CANADIAN THESES

THÈSES CANADIENNES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ MICROFILMÉE TELLE QUE NOUS L'AVONS REÇUE

Performance Modeling of a Loosely coupled Multi-microcomputer System

D.Muralidharan

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montréal, Québec, Canada.

August 1985

ISBN 0-315-30634-3

ABSTRACT

Performance Modeling of a loosely coupled

Multi-microcomputer system.

D.Muralidharan

A loosely coupled Multi-microcomputer system comprises of autonomous, multiple microcomputers which interact with each other, by passing messages over a communication network. Such a system can be designed to provide higher processing power or better system flexibility and reliability in contrast to a single microcomputer system.

CUENET (Concordia University Educational NETwork) is one such multi-microcomputer system consisting of both 8 and 16 bit microcomputers, arranged in a master-slave hierarchy. They communicate with each other by passing messages over a parallel bus called C-Bus under the control of a centralized bus controller.

The performance of a multi-microcomputer system , besides other factors, is influenced by the manner in which the processes running on the individual computers interact. In performance modeling of multi-microcomputer systems, these aspects are studied using a functional model of the system.

Stochastic Petrinets (SPNs) and Generalized Stochastic Petrinets (GSPNs) have been employed in the performance modeling of multiple computer systems. SPN and GSPN modeling techniques have the ability to describe the concurrency and conflict inherent in the system in parallel processing. Delay and throughput are the two performance mesaures that are derivable from the analysis of such models. The application of SPN and GSPN modeling techniques, found in the recent publications, are confined to modeling tightly coupled systems.

In this thesis, we have applied the SPN and GSPN modeling techniques to study the performance of CUENET from the message communication point of view. The detailed SPN, GSPN models and aggregate GSPN models for two and three processor CUENET configuration are presented and results of the analysis of these models are discussed. The applicability of the SPN and GSPN techniques to loosely coupled multi-microcomputer systems is demonstrated.

ACKNOWLEDGEMENT.

I would like to express my indebtness to my supervisor Dr. T.Radhakrishnan. His guidance, assistance and encouragement were an asset during the thesis work in particular and in my stay at Concordia, in general.

My thanks are due to Dr. V.S.Alagar who gave me valuable advice during the thesis work despite his busy schedule.

I wish to express my sincere and heartfelt thanks to my friends Mr. K.Venkatesh and Mr. R.N.Prasad who have helped me at various stages of my thesis work and especially in bringing the thesis report to the final shape by their drafting skills. Their presence has made my stay in Concordia very enjoyable and memorable.

Dr. M.A.Marsan of Polytechnico de Torino, Torino, Italy has helped me in the thesis work by providing the GSPN software. My thanks are due to him and his colleagues.

My thanks are due to Mr. Clifford Grossner who was always available for consultation regarding CUENET.

I wish to express my sincere thanks to all of my department colleagues and staff who have made my stay at Concordia a very pleasant experience.

My sincere thanks are due to the Canadian Commonwealth Scholarship committee for the financial support during my

graduate studies.

I wish to express my heartfelt thanks to my dear wife for taking care of our family in India during my stay here.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

FIGURES

TABLES

CHAPTER I

INTRODUCTION

1.1 Evolution of Distributed computing systems:

The motivating factors for the continuing development of computing systems are[ENSLO-77]:

(1)speed of computation and throughput,

(2)Flexibility,

(3)Availability,

(4)Reliability.

Initially the computer system manufacturers concentrated on increasing the speed of computation and throughput by increasing the speed of the hardware circuitry and then by developing efficient operating system software [ENSLO-77,LIEBO-81]. Thus resulted the complex centralized main frame systems like the IBM system 360 and 370 series and recent 4300 series(4331,4341),DEC's series 10. As the applications of computers grew in the areas like banking, patient monitoring and industrial process control the other motivating factors(i.e. Flexibility, Availability and Reliability) along with increased computing throughput, gained importance. These system requirements caused the computing system designers to think in terms of multiple processing units interlinked in some fashion, thus giving rise to multiple processors systems and Distributed Computing Systems[LIEBO-81]. Enslow characterizes a fully Distributed Computing System(DCS) by the following

attributes[ENSLO-78]:

1. A multiplicity of general purpose components including both physical and logical resources, that can be assigned to specific tasks on a dynamic basis.

2. A physical distribution of these physical and logical components of the system interacting through a communication network.

3. A high level operating system that unifies and integrates the control of the distributed components. Individual processors can have their own operating systems.

4. Cooperative autonomy, characterizing the operation and interaction of both physical and logical resources.

5. System transparency, permitting service to be requested by name only.

Generally these characteristics are present in varying degree in the actual systems thus providing varying shades of distribution of control, computation and processor interaction. Thus there exists a wide diversity of system architecture, control and processor interaction in DCSs. There are various catagories depending on the nature of location, degree of coupling, processor logical relationship, processor connectivity and so on. One such catagorization based on the degree of coupling and processor connectivity classifies DCSs as:

(a) Multiprocessors,

(b) Computer networks,

(c) Local area networks.

(a) Multiprocessors:

A Multiprocessor is a DCS which has the following charateristics[ENSLO-77,SATYA-80]:

* contains two or more processors with approximately comparable capabilities,

* these processors are tightly coupled and communicate through shared memories.,

* all processors share access to input/output channels, control units and secondary storage devices,

* the entire system is controlled by one operating system providing interaction between processors, their programs etc.

The concept of mutliprocessors is not new and some early examples of them are IBM 360/65, Burroughs D8.5, Bendix G21, IBM 370/168 and CDC Cyber 170 [FULLER-78,SATYA-80,ENSLO-77]. In these early systems, the degree of tight coupling is quite high in that each processor accesses the common memory for each program code execution and data manipualation [WEITZ-80,FATHI-83]. With the advances in semiconductor memories and integrated circuit technology, the present day mini or microcomputer has its own private memory and also exceutive software. In such multiple computer systems, employing tight coulping, the shared memories are employed for moving data from one processing element to the other. The program execution is done in the private memory of the processors. Here the degree of tight coupling is moderate.

Such DCS are also catagorised as Multiprocessors. Some recent examples of such multiprocessors are: Cm* developed at the Carnegie Mellon university [FULLER-78], Bell Canada's DATAPAC SL-10[WEITZ-80] , Torino MultiProcessor(TOMP) developed, as part of the Mumicro project, at Polytechnico De Torino, Torino,Italy [CONTE-81].

(b) Computer networks:

Computer networks are DCSs possessing the following characteristics [TANEB-81]:

*Each processor is a fullfledged conventional computer systems with independent Input/Output devices and secondary memories.

*the processors are geographically dispersed and loosely coupled,

*inter processor communication is by passing messages in the form of packets over serial communication links under the control of communication controllers which route the message in an optimum fashion,

* each processor has its own operating system with a overall network operating system for controlling the intercomputer communication.

The computer networks are also known by the names, Long haul networks and public data networks. ARPANET, TYMNET are examples of computer networks.

(c) Local area networks(LANs):

The rapid and continuing advances in integrated semiconductor technology (LSI, VLSI) gave rise to the development of low cost microprocessors. Now microprocessors are available from 8 bits to 32 bits with varying computational power and their cost is still on the decreasing trend [FULLER-78,GUPTA-84]. With the advent of these low cost, high capability microprocessors, it is now possible to perform computations at the site of data. This has resulted in intelligent terminals, controllers, personal computers and various consumer goods [FULLER-78]. Also interest grew in interconnecting such diverse computing systems within an organization leading to the evolution of LANs.

A LAN has the following characteristics [TSAO-84,CLARK-78]:

* limited in geographic scope- within 0.1 to 1.0 km range, usually linking processors within a building or within a single floor,

* provides high bandwidth, short delay communication (over 1Mbits/sec) over inexpensive communication media (twisted pair wires, coaxial cables),

* processors are loosely coupled and communicate by passing massages over the communication medium of the network,

* employs simple routing and control algorithms as

opposed to the long haul computer networks. Examples of early Local area networks are ETHERNET ,WANGNET etc[CLARK-78].

The typical enviornment where LANs are applicable today are [TSAO-84]:

(1)Office automation- supporting functions like electronic mail, text processing ,document distribution and vioce storage.

(2)Universities- to provide communication network to access central or distributed facilities, special purpose application software etc.

(3)Factories-help in automating and controlling manufacturing techniques such as CAD/CAM, robotics, numerically controlled processes etc.

(4)Hospitals- to provide communication support for patient monitoring, patient file retrival etc.

Several different configurations and communications techniques are possible in the case of LANs, since they have do not have the same operational restrictions of long haul computer networks.

Several eduactional institutions and laboratories have built and been building different types of multiple microprocessor DCS with tight coupling and loose coulping, for studying their performance and try different alternative architectures [MARSAN-83]. The Cm* at the Carnegie Mellon

University, X-tree at the University of California, Berkeley, MICRONET at the University of New york at Buffalo, [WITTIE-80], TOMP at Polytechnico De Torino, Torino, Italy [CONTE-81] and CUENET[GROSS-82] are some examples.

The CUENET(Concordia University Educational NETwork) is loosely coupled LAN comprising of 8 and 16 bit microprocessor systems, developed here at the Department of Computer Science, Concordia University, Montreal, Canada.

1.2 Factors in the design of DCS:

The major aspects of the design of a DCS and their applications lies in the design or selection of the following:

(a) A set of processors(homogeneous or hetrogeneous) and memories that are interconnected.

(b) An interconnection network that physically connects the various processors and memories.

(c) Subdivision of the sequential program into subprograms that can be run concurrently on the various processing elements.

(d) An operating system that will coordiante the execution of different subprograms, allocate the subprograms to different processing elements and manage them.

There are various ways these design steps can be carried out and also there areas which present interesting and challenging problems for research. Some of the problems a

designer of a DCS has to deal with are [FULLER-78]:

1.)Task decompostion:How should tasks executed on a uniprocessor be partitioned so that they can be run on a set of smaller processors? The task decomposition must take into account the architecture of the DCS on which it is intended to run. The optimum way this partitioning can be done is still an open problem [KUCK-77,GAJSK-85].

2.)Interconnection structure:What are the most effective types of processor/memory and processor/processor interconnection structure? How flexible the interconnection structure be?

3.)Operating system structure: What kind of operating system is suitable for controlling, resource managing, distribution of tasks among the processing elements and their protection in a distributed computing system comprising of hundreds of processing elements?

4.) Interprocessor interference:The partitioned tasks distributed on the processing elements in a DCS will need to cooperate with each other so that they can provide the requiste service or perform the required computation. This would involve contention for memories, I/O devices for communication paths. These must be kept to the minimum.

5.)Dead lock aviodance:With multiple processors contending for resources a dead lock situation can develop, where each of a group of processors is waiting for resources assigned to the other processors in the group, and none of the processors in the group being in a position to proceed until

its demands are satisfied. The designer must provide a facility for avioding the dead lock or for taking remedial measures in the event of a dead lock.

6.)Input-Output: How the Input/Output devices in general and secondary storage in particular are integrated into a DCS?

7.)Fault tolerance:How one can utilize the hardware and software structure of a DCS to provide capability to survive failures of components in the system?

Different designers have overcome these problems differently. Thus each DCS performs in different ways.

1.3 Performance Evaluation of DCS:

The term "Performance " of a computer system has many different interpretations. Normally "Preformance" answers questions like "How well does the system enables one to do what he wants to do?" and " How well the system does what it is intended to do?"

The Performance of a system can be discussed from two different viewpoints: first the effectiveness with which the system handles a specific application and second the internal efficiency of the system [SVOBOD-76].

Effectiveness is what is seen by the system user. Efficiency is how the system uses its resources in order to process the respective workload. System effectiveness, ofcourse, is influenced by the internal efficiency.

Thus the performance of a DCS is the effectiveness with which the various processors, memories and other devices, the intercommunication structure are utlized under a particular workload.

The system components influence the system performancee through their own characteristics and through their mutual interaction. The performance of a system is assessed with respect to a set of parameters which are termed Performance measures.

Performance Evaluation is the process of evaluating the performance of a system. The major steps involved in performance evaluation of a system are: (1) Definition of the relavent performance measures,
(2) Determination of the values of the Performance measures with respect to the system structure and system workload.

Performance evaluation of a DCS is the study of how the various components, both hardware and software, interact with each other and how the flow of data and information is maintained and controlled in the interconnection structures and how efficiently the system fuctions [AGARWA-83]. The efficiency of a DCS depends upon many factors related to different ways hardware and/or software components interact i.e. contention for shared resource, synchronization of tasks etc. By doing a performance evaluation study of a system one can know how efficiently the various processing

elements are structured, the efficiency of interprocessor communication schemes and the distributed software [MARSAN-83].

We will discuss the various performance measures of DCS and different methods of performance evaluation.

1.3.1 Performance measures of DCS:

Performance measures are a set of quantitative parameters which characterize the performance of a system [SVOBOD-76,WEITZ-80].

The performance measures of a conventional computing system , principally are, the throughput,turnaround time,response time and sytem availability.

The throughput is the amount of work completed in unit time for a given workload. It is normally expressed as MIPS(Millions of Instructions Per Second).

The turnaround time is the elapsed time between submitting the job and receiving the output. Normally it is expressed in secs or minutes. Normally this is for batch jobs.

Response time is the turnaround time of requests and transactions in an interactive real time systems.

Availability of the system is the percentage of time a system is available to the user.

In a DCS the performance of the system depends on the individual capabilities of the processing elements, the type of interaction between them, the type of interconection network and the way contention for a shared resource is managed. Here each processor will have individual performance measures similar to the conventional Systems. Besides these The main performance measures of DCS are processing power, throughput, waiting time for a shared resource like the global bus or the common memory.

The processing power for a distributed computing system expresses the amount of speedup or the increase in actual processing capability over a uniprocessor. Ideally the processing power of a DCS with 'n' processors must be 'n'(assuming unit processing capability for each processor).In reality this is not so since, the processing elements in a DCS spends some processing time in sending and/or receiving information from other processing elements. The processing power expresses the effective processing capability over a uniprocessor. Hence processing power P is expressed by the following expression:

$P = E(N_a)$ .....(1.1)

where N_a = number of active processors in the DCS.

Processing Efficiency of a processor in a DCS is given by the following relation.

Processing Efficiency = (P/N)X100 .....(1.2)

where N = Number of processors in the DCS.

Throughput is a measure of the amount of work done per unit time. It is normally expressed as number of jobs or programs executed or messages transmitted per second. The unit for the throughput depends on whether one is looking at the system as a whole or at a subsystem. While one is considering the complete DCS, the throughput is proportional to the speedup or processing power P. When one is looking at the intercommunication subsystem then it is the number of messages transmitted. The throughput of a system or a subsystem can be given by the following expression:

Throughput = fraction of time the (sub)system is busy X rate of operation .....(1.3)

Thus throughput expresses how effectively the system or the subsystem is functioning. Figl.1 shows the relationship between system throughput and the number of processors.Ideally the system throughput is to increase linearly with the number of processors(curve a). In practice the relationship would be more as shown by the curves b and c. In each case the system throughput levels off for a particular number of processors and then starts to decrease due to the "saturation effect" [FATHI-83]. The point at which the system throughput levels off and then saturates depends on a number of factors. Some of the factors are: the type of coupling, the nature of intercommuniation network, the nature of application, the method of arbitration for the shared system resources etc. Thus the study of the throughput variation helps one in

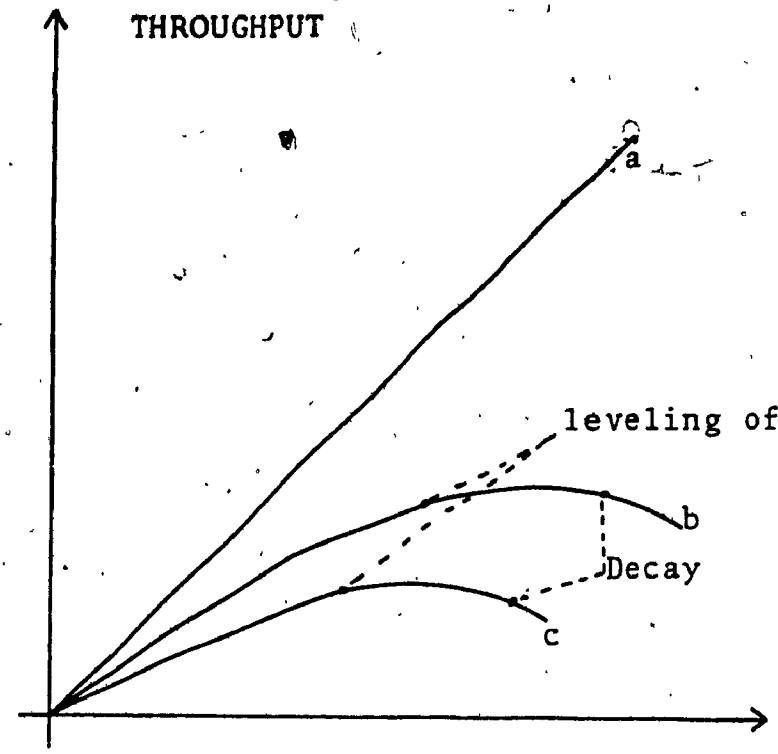


Figure 1.1 THROUGHPUT - "Saturation effect"

estimating the optimum size of the system. The waiting time or queueing time for the various shared resources like global bus, common memory indicate how well the resources are utilized in a DCS.

These are some major performance measures which are considered in the performance modeling and measurement.

1.3.2 Methods of Performance Evaluation:

Performance evaluation of a system can be carried out by different methods. These methods are[AGARWA-83]:

(a) Performance Analysis,

(b) Simulation,

(c) Performance measurements.

(a) Performance Analysis:

Performance Analysis of a system is the study of how the various components of the system interact and affect the system's performance by representing the system and its components using a model [SVOBOD-76]. A model is an abstraction of the actual system representing the various system's components and their interactions. Also the model represents the main factors determining the system's performance in statistical or mathematical way [CHANDY-81]. From these relations the performance measures of the system can be computed. A wide spectrum of models can be derived depending upon the level of detail used for the representation of the system's behavior. A very detailed description of the system will lead to an extremely complex models where as a more abstract or higher level view of the system can result in a fairly simple model which is easier to analyse [MARSAN-83,BUZEN-77]. Thus different kinds of models are used in performance analysis of computer systems. Some of the types of models used are: queueing (network) models, stochastic and graph models like stochastic Petrinets, Generalized Stochastic Petrinets and Markov models [KLEIN-75,MARSAN-83]. These models possess markovian property which makes the derivation of the various performance measures tractable. For these models usually probabilistic workloads are assumed [SVOBOD-76].

Performance analysis is normally employed for a system which is in its design stage or early stage of realisation and for a system on which the measurement of various parameters is invloved or is not cost effective [SVOBOD-76]. The performance analysis is also used to understand the interaction of the various components of a system and their effect on the system's performance so that with minimal measurement of certain parameters one can monitor the performance of the system.

(b) Simulation:

The technique of simulation can be viewed as a combination of modeling and measurement. The simulation process requires a model of the system and a simulator. The simulator is a mechanism that simulates the system behavior as specified by the functional model of the system and the work load. It yields the necessary data required for performance analysis [SVOBOD-76].

Simulation helps in estimating the performance of new designs and new configurations before their implementation. The complexity of simulation depends upon the level of detail included in the model. Simulation based on a very detailed model becomes very costly both in development and use. For performance analysis, a simulator must only simulate those events that change the system's state.Such simulators are called Discrete Event simulators. These simulators are quite popular for simulating computing

systems. The development of simulation languages like GASP(General Activity Simulation Program), GPSS(General Purose System Simulator) and SIMSCRIPT provide the basic support for building discrete event simulator for a system under study.

(c) Performance Measurement:

Performance measurement constitutes part of performance evaluation where certain data like amount of CPU secs, number of messages received and/or sent by each processor, waiting time for each processor while contending for the shared resource etc are determined quantitatively. These measurements are useful in the determination of the various performance measures of the system under study. Normally some kind of monitoring scheme is employed for the measurement of these data. There are hardware monitors, software monitors and hybrid monitors [NUTT-75]. The choice of a particular type of monitoring scheme is dependent upon what type of data to be collected, the amount of overhead that the system can tolerate, the location of the monitoring facility and so on [AMER-82]. In general a suitable hybrid monitoring scheme is empolyed.

Some practical examples of monitoring systems used for performance measurement are:

(a)The Integrated Instrumentation Enviornment(IIE), is a comprehensive performance monitoring scheme which emphasizes an integrated experiment management concept. It is

implemented on Cm* multiprocessor [SEGALL-83].
(b)A hybrid monitor consisting of a hardware monitor called ZAHLMONITOR III and a software monitor, designed for the instrumentation and performance monitoring of the Erlangen general Purpose Array(EGPA) [FROMM-83]. The EGPA is a hierarchical multiprocessor array.

The term performance evaluation refers to both the performance analysis and performance measurement which are complementary to each other [SVOBOD-76]. The analytical model provides the frame work for the measurement and the measurement furnishes data for validating the analytical model. Further the analytical model aids in testing a hypothesis and finding solution to performance problems while the correctness of the predictions of the model is finally verified by measurement.Thus while doing the performance evaluation of a system both the performance analysis by modeling and the measurement of the performance measures derived must be carried out. Normally at the design and early implementation stage analysis by modeling is carried out. Once the system implementation is complete and running fairly well the measurements of the various performance measures must be done so that the analytical model is validated and also improved.

1.4 Thesis motivation and outline:

In the preceding sections we have discussed the evolution of DCS and main thrust behind the development of both tightly

coupled and loosely coupled DCS at various universities and laboratories.

It is observed that it is easier to model the communication aspects in a tightly coupled systems than in loosely coupled systems [REID-82]. The reason according to Reid, is the difficulty in obtaining the bounds on the number of messages and the number of resources required in a loosely coupled system. This seems to be coroborated by the fact that recently published performance analysis studies have been on tightly coupled systems. Some examples are: Performance analysis of TOrino MultiProcessor(TOMP) by Marsan et al [MARSAN-83,MARSAN-84,MARSAN-85] and performance modeling of Fault Tolerant Multiprocessor (FTMP) by Woodbury et al [WOOD-84]. In these performance modeling the graphic models Stochastic Petrinets(SPNs) and the Generalized Stochastic Petrinets(GSPNs) have been used for easy system description. The existence of a loosely coupled DCS(CUENET) in our department and the recent interest in graph models motivated us to attempt this performance evaluation study using SPNs and GSPNs.

Chapter II discusses the modeling requirements of DCS, descirbes the graph models SPNs and GSPNs used in our study. Chapter III briefly discusses the hardware/software architecture,message communication aspects and performance measures of CUENET. Chapter IV deals with the SPN models for 2 and 3 processors configuration of CUENET. Here the

steps invloved in the analysis of the models and the limitations of SPN in modeling CUENET are discussed. The GSPN modeling of CUENET is dealt with in Chapter V. The last chapter deals with the conclusions and suggestions for future work.

CHAPTER II

MODELING TECHINIQUES FOR PERFORMANCE ANALYSIS OF DISTRIBUTED COMPUTING SYSTEMS

2.1 General requirements for modeling Distributed Computing Systems(DCS):

In general a DCS consists of a set of physically seperate processors which are connected to each other by an interconnection network. The nature of the DCS varies depending on the extent of coupling among the various processors and the distribution of control of various functions of the system. The processors can be "tightly coupled" where they communicate amongst them by means of shared memories or they can be "loosely coupled" where they communicate by passing messages to each other over the interconection network. The control of the system can be done from a central site or it can be decentralized so that each processor can function in an autonomous fashion[FATHI-83,WEITZ-80].

As discussed in chapter I, the performance of a DCS depends on a variety of factors:

1)The number and nature of processors,

2)The degree of coupling among them,

3)The type of control and the kind of interconnection

network which connects all the processors,

4)the type of applications that can be or is run on the system.

All these factors have varying degree of effect on the performance of the DCS. In order to study the effects of these factors on the system's performance, various modeling techniques have emerged over the years[MOLLOY-81,MARSAN-83].

To have meaningful study of the DCS being modeled, the modeling technique must have certain attributes[MOLLOY-81]. They are:

1)Logical correctness- the model must depict clearly and correctly the logical relationship of components of the system and be able to handle varying degree of logical complexity.

2)Flexibility- The modeling technique should allow for varying degree of abstraction. For instance, the abstraction may be at the level of software modules or at the level of instructions.

3)Ease of representation of concurrency- the modeling technique must allow easy representation of concurrency and conflicts within the system.

4)Deal with performance issues- the modeling technique must provide good facility to define certain performance parameters and determine their values mathematically.

Some of the modeling techniques employed in the preformance analysis of DCSs are:

a)Stochastic models,

b)Graph models,

c)Integrated models.

a) Stochastic models: They include Queueing network models and direct Markov models.

In Queueing network models, the various components of the system are represented by servers to which jobs/customers arrive at a particular rate and the server provides the service at a particular rate [KLEIN-75,CHANDY-81]. Usually exponential distributions are assumed for the various arrival and service rates. In the queueing network, the movement of the jobs among the servers is represented by a discrete Markov chain.

In direct Markov chain models, the various states the system goes through is represented by a finite Markov chain. Here the way the various system states are defined is very important.

In the above two stochastic models the emphasis is on the performance issue.

B)Graph models: In these models the logical relationship between the system components can be easily represented due to the graphical nature of these models[MOLLOY-81]. For modeling interaction between software modules control graphs and precedence graphs have been employed. The standard Petrinets, orginally proposed by C.A.Petri[PETER-81] enable easy representation of concurrency and conflict within a

DCS. In these graph models the logical correctness can be easily checked by applying certain basic rules in their construction [MOLLOY-81,PETER-81].

C)Integrated models: In these models effort is made to integrate both the logical correctness and performance issues. One such model is the Timed Petrinets[ZUBER-80] which associates a fixed time interval with each transition of a standard Petrinet. In timed petrinets, due to the deterministic time, modeling of conflicts within a system becomes difficult[MOLLOY-81].

In another method of integration, the integrated model is obtained by introducing stochastic properties within a graph model. This integration can be best viewed when the interface of seperate correctness and performance modeling is considered. Graph modeling takes the structure as input and by formal rules constructs the state space of operation. Performance modeling starts with a state space and makes assumptions on the timing of operations (typically memoryless to make it markovian) and calculates delay and throughput measures in a straight forward manner.For a graph model which is stochastic these two operations overlap. This allows a testing procedure to be employed which investigates the resulting state space of the graph model for logical correctness while ignoring time. Then performance testing procedure (probably automated) can be appiled on the verified state space.

The stochstic Petrinets[MOLLOY-81] proposed by M.K.Molloy and the Generalized Stochastic Petrinets [MARSAN-84] proposed by Marsan et al belong to this type of integrated models. These integrated models provide a good integration of logical and performance modeling power for studying the performance of DCS.

2.2 Petrinets:

2.2.1 General Description and Definitions:

Petrinet is a graph model originally proposed by C.A.Petri and later developed by other researchers for modeling computer systems[PETER-81]. The Petrinets have recently gained more popularity for the modeling of DCSs, because of the ease with which they model concurrency, conflicts and synchronization between processors or processes.

Petrinet, from graph theory point of view, can be thought as a bipartite, directed graph comprising of objects called Places and Transitions[AGERWA-79].

Definition 2.2.1:The sturcture of a Petrinet C is given by the triple,

$$C = [P,T,A] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.1)$$

where, $P = \{p_1, p_2, \ldots, p_n\}$ is a finite set of places,

$T = \{t_1, t_2, \ldots, t_m\}$, is a finite set of transitions,

and $n, m \geq 0$.

$A = \{TXP\} \cup \{PXT\}$ is a set of arcs.

The places and transitions together represent a set of events or actions being performed by the system being modeled. The way the places and transitions are connected by the directed arcs represent the structure of the system under study.

If there is a directed arc from a place to a transition, then the place is termed input place of that transition.

Similarly, if a directed arc exists between a transition and a place, then the place is termed as output place of that transition. In general a place can be connected to a transition by more than one directed arc.

Graphically, the places are represented by circles, transitions by bars and tokens by small dark dots within a circle. Fig 2.1 gives an example of a Petrinet.

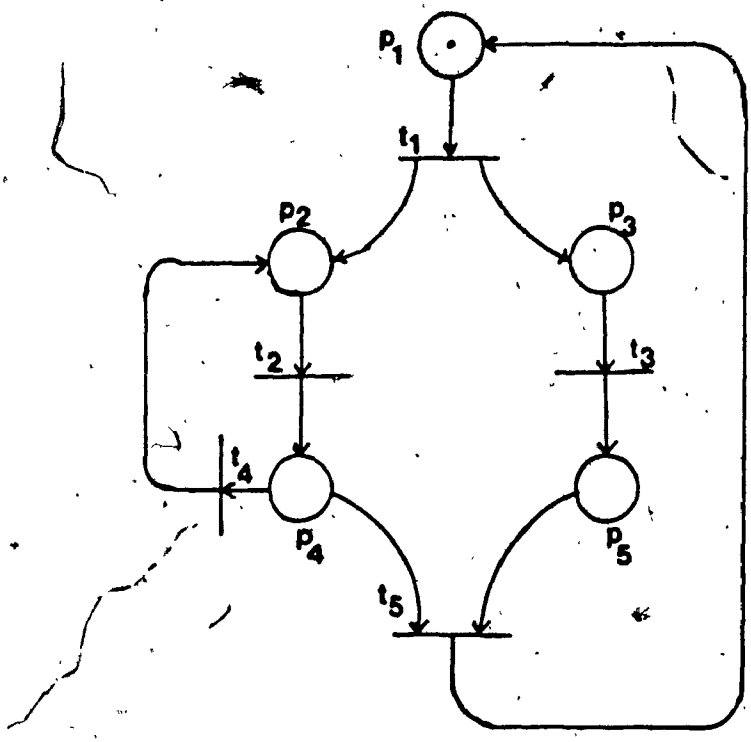


Figure 2.1 Example of a Petrinet

This Petrinet has 5 places and 5 transitions. The set P is given by $\{p_1, p_2, p_3, p_4, p_5\}$ and the set T is given by $\{t_1, t_2, t_3, t_4, t_5\}$.

Definition 2.2.2: A place is called a input place of a trasnition if there exist a directed arc from the place to the transition.

Definition 2.2.3:A place is an output place of a transition if there exist a directed arc from the transition to the place.

Definition 2.2.4:Input function I(t) of a transition is the set of places which form the input places to the transition.

Thus, $I(t) = (p \mid (p,t) \in A)$ .....(2.2)

Example, In fig.2.1, $I(t_1) = (p_1)$.

Definition 2.2.5: Output function O(t) of a transition t is the set of places which form the output places to that transitiion.

Thus $O(t) = (p \mid (t,p) \in A)$ .....(2.3)

Example, in fig.2.1, for the transition t_1, $O(t_1) = (p_2, p_3)$.

Each place in Petrinet can hold a set of tokens representing certain event or condition.

Definition 2.2.6: A marking M of a Petrinet is the assignment of tokens to the places in a Petrinet.

M : P---> I where I{ 0,1,2...}.

M can also be viewed as a vector whose ith component m_i represents the number of tokens assigned to the place p_i. Thus M is defined as,

$M = \{m_1, m_2, \ldots m_n\}$ .....(2.4)

Example: In fig.2.1 the petrinet has the following marking-(1,0,0,0,0). A petrinet C with marking M is a marked petrinet.

Definition 2.2.7: The structure of a marked petrinet PN is given by a four tuple,

$PN = [P,T,A,M]$ .....(2.5)

Definition 2.2.8: A transition is "enabled" if each of its input places has atleast as many tokens as the number of arcs connecting that place to the transition.

Definition 2.2.9: An enabled transition can "fire" in a Petrinet by removing all the enabling tokens from its input places and depositing in each of its output place one token for each arc connecting that place.

The firing of an enabled transition t in a given marking M_o produces a new marking M_i which are related by the following relation.

$M_i = M_o - I(t) + O(t)$ .....(2.6)

Definition 2.2.10: A Petrinet is "executed" by firing all the enabled transitions in a given marking.

Thus in executing a marked Petrinet, the position and number of tokens change in the marked Petrinet. Also a set of markings will be produced by the firing of transitions during the execution of a marked Petrinet.

Definition 2.2.11: Reachability set R(PN) of a marked

Petrinet is the set of all markings which are reachable from a given marking M_o.

Definition 2.2.12: A marking M_i is immediately reachable from a marking M_o if and only if some enabled transition t in the marking M_o yields M_i.

Definition 2.2.13: A marking M_i is reachable from marking M_o if it is immediately reachable from marking M_o or is reachable from any marking which is immediately reachable from M_o.

Definition 2.2.14: A transition t in a marked Petrinet PN is "live", if for each $M_i \in R(PN)$ there exists a marking reachable from M_i in which t is enabled.

Definition 2.2.15: A marked Petrinet PN is "live" if all of its transitions are live.

Definition 2.2.16: A place in a marked Petrinet PN is k-bounded if and only if there exists a k such that $m_i \leq k$ for all $M_i \in R(PN)$.

Definition 2.2.17: A marked Petrinet PN is k-bounded (k-safe), if for some fixed k each place in it is k-bounded.

Definition 2.2.18: A place p_i in a marked Petrinet PN is safe if it contains at most 1 token (i.e. 1-bounded).

Example:In fig2.1, for the given marking place p_1 is a safe place.

Definition 2.2.19: A marked Petrinet PN is safe if each place in it is safe.

In the Petrinet shown in fig 2.1 the place p_1 is having one token and the other places have zero tokens. The place p_1 is a safe place and it can be shown(see section 2.2.2) that the marked Petrinet is also safe, since no place will at any marking contain more than one token in it.

The liveness and boundedness are very important properties of marked Petrinets which are useful in modeling systems for studying their operation and acertaining their logical correctness [MURATA-84].

Definition 2.2.20: A marking M_i of a marked Petrinet PN is said to be a "terminal" marking if no transition in the Petrinet is enabled in that marking.

The presence of a terminal marking in the reachability set of the system modeled indicates that there are conditions under which the system can exhibit a deadlock. A marked Petrinet containing a terminal marking is not a live Petrinet.

2.2.2 Analysis of Petrinets:

The analysis of Petrinets mainly consists of determination of the various markings that are reachable from a given marking, which also form the reachability set of the Petrinet. Also during the analysis whether any "terminal"

marking(s) exist or not is determined. In general the reachability set of a marked petrinet is finite. It is more so for a marked Petrinet which is bounded and live.

The reachability set is determined by constructing a reachability tree which clearly depicts the relationship between markings and enabled transitions. In one marking more than one transition can be enabled. The resulting marking is found by using the relation(2.6). The successive generation of markings from the initial marking is followed till a known marking or a terminal marking is reached.The analysis is terminated when all enabled transitions and their resulting markings are examined.

The determination of reachability set of a marked petrinet PN, in the similar lines as described above, can also be done by assigning unique and easily manipulatable number to each marking and also to each transition. For each transition is associated one number for its input places and one for its output places.There are two methods: one using prime numbers (as proposed by M.K.Molloy) [MOLLOY-81] and another method, devised by us, using binary and hexadecimal numbers. These methods are detailed below.

(a)Prime number method:

Here a distinct prime number is assigned to each place in the Petrinet. A unique prime number called State number is associated with each marking of the marked petrinet.Also

with each transition an input number and an output number (both prime numbers) are associated.

These prime numbers are determined in the following way: The marking of the marked Petrinet describes the places which have tokens. The state number for this marking is determined by multiplying the various prime numbers associated with places having tokens. If the place has multiple tokens then the prime number is multiplied as many times as the number of tokens in that place.

Example:In fig 2.1, the given marking is (1,0,0,0,0) and the prime numbers 2,3,5,7,11 are associated with the places p_1,p_2,p_3,p_4 and p_5 respectively. The state number for this marking is 2.

The input and output prime numbers for each transition is determined in the same manner,i.e. by multiplying the prime number associated with the input places and the output places.

Example: In fig 2.1, for the transition t_1, input number is 2 (its input function consists of p_1) and the output number is 3X5 = 15 ((p_2, p_3) form the output function).

(a) Determination of enabling of a transition in a given marking:

A transition is enabled in a given marking, if the state number of the marking is divisible by the input number of the transition.

Example:In fig2.1, for the given marking (1,0,0,0,0) t_1 is the only enabled transition. Thus the state number for this marking (2) is divisible by the input number of t_1 (2).

(b) Determination of new marking:

The marking M_i immediately reachable from a given marking M_o by firing an enabled transition t_i is obtained by dividing the state number by the input number of t_i and multiplying by the output number of t_i.

Thus, $M_i = (M_0 XO(t_i))/I(t_i)$.

Example: In fig 2.1, for t_1 the new marking state number is (2/2)X15 = 15.

The steps for determining the markings in the reachability set in a given marked Petrinet, using this prime number method, are as shown below:

1) Describe the Petrinet in terms of input and output functions of each transition of the Petrinet.

2) Determine the input and output prime numbers for each transition as described earlier.

3) From the initial marking given, determine the initial state number.

4) Now for this marking determine which are the transitions that are enabled in this marking, as described in (a). If it is determined that no transition is enabled in the given marking, then that marking will be labeled as a "terminal" marking.

5) Determine the next marking ,which is obtained by firing

an enabled transition, as shown in (b).

The number of tokens in each place in the given marking is determined by noting how many times the state number is divisible by the place prime number.

6) Check to see if this state number already exists if not give a new state number and increment the state count.

7) The steps 4 to 6 are repeated for each marking that are obtained.

8) The steps 4 to 7 are repeated till we get markings that are already existing or a terminal marking i.e. in step 4 no transition is enabled for that marking.

The program listing for the determination of the reachability set using the above steps is given in Appendix A.

Applying these rules for determining the reachability set for the given Petrinet shown in fig 2.1 we obtain its reachability set. There are 5 markings in its reachability set and they are as shown :

1 0 0 0 0

0 1 1 0 0

0 0 1 1 0

0 1 0 0 1

0 0 0 1 1

From this reachability set, we can infer that the given Petrinet in fig2.1 is live and also for the given marking it is safe.

This prime number method has a drawback. As the number of places increases the size of the state number for a marking exeeds the word length(i.e 48 bits in CDC Cyber 835), thus placing an upper limit on the number of places (with single token in each) it can handle. The upper limit is about 23 places.
To overcome this we divised the binary and the hexadecimal method.

(b) Binary and Hexadecimal method:
These methods adopt, in principle, the same steps in determining the various markings in the reachability set of a given marked Petrinet, as described in the previous method.

The major difference is in the way the state number, input and output numbers for the transitions of the Petrinet are determined. These methods overcome the place limitations of the prime number method.

In the binary method, a number to the base 2 is determined for each of the markings and the input and output functions of all trasnitions in the given Petrinet.
Example:The initial marking in the Petrinet shown in fig 2.1 is 1,0,0,0,0. The places p_1,p_2,p_3,p_4 and p_5 are assigned the weightage $2^0,2^1,2^2,2^3$ and 2^4 respectively. Thus the state number is given by $1X2^0 + 0X2^1 + 0X2^2 + 0X2^3 + 0X2^4=1$. Similarly the binary number for the input and output

functions of each transition are determined.

Example:for the transition t_1 in fig. 2.1, the input function binary number is 1 and the output function binary number is $1X2^1 + 1X2^2 = 6$.

Thus the various 1's in the state number for each marking indicate which of the places in the given Petrinet contain a token. Similarly the 1's in the input number of transition indicate the enabling conditions and the 1's in the output number of a transition indicate which places get a token when the enabled transition fires.

To determine which of the transitions are enabled in a given marking and the resluting new markings, we do logical operations on the state numbers, input and output numbers of transitions.

1. To determine whether a transition is enabled in a given marking or not, we do the following logical operations:

(i) BitAND the state number and the input number of the transition.

(ii) Check if the result is equal to the input number of the transition. If so the transition is enabled in the given marking.

2. To determine the new marking by the firing of the enabled marking we do:

(i) INTRES = State number bitANDed with (complement of the input number).

(ii) The new marking= INTRES BitORed with output number of

the transition.

Appendix B contains the program listing for this method.

In binary method, in one word, upto 48 places can be accommodated. The drawback is multiple tokens cannot be represented by this method. So we adopted the hexadecimal method where for each place we allot 4 bits in the state number thus providing upto 15 tokens at each place.

Here we determine the state number to the base 16.

Example: In fig 2.1 for the places p_1, p_2, p_3, p_4 and p_5 we assign weightage $16^0, 16^1, 16^2, 16^3$ and 16^4 respectively. For the given initial marking shown in fig 2.1, the state number is given by $1X16^0 + 0X16^1 + 0X16^2 + 0X16^3 + 0X16^4 = 1$. The state number, in general, $S.N. = n_1X16^0 + n_2X16^1 + \ldots n_iX16^{i-1} \ldots N_nX16^{n-1}$ Where n_i = number of tokens in place p_i.

So for the given marking shown in fig2.1, the state number is 1.

The program listing for this method is given in Appendix C. Here one word can handle upto 12 places and this program can handle upto 36 places.

The determination of enabled transition for a given marking and the resulting new marking is essentially on the same lines as done in binary case. The steps followed are:

1) We find the marking in terms of no of tokens in each place from the state number using the subroutine FNDMRK.

2)The function ITRENA determines which of the transitions are enabled in a given marking.

3)The resulting marking obtained by the firing of an enabled marking is determined by the subroutine FNDNWST.

2.2.3 Applications:

Petrinets have been empolyed in modeling logic operations [HURA-81], proving correctness of concurrent processes [LAUTE-74] and modeling the logical behavior of computer systems [MARSAN-83]. Since Petrinets do not involve any timing considerations, it is difficult to model the various dynamic and time dependent aspects of computing systems for their evaluation. Various researchers have introduced the concept of time to marked petrinets.

Some have introduced a fixed time period within which an enabled transition must fire. This is the basis for Timed Petrinets introduced by Zuberek[ZUBER-80]. With the use of Timed petrinets, modeling of conflicts in a system is difficult because of the deterministic time interval. Therefore Molloy[MOLLOY-81] and Shapiro[SHAPI-79] introduced probabilistic random firing of the various transitions of a Petrinet.

2.3 Stochastic Petrinets(SPNs):

2.3.1 Preliminaries:

Stochastic petrinets were initially proposed by M.K.Molloy[MOLLOY-81]. A SPN, like a Petrinet, is associated with a finite set of places P, a finite set of transitions T, the set of input and output arcs A and an initial marking M. In addition a firing rate g_i is associated with each transition.

Definition 2.3.1: A Stochastic Petrinet structure is defined by the 5-tuple,

SPN = [P,T,A,M,G] .....(2.7)

where P,T,A are as defined in (2.1) and M as defined in (2.4) and G is the set of firing rates for the transitions of the SPN.

$G = \{g_1, g_2, \ldots g_m\}$ .....(2.8)

The Petrinet shown in fig2.1 will become a SPN once we associate the set of firing rates G to it. In this example G is a vector of 5 elements, each element corresponds to a transition and specifies the rate at which that particular transition will fire.

2.3.2 Properties of SPN:

The SPN has all the properties of the underlying Petrinet, (as discussed in section 2.2.1), like finite reachability set, liveness and boundedness.

Further SPN has certain other properties by virtue of the introduction of the probabilistic rates for the firing of each transiton in the SPN.

By associating a continuous distribution with each g_i, say an exponential distribution with its own mean, one obtains a Continuous time SPN. Similarly by associating, with each of the firing rate g_i, a discrete distribution like geometric distribution with its own mean, one obtains a Discrete time SPN.

The reasons for choosing the exponential or geometric distributions for the firing of the transitions in a SPN are discussed below.

A stochastic Process [KLEIN-75,MOLLOY-81] is a collection of random variables X(t) defined on a common probability space where t is the time variable and is the subset of the time range $(-\infty,+\infty)$. Given a set of states S and a stochastic process on the states S, the process has Markovian property if and only if the probability of occurence of a particular state as the next state depends solely on the present state. Thus for any sequence of states $x_1, x_2, \ldots x_n, x_{n+1}$ in the process,

$$P(X_{n+1}=x_{n+1} | X_1=x_1, X_2=x_2, \ldots X_n=x_n)$$
$$= P(X_{n+1}=x_{n+1} | X_n=x_n) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.9)$$

The only probabilistic distributions that have this "memoryless" property are the exponential and geometric

distributions[KLEIN-75].

Molloy has shown that a finite place, finite transition, marked SPN is isomorphic to a one dimensional discrete Markov chain. Also he has proved that there is one to one mapping between the markings of the SPN and the states of the Markov chain. The rate at which the transitions take place between the states of the equivalent Markov chain is equal to the sum of the rates of the enabled transitions which would give a marking equivalent to the next state. There are certain properties of SPN which are useful in the analysis of the system being modeled.

2.3.3 SPN Analysis:

In SPN analysis, as in Markov analysis, ergodic (recurrent, non-null, aperiodic ,irreducible) systems are of interest[HOEL-72] They have steady state probability distribution of tokens which can be determined after the equivalent Markov chain is obtained. To assure the ergodicity of the underlying Markov chain of the SPN, some of the properties of the Petrinets are used.Molloy has shown that in a live, bounded Petrinet with an initial marking M_o and a reachability set S, there exists a unique reachability set of recurrent markings S' ,possibly a subset of S,which entirely describes the steady state behavior of the Petrinet. Also any live, bounded continuous time SPN is ergodic or reducible to a single ergodic sub-chain and therefore has a solution for the steady state probabilities

for the tokens in each place of the SPN[MOLLOY-81].

Thus the SPN, which combines the modeling power of the Petrinets and the mathematical richness of discrete Markov chain, is a good tool for studying the performance of DCSs.

In SPN the dynamic behavior can be easily studied by constructing the reachability set of the SPN, by adopting the method for constructing the reachability set shown in section 2.2.2. In the process of determining the reachability set of the markings of the system, the basic properties of the associated Petrinet like liveness, boundedness are also determined.

The various markings in the reachability set of the SPN form the different states of the isomorphic continuous Markov chain. In the SPN, as mentioned earlier, the various transitions are associated with a mean exponential firing rate. Hence the sojourn time in each marking (state) is an exponentially distributed random variable with average,

$$\sum_{i \in K} (r_i)^{-1} \qquad (2.10)$$

where K is the set of transitions that are enabled by the marking. The transition rate from marking M_i to marking M_j is obtained as,

$$\sum_{l \in K_{ij}} r_l \qquad (2.11)$$

where k_{ij} is the set of transitions (which may contain only

one element in most of the cases) enabled by the marking M_i and whose firing generates the marking M_j. Thus the infinitestimal generator (or rate matrix) Q of the Markov chain can be obtained and the steady state probability distribution Y of the various states of the Markov chain can be determined by solving the equation,

$$YQ = 0 \text{ or } Q^T Y^T = 0 \quad \ldots\ldots(2.12)$$

Since the system is irreducible, the rank of Q is (n-1) and thus we have (n-1) independent equations in (2.12) [MOLLOY-81,RAU-81]. To make all the n equations independent so that (2.12) can be solved to obtain the steady state probabilities, we introduce as the total probability constraint i.e $\sum_{i \epsilon n} y_i = 1$, in place of the last dependent equation in (2.12). This makes the (2.12) to have the form AX=b which is solved easily, to obtain the steady state probabilities y_is.

After the determination of the steady state probabilities of each marking in the underlying Markov chain, the token densities at various places of the SPN are calculated. The token density d_i of a place p_i in a SPN is determined by the following relation.

$$d_i = \sum_{j \epsilon R(PN)} y_j . m_{ij} \quad \ldots\ldots(2.13)$$

Where y_j is the steady state probability of the marking j and m_{ij} is the number of tokens in place p_i in the marking j. The token densities of all places in a SPN are

determined using this relation (2.13).

2.3.4 Applications:

SPNs are suitable for modeling the various hardware/software aspects like contention for the shared resources (buses,memories etc), synchronization of tasks that influence the performance of a DCS [MARSAN-83].

The choice of firing rates of the varoius transitions depend on the workload assumptions and details of operations included in the model. For example, in a SPN modeling memory contention, the transitions modeling the request and access of the common memory fire as per the workload assumed. Other transitions, which represent the bus arbitration or release etc, whose effect is not that relavant in modeling the memory contention can fire at a very fast rate thus minimizing their effect on the performance measures of the SPN model.

Thus the countability of the markings and the memoryless property of the exponential (and geometric) distributions are the key factors which make the SPN amenable for modeling systems which exhibit concurrency and contention[MOLLOY-81].

The introduction of SPNs have established a link between two important classes of computer system models: the graph models and the probabilistic models[MARSAN-85a]. SPN, because of its graphical nature, provides an easy to use high level interface for the specifications of a model of

the system under study. Queueing theory had long been used for this purpose. The SPN has certain advantages over the queueing models[MARSAN-84]. First, they allow easy portrayal of concurrency and synchronization within the system, which are not easily described with queueing network models. Secondly, the SPNs allow the description of a system at different levels of abstraction, so that the user can choose the one that best suits his needs.

Also in the direct markov models, the determination of the various states of the system is very difficult and requires good working knowledge of the mathematics of Markov chains. The analyst using SPN can generate a proper and complete model of the system by virtue of the structure of the SPN.

Thus the SPNs can be easily used by people who are not very familiar with the probabilistic modeling approach. Further the whole procedure of determining the states in the Markov chain underlying the SPN and the solution to the steady state probabilities of the various states can be automated.

But SPN has one drawback. Even with the small increase in the complexity or detail in the modeled system, the resulting markov chain grows at a faster rate in its complexity. As a result, the numerical solution of the Markov states presents major computational problems.

As a starter, SPNs are ideal for modeling the system at a fairly high level of abstraction and to acertain the logical correctness of the model. In order to model the system with finer details and at the same time maintain the solution complexity of the underlaying Markov chain at a reasonable level, a derivative of SPN called Generalized Stochastic Petrinets(GSPN) has been proposed by Marsan et al[MARSAN-84].

2.4 Generalized Stochastic Petrinets(GSPNs)

2.4.1 Definitions:

In SPNs we associate a random firing time or rate with each transition. Often this is not desirable since one would want to associate time with those events which are belived to have the largest impact on the system performance. This is more so in a system comprising of activities differing in orders of magnitude. Then it is conceivable to model the brief activities only from a logical point of view, and associate time with the longer lasting ones. This choice becomes particularly convenient if by doing so the number of states of the associated Markov chain can also be reduced, thereby reducing the solution complexity. This is the main thrust behind the GSPN proposed by Marsan et al [MARSAN-84].

GSPNs are derived from the SPNs by partitioning the set of various transitions into two disjoint subsets comprising of immediate and timed transitions.

Definition2.4.1:Immediate transitions fire in zero time once they are enabled.

Definition2.4.2: Timed transitions fire after a random (exponentially distributed) firing time.

A GSPN has the same graphical structure as SPN and in addition the timed transitions are represented by thick bars and the immediate transitions by thin bars. The firing rates are obviously associated with the timed transitions

and they may also depend on the GSPN marking.

Definition2.4.3: The Generalized Stochastic Petrinet(GSPN) is defined by the five-tuple,

GSPN = [P,T,A,M,R] .....(2.14)

where P,T,A are as defined in (2.1) and M as in (2.4). R is exactly similar to G in (2.8) but here m = number of timed transitions in the GSPN.

As in SPN, in GSPN also several transitions may be simultaneously enabled by a marking. Let H be the set of enabled transitions in a given marking. If the set of enabled transitions H comprises only timed transitions, then the enabled timed transition $t_i (i \epsilon H)$ fires with a probability r_i/R.....(2.15)

where $R = \sum_{K \epsilon H} r_K$ exactly as in SPN.

If H comprises both immediate and timed transitions, then only immediate transitions can fire. If H comprises zero or more timed transitions and only one immediate transition, then this immediate transition is the one which fires. When H comprises of several immediate transitions, then it is necessary to specify a probability density function according to which the firing transition is selected. The subset of H comprising all enabled immediate transitions together with the associated probability distribution is called a "Random switch". The associated probability function is called a "switching" distribution. Different

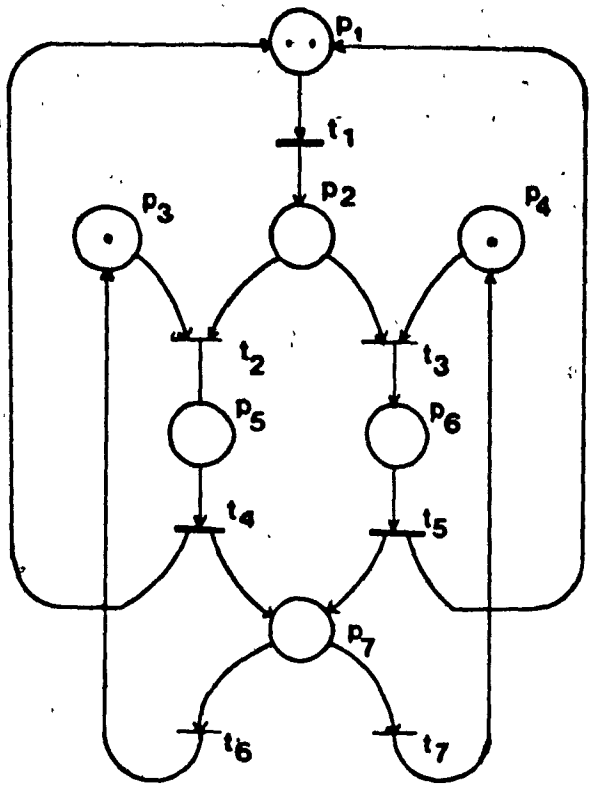
markings of the GSPN may orginate a single random switch whenever they enable the same set of immediate transitions, upon which a single (possibly marking dependent) switching distribution may be defined. If the probability associated with an enabled immediate transition is zero, the transition cannot fire and thus it behaves as if it were not enabled. This condition, in some cases can be modeled in a simpler way by means of "inhibitor arcs". An inhibitor arc connects an input place to a transition and is represented by a line terminating with a circle rather than with an arrowhead at the transition. Thus the definition for an enabled transition(Defintion2.2.6) can be generalized to include the inhibitor arcs.

Definition 2.4.4:A transition in a marked Petrinet is enabled if all of its "normal" places contain atleast as many tokens as the number of arcs connecting them to the transition and all the "inhibitor" input places contain no tokens.

Thus the inhibitor arcs help in testing for zero token in a particular place. The inclusion of inhibitor arcs, in certain cases, simplifies the GSPN by reducing the number of random switches to be defined.

2.4.2 An example of GSPN:

Fig 2.2 shows a GSPN comprising of seven places and seven transitions of which three transitions t_1, t_4 and t_5 are



Switching probabilities:

$Pr(t_2) = m_3/(m_3 + m_4)$,

$Pr(t_3) = m_4/(m_3 + m_4)$,

$Pr(t_6) = m_4/(m_3 + m_4)$,

$Pr(t_7) = m_3/(m_3 + m_4)$,

} if $m_3 \neq 0$ & $m_4 \neq 0$,

$Pr(t_6) = Pr(t_7) = 1/2$ if $m_3 = m_4 = 0$.

Figure 2.2 Example of a GSPN.

timed and the remaining four transitions t_2, t_3, t_6 and t_7 are immediate transitions. Transition t_1 fires at a marking dependent rate equal to u times the number of tokens in place p_1. Transitions t_4 and t_5 fire at fixed rates v and z respectively. Transitions t_6 and t_7 are two conflicting imeediate transitions, they are always enabled simultaneously so that it is necessary to define a switching distribution for each marking where m_7 is larger than zero. The immediate transitions t_2 and t_3 may be simultaneously enabled if P_3 and P_4 contain tokens. Here a switching distribution must be defined for each marking in which m_2, m_3 and m_4 are greater than zero. Thus two random switches are to be defined for this GSPN. One possible switching distribution definition is shown in fig 2.2.

For the initial marking shown for the GSPN in fig2.2 only the timed transition t_1 is enabled. After an exponentially distributed random time with an average 1/2u transition t_1 fires and one of the two tokens from P_1 is put in P_2. Now the immediate transitions t_2 and t_3 are enabled and any one can be selected to fire as per the switching distribution defined in fig 2.2.Here they are assigned equal probabilities. Assume that t_3 fires removing tokens from P_2 and P_4 and placing one in place P_6. Now two timed transitions t_1 and t_5 are enabled, transition t_1 fires with a probability u/(u+z),where as t_5 fires with the probability z/(u+z). Assume that t_1 fires first, one token moves from P_1 to P_2, thus enabling transition t_2. This transition

fires immediately, being the only enabled immediate transition, thus moving one token each from places P_2 and P_3 and putting one token in place P_5. The resulting GSPN marking is such that the timed transitions t_4 and t_5 are enabled , each of which can fire first with the following probabilities:

$Pr(t_4) = v/(v+z)$, $Pr(t_5) = z/(v+z)$.

Assume t_4 fires first, so that a token is moved from P_5 to P_7 and a token is put in place P_1. The two immediate transitions t_6 and t_7 are now simultaneously enabled and as specified by the switching distribution in fig2.2, each of them can fire with a probabilty 1/2, so that the token in place P_7 can move either to P_3 or P_4. Now the timed transitions t_1 and t_5 are enabled and this way the Petrinet execution continues.

In general, the reachability set of a GSPN is a subset of the reachability set of the associated Petrinet because the precedence rules introduced with the immediate transitions do not allow some states to be reached. The reachability set of a SPN is, instead, the same as for the assocaited Petrinet. The GSPN of fig 2.2 comprises of 17 markings, where as the reachability set of the associated Petrinet comprises of 33 markings. Further, the reachability set of the GSPN can be divided into two disjoint subsets, one of which comprises markings that enable timed transitions only, while the other comprises

markings that enable only immediate transitions.

The crucial part in the definition of GSPN of a system under modeling, is in the definition of random switches for the immediate transitions. Usually this requires some ingenuity and good insight into the functioning of the underlying system.

2.4.3 Evaluation of the GSPN markings steady state probability distribution:

If we observe the various state changes in a GSPN with respect to time, we may find "multiple discontinuities" in the system state changes as shown in fig 2.3.

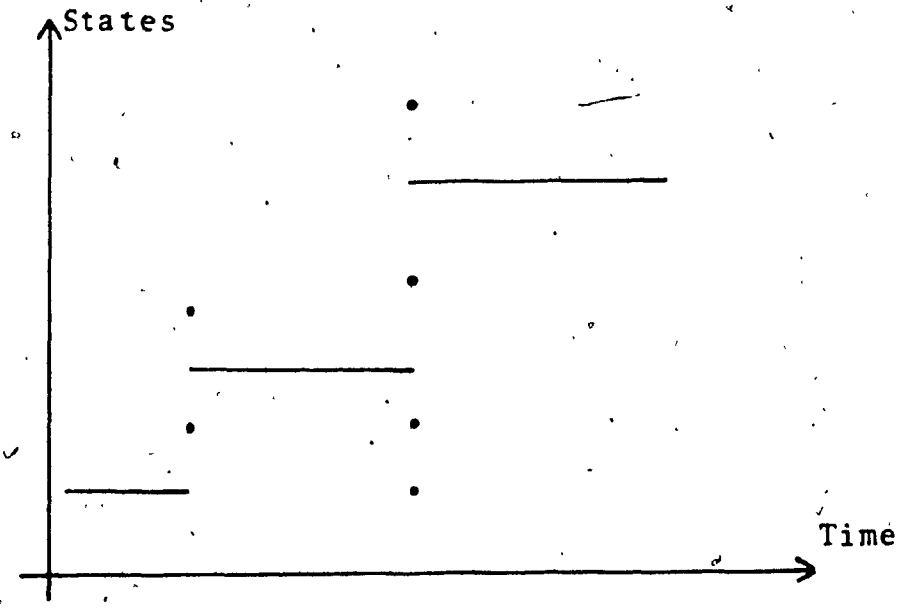


Figure 2.3 Time Behavior of GSPN

This is because of the sequential firing of one or more immediate transitions which are enabled in a particular marking. It should be noted that the stochastic process underlying the GSPN spends a nonnegative amount of time in those markings which enable timed transitions only; but it transits in zero time through markings which enable immediate transitions. This time behaviour of the GSPN is equivalent to the time behaviour of a Stochastic Point Process(SPP){X(t),t≥0 } with a finite state space. A one to one correspondence exists between the GSPN markings and the SPP state space[MARSAN-84].

The states or markings in which the GSPN spends non-negative amount of time are called "Tangible" markings and those markings in which it spends zero time are termed "Vanishing" markings.

Similar to the SPN, a GSPN with a finite set of places and finite set of transitions (including both the timed and immediate transitions) is ismorphic to a one dimensional Markov chain which in this case is an "Embedded Markov Chain"(EMC). The following properties and assumptions regarding the GSPN will help in fully categorizing the associated EMC[MARSAN-84]:

1) The GSPN has a finite reachability set which is usually a subset of the reachability set of the underlying Petrinet. Also there will not be any terminal (or absorbing) state in the reachability set of the GSPN due to the liveness

property of the underlying Petrinet. The finiteness of the reachability set is ensured by the boundedness of the underlaying Petrinet.

2) The initial marking is reachable with a nonzero probability from any marking in the reachability set. This is made possible by the liveness of the underlying Petrinet.

3) The firing rates for the timed transitions are not a function of time but are constants.

These points help further to specify the nature of the SPP, as finite, state spaced, stationary(homogeneous), irreducible, continuous time Stochastic Point Process. The associated EMC will be a finite state space, continuous time, irreducible stationary chain. In other words, the GSPN is ismorphic to an ergodic EMC whose states are equivalent to the various markings of the reachability set of the GSPN. This ergodic nature of the EMC ensures the existance of the limiting steady state probabilities of various states in the EMC and the calculations of steady state probabilities of tokens at each place of GSPN.

Let S denote the state space of the EMC and within S we can distinguish the Tangible and Vanishing states. This also helps in partitioning the state space and transition matrix. Let K_s denote the number of states in the EMC state space and K_t and K_v indicate the number of Tangible and Vanishing states such that

$K_s = K_t + K_v$.

The transition probability matrix U of the EMC can be written as follows:

$$U = A + B = \begin{bmatrix} C & D \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ E & F \end{bmatrix} \quad \text{.....(2.16)}$$

where C has dimension K_vXK_v, D has dimension K_vXk_t, E has dimension K_tXK_v and F has dimension K_tXK_t.

The elements of A can be obtained using the characteristics of the random switches and the elements of matrix B can be obtained using the firing rates of the timed transitions as given by the relation (2.15).

The solution of the system of linear eqautions $Y = YU$. (2.17) yields the steady state probability distribution of the states of the EMC[MARSAN-84]. The vector Y gives these steady state probability distribution of these states of the EMC.

This system of linear equations can be solved in the same manner as is done in SPN analysis, by assuming that the system remains in the vanishing state for a infintestimally small time instead of zero,i.e. we can associate a very high firing rate 'x' for the firing of the immediate transitions and obtain the transitions rate matrix. After we obtain the steady state probability vector, the steady state probability vector for the original GSPN is obtained by taking limits for 'x' going to infinity. This method of solution has the computational complexity same as the SPN

and since the GSPNs involve large state spaces, the computational complexity is rather large. Further this method does not make use of the state bifurcation into Tangible and Vanishing states.

Hence Marsan et al[MARSAN-84] have suggested a solution method which is computationally more efficient and also makes use of the tangible and vanishing state classification of the states of GSPN. This solution method is briefly described below:

Here a "Reduced Embedded Markov Chain"(REMC) is defined over the tangible states only. In order to do this it is required to compute the total transition probabilities among tangible states only. A tangible state may be reached from another tangible state either directly or through some vanishing states. These various possible ways must be reflected in the transition probabilities of the REMC. This is achived in the following manner.

Let i,j represent arbitrary tangible states, while r and s represent arbitrary vanishing states. C_{rs}, D_{rj}, E_{is} and F_{ij} represent the elements in the submatrices C,D,E and F respectively of the transition matrix U of the original EMC.The total transition probability between any two tangible states i and j can be computed in the following way:

$$U'_{ij} = f_{ij} + \sum_{r \in V} e_{ir} Pr(r \dashrightarrow j) \qquad (2.18)$$

where Pr(r-->j) represents the probability that the EMC moves from vanishing state r to the tangible state j in an arbitrary number steps, following a path through vanishing states only. Thus the total transition probability matrix for the REMC can be obtained and it is denoted by,

$$U' = F + EG^{\infty} \quad \ldots\ldots(2.19)$$

where G^{∞} is the explicit expression for Pr(r-->j) for the various vanishing states in the orginal EMC(and orginal transition matrix U).In order to determine G^{∞} we focus our attention on the matrix A whose elements represent the transition probabilities of movement between Vanishing states and Vanishing state to Tangible states. The Kth power of A represents the probability of moving from any vanishing state r to any other state of the original EMC in exactly K steps.

Hence $A^K = \begin{bmatrix} C^K & C^{K-1}.D \\ 0 & 0 \end{bmatrix}$ .....(2.20)

From this we have $G^K = \sum_{h=0}^{K-1} C^h D$ .....(2.21)

representing the probability of reaching any tangible state i moving from any vanishing state in no more than K steps, visiting intermediate vanishing states only.When K can vary to ∞ we have,

$$G^{\infty} = \sum_{h=0}^{\infty} C^h D \quad \ldots\ldots(2.22)$$

The irreducibility property of the EMC ensures that the spectral radius of the submatrix C is smaller than

one[VARGA-62].

Hence $\lim_{K\to\infty} C^K = 0$ .....(2.23)

With this result we can state that a limit for the sum, $\lim_{K\to\infty} \sum_{h=0}^{K} C^h$ exists and is finite.

Also the submatrix C can be written as an upper triangular matrix by suitable reordering of the states of the EMC, so that there exists a value $K_O \leq K_V$, such that $C^K=0$ for any $K \geq K_O$.

From these results the infinite sum for G reduces to a sum of finite terms and it is expressed as,

$$G^{\infty} = \left(\sum_{h=0}^{K_O} C^h\right) D \quad \ldots\ldots(2.24)$$

With the determination of G^{∞}, U' can be computed.

The solution of the linear equations $Y' = Y'U'$ ...(2.25) yields the steady state probability distribution Y' for the REMC.

The advantage of this solution method is two fold. First, the time and space complexity of the solution is reduced, since instead of solving a system of K_S linear equations we must now (in the worst case) find the inverse of $k_V X K_V$ matrix and then solve a system of equation of k_t linear equations. It is known that the complexity of the Gaussian ellimination solution is $O(k^3)$. Hence the solution approach proposed by Marsan et al reduces the complexity of the solution of the GSPN from $O(K_S{}^3)$ to $O(K_t{}^3) + O(K_V{}^3)$.

The other advantage is by decreasing the impact of the size of the set of vanishing states on the complexity of the solution method, a greater freedom is available in the explicit specification of the logical conditions of the original GSPN. This enables larger systems to be conveniently modelled with greater detail using GSPN.

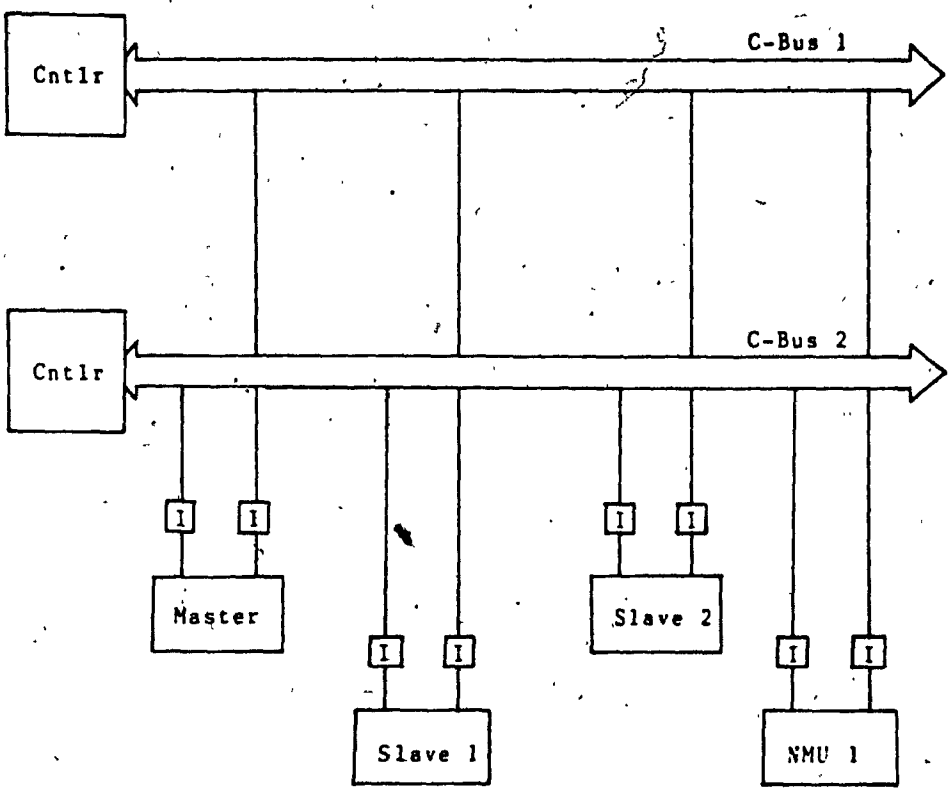
CHAPTER III

THE ARCHITECTURE OF CUENET

3.1 Hardware and Software aspects.

Concordia University Educational NETwork (CUENET) is a reconfigurable network of loosely coupled multiple micro computers which are interconnected by one or more buses called C-Buses [GROSS-82]. Its network configuration is as shown in fig 3.1.Each C-Bus is controlled by a C-Bus controller which is a processor dedicated for controlling the transfer of messages over the C-Bus. Each processor in the network connects to a C-Bus through an interface called C-Bus Interface. If a processor is connected to more than one C-Bus then it will have more than one C-Bus interface, one for each C-Bus to which it is connected.

CUENET comprises of a master computer, several slave computers and network memory units(NMUs). The master computer is responsible for the coordination of all the computers in CUENET and also acts as the interface between the network and the end user. The computational tasks required by the end users are carried out by the slave computers. The slaves will accept and perform the commands of the master such as load a user program, obtain data/user algorithm from some other location in the network, and start or terminate a user routine. The network memory unit is accessible to all the computers of CUENET,as a common data



I : C-Bus Interface
NMU : Network_Memory Unit
Cntlr : C-Bus Controller

Figure 3.1 Configuration of CUENET

bank, through the C-Bus.

The C-Bus is a bit parallel, byte serial passive bus over which the CUENET processors communicate by passing messages under the control of the C-Bus controller. The C-Bus controller is responsible for resolving the contention for the use of C-Bus and ensuring the safe transfer of message to the receiver. In the C-Bus controller some special control function like incrementing the bus address lines, parity checking are implemented using special purpose hardware so that the C-Bus controller can be implemented using a general purpose microcomputer. Also this simplifies the Bus control software and aids in quicker transfer of message.

In CUENET the various slave computers need not be homogeneous. Infact a slave processor can be a special purpose processor such as a processor for signal processing etc. This hetrogeneity allows that all processing elements in the CUENET need not be of the same power and capacity. It is possible to select the type and configuration of the processing elements so that they match as closely as possible to the user application requirements.

Further the master processor does not contain any special purpose hardware and thus any computer in the network can function as the master as long as its internal configuration is capable of running the master processor

software. This will enable quick restart of the network in the event of a failure of the master computer. In CUENET the master computer differs from the slave processors in having certain special rights. One such is loading the user's programs into slave processors and the other is setting up the configuration of the network as required by the user. In CUENET the reconfiguration of the network is variable under program control.This is made possible by providing an access vector at each processor which indicates what are the processors this processor can communicate with. Except the master processor the slave processors can only read this access vector but not alter it. Master processor can alter the contents of the access vector by sending a special message over the C-Bus thus reconfiguring the network.

Besides this each processor in the network has an interface lock control unit which prevents any accidental or unauthorized access to the C-Bus interface area. In the C-Bus interface there are two registers called Lock and Key registers. The contents of the Lock register is set to a predetermined value (combination). Access to the interface is allowed only when the communication software writes the exact combination into the Key register. Only system programs but not the application programs are supposed to know the "password"(key) combination.This hardware lock mechanism increases the reliability of the application

software by preventing any unintentional erroneous messages to other processors in the event of a program crash or malfunction. Further this lock mechanism allows general purpose microcomputers to be employed as processors in CUENET.

Thus in CUENET the system reliability requirements are amply met by the provision of multiple C-Buses and special purpose hardware for access control and message transfer and parity checking, at the same time capitalizing on the advantages of centralized control, viz, hardware and software simplicity and system flexibility to use general purpose microcompurters.

Presently the CUENET implementation comprises of a single C-Bus on to which three slave MC 6809 processors and one MC 68000 master processor are connected. The C-Bus controller is implemented using another MC 6809 processor. The C-Bus, in the present implementation, has length of about 100 feet, runs around our laboratory connecting the three 6809 slaves and 68000 master. It comprises 32 address lines, 8 control lines and 9 data/parity lines-a total of 49 signal lines (twisted pairs).

3.2 Message Communication :

In the CUENET the various processors interact with each other by passing messages over the C-Bus under the control of the C-Bus controller.The C-Bus interface of each

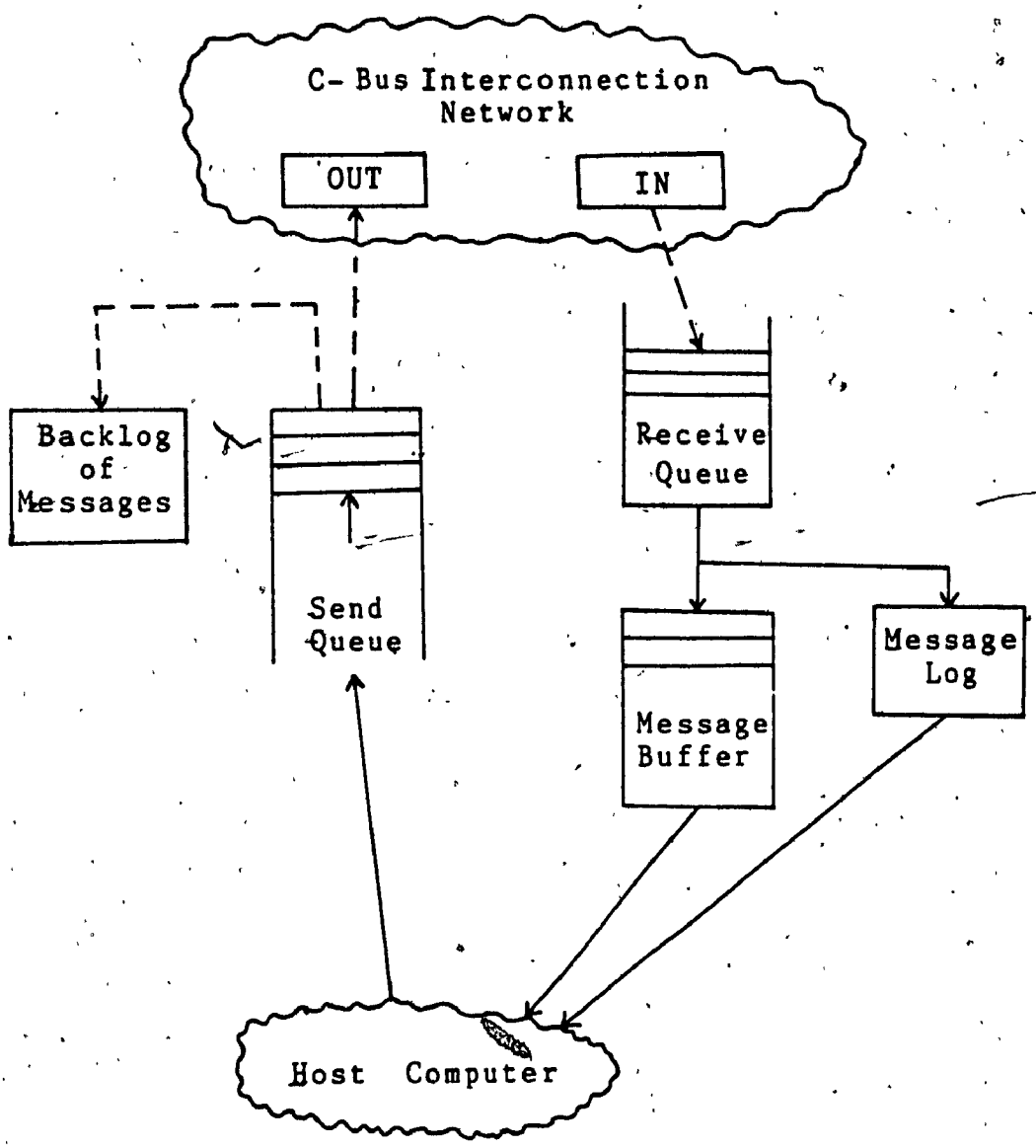
processor has two communication buffers- an output buffer and an input buffer. These buffers are accessable by the individual processors and the C-Bus controller. The output buffer in the C-Bus interface holds a message which is destined to another processor. The input buffer holds a message which is sent to this processor by the other processors of the network. Apart from this, each processor has resident communication software which is executed by the processor for either sending or receiving a message. The special purpose hardware in each of the C-Bus interfaces and message communication software in conjunction with the hardware and software of C-Bus controller perform the message communication in CUENET. A message communication scenario in CUENET is described below:

The sender processor puts the message it wants to send in the output buffer of its C-Bus interface and draws the attention of the C-Bus controller. The C-Bus controller examines the output buffer of the sender to determine the receiver address and finds if the input buffer of the specified receiver is free. If it is free, the C-Bus controller deposits the message into the receiver's input buffer and marks the output buffer of the sender empty and the input buffer of the receiver full. In CUENET because of the loose coupling of the processors ,the sending processor can be active working on the application program in its memory while the C-Bus controller is transmitting the message. Likewise,the receive processor can be operating on

its program till it gets the signal that it has received a message. This type of message communication enhances the parallelism of the distributed processing system.

In CUENET the messages sent by various processors during the execution of the user's application programs follows a fixed format. The message format consists of a header, body of the message and footer. The header contains information like sender address, receiver address, type of message, message length and time of sending the message etc. The body of the message can vary in length but should let the total message fit into the capacity of the output and input buffers in the C-Bus interface. The footer contains checksum which aids in the determination of the acuracy of the transmitted message.

At each processor the C-Bus interface supports an interrupt driven message system. An interrupt request is issued when the output buffer is empty or when the input buffer is full. The message control mechanism adopted for the message communication over the C-Bus, at each processor, is as shown in fig.3.2.Further this interrupt driven message system ensures that the output and input buffers, which form the critical resource in the message communication mechanism of CUENET are serviced as quickly as possible so that they are available for passing messages more effectively.



- - - : Upon interrupt request

Figure 3.2 Message Control Mechanism in CUENET

The message communication software, which supports this control mechanism, consists of an interrupt handler, a message transmit module and a message receive module. When a user task requests a message transfer, it invokes the message transmit routine and supplies the address of the receiver(LPN), address of the starting location of the message and the message length. The message transmit module performs the following operations:

1.It determines the physical address of the receiver using the given LPN of the receiver and the access vector. Also it checks whether this processor is allowed access to the receiver processor.

2. If the message to be transmitted is longer than the output buffer size, it divides the message into fragments so that each fragment can fit into the buffer.

3. Then it transfers this message from the memory location indicated by the sender to the transmit message queue.Also it puts the relevent information for forming the header of the message in a header table.

The message receive module is invoked whenever there is a request for receiving a message from another processor by the user's application program. It performs the following operations:

1. It determines if the message is a retransmission request or information for other software process. If it is a retransmission request, it checks the backlog of messages to

see if the required message can be found. If so it places it in the send queue with the destination marked. If the message is not found then it sends an error message to the master computer.

2.If the message is not a retransmission request, then it verifies the check sum. If error is detected it places a request for retransmission in the send queue. If no error is found, it removes the message header/footer and places the message body into the decoded message queue. Then it places an entry into the message log into which user process looks in when it is waiting for a message.

The interrupt handler is activated when either the input buffer becomes full or the output buffer becomes empty. The various operations it performs are:

1. Determines the nature of the interrupt, whether it is due to an empty output buffer or full input buffer.
2. If it is due to an empty output buffer, it moves the next message from the send queue to the output buffer and alters the status register to indicate to the C-Bus controller that a message is waiting for transmission. Also it places a copy of the message in the backlog of messages.
3. If the interrupt is due to a full input buffer, it moves the message to the receive message queue and alters the status register to indicate that the input buffer is empty. This will indicate to the C-Bus controller that it can perform the next message transfer to that receiver.

The role of the C-Bus controller in the message transmission amongst the processors in the CUENET is the most important one. The C-Bus controller has the overall control of the C-Bus and also has the responsibility of safely delivering the message to the receiver from the sender. Besides it should also resolve the contention for the use of the C-Bus if more than one processor requires to send message at the same time. The C-Bus controller employs the daisy chain scheme for resolving the contention for C-Bus. That is, if two processors request the use of the C-Bus for message transmission, then the processor which is electrically nearer to the C-Bus controller gets the use of the C-Bus. Only when this electrically nearer processor does not have any message to transmit, will the processor next to it get the chance to transmit. At each interface there is a mask bit under the control of the C-Bus controller using which the C-Bus controller can alter the priority of that processor for transmitting a message.

The interaction between the C-Bus controller and the various processors starts with the C-Bus controller activating the "Bus grant" line. The processor first in the line in the daisy chain will respond if it has a message to transmit and its mask bit is not set. In that case it prevents further propagation of the Bus grant signal and activates the Bus grant acknowledge signal and also places its address on the interprocessor data bus. The C-Bus

controller then reads the header of the output buffer of the sender and determines the receiver's address. The C-Bus controller determines whether the receiver's input buffer is free. If so. it will initiate the message transfer in bit parallel and byte serial fashion. At the end of the message transmission the C-Bus controller alters the status register of the sender to indicate that its output buffer is empty and the status register of the receiver to indicate that its input buffer is full. This will enable the interrupt to be raised which calls the attention of the particular processor's communication software.

If on other hand, the C-Bus controller determines that the receiver input buffer is full, it means that the receiver is not ready to receive this message. So the C-Bus controller sets the mask bit in the sender C-Bus interface so that the sender now will propagate the Bus grant signal down the daisy chain. The C-Bus software stores this sender's address in the stack for further processing at a later stage. This masking achives two things. It enables better utilization of the C-Bus and C-Bus controller by enabling the other processors, whose destination input buffers are free, to send their messages. Also this mask bit ensures that one high priority processor will not "hogg" the C-Bus all the time. The C-Bus controller after initiating the transfer by setting up the various hardware register pointers for the transfer of the message from the

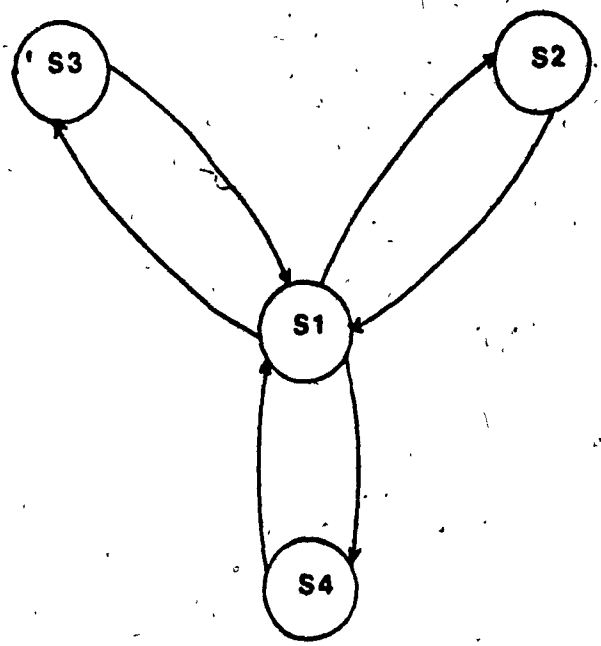
sender's output buffer to the receiver's input buffer, monitors the transmission by checking the parity of the transmitted word. If an error is found, it reinitiates the transfer and after certain number of repeated trials, the C-Bus controller will initiate the master computer for taking corrective action.

The C-Bus controller software, when idle, will empty the stack and unmask the mask bit in the various status registers of the processors whose addresses were in the stack. Then these processors can again ask for the services of the C-Bus controller for transmission of the message in their output buffer.

3.3 Performance measures :

The performance of a DCS is influenced by the way the various processors interact with each other, the characteristics of the interconnection network and the type of workload. The performance measures are the parameters of the DCS with respect to which the performance is assessed.

In CUENET the various processors communicate only by sending/receiving messages over the C-Bus under the control of the C-Bus controller. So a processor in the CUENET can be executing on the application program resident in its memory, be in the process of sending a message over the C-Bus or be in the process of receiving a message. Fig 3.3 gives a state diagram representation of these states. As discussed in section 1.3.1, the performance measures



S1- Processor Active on the task

S2- Processor in the process of sending a message

S3- Processor in the process of receiving a message

S4- Processor idle after completing the task

Figure 3.3 State diagram of a CUENET Processor

considered are processing power, interconnection network throughput and processor wait time.

The processing power of CUENET is the average number of processors which are busy operating on the aplication program (state S1 in fig 3.3). Thus, using relation 1.1, the processing power of CUENET can be expressed in the following manner:

Processing power = $E(N_1)$.....(3.1)
where N_1 is the number of processors in state S1 (see fig 3.3).

The throughput of C-Bus is another important measure of performance. This will indicate how quickly the C-Bus is transmitting the messages and how quickly they are consumed at the various destinations. The C-Bus throughput, using relation 1.2, can be expressed in the following way:
C-Bus throughput = Fraction of time C-Bus is busy X Rate of message transmission on C-Bus.....(3.2)

The amount of time a processor spends in participating in the message communication in CUENET gives an indication how well the system is suitable for running the particular application. This also includes the wait time if the message queue are full or the message has not arrived.

3.4 CUENET Applications:

The various applications that can be run on a distributed processing system depends on the way it can be partitioned so that it can efficiently be run on the distributed system. Further each application may require different interconnection structure for their interaction.

Since the optimal partitioning of a user job for the purpose of concurrent execution is a hard problem, one would want to experiment with different job partitioning and interconnection strategy before arriving at one which best

suites his application. In such cases the flexibility of the interconnection network architecture plays an important role.Thus CUENET is well suited for this purpose.

Some of the applications investigated for CUENET are the quick sort [TAMIR-83] and the numerical computation problems like the Dirichlet problem.

In quick sort the data to be sorted is partitioned equally onto the various processors and the final sort file is obtained either by concatenation or by performing k-way merge sort on these sorted subfiles.Here a star type interconnection architecture is used since the slave processors interact only with the master processor.The quick sort was tried using 2 and 3 processors in the system.

The Dirichlet problem is solved using parallel, synchronous iterative algorithm. Here the entire problem grid is subdivided into n- equal sub regions (n is the number of processors) and each sub region is allotted to a processor. Each processor computes the values for the next time step using Gauss-Seidel iterative method. At the end of each iteration the values of bordering points are communicated to the neighbouring processors. This commupication technique simulates a grid type (n.n) interconnection, as each processor send and receive data only from its neighbours. The above algorithm has been implemented on 2 processor system and further investigations

are being made to implement this on higher number of processors.

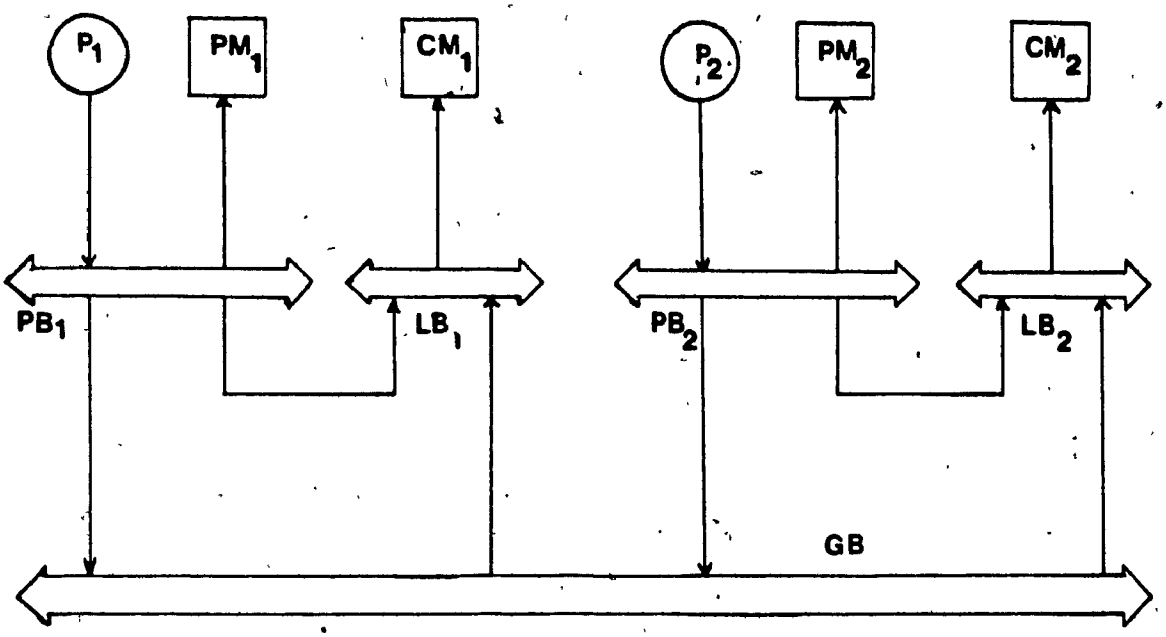
Another feasible application is to implement an office information system[GROSS-82]. Here each processor function as a workstation of an office and interstation message communication will be carried over the C-Bus.

The parallel processing power of CUENET can be utilized for carrying out linear predictive analysis of speech signals. Since this analysis has to be carried out in real time a pipeline interconnection architecture could be used which can enhance the speed of processing.

These are some of the representative applications of CUENET.

3.5 TOMP- TOrino MultiProcessor:

Different types of distributed systems are designed in several educational institutions to mainly study the various aspects of design, implementation and performance of distributed systems. Both loosely coupled and tightly coupled systems have been built.CUENET is a loosely coupled distributed system. TOMP[MARSAN-83] developed at Politechnico of Torino, Torino, Italy is a tightly coupled multiprocessor system. This system is designed around 16 Bit microprocessors and the prototype version comprises of three Zilog Z8001 processors tightly coupled via a global bus and with shared memory. Fig 3.4 shows the two processor



P_1,P_2 - Processors,

PM_1,PM_2 - Private Memories, CM_1,CM_2 - Common Memories,

LB_1,LB_2 - Local Buses, PB_1,PB_2 - Processor Buses.

GB- Global Bus

Figure 3.4 Two Processor Architecture of TOMP

architecture of TOMP.

The Global bus called M3BUS is a 30 lines parallel time multiplexed bus in which the address, data and status information are multiplexed [CONTI-81].The exchange of information among various processors is done on a message passing basis. Each processor, requiring to send a message to another, acquires the control of the M3BUS and the

destination processor common memory and deposits the message in that common memory.In TOMP the sending processor is actively involved throughout the message transmission process.In TOMP a typical message transfer to common memory areas consists of three phases:the bus arbitration phase, selection of common memory phase and the final actual data transfer phase. The control mechanism for the bus arbitration is distributed among the processors of TOMP.

CUENET and TOMP illustrate the diverse design techniques that can be exercised in designing a distributed porcessing system.The two systems differ in many aspects, the fundamental one being the nature of coupling. Further CUENET employs a centrally controlled message communication scheme where as in TOMP the M3BUS global bus arbitration and control is decentralized. The C-Bus is a non-multiplexed parallel bus as compared to M3BUS which is time multiplexed parallel bus.In CUENET one can use a mixture of 8 and 16 bit processors where as TOMP is predominently designed for 16 bit processors.

The common feature in both CUENET and TOMP is that both have made use of single board microprocessor system and have added some specialized hardwares to realize functions which are specific to their architectures.

In the performance study of CUENET the graphical modeling techniques SPN and GSPN have been employed. Since

these techiques were also applied in the performance studies of TOMP, we have chosen TOMP for comparing the results of the modeling. From these comparisions we can learn about the versatility of these graphical modeling techniques.

CHAPTER IV

STOCHASTIC PETRINET(SPN) MODELING OF CUENET.

4.1 Modeling Assumptions:

Here we are concerned with the intercommunication aspects of CUENET at the message passing level, in order to study how the input and output buffers of the different computers of the DCS and the characteristics of the C-Bus controller affect the overall processing power of the DCS.

The following assumptions are made regarding the functioning of CUENET:

1) We can clearly identify two periods of activity in each computer of the DCS, a processing period and a message transfer period which alternate.

2) The processing period for each processor is asssumed to be an exponentially distributed random variable with a mean rate λ. All processors of CUENET are assumed to process at this mean rate.

3) The message transfer period for the C-Bus controller is also assumed to be an exponentially distributed random variable but with mean rate μ.

4) At the end of a processing period each processor in the DCS may either have a message for transmission or might want to receive a message. Thus for each processor the message generation and message reception have equal probability of ocuurence.

5) The processor can send a message to any processor(including itself) with equiprobability i.e. if there are N processors in the system, the probability of sending or receiving a message to or from a processor is 1/N.

6) For simplifying the modeling, it is further assumed that each processor in the CUENET is allocated one task only.

7) When a processor needs to send a message, if the required buffers, queues and C-Bus are not free , it idles till these resources are available. Similarly when the processor needs to receive a message and its input buffer(queue) does contain it, then also the processor idles till it receives the required message.

8) The time required for determining the presence of message in the output buffer and checking the availability of the destination input buffer are assumed to be negligibly small when compared to the message transfer time and the processing period. Similarly the release of the output buffer and C-Bus at the end of a message transfer and indication of the destination input buffer being full are assumed to take negligible amount of time.

The assumptions 7 and 8 help in reducing the complexity of the SPN model. The assumptions of exponential distributions for the message generation and message transfer provide Markovian properities to the SPN model and makes the solution of the underlying Markov chain feasiable.

4.2 Estimation of system parameters:

The mean rates λ and μ are said to be the system parameters of the SPN model of CUENET. Then the system load factor ρ is given by the ratio (λ/μ). The performance of the system varies with the system's load factor ρ. To study this variation in our SPN model we need some representative values for λ and μ.

The values of μ depends on the characteristics of the communication subnet of CUENET. The rate at which the C-Bus controller transfers the message on the C-Bus depends on the speed of the C-bus controller software and the message communication software at each of the processor of CUENET. Let us assume a clock rate of 1 MHz for the C-bus controller and the various processors of the CUENET. The mean length of messages is assumed to be 256 bytes.

The approximate time required for the various stages in the message communication in CUENET are as given below:

(a) The table set up and update time required for the message transmit and interrupt handler part of the message communication software at the sending processor is about 600 μsecs.

(b) Transfer time for the transmit interrupt handler to move the message from the send queue to the output buffer is 60 μsecs/byte.

(c) Overhead time incurred by the C-Bus controller software is 300 μsecs.

(d) Transfer from the sender's output buffer to the receiver's input buffer by the C-Bus controller is 9 μsecs/byte.

(e) The table look up and update overheads for the message receive and interrupt handler part of the message communication software at the receiver requires about 800 μsecs.

(f) The transfer time for the receive interrupt handler to move the message from the input buffer to the receive queue is 15 μsecs/byte.

In order to simplify the SPN modeling of the message communication of CUENET, all these various transfer and processing times are combined together and we get a total message transfer time of 23.6 msecs which gives a rate of 41.66 messages per second.

The values of λs depend on the rate at which message are generated and this, in reality, depends on the type of application that is run on the system. For the sake our study we assume different values for the load factor ρ and thus determine the values for λs.

4.3 Two processor SPN model:

The SPN model(SPN2A) in fig4.1 models the physical interaction between the two processors PE_1 and PE_2 and the C-Bus controller during message communication on the C-Bus.In order to simplify the model, in the first attempt,

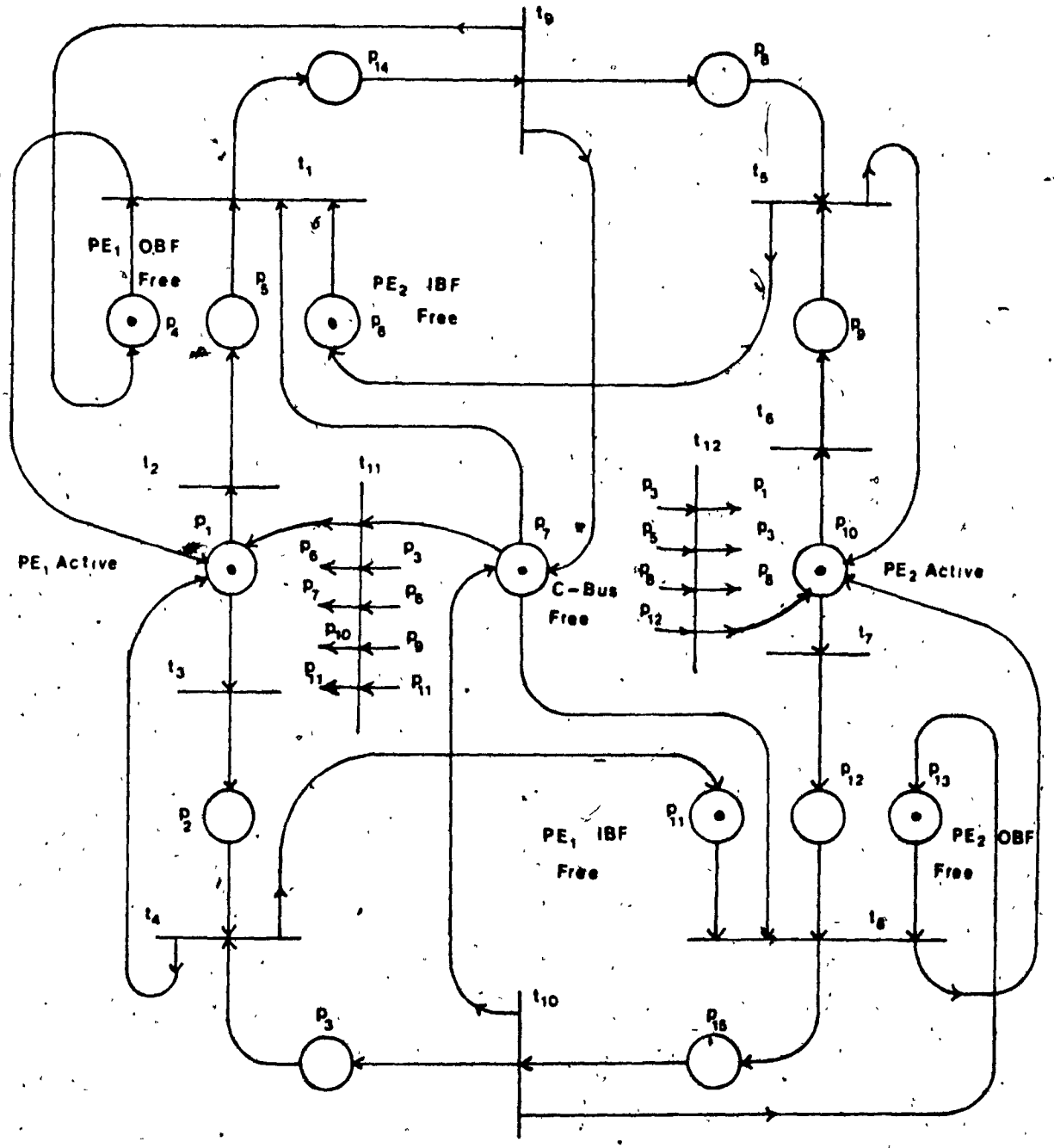


Figure 4.1 Two Processor SPN model -SPN2A

we have assumed that there are no send and receive queues. The two processors are assumed to have equal priority in using the C-Bus for message transmission. However, since the C-Bus controller employs a daisy chain for bus arbitration, there is a built in priority scheme.

The SPN2A has 15 places and 12 transitions. For modeling purpose, we assume that at the start both the processors are "active", meaning they are working on the task, allotted to them, which are present in their local memory. Also the initial condition is that the C-Bus and output and input buffers of both the processors are free. This initial state is indicated by the token distribution of SPN2A in fig 4.1. Tokens in places p_1,p_4 and p_{11} indicate that processor PE_1 is active and its output and input buffers are free. Similarly tokens in places p_{10}, p_{13} and p_6 indicate that processor PE_2 is active and its output and input buffers are free. The token in place p_7 denotes that the C-Bus and C-Bus controller are free.

Let assume that PE_1 needs to send a message to PE_2. This is modeled by the firing of the enabled transition t_2, putting a token in place p_5. This enables transition t_1, which models both the message transfer to the output buffer by the transmit part of PE_1's interrupt handler and the C-Bus controller operation of checking the availability of the receiver's input buffer and then initiating the message transfer. The firing of the transition t_1 puts a token in

places p_1 and p_{14} indicating that now the processor PE_1 can go on with its work while the message transmission is in progress. The token in place p_{14} enables the transition t_9, the firing of which denotes the actual message transfer by the C-Bus controller. The firing of this transition puts tokens in places p_4, p_7 and p_8 indicating that PE_1 output buffer and C-Bus are free and message is now present in the input buffer of PE_2. The firing of transition t_6 models PE_2 needing to receive a message, by putting a token in place p_9. The tokens in places p_8 and p_9 enables the transition t_5 which models the receive part of the interrupt handler of PE_2. The firing of t_5 indicates the receipt of message by PE_2 sent by processor PE_1. This completes one message communication cycle. Similarly the message transimission from processor PE_2 to processor PE_1 is modeled by transitions t_7,t_8 and t_{10} and the reception of this message by PE_1 is modeled by t_3 and t_4.

The transitions t_{11} and t_{12} are included to take care of certain situations which may cause the system to deadlock. In the real system, these sitautions may be handled by the operating system on each processor. One deadlock situation is when the processors PE_1 and PE_2 want to receive messages i.e. places p_2 and p_9 have tokens but neither of the processors have transmitted messages i.e. places p_3 and p_8 do not have any token. Transition t_{11} resolves this sitautions in the SPN model, by putting both processors into

active state.

Another deadlock situation ocuurs when both processors want to transmit a message while their destination input buffers are not free. Transition t_{12} detects this situation and resolves it by putting the two processors into active state at the same time the tokens in places p_3 and p_8 are retained.

The modeling assumptions discussed in section 4.1 determine the rates at which the various transitions fire in a given SPN model. The transitions t_2, t_3, t_6 and t_7, which model the processors generation reception of messages fire at the rate $\lambda/2$. The transitions t_9 and t_{10}, which model the transmission of message by the C-Bus controller, fire at the rate μ. The transitions $t_1, t_8, t_4, t_5, t_{11}$ and t_{12}, which model the activities like determination of the availability of the C-Bus, receiver's input buffer, receipt of the messsage by the destination processor etc, fire at a very high rate, so as to minimize their effect on the system parameters and performance measures.

SPN2A will give a pessimistic estimate of the processing power. This is because the transition t_1 (t_8) model the C-Bus controller selection function and, the message transfer to the output buffer by the sender processor's interrupt handler. Consider the enabling conditions for transition t_1. It cannot fire till all its input places p_4, p_5, p_6 and

p_7 contain a token. Thus in a situation, where p_4,p_5 and p_7 have tokens and p_6 does not have a token which indiactes that the input buffer of PE_2 is not free, the processor PE_1 is held in wait state, till the time PE_2 receives the message in the input buffer and frees it. But in actual case this does not happen, since the processor PE_1 just checks if its output buffer is free and if so puts the message in it and returns to its active state.

In order to overcome this drawback we modified this SPN to model these actions seperately.

4.4 Modified two processor SPN model:

The SPN model(SPN2B) shown in fig4.2, models the C-Bus controller selection function in more detail, by using an additional place and an additional transition for each processor. Thus it has 17 places and 14 transitions. The transitions t_1 and t_2 model the selection function that was modeled by t_1 in SPN2A. Here transition t_2 models the message movement to the output buffer by processor PE_1's interrupt handler and transition t_1 models the C-Bus controller selection and message transfer initiation. Similarly the transitions t_{10} and t_{11} model the functions modeled by the transition t_8. Transition t_2 gets enabled when the places p_3 and p_4 have tokens i.e. PE_1 has generated a message and its output buffer is free. The transition t_2 fires by putting a token in places p_1 and p_5 i.e. the message is in its output buffer and PE_1 returns to active

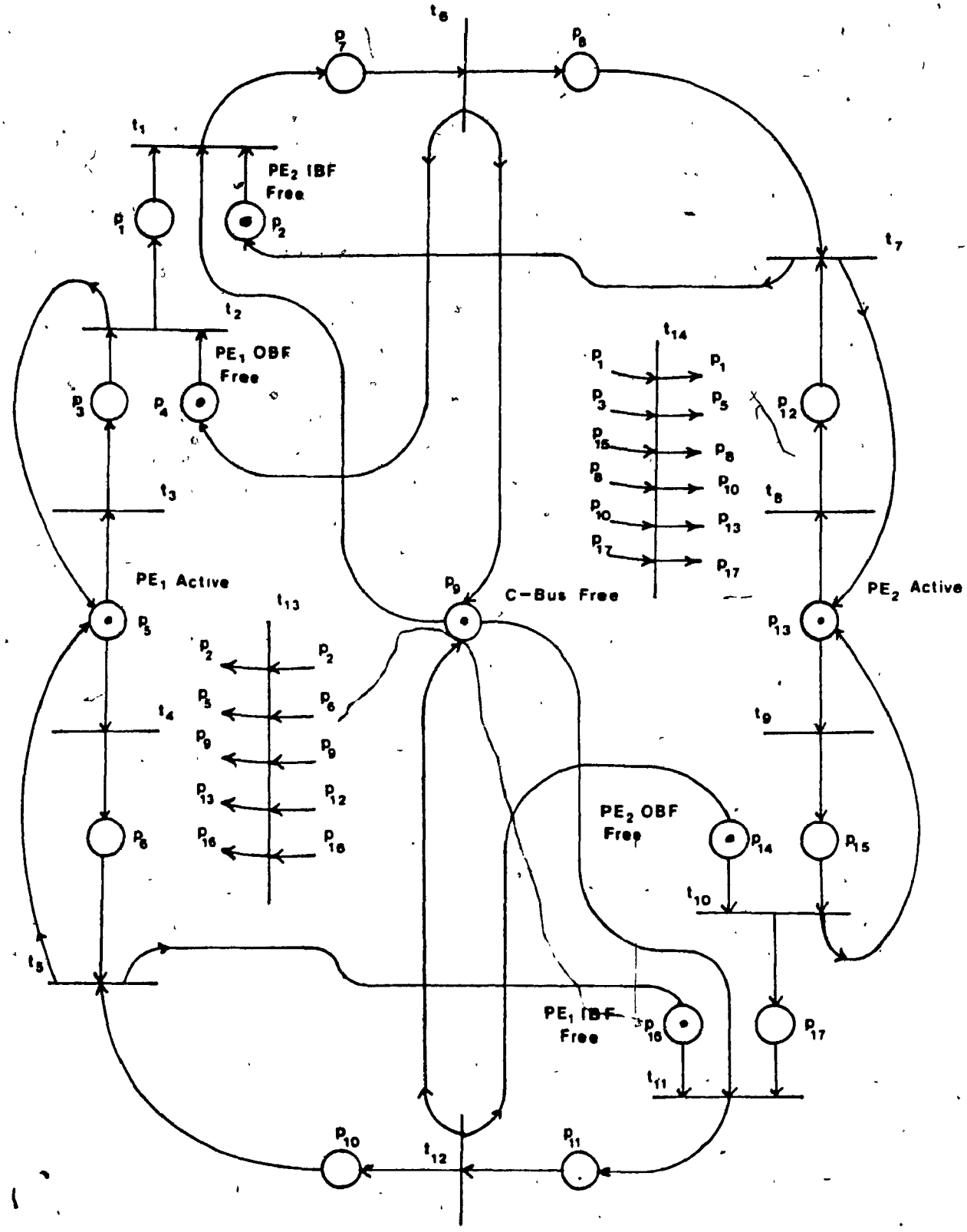


Figure 4.2 Modified Two Processor SPN model - SPN2B

state. The transition t_1 gets enabled only when there are tokens in places p_1, p_2 and p_9 i.e. a message is in PE_1 output buffer, PE_2 input buffer and C-Bus are free. The other part of message communication is modeled in the same way as done in SPN2A. This improvement models the concurrency in the system more closely.

The transitions t_3, t_2, t_1 and t_6 model the message transmission from PE_1 to PE_2 while the message transmission from PE_2 to PE_1 is modeled by transitions t_9, t_{10}, t_{11} and t_{12}.

The message reception by PE_1 is modeled by transitions t_4 and t_5, while transitions t_8 and t_7 model the message reception by PE_2.

Transitions t_{13} and t_{14} are similar to transitions t_{11} and t_{12} of SPN2A, for resolving the two deadlock situations.

In SPN2B, similar to SPN2A, the transitions t_3, t_4, t_8 and t_9 fire at the rate $\lambda/2$. The transitions t_6, t_{12} modeling message transmission fire at the rate μ. The rest of the transitions modeling availability of the output buffer, receiver input buffer etc. fire at a very high rate.

4.5 Three Processor SPN Model:

Fig4.3 shows the three processor SPN model of CUENET(SPN3A). SPN3A is on the same lines as SPN2A shown in fig4.1.

In this SPN model also, for the sake of simplifying the model, we assume that there are no transmit and receive

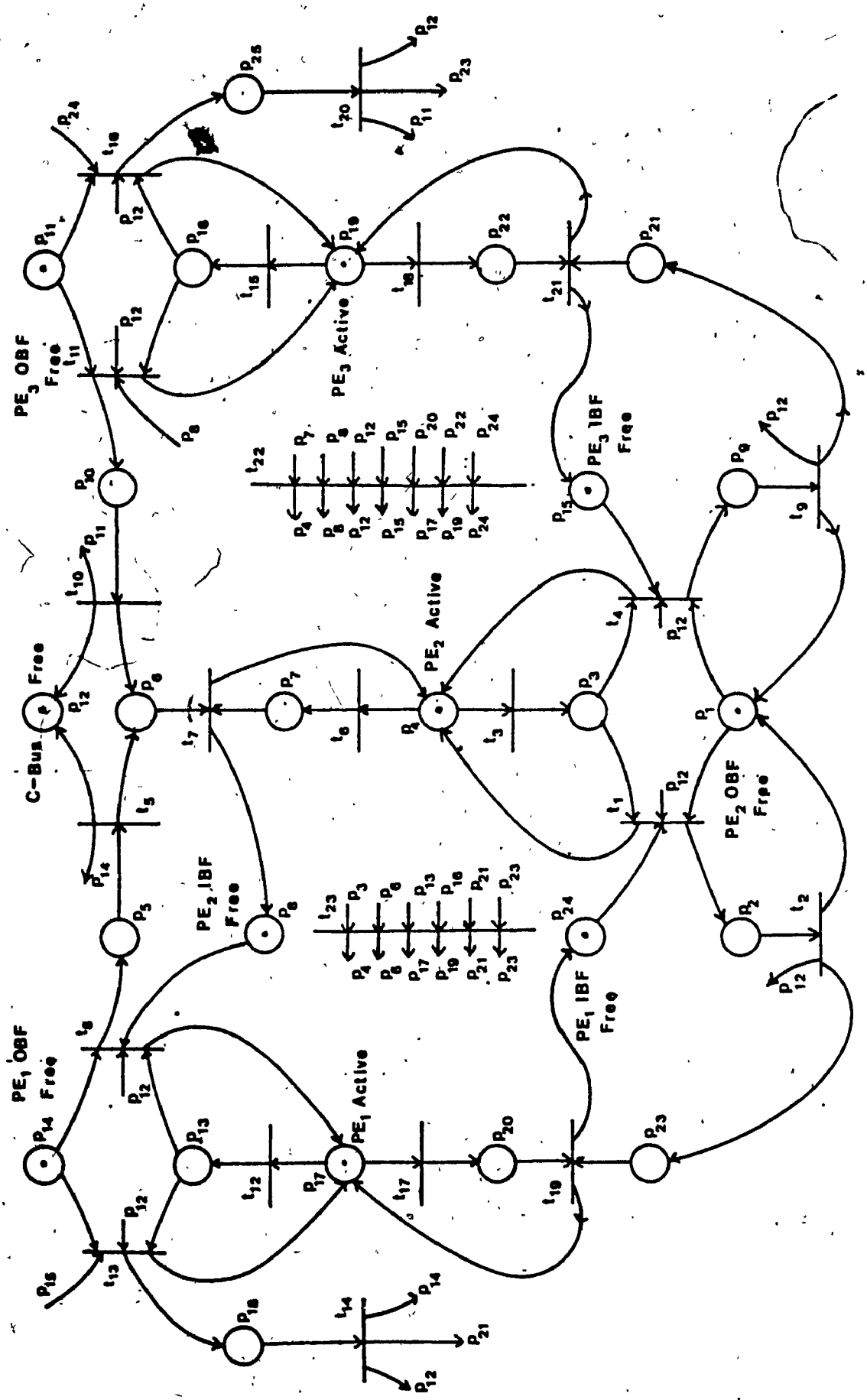


Figure 4.3 Three Processor SPN model - SPN3A

message queues at each processor. Each processor has equal priority in using the C-Bus and C-Bus controller for sending a message. Also each processor sends messages to the other with equal probability.

SPN3A comprises of 25 places and 23 transitions. We assume, initially all the processors are "active" working on the program segment alloted to them in their memory. The C-Bus and the various output and input buffers are free. This initial condition is represented by the token distribution shown in fig 4.3.

Let us assume that PE_1 needs to send a message to either PE_2 or PE_3. This is indicated by the firing of the enabled transition t_{12}, which moves the token in place p_{17} to the place p_{13}. Now the transitions t_8 and t_{13} are both simultaneously enabled, provided that the places p_8, p_{12}, p_{14} and p_{15} have tokens in them. Here transition t_8 models the message transfer to the output buffer by PE_1's interrupt handler and the C-Bus selection function for message transfer between PE_1 and PE_2. Similarly t_{13} models these functions for message transfer from PE_1 to PE_3. Since we have equal probability of sending a message to either processor PE_2 or PE_3, let us assume that the transition t_8 fires , thus routing the message to processor PE_2. The firing of transition t_8 puts a token in places p_{17}(PE_1 active) and place p_5, removes one token each from places p_8, p_{12}, p_{13} and p_{14} respectively. Transition t_5 is enabled

and firing of which indicates the completion of message transfer by the C-Bus controller to the processor PE_2. The firing of transition t_5 removes a token from place p_5 and puts one token each at places p_6,p_{12} and p_{14} indicating that PE_2 input buffer contains the message and the output buffer of PE_1 and C-Bus are free. The firing of enabled transition t_6, removes the token from place p_4 and puts a token in place p_7 indicating that PE_2 wants to receive a message. Now places p_6 and p_7 contain tokens thus enabling transition t_7 which models the function of removing the message from the input buffer by the interrupt handler of PE_2. The firing of transition t_7 removes tokens from places p_6,p_7 and puts tokens in places p_4,p_8. This indicates the reception of message by processor PE_2 and completes one message communication cycle.

The following is the relationship between the group of transitions and the pairs of processors from the messsage transmission point of view.

Transitions	Processor (from,to)
(t_{12},t_{13},t_{14})	(PE_1,PE_3)
(t_3,t_1,t_2)	(PE_2,PE_1)
(t_3,t_4,t_9)	(PE_2,PE_3)
(t_{15},t_{16},t_{20})	(PE_3,PE_1)
(t_{15},t_{11},t_{10})	(PE_3,PE_2)

The message reception by PE_3 and PE_1 are modeled by

transitions t_{18}, t_{21} and t_{17}, t_{19} respectively. The transitions t_{22} and t_{23} are provided to do the same operations as done by the transitions t_{11} and t_{12} respectively in the model SPN2A.

The firing rates of the various transitions are similar to the firing rates of transitions of SPN2A. The transitions $t_6, t_7, t_{12}, t_{15}, t_{17}$ and t_{18} fire at the rate $\lambda/2$ and transitions $t_2, t_5, t_9, t_{10}, t_{14}$ and t_{20} fire at the rate μ. The rest of the transitions which model the operations like determination of the availability of the receiver input buffer by the C-Bus controller etc. fire at a very high rate.

4.6 Modified Three processor SPN model:

The SPN model shown in fig 4.4 gives the three processor SPN model(SPN3B) similar to the model SPN2B.SPN3B has 28 places and 26 transitions. The following is the correspondence between the group of transitions and the pairs of processors, from the message transmission point of view:

Transitions	Processors (from,to)
$(t_{14}, t_{12}, t_8, t_3)$	(PE_1, PE_2)
$(t_{14}, t_{12}, t_{15}, t_{16})$	(PE_1, PE_3)
(t_6, t_7, t_2, t_1)	(PE_2, PE_1)
$(t_6, t_7, t_{10}, t_{17})$	(PE_2, PE_3)
$(t_{18}, t_{13}, t_{11}, t_{23})$	(PE_3, PE_1)
$(t_{18}, t_{13}, t_{24}, t_9)$	(PE_3, PE_2)

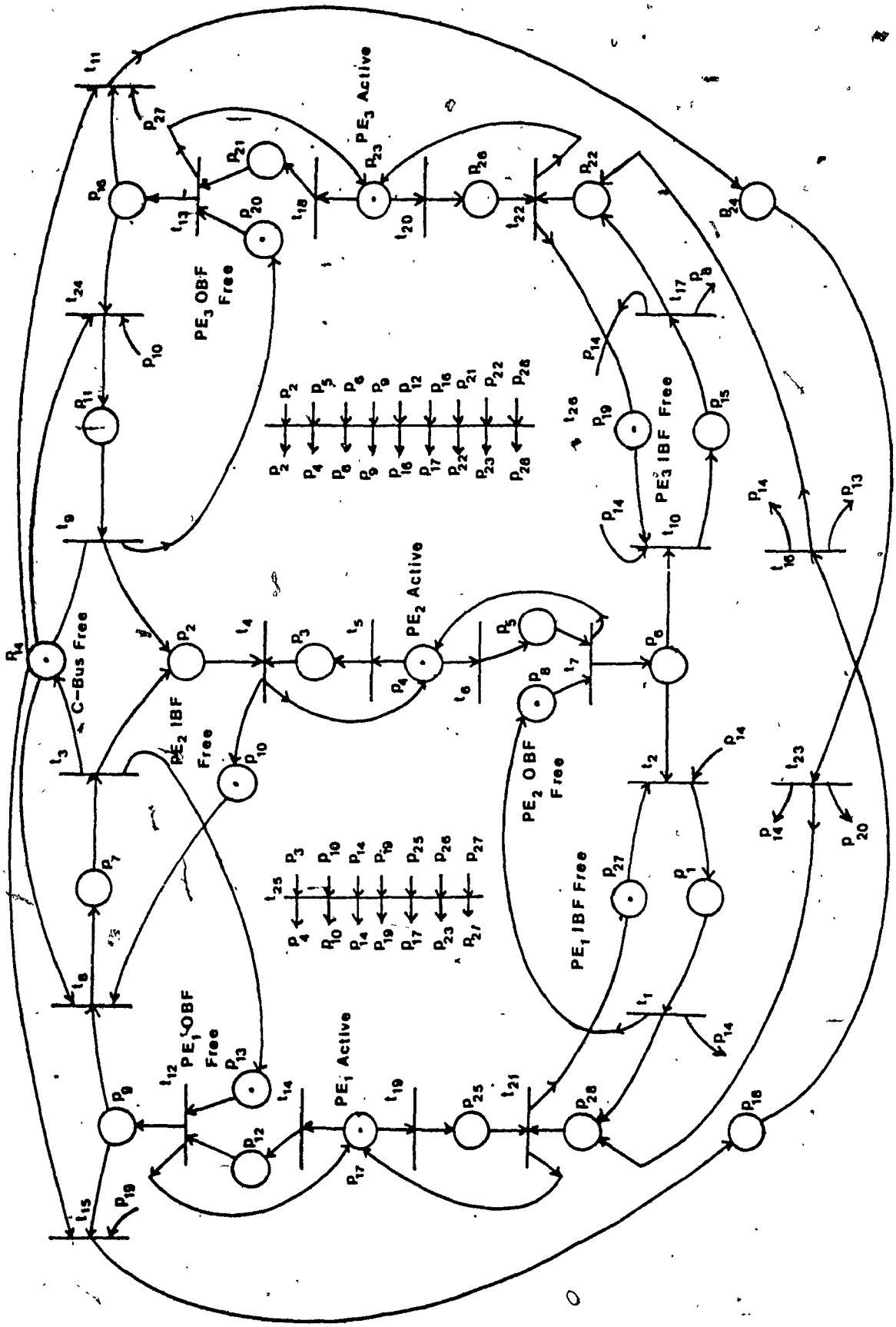


Figure 4.4 Modified Three Processor SPN model - SPN3B

From the message reception modeling aspect, the following is the relationship between the transitions and the corresponding processors:

Transitions	Processor
(t_{19}, t_{21})	PE_1
(t_5, t_4)	PE_2
(t_{20}, t_{22})	PE_3

The transitions t_{25} and t_{26} function similar to trasnitions t_{13} and t_{14} in SPN2B, resolving the two deadlock situations.

In this SPN model the transitions $t_5, t_6, t_{14}, t_{18}, t_{19}$ and t_{20} fire at the rate $\lambda/2$ and transitions $t_1, t_3, t_9, t_{16}, t_{17}$ and t_{23} fire at the rate μ. The rest of the transitions modeling the operations like checking the availability of the output buffer, availability of receiver's input buffer etc. fire at a very high rate .

4.7 Analysis of SPN models:

Determination of Reachability set:

In chapter 2, we have discussed the two methods, for the determination of reachability set - one using prime numbers and the other using binary and hexadecimal numbers.

The prime number method is employed in determining the reachability set for the two SPN models SPN2A and SPN2B. Since the prime number method has a limitation on the number of places due to the reasons stated in section 2.2.2, the

reachability sets for the three processor models SPN3A and SPN3B are determined using the binary number method.

As discussed in section 2.2.2, the presence of a "terminal" marking in the reachability set indicates the presence of a deadlock situation within the system being modeled. Thus during the process of determination of the reachability set, these methods also check for the presence of a terminal marking in the reachability set generated.

Table4.1 gives the number of markings in a each reachability set of these SPN models discussed. A representative listing of the various markings in a particular reachability set is given in Appendix D. This is the reachability set for the two processor model SPN2A.

Determination of the steady state probabilities of the states of the underlying Markov chain:

To determine the steady state probabilities of the states of an ergodic Markov chain, the transition rate matrix Q must be formed first. The order of the transition matrix Q is the number of states present in the underlying Markov chain. A nonzero entry q_{ij} in row i and column j in the transition matrix Q indicates that the system being in state i will move to state j at a rate of q_{ij}. This movement from state i to state j is represented in a SPN model in the following way. The state i represents a distribution of tokens in the SPN such that a transition t[illegible]s enabled firing of which

TABLE 4.1

DETAILS OF MODEL AND REACHABILITY SET SIZE FOR SPN MODELS OF CUENET.

SPN MODEL	NO. OF PLACES	NO. OF TRS.	SIZE OF R S.	SPARSITY OF TR. MATX NZ. ELES %
SPN2A	15	12	72	311 6.0
SPN2B	17	14	216	991 2.12
SPN3A	25	23	864	4578 0.61
SPN3B	28	26	4320	27995 0.15

NOTE: TRS. :Transitions.
TR. MATX :Transition Matrix.
NZ. ELES : Non- zero Elements.
R S : Reachability Set.

produces a new marking represented by state j. The value of the nonzero entry q_{ij} in the transition matrix Q corresponds to the rate at which this enabled transition t fires. In a particular marking(state) i more than one transition may be enabled. This is represented by the number of nonzero elements in the in the row i of Q. The diagonal element q_{ii} is obtained by the relation,

$$\sum_{i \neq j} q_{ij} = -q_{ii} \quad \text{.....(4.1)}$$

This relation makes the sum of q_{ij}s along a rwo equals to zero and also makes the transition matrix Q diagonally dominant which gives it good numerical stability [KLEIN-75,MEDHI-82]. The transition matrix Q has large number of zero entries thus making it highly sparse. This has a bearing on the way it can be solved.

After the formation of the transition matrix Q, the steady state probabilities of the various states of the markov chain are determined by solving the equation,

$$Q^T Y^T = 0 \quad \text{.....(4.2)}$$

The vector Y^T is a column vector which gives the steady state probability of each state in the ergodic Markov chain.

As discussed in section 2.3.3, the equation 4.2 is converted to Ax=b form and then solved. The solution can be obtained using any of the methods of solving a linear system of simultaneous equations. In the initial investigation of this thesis the Gauss-Jordan ellimination method with optimum pivoting was used.

Determiantion of the values of the Performance measures:

From the steady state probabilities the token densities of the places of the SPN model are determined. As discussed in section 2.3.2, the token density of a place p_i is given by the relation,

$$d_i = \sum_{j \epsilon R(PN)} y_j . m_{ij} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4.3)$$

where y_j is the steady state probability of state j and m_{ij} is the number of tokens in place p_i in state j.

Once the token densities of the various places of the SPN model are determined the various performance measures are obtained as shown below.

Processing Power P = Sum of the token densities of the places which correspond to the active processors in the DCS (see relation 3.1).

Example, for SPN2A, Processing power = $d_1 + d_{10}$.

C-Bus Throughput = (1-tokendensity of the place which models the free condition of C-Bus) X μ (see relation 3.2).

Example, In SPN2A, C-Bus Throughput = $(1-d_7) X \mu$.

Processor wait time = sum of token densities of places representing message for transmission and message receive request by that processor.

Example, in SPN2A, PE_1 wait time = $d_2 + d_5$.

4.8 Discussion of SPN models of CUENET:

The table 4.1 shows how the reachability set increases with the number of places and number of transitions. In this table we observe that in case of two procesor SPN models,

SPN2B differs from SPN2A in having 2 extra places and 2 extra transitions. But the number of markings of SPN2B is 300% of that of the markings of SPN2A. This indicates an enormous growth in the number of markings in the SPN with a small increase in the number of places and transitions. The direct consequece of this is the increased time complexity in obtaining the steady state probabilities. The sparsity of the rate matrix Q is also shown in Table 4.1. The non-zero elements of Q do not exhibit any structure that are well known to sparse matrix techniques such as band matrix, tridiagonal matrix etc. Hence in the initial stages of this thesis, Gauss-Jordan ellimination method was adopted for obtaining the solution to equation 4.2.

Table 4.2 indicates the time requirements for obtaining the solution by this method for the SPN models considered. The estimated time requirements for SPN3A is done with maintaining the data file externally, since the complete matrix cannot be resident in the central memory due to the restriction in its availability. Hence the difficulty in obtaining the solution to SPN3A and SPN3B.

The reachability set of a loosely coupled system is relatively large. This becomes evident from the following comparision. Marsan et al[MARSAN-83] have employed SPNs in their performance analysis of the TOMP multiprocessor (see section 3.5). Table 4.3 gives a comparision of the two SPN models (i.e. CUENET and TOMP) in terms of the number of

TABLE 4.2

COMPUTATION TIMES FOR SPN ANALYSIS:

SPN MODEL	SIZE OF R S	T1	T2
SPN2A	72	0.233	14.1
SPN2B	216	1.340	367.0
SPN3A	864	29.360	6000.0 *

Note:
T1 : Time in secs for determining the Reachability set and form Transition rate Matrix.
T2 : time in secs to obtain the solution from the Transition rate Matrix.
R S : Reachability Set.
* : Estimated Value

TABLE 4.3

COMPARISION OF CUENET AND TOMP SPN MODELS:

CUENET				TOMP			
SPN MODEL	NO.OF PLACES	NO.OF TRS.	SIZE OF R S	SPN MODEL	NO.OF PLACES	NO.OF TRS.	SIZE OF R S
SPN2A	15	12	72	2 PROC	15	14	26
SPN3A	25	23	864	3 PROC	31	33	395

Note:
TRS. : Transitions.
R S : Reachability Set.

SPN2B differs from SPN2A in having 2 extra places and 2 extra transitions. But the number of markings of SPN2B is 300% of that of the markings of SPN2A. This indicates an enormous growth in the number of markings in the SPN with a small increase in the number of places and transitions. The direct consequece of this is the increased time complexity in obtaining the steady state probabilities. The sparsity of the rate matrix Q is also shown in Table 4.1. The non-zero elements of Q do not exhibit any structure that are well known to sparse matrix techniques such as band matrix, tridiagonal matrix etc. Hence in the initial stages of this thesis, Gauss-Jordan ellimination method was adopted for obtaining the solution to equation 4.2.

Table 4.2 indicates the time requirements for obtaining the solution by this method for the SPN models considered. The estimated time requirements for SPN3A is done with maintaining the data file externally, since the complete matrix cannot be resident in the central memory due to the restriction in its availability. Hence the difficulty in obtaining the solution to SPN3A and SPN3B.

The reachability set of a loosely coupled system is relatively large. This becomes evident from the following comparision. Marsan et al[MARSAN-83] have employed SPNs in their performance analysis of the TOMP multiprocessor (see section 3.5). Table 4.3 gives a comparision of the two SPN models (i.e. CUENET and TOMP) in terms of the number of

TABLE 4.2

COMPUTATION TIMES FOR SPN ANALYSIS:

SPN MODEL	SIZE OF R S	T1	T2
SPN2A	72	0.233	14.1
SPN2B	216	1.340	367.0
SPN3A	864	29.360	6000.0 *

Note:
T1 : Time in secs for determining the Reachability set and form Transition rate Matrix.
T2 : time in secs to obtain the solution from the Transition rate Matrix.
R S : Reachability Set.
* : Estimated Value

TABLE 4.3

COMPARISION OF CUENET AND TOMP SPN MODELS:

CUENET				TOMP			
SPN MODEL	NO.OF PLACES	NO.OF TRS.	SIZE OF R S	SPN MODEL	NO.OF PLACES	NO.OF TRS.	SIZE OF R S
SPN2A	15	12	72	2 PROC	15	14	26
SPN3A	25	23	864	3 PROC	31	33	395

Note:
TRS. : Transitions.
R S : Reachability Set.

places and transitions and size of the reachability sets for the case of two and three processors. The two processor SPN model of TOMP has more number of transitions as compared to the two processor CUENET model(SPN2A) while both have the same number of places. We observe that the two processor TOMP SPN has only 26 markings in its reachability set whereas the SPN2A has 76 states. This large difference is seen in the case of three processor models also. It is interesting to note that SPN3A has lesser number of places and transitions as compared to the TOMP model.

This increase in the number of states is attributable to the loose coupling nature of CUENET. A loosely coupled system exhibits more asynchronism. This enables the system components to function more independetly. Thus an SPN which models these actions will have more number of states in its reachability set.

The results of the SPN analysis of the two processor models SPN2A and SPN2B are presented in Tables 4.4 and 4.5 respectively. These tables list the variation of the procesing power, C-Bus throughput and the processor wait time with the system load factor ρ. From these results it can be seen that SPN2B gives a more optimistic results as compared to SPN2A, as discussed earlier in this chapter.

The large growth of the number of states in the rechability set and the consequent time complexity in the

TABLE 4.4

SPN ANALYSIS RESULTS OF SPN2A.

ρ	λ	μ	PP	CT	P1W	P2W
0.01	0.417	41.66	1.4626	0.2249	0.2687	0.2687
0.05	2.803	41.66	1.4588	1.1082	0.2706	0.2706
0.1	4.166	41.66	1.4530	2.2080	0.2736	0.2736
0.2	8.333	41.66	1.4376	4.3409	0.2812	0.2812
0.4	16.666	41.66	1.3942	8.3195	0.3030	0.3030
0.6	24.996	41.66	1.3384	11.8648	0.3307	0.3307
0.8	33.328	41.66	1.2766	14.9601	0.3617	0.3617
1.0	41.66	41.66	1.2130	17.6347	0.3965	0.3965

NOTE:

PP: Processing Power.

CT: C-Bus Throughput in messages/sec.

P1W: Wait time for PE1.

P2W: Wait time for PE2.

TABLE 4.5

SPN ANALYSIS RESULTS OF SPN2B.

ρ	λ	μ	PP	CT	P1W	P2W
0.01	0.417	41.66	1.5406	0.2625	0.2297	0.2297
0.05	2.083	41.66	1.5342	1.3040	0.2329	0.2329
0.1	4.166	41.66	1.5246	2.5871	0.2377	0.2377
0.2	8.333	41.66	1.5014	5.0825	0.2492	0.2492
0.4	16.666	41.66	1.4408	9.6860	0.2796	0.2796
0.6	24.996	41.66	1.3674	13.7061	0.3163	0.3163
0.8	33.328	41.66	1.2890	17.1306	0.2890	0.2890
1.0	41.66	41.66	1.2104	20.0051	0.3949	0.3949

NOTE:

PP : Procesing Power.

CT : C-Bus Throughput in Messages/sec.

P1W : Wait time for PE1.

P2W : Wait time for PE2.

mathematical solution limits the use of SPN in modeling CUENET. This prompted us to consider the Generalized Stochastic Petrinets(GSPN), which are derivatives of the SPNs for the modeling of CUENET.

CHAPTER V

GENERALIZED STOCHASTIC PETRINET(GSPN) MODELING OF CUENET.

5.1 Preliminaries:

In chapter IV the SPN modeling of CUENET has brought home the fact that the increase in the solution complexity ,when we progressed from two to three processors, is because of the enormous increase in the number of states in the reachability set. In an effort to limit the size of the reachability set and at the same time maintain reasonable level of details in the model, we have adopted the GSPN(proposed by Marsan et al)[MARSAN-84] for the modeling of message communication in CUENET.

In the GSPN modeling of CUENET, the assumptions made for the SPN models still hold. Also we are employing the same values of λs and μs as used in the SPN models.

Since GSPN has two kinds of transitions, namely timed and immediate, in our models we have adopted the following assignment rules for the various transitions. Those transitions which model message generation, transmission and reception are treated as transitions. Each of them fire at a mean rate following an exponential distribution. The system parameters λ and μ determine these mean rates. The other transitions which model the various other operations of CUENET like checking the availability of the different

resources required, resolving deadlock situations are represented as immediate transitions which fire in zero time.

5.2 Two processor GSPN models:

Fig5.1 shows the two processor GSPN model(GSPN2A) of CUENET which is similar to the SPN2A.The assumptions made for simplifying the SPN2A also hold good for GSPN2A(See page No.87). The GSPN2A has 15 places, 6 timed transitions and 6 immediate transitions. The transitions t_2, t_3, t_6, t_7, t_9 and t_{10} are timed transitions while the transitions $t_1, t_4, t_5, t_8, t_{11}, t_{12}$ are immediate transitions.

The interpretations for these transitions are exactly similar to that of SPN2A. The main difference of GSPN is the way in which the various immediate transitions are assigned probabilities of firing when more than one immediate transitions get enabled in a given marking. The firing probability for each immediate transition depends on the number of immediate transitions that get enabled in a particular marking and on certain characteristics of the system being modeled. These various probabilities are given in the form of a probability distribution called "switching distribution".

In this model, the immediate transitions t_1, t_4 cannot get enabled at the same time, due to the conflicting timed transitions t_2 and t_3. For the same reason, the immediate

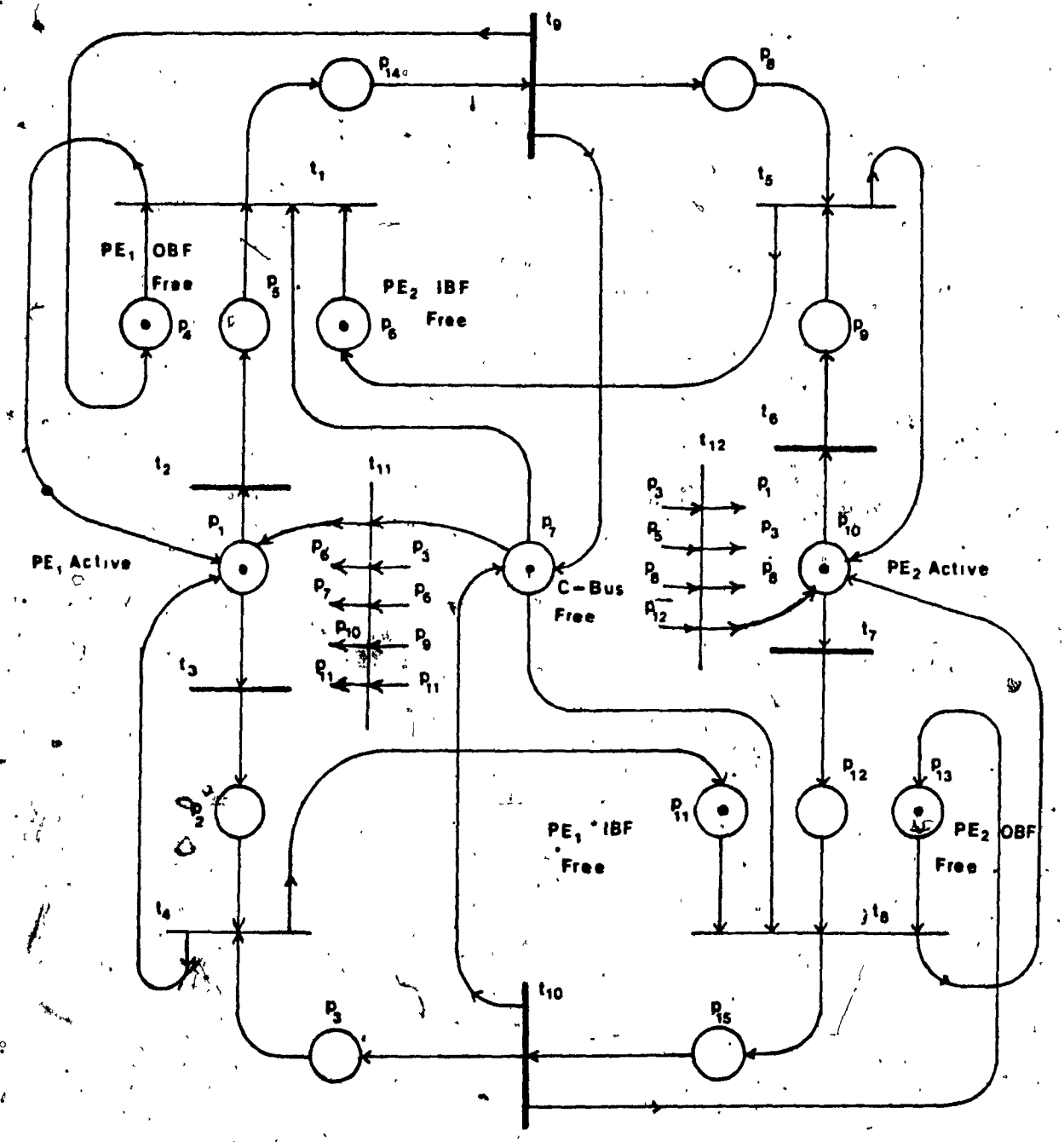


Figure 5.1 Two processor GSPN model - GSPN2A

transitions t_5 and t_8 cannot get enabled simultaneously.

The immediate transitions t_1 and t_5 cannot get enabled at the same instant, due to the reason places p_6 and p_8 cannot have tokens at the same time. This reason also holds for transitions t_4 and t_8.

The immediate transitions t_{11} and t_{12}, being transitions for resolving deadlock situations, get enabled independently.

Due to the above reasons, only one of these six immediate transitions get enabled in any given vanishing state. Hence their switching distribution is that each of them will fire with a probability 1.0, when they are enabled.

Fig5.2 shows the modified two processor GSPN model(GSPN2B) similar to the SPN2B.The GSPN2B has 17 places, 6 timed and 8 immediate transitions. Once again the interpretations for these timed and immediate transitions are similar to that of the SPN2B.

The switching distributions for the various immediate transitions are as described below:

In this model, like in GSPN2A, immediate transitions t_2 and t_5 cannot get enabled at the same time due to the two conflicting timed transitions t_3 and t_4. For the same reason, t_7 and t_{10} cannot get enabled at the same time.

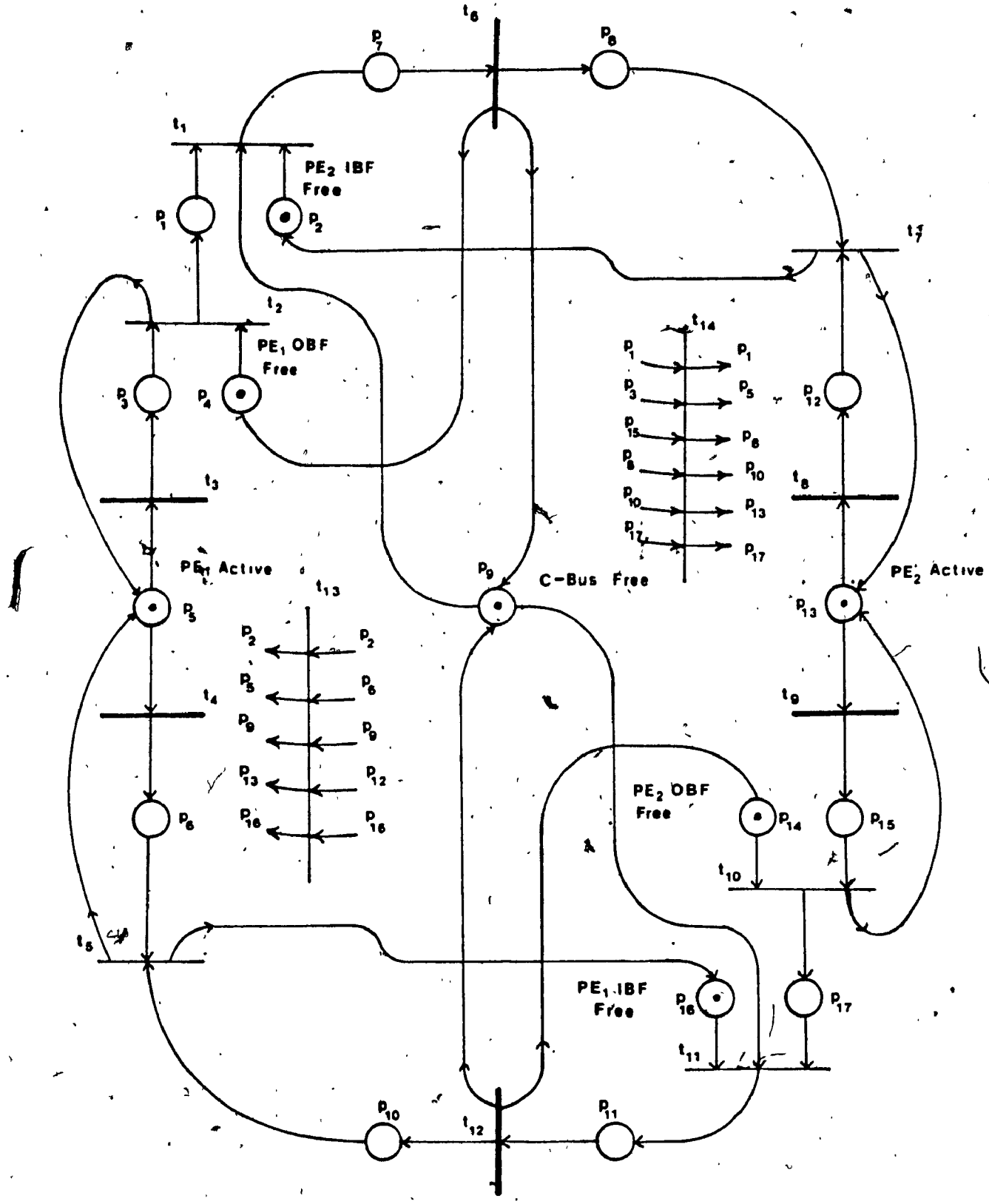


Figure 5.2 Modified Two Processor GSPN model - GSPN2B

The immediate transitions t_1 and t_7 cannot get enabled at the same time since the places p_2 and p_8 cannot contain a token each at the same time(token p_2 indicates that the particular input buffer free while a token in p_8 indicates the input buffer contains a message). The immedaite transitions t_5 and t_{11} cannot get enabled at the same time, for a similar reason.

Thus it is observed, from the various vanishing states, that the transitions t_1,t_5 and t_{10} get enabled individually or in pairs. Also the transitions t_2,t_7 and t_{11} get enabled individually or in pairs. Transitions t_1 and t_{11} get enabled together in certain vanishing states of the GSPN. The switching probability for each of these immediate transitions will include all these possibilities.

The following are the switching probabilities for the immediate transitions:

t_1: when t_5 or t_{10} or t_{11} is enabled :prob=1/2; Otherwise, Prob=1.

t_2: when t_7 or t_{11} is enabled :Prob=1/2; Otherwise: Prob =1.

t_5: when t_1 or t_{10} is enabled : Prob=1/2; Otherwise : Prob=1.

t_7: when t_2 or t_{11} is enabled: Prob = 1/2; Otherwise : Prob=1.

t_{10} : when t_1 or t_5 is enabled: Prob = 1/2; Otherwise: Prob=1.

t_{11} : when t_1 or t_2 or t_7 is enabled: Prob =1/2;
Otherwise : Prob=1.

As we have found the size of the tangible states in the reachability sets were not too large, a more detailed GSPN was evolved. In the case of GSPN2A and GSPN2B models there are no send and receive message queues. In the new model we have included the send and receive queues at each processor and the resultant GSPN model(GSPN2C) is shown in fig 5.3.The GSPN2C has 23 places, 6 timed and 10 immediate transitions.

The initial state of the model is assumed to be that both the processors PE_1 and PE_2 are "active", the various send and receive message queues are empty and the respective input and output buffers and the C-Bus are free. This initial state is indicated by the token distribution shown in fig 5.3. The token p_7 indicates that PE_1 is "active" and tokens in p_2 and p_{20} represent the output and input buffers of PE_1 are free. The tokens in p_5 and p_{10} indicate the size of the send and receive queues. The places p_{17},p_{22},p_4,p_{19} and p_{14} correspond to the processor PE_2 and have the same interpretation. The token in place p_{12} represents that the C-Bus and C-Bus controller are free.

In this model, except for the transitions which model the send and receive queues at each processor, the operation of the other transitions are exactly similar to the ones in GSPN2A.

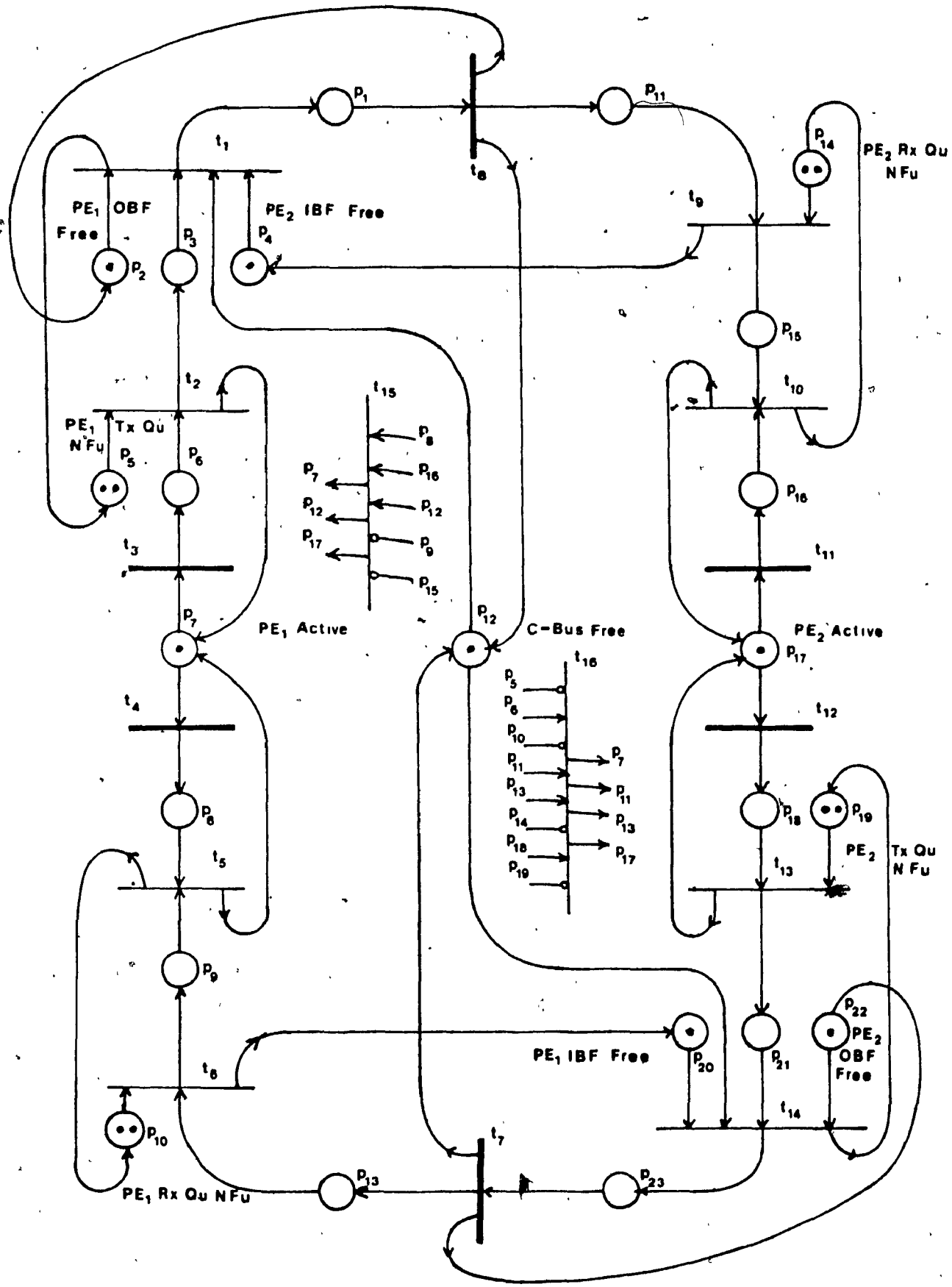


Figure 5.3 Two processor (with message queues) GSPN model - GSPN2C.

The immediate transitions t_2 and t_6 represent the message queueing at the send and receive points in processor PE_1. Similarly, t_{13} and t_9 model the message queueing in PE_2.

The message generation by PE_1 and the transfer of the message to PE_2 by the C-Bus controller are modeled by the sequetial firings of timed transition t_3,immediate transitions t_2,t_1, timed transition t_8 and the immediate transition t_9. The message reception by PE_2 is modeled by the timed transition t_{11} and immediate transition t_{10}.

Similarly the message communication from PE_2 to PE_1 is modeled by timed transitions t_{12},t_7 and immediate transitions t_{13},t_{14} and t_6. The reception of the message by PE_1 is modeled by timed transition t_4 and immediate transition t_5.

The immediate transitions t_{15} and t_{16} resolve the two deadlock situations similar to the ones in SPN2A and SPN2B. Since in GSPN2C we are employing multiple tokens, the use of inhibitor arcs simplify the modeling of these deadlock situations.

The switching distributions for each of the immediate transitions are as given below. Here, like in GSPN2A, certain immediate transitions do not get enabled at the same time. They are (t_2,t_5),(t_{10},t_{13}),(t_1,t_9), and (t_6,t_{14}). Also the transitions t_{15} and t_{16} get enabled seperately.

With these exceptions, the immedaite transitions in GSPN2B get enabled singly, in pairs or in triples.

For example, t_1 get enabled together with t_2 or t_5 or t_6 or t_{13} or t_{14} or t_{10} and in combinations of (t_{14},t_{10}) and (t_5,t_{14}). We will describe the switching distributions of transitions t_1,t_2,t_5 and t_6 associated with PE_1. In a similar manner, from the homogeneous nature of these processors, the switching distributions for transitions t_{14},t_{13},t_{10} and t_9 can be defined.

Switching distributions for:

t_1 :when t_2 is enabled : Prob = $m_3/(m_3 + m_5)$; when t_5 or t_6 or t_{10} or t_{13} or t_{14} is enabled: prob =1/2;
when both $t_{14}$and t_{10} or t_5 and t_{14} are enabled :Prob=1/3:
otherwise:prob=1.

t_2 : when t_1 is enabled : Prob = $m_5/(m_3 + m_5)$;
when t_6 or t_{10} or t_{14} is enabled : prob = 1/2;
otherwise prob= 1.

t_5 : when t_6 is enabled : prob = $m_9/(m_9 + m_{10})$;
when t_{13} or t_{14} is enabled : prob = 1/2;
when both t_1 and t_{14} are enabled : prob =1/3;
otherwise: prob =1.

t_6 : when t_1 or t_2 is enabled : prob =1/2;
when t_5 is enabled : prob = $m_{10}/(m_9 + m_{10})$;
otherwise: prob=1.

In the GSPN2C we model the effect of different message capacity for the message queues by simply altering the

number of tokens in places which model them. For example, places p_5,p_{10},p_{14} and p_{19} in fig 5.3 contain 2 tokens each indicating that this GSPN models a two processor CUENET with size of the message queue 2. To study the effects with message queue size 3 messages we just alter the number of tokens in these places to 3.

5.3 Three processor GSPN model:

Fig5.4 shows the three processor GSPN model of CUENET(GSPN3) which is similar to the three processor model SPN3A. GSPN3 has 25 places, 12 timed transitions and 11 immediate transitions.

The timed transitions t_{12},t_3 and t_{15} model the message generation by processors PE_1,PE_2 and PE_3 respectively. The message transmission from PE_1 to (PE_2,PE_3), from PE_2 to (PE_1,PE_3) and from PE_3 to (PE_1,PE_2) are modeled by the pairs of timed transitions (t_5,t_{14}),(t_2,t_9) and (t_{20},t_{10}) respectively. The message reception by PE_1 is modeled by the timed transition t_{17} and the immediate transition t_{19}. In the similar manner, the timed and immediate transition pairs (t_6,t_7) and (t_{18},t_{21}) model the message reception by PE_2 and PE_3 respectively. The C-Bus controller functions of determining the availability of destination resources are modeled by the immediate transitions t_1,t_4,t_8,t_{11}, t_{13} and t_6. The operation of this GSPN model is exactly similar to the SPN model SPN3A. The major difference is the way the various immediate transitions are assigned probabilities of

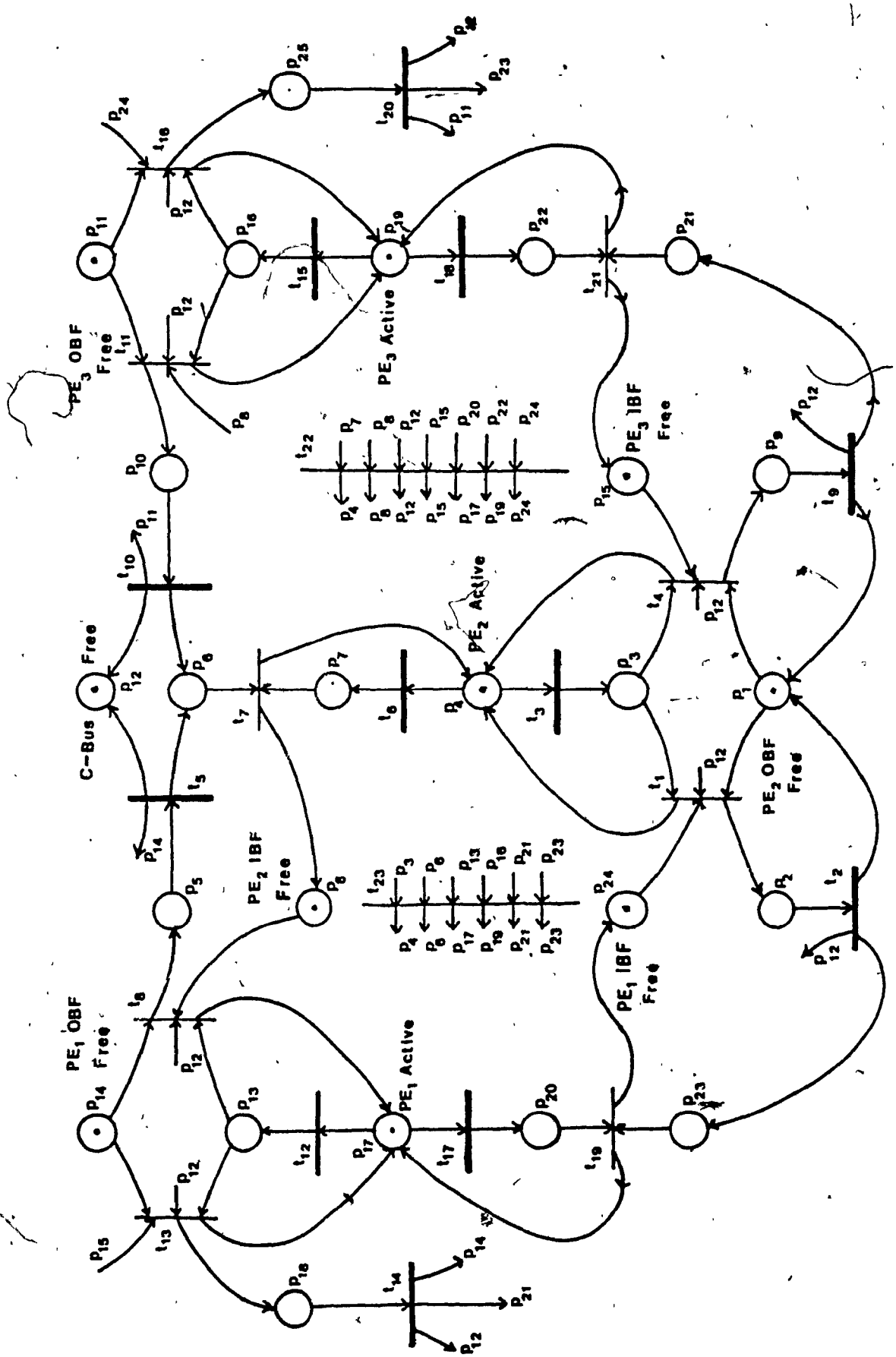


Figure 5.4 Three Processor GSPN model - GSPN3

firing when more than one of the immediate transitions get enabled in a given marking. The switching distributions of the various immediate transitions are as follows.

There are certain immediate transitions which do not get enabled at the same time, for reasons similar to ones given in GPSN2A. We shall consider the immediate transitions connected with processor PE_1 and describe the switching distributions to them. In a similar manner, using the symetrical nature of the model, the switching distributions of the immediate transitions associated with processors PE_2 and PE_3 can be derived.

The immediate transition t_{19} do not get enabled at the same time with the transitions t_8 and t_{13} due to the two conflicting timed transitions t_{12} and t_{17}. Also t_{19} cannot get enabled with the t_1, t_{16} since places p_{23} and p_{24} cannot have tokens at the same time. Transition t_{19} can get enabled at the same time with either t_4 or t_{11}.

The immediate transitions t_8 and t_{13} get enabled individually or together with one of the other four immediate transitions t_1, t_4, t_{11} and t_{16}. Thus the switching distribution for these transitions will include all these various possibilities.

The switching distributions are:

t_8: when t_{13} is enabled: Prob=1/2;

when (t_{13}, t_1) or (t_{13}, t_{14}) or (t_{13}, t_{11}) or (t_{13}, t_{16}) or

(t_1,t_4) or (t_{11},t_{16}) are enabled together : prob= 1/3;
otherwise : prob=1.

t_{13} : when t_8 is enabled : prob =1/2;
when (t_8,t_1) or (t_8,t_4) or (t_8,t_{11}) or (t_8,t_{16}) or (t_1,t_4) or (t_{11},t_{16}) are enabled together : prob = 1/3;
otherwise : prob=1.

t_{19}:when t_4 or t_{11} is enabled : prob=1/2;
otherwise : prob =1.

5.4 Analysis of GSPN models:

The analysis of a GSPN model, like the SPN analysis, invloves first the determination of the reachability set of the GSPN which comprises of both the tangible markings and the vanishing markings(see chapter II). The second part of the analysis concerns with the determination of transition probability matrix for the Reduced Embedded Markov Chain(REMC) formed over only the tangible markings of the GSPN under study.

We have used the software package developed for this purpose in Pascal language by Marsan et al[MARSAN-85b]. This software is designed to run on VAX11/780. It mainly consists of an interactive editor and various modules for determining the reachability sets, transition probability matrix and solution to the steady state probabilities of the resulting REMC. The interactive editor helps in describing the GSPN model in terms of the places, timed transitions and their firing rates, and immediate transitions along with

their switching distributions and the initial markings. The firing rates and the initial markings are treated as variable parameters so that the GSPN model can be executed for different initial markings and firing rates without specifying the model description every time. Firing rates for the timed transitions can be assigned with a fixed rate or a rate which varies with the number of tokens in its input places. Also this software allows the definition of firing rates and switching distribution with complex marking dependecies.

Determination of the reachability set:

In this the number of tangible markings and vanishing markings present in the reachability set are determined by the program GRG. The program starts with the initial marking of the GSPN model and fires all the enabled transitions in that marking. As discussed in section 2.4, the rules for the firing of the timed and immediate trasnitions are followed while firing the various timed and immediate transitions in a particular marking. Also in determining the nature of a new marking generated from an existing markings the rules for determining tangible and vanishing markings are used. In this software the timed transitions proceed on breadth-first manner, in which all the enabled timed transitions in a given marking are fired in sequence. The immediate transitions are fired on a depth-first manner where in the firing of successive

immediate transitions in the resulting markings are followed till a tangible marking is reached. This ordering of the markings in the reachability set help in determining the transition probabilities for the various vanishing to tangible path. These probabilities are used in determining the entries of the transition matrix defined over the tangible markings only.

Table 5.1 gives the details of reachability set for the models GSPN2A, GSPN2B,GSPN2C and GSPN3. A representative listing of the tangible and vanishing markings of a reachability set is shown in Appendix E. This is the reachability set for GSPN2B.

Determination of the steady state probabilities of the tangible markings:

The program GMD first convertes the marking dependent firing definitions and the switching distributions into firing rate / probability definitions. Here a recursive descendent algorithm is employed to perform syntatic analysis of the logical definition statements and then translates them into a sequence of if-then-else statements. Then the transition probabilities for the transition matrix formed only by the tangible markings are determined by the program GMT using these definitions statements and information collected during the determination of the tangible and vanishing markings in the reachability set. This transition matrix

formed over only the tangible markings is solved for the determination of the steady state probabilities. The GSPN software employs Gauss ellimination and Gauss-Seidel iterative methods(programs GGE and GGS) for the determination of these steady state probabilities.

Determination of the values of the Performance measures: The token density d_i for each place p_i in the given GSPN model is determined by the relation 4.3. In this context, y_j is the steady state probability of tangible marking j in the reachability set of the GSPN model.

From the various token densities, the performance measures are derived in a similar manner as done in the case of SPN analysis. As an example, the performance measures for the two processor model GSPN2B are determined by the following relations:

Processing Power $P = d_1 + d_{13}$,

C-Bus throughput = $(1-d_9)X\mu$,

PE_1meanwait = $d_1 + d_6$,

PE_2meanwait = $d_{15} + d_{12}$.

5.5 Discussion of GSPN models of CUENET:

Table 5.1 shows how the reachability set of GSPN varies with the GSPN complexity of the model. It is seen that the growth of the overall reachability set is similar, but lesser than the growth of the SPN reachability set

TABLE 5.1

DETAILS OF MODEL AND REACHABILITY SET SIZE
OF GSPN MODELS OF CUENET.

GSPN MODEL	NO. OF PLACES	NO. OF TMD TRS	NO. OF IMM RES	SIZE OF R S	NO. OF T St	NO. OF V St
GSPN2A	15	6	6	70	40	30
GSPN2B	17	6	8	182	67	115
GSPN2C (Qu Sz =1)	23	6	10	763 (1152)*	200	563
GSPN2C (Qu Sz =2)	23	6	10	2828 (5832)*	612	2216
GSPN3	25	12	11	826	479	347

NOTE:

TMD TRS : Timed Transitions.
IMM TRS : Immediate Transitions.
R S : Reachability Set.
T St : Tangible States.
V St : Vanishing States.
Qu Sz : Send and receive message queue size.
* : Equivalent SPN reachability set size.

TABLE 5.2

COMPUTATIONS TIMES FOR GSPN ANALYSIS:

MODEL	T1	T2	T3
GSPN2A	0.830	1.4	0.4
GSPN2B	2.59	4.8	0.8
GSPN2C (Qu Sz =1)	24.0	82.9	2.2
GSPN2C (Qu Sz =2)	87.0	2900.0	25.0
GSPN3	41.0	2750.0	10.0

T1 : Time in secs to determine the Reachability set and form the Transition Matrix.
T2 : Time in secs to determine Solution from Tr.matrix using Gaussian Elimination Method
T3 : Time in secs to determine the solution from the transition matrix using Gauss-Seidel iterative method.
Qu Sz : Send and receive message queue size.

(see Table4.1) (This difference is quite significant for GSPN2C, for which, the figures given in parenthese correspond to the reachability set of equivalent SPN). But the states of GSPN are bifurcated into tangible and vanishing states and the computational complexity of the solution depends mainly on the size of the tangible markings of the GSPN. By observing the two tables 5.1 and 4.1 the computational complexity in the case of GSPN will be about one-third to one half the solution complexity of SPN.

The computational time requirement, for doing GSPN analysis using the Gauss Ellimination method and the Gauss-Seidel Iterative method, are shown in Table 5.2. It is observed from Table 5.2, that the Gauss ellimination method takes a large amount of time, even with moderate size of the reachability (tangible) set of the GSPN. Thus for GSPNs with larger size tangible states the Gauss-Seidel iterative method was employed for obtaining the solution. The approximate solution provided by this iterative method compares quite well with those obtained using Gauss ellimiantion method.

The results of GSPN analysis done on GSPN2A,GSPN2B, GSPN2C and GSPN3 are shown in Tables 5.3,5.4, 5.5 and 5.6 respectively. Tables 5.3 and 5.4 gives the variation of processing power, C-Bus throughput, and processor wait time with the system load factor. In Table 5.5 the variation of processing power and C-Bus throughput with system load

TABLE 5.3

GSPN ANALYSIS RESULTS OF GSPN2A.

ρ	λ	μ	PP	CT	P1W	P2W
0.01	0.4166	41.66	1.4539	0.2270	0.2731	0.2731
0.05	2.083	41.66	1.4507	1.1307	0.2747	0.2747
0.1	4.166	41.66	1.4457	2.2485	0.2772	0.2772
0.2	8.333	41.66	1.4319	4.4346	0.2821	0.2821
0.4	16.666	41.66	1.3912	8.5423	0.3044	0.3044
0.8	33.328	41.66	1.2772	15.4614	0.3614	0.3614
1.0	41.66	41.66	1.2142	18.2701	0.3929	0.3929

NOTE:

PP : Processing Power.
CT : C-Bus Throughput in Messages/sec.
P1W : wait time for PE1.
P2W : wait time for PE2.

TABLE 5.4

GSPN ANALYSIS RESULTS OF GSPN2B:

ρ	λ	μ	PP	CT	P1W	P2W
0.01	0.417	41.66	1.5261	0.2625	0.2369	0.2369
0.05	2.083	41.66	1.5209	1.3186	0.2395	0.2395
0.1	4.166	41.66	1.5131	2.6205	0.2434	0.2324
0.2	8.333	41.66	1.4934	5.1599	0.2533	0.2533
0.4	16.666	41.66	1.4396	9.8921	0.2802	0.2802
0.6	24.996	41.66	1.3723	14.0643	0.3139	0.3139
0.8	33.328	41.66	1.2984	17.6494	0.3508	0.3508
1.0	41.66	41.66	1.2230	20.6818	0.3885	0.3885

NOTE:

PP : Processing Power.
CT : C-Bus Throughput in messages/sec.
P1W : wait time for PE1.
P2W : wait time for PE2.

factor for different queue sizes are depicted.

The comparision of the results in Tables 5.3 and 5.4 with results in Tables 4.4 and 4.5 shows that the results of SPN and GSPN models for two processor CUENET agree.

In a loosely coupled system the presence of message queues at each processor facilitates a better use of the resources of the system. The comparision of the results in Tables 5.3 and 5.4 with those of Table 5.5 indicate how the increase in the C-Bus throughput has less decreasing effect on the processing power, with the introduction of message queues at each processor, in the model.

The results for three processor case (GSPN3) shown in Table 5.6 indicates that the processing power, C-Bus throughput increase proportional to the number of processors, as compared to the results of two processor case (GSPN2A). The processor wait time increases due to the increased contention for the use of C-Bus.

5.6 Aggregated GSPN model for two processor CUENET:

In chapter IV and so far in this chapter we have modeled the various message comunications aspects of CUENET for two and three processors cases. In these models the action of each processor are explicitly modeled. Also we have considered the different system characteristics such as the presence of only output and input buffers and message queues at each processor.

TABLE 5.5

GSPN ANALYSIS RESULTS OF GSPN2C:

ρ	λ	μ	QU SIZE = 1 PP	QU SIZE = 1 CT	QU SIZE =2 PP	QU SIZE =2 CT
0.01	0.417	41.66	1.5778	0.2876	1.6479	0.3466
0.05	2.083	41.66	1.5758	1.4357	1.6462	1.5716
0.1	4.166	41.66	1.5725	2.8647	1.6439	3.1382
0.2	8.333	41.66	1.5636	5.6939	1.6376	6.2514
0.4	16.666	41.66	1.5321	11.1404	1.6155	12.3247
0.6	24.996	41.66	1.4797	16.1050	1.5737	17.9870
0.8	33.328	41.66	1.4095	20.4126	1.5082	22.9520
1.0	41.66	41.66	1.3287	24.0041	1.4231	27.0269

NOTE:

PP : Processing Power
CT : C-Bus Throughput in messages/sec.

TABLE 5.6

GSPN ANALYSIS RESULTS OF GSPN3:
(FOR μ=41.66)

ρ	λ	PP	CT	P1W	P2W	P3W
0.01	0.417	2.0663	0.3586	0.3112	0.3112	0.3112
0.05	2.083	2.0603	1.7868	0.3132	0.3132	0.3132
0.1	4.166	2.0501	3.5526	0.3166	0.3166	0.3166
0.2	8.333	2.0191	6.9861	0.3270	0.3270	0.3270
0.4	16.666	1.9206	13.2453	0.3598	0.3598	0.3598
0.6	24.996	1.7919	18.4773	0.4027	0.4027	0.4027
0.8	33.238	1.6537	22.6734	0.4488	0.4488	0.4488
1.0	41.66	1.5190	25.9727	0.4937	0.4937	0.4937

NOTE:
PP : Processing Power.
CT : C-Bus Throughput in messages/sec.
P1W : wait time for PE1.
P2W : wait time for PE2.
P3W : wait time for PE3.

Table5.1 illustrates how the model size and the reachability set size increase with increase in the details of the system being modeled. The number of places and transitions required to model two processor CUENET with message queues is almost equal to those required to model three processors without message queues. The size of the three processor model with the message queues will be 37 places and 29 transitions (12 timed and 17 immediate).

Also in GSPN models, the definition of the switching probability for the various immediate transitions can get quite involved with the increase in the number of immediate transitions in the model.

Since, in the final analysis, one is interested in the behavior of the overall system, we could represent the actions of individual processors collectively through the use of multiple tokens in selected places. By doing this the increase in the size of the model will not be high when additional details are included in the model. Further by doing such aggregation, the extension of the model for larger number of processors can be done easily with the change in the initial token distribution and/or by adding a relatively small but a new section to the model.

Since GSPN offers a better modeling capability than SPN, we have adopted GSPN for doing these aggergated CUENET models.

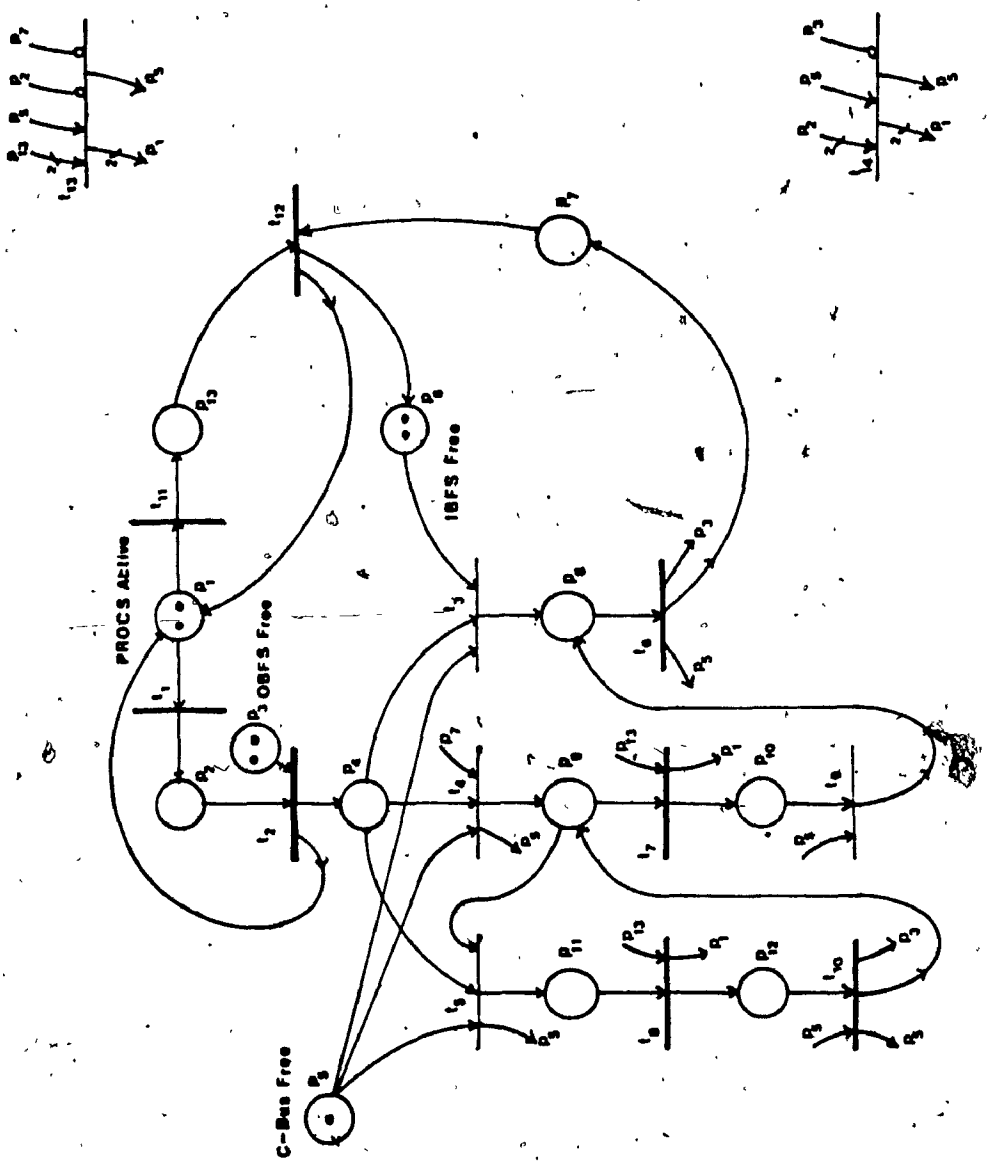


Figure 5.5 Two processor Aggregated GSPN model- GSPNR2

Fig5.5 shows the aggregated two processor GSPN model of CUENET(GSPNR2). This model has 13 places, 8 timed transitions and 6 immediate transitions. GSPNR2 essentailly models the two processor CUENET without message queues. It is similar to the GSPN2B but differs from GSPN2B in two aspects: One is we are using multiple tokens which may be used to denote the number of processors, input buffers or output buffers. The second aspect is that the rates of firing of the timed transitions vary with the number of tokens at their input places.

In GSPNR2 the activities of the various processors are modeled by places $p_1, p_2, p_3, p_4, p_6, p_7, p_{13}$ and timed transitions t_1, t_2, t_{11} and t_{12}.

The C-Bus controller activities include checking the availability of the input buffer of the destination ,queueing the message if necessary and the actual message transfer. The places ($p_8, p_9, p_{10}, p_{11}, p_{12}$), timed transitions ($t_6, t_7, t_8, t_{10}$) and immediate transitions (t_3, t_4, t_5, t_9) model these activities.

The two immediate transitions t_{13} and t_{14} resolve the two deadlock situations similar to the ones in SPN2B and GSPN2B.

We shall describe the opeartion of the model by assuming an initial condition in which both the processors are

"active,the output and input buffers and the C-Bus controller are free. This initial state is indicated by the token distribution shown in fig 5.5, where place p_1 contains 2 tokens indicating both processors are active and places p_3 and p_6 contain 2 tokens each indicating that both the input and output buffers of the two processors are free. Place p_5 has one token to represent C-Bus controller being free.

The message generation by the sender and its placement in the output buffer is modeled by the timed transitions t_1 and t_2. The immediate transition t_3 and timed transition t_6 represent the C-Bus controller functions of determining the availability of the free receiver and the transfer of message to it. The message reception by the receiver processor is modeled by the timed transitions t_{11} and t_{12}. This forms a message communication cycle.

When the destination's input buffer is busy, the C-Bus controller queues this message. This is modeled by the immediate transition t_4. The immediate transition t_5 models the queueing of second message for the same destination. The timed transitions t_7 and t_8 model the message reception by the destination processor with the additional information, one and two message are queued for the same destination. The operation of immediate transition t_9 is similar to t_3 with the information that this message was queued for this destination. Timed transition t_{10} represents the C-Bus controller's message transfer of one of

the two queued messages.

In the previous models the transfer rate comprised of the message transfer rate at the sender to its output buffer, the transfer rate of the C-Bus controller and receiver processor rate of removing the message from the input buffer. In this model these components are seperately modeled by different timed transitions. The timed transition t_2 fires at a marking dependent rate α_T, where α_T is the rate at which the sender processor transfers the message to its output buffer. The timed transitions t_7, t_8 and t_{12} fire at a marking dependent rate α_R, where α_R is the rate at which the receiver processor removes the message from its input buffer. The timed transitions t_6 and t_{10} fire at a constant rate μ_C, where μ_C is the rate at which the C-bus controller transfers messages from the senders' output buffers to the receivers' input buffers. The message transfer rate μ used in the earlier models and these three rates are related by the following expression:

$$1/\mu = 1/\mu_C + 1/\alpha_T + 1/\alpha_R \quad \text{.....(5.1)}$$

The immediate transitions t_3, t_4 and t_5 model the selection of the receiver's input buffer. At certain markings more than one i.e either t_3, t_4 or t_3, t_5 will be enabled indicating that the message may be directed to a free receiver's input buffer or can queue for a busy input buffer. To model this we have to assign probabilities of firing to these transitions, when they are enabled together.

This probability will depend on the marking in which this immediate transition gets enabled. Thus the switching distribution for each immediate transition in terms of the tokens at their input places can be determined. The switching distribution for the transitions t_3, t_4 , t_5 and t_9 are as given below:

For t_3 : m_6/PROCS,

For t_4 : m_7/PROCS,

For t_5 : m_9/PROCS,

For t_9: when t_3 is enabled , m_{10}/PROCS

Otherwise 1.

where PROCS is the number of processors in the model.

The above switching distributions holds good for all marking in the reachability set of the GSPN.

Fig 5.6 shows the reduced GSPN model including the send and message queues(GSPNRQ2).This GSPN model has 17 places,8 timed transitions and 8 immediate transitions. The operation of this model is essentially same as the GSPNR2. The extra information contained is the queueing of messages at the sending and receiving ends. This aspect is modeled by the immediate transitions t_2 and t_{14}. The immediate transition t_2 models the message queueing at the senders' side and t_{14} models the reception of the messages contained in the receive message queue. Thus the message generation and placing the generated message in the sender processor send queue are modeled by the timed transition t_1 and

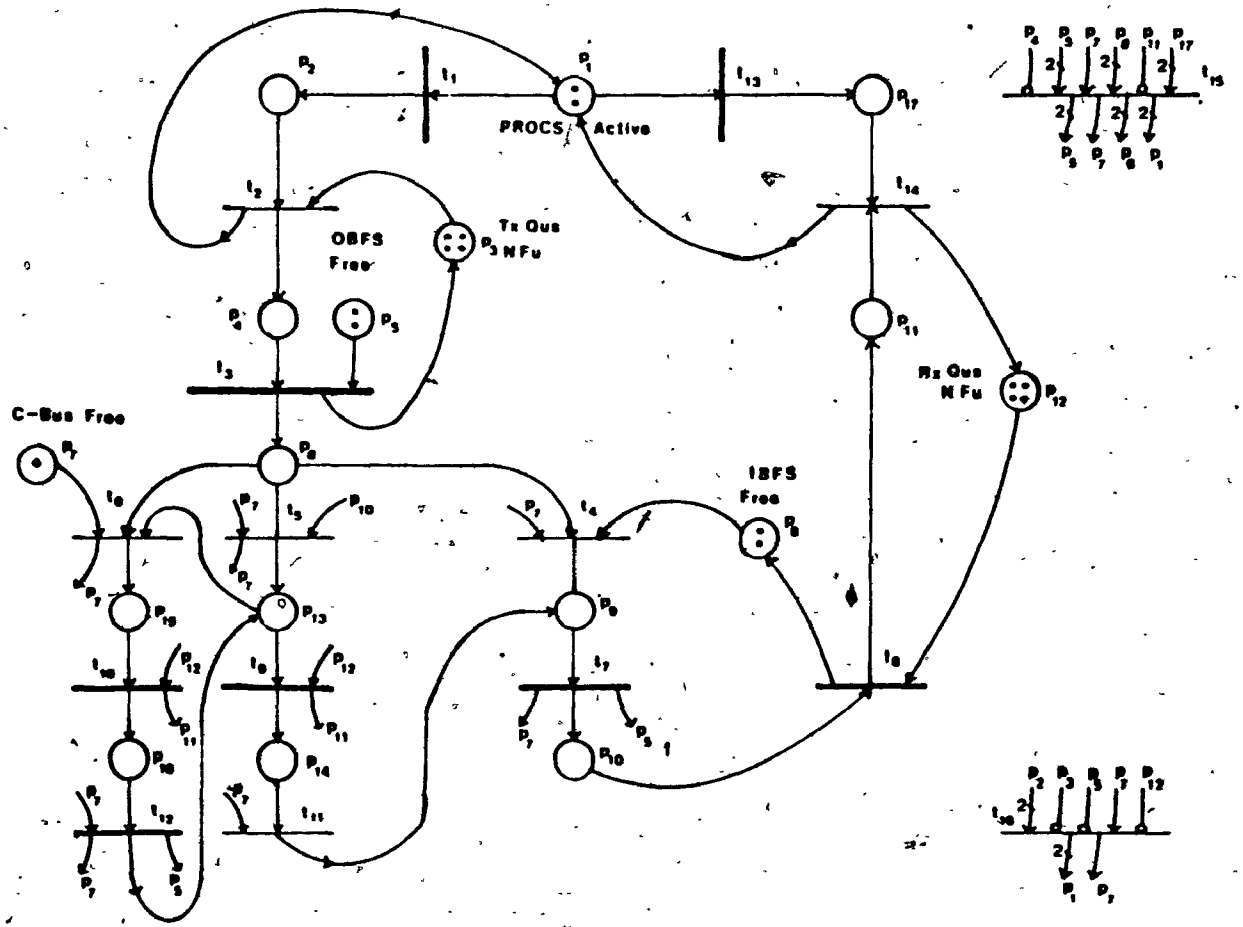


Figure 5.6 Two processor (with message queues)

Aggregated GSPN model - GSPNRQ2

immediate transition t_2. Timed trasnition t_3 models the movement of the message from the send queue to the output buffer of the sender processor.

For the sake of simplicity, in this model we have assumed that the interaction of the processor is minimal during the movement of the message from the send queue to the output buffer. Alternately, this model also represents the situation if a coprocessor, like the 82586 LAN communication controller [INTEL-84], were to be present to take care of the message communication functions of the processor.

The immediate transitions t_4, t_5 and t_6 model the selection of free or busy destination receiver for message transfer. Timed transition t_7 models the message transmission by the C-Bus controller to a free receiver's input buffer. The timed transition t_8 represents the movement of message from the receiver's input buffer to its receive queue.

The timed transition t_{13} and immediate transition t_{14} together model the reception of a message in the receive queue by a processor.

The timed transitions t_9 and t_{10} model the same operation as done by the timed transition t_8 with the additional information that one and two output buffers are queued, respectively, for that busy receiver's input buffer.

The timed transition t_{12} represents the transfer of a message from one of the two queued output buffers to that receiver's input buffer.

The immediate transition t_{11} models the selection of the queued output buffer message for transfer by the C-Bus controller to the required receiver's input buffer.

The immediate transitions t_{15} and t_{16} resolve the deadlock situations similar to t_{13} and t_{14} in GSPNR2.

The firing rates of the timed transitions are similar to that in GSPNR2. The timed transitions t_1 and t_{13} fire at a marking dependent rate $m_1 X \lambda/2$, the timed transitions t_7 and t_{12} fire at a constant rate μ_C and the transition t_3 fires at the marking dedpendent rate $m_5 X \alpha_T$. Transitions t_8, t_9 and t_{10} fire at a marking dependent rates $m_{10} X \alpha_R$, $m_{13} X \alpha_R$ and $m_{15} X \alpha_R$ repectively. The rates μ_C, α_T, α_R have the same relation as given in equation 5.1. As it happens in GSPNR2, in this model also in certain markings more than one immediate transition can get enabled at the same time. The immediate transitions t_2, t_4, t_5 and t_6 will get enabled singly or in groups of two or three.Also the immediate transitions t_{11} and t_{14} get enabled simultaneously in certain markings.

In these groups, for the sake of simplicity, we assume that the immediate transition modeling the processor activity will get priority. For example, if immediate

transitions t_2 and t_6 get enabled in a given marking then a probability of 1 is assigned to the firing of t_2 while a probability of 0 is assigned to t_6. With this assumptions the switching distributions for the various immediate transitions are as follows:

t_2 and t_{14} will fire with probability 1,

t_4 : when t_2 enabled : probability =0,

otherwise probability = m_8/PROCS,

t_5: when t_2 is enabled : probability = 0,

otherwise, probability = m_{10}/PROCS,

t_6: when t_2 enabled : probability = 0,

otherwise, probability = m_{13}/PROCS.

t_{11}: when t_{14} enabled: probability =0,

otherwise, probability =1,

where m_i is the number of tokens in place p_i and PROCS= number of processors in the model.

The extension of the reduced two processor GSPN models (GSPNR2,GSPNRQ2) to three and higher number of processors is quite stright forward. Fig 5.7 shows the three processor version of GSPNRQ.Here the processors activities portion do not get altered, only the number of tokens in the initial marking gets changed. The portion modeling the C-Bus controller needs to be extended to queue one more processor. In fig 5.7 the places p_{18},p_{19},p_{20}, the timed transitions t_{18},t_{20} and the immediate transitions t_{17},t_{19} represent this extension of C-Bus queue (shown enclosed in dotted lines).

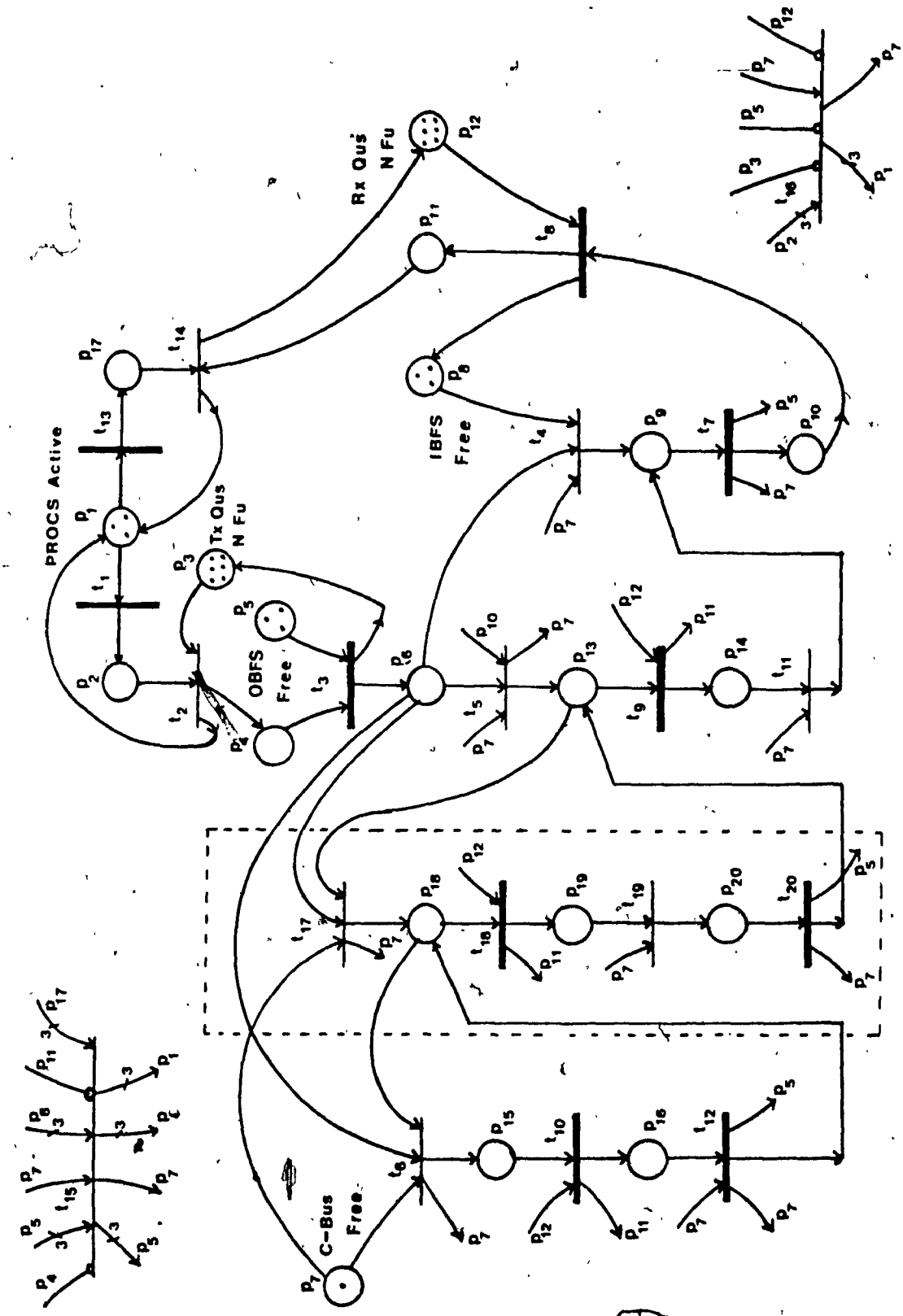


Figure 5.7 Three Processor version of GSPNRQ

Thus this three processor aggregated GSPN model has 20 places, 10 timed and 10 immediate transitions. To model a system with "i" processors this portion of the C-Bus controller queues must be extended (i-2) times. Thus for modeling a 5 processor CUENET we need three such extra portions and thus the total number of places will come to 26, timed transtions 14 and immediate transitions also to 14.

5.7 Discussion of Agrregated GSPN models of CUENET:

Table 5.7 gives the details regarding the size of the aggregated GSPN models and the size of their reachability set. By comparing the entries in Table 5.7 with those in Table 5.1 we observe that the aggregation has resulted in smaller and compact GSPN models for the two processor CUENET. The savings one obtains in terms of places and transitions is quite significant as one progresses from two processor to three processor aggregate model with message queues.

But the overall reachability set for both the aggregate and detailed models are of the same magnitude. This is due to the fact that with the aggregate model we are still presenting the same or greater amount of details of the system. Hence the reason for the same size in the reachability sets.

TABLE 5.7

DETAILS OF MODEL AND REACHABILITY SET SIZE
FOR AGGREGATE GSPN MODELS:

GSPN MODEL	NO. OF PLACES	NO. OF TMD TRS	NO. OF IMM TRS	SIZE OF R S	NO. OF T St	NO. OF V St
GSPNR2	13	8	6	129	90	39
GSPNRQ2 (Qu Sz=1)	17	8	8	876	342	534
GSPNRQ2 (Qu Sz=2)	17	8	8	2012	708	2012
GSPNRQ2 (Qu Sz=3)	17	8	8	3616	1202	2414
TOMP	9	6	5	24	16	8

NOTE:

TMD TRS : Timed Transitions.
IMM TRS : Immediate Transitions.
R S : Reachability Set.
T St : Tangible States.
V St : Vanishing States.
Qu Sz: Send and Receive message queue size.

TABLE 5.8

COMPUTATIONAL TIMES FOR GSPN ANALYSIS OF AGGREGATE GSPN MODELS:

MODEL	T1	T2	T3
GSPNR2	2.89	2.40	0.5
GSPNRQ2 (Qu Sz=1)	22.74	2716	8.5
GSPNRQ2 (Qu Sz=2)	55.0	6000 *	20.0
GSPNRQ2 (Qu Sz=3)	96.0	9000 *	100.0

NOTE:
T1: Time in secs to determine the reachability set and form transition matrix.
T2: Time in secs to determine the solution from the transition matrix by Gaussian ellimination.
T3: Time in secs to determine the solutionn from the transition matrix by Gauss-Seidel iterative method.
* : Estimated value.
Qu Sz : Send and receive message queue size.

The last entry in Table 5.7 corresponds to the reachability set of the reduced GSPN model of TOMP multiprocessor, modeling 5 processors, 3 common memories and 2 buses. Here again, as seen in SPN analysis, the loose coupled nature of CUENET results in a larger size reachability sets.

Table 5.8 gives the computational time required for doing GSPN analysis of these aggregate models.

From these results, we see that the aggregate GSPN model does not provide much computational advantage. But it does provide the advantage of compactness and ease of expansion, as illustrated in the previous section, to represent the system with higher number of processors.

Table 5.9 gives the GSPN analysis results of GSPNR2, while tables 5.10 and 5.11 contain the results of GSPNRQ2. The Table 5.9 gives the GSPN analysis results of GSPNR2, which is the agregate version of the two processor model GSPN2B. Here the different aspects of message communication are modeled in more detail as compared to GSPN2B. The size of average inbuffer queue length is due to the fact in this model, the situation of processor sending a message to itself is also modeled. When a processor is sending a message to itself, the other processors will have to wait for an extra amount of time, which is reflected by the average size of the inbuffer queue.

TABLE 5.9

GSPN ANALYSIS RESULTS OF GSPNR2:

ρ	λ	μ	PP	CT	CBQ	IBFQ
0.01	0.417	41.66	1.6887	0.2364	0.0000	0.5618
0.05	2.083	41.66	1.6662	1.1684	0.0000	0.5568
0.1	4.166	41.66	1.6386	2.3034	0.0002	0.5506
0.2	8.332	41.66	1.5848	4.4751	0.0001	0.5390
0.4	16.664	41.66	1.4837	8.4406	0.0003	0.5181
0.6	24.996	41.66	1.3914	11.9441	0.0007	0.4999
0.8	33.328	41.66	1.3077	15.0424	0.0012	0.4840
1.0	41.66	41.66	1.2317	17.7877	0.0016	0.4700

NOTE:
PP : Processing Power.
CT : C-Bus Throughput in messages/sec.
CBQ : C-Bus average queue length.
IBFQ : Input Buffers average queue length

TABLE 5.10

GSPN ANALYSIS RESULTS OF GSPNRQ2:
(Processing Power for different queue sizes.)

ρ	λ	μ	PP1	PP2	PP3
0.01	0.417	41.66	1.8093	1.8617	1.8916
0.05	2.083	41.66	1.8062	1.8594	1.8898
0.1	4.166	41.66	1.8017	1.8562	1.8872
0.2	8.332	41.66	1.7908	1.8484	1.8810
0.4	16.664	41.66	1.7610	1.8279	1.8647
0.6	24.996	41.66	1.7216	1.8013	1.8436
0.8	33.328	41.66	1.6739	1.7690	1.8180
1.0	41.66	41.66	1.6200	1.7311	1.7881

NOTE:
PP1 : Processing Power for Queue size= 1 message.
PP2 : Processing Power for Queue size = 2 messages.
PP3 : Processing Power for Queue size = 3 messages.

TABLE 5.11

GSPN ANALYSIS RESULTS OF GSPNRQ2:
(C-Bus Throughput for different queue sizes.)

ρ	λ	μ	CT1	CT2	CT3
0.01	0.417	41.66	0.2664	0.2793	0.2867
0.05	2.083	41.66	1.3377	1.4045	1.4424
0.1	4.166	41.66	2.6874	2.8270	2.9062
0.2	8.332	41.66	5.4121	5.7171	5.8894
0.4	16.664	41.66	10.8797	11.6106	12.0159
0.6	24.996	41.66	16.2352	17.5440	18.2524
0.8	33.328	41.66	21.3500	23.4014	24.4940
1.0	41.66	41.66	26.1281	29.0713	30.6399

NOTE:
CT1 : C-Bus Throughput in messages/sec for queue size = 1 message.
CT2 : C-Bus Throughput in mesages/sec for queue size = 2 messsages.
CT3 : C-Bus Throughput in messages/sec for queue size = 3 messages.

The improvement in the processing power and C-Bus throughput with the inclusion of message queues at the send and receive points is indicated in the results presented in Tables 5.10 and 5.11 respectively.

The study of these detailed and aggregate GSPN models of CUENET suggests that the GSPN modeling technique can be utilized to study a small loosely coupled DCS, say with two or three processors.

CHAPTER VI

CONCLUSIONS.

This thesis aims at the performance evaluation of CUENET (Concordia University Educational NETwork) a loosely coupled Distributed Computing System (DCS), using Stochastic Petrinets(SPNs) and Generalized Stochastic Petrinets (GSPNs).

Of late, SPNs and GSPNs are employed in the performance studies of tightly coupled DCS like the Torino MultiProcessor(TOMP) [MARSAN-83], and the Fault Tolerant MultiProcessors(FTMP) [WOOD-84]. These recent applications of SPNs and GSPNs on one hand and the presence of CUENET here in our department on the other hand are the main motivating factors in undertaking this thesis work. Also, to the author's knowledge, to date, these techniques have not been employed in the performance study of a loosely coupled DCS.

The SPN models, presented in chapter IV, for the two and three processor CUENET reveal two aspects:

(1) The ease with which concurrency and conflicts in the system can be modeled. Also using the Petrinet model properties like boundedness and liveness, the modeled system can be easily checked for deadlock.

(2) The reachability set of the SPN grows enormously as one proceeds to add more and more details of the system into the

model. Table4.1 illustrates this increase.

The large size of the reachability set reported in this thesis is attributable to the loose coupling and the protocol details employed in a network such as CUENET. The autonomous and asynchronous nature of a loosely coupled system results in a large number of system states. This contrasts with the size of the SPNs and reachability sets of the tightly coupled TOMP shown in Table 4.3. The large size of the reachability set results in high computational times to obtain the steady state solution of the underlying discrete Markov chain.

The classification of transitions into timed and immediate 'types' in GSPN allows one to express more details of the system for the same time complexity of the model. The GSPN models of chapter V illustrate this aspect. Table 5.1 gives the details of the reachability sets of the GSPN models. The models GSPN2A, GSPN2B have lesser number of states as compared to their SPN counterparts, where the level of details are the same. The GSPN2C models the two processor CUENET with additional details of message queueing at each processor. This has far lesser number of states as compared to an equivalent SPN. Also the results of the two processor GSPN models GSPN2A and GSPN2B compare well with the results of the corresponding SPN models SPN2A and SPN2B, as indicated by the Tables 5.3,5.4 and Tables 4.4,4.5 respectively.

The main advantage of aggergate models is in their compactness and the ease of expansion of the model to represent additional processors. However, in an aggregate model one cannot get the details of the activities of individual processors. The aggregate GSPN models for two processor CUENET are shown in figures 5.5,5.6; and for three processor CUENET is shown in fig 5.7. Table 5.7 gives the details of the reachability sets of the aggregate models. Here also we see the fast growth of the reachability set with addition of more system details in the model.

For the analysis of GSPN models, the software package supplied by Dr. M.A.Marsan of Politechniquo de Torino, Torino, Italy [MARSAN-85b] was used. This software was run on VAX 11/780. The SPN models were analysed by means of a set of programs developed by the author on Cyber 835. The computational times for GSPN models were given in tables 5.2, 5.8 and for SPN models in table 4.2. From these results it is observed that analysis of loosely coupled systems, even with as few as two processors, incurrs a substantial amount of increase in the computational times with addition of more details of the system. In the GSPN analysis of the two processor model, the computation time varies from 0.5 secs to 100 secs with the addition of message queues at each processor. Hence the analysis of tens and hundreds of processors would be impractical in this way.

For the analysis and modeling of loosely coupled systems involving hundreds of processors, like the Cosmic Cube [SEITZ-85], methodologies of applying GSPN models have to be evolved in future. It is an open problem worth future study. One way to study them could be to decompose a large system into a set of small and computationally managable sizes and solve this set, perhaps using a DCS.

REFERENCES

[AGERWA-79] Tilak Agerwala -Putting Petrinets to work - IEEE Computers, December 1979,P85-94.

[AGERWA-83] A.K.Agerwala and U.Herzog--Performance Evaluation of of Multiple Processor Systems- IEEE Trans. on Computers, Vol C-32, No.1, January 1983, P2-3

[AMER-82] P.D.Amer-A Measurement centre for NBS Local Area Network- IEEE Trans. on Computers, Vol.c-31,August 1982, P723-9

[BUZEN-77] J.P.Buzen-Principles of Computer Performance modeling and Prediction - Performance Modeling and Prediction, Infotech State of art report- 1977, P3-18.

[CHANDY-81] K.Manichandy and Charles.H.Sauer-Computer Systems Performance Modeling- Prentice Hall Inc(1981), Engleswood Cliffs-New jersey 07632.

[CINLAR-75] Erhan.Cinlar-Introduction to Stochastic Processes- Prentice Hall, 1975, Engleswood Cliffs,New Jersey,07632.

[CLARK-78] David.Clark Et Al -An Introduction to Local Area Networks- Proceedings of IEEE, Vol.66, No.11, November 1978, P1497-1517

[COTTON-80] Ira.W.Cotton-Technologies for Local Area Computer Networks- Computer Networks, No.4(1980), P197-208.

[CONTE-81] G.Conte et al-A Multiprocessor prototype- Euromicro Symposium, Paris, Sept 1981, North Holland,P401-410.

[ENSLOW-77] Philip.H.Enslow Jr.- Multiprocessor Organization:A Survey- Computing Surveys, Vol.9, No.1,March 1977, P103-129.

[ENSLOW-78] Philip.H.Enslow Jr.- What is a "Distributed Data processing system?" -IEEE Computer, January 1978,P3-11.

[FATHI-83] Eli.T.Fathi and Moshe Kriger,-Multiple Microprocessor Systems:What, Why and When?- IEEE Computer, March 1983, P23-32.

[FULLER-78] S.H.Fuller et al -Multi-microprocessors:An overview and working example- Proceedings of IEEE, Vol.66, No.2, February 1978, P216-228.

[FROMM-83] H.Fromm et al - Experiment with Performance measurement and modeling of a Processor array- IEEE Trans. on Computers, Vol.C32,no.1, January 1983, P15-31.

[GAJSKI-85] Daniel.D.Gajski -Esential issues in Multiprocessor Systems- IEEE Computer, June 1985, P9-27.

[GROSS-82] Clifford.Grossner - The design and implementation of CUENET, a reconfigurable network of loosely coulped Microcomputers - M.Comp.Sc Thesis, 1982, concordia University, Montreal, Canada.

[GUPTA-84] Amar Gupta and Hoo-min.D. Toong-The first decade of personal computers- IEEE Proceedings, Vol.72, No.3, March 1984, P246-258

[HOEL-72] Paul.G.Hoel -Introduction to Stochastic Processes- Houghton Miffin Comapny, Boston, 1972.

[HURA-81] G.S.Hura et al -State equation representation of logic operations through Petrinets- Proceedings of IEEE, Vol.69,No.2,April 81, P485-487.

[INTEL-84] Intel LAN Component user's Manual-March 1984.

[KLIEN-75] L.Kleinrock -Queueing Systems-Volume I:Theory, John Wiley and Sons, Newyork, 1975.

[KUCK-77] David.J.Kuck - A survey of parallel machine organization and programming- Cpmputing Survey, Vol.9, No.1, March 1977, P29-60.

[LAUTEN-74] Kurt Lautenbach et al -Use of Petrinets for proving correctness of concurrent process systems- Proceedings of IFIP congress, 1974, North Holland Publishing, 1974, P187-191.

[LIEBO-81] Burt.H.Leibowitz and John.H.Carlsan, Editors - Tutorial:Distributed Processing- IEEE Computer Society, 1981.

[MARSAN-83] M.A.Marsan et al -Modeling Bus contention and Memory interference in a Multiprocessor system-IEEE Trans. on Computers, Vol.C-32, No.1, January 1983, P60-71.

[MARSAN-84] M.A.Marsan et al -A class of Generalized Stochastic Petrinets for the Performance evaluation of Multiprocessor systems- ACM Trans. on computer Systems,Vol.2,No.2, May 1984,P93-122.

[MARSAN-85a] M.A.Marsan et al -Construction of Generalized Stochastic petrinets models of Bus oriented Multiprocessor systems by stepwise refinements :case study- Presented in the International Workshop on Timed Petri Nets,held at Torino,Italy,1985.

[MARSAN-85b] M.A.Marsan et al-A software package for the analysis of Generalized Stochastic Petrinets- Presented in the International Workshop on Timed Petri Nets, held in Torino,Italy, 1985.

[MEDHI-82] J.Medhi-Stochastic Processes- Wiley Eastern Limited, New Delhi, India, 1982.

[MOLLOY-81] M.K.Molloy -On the integration of Delay and Throughput measures in Distributed Processing Models- Doctoral Thesis dissertation,

University of california, Los Angeles,1981.

[MURATA-84] Tadao Murata -Modeling and analysis of concurrent systems- Van Nostrand Reinhold Co., 1984.

[NUTT-75] J.Nutt A Tutorial: Computer system Monitoring- IEEE Computer, November 1975, P51-60.

[PETER-81] J.L.Peterson -Petrinet Theory and the modeling of systems- Prentice Hall Inc,1981, Engleswood Cliffs, New Jersey, 07632.

[RAU-81] Nicholas Rau- Matrices and mathematical Programming - St. Martin's Press, Newyork, 1981.

[REID-82] Lorreta Guarino Reid- Control and communication in programs- UMI Research Press, 1982.

[SATYA-80] M.Satyanarayana - Commercial Multiprocessing Systems - IEEE Computer, May 1980, P75-96.

[SEGALL-83] Z.Segall et al -An Integrated Instrumentation Enviornment for Multiprocessors - IEEE Trans. on Computers, Vol.C-32, No.1, January 1983, P 4-14.

[SEITZ-85] Charles.L.Seitz- The Cosmic Cube- Communications of the ACM, Vol.28, No.1, January 1985, P22-33

[SHAPI-81] S.D Shapiro - A Stochastic Petrinet with applications to modeling occupancy times for the concurrent task systems- Networks, Vol.9, 1979, P 375-379.

[SVOBOD-76] Liba Svobodova - Computer performance measurement and Evaluation methods: Analysis and applications - Elsevier Computer Science Library Edition, 1976, American Elesevier publishing Co. Inc.

[TAMIR-83] Gopalachary Tamirisa - Application of Parallel processor system to Quicksort- M.S. Thesis Report, 1983, McGill University, Montreal, Canada.

[TANENB-81] A.S.Tanenbaum -Computer Networks- Prentice Hall Inc,1981, Engleswood Cliffs, New Jersey, 07632.

[TSAO-84] C.David Tsao - A Local Area Network Architecture overview - IEEE Communications Magazine, Vol.22, No.8, August 1984, P7-11.

[VARGA-62] Richard.S.Varga - Matrix Iterative Analysis - Prentice Hall, 1962, Engleswood Cliffs, New Jersey, 07632.

[WEITZ-80] Cary Weitzman - Distributed Micro /Minicomputer Systems - Structures, implementation and applications - Prentice Hall Inc, 1980, Engleswood Cliffs, N.J., 07632.

[WITTIE-80] Larry.d.Wittie et al - MICROS: A distributed operating systems for MICRONET, A reconfigurable Network Computer - IEEE Trans. on Computers, Vol.C-29, No.2, December 1980, P 1133-1144.

[WOOD-84] Michel.H.Woodbury et al - Performance modedling of Real-time Multiprocessors with time-shared buses - Computer Research Labaratory, Report No. CRL-TR-46-84, University of Michgan, Ann Arbor, MI.

[ZUBER-80] W.M.Zubereck - Timed Petrinets and Priliminary Performance evaluation - Proceedings of the 7th Annual Symposium on Computer Architecture, 1980, P88-96.

APPENDIX A:

Program Listing for Determining the SPN rechability set using Prime number method.

```
      PROGRAM REACPET ( INPUT,OUTPUT )
C THIS PROGRAM TAKES THE DESCRIPTION OF A STOCHASTIC PETRINET AND DERIVES
C THE REACHABILITY SET FOR IT.
C
C
      REAL PROB(200)
      DIMENSION RTRANS(50),DENS(15),NPRIME(50),NSTATE(2000),NINPUT(50)
      DIMENSION NTEMP(50),NOUT(50)
      DATA (NPRIME(I), I=1,50) /2,3,5,7,11,13,17,19,23,29,31,
     1                          37,41,43,47,53,59,61,67,71,
     2 73,79,83,89,97,101,103,107,109,113,127,131,137,139,
     3 149,151,157,163,167,173,179,181,191,193,197,199,211,
     4 223,227,229/
      PRINT 10
10    FORMAT ('1','THE FOLLOWING IS THE DESCRIPTION OF THE',
     2 ' GIVEN PETRINET.')
      READ 11,NPLACE
11     FORMAT( I8 )
      PRINT 15,NPLACE
15    FORMAT('0',' THE NUMBBER OF PLACES FOR THE GIVEN',
     2  ' PETRINET IS ', I5 )
      READ 11,NTRANS
      PRINT 20,NTRANS
20     FORMAT ('0','THE NUMBER OF TRANSITIONS FOR THE GIVEN',
     2 ' PETRINET IS ',I5 )
C
C LOOP THROUGH THE TRANSITIONS CREATING THE INPUT AND OUTPUT .
C
      PRINT 22
22    FORMAT('1','FOLLOWING IS THE DESCRIPTION OF THE INPUT AND',
     2 ' OUTPUT PLACES FOR EACH TRANSITION IN THE NET.')
      DO 50, I=1,NTRANS
      NINPUT(I)=1
      NOUT(I)=1
25    READ 11,NP
      PRINT 30,I,NP
30    FORMAT ('0','THE INPUT PLACEFOR THE TRANSITION',I5,
     2 ' IS ',I5)
      IF (NP.GT.NPLACE) GO TO 25
      IF (NP.LE.0) GO TO 35
      NINPUT(I)= NINPUT(I) * NPRIME(NP)
      GO TO 25
35    READ 11,NP
      PRINT 40,I,NP
40    FORMAT('0',' THE OUTPUT PLACEFOR TRANSITION ',I5,
     2 ' IS ',I5)
      IF (NP.GT.NPLACE) GO TO 35
      IF (NP.LE.0) GO TO 50
      NOUT(I) = NOUT(I) * NPRIME(NP)
      GO TO 35
50    CONTINUE
      PRINT 23
23    FORMAT('1',' THE AVERAGE FIRING RATES FOR THE TRANSITION ARE: ',/)
      DO 54 I=1,NTRANS
```

```
      READ 12,RTRANS(I)
12    FORMAT(F8.4)
      PRINT 53,I,RTRANS(I)
53    FORMAT ('0','THE AVERAGE TRANSITION RATE OF TRANSITION',I4 ,
     2 ' IS ',F10.4 )
54     CONTINUE
      PRINT 55
55     FORMAT('0','END OF THE DESCRIPTION OF THE PETRINET. ')
      PRINT 60
60    FORMAT ('1','DESCRITION OF THE  INITIAL MARKING OF THE NET. ')
      NSTATE(1)= 1
      DO 65, I= 1,NPLACE
      READ 11,NTOKEN
      PRINT 61,I,NTOKEN
61    FORMAT('0',' THE PLACE ',I5, ' HAS ',I5, ' TOKENS IN IT.')
      NSTATE(1) = NSTATE(1)*NPRIME(I)**NTOKEN
65    CONTINUE
C
C NOW GENERATE THE VARIOUS REACHABLE STATES.
C
      NUMST=1
67     NEWFLG = 0
      LIMIT = NUMST
       DO 80, M=1,LIMIT
      TERNODE = 0
       DO 70, J=1,NTRANS
      NTEMP(J)=0
      IF (MOD(NSTATE(M),NINPUT(J)). NE.0) GO TO 70
      TERNODE = 1
      NTEMP(J) = NSTATE(M)/NINPUT(J)*NOUT(J)
70    CONTINUE
      IF (TERNODE .NE. 0) GO TO 71
      PRINT 72,M
72    FORMAT('0','THE MARKING WITH STATE',I3,' IS A',
     2 ' TERMINAL MARKING.')
C
C PUT THE NEW STATE IN WITH THE OLD IN ORDER.
C
71    DO 80, J=1,NTRANS
      IF(NTEMP(J).EQ.0) GO TO 80
      DO 75 , I= 1,NUMST
      IF (NTEMP(J).EQ.NSTATE(I)) GO TO 80
75    CONTINUE
      NUMST = NUMST + 1
      NSTATE(NUMST) = NTEMP(J)
      NEWFLG = 1
80    CONTINUE
C
C SEE IF ANY NEW STATES.
C
       IF (NEWFLG.NE.0) GO TO 67
      PRINT 85, NUMST
85     FORMAT('0','THERE ARE ',I5,'  STATES IN THE REACHABILITY ',
     2 ' SET FOR THE GIVEN PETRINET.')
```

```
      PRINT 95
95     FORMAT('1','THE MARKINGS CORRESPONDING TO THE VARIOUS   ',
     3 ' STATES IN  REACHABILITY SET OF THE GIVEN '/
     4 ' PETRINET ARE AS SHOWN.')
       DO 110 I= 1,NUMST
      DO 115, J= 1,NPLACE
      NTEMP(J)= 0
      NUMB = NSTATE(I)
120   IF (MOD(NUMB,NPRIME(J)) .NE. 0) GO TO 115
C
C COUNT THE TOKENS AND REMOVE THEM.
C
      NTEMP(J) = NTEMP(J) + 1
      NUMB = NUMB/NPRIME(J)
       GO TO 120
115    CONTINUE
      PRINT 125, I,(NTEMP(J), J=1,NPLACE)
125   FORMAT ('0','STATE ',I3,':',40I3)
110   CONTINUE
      STOP
      END
```

APPENDIX B:

Program Listing for determining the SPN reachability set using Binary number method.

```
      PROGRAM BIREPET (INPUT,OUTPUT)
      REAL PROB(200)
      DIMENSION NSTATE(6000),NINPUT(100),NOUT(100),NTEMP(1000),
     1 RTRANS(100)
      STATIME = SECOND()
      PRINT 10
   10 FORMAT('1',' THE FOLLOWING IS THE DESCRIPTION OF THE ',
     2 ' GIVEN PETRINET. ')
      READ 15,NPLACE
   15 FORMAT(I8)
      PRINT 20,NPLACE
   20 FORMAT('0',' THE NUMBER OF PLACES FOR THE GIVEN PETRINET IS ',I5)
      READ 15,NTRANS
      PRINT 25,NTRANS
   25 FORMAT('0',' THE NUMBER OF TRANSITIONS FOR THE GIVEN ',
     2 ' PETRINET IS ',I5)
      PRINT 30
   30 FORMAT('1',' FOLLOWING IS THE DESCRIPTION OF THE INPUT AND',
     3 ' OUTPUT PLACES FOR EACH TRANSITION IN THE PETRINET. ')
      DO 35, I= 1,NTRANS
      NINPUT(I) = 0
      NOUT(I) =0
   40 READ 15,NP
      PRINT 45, I,NP
   45 FORMAT('0',' THE INPUT PLACE FOR TRANSITION ',I5,
     4 ' IS ',I5)
      IF(NP.LE.0) GO TO 50
      IF(NP.GT.NPLACE) GO TO 40
      NINPUT(I) = NINPUT(I) + 2**(NP-1)
      GO TO 40
   50 READ 15,NP
      PRINT 55,I,NP
   55 FORMAT('0',' THE OUTPUT PLACE FOR TRANSITION ',I5,
     4 ' IS ',I5)
      IF(NP.LE.0) GO TO 35
      IF(NP.GT.NPLACE) GO TO 50
      NOUT(I) = NOUT(I) + 2**(NP-1)
      GO TO 50
   35 CONTINUE
      DO 51,I= 1,NTRANS
      PRINT 60,I,NINPUT(I),NOUT(I)
   60 FORMAT('0',' FOR TRANSITION ',I5,' NINPUT= ',I20,
     4 ' NOUT= ',I20)
   51 CONTINUE
      PRINT 36
   36 FORMAT('1', ' THE AVERAGE FIRING RATES FOR THE TRANSITIONS ARE:'/)
      DO 37, I= 1, NTRANS
      READ 16,RTRANS(I)
   16 FORMAT(F8.4)
      PRINT 38,I,RTRANS(I)
   38 FORMAT('0',' THE AVERAGE FIRING RATE OF TRANSITION ',
     2 I5,' IS ',F15.5)
   37 CONTINUE
      PRINT 65
```

```
65    FORMAT('0',' END OF DESCRIPTION OF THE GIVEN PETRINET. ')
      PRINT 70
70    FORMAT('1',' DESCRIPTION OF THE INITIAL MARKING OF THE PETRINET.')
      NSTATE(1) = 0
      DO 75, I = 1,NPLACE
      READ 15,NTOKEN
      PRINT 80, I,NTOKEN
80    FORMAT('0',' THE PLACE ',I5,' HAS ',I4,' TOKENS IN IT.')
      NSTATE(1)= NSTATE(1) + NTOKEN*(2**(I-1))
      PRINT 76,NSTATE(1)
76    FORMAT('0','THE VALUE OF NSTATE(1) IS ',I20)
75    CONTINUE
C
C NOW GENERATE THE VARIOUS REACHABLE STATES.
C
      NUMST = 1
85    NEWFLG = 0
      IF(NUMST.EQ.1) ISTART = 1
      LIMIT = NUMST
      PRINT 86,LIMIT
86    FORMAT('0',' LIMIT = ',I5)
      DO 90, M= ISTART,LIMIT
      TERNODE = 0
      DO 95, J= 1, NTRANS
      NTEMP(J) = 0
      IAND = NSTATE(M).AND.NINPUT(J)
      INEQ = IAND.NEQV.NINPUT(J)
      IF(INEQ.NE.0) GO TO 95
      TERNODE = 1
      INTRES = NSTATE(M).AND.(.NOT.NINPUT(J))
      NTEMP(J) = INTRES.OR.NOUT(J)
95    CONTINUE
      IF(TERNODE.NE.0)GO TO 100
      PRINT 105,M,NSTATE(M)
105   FORMAT('0',' THE MARKING WITH STATE ',I4,' IS A ',
     5 ' TERMINAL MARKING. AND NSTATE(M) = ',I10)
100   DO 110,J= 1,NTRANS
      IF(NTEMP(J).EQ.0) GO TO 110
      DO 115, I= 1,NUMST
      IF(NTEMP(J).EQ.NSTATE(I)) GO TO 110
115   CONTINUE
      NUMST = NUMST + 1
      NSTATE(NUMST)= NTEMP(J)
      NEWFLG = 1
110   CONTINUE
90    CONTINUE
C
C SEE IF ANY NEW STATES.
C
      IF(NEWFLG.NE.0) THEN
      ISTART = LIMIT
      PRINT 300,ISTART
300   FORMAT('0',' ISTART= ',I4)
      GO TO 85
```

```
      ENDIF
      PRINT 120,NUMST
120   FORMAT('1',' THERE ARE ',I5,' STATES IN THE REACHABILITY ',
     4 ' SET OF THE GIVEN PETRINET. ')
      DO 125, I= 1,NUMST
      PRINT 130,I,NSTATE(I)
130   FORMAT('0',' THE STATE NUMBER OF THE MARKING ',I5,
     5 ' IS ',I20)
125   CONTINUE
C
C DETERMINE THE TOKENS IN VARIOUS PLACES FOR EACH STATE AND PRINT THEM...
C
      DO 140,I=1, NUMST
      NUMB = NSTATE(I)
      DO 145, J= 1,NPLACE
      NTEMP(J) = 0
      IF(NUMB.EQ.1) GO TO 155
      IF(NUMB.EQ.0) GO TO 145
      IODD = MOD(NUMB,2)
      IF(IODD.EQ.1) GO TO 155
      NUMB= NUMB/2
      GO TO 145
155   NUMB = NUMB - 1
      NUMB = NUMB/2
      NTEMP(J) = NTEMP(J) + 1
145   CONTINUE
      PRINT 160, I,(NTEMP(J),J= 1,NPLACE)
160   FORMAT('0',' STATE ',I3,' : ',40I3)
140   CONTINUE
      ONETIME = SECOND()
      TMEINT = ONETIME - STATIME
      PRINT 500,TMEINT
500   FORMAT('0',' THE TIME FOR DETERMINING THE REACHABILITY ',
     2 ' SET IS:',F15.6)
      STOP
      END
```

APPENDIX C:

Program Listing for determining the GSPN reachability set using Hexadecimal number method.

```
      PROGRAM BIREPET (INPUT,OUTPUT)
      DIMENSION NSTATE(4000,3),NINTIM(50),NOUTIM(50),NTEMP(50,3),
     1 RTRANS(50),NTKTMP(50),NINIMM(50),NOUIMM(50),
     4 NSTANG(4000,3),NSTVAN(4000,3)
      STATIME = SECOND()
      PRINT 10
10    FORMAT('1',' THE FOLLOWING IS THE DESCRIPTION OF THE ',
     2 ' GIVEN GENERALIZED STOCHASTIC  PETRINET. ')
      READ 15,NPLACE
15    FORMAT(I8)
      PRINT 20,NPLACE
20    FORMAT('0',' THE NUMBER OF PLACES FOR THE GIVEN PETRINET IS ',I5)
      READ 15,NTIMTR
      PRINT 25,NTIMTR
25    FORMAT('0',' THE NUMBER  OF TIMED  TRANSITIONS FOR THE  ',
     2 '  GIVEN GENERALIZED STOCHASTIC PETRINET IS ',I5)
      READ 15,NIMMTR
      PRINT 26,NIMMTR
26    FORMAT('0',' THE NUMBER OF IMMEDIATE TRANSITIONS IN THE ',
     2 ' GIVEN GENERALIZED STOCHASTIC PETRINET IS ',I5)
      PRINT 30
30    FORMAT('1',' FOLLOWING IS THE DESCRIPTION OF THE INPUT AND',
     3 ' OUTPUT PLACES FOR  EACH TIMED  TRANSITION IN THE  ',
     4 ' GENERALIZED STOCHASTIC PETRINET..')
      DO 35, I= 1,NTIMTR
      NINTIM(I) = 0
      NOUTIM(I) =0
40    READ 15,NP
      PRINT 45, I,NP
45    FORMAT('0',' THE INPUT PLACE FOR  TIMED TRANSITION ',I5,
     4 ' IS ',I5)
      IF(NP.LE.0) GO TO 50
      IF(NP.GT.NPLACE) GO TO 40
      NINTIM(I) = NINTIM(I) + 2**(NP-1)
      GO TO 40
50    READ 15,NP
      PRINT 55,I,NP
55    FORMAT('0',' THE OUTPUT PLACE FOR  TIMED TRANSITION ',I5,
     4 ' IS ',I5)
      IF(NP.LE.0) GO TO 35
      IF(NP.GT.NPLACE) GO TO 50
      NOUTIM(I) = NOUTIM(I) + 2**(NP-1)
      GO TO 50
35    CONTINUE
      PRINT 31
31    FORMAT('0',' THE FOLLOWING IS THE INPUT AND OUTPUT PLACE ',
     2 ' DESCRIPTION FOR EACH IMMEDIATE TRANSITIONS OF THE GIVEN ',
     3 ' GENERALISED STOCHASTIC PETRINET.')
      DO 300, I= 1,NIMMTR
      NINIMM(I) = 0
      NOUIMM(I) =0
305    READ 15,NP
      PRINT 310, I,NP
310    FORMAT('0',' THE INPUT PLACE FOR  IMMEDIATE TRANSITION ',I5,
```

```
     4 ' IS ',I5)
      IF(NP.LE.0) GO TO 315
      IF(NP.GT.NPLACE) GO TO 305
      NINIMM(I) = NINIMM(I) + 2**(NP-1)
      GO TO 305
315    READ 15,NP
      PRINT 320,I,NP
320    FORMAT('0',' THE OUTPUT PLACE FOR IMMEDIATE TRANSITION ',I5,
     4 ' IS ',I5)
      IF(NP.LE.0) GO TO 300
      IF(NP.GT.NPLACE) GO TO 315
      NOUIMM(I) = NOUIMM(I) + 2**(NP-1)
      GO TO 315
300    CONTINUE
      DO 51,I= 1,NTIMTR,
      PRINT 60,I,NINTIM(I),NOUTIM(I)
60     FORMAT('0',' FOR TIMED TRANSITION ',I5,' NINTIM= ',I20,
     4 ' NOUTIM= ',I20)
51     CONTINUE
      DO 52,I= 1,NIMMTR
      PRINT 61,I,NINIMM(I),NOUIMM(I)
61     FORMAT('0',' FOR IMMEDIATE TRANSITION ',I5,' NINIMM= ',I20,
     4 ' NOUIMM= ',I20)
52     CONTINUE
      PRINT 36
36     FORMAT('1', ' THE AVERAGE FIRING RATES FOR THE TRANSITIONS ARE:'/)
      DO 37, I= 1, NTIMTR
      READ 16,RTRANS(I)
16     FORMAT(F8.4)
      PRINT 38,I,RTRANS(I)
38     FORMAT('0',' THE AVERAGE FIRING RATE OF TRANSITION ',
     2 I5,' IS ',F15.5)
37     CONTINUE
      PRINT 65
65     FORMAT('0',' END OF DESCRIPTION OF THE GIVEN  ',
     2 ' GENERALIZED STOCHASTIC PETRINET.')
      PRINT 70
70     FORMAT('1',' DESCRIPTION OF THE INITIAL MARKING OF THE PETRINET.')
      DO 75, I = 1,NPLACE
      READ 15,NTOKEN
      PRINT 80, I,NTOKEN
80     FORMAT('0',' THE PLACE ',I5,' HAS ',I4,' TOKENS IN IT.')
      NTKTMP(I) = NTOKEN
75     CONTINUE
      PRINT 81,(NTKTMP(J), J= 1,NPLACE)
81     FORMAT('0',' INITIAL MARKING IS : ',20I5)
      CALL FORMST(NPLACE,NTKTMP,I1STAT,I2STAT,I3STAT)
      NSTATE(1,1) = I1STAT
      NSTATE(1,2) = I2STAT
      NSTATE(1,3) = I3STAT
      PRINT 76,(NSTATE(1,J),J= 1,3)
76     FORMAT('0','THE VALUE OF NSTATE(1) IS ',3I20)
C
C NOW GENERATE THE VARIOUS REACHABLE STATES FOR BOTH TIMED AND IMMEDIATE
```

```
C  TRANSITIONS OF THE GIVEN GENRALIZED STOCHASTIC PETRINET.
C
      NUMST = 1
      NIMM = 0
      NTIME = 0
 85   NEWFLG = 0
      LIMIT = NUMST
      IF(NUMST.EQ.1) ISTART = 1
      PRINT 86,LIMIT
 86   FORMAT('0',' LIMIT = ',I5)
      DO 90, M= ISTART,LIMIT
      ENAIMM = 0
      ISTMONE = NSTATE(M,1)
      ISTMTWO = NSTATE(M,2)
      ISTMTHR = NSTATE(M,3)
      CALL FNDMRK(NPLACE,ISTMONE,ISTMTWO,ISTMTHR,NTKTMP)
      DO 95, J= 1, NIMMTR
      NTEMP(J,1) = 0
      NTEMP(J,2) = 0
      NTEMP(J,3) = 0
      INVEC = NINIMM(J)
      IOUTVEC = NOUIMM(J)
          IENIMM = ITRENA(NPLACE,INVEC,NTKTMP)
      IF(IENIMM.EQ.0) GO TO 95
      ENAIMM = 1
      CALL FNDNWST(NPLACE,INVEC,IOUTVEC,NTKTMP,NTMPONE,NTMPTWO,NTMPTHR)
      NTEMP(J,1) = NTMPONE
      NTEMP(J,2) = NTMPTWO
      NTEMP(J,3) = NTMPTHR
 95   CONTINUE
      IF(ENAIMM.EQ.0) GO TO 200
      IF(NIMM.EQ.0) GO TO 210
      DO 220, JK= 1, NIMM
      IF((NSTVAN(JK,1).EQ.NSTATE(M,1)).AND.
     2 (NSTVAN(JK,2).EQ.NSTATE(M,2)).AND.
     3 (NSTVAN(JK,3).EQ.NSTATE(M,3))) GO TO 225
 220  CONTINUE
 210  NIMM = NIMM + 1
      DO 230, LK = 1,3
      NSTVAN(NIMM,LK) = NSTATE(M,LK)
 230  CONTINUE
 225   DO 240, J = 1, NIMMTR
      IF((NTEMP(J,1).EQ.0).AND.(NTEMP(J,2).EQ.0)
     2 .AND.(NTEMP(J,3).EQ.0)) GO TO 240
       DO 250, I = 1, NUMST
      IF ((NTEMP(J,1).EQ.NSTATE(I,1)).AND.
     2 (NTEMP(J,2).EQ.NSTATE(I,2)).AND.
     3 (NTEMP(J,3).EQ.NSTATE(I,3))) GO TO 240
 250  CONTINUE
      NUMST = NUMST + 1
       DO 260, L = 1,3
      NSTATE(NUMST,L) = NTEMP(J,L)
 260  CONTINUE
      NEWFLG = 1
```

```
240   CONTINUE
       GO TO 90
200    ENATIM = 0
       DO 265, J = 1, NTIMTR
      NTEMP(J,1) = 0
      NTEMP(J,2) = 0
      NTEMP(J,3) = 0
      INVEC = NINTIM(J)
      IOUTVEC = NOUTIM(J)
      IENTIM = ITRENA(NPLACE,INVEC,NTKTMP)
      IF(IENTIM.EQ.0) GO TO 265
      ENATIM = 1
      CALL FNDNWST(NPLACE,INVEC,IOUTVEC,NTKTMP,NTMPONE,NTMPTWO,NTMPTHR)
      NTEMP(J,1) = NTMPONE
      NTEMP(J,2) = NTMPTWO
      NTEMP(J,3) = NTMPTHR
265   CONTINUE
      IF (ENATIM.NE.0) GO TO 100
      PRINT 105,M,(NSTATE(M,J),J=1,3)
105   FORMAT('0',' THE MARKING WITH STATE ',I4,' IS A '/,
     5 ' TERMINAL MARKING. AND NSTATE(M) = ',3I20)
      PRINT 500,M,(NTKTMP(J), J= 1,NPLACE)
500   FORMAT('0',' M= ',I5,' THE TERMINAL MARKING IS: ',//,30I3)
      GO TO 90
100   IF( NTIME .EQ.0) GO TO 400
      DO 405, JK= 1,NTIME
      IF((NSTANG(JK,1).EQ.NSTATE(M,1)).AND.
     2 (NSTANG(JK,2).EQ.NSTATE(M,2)).AND.
     3 (NSTANG(JK,3).EQ.NSTATE(M,3))) GO TO 410
405   CONTINUE
400   NTIME = NTIME + 1
      DO 415, LK = 1,3
      NSTANG(NTIME,LK) = NSTATE(M,LK)
415   CONTINUE
410      DO 110,J= 1,NTIMTR
      IF((NTEMP(J,1).EQ.0).AND.(NTEMP(J,2).EQ.0)
     2 .AND.(NTEMP(J,3).EQ.0)) GO TO 110
      DO 115, I= 1,NUMST
      IF((NTEMP(J,1).EQ.NSTATE(I,1)).AND.
     3 (NTEMP(J,2).EQ.NSTATE(I,2))
     2 .AND.(NTEMP(J,3).EQ.NSTATE(I,3))) GO TO 110
115   CONTINUE
      NUMST = NUMST + 1
      DO 112,N = 1,3
      NSTATE(NUMST,N) = NTEMP(J,N)
112   CONTINUE
      NEWFLG = 1
110   CONTINUE
      IF(NUMST.GT.8500) THEN
      PRINT 600,NUMST
600   FORMAT('0',' THE VALUE OF NUMST IS ',I5,' AND EXEEDS THE '
     2 ' ARRAY RANGE OF NSTATE.')
      STOP
      ENDIF
```

```
 90     CONTINUE
C
C SEE IF ANY NEW STATES.
C
       IF(NEWFLG.NE.0) THEN
       ISTART = LIMIT
       PRINT 700,ISTART
700    FORMAT('0',' ISTART= ',I5)
       GO TO 85
       ENDIF
       ICOUNT = NTIME + NIMM
       IF(NUMST .EQ. ICOUNT) GO TO 425
       PRINT 430
430    FORMAT('0',' ERROR IN THE DETERMINATION OF REACHABILITY',
      2 ' SET: ',/,'  THE TOTAL NUMBER OF STATES DO NOT TALLY WITH  ',
      3 '  SUM OF NTIME AND NIMM STATES.')
       PRINT 435,NUMST,NTIME,NIMM
435    FORMAT('0',' NUMST = ',I5,' NTIME= ',I5,' NIMM= ',I5)
       STOP
425       PRINT 440, NUMST
440    FORMAT('0',' THERE ARE A TOTAL OF ',I5,' STATES IN THE ',
      2 ' REACHABILITY SET OF THE GIVEN GENERALIZED STOCHASTIC ',
      3 ' PETRINET.')
       PRINT 450, NTIME
450    FORMAT('0',' THE GSP NET HAS ',I5,' TANGIBLE STATES.')
       PRINT 455,NIMM
455    FORMAT('0',' THE GSP NET HAS ', I5,' VANISHING STATES.')
       PRINT 120,NUMST
120    FORMAT('1',' THERE ARE ',I5,' STATES IN THE REACHABILITY ',
      4 ' SET OF THE GIVEN PETRINET. ')
       PRINT 460
460    FORMAT('1',' THE MARKINGS AND VALUES FOR EACH STATE OF THE ',
      2 ' REACHABILITY SET OF THE GIVEN GSP NET IS AS SHOWN:')
       PRINT 465
465    FORMAT('0',' THE STATE VALUE AND THE CORRESPONDING MARKING ',
      2 ' FOR THE VARIOUS TANGIBLE STATES ARE AS GIVEN BELOW:')
       CALL LISTAT(NTIME,NPLACE,NSTANG)
       PRINT 470
470    FORMAT('0',' THE STATE VALUE AND THE CORRESPONDING MARKING ',
      2 ' FOR THE VARIOUS VANISHING STATES ARE AS GIVEN BELOW:')
       CALL LISTAT(NIMM,NPLACE,NSTVAN)
       ENDTIME = SECOND()
       ELAPTME = ENDTIME - STATIME
       PRINT 710,ELAPTME
710    FORMAT('0',' THE EXECUTION TIME IS : ',F15.6)
       STOP
       END
       SUBROUTINE LISTAT(NCOUNT,NPLACE,NSTARR)
C
C THIS SUBROUTINE LISTS THE VALUES OF VARIOUS STATE VALUES AND
C THEIR CORRESPONDING MARKINGS.
C
       DIMENSION NSTARR(4000,3),NTOKEN(50)
       DO 125, I= 1,NCOUNT
```

```
      PRINT 130,I,(NSTARR(I,J),J=1,3)
130   FORMAT('0',' THE STATE NUMBER OF THE MARKING ',I5,
     5 ' IS ',3I20)
125   CONTINUE
C
C DETERMINE THE TOKENS IN VARIOUS PLACES FOR EACH STATE AND PRINT THEM...
C
      DO 140,I=1, NCOUNT
      ISTONE = NSTARR(I,1)
      ISTTWO = NSTARR(I,2)
      ISTTHR = NSTARR(I,3)
      CALL FNDMRK(NPLACE,ISTONE,ISTTWO,ISTTHR,NTOKEN)
      PRINT 160, I,(NTOKEN(J),J= 1,NPLACE)
160   FORMAT('0',' STATE ',I3,' : ',40I3)
140   CONTINUE
      RETURN
      END
      SUBROUTINE FORMST(ICOUNT,NTKARR,ILOWNUM,IMIDNUM,IHIGNUM)
C
C THIS SUBROUTINE DETERMINES A STATE NUMBER FOR BASE 16 AND ON THE
C NUMBER OF TOKENS AT EACH PLACE AS GIVEN BY THE ARRAY NTKARR.
C
      DIMENSION NTKARR(100),NHEXNUM(3)
       DO 10,J = 1,3
       NHEXNUM(J) = 0
10    CONTINUE
C
C DETERMINE THE STATE NUMBER FROM THE TOKEN INFO IN NTKARR.
C
       K = 0
      DO 15,I = 1,ICOUNT
      NTOKEN = NTKARR(I)
      IMINUS = I - 1
      IMOD = MOD(IMINUS,12)
      IF(IMOD.EQ.0) K = K + 1
      NHEXNUM(K) = NHEXNUM(K) + NTOKEN*(16**IMOD)
15    CONTINUE
      ILOWNUM = NHEXNUM(1)
      IMIDNUM = NHEXNUM(2)
      IHIGNUM = NHEXNUM(3)
      RETURN
      END
       SUBROUTINE FNDMRK(ICOUNT,ISTONE,ISTWO,ISTHR,NTKARR)
C
C THIS SUBROUTINE DETERMINES THE VARIOUS TOKENS AT EACH OF THE PLACE
C IN THE GIVEN PETRINET AND STORES THIS INFORMATION IN THE NTKARR
C ARRAY.
C
      DIMENSION NTKARR(100),NSTNUM(3)
      NSTNUM(1) = ISTONE
      NSTNUM(2) = ISTWO
      NSTNUM(3) = ISTHR
       DO 10, I= 1,ICOUNT
      NTKARR(I) = 0
```

```
10    CONTINUE
C
C FIND THE NUMBER OF TOKENS IN EACH PLACE AND STORE THE VALUE
C IN THE NTKARR.
C
      IMASK = 15
       K = 1
       ITMPNUM = NSTNUM(K)
       DO 15,J = 1,ICOUNT
      NTKARR(J) = ITMPNUM.AND.IMASK
      NUMSHF = SHIFT(ITMPNUM,-4)
      ITMPNUM = NUMSHF
      IMOD =  MOD(J,12)
      IF(IMOD.EQ.0) THEN
      K = K + 1
      ITMPNUM = NSTNUM(K)
      ENDIF
15    CONTINUE
      RETURN
      END
      FUNCTION  ITRENA(ICOUNT,ITRIN,NTKARR)
C
C THIS SUBROUTINE DETERMINES WHETHER A GIVEN TRANSITION IS ENABLED
C FOR A GIVEN STATE AND ACCORDINGLY SETS A FLAG.
C
      DIMENSION NTKARR(100)
       IFLAG = 0
      NUMB = 0
       IONE = 1
      IMARK = ITRIN
       DO 20, K = 1,ICOUNT
      IAND = IMARK.AND.IONE
      IF((IAND.EQ.1).AND.(NTKARR(K).NE.0)) NUMB = NUMB + 2**(K-1)
      ISHFNUM = SHIFT(IMARK,-1)
      IMARK = ISHFNUM
20    CONTINUE
        IF(NUMB .EQ. ITRIN) IFLAG = 1
      ITRENA = IFLAG
      RETURN
      END
      SUBROUTINE FNDNWST( ICOUNT,NINTR,NOUTR,NTKARR,ISTONE,ISTWO,ISTHRE)
C
C THIS SUBROUTINE DETERMINES THE NEW TOKEN DISTRIBUTION FOR THE ENABLED
C TRANSITION AND FINDS THE MARKING STATE FROM THIS NEW TOKEN
C DISTRIBUTION.
C
      DIMENSION NTKARR(100),NTOKEN(100)
      DO 10, I= 1,ICOUNT
      NTOKEN(I)= NTKARR(I)
10    CONTINUE
      IONE = 1
       DO 15, K = 1,ICOUNT
      INAND = NINTR.AND.IONE
      IF(INAND.EQ.1) NTOKEN(K) = NTOKEN(K) - 1
```

```
      ISHFIN = SHIFT(NINTR,-1)
      NINTR = ISHFIN
      IOUTAND = NOUTR.AND.IONE
      IF(IOUTAND.EQ.1) NTOKEN(K) = NTOKEN(K) + 1
      ISHFOUT = SHIFT(NOUTR,-1)
      NOUTR = ISHFOUT
15    CONTINUE
C
C FIND THE NEW STATE FROM THE NEW TOKEN DISTRIBUTION GIVEN IN
C THE ARRAY NTOKEN.
C
      CALL FORMST(ICOUNT,NTOKEN,IONETMP,ITWOTMP,ITHRTMP)
      ISTONE = IONETMP
      ISTWO = ITWOTMP
      ISTHRE = ITHRTMP
      RETURN
      END
```

APPENDIX D:

Reachability set listing of SPN model SPN2A.

THERE ARE 72 STATES IN THE REACHABILITY SET OF SPN2A.

THE VARIOUS STATES IN THE REACHABILITY SET ARE LISTED BELOW:

STATE	1 :	1	0	0	1	0	1	1	0	0	1	1	0	1	0	0
STATE	2 :	0	0	0	1	1	1	1	0	0	1	1	0	1	0	0
STATE	3 :	0	1	0	1	0	1	1	0	0	1	1	0	1	0	0
STATE	4 :	1	0	0	1	0	1	1	0	1	0	1	0	1	0	0
STATE	5 :	1	0	0	1	0	1	1	0	0	0	1	1	1	0	0
STATE	6 :	1	0	0	0	0	0	0	0	0	1	1	0	1	1	0
STATE	7 :	0	0	0	1	1	1	1	0	1	0	1	0	1	0	0
STATE	8 :	0	0	0	1	1	1	1	0	0	0	1	1	1	0	0
STATE	9 :	0	1	0	1	0	1	1	0	1	0	1	0	1	0	0
STATE	10 :	0	1	0	1	0	1	1	0	0	0	1	1	1	0	0
STATE	11 :	1	0	0	1	0	1	0	0	0	1	0	0	0	0	1
STATE	12 :	0	0	0	0	1	0	0	0	0	1	1	0	1	1	0
STATE	13 :	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0
STATE	14 :	1	0	0	0	0	0	0	0	1	0	1	0	1	1	0
STATE	15 :	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0
STATE	16 :	1	0	0	1	0	0	1	1	0	1	1	0	1	0	0
STATE	17 :	0	0	0	1	1	1	0	0	0	1	0	0	0	0	1
STATE	18 :	0	1	0	1	0	1	0	0	0	1	0	0	0	0	1
STATE	19 :	1	0	0	1	0	1	0	0	1	0	0	0	0	0	1
STATE	20 :	1	0	0	1	0	1	0	0	0	0	0	1	0	0	1
STATE	21 :	1	0	1	1	0	1	1	0	0	1	0	0	1	0	0
STATE	22 :	0	0	0	0	1	0	0	0	1	0	1	0	1	1	0
STATE	23 :	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0
STATE	24 :	0	0	0	1	1	0	1	1	0	1	1	0	1	0	0
STATE	25 :	0	1	0	0	0	0	0	0	1	0	1	0	1	1	0
STATE	26 :	0	1	0	0	0	0	0	0	0	0	1	1	1	1	0
STATE	27 :	0	1	0	1	0	0	1	1	0	1	1	0	1	0	0
STATE	28 :	1	0	0	1	0	0	1	1	1	0	1	0	1	0	0
STATE	29 :	1	0	0	1	0	0	1	1	0	0	1	1	1	0	0
STATE	30 :	0	0	0	1	1	1	0	0	1	0	0	0	0	0	1
STATE	31 :	0	0	0	1	1	1	0	0	0	0	0	1	0	0	1
STATE	32 :	0	0	1	1	1	1	1	0	0	1	0	0	1	0	0
STATE	33 :	0	1	0	1	0	1	0	0	1	0	0	0	0	0	1
STATE	34 :	0	1	0	1	0	1	0	0	0	0	0	1	0	0	1
STATE	35 :	0	1	1	1	0	1	1	0	0	1	0	0	1	0	0
STATE	36 :	1	0	1	1	0	1	1	0	1	0	0	0	1	0	0
STATE	37 :	1	0	1	1	0	1	1	0	0	0	0	1	1	0	0
STATE	38 :	0	0	0	1	1	0	1	1	1	0	1	0	1	0	0
STATE	39 :	0	0	0	1	1	0	1	1	0	0	1	1	1	0	0
STATE	40 :	0	1	0	1	0	0	1	1	1	0	1	0	1	0	0
STATE	41 :	0	1	0	1	0	0	1	1	0	0	1	1	1	0	0
STATE	42 :	1	0	0	1	0	0	0	1	0	1	0	0	0	0	1
STATE	43 :	0	0	1	1	1	1	1	0	1	0	0	0	1	0	0
STATE	44 :	0	0	1	1	1	1	1	0	0	0	0	1	1	0	0
STATE	45 :	1	0	1	0	0	0	0	0	0	1	0	0	1	1	0
STATE	46 :	0	1	1	1	0	1	1	0	1	0	0	0	1	0	0
STATE	47 :	0	1	1	1	0	1	1	0	0	0	0	1	1	0	0
STATE	48 :	0	0	0	1	1	0	0	1	0	1	0	0	0	0	1
STATE	49 :	0	1	0	1	0	0	0	1	0	1	0	0	0	0	1

STATE	50	:	1	0	0	1	0	0	0	1	1	0	0	0	0	0	1
STATE	51	:	1	0	0	1	0	0	0	1	0	0	0	1	0	0	1
STATE	52	:	1	0	1	1	0	0	1	1	0	1	0	0	1	0	0
STATE	53	:	1	0	1	0	0	0	0	0	1	0	0	0	1	1	0
STATE	54	:	1	0	1	0	0	0	0	0	0	0	0	1	1	1	0
STATE	55	:	0	0	1	0	1	0	0	0	0	1	0	0	1	1	0
STATE	56	:	0	1	1	0	0	0	0	0	0	1	0	0	1	1	0
STATE	57	:	0	0	0	1	1	0	0	1	1	0	0	0	0	0	1
STATE	58	:	0	0	0	1	1	0	0	1	0	0	0	1	0	0	1
STATE	59	:	0	0	1	1	1	0	1	1	0	1	0	0	1	0	0
STATE	60	:	0	1	0	1	0	0	0	1	1	0	0	0	0	0	1
STATE	61	:	0	1	0	1	0	0	0	1	0	0	0	1	0	0	1
STATE	62	:	0	1	1	1	0	0	1	1	0	1	0	0	1	0	0
STATE	63	:	1	0	1	1	0	0	1	1	1	0	0	0	1	0	0
STATE	64	:	1	0	1	1	0	0	1	1	0	0	0	1	1	0	0
STATE	65	:	0	0	1	0	1	0	0	0	1	0	0	0	1	1	0
STATE	66	:	0	1	1	0	0	0	0	0	1	0	0	0	1	1	0
STATE	67	:	0	0	1	0	1	0	0	0	0	0	0	1	1	1	0
STATE	68	:	0	1	1	0	0	0	0	0	0	0	0	1	1	1	0
STATE	69	:	0	0	1	1	1	0	1	1	1	0	0	0	1	0	0
STATE	70	:	0	0	1	1	1	0	1	1	0	0	0	1	1	0	0
STATE	71	:	0	1	1	1	0	0	1	1	1	0	0	0	1	0	0
STATE	72	:	0	1	1	1	0	0	1	1	0	0	0	1	1	0	0

APPENDIX E:

Rechability set listing of GSPN model GSPN2B.

THERE ARE A TOTAL OF 182 STATES IN THE REACHABILITY SET OF GSPN2B.

GSPN2B HAS 67 TANGIBLE STATES AND 115 VANISHING STATES.

THE FOLLOWING IS THE LISTING OF THE TANGIBLE STATES OF GSPN2B:

STATE	1 :	0	1	0	1	1	0	0	0	1	0	0	0	1	1	0	1	0
STATE	2 :	0	1	0	1	0	1	0	0	1	0	0	0	1	1	0	1	0
STATE	3 :	0	1	0	1	1	0	0	0	1	0	0	1	0	1	0	1	0
STATE	4 :	0	0	0	0	1	0	1	0	0	0	0	0	1	1	0	1	0
STATE	5 :	0	1	0	1	1	0	0	0	0	0	1	0	1	0	0	0	0
STATE	6 :	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	1	0
STATE	7 :	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	1	0
STATE	8 :	0	0	0	1	1	0	0	1	1	0	0	0	1	1	0	1	0
STATE	9 :	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0	1	0
STATE	10 :	0	1	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0
STATE	11 :	0	1	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0
STATE	12 :	0	1	0	1	1	0	0	0	0	0	1	0	0	0	1	0	0
STATE	13 :	0	1	0	1	1	0	0	0	1	1	0	0	1	1	0	0	0
STATE	14 :	0	0	1	0	0	0	1	0	0	0	0	1	0	1	0	1	0
STATE	15 :	0	0	0	1	0	1	0	1	1	0	0	0	1	1	0	1	0
STATE	16 :	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	1
STATE	17 :	0	1	0	1	0	1	0	0	0	0	1	0	0	0	1	0	0
STATE	18 :	1	1	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0
STATE	19 :	0	1	0	1	1	0	0	0	1	1	0	1	0	1	0	0	0
STATE	20 :	1	0	0	0	1	0	0	1	1	0	0	0	1	1	0	1	0
STATE	21 :	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1
STATE	22 :	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	1
STATE	23 :	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	1	1
STATE	24 :	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1
STATE	25 :	1	1	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0
STATE	26 :	1	1	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0
STATE	27 :	1	1	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0
STATE	28 :	1	1	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0
STATE	29 :	0	1	0	1	1	0	0	0	1	1	0	0	1	0	0	0	1
STATE	30 :	1	0	1	0	0	0	0	1	1	0	0	0	1	1	0	1	0
STATE	31 :	1	0	0	0	0	1	0	1	1	0	0	0	1	1	0	1	0
STATE	32 :	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	1
STATE	33 :	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	1	1
STATE	34 :	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	1
STATE	35 :	0	0	0	1	1	0	0	1	0	0	1	0	1	0	0	0	0
STATE	36 :	1	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0
STATE	37 :	1	1	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0
STATE	38 :	1	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0
STATE	39 :	0	0	0	0	1	0	1	0	0	1	0	0	1	1	0	0	0
STATE	40 :	0	1	0	1	1	0	0	0	1	1	0	1	0	0	0	0	1
STATE	41 :	0	1	0	1	1	0	0	0	1	1	0	0	0	0	1	0	1
STATE	42 :	0	0	0	1	0	1	0	1	0	0	1	0	1	0	0	0	0
STATE	43 :	0	0	0	1	1	0	0	1	0	0	1	0	0	0	1	0	0
STATE	44 :	0	0	0	1	1	0	0	1	1	1	0	0	1	1	0	0	0
STATE	45 :	0	0	1	0	0	0	1	0	0	1	0	0	1	1	0	0	0
STATE	46 :	0	0	0	0	1	0	1	0	0	1	0	1	0	1	0	0	0
STATE	47 :	1	0	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0

STATE	48 :	0	0	0	1	0	1	0	1	0	0	1	0	0	0	1	0	0
STATE	49 :	0	0	1	0	0	0	1	0	0	1	0	1	0	1	0	0	0
STATE	50 :	0	0	0	0	1	0	1	0	0	1	0	0	1	0	0	0	1
STATE	51 :	1	0	1	0	0	0	0	1	0	0	1	0	1	0	0	0	0
STATE	52 :	1	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0
STATE	53 :	1	0	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0
STATE	54 :	1	0	0	0	1	0	0	1	1	1	0	0	1	1	0	0	0
STATE	55 :	0	0	0	1	1	0	0	1	1	1	0	0	1	0	0	0	1
STATE	56 :	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	1
STATE	57 :	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	1
STATE	58 :	0	0	0	0	1	0	1	0	0	1	0	0	0	0	1	0	1
STATE	59 :	1	0	1	0	0	0	0	1	0	0	1	0	0	0	1	0	0
STATE	60 :	1	0	1	0	0	0	0	1	1	1	0	0	1	1	0	0	0
STATE	61 :	1	0	0	0	0	1	0	1	0	0	1	0	0	0	1	0	0
STATE	62 :	0	0	0	1	1	0	0	1	1	1	0	0	0	0	1	0	1
STATE	63 :	0	0	1	0	0	0	1	0	0	1	0	1	0	0	0	0	1
STATE	64 :	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1	0	1
STATE	65 :	1	0	0	0	1	0	0	1	1	1	0	0	0	1	0	0	1
STATE	66 :	1	0	1	0	0	0	0	1	1	1	0	0	0	1	0	0	1
STATE	67 :	1	0	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1

THE FOLLOWING IS THE LISTING OF THE VANISHING STATES OF GSPN2B:

STATE	1 :	0	1	1	1	0	0	0	0	1	0	0	0	1	1	0	1	0
STATE	2 :	0	1	0	1	1	0	0	0	1	0	0	0	0	1	1	1	0
STATE	3 :	1	1	0	0	1	0	0	0	1	0	0	0	1	1	0	1	0
STATE	4 :	0	1	0	1	0	1	0	0	1	0	0	1	0	1	0	1	0
STATE	5 :	0	1	0	1	0	1	0	0	1	0	0	0	0	1	1	1	0
STATE	6 :	0	1	1	1	0	0	0	0	1	0	0	1	0	1	0	1	0
STATE	7 :	0	1	0	1	1	0	0	0	1	0	0	0	1	0	0	1	1
STATE	8 :	0	1	0	1	0	1	0	0	1	0	0	0	1	0	0	1	1
STATE	9 :	1	1	0	0	1	0	0	0	1	0	0	1	0	1	0	1	0
STATE	10 :	0	0	0	0	1	0	1	0	0	0	0	0	0	1	1	1	0
STATE	11 :	0	1	1	1	0	0	0	0	0	0	1	0	1	0	0	0	0
STATE	12 :	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	1	0
STATE	13 :	0	0	1	0	0	0	1	0	0	0	0	0	0	1	1	1	0
STATE	14 :	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	1	0
STATE	15 :	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1	0
STATE	16 :	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0
STATE	17 :	0	0	0	1	1	0	0	1	1	0	0	0	0	1	1	1	0
STATE	18 :	0	1	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0
STATE	19 :	0	1	0	1	0	1	0	0	1	1	0	0	1	1	0	0	0
STATE	20 :	0	1	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0
STATE	21 :	0	1	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0
STATE	22 :	0	1	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0
STATE	23 :	0	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	0
STATE	24 :	0	0	1	1	0	0	0	1	1	0	0	1	0	1	0	1	0
STATE	25 :	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	1	0
STATE	26 :	0	0	0	1	0	1	0	1	1	0	0	0	0	1	1	1	0
STATE	27 :	0	0	0	1	1	0	0	1	1	0	0	0	1	0	0	1	1
STATE	28 :	0	1	0	1	0	1	0	0	1	1	0	0	0	1	1	0	0
STATE	29 :	1	1	0	0	1	0	0	0	1	1	0	0	1	1	0	0	0

STATE	30 :	0	1	1	1	0	0	0	0	1	1	0	1	0	1	0	0	0
STATE	31 :	0	1	0	1	0	1	0	0	1	1	0	1	0	1	0	0	0
STATE	32 :	1	0	0	0	1	0	0	1	1	0	0	1	0	1	0	1	0
STATE	33 :	1	0	0	0	1	0	0	1	1	0	0	0	0	1	1	1	0
STATE	34 :	0	0	1	1	0	0	0	1	1	0	0	0	1	0	0	1	1
STATE	35 :	0	0	0	1	0	1	0	1	1	0	0	0	1	0	0	1	1
STATE	36 :	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	1	1
STATE	37 :	0	0	0	1	1	0	0	1	1	0	0	1	0	0	0	1	1
STATE	38 :	0	0	0	1	1	0	0	1	1	0	0	0	0	0	1	1	1
STATE	39 :	0	1	0	1	0	1	0	0	1	1	0	0	1	0	0	0	1
STATE	40 :	1	1	1	0	0	0	0	0	1	1	0	0	1	1	0	0	0
STATE	41 :	1	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0
STATE	42 :	1	1	0	0	0	1	0	0	1	1	0	0	1	1	0	0	0
STATE	43 :	1	1	0	0	1	0	0	0	1	1	0	1	0	1	0	0	0
STATE	44 :	1	1	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0
STATE	45 :	0	1	1	1	0	0	0	0	1	1	0	0	1	0	0	0	1
STATE	46 :	1	0	1	0	0	0	0	1	1	0	0	1	0	1	0	1	0
STATE	47 :	1	0	1	0	0	0	0	1	1	0	0	0	0	1	1	1	0
STATE	48 :	1	0	0	0	0	1	0	1	1	0	0	1	0	1	0	1	0
STATE	49 :	1	0	0	0	0	1	0	1	1	0	0	0	0	1	1	1	0
STATE	50 :	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
STATE	51 :	0	0	1	1	0	0	0	1	0	0	1	0	1	0	0	0	0
STATE	52 :	0	0	1	1	0	0	0	1	1	0	0	1	0	0	0	1	1
STATE	53 :	0	0	1	1	0	0	0	1	1	0	0	0	0	0	1	1	1
STATE	54 :	0	0	0	1	0	1	0	1	1	0	0	0	0	0	1	1	1
STATE	55 :	0	0	0	1	1	0	0	1	0	0	1	1	0	0	0	0	0
STATE	56 :	1	1	1	0	0	0	0	0	1	1	0	1	0	1	0	0	0
STATE	57 :	1	1	1	0	0	0	0	0	1	1	0	0	0	1	1	0	0
STATE	58 :	1	1	0	0	0	1	0	0	1	1	0	0	0	1	1	0	0
STATE	59 :	0	0	0	0	0	1	1	0	0	1	0	0	1	1	0	0	0
STATE	60 :	0	0	0	0	1	0	1	0	0	1	0	0	0	1	1	0	0
STATE	61 :	1	1	0	0	1	0	0	0	1	1	0	0	1	0	0	0	1
STATE	62 :	0	1	1	1	0	0	0	0	1	1	0	1	0	0	0	0	1
STATE	63 :	0	1	0	1	0	1	0	0	1	1	0	1	0	0	0	0	1
STATE	64 :	0	1	1	1	0	0	0	0	1	1	0	0	0	0	1	0	1
STATE	65 :	0	1	0	1	0	1	0	0	1	1	0	0	0	0	1	0	1
STATE	66 :	1	1	1	0	0	0	0	0	1	0	0	0	1	1	0	1	0
STATE	67 :	1	0	1	0	0	0	0	1	1	0	0	0	1	0	0	1	1
STATE	68 :	1	1	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0
STATE	69 :	1	0	0	0	0	1	0	1	1	0	0	0	1	0	0	1	1
STATE	70 :	1	0	0	0	1	0	0	1	1	0	0	1	0	0	0	1	1
STATE	71 :	0	1	1	1	0	0	0	0	1	0	0	0	1	0	0	1	1
STATE	72 :	0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0
STATE	73 :	1	0	0	0	1	0	0	1	1	0	0	0	0	0	1	1	1
STATE	74 :	0	0	1	1	0	0	0	1	0	0	1	0	0	0	1	0	0
STATE	75 :	0	0	0	1	0	1	0	1	0	0	1	1	0	0	0	0	0
STATE	76 :	0	0	0	1	0	1	0	1	1	1	0	0	1	1	0	0	0
STATE	77 :	0	0	0	1	1	0	0	1	1	1	0	0	0	1	1	0	0
STATE	78 :	0	0	1	1	0	0	0	1	1	1	0	0	1	1	0	0	0
STATE	79 :	0	0	0	1	1	0	0	1	1	1	0	1	0	1	0	0	0
STATE	80 :	0	0	1	0	0	0	1	0	0	1	0	0	0	1	1	0	0
STATE	81 :	1	1	1	0	0	0	0	0	1	1	0	0	1	0	0	0	1
STATE	82 :	0	0	0	0	0	1	1	0	0	1	0	0	0	1	1	0	0
STATE	83 :	1	1	0	0	1	0	0	0	1	0	0	0	0	1	1	1	0

STATE	84 :	1	1	0	0	0	1	0	0	1	1	0	0	1	0	0	0	1
STATE	85 :	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	0	0
STATE	86 :	1	1	0	0	1	0	0	0	1	1	0	1	0	0	0	0	1
STATE	87 :	0	1	0	1	1	0	0	0	1	0	0	1	0	0	0	1	1
STATE	88 :	1	1	0	0	1	0	0	0	1	1	0	0	0	0	1	0	1
STATE	89 :	0	1	0	1	1	0	0	0	1	0	0	0	0	0	1	1	1
STATE	90 :	1	0	0	0	1	0	0	1	0	0	1	1	0	0	0	0	0
STATE	91 :	1	1	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1
STATE	92 :	0	0	0	1	0	1	0	1	1	1	0	0	0	1	1	0	0
STATE	93 :	0	0	1	1	0	0	0	1	1	1	0	1	0	1	0	0	0
STATE	94 :	0	0	0	0	0	1	1	0	0	1	0	0	1	0	0	0	1
STATE	95 :	1	0	1	0	0	0	0	1	0	0	1	1	0	0	0	0	0
STATE	96 :	1	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0	0
STATE	97 :	1	0	0	0	0	1	0	1	1	1	0	0	1	1	0	0	0
STATE	98 :	1	0	0	0	1	0	0	1	1	1	0	0	0	1	1	0	0
STATE	99 :	1	0	0	0	1	0	0	1	1	1	0	1	0	1	0	0	0
STATE	100 :	0	0	0	1	0	1	0	1	1	1	0	0	1	0	0	0	1
STATE	101 :	0	0	1	1	0	0	0	1	1	1	0	0	1	0	0	0	1
STATE	102 :	0	0	0	1	1	0	0	1	1	1	0	1	0	0	0	0	1
STATE	103 :	0	0	0	0	0	1	1	0	0	1	0	1	0	0	0	0	1
STATE	104 :	0	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	1
STATE	105 :	1	0	1	0	0	0	0	1	1	1	0	0	0	1	1	0	0
STATE	106 :	1	0	1	0	0	0	0	1	1	1	0	1	0	1	0	0	0
STATE	107 :	1	0	0	0	0	1	0	1	1	1	0	0	0	1	1	0	0
STATE	108 :	0	0	1	1	0	0	0	1	1	1	0	0	0	0	1	0	1
STATE	109 :	0	0	0	1	0	1	0	1	1	1	0	0	0	0	1	0	1
STATE	110 :	0	0	1	1	0	0	0	1	1	1	0	1	0	0	0	0	1
STATE	111 :	1	0	0	0	0	1	0	1	1	1	0	0	1	0	0	0	1
STATE	112 :	1	0	0	0	1	0	0	1	1	1	0	1	0	0	0	0	1
STATE	113 :	1	0	1	0	0	0	0	1	1	1	0	1	0	0	0	0	1
STATE	114 :	1	0	1	0	0	0	0	1	1	1	0	0	0	0	1	0	1
STATE	115 :	1	0	0	0	0	1	0	1	1	1	0	0	0	0	1	0	1