $h(i)$, respectively. Thereafter, the bandwidth $BW$ can be obtained from equations (7.3) or (7.5). The new value of $\alpha$ can be obtained using the following equation:

$$\alpha_{new} = \frac{BW(\alpha_{old})}{N r^2}(1-r) + 1 \quad . \tag{7.6}$$

Obviously, this is a model that will be solved by iteration techniques; the iteration will stop when $\alpha$ converges to a specific value. The models in [MHB84, GoA84] use $f(i)$ derivations while the model in [MHB85] uses $h(i)$. Moreover, the model in [GoA84] extended this result to cover the case where the connection between the memory module and the processing element will last for a constant number of cycles more than one.

Other models reported in [MaG82, OnI83] assume that the multiprocessor system is asynchronous. They assumed that the request process from any processing element is Poisson process and the connection time between the processing element and the memory module is exponentially distributed. Both of these studies demonstrated that when $B \approx \min(N,M)/2$ the processing power, i.e., the processing element utilization, obtained by using the multiple-bus network will be similar to the value obtained using the crossbar network. Moreover, for lightly utilized systems these asynchronous systems will produce similar results to the synchronized systems. Both of these models are discussed in Section 2.2.1.

In the following section the operation assumption that characterizes the behavior of the multiprocessor system is outlined. In Section 7.3 the modified SMI model for this system is presented. Finally Section 7.4 contains a presentation of some simulation results and the comparisons with the model result.

## 7.2. System Description

The system of interest is a synchronized $N \times M \times B$ multiprocessor system similar to the one depicted in Figure 7.1(a). The multiprocessor system operation is characterized by the

following assumptions:

I. The behavior of the $PE$s can be modeled as identical stochastic processes.

II. The $PE$s think for an integer number of system cycles. The thinking period of any $PE$ is characterized by a discrete independent random variable, $\tilde{T}$.

III. Each $PE$ will submit a memory request after its thinking period; requests originating from the same processing element are independent of each other. The destination of the request originating from any $PE$ will be distributed uniformly between the $M$ modules.

IV. The system uses a two-stage arbitration scheme, the same as the one described in [LVA82]. In the first stage the conflict due to the memory modules (first conflict type) will be resolved by $M$ arbiters of the $N$-users 1-server type. In the second stage the conflict due to the buses (third conflict type) will be resolved by one arbiter of the $M$-users $B$-servers type. The blocked processing elements will try again to the same module in the next cycle.

V. When the second type of memory conflict occurs, i.e., when the memory module is busy when requested by a processing element, the blocked processing element waits until the connection is completed and then resubmits its request to the same memory module.

VI. The connection time between any processing element and any memory module is characterized by a discrete independent random variable, $\tilde{C}$.

As in the previous cases, in order to obtain numerical information from the SMI model, the values of $N$, $M$, $B$, $\overline{T}$, $\overline{C}$, and $\overline{C^2}$ must be obtained through measurements or by hypothesis. These quantities can be regarded as input parameters of the SMI model; knowledge of the full distributions of $\tilde{T}$ and $\tilde{C}$ is not necessary for solving the SMI model. A number of performance measures can be derived from the analytical model. These measures are: the memory bandwidth, $BW$; the $i^{th}$ processing element utilization, $PU_i$; the $j^{th}$ memory module utilization, $MU_j$; the $k^{th}$ bus utilization, $BU_k$; the average queue length of the $j^{th}$ memory module queue, $L_j$; and the average waiting time experienced by the $i^{th}$ processing element in the $j^{th}$ memory module queue, $W_{ij}$. Because of the symmetry between the behavior of the processing elements the subscripts of the performance measures can be omitted. Hence, the performance measures are: $BW$, $PU$, $MU$, $BU$, $L$, and $W$.

## 7.3. The SMI Model for the Multiple-Bus System

As described in the previous chapters the SMI model uses an SMP to approximate the behavior of a $PE$ that functions according to the system operation assumptions outlined in the last section. Hence, $N$ identical SMPs will approximate the behavior of the multiprocessor system. The SMP in this case is depicted in Figure 7.2. The states of the SMP denote the different states of any $PE$. The states can be partitioned to four disjoint subsets similar to the partition of the SMPs in Chapter IV. The first subset is the thinking subset, $S^{th} = \{0\}$. The process enters state 0 and remains there for a duration of time equivalent to the thinking time of the $PE$. The mean sojourn time in this state is $\eta_0$. A memory request is modeled by the SMP leaving state 0. The destination state depends on the state of the requested $MM$ and also on whether the memory request went through the two levels of the arbitration logic or not. The second subset is the accessing subset, $S^{ac} = \{1\}$. The process enters state 1 if the memory module was idle and the

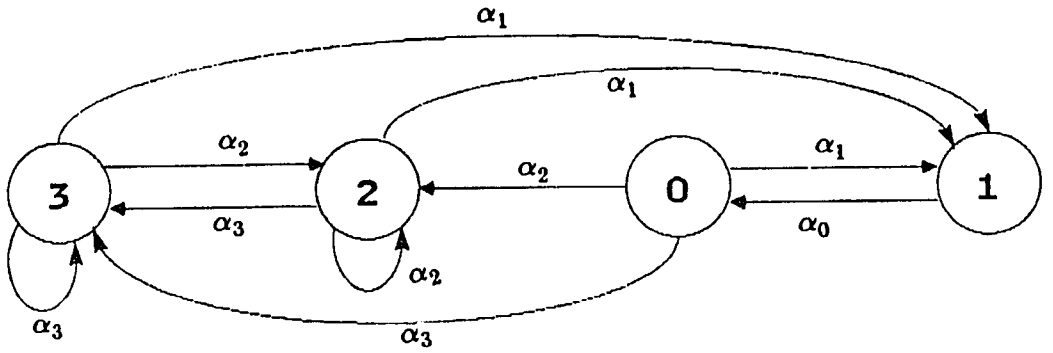



**Figure 7.2 The SMP that describes PE behavior in the multiple-bus system.**

memory request went through the two levels of the arbitration successfully. The process remains in state 1 for a duration equivalent to the connection time between the *PE* and any *MM*, the mean value of the sojourn time in state 1 is $\eta_1$. From state 1 the process returns to state 0, i.e., the *PE* resumes thinking after it has completed its memory access. The third subset is the full waiting subset, $S'^{w} = \{2\}$. The process enters state 2 when the *PE* requests an idle *MM* simultaneously with at least one other request, and of these *PE*s is able to obtain an access to the *MM* by going through the two levels of arbitration. In this case the *PE* must wait for the full duration of the connection time between the *MM* and the selected *PE*, this duration has a mean value of $\eta_2$. The original *PE* will retry to access the same *MM* when the selected *PE* releases the module. If it succeeds, the process enters state 1; if other *PE*s requested the same idle memory module simultaneously and one of these *PE*s obtains the connection with the *MM*, the process reenters state 2; otherwise, the process enters state 3. The fourth subset is the residual waiting subset, $S''^{w} = \{3\}$. The process enters state 3 when the *PE* requests a busy *MM* or, due to bus contention, no *PE* was able to access that particular *MM* even though it is idle. The *PE* must wait for the residual connection time before retrying to access that particular *MM*; the mean value for the sojourn time in the state is $\eta_3$. The process then enters state 1 if the *PE* succeeds in accessing the *MM*, it enters state 2 if the *PE* requests an idle *MM* simultaneously with other *PE*s and one of these *PE*s is able to obtain the connection with that *MM*, or it reenters state 3, otherwise. Clearly, the SMP description does not include which module the *PE* is accessing or which module the *PE* is waiting to access. This does not represent an approximation of the *PE*s behavior because of the symmetry in this case. The underlying approximation of the SMI model is in describing any *PE* behavior independently from the other *PE*s while compensating for the coupling between the *PE*s behaviors in the transition probabilities between the states of the SMP (the coupling results from the *PE*s sharing the *MM*s).

In order to derive numerical information from the SMI model, the values of $N$, $M$, $B$, the first moment of $\tilde{T}$, and the first two moments of $\tilde{C}$ must be obtained through measurement or, if it is considered satisfactory, by hypothesis. These quantities can be regarded as the input parame-

ters to the model. These parameters are defined as follows:

$$N \triangleq \textit{the number of PEs} \ ,$$

$$M \triangleq \textit{the number of MMs} \ ,$$

$$B \triangleq \textit{the number of buses} \ ,$$

$$\overline{T} \triangleq \textit{the first moment of } \tilde{T} \ ,$$

$$\overline{C} \triangleq \textit{the first moment of } \tilde{C}$$

and

$$\overline{C^2} \triangleq \textit{the second moment of } \tilde{C} \ .$$

The average sojourn times of the different states of the SMP can be obtained from the parameters of the model as follows:

$$\eta_j = \begin{cases} \overline{T} & j = 0 \\ \overline{C} & j = 1 \\ \overline{C} & j = 2 \\ \dfrac{\overline{C^2} - \overline{C}}{2(\overline{C} - 1)} & j = 3 \end{cases} \tag{7.7}$$

The average sojourn times in states 0, 1 and 2 arise directly from the definition of these states. The average sojourn time in state 3 is obtained by using Theorem 1 of Chapter IV.

It is convenient to introduce some terms that are similar to the terms introduced in Definitions 1 through 3 in Chapter IV that will be used in formulating the model. These terms are: $R$, *BUSY*, *WIN* 1, and *WIN* 2. The term $R$ is defined as the probability that a *PE* makes a request to access a particular *MM* at the beginning of a system cycle. Therefore, it can be computed as the probability of leaving state 0, 2, or 3 to access a particular *MM*. Therefore, $R$ is defined as follows:

$$R = \frac{1}{M} \left( \lambda_0 + \lambda_2 + \lambda_3 \right) \ . \tag{7.8}$$

The term *BUSY* is defined as the probability that a *PE* finds a particular *MM* busy at the beginning of a cycle (type 2 conflict). In other words, one of the other $(N-1)$ PEs is accessing that

*MM* and is not at the point of releasing it. Hence, *BUSY* is the probability that one of $(N-1)$ PEs is accessing a particular *MM* and the accessing *PE* is not at the point of releasing the *MM*. By definition, the probability that a *PE* is accessing a *MM* is $P_1$. Thus, the probability that it is accessing and will not leave state 1 (release the *MM*) in the next cycle is $P_1 - \lambda_1$. Therefore, *BUSY* can be expressed as,

$$BUSY = \frac{N-1}{M} \left( P_1 - \lambda_1 \right) = \frac{N-1}{M} \left( \bar{C} - 1 \right) \lambda_1 \tag{7.9}$$

The term *WIN* 1 is the probability that the memory request initiated by a *PE* passed the first level of arbitration, i.e., one of the $M$ arbiters of the $N$-users 1-server type selected it. The term *WIN* 1 is derived by the following argument. The probability that a *PE* will not request a particular *MM* is $1 - R$; the probability that none of the $N$ *PE*s request that *MM* is $\left( 1 - R \right)^N$ [see equation (7.1)]; and therefore the probability that a particular *MM* is requested by at least one of the *PE*'s is $\left[ 1 - \left( 1 - R \right)^N \right]$. One of these requests will pass the first level of arbitration, therefore, the probability that a request from any *PE* passes the first level of arbitration, $p$, is given by,

$$p = 1 - \left( 1 - R \right)^N$$

The expected number of *PE*s which requested that *MM* at the beginning of a cycle is $NR$. Therefore, *WIN* 1 can be defined as follows:

$$WIN\,1 = \frac{p}{NR} \tag{7.10}$$

Finally, the term *WIN* 2 is the probability that the memory request initiated by a *PE* will pass the second level of arbitration given that it passed the first level of arbitration, i.e., the arbiter of the $M$-users $B$-server type selected the *PE* after it had been passed by one of the $M$ arbiters of the $N$-users 1-server type. The term *WIN* 2 is calculated by conditioning on the number of free buses. Two quantities need to be calculated in this case. The first, $X(k)$, is the probability that the request will pass the second level of arbitration given there are $k$ free buses and that the request has already passed the first level of arbitration. The quantity $X(k)$ can be

expressed as,

$$X(k) = \sum_{i=1}^{M} \frac{min(k,i)}{i} \binom{M-1}{i-1} p^{i-1} (1-p)^{M-i}$$

The factor $min(k,i) / i$ is the probability that if $i$ requests pass the first level then $min(i,k)$ will obtain buses (pass the second level). The factor $\binom{M-1}{i-1} p^{i-1} (1-p)^{M-i}$ is the probability that $(i-1)$ additional requests pass the first level given that one request has with certainty. The second quantity, $Y(k)$, is the probability that there are $k$ free buses. The quantity $Y(k)$ can be derived as follows. The probability that $k$ out of $B$ buses are free is $\binom{B}{k} b^{B-k} (1-b)^{k}$, where $b$ is the probability that a bus is busy. The term $b$ can be found through an argument similar to that used to derive the term $BUSY$, and can be expressed as $(N-1) \frac{(P_1 - \lambda_1)}{B}$. The term $WIN2$ can now be obtained from,

$$WIN2 = \sum_{k=1}^{B} X(k) Y(k) \tag{7.11}$$

The transition probabilities between the states of the SMP can be derived as the following functions of $BUSY$ and $WIN$:

$$\alpha_j = \begin{cases} 1 & j = 0 \\ (1 - BUSY)\, WIN1\, WIN2 & j = 1 \\ (1 - BUSY)(1 - WIN1)\, WIN2 & j = 2 \\ BUSY + (1 - BUSY)(1 - WIN2) & j = 3 \end{cases} \tag{7.12}$$

The derivation proceeds as follows: When the process, shown in Figure 7.2, leaves any of states 0, 2, or 3 it enters the accessing state (state 1) with probability $\alpha_1$ if the requested $MM$ is idle and the $PE$s request went through the two levels of arbitration; the process enters the full waiting state (state 2) with probability $\alpha_2$ if the requested $MM$ is idle and the $PE$s request fails to be selected in the first level of arbitration and the selected request went through the second level of arbitration; or the process enters the residual waiting state (state 3) with probability $\alpha_3$ if the requested $MM$ is busy or the requested $MM$ is idle but none of the requests manages to obtain an

access with it due to the bus contention. The process always enters the thinking state after it leaves the accessing state ($\alpha_0 = 1$).

The embedded Markov chain can be solved and the $\pi$s can be represented as functions of the transition probabilities, i.e., of *BUSY*, *WIN* 1 and *WIN* 2. The SMP limiting probabilities can be derived by substituting the limiting probabilities of the embedded Markov chain ( $\pi$ s) into equation (4.1). Therefore, the SMP limiting probabilities can be expressed as functions of $R$ and the transition probabilities as shown below:

$$P_j = \begin{cases} \eta_0 \alpha_1 M R & j = 0 \\ \eta_1 \alpha_1 M R & j = 1 \\ \eta_2 \alpha_2 M R & j = 2 \\ \eta_3 \alpha_3 M R & j = 3 \end{cases} \tag{7.13}$$

It can be seen from these equations that a set of nonlinear equations, must be solved. The nonlinearity is introduced because the transition probabilities are defined as functions of the limiting probabilities of the SMP; meanwhile, the limiting probabilities are defined as functions of the transition probabilities. An iterative algorithm can be used to solve these equations. The algorithm will iterate on the values of $R$ and $\lambda_1$. Then the performance measures of the system can be derived. The algorithm breaks down as follows:

1. The average sojourn times of the states are calculated using equation (7.7).
2. An initial value for $R$ is chosen in the range $0 < R < 1$ (in this experiment $R = 1/M$ was used). An initial value for $\lambda_1$ ($\lambda_1 = 0$) is chosen.
3. The terms *BUSY*, *WIN* 1 and *WIN* 2 are calculated using equations (7.9), (7.10) and (7.11), respectively.
4. The transition probabilities are calculated using equation (7.12).
5. A new value for $R$ is calculated by summing the four equations of equation (7.13) into one. Then $R$ can be expressed as follows:

$$R = \frac{1}{\left( \eta_0 \alpha_1 + \eta_1 \alpha_1 + \eta_2 \alpha_2 + \eta_3 \alpha_3 \right)} \quad .$$

6. A new value for $\lambda_1$ is calculated as follows:

$$\lambda_1 = \alpha_1 M R \quad .$$

7. Steps 3 through 6 are repeated until $R$ and $\lambda_1$ have the desired accuracy.[1]

The solution for $R$ may be used to calculate the limiting probabilities of the states using equation (7.13). These can, in turn, be used to calculate the performance measures discussed earlier, as follows:

$$\begin{aligned} BW &= N P_1 \quad , \\ PU &= P_0 + P_1 \quad , \\ MU &= \frac{N}{M} P_1 \quad , \\ BU &= \frac{N}{B} P_1 \quad , \\ L &= \frac{N}{M} \left( P_2 + P_3 \right) \end{aligned}$$

and

$$W = \frac{\eta_2 \alpha_2 + \eta_3 \alpha_3}{\alpha_1} \quad .$$

The last equation is the only one that does not follow directly from the definition of the states of Figure 7.2. It can be derived by calculating the expected value of $\tilde{W}$ in the usual way from the pmf of $\tilde{W}$. The pmf of $\tilde{W}$ can be expressed as follows:

$$Pr \left[ W = (i-j) \eta_2 + j \eta_3 \right] = \binom{i}{j} \alpha_1 \alpha_2^{i-j} \alpha_3^{j} \quad .$$

The derivation of this equation proceeds as follows: The probability that the process moves from state 0 to state 1 after making $(i-j)$ consecutive visits to state 2 followed by $j$ consecutive visits to state 3 is $\alpha_2^{i-j} \alpha_3^{j} \alpha_1$. Since there is $\binom{i}{j}$ combinations of these $i$ visits to states 2 and 3 (i.e., not necessarily consecutive visits), the probability that the process moves from state 0 to state 1 after making $(i-j)$ visits to state 2 and $j$ visits to state 3 is $\binom{i}{j} \alpha_1 \alpha_2^{i-j} \alpha_3^{j}$.

[1] This is a fixed-point iteration scheme. The Steffensen iteration algorithm was used to accelerate the convergence, see [CoB80].

Thereafter, the average value of the waiting time in the queue, $W$, is calculated from the pmf of the waiting time described previously. Therefore, $W$ can be expressed as follows:

$$W = \sum_{i=0}^{\infty} \sum_{j=0}^{i} \binom{i}{j} \alpha_1 \alpha_2^{i-j} \alpha_3^{j} \left[ (i-j)\,\eta_2 + j\,\eta_3 \right]$$
$$= \frac{\eta_2 \alpha_2 + \eta_3 \alpha_3}{\alpha_1} \quad .$$

## 7.4. Simulation Results

In this section the validity of the model presented in the previous section is examined by comparing the results of the model with the simulations. Some hypothetical cases were considered. In all of these cases the multiprocessor system has eight $PE$s and eight $MM$s. In these cases the effect of the input parameters ($B$, $\overline{T}$, $\overline{C}$, and $C_v$) on the performance measures ($BW$, $PU$, $L$, and $W$) were studied. In these cases the relative percentage error, $\%Error$, between the results of the model and the results of the simulation is calculated as follows:

$$\%Error = \frac{result\ from\ the\ model - result\ from\ the\ simulation}{result\ from\ the\ simulation} \times 100 \quad .$$

The input parameters are altered in these cases in order to see the effect of these alterations on the different performance measures. It is noted that the memory utilization and the bus utilization are not reported since they can be calculated easily from the memory bandwidth.

In the first case the connection time between a $PE$ and an $MM$ lasts for one cycle and the average think time is zero cycles. In this case $C_v = 0$ since the connection time is deterministic. Figure 7.3 shows the simulation results of $BW$, $PU$, $L$, and $W$ as functions of $B$. Clearly, when $B$ is low ($< 5$) the number of buses will be the bottleneck of the performance of the system. Hence, the performance will be degraded because of the third type of memory conflict. However, for higher values of $B$ ($\geq 5$) the multiple-bus connection network will look like a crossbar network due to the first two types of memory conflicts. Therefore, increasing the value of $B$ beyond some point ($\approx 5$) will not affect the performance. Figure 7.4 shows the relative percentage error between the results of the simulation and the results of the model. It can be seen from Figure 7.4

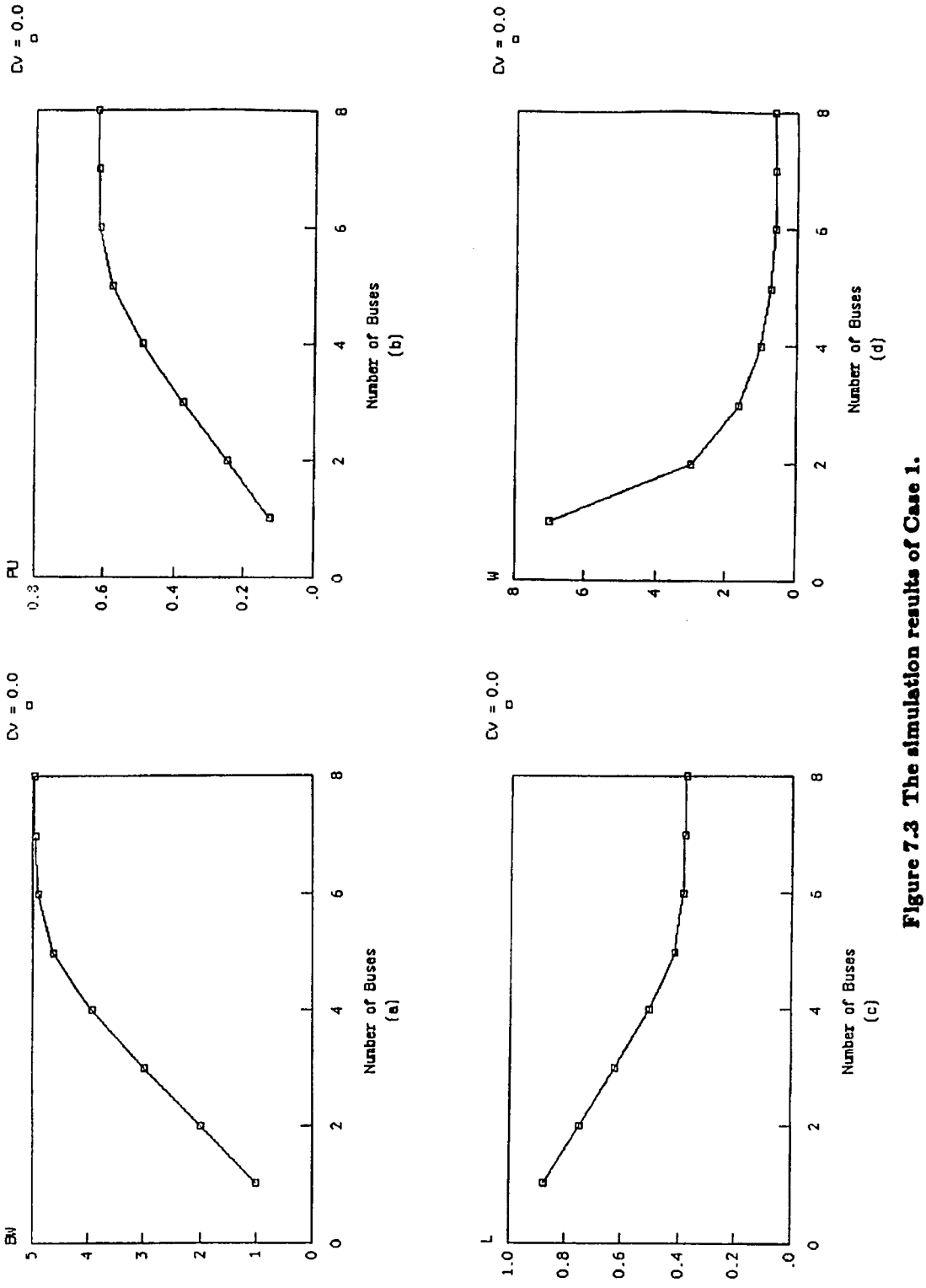



**Figure 7.3 The simulation results of Case 1.**

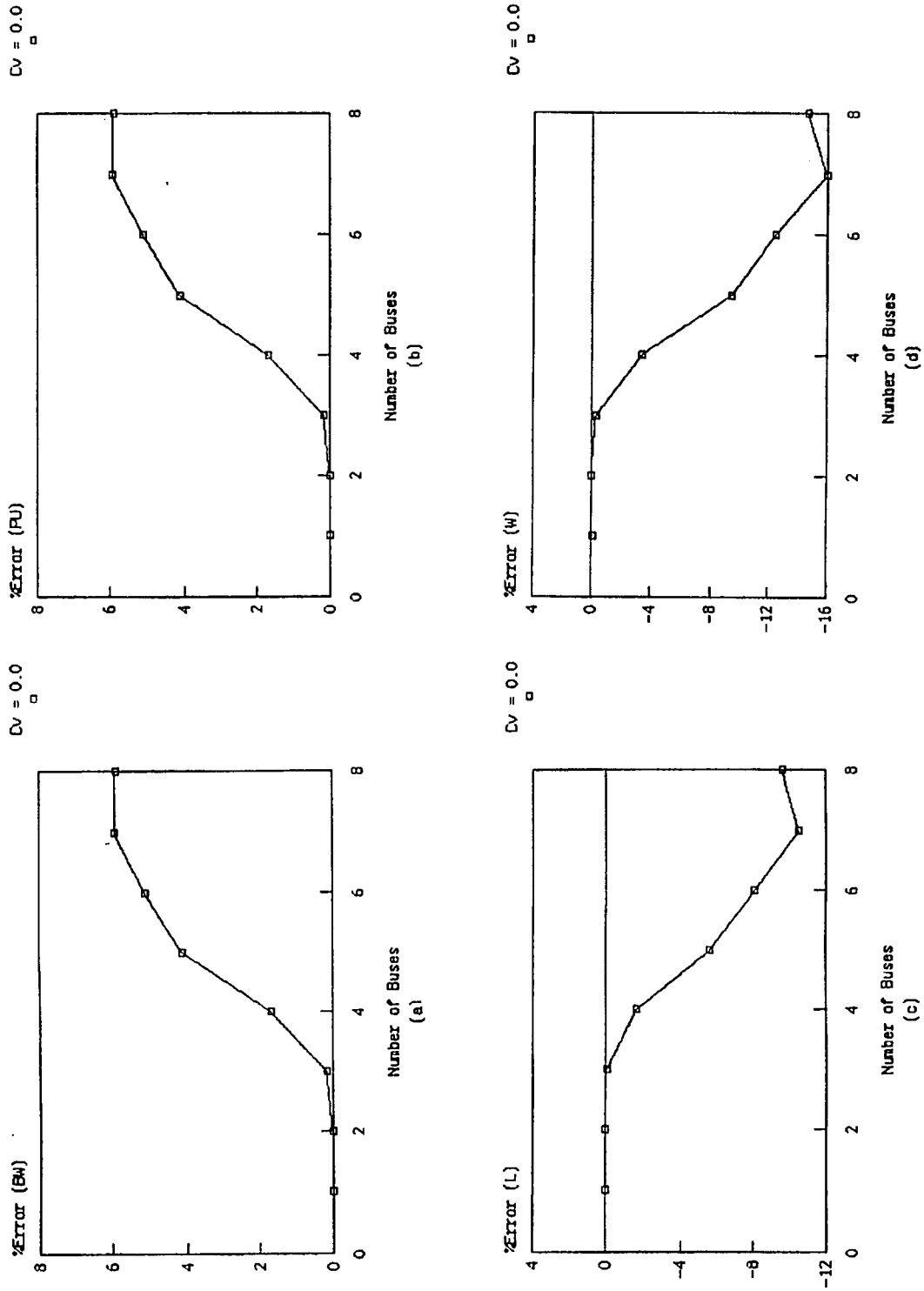



**Figure 7.4 The relative percentage error of Case 1.**

that the results of the model are in close agreement with the results of the simulation. The utilization measures ($BW$ and $PU$) are within approximately 7 percent of the simulations. The queue measures ($L$ and $W$) are within 15 percent of the simulations.

The second case has the same operating conditions as the first case except that the average think time for a $PE$ in this case is one cycle. Figure 7.5 shows the simulation's results of $BW$, $PU$, $L$, and $W$ as functions of $B$. Similar to the previous case, the number of buses is the bottleneck of the system if $B$ is small ($B < 5$). The multiple-bus network will act as a crossbar network when the number of buses is high ($B \geq 5$). Comparing Figures 7.3 and 7.5 shown the effect of $\overline{T}$ on the behavior of the system. The memory bandwidth $BW$, when $B \leq 4$, will not be affected by $\overline{T}$; however, $BW$ will decrease as $\overline{T}$ increases when $B > 4$. The reason for this is that at low values of $B$ the multiple-bus is the bottleneck to the system and the bandwidth cannot exceed the number of buses in the system. Nevertheless, if the think time is extremely large a multiple-bus network with $B = 1$ will act as a crossbar network. The other measures of performance $PU$, $L$, and $W$ will be affected by $\overline{T}$ in the same way as in the crossbar case, see Section 5.1. The processor utilization $PU$ increase as $\overline{T}$ increases. The queue measures $L$ and $W$ decreases as $\overline{T}$ increases. Figure 7.6 shows the relative percentage error between the results of the simulation and the results of the model. The accuracy of the model in this case is similar to its accuracy in the previous case.

In the third case the average connection time between a $PE$ and an $MM$ is four cycles and the average think time for a $PE$ is zero cycles. The constant of variation of the connection time, $C_v$, varies from zero to two in order to examine the effect of the connection variation on the performance of the system. Figure 7.7 shows the results of the simulation of $BW$, $PU$, $L$, and $W$ as functions of $B$ and $C_v$. Similar to the previous cases, the limitation of the multiple-bus network is clear when $B$ is low ($B < 5$). For higher values of $B$ the multiple-bus network acts as a crossbar. The variation in the connection time affects the performance of the multiprocessor system depicted in Figure 7.1(a), particularly when $B$ is high, i.e., when the multiple-bus network acts as a crossbar network. As $C_v$ increases the utilization measures ($BW$ and $PU$) decrease and

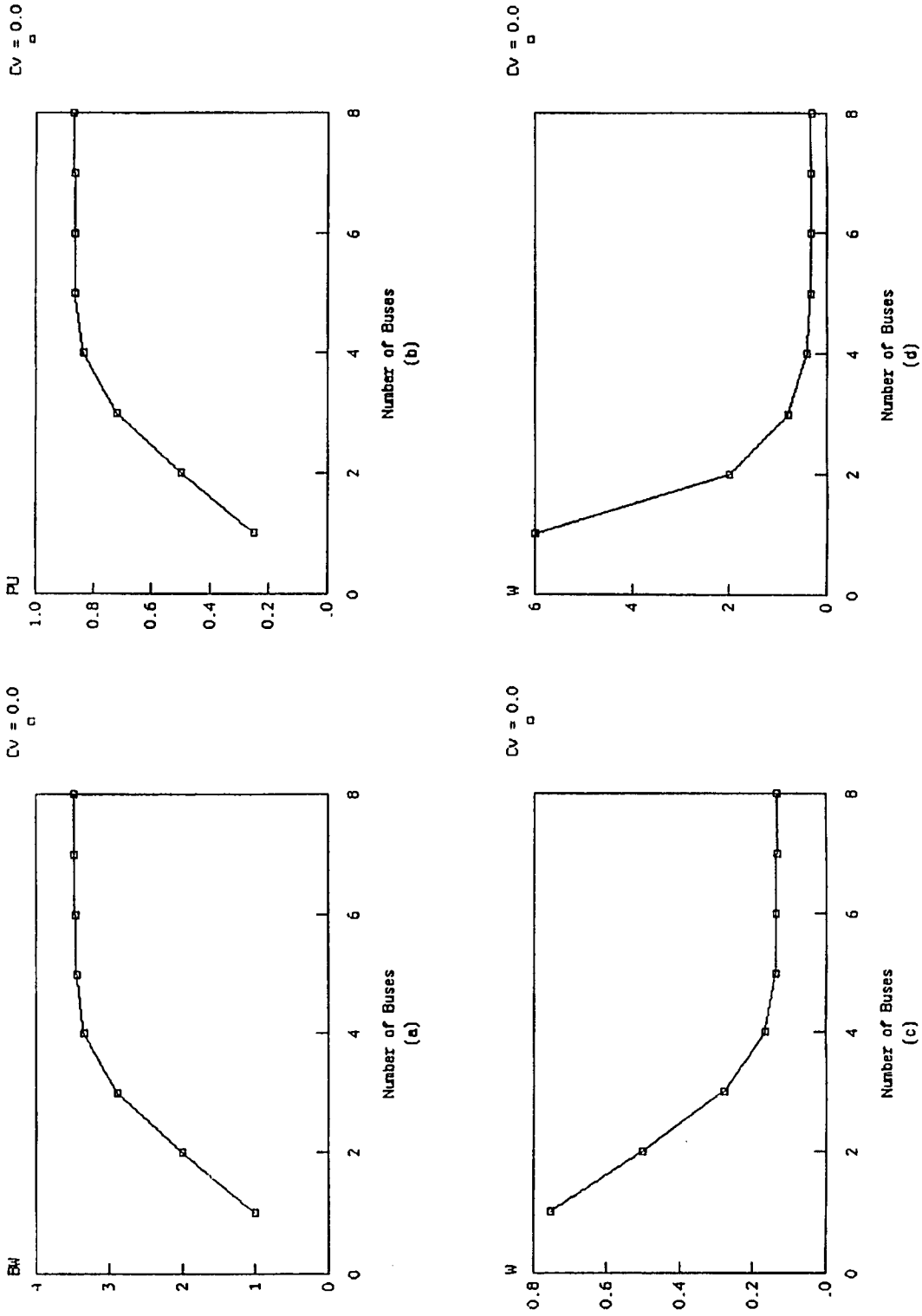



**Figure 7.5 The simulation results of Case 2.**

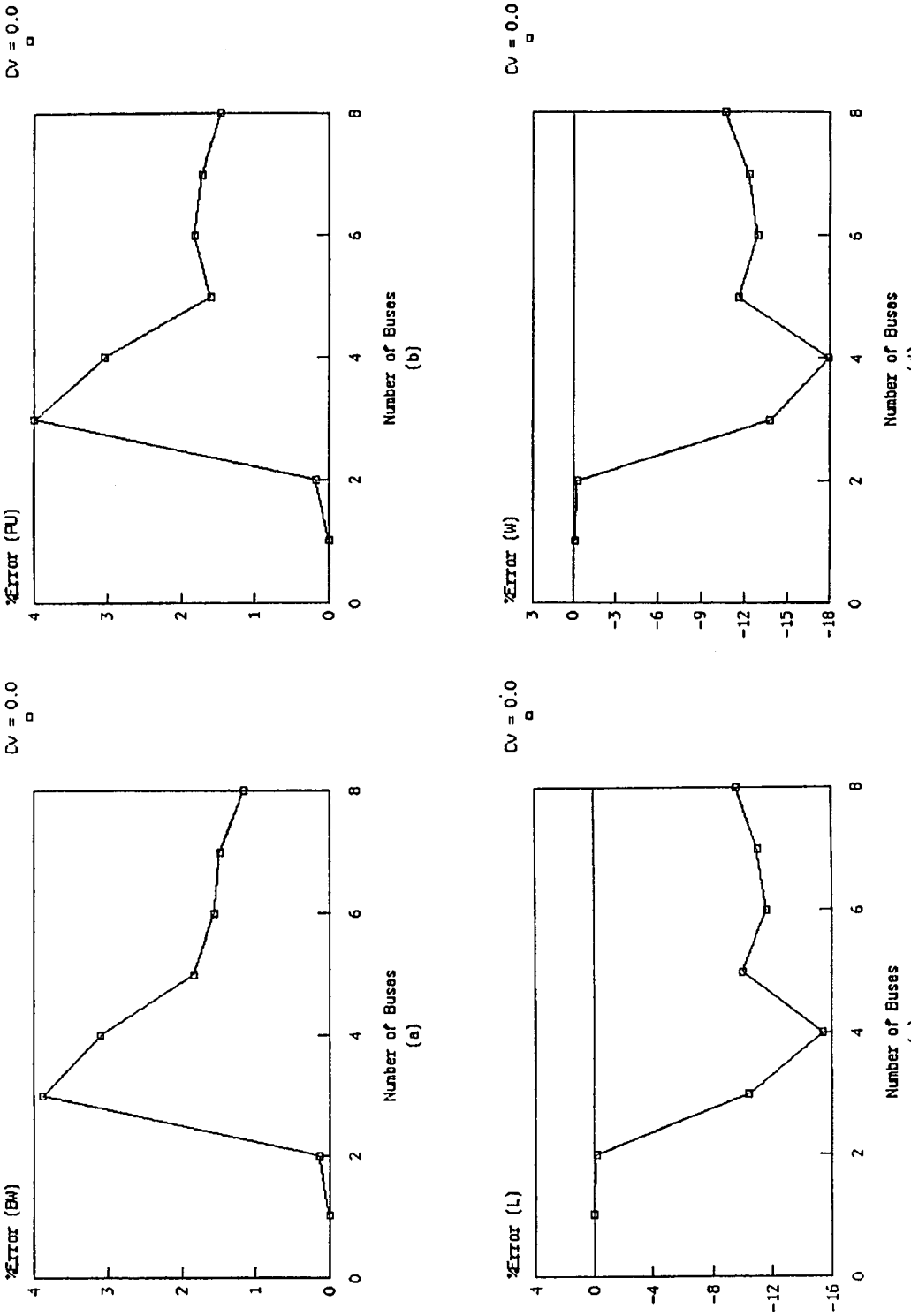



**Figure 7.6 The relative percentage error of Case 2.**

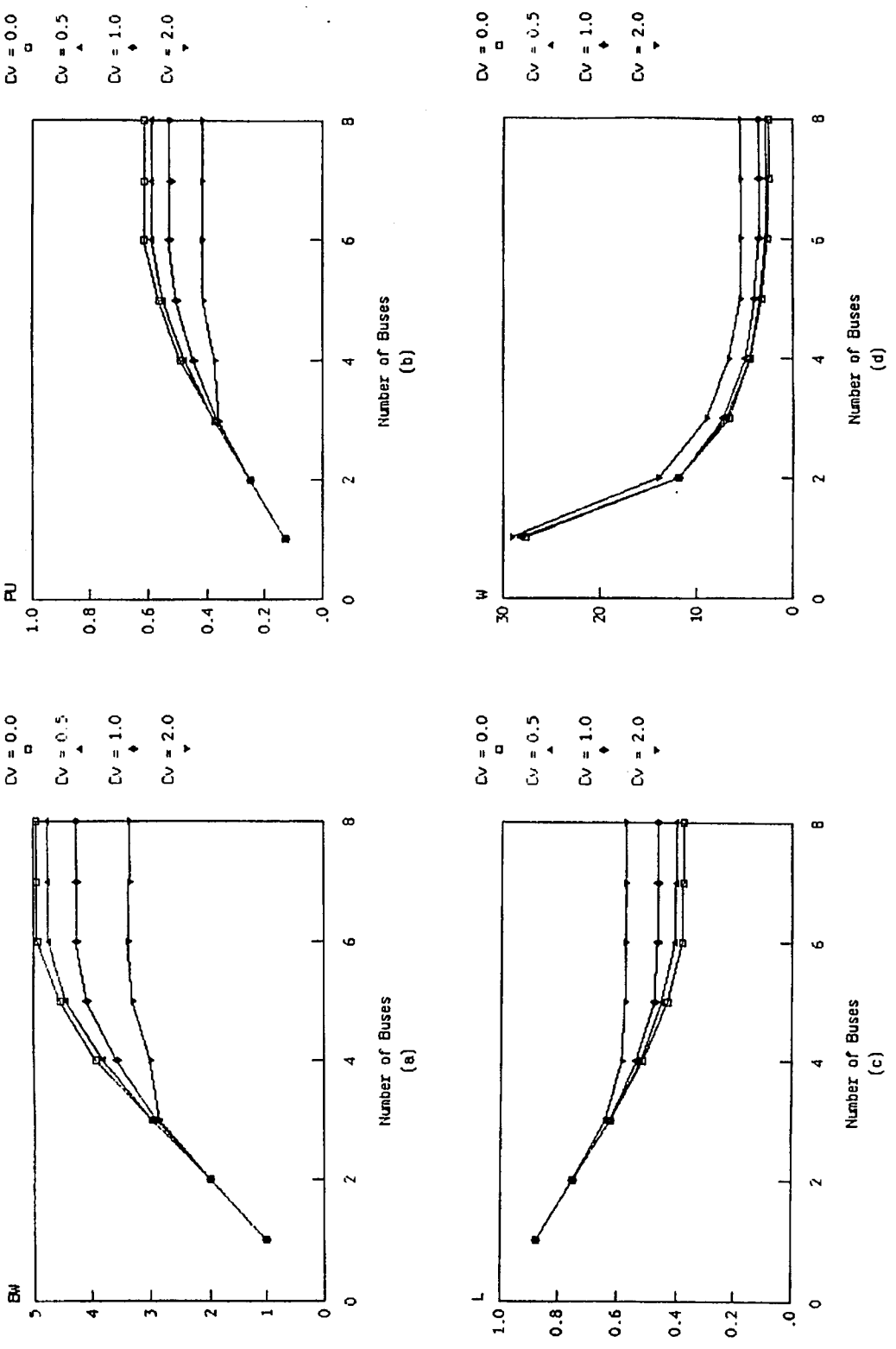



**Figure 7.7 The simulation results of Case 3.**

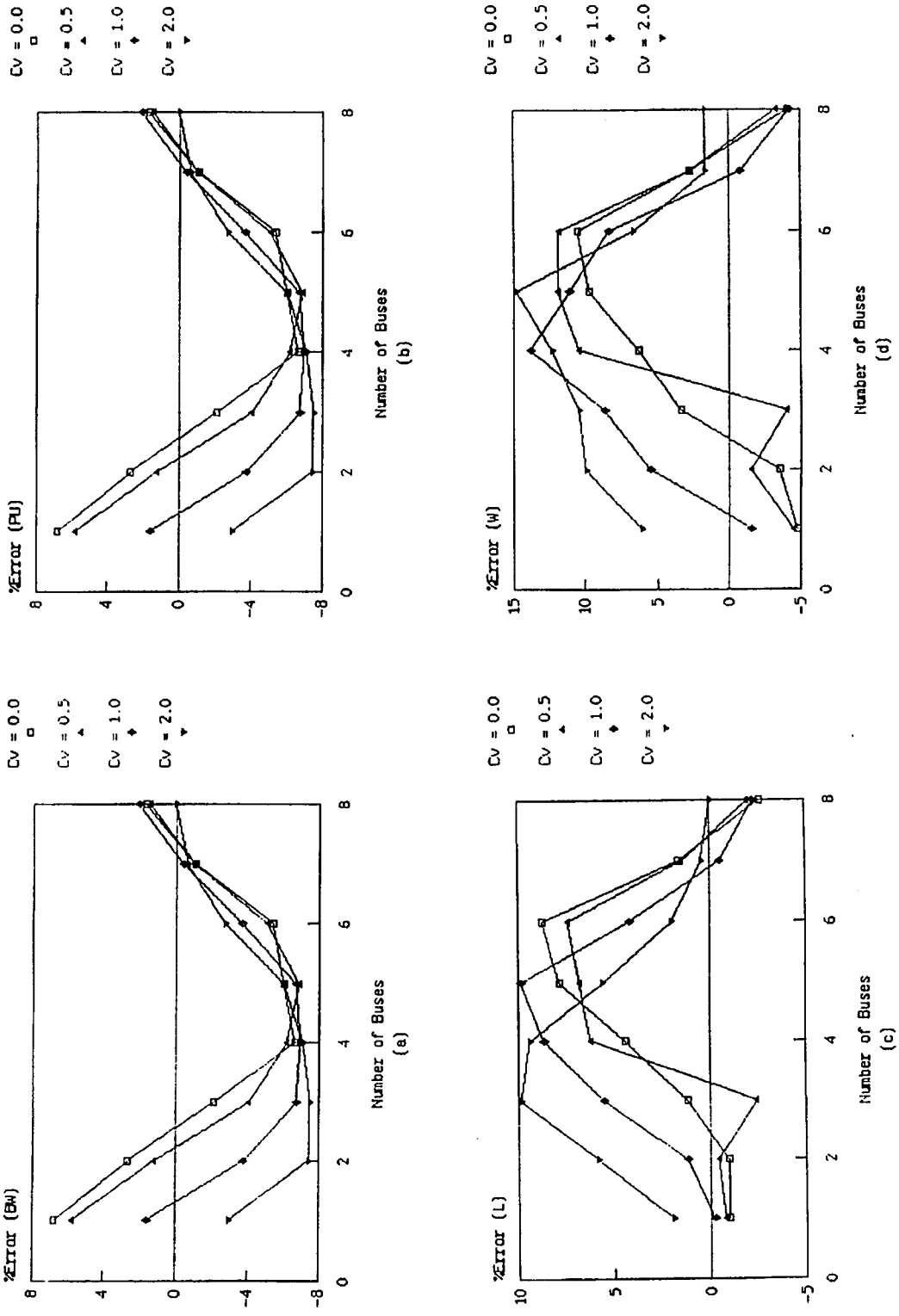



**Figure 7.8 The relative percentage error of Case 3.**

the queue measures ($L$ and $W$) increase. This is the same behavior that was observed when the crossbar network was used, see Section 5.1. Figure 7.8 shows the relative percentage error between the results of the simulation and the results of the model. The utilization measures ($BW$ and $PU$) are within 8 percent of simulation. The queue measures ($L$ and $W$) are within 15 percent of simulation.

The fourth case has the same operating conditions of the third case except that the thinking time for a $PE$ in this case is one cycle. Figure 7.9 shows the results of the simulation of $BW$, $PU$, $L$, and $W$ as functions of $B$ and $C_v$. Similar to the previous case, the number of buses is the bottleneck of the system if $B$ is small ($B < 5$). The multiple-bus network will act as a crossbar network when the number of buses is high ($B \geq 5$). Comparing Figures 7.7 and 7.9 shows the effect of $\overline{T}$ on the behavior of the system. The memory bandwidth $BW$, when $B \leq 4$, will not be affected by $\overline{T}$; however, $BW$ will decrease as $\overline{T}$ increases when $B > 4$. The reason for this is that at low values of $B$ the multiple-bus is the bottleneck to the system and the bandwidth cannot exceed the number of buses in the system. The other measures of performance $PU$, $L$, and $W$ will be affected by $\overline{T}$ in the same way as in the crossbar case, see Section 5.1. The processor utilization $PU$ increases as $\overline{T}$ increases. The queue measures $L$ and $W$ decrease as $\overline{T}$ increases. The variation in the connection time affect the performance of the system in the same way as the previous case. Figure 7.10 shows the relative percentage error between the results of the simulation and the results of the model. The accuracy of the model in this case is similar to its accuracy in the previous case.

The last two cases highlight the importance of keeping $C_v$ low. Reducing $C_v$ from 2 to 0 can increase the $BW$ by about 65% (Fig. 7.7(a)) and more than halve $W$. In systems where some $PE$s may be DMA channels that can perform block transfers and other $PE$s may be simply be transferring cache lines it may be advantageous to break up the block transfers and/or increase the cache line size so that $C_v$ is reduced.

In the fifth case the average connection time between a $PE$ and an $MM$ is eight cycles and the average think time for a $PE$ is zero cycles. The constant of variation of the connection time,

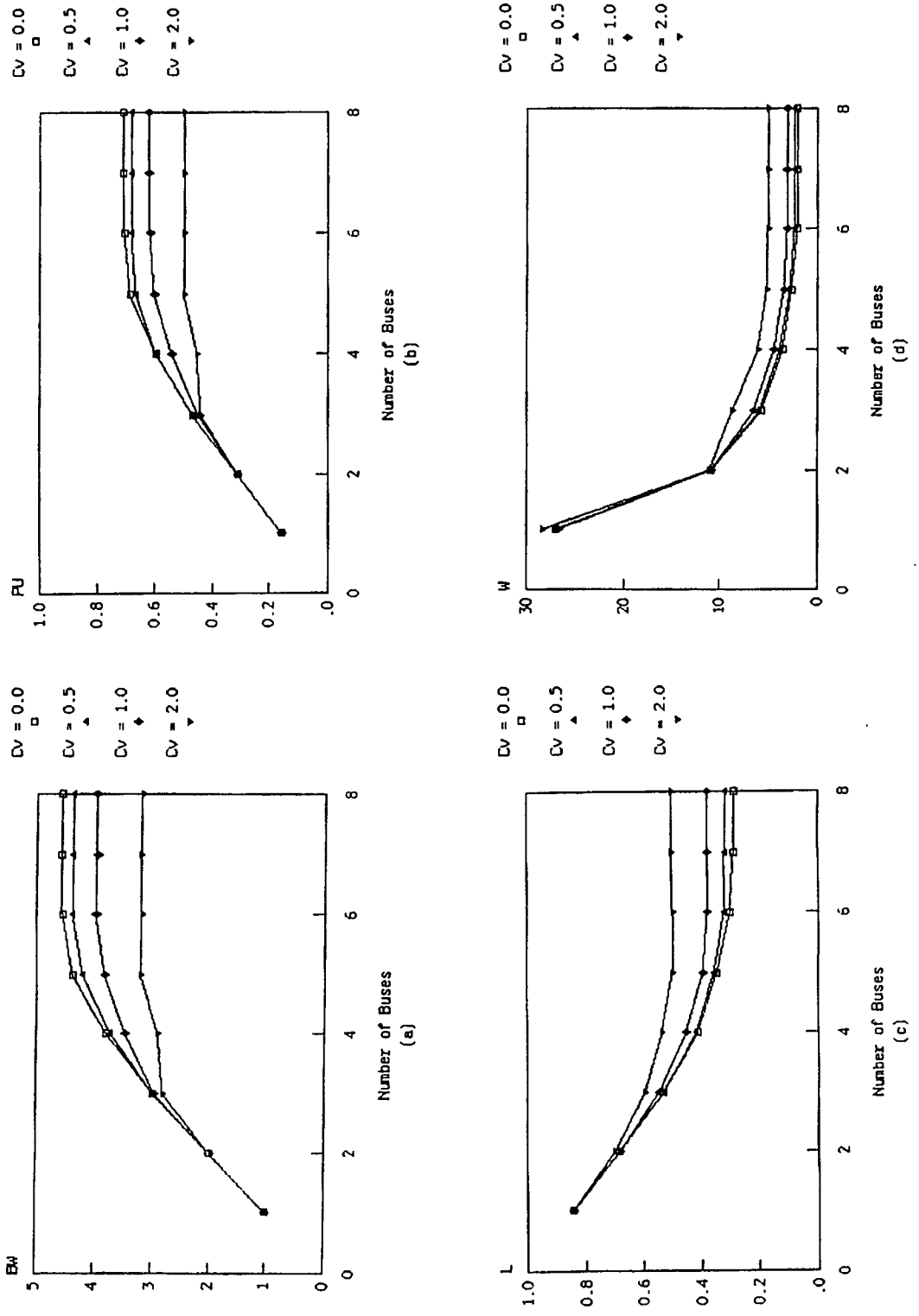



**Figure 7.9 The simulation results of Case 4.**

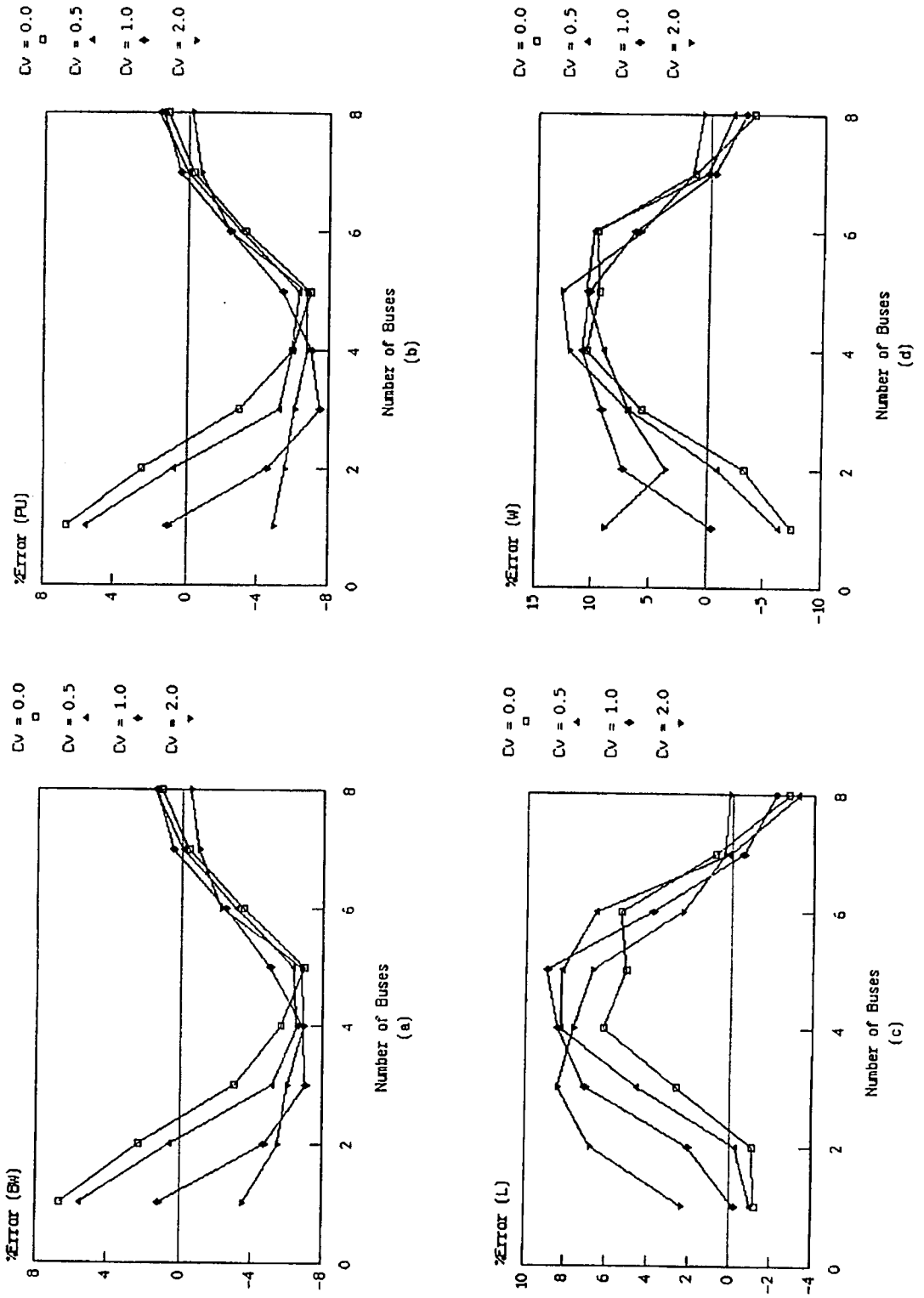



**Figure 7.10 The relative percentage error of Case 4.**

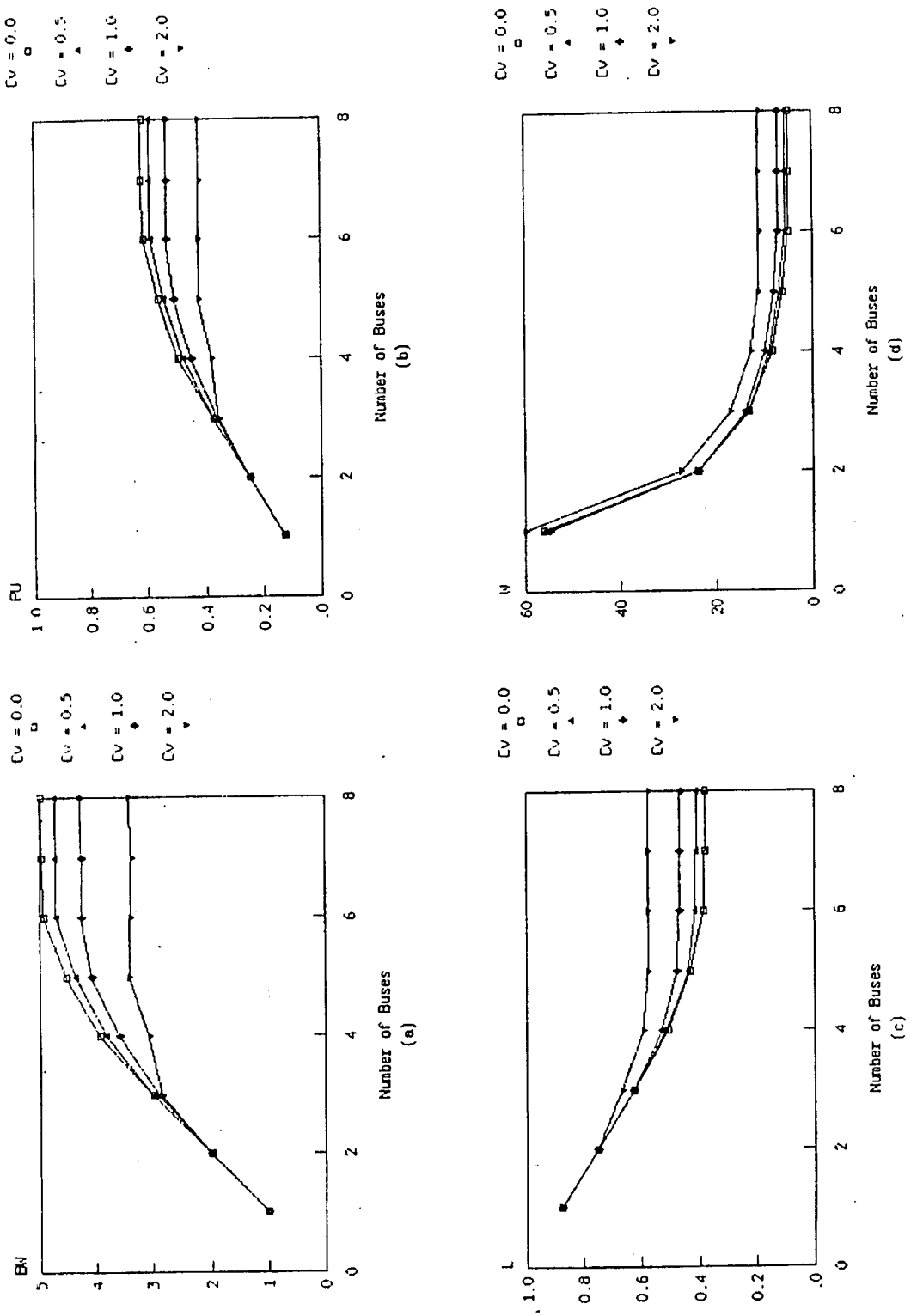



**Figure 7.11 The simulation results of Case 5.**

$C_v$, varies from zero to two in order to examine the effect of the connection variation on the performance of the system. Figure 7.11 shows the results of the simulation of $BW$, $PU$, $L$, and $W$ as functions of $B$ and $C_v$. Similar to the previous cases, the limitation of the multiple-bus network is clear when $B$ is low ($B < 5$). For higher values of $B$ the multiple-bus network acts as a crossbar. The variation in the connection time affects the performance of the multiprocessor system depicted in Figure 7.1(a), particularly when $B$ is high, in the same way as in the previous two cases. Comparing Figures 7.3, 7.7 and 7.11 shows the effect of varying the average connection time, $\overline{C}$, on the performance of the system. It can be seen that increasing $\overline{C}$ only yields an increase in the average waiting time $W$ only, while the other measures remain almost constant in spite of varying $\overline{C}$. Figure 7.12 shows the relative percentage error between the results of the simulation and the results of the model. The utilization measures ($BW$ and $PU$) are within 8 percent of simulation. The queue measures ($L$ and $W$) are within 15 percent of simulation.

The sixth case has the same operating conditions of the fifth case except that the thinking time for a $PE$ in this case is one cycle. Figure 7.13 shows the results of the simulation of $BW$, $PU$, $L$, and $W$ as functions of $B$ and $C_v$. Similar to the previous case, the number of buses is the bottleneck of the system if $B$ is small ($B < 5$). The multiple-bus network will act as a crossbar network when the number of buses is high ($B \geq 5$). Comparing Figures 7.11 and 7.13 shows the effect of $\overline{T}$ on the behavior of the system. The effect of $\overline{T}$ is similar to the previous cases. The variation in the connection time affects the performance of the system in the same way as the previous case. Figure 7.14 shows the relative percentage error between the results of the simulation and the results of the model. The accuracy of the model in this case is similar to its accuracy in the previous case.

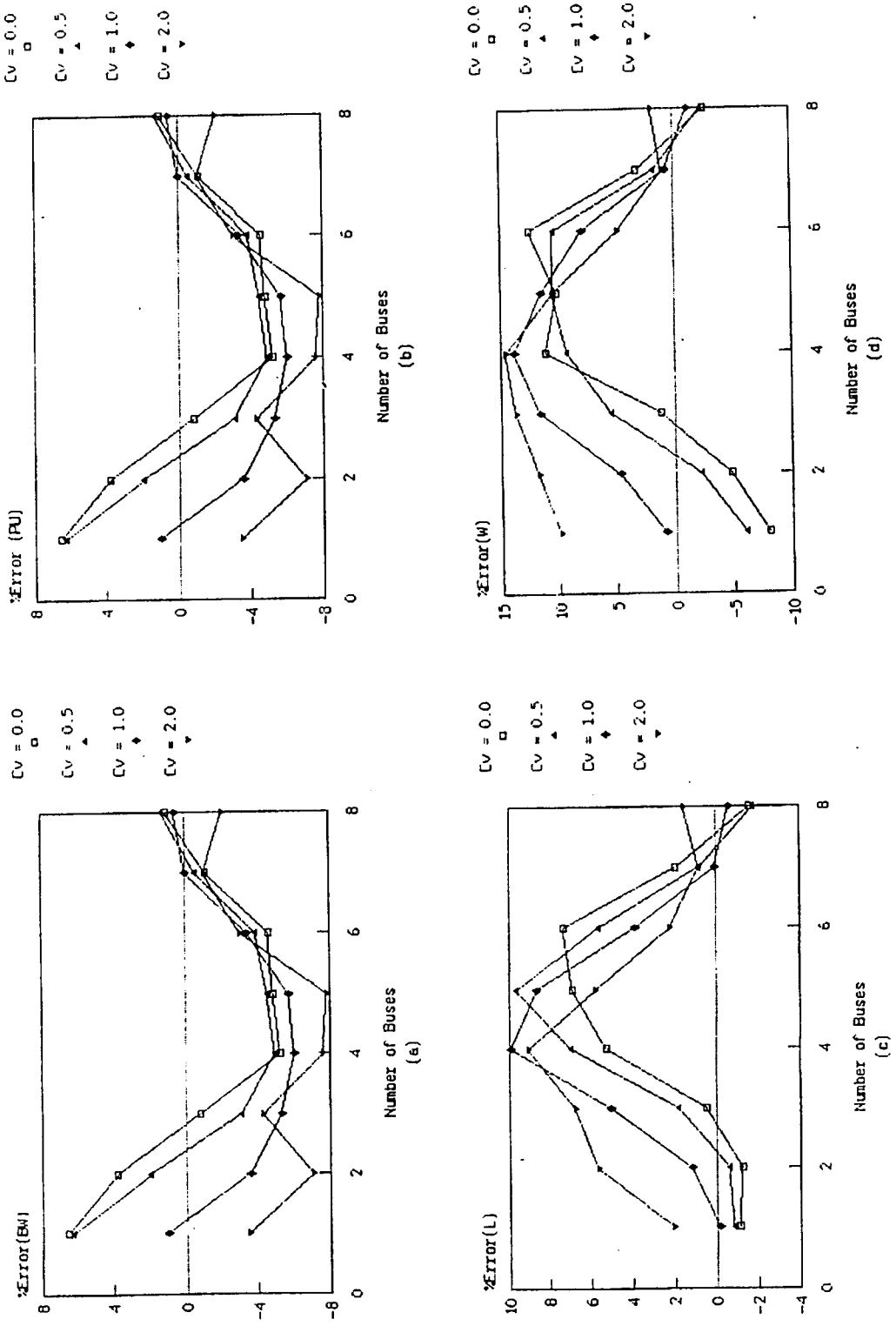



**Figure 7.12 The relative percentage error of Case 5.**

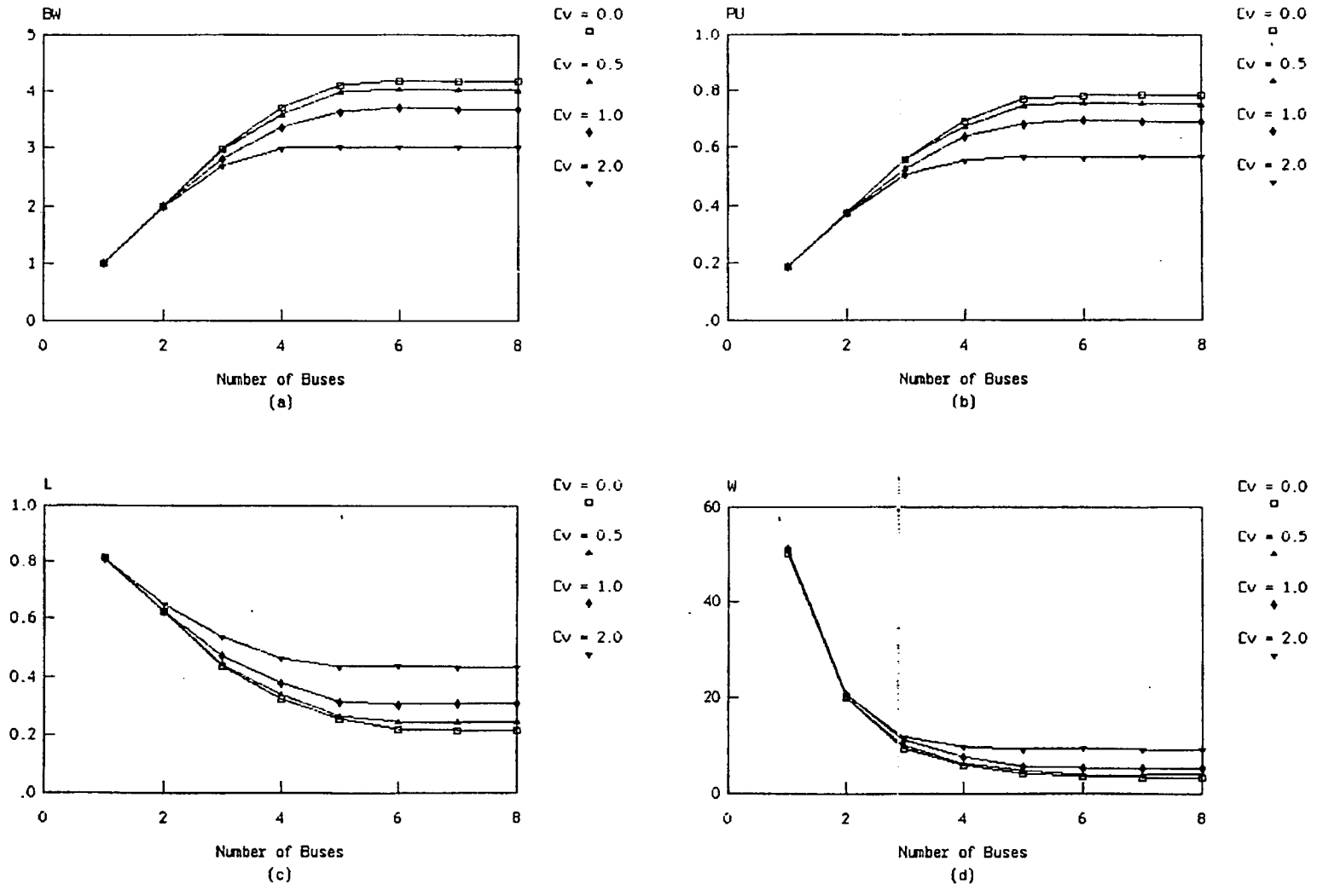



**Figure 7.13 The simulation results of Case 6.**

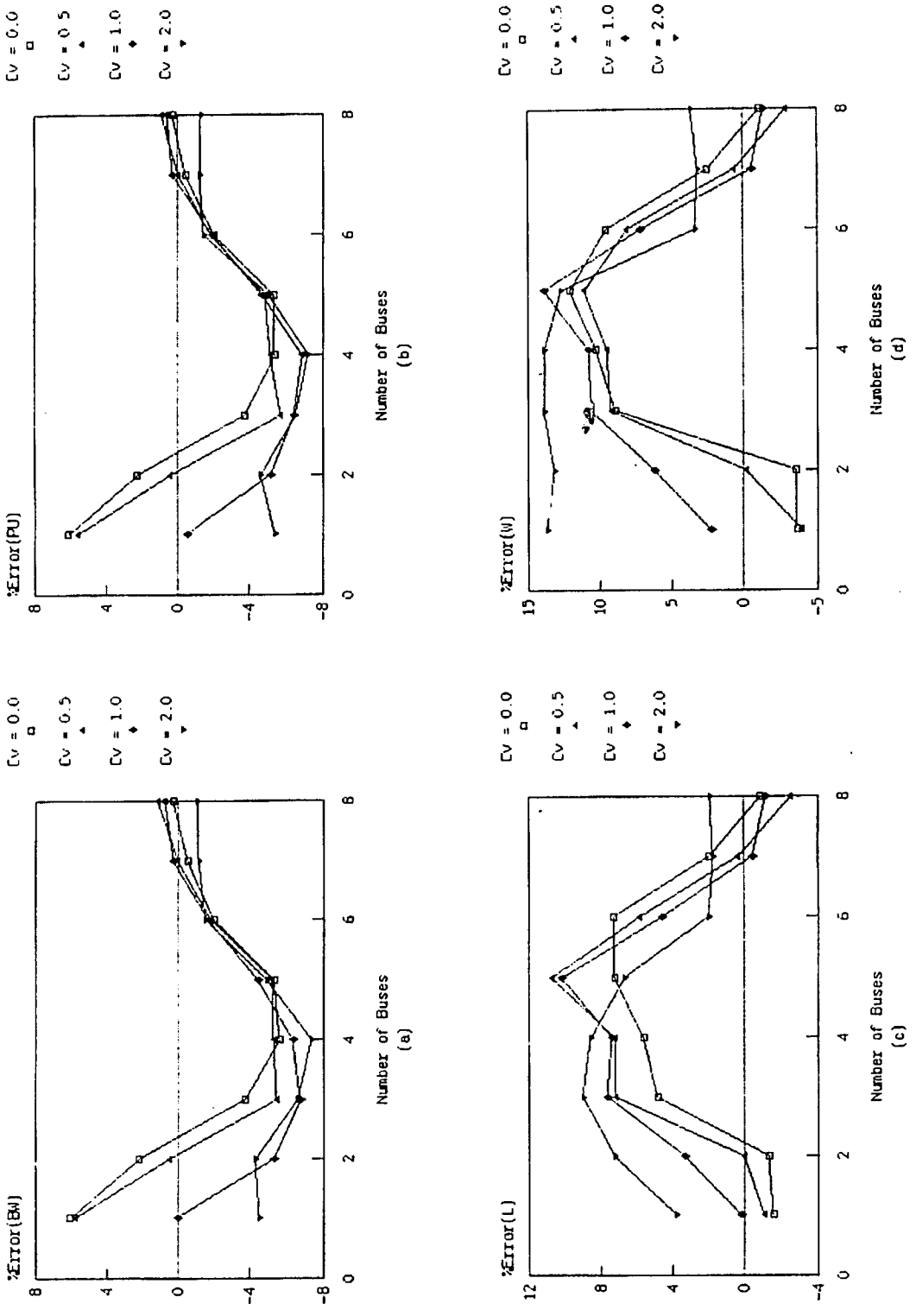



**Figure 7.14 The relative percentage error of Case 6.**

# CHAPTER VIII

# CONCLUSIONS AND FUTURE RESEARCH

## 8.1. Summary and Conclusions

The increasing complexity of designing multiprocessor systems makes increasingly attractive the design of analytical models that will approximate the behavior of these multiprocessor systems. These models will be used in tuning and modifying the multiprocessor system design. In this dissertation a semi-Markov memory interference (SMI) model is introduced that can be used by system designers in the design and the evaluation phases of multiprocessor systems similar to the system shown in Figure 1.2. The interconnection network for these systems is a crossbar or multiple-bus network. The SMI model will provide designers with quantitative measures of the multiprocessor system performance. Furthermore, it provides the designer with a better understanding of the quantitative relationships between the system parameters.

The SMI model is based on a widely accepted set of operation assumptions that characterize multiprocessor behavior as a stochastic process. These assumptions were justified by comparing the results of the stochastic process with the results obtained from trace-driven simulations of real programs. The SMI model explicitly describes the behavior of each processing element by means of a semi-Markov process. Three special cases of the operation assumptions were investigated. These cases were: the uniform case, the mailbox case, and the favorite module case. The examination of these cases shows that the number of states in the semi-Markov process describing a processing element is dependent only on the probability mass function describing the destination of the memory requests.

The SMI model has been tested over a broad range of hypothetical cases. It produced "acceptable" results when compared with the simulation results. The utilization measures were predicted within approximately 8 percent of the simulation results. However, the queue measures were predicted within approximately 20 percent of the simulation results. Actually, the utilization measures were found to be robust quantities with respect to the queue measures. Nevertheless, the queue measures were not reported in most of the memory interference models which were reported in the literature. Among the present available memory interference models, the SMI model will produce the "best" predictions for the average queueing time and the average queue length.

To study more realistic cases, multiprocessor systems with cache memories were examined using the SMI model. Two types of systems were considered: a multiprocessor system with private caches and a multiprocessor system with a shared cache. These two systems were discussed thoroughly with their inherent problems and the proposed solutions. The employment of cache coherency checks by the system yields a variable connection time between the processing element and the memory module. Therefore, the SMI model produced better results than the models proposed in the literature simply because it considers the second moment of the connection time in its calculations. These cache examples illustrated the need for memory interference models that allow variations in the connection time between the processing elements and the memory modules.

Finally, the SMI was extended to cover the case where a multiple-bus network was used instead of a crossbar network in the multiprocessor system. In this case, with a small number of buses in the system $[B < \min(N,M) / 2]$ the memory conflicts due to bus contentions will degrade the performance of the system. However, with a higher number of buses $[B \geq \min(N,M) / 2]$ the conflicts due to bus contentions will have minimal effects on the performance of the system. These conclusions were reported in other studies. Nevertheless, the SMI model is the only model to show the effect of the variation in the connection time on the performance of the system.

The contributions of this dissertation can be summarized as follows:

- A taxonomy of the memory interference models was proposed.
- The SMI model, which is a memory interference model that permits variations in the connection time between the processing element and the memory module, was introduced.
- The effect of the second moment of the connection time on the performance of the multiprocessor system was illustrated.
- The limited applicability of the models that use only the first moment of the connection time in their calculations was demonstrated.
- An analytical model was introduced to study a multiprocessor system with cache memories that employ software checks for cache coherency.
- An analytical model was introduced to study a multiprocessor system with a multiple-bus network in which the connection time between a processing element and a memory module is variable.

## 8.2. Extensions and Future Research

A number of extensions to this research were suggested throughout this dissertation. In this section, they will be summarized and explained. One of the most obvious extensions is the enhancement of the SMI model, particularly in its calculations of the queue measures. One possible technique is to introduce virtual parameters (fudge factors) that will force the results of the model to have better agreement with the results of the simulation.

Another natural extension of the SMI model is to determine the effect of higher moments of the connection time and think time on the performance of the system. If these moments have a

noticeable effect on the performance, then they should be included in the calculations of the SMI model.

The SMI model can be modified to include the set-up time of the interconnection network in its parameters. This component has been ignored in most of the memory interference models. Actually, they assume that it is part of the connection time between the processing element and the memory module.

The SMI model can be modified to cover the multiprocessor systems that employ other types of networks, such as the delta network or the single shuffle-exchange network. In this case the development of the model is similar to the development of the SMI model. However, the conflicts over the resources are slightly different.

Another extension to this research is to study other types of arbitration to resolve memory conflicts. The random arbitration that was assumed in this study has a number of disadvantages, e.g., the time to resolve the conflict is large and the cost of the arbiters could be high for large systems. Two possible solutions are to have arbiters that have some priority ordering to resolve conflicts or to have the arbiters adopt an ethernet-type philosophy.

The reliability of the multiprocessor system can be studied using a modified SMI model, particularly in the case when the multiple-bus network was used. One of the major advantages of the multiple-bus network over the crossbar network is its fault tolerance. The SMI model can be modified by introducing other operation assumptions in order to study the reliability of the multiprocessor system.

# BIBLIOGRAPHY

# BIBLIOGRAPHY


[BaS76] F. Baskett and A. J. Smith, "Interference in Multiprocessor Computer Systems with Interleaved Memory," *Comm. of ACM*, vol. 19, no. 6, pp. 327-334, June 1976.

[BhF73] D. P. Bhandarkar and S. H. Fuller, "Markov Chain Models for Analyzing Memory Interference in Multiprocessor Computer Systems," in *Proc. 1st Ann. Symp. Computer Architecture*, pp. 1-6, December 1973.

[Bha75] D. P. Bhandarkar, "Analysis of Memory Interference in Multiprocessors ," *IEEE Trans. on Computers*, vol. C-24, no. 9, pp. 897-908, September 1975.

[BhL83] L. N. Bhuyan and C. W. Lee, "An Interference Analysis of Interconnection Networks," in *Proc. 1983 Int'l Conf. on Parallel Processing*, pp. 2-9, August 1983.

[Bhu84] L. N. Bhuyan, "A Combinatorial Analysis of Multibus Multiprocessors," in *Proc. 1984 Int'l Conf. on Parallel Processing*, pp. 225-227, August 1984.

[BrD77] F. A. Briggs and E. S. Davidson, "Organization of Semiconductor Memories for Parallel-Pipelined Processors," *IEEE Trans. on Computers*, vol. C-26, no. 2, pp. 162-169, February 1977.

[BrD81a] F. A. Briggs and M. Dubois, "Cache Effectiveness in Multiprocessor Systems with Pipelined Parallel Memories," in *Proc. 1981 Int. Conf. Parallel Processing*, pp. 306-313, August 1981.

[BrD81b] F. A. Briggs and M. Dubois, "Performance of Cache-based Multiprocessors," in *ACM Conf. Measurement Modeling Comput. System*, September 1981.

[BrD83] F. A. Briggs and M. Dubois, "Effectiveness of Private Caches in Multiprocessor Systems with Parallel-Pipelined Memories," *IEEE Trans. on Computers*, vol. C-32, no. 1, pp. 48-59, January 1983.

[CeF78] L. M. Censier and P. Feautrier, "A New Solution to Coherence," *IEEE Trans. on Computers*, vol. C-27, no. 12, pp. 1112-1118, December 1978.

[CKL77] D. Y. Chang, D. J. Kuck, and D. H. Lawrie, "On the Effective Bandwidth of Parallel Memories," *IEEE Trans. on Computers*, vol. C-26, no. 5, pp. 480-489, May 1977.

[Cin75] E. Cinlar, in *Introduction to Stochastic Processes.* Englewood Cliffs, N.J.: Prentice-Hall Inc., 1975.

[Cla83] D. W. Clark, "Cache Performance in the VAX-11/780," *ACM Trans. on Computer Systems*, vol. 1, no. 1, pp. 24-37, February 1983.

[CoB80] S. D. Conte and C. de Boor, in *Elementary Numerical Analysis* McGraw-Hill Book Company, 1980.

[DeB78] P. J. Denning and J. P. Buzen, "The Operational Analysis of Queueing Network Models," *ACM Computing Surveys*, vol. 10, no. 3, pp. 225-261, September 1978.

[DuB81] M. Dubois and F. A. Briggs, "Efficient Interprocessor Communication for MIMD Multiprocessor Systems," in *Proc. 8th Int. Symp. Computer Arch.*, pp. 187-196, May 1981.

[DuB82] M. Dubois and F. A. Briggs, "Effects of Cache Coherency in Multiprocessors," *IEEE Trans. on Computers*, vol. C-31, no. 11, pp. 1083-1099, November 1982.

[EmD78] J. S. Emer and E. S. Davidson, "Control Store Organization for Multiple Stream Pipelined Processors," in *Proc. 1978 Int'l Conf. on Parallel Processing*, pp. 43-48, August 1978.

[GoA84] A. Goyal and T. Agerwala, "Performance Analysis of Future Shared Storage Systems," *IBM Journal of Res. and Develop.*, vol. 28, no. 1, pp. 95-108, January 1984.

[GrH74] D. Gross and C. M. Harris, *Fundamentals of Queueing Theory.* John Wiley & Sons, 1974.

[HeS82] D. P. Heyman and M. J. Sobel, in *Stochastic Models in Operations Research*, vol. 1 McGraw-Hill Book Co., 1982.

[Hoo77] C. H. Hoogendoorn, "A General Model for Memory Interference in Multiprocessors," *IEEE Trans. on Computers*, vol. C-26, no. 10, pp. 998-1005, October 1977.

[KaW73] K. R. Kaplan and R. O. Winder, "Cache-based Computer Systems," *Computer*, vol. 6, no. 3, pp. 30-36, March 1973.

[Kle75] L. Kleinrock, *Queueing Theory Vol.1: Theory.* New York: John Wiley & Sons Inc., 1975.

[LaV82] T. Lang and M. Valero, "M-users, B-servers Arbiter for Multiple-Bus Multiprocessor," in *Microprocessing and Microprogramming, J. Euromicro*, pp. 11-18, August 1982.

[LVA82] T. Lang, M. Valero, and I. Alegre, "Bandwidth of Crossbar and Multiple-Bus Connections for Multiprocessors," *IEEE Trans. on Computers*, vol. C-31, no. 12, pp. 1227-1233, December 1982.

[LVF83] T. Lang, M. Valero, and M. Fiol, "Reduction of Connections for Multibus Organization," *IEEE Trans. on Computers*, vol. C-32, no. 8, pp. 707-716, August 1983.

[Lin68] C. L. Lin, in *Introduction to Cominatorial Mathematics*. New York: McGraw-Hill Co., pp. 38-40, 1968.

[Lip68] J. S. Liptay, "Structural Aspects of the System/360 Model 85, Part II: The Cache," *IBM Syst. Journal*, vol. 7, pp. 15-21, 1968.

[Mak84] B. A. Makrucki, in *A Stochastic Model of Multiprocessing* Ph.D. Thesis, The University of Michigan, 1984.

[MaG82] M. A. Marsan and M. Gerla, "Markov Models for Multiple Bus Multiprocessor Systems," *IEEE Trans. on Computers*, vol. C-31, no. 3, pp. 239-248, March 1982.

[Mea70] R. M. Meade, "On Memory System Design," in *AFIPS Proc., Fall Joint Computer Conf.*, vol. 37, pp. 33-43, 1970.

[MuM82] T. N. Mudge and B. A. Makrucki, "Probabilistic Analysis of a Crossbar Switch," in *Proc. IEEE 9th Ann. Symp. on Computer Architecture*, pp. 311-319, April 1982.

[MuA84] T. N. Mudge and H. B. Al-Sadoun, "Memory Interference Models with Variable Connection Time," *IEEE Trans. on Computers*, vol. C-33, no. 11, pp. 1033-1038, November 1984.

[MHB84] T. N. Mudge, J. P. Hayes, G. D. Buzzard, and D. C. Winsor, "Analysis of Multiple-Bus Interconnection Networks," in *Proc. 1984 Int'l Conf. on Parallel Processing*, pp. 228-232, August 1984.

[MHB85] T. N. Mudge, J. P. Hayes, G. D. Buzzard, and D. C. Winsor, "Analysis of Multiple-Bus Interconnection Networks", in review.

[OnI83] I. H. Onyuksel and K. B. Irani, "A Markov Queueing Network Model for Performance Evaluation of Bus-Deficient Multiprocessor Systems," in *Proc. 1983 Int'l Conf. on Parallel Processing*, pp. 437-439, August 1983.

[Pat81] J. H. Patel, "Processor-Memory Interconnections for Multiprocessors," *IEEE Trans. on Computers*, vol. C-30, no. 10, pp. 771-780, October 1981.

[Pat82] J. H. Patel, "Analysis of Multiprocessors with Private Cache Memories," *IEEE Trans. on Computers*, vol. C-31, no. 4, pp. 296-304, April 1982.

[Rao78] G. S. Rao, "Performance Analysis of Cache Memories," *JACM*, vol. 25, no. 7, pp. 378-395, July 1978.

[Rau79] B. R. Rau, "Interleaved Memory Bandwidth in a Model of a Multiprocessors," *IEEE Trans. on Computers*, vol. C-28, no. 9, pp. 678-681, September 1979.

[Rav72] C. V. Ravi, "On the Bandwidth and Interference in Interleaved Memory Systems," *IEEE Trans. on Computers*, vol. C-21, no. 8, pp. 899-901, August 1972.

[Ros70] S. M. Ross, in *Applied Probability Models with Optimization Applications*. San Francisco, Calif.: Holden-Day, Inc., 1970.

[SkA69] C. E. Skinner and J. R. Asher, "Effects of Storage Contention on System Performance," *IBM Systems Journal*, vol. 8, no. 4, pp. 319-333, 1969.

[Smi78a] A. J. Smith, "Comparative Study of Set Associative Memory Mapping Algorithms and Their Use for Cache and Main Memory," *IEEE Trans. Software Eng.*, vol. SE-4, pp. 121-130, March 1978.

[Smi78b] A. J. Smith, "Bibliography on Paging and Related Topics," *Operating Systems Review*, vol. 12, no. 4, pp. 39-56, October 1978.

[Smi79] A. J. Smith, "Characterizing the Storage Process and its Effect on the Update of Main Memory by Write Through," *JACM*, vol. 26, no. 1, pp. 6-27, January 1979.

[Smi82] A. J. Smith, "Cache Memories," *ACM Computing Surveys*, vol. 14, no. 3, pp. 473-530, September 1982.

[Str70] W. D. Strecker, in *Analysis of the Instruction Execution Rate in Certain Computer Structures* Ph.D. Thesis, Carnegie-Mellon Univ., 1970.

[Str76] W. D. Strecker, "Cache Memories for PDP-11 Family Computers," in *Proc. of 3rd Annual Symp. Computer Arch.*, pp. 155-158, January 1976.

[Tan76] C. K. Tang, "Cache System Design in the Tightly Coupled Multiprocessor System," in *AFIPS Proc. National Computer Conf.*, vol. 45, pp. 749-753, 1976.

[VLL83] M. Valero, J. Llaberia, J. Labarta, E. Sanvicente, and T. Lang, "A Performance Evaluation of the Multiple-Bus Network for Multiprocessor Systems," in *Proc. ACM Conf. on Performance Eval.*, pp. 200-206, 1983.

[WuB72] W. A. Wulf and C. G. Bell, "C.mmp-A Multi-mini-processor," in *Proc. Fall Joint Computer Conf.*, pp. 765-777, 1972.

[WLH81] W. A. Wulf, R. Levin, and S. P. Harbison, in *Hydra/C.mmp: An Experimental Computer System*. New York: McGraw-Hill, 1981.

[Yeh81] C. C. Yeh, in *Shared Cache Organization for Multiple-Stream Computer Systems* Ph.D. Thesis, University of Illinois, January 1981.

[YPD83] P. C. C. Yeh, J. H. Patel, and E. S. Davidson, "Shared Cache for Multiple-Stream Computer Systems," *IEEE Trans. on Computers*, vol. C-32, no. 1, pp. 38-47, January 1983.

[YPD82] D. W. L. Yen, J. H. Patel, and E. S. Davidson, "Memory Interference in Synchronous Multiprocessor Systems," *IEEE Trans. on Computers*, vol. C-31, no. 11, pp. 1116-1121, November 1982.

[YeF80] W. C. Yen and K. S. Fu, "Performance Analysis on Multiprocessor Memory Organization," in *Proc. ACM Pacific '80 Conf. Distributed Processing*, pp. 142-153, November 1980.