

Global Petri Net Modeling of Hybrid Systems and Fault Analysis

By
Mohammad Rezai

B.E. Electrical and Electronics Engineering,

Bharathidasan University, Tiruchirapalli, India, 1987

M.Sc.Eng. Electrical Engineering, University of New Brunswick, Fredericton, 1990

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY**

in

**THE FACULTY OF GRADUATE STUDIES
(DEPARTMENT OF ELECTRICAL ENGINEERING)**

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

1996

© Mohammad Rezai, 1996

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Electrical Engineering

The University of British Columbia
Vancouver, Canada

Date July 18, 1996

Abstract

In this dissertation, a new methodology for the modeling and analysis of hybrid systems is presented. Hybrid systems are those which have both event-driven (asynchronous) and time-driven (synchronous) elements. This new methodology is based on an extension of the Petri net (PN) theory. Petri nets have proven themselves to be an excellent modeling tool for discrete-event systems and computer architectures. This new extension of Petri nets, which is called a Global Petri Net (GPN), provides a means for extending these capabilities to discrete-time systems. Although, many extensions of PNs have been developed over the years, the GPN methodology is the first one to perform the modeling, analysis, and simulation of discrete-event and discrete-time dynamic systems in a unified PN framework.

The GPN is formally defined, and the structural and behavioral differences between PNs and GPNs are presented. The derivation of GPNs from the basic PN, and derivation of the GPN dynamic equations are also given. Next, two modeling examples are provided. These two examples show that the GPN formalism can be used to model hybrid systems in various application areas.

Analysis techniques, which can be used to investigate the system properties, are developed. These analysis techniques are based on the construction of the system reachability tree and linear algebraic equations. The properties which are analyzed by these techniques include controllability, boundedness, stability, and conservation. Theorems and proofs of these properties are stated and proven.

An example of a distributed hybrid system is modeled at various levels of abstraction. This system consists of a hydraulic control system and its interfaces with a bus-based communication system. At the highest level, the system is modeled as a discrete-event system. At the lowest level, where the details of the hydraulic control system are modeled, a hybrid model is used. The nets at all levels are simulated and analyzed by the global Petri net simulation and analysis tool (GPNSAT), written specifically for this research.

Various system level faults for a hydraulic control system are modeled and simulated. For each fault type, the simulation shows how various system outputs are affected. At each level of abstraction, the hybrid system is simulated and analyzed for various properties, such as stability, boundedness, controllability, conservativeness, and liveness. Analysis methods developed for GPNs provide distinct fault signatures for each of the system fault types. These fault signatures can be used to distinguish successfully each fault in a detection and recognition scheme.

The major contribution of this thesis is the extension of Petri nets to encompass hybrid systems. This new modeling approach can be applied to a variety of systems in application areas such as, manufacturing, multimedia, production, digital, and communication systems. This extension also enables one to examine the impact of faults in either part of the system (synchronous or asynchronous) and the effect that fault would have on the other parts of the system. These effects can be analyzed or simulated either at design time or run-time.

Contents

Abstract	ii
List of Tables	ix
List of Figures	x
List of Definitions	xiv
List of Symbols and Abbreviations	xv
Acknowledgments	xvii
Dedications	xviii
Chapter I	Introduction 1
I.1	Problem Definition and Research Motivation 1
I.2	Real-Time Control Systems 2
I.3	Hybrid Systems 3
I.4	Dissertation Contributions 7
I.5	Dissertation Layout 8
Chapter II	Petri Net Modeling of the Systems 10
II.1	Petri Net Definition 10
II.2	Petri Net Graph 12
II.3	Petri net Dynamics 12
II.4	An Example of a PN 13
II.5	Why Petri Nets? 14
II.6	Assigning Time to Petri Nets 16
II.7	Review of Some Relevant Petri nets 17
II.7.1	Fault Detection by Petri Nets: 17
II.7.2	Digital Control Systems: 17
II.7.3	Petri Nets as Discrete Controllers: 18

II.7.4	Failure Modeling and Analysis in a Material Handling System:	18
II.7.5	Continuous Petri Nets:	18
II.7.6	Hybrid Petri Nets:	19
Chapter III	Global Petri Nets: Definitions and Examples	21
III.1	Timed Petri Net Definition	22
III.2	Global Petri Net Definition	22
III.3	GPN Graph	23
III.4	GPN Dynamics	24
III.5	An Example of a GPN	25
III.6	Derivation of the GPN from Conventional PN	28
III.7	Derivation of GPN Dynamics Equation	29
III.8	Comparison of the PN and the GPN Modeling	34
III.9	Place and Transition Types in GPN	37
III.9.1	Transition Types:	37
III.9.2	Place Types:	39
III.10	Advantages of Modeling with the GPN	40
III.10.1	Hybrid Modeling; Water Tank Flow Control Example:	40
III.11	GPN Modeling of a Six-Transistor XOR Gate	43
III.11.1	MOS Transistor:	43
III.11.2	GPN Modeling with MOS Transistors:	46
III.11.3	Simulation Conclusions	49

Chapter IV	GPN Properties and Analysis Methods	50
IV.1	GPN Modeling Issues	51
IV.1.1	Hybrid Transition Matrix:	51
IV.1.2	A Sub-class of GPNs (When A is a Diagonal Matrix):	53
IV.2	Condition for GPN Modelability	55
IV.2.1	An Example:	57
IV.2.2	Modelability Condition for the Synchronous Part of GPNs:	58
IV.3	GPN Hierarchy	58
IV.4	Analysis Methods	59
IV.4.1	Reachability Tree:	59
IV.4.2	Linear Algebraic Method:	65
IV.5	Controllability	67
IV.5.1	Controllability (When A is an Identity Matrix):	67
IV.5.2	Controllability of Time-variant Systems:	68
IV.6	Conservation	69
IV.6.1	Conservation of GPNs with Diagonal A Matrix:	70
IV.6.2	Conservation of any Given GPN:	70
IV.7	Boundedness and Stability	71
IV.8	Liveness	74
IV.9	Safeness	75
IV.10	Reachability	75

Chapter V	Global Petri Net Simulation and Analysis Tool (GPNSAT) . 76
V.1	Modeling Procedure 76
V.2	GPNSAT Top View Structure 77
V.3	GPNSAT Substructures 78
V.3.1	Human Interface: 78
V.3.2	Net Utilities: 79
V.3.3	Simulators: 82
V.3.4	Analysis Tools: 83
Chapter VI	Modeling and Analysis of a Hydraulic Control System . . 85
VI.1	Overall System Structure: Level 0 85
VI.1.1	Petri Net Analysis of the Model at Level 0: 91
VI.2	A More Detailed Model of the Overall System: Level 1 94
VI.3	A Hydraulic Control System 96
VI.4	GPN Model of the Hydraulic System: Level 2 99
VI.5	GPN Analysis of the Hydraulic System 103
VI.5.1	Linear Algebraic Analysis: 103
VI.5.2	Reachability Tree Analysis: 105
VI.6	Simulation and Analysis Performance 106
Chapter VII	Fault Modeling and Analysis by GPN 108
VII.1	System Level Faults 108
VII.2	Hydraulic System Faults 110
VII.2.1	Sensor Faults Model: 111

VII.3	Fault Scenarios: Simulation and Analysis	113
VII.3.1	Sensor One Gain Change Fault Simulation:	113
VII.3.2	Sensor One Gain Change Fault Analysis:	114
VII.3.3	Sensor Two Gain Change Fault Simulation and Analysis: . .	117
VII.3.4	Sensors Bias Faults Simulation and Analysis:	119
VII.4	Other Fault Types	121
VII.4.1	Actuator Faults Modeling:	121
VII.4.2	Actuator Faults Simulation and Analysis:	122
VII.4.3	Disturbances and System Parameter Changes:	124
VII.5	Fault Conditions Modeled by a GPN	125
Chapter VIII	Conclusions and Future Work	127
VIII.1	Future Work	129
Chapter IX	Bibliography	131
Appendix A	Modeling of Logic Gates by GPNs	141
A.1	Inverter Gate	141
A.2	AND Gate	142
A.3	NAND Gate	142
A.4	NOR Gate	142
A.5	OR Gate	143
Appendix B	Derivation of the Hybrid Transition Matrix	144
Appendix C	Diagonal A Matrix Transformation	147
Appendix D	The Hybrid System Global Petri Net Parameters	155
Appendix E	Fault Detection, Identification and Reconfiguration (FDIR) Scheme	157

List of Tables

Table III.1	Structural Differences between the PN and the GPN.	34
Table III.2	Simulation Results of the GPN Representing an XOR Gate. .	47
Table VI.3	Roots of $H_k + I$ Matrices Formed by the Firing of Various Hybrid Transitions.	105
Table VII.4	Sensor One Gain Change Faults and Their Analysis Results.	117
Table VII.5	Sensor Two Gain Faults and Their Analysis Results.	119
Table VII.6	Sensor Bias Faults and Their Analysis Results.	121
Table VII.7	Actuators Faults and Their Analysis Results.	124
Table A.8	The Truth Table for Various Digital Logic Gates.	141

List of Figures

Figure I.1	A Typical Real-Time Control System.	3
Figure I.2	System Classification and Thesis Scope.	7
Figure II.3	A Simple Example of a Petri Net Modeling a Printing Process.	13
Figure III.4	An Example of a Simple GPN.	25
Figure III.5	A General GPN with all Possible Arcs	31
Figure III.6	Various Transition Types in the GPN.	38
Figure III.7	Various Place Types in the GPN.	40
Figure III.8	(a) A Simple Flow control System. (b) The GPN Model of (a). .	41
Figure III.9	Place Markings Versus Time Plots for the Flow Control Example.	42
Figure III.10	nMOS and pMOS Transistor Symbols.	43
Figure III.11	Hybrid GPN Model of an nMOS Transistor.	44
Figure III.12	Simulation Plots of Different Voltages in an Analog GPN Model of an nMOS Transistor.	46
Figure III.13	The Six-Transistor XOR Gate.	47
Figure III.14	Plots of the Input and Output Signals of the Hybrid Model of the XOR Gate.	48
Figure III.15	HSPIICE Simulation Results of the Six-transistor XOR Gate. .	49
Figure IV.16	Hybrid Transitions.	51
Figure IV.17	Two Examples of Nets not Allowed by Diagonal A Restriction.	54
Figure IV.18	Two Examples of Nets Allowed by Diagonal A Matrix Restriction.	55
Figure IV.19	An Algorithm for Constructing GPN and PN Reachability Trees.	61

Figure IV.20	Examples of a GPN used for Construction of GPNRTs. (a) A GPN with only Asynchronous Transitions. (b) A GPN with an Extra Synchronous Transition.	63
Figure IV.21	Reachability Tree of the GPN Given in Figure IV.20(a).	64
Figure IV.22	Reachability Tree of the GPN Given in Figure IV.20(b).	65
Figure V.23	Modeling Procedure Used for Translation of System Specifications into a Complete Net Model.	77
Figure V.24	The Overall Structure of the GPN Simulation and Analysis Tool (GPNSAT).	78
Figure VI.25	Block Diagram of a Hydraulic Control System Hardware Structure.	86
Figure VI.26	Block Diagram of a Distributed System at Level 0.	87
Figure VI.27	Petri Net Model of the System shown in Figure VI.26.	88
Figure VI.28	Simulation Results of the PN Model Given in Figure VI.27. (a) Place p_1 Marking. (b) Place p_2 Marking. (c) Place p_3 Marking. (d) Place p_4 Marking.	90
Figure VI.29	Simulation Results of the PN Model Given in Figure VI.27 when the Number of Bus Channels is Increased to Two. (a) Place p_1 Marking. (b) Place p_2 Marking. (c) Place p_3 Marking. (d) Place p_4 Marking.	91
Figure VI.30	The Reachability Tree of the Petri Net Shown in Figure VI.27 .	92
Figure VI.31	A More Detailed Model of the Overall System: Level 1.	94
Figure VI.32	A Petri Net Model of the Distributed System Given in Figure VI.31.	95
Figure VI.33	Schematic Diagram of a Two-stage Hydraulic Valve System .	97
Figure VI.34	Block Diagram Representation of the Plant and the Controller Corresponding to Transitions t_4 and t_6 in Figure VI.32.	98

Figure VI.35	Global Petri Net Model of the Plant Shown in Figure VI.34.	99
Figure VI.36	Global Petri Net Model of the Controller Shown in Figure VI.34.	100
Figure VI.37	Comparison of Simulation Results of the Complete System by the GPN Model and a Conventional Control System Simulator. (a) Plant Output Place X_o in Figure VI.35, (b) Control Input Place V_e in Figure VI.36, (c) Plant Output X_o , (d) Control Input V_e	101
Figure VI.38	Result of the Simulation of the Complete System by the GPN Model (a) Plant Output Validity Place p_5 in Figure VI.32, (b) Control Input Validity Place p_{10} in Figure VI.32.	102
Figure VII.39	Simulation Results of the Hybrid System when the Bus Accesses are not Met. (a) Hydraulic System Output (Place X_o Marking). (b) Control Input (Place V_e Marking). (c) Desired Input Validity (Place p_2) Marking. (d) "All Other" Subsystem (Place p_7) Marking.	110
Figure VII.40	A Sensor Fault Model.	111
Figure VII.41	A Sensor Fault Model as a More Detailed Representation of Sensor Transition.	112
Figure VII.42	Simulation Plots of the First Sensor Gain Faults. a-b) Sensor Cut off, $G_f = 1$. c-d) Sensor Gain Change to $G_f = -0.5$. e-f) Sensor Gain Change to $G_f = -1$. g-h) Sensor Gain Change to $G_f = -1.5$	115
Figure VII.43	Simulation Plots of the Second Sensor Gain Faults. a-b) Sensor Cut off, $G_f = 1$. c-d) Sensor Gain Change to $G_f = -4$. e-f) Sensor Gain Change to $G_f = -7$	118

Figure VII.44	Simulation Plots of the Sensors Bias Faults. a-b) Sensor Bias of 10, $d = 10$. c-d) Sensor Bias of 100, $d = 100$	120
Figure VII.45	An Actuator Fault Model.	122
Figure VII.46	Simulation Plots of the Actuator Faults. a-b) Actuator Cutoff $G_f = 1$. c-d) Actuator Gain Change to $G_f = -0.2$. e-f) Actuator Gain Change to $G_f = -1.2$	123
Figure A.47	The Inverter Gate Modeled by a GPN.	141
Figure A.48	The AND Gate Modeled by a GPN.	142
Figure A.49	The NAND Gate Modeled by a GPN.	142
Figure A.50	The OR Gate Modeled by a GPN.	143
Figure C.51	A Two-Place, Two-Transition GPN with All Possible Arcs .	147
Figure E.52	The FDIR Scheme Block Diagram	158

List of Definitions

Definition 2.1 :	Petri Net	Section II.1
Definition 2.2 :	Enabled Transition	Section II.3
Definition 3.1 :	Timed Petri Net	Section III
Definition 3.2 :	Global Petri Net	Section III.2
Definition 3.3 :	Diagonalizing Function	Section III.7
Definition 3.4 :	One Function	Section III.7
Definition 3.5 :	PN Incidence Matrix (N Matrix)	Section III.7
Definition 3.6 :	PN Transition Firing Vector	Section III.7
Definition 3.7 :	GPN Transition Firing Matrix	Section III.7
Definition 3.8 :	GPN Dynamic Equation	Section III.8
Definition 3.9 :	PN Dynamic Equation	Section III.8
Definition 3.10 :	Synchronous Transition	Section III.9
Definition 3.11 :	Asynchronous Transition	Section III.9
Definition 3.12 :	Real Place	Section III.9
Definition 3.13 :	Integer Place	Section III.9
Definition 4.1 :	Hybrid Transition	Section IV.1

List of Symbols and Abbreviations

Symbol	Definition
A	Synchronous Arc Weight Matrix Connecting Places to Transitions, Standard Control Matrix
B	Synchronous Arc Weight Matrix Connecting Transitions to Places, Standard Control Matrix
DEDS	Discrete Event Dynamic System
Diag	Diagonalizing Function
F_k	GPN Transition Firing Matrix at Instant k
f_k	PN Transition Firing Vector at Instant k
FDIR	Fault Detection, Identification and Reconfiguration
FTP	File Transfer Protocol
GPN	Global Petri Net
GPNRT	Global Petri Net Reachability Tree
GPNSAT	Global Petri Net Simulation and Analysis Tool
$H(k), H_k$	Hybrid Matrix at Time Instant k
HI	Human Interface
HMP	Health Monitoring Process
I	Identity Matrix
k	Time Instant, Sampling Instant
l	Number of Places
$M(k), M_k$	Marking at Time Instant k
$M(0)$	Initial Marking
N	Incidence Matrix
n	Number of Transitions
One	One Function

Symbol

$P = \{p_1, p_2, \dots, p_m\}$

p_i

p_o

PN

PNRT

RT

$T = \{t_1, t_2, \dots, t_n\}$

TPN

$TT = \{tt_1, tt_2, \dots, tt_n\}$

U

W_{pt}

W_{tp}

X

Y

z

ω

Definition

Places

Input Place

Output Place

Petri Net

Petri Net Reachability Tree

Reachability Tree

Transitions

Timed Petri Net

Transition Times

Input Vector

Asynchronous Arc Weight Matrix

Connecting Places to Transitions

Asynchronous Arc Weight Matrix

Connecting Transitions to Places

State Vector

An Arbitrary Vector, Output Vector

z Transform

Infinity Element in a Reachability Tree

Acknowledgments

My first and foremost thanks go to my supervisors, Professor Mabo R. Ito and Professor Peter D. Lawrence, for their guidance and insight. They remained my source of inspiration and faith in this work. Their support and assistance are gratefully appreciated.

I wish to acknowledge my Ph.D. supervisory committee members, Dr. Andre Iyanov and Dr. Jeffrey Joyce, who provided me with the most needed help since the start of this research.

I also hope to thank Dr. D. Cherkas for acting as the chair of the examining committee and conducting the defence. I am also heavily indebted to my external examiner, Dr. K. Valavanis, and members of my examining committee, Drs. A. Mackworth, M. Davies and G. Bond. I would like to thank them all for their excellent comments and suggestions.

Next, I would like to sincerely thank our EE secretaries, Leslie Nichols and Doris Metcalf, who always looked after all the forms and office work.

And finally I should thank my friends in the EE department and high performance computing lab, especially Kendra Cooper for proofreading my thesis draft, and my Iranian friends at UBC who provided the "home away from home" feeling for my family.

Dedications

To my wonderful wife Fereshteh, whose support, understanding and encouragements made it all possible.

And

To our sweet daughter, Raumina who has made our lives so much more fun, purposeful, and motivated.

Chapter I Introduction

In this dissertation, a new methodology for the modeling of hybrid systems is developed. Hybrid systems are defined as systems which have both time- and event-driven parts. Such systems can be found in many application areas, such as manufacturing [1], real-time control systems [2], communication [3], robotics, and production [4]. This modeling technique can also be used for the modeling and simulation of faults in these systems [5]. The system is modeled by a new extension of Petri nets which can replicate the system at various levels of abstraction. This analytical and redundant model can be used to detect and recognize faults.

This chapter is an overview of the preliminary notions, concepts, and definitions which are required in the presentation of these research findings. It starts by defining the problem and stating the research objectives. Then the primary application area, the control of real-time systems, is described. Hybrid systems are also defined, and their characteristics are discussed. The chapter ends by enumerating the dissertation contributions and presenting the dissertation layout.

I.1. Problem Definition and Research Motivation

Control and system scientists have mainly considered systems whose states change with time. Such systems are referred to as time-driven systems. But with the advent of computers and digital devices, we are faced more and more with systems whose dynamics change with events [6]. These events, which are either internal to the systems or are due to the environment, result in discontinuous states. The state trajectories of such systems are made up of these changes. Such systems are referred to as discrete-event or event-driven systems.

Modeling, simulation, and analysis of any practical system (with a computer and communication parts), has to consider both the time-driven and event-driven sub-systems. Systems which comprise both these sub-systems are called hybrid systems. Event-driven systems have been modeled by many methods, such as Petri nets [7], state machines [8],

finite state Markov chains, queueing networks, and Statecharts. On the other hand, control system theory has dealt with the modeling, design, and simulation of time-driven systems.

In this thesis, a new methodology and technique is developed and presented which can study hybrid systems using a single modeling tool. This methodology can be applied to any system which is to be modeled and studied as a hybrid system. The study of faults in dynamic systems is a very good example for application of hybrid system modeling [9]. Faults are considered events which change the dynamic (time-driven) behavior of the systems. Modeling fault-prone systems as hybrid systems will allow one to study and predict the effects of faults on overall system behavior [10]. In this way these faults can be detected and recognized by schemes developed for this purpose.

The modeling methodology, which is presented in this thesis, can be applied to any type of hybrid systems. This is shown through a series of examples. One example, which is dealt with quite thoroughly in Chapter III, is modeling of a digital XOR gates composed of MOS transistors. These transistors are modeled at both the analog and digital (switch) levels. However, our primary target area is modeling and analysis of real-time control systems which is described next.

1.2. Real-Time Control Systems

Real-time controllers are used increasingly in various facets of life, ranging from home appliances to large and complex systems for industrial and military applications. Real-time means that the correctness of the computation, or the decision, depends not only on the logical correctness, but also on the time at which the result is produced [11]. It should be noted that a fast computation does not guarantee real-time correctness since it may not be completed at the appropriate time [12]. Typical real-time control systems consist of an object or controlled environment connected to a control system (computer) via sensors and actuators (Figure I.1). Sensors accept data at regular periods or are event-driven [13]. The

sensor inputs are used, along with the control strategy, to calculate the control inputs. These control inputs are applied to the system through the actuators.

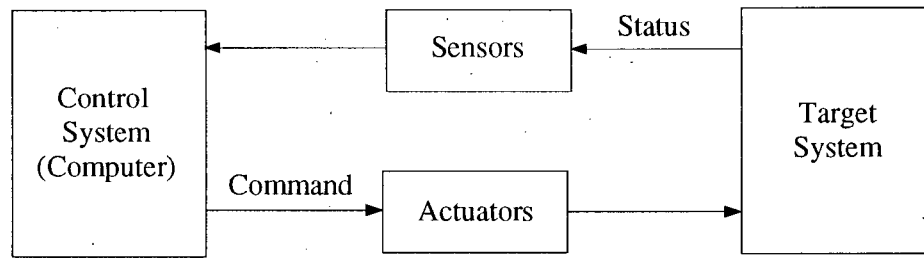


Figure I.1 A Typical Real-Time Control System.

1.3. Hybrid Systems

Since the advent of computers, and their widespread use in everyday life, a new class of systems has emerged called hybrid systems. The prime characteristic of hybrid systems is that they incorporate both continuous components, usually called plants, and also digital components such as digital computers, sensors and actuators controlled by programs. These programs are designed to select, control and supervise the behavior of the continuous parts. The control program reads the sensor data, sampled at discrete times, computes the next control law and imposes it on the plant.

The challenge is to develop methodologies which, given a performance specification and system description, extract control programs which will force the plants to meet their performance specifications. This objective cannot be met with any of the traditional modeling and analysis methodologies, which are aimed specifically at either of the continuous-time or discrete-event systems [14].

The modeling and analysis of hybrid systems is a complex task which has slowly been gaining attention. Examples of hybrid systems include robots, multimedia applications, communication networks, and supervisory control systems. The motivation for studying hybrid systems is to extend the scope of system theory to handle these examples. The ability to model such systems will provide a means for their control and performance evaluation.

The computer and its interfaces in all systems can be characterized as discrete-event dynamic systems (DEDS) [15]. The state of DEDS changes only when an event occurs. It is assumed that nothing important occurs between two successive events. The dynamics of the system are the result of complex interactions of various conditions and events in it [16]. Examples of an event are the pressing of a floor button in an elevator, arrival of a part in a manufacturing system, or failure of a computing system [17].

Another class of systems commonly encountered are those whose dynamics depend on time, referred to as time-driven systems. These systems (continuous time, discrete-time, and sampled-data) have been studied under traditional control system theory. In these systems, the maps from output measurements to control inputs are continuous. Time-driven systems are characterized by differential or difference equations and have been studied in much greater detail compared to DEDS.

There has been an increasing interest in the study of hybrid systems which has resulted in many approaches to their modeling and analysis. A good starting point for the study of these works are [18, 19, 20]. In the following some of these works are reviewed. Modeling and analysis of hybrid systems by Petri nets are reviewed at the end of next chapter after PNs are formally defined.

Hybrid systems are inherently concurrent and reactive [21]. A hybrid modeling methodology should incorporate techniques for specifying real-time constraints. Hybrid systems are mostly modeled as interacting networks of automata, possibly with infinite number of states, and input and output letters [22]. An automaton consists of sets of states, input symbols, output symbols, transition functions and initial conditions. Timed automata are defined as those which accept timed input strings [23]. This modeling is used for control of intelligent space vehicles [22]. The main problem with this approach is that it dichotomizes the system into symbolic (discrete) and non-symbolic (continuous) parts. As a result an interface is needed to convert continuous-time signals into sequence of symbols and vice versa. The complexity of the interface determines the ability to model and analyze various systems.

The interface given in [22] is quite simple and works only for constant plant inputs. More complex interfaces complicate the analysis of the overall system.

Another approach is to model the hybrid systems as products of nonlinear control and finite state automata [24]. According to this view, the automaton switches between the control systems, and that the switching is a function of the discrete input symbols or letters that it receives. The plant state-space is divided into regions (modes). For example, an aircraft control system may have climbing, descending and level flight modes. To go from one mode to a desired one needs a look-up table for suitable control [18]. The mode switching is quite ad hoc since even for simple continuous plants, the identification of possible behaviors is mathematically a very complex task. Moreover, identifying the effects of the proposed mode switching scheme is even more complicated. The stumbling block for the implementation of this scheme is the need for high speed database retrieval in real-time applications.

Manna and Pnueli use an extended version of discrete transition systems (itself an extended version of Statecharts), called phase transition systems to specify hybrid systems [25, 26]. Two types of semantics are considered. *Super dense semantics* is based on hybrid traces, and *sampling computations* sample the continuous behavior of a hybrid system at countably many observation points. The first one provides a more accurate description of the behavior whereas, the latter semantics is easier for verification. A compromise can be achieved by considering important events. Determination of important events is system dependent and requires a through knowledge of the system behavior which is very difficult for any moderately-complex system. The verification is based on an extension of temporal logic approach which has proven useful for the formal analysis of discrete systems. Their method uses sampled computation and important events together with an inductive proof rule to verify properties of hybrid systems [21].

Constraint nets are developed as an algebraic computation model of general dynamic systems [27, 28, 29]. The plants, control structure and environment are modeled in a single on-line framework under multiple levels of abstractions. The system requirements

are specified by temporal logic and timed \forall -automata. Extended linear temporal logic is used to represent temporal properties such as safety, liveness and real-time response. Timed \forall -automata are developed by defining timed states from \forall -automata and are used to represent any formula in real-time temporal logic. Model checking and stability analysis are used for behavior verification. The constraint nets approach falls short in analyzing the system properties such as controllability and observability. Such properties may possibly be specified and analyzed at the requirement specification stage but the current work does not address them explicitly. Another limitation is that this method cannot specify uncertainties using probabilistic and stochastic system behavior.

As a general observation it is worth noting that there seems to be a gap between the hybrid system modeling approaches attempted by the computer science and the control system research communities. The language used and the problem statement are very different even though they are aimed at modeling and analyzing the same systems. There is an urgent need for methods which can close this gap and pose the problem in a manner and structure which is understandable by both communities. The methodology developed in this thesis is an important step in this direction.

Figure I.2, which is adapted from [30], shows the scope of our research and its domain as a part of system theory classification. Our research is concerned with a restricted class of hybrid systems. They include the systems that can be modeled by the grey ovals in this classification. The term hybrid system in this thesis refers to the class of dynamic, time-invariant, discrete-time systems which are either event- or time-driven. Continuous-time systems which can be discretized (for example, by sampling) can also fall within this domain.

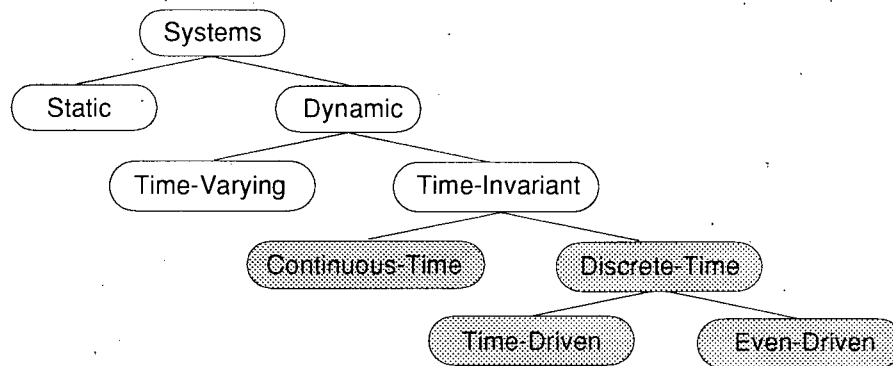


Figure I.2 System Classification and Thesis Scope.

I.4. Dissertation Contributions

The main contributions of this dissertation are the development of a method for the modeling and analysis of hybrid systems within the Petri Net framework and an examination of their fault-modeling and detection issues. These contributions can be listed as follows:

1. Development of a new methodology for modeling hybrid systems, based on a new extension of Petri net formalism. This extension, called global Petri net, can be used to model both discrete-time and discrete-event systems (Chapter III).
2. Definition and derivation of GPN dynamic equations (Chapter III).
3. Development of a set of analysis tools for checking the properties of systems modeled by GPNs. These tools examine the net properties, such as boundedness, stability, conservation, and controllability (Chapter IV). A sub-class of GPNs is also developed and defined, which can ease the analysis burden.
4. Development of a simulation and analysis package called GPNSAT. This package can model, simulate, and analyze any given hybrid system (Chapter V).
5. Application of the methodology developed to a hybrid system consisting of a hydraulic control system, its input-output interfaces, and a communication system. This system is simulated and analyzed for the relevant properties (Chapter VI).
6. Application of the above principles in modeling faults in a hybrid system. These fault models are simulated and analyzed by the GPN methodology (Chapter VII).

1.5. Dissertation Layout

This dissertation comprises eight chapters. The present chapter was devoted to problem specification and research motivation.

The second chapter provides the fundamental definitions related to the modeling of discrete-event systems by Petri nets (PNs). We define the PN structure and dynamics. These definitions are essential for following the rest of this thesis' notations and concepts. The reasons for choosing this platform, to solve the problem at hand, are given and its power and shortcomings are also described. A brief survey of some of the PN-related research in modeling control systems and fault analysis is provided at the end of this chapter.

Chapter III formally defines global Petri nets. Derivation of the GPN from conventional PN and derivation of the GPN dynamic equations are also given. At the end of this chapter some modeling examples are included. These examples are used to show the ease and power of modeling with the GPN, and are chosen from different areas of application.

Chapter IV is devoted to the investigation of GPN properties and analysis methods. First, some modeling issues, which will be useful in the later analysis, are discussed. The GPN modelability conditions are derived next. It is shown what sort of systems can be modeled with this net. Then, the GPNs hierarchy is discussed. The analysis methods used for GPNs are explained in the final two sections. The application of analysis methods in finding the net properties such as controllability, reachability, boundedness, and stability are also explained.

Chapter V describes the tool which has been developed to assist in the modeling, simulation, and analysis of hybrid and discrete-event systems. This tool is called the "Global Petri Net Simulation and Analysis Tool (GPNSAT)". Some of its salient features are presented and use of the tool is described.

In Chapter VI, application of the GPN in the modeling and analysis of a complete real-time hybrid control system is demonstrated. This system includes a hydraulic control system with its interfaces with the other parts of the system. The system is modeled as a distributed

computing system, with nodes dedicated to various tasks such as control, actuation, and operation.

Chapter VII is dedicated to fault modeling, simulation, and analysis by global Petri nets. The system, described in the previous chapter, is used to model and analyze faults in a hybrid system. The chapter starts with system level faults (bottlenecks) and then models and analyzes hydraulic system faults.

Conclusions are given in the last chapter. The scope and limitations of the GPN modeling and simulation are discussed. This chapter also includes some recommendations for future research directions.

Chapter II Petri Net Modeling of the Systems

Petri net (PN) theory [31, 32] was developed by Carl Petri [33] in 1962, primarily as an abstract and formal representation of information flow. Over the years it has turned into one of the most powerful tools for the modeling of systems exhibiting concurrency and synchronization characteristics. Petri nets in their various forms have been used to model and analyze systems in different application areas such as manufacturing [34–36], real-time processing [37, 38], computer architecture [39, 40], dynamic control [41, 42], supervisory control [43–45], material handling [46, 47], and robotics [48].

In this chapter we start by providing the basic Petri net concepts and definitions which are essential for following the rest of this thesis. We then present an outline of the reasons for choosing the Petri net as our modeling tool. At the end of this chapter, a survey of the works related to modeling of real-time control systems by Petri nets is given. This survey also includes the works which use PNs for fault detection and identification.

II.1. Petri Net Definition

The Petri net is a directed graph consisting of two types of nodes, called places and transitions. Weighted and directed arcs connect places to transitions, or vice versa. Any given system is modeled as sets of conditions and events. Places represent conditions, and transitions represent events. Each transition has a set of input and output places which represent the pre-conditions and post-conditions of the transition. The state of a net is modeled by the presence or absence of a token in the places. The number of tokens in a place is also referred to as the marking of the place. The initial marking represents the initial condition or the initial state of the net. The state of the net is changed by the firing of transitions, which models the events taking place. An event can happen only when its pre-conditions are satisfied and is represented by an enabled transition. The firing of a

transition changes the marking of its input and output places, modeling a change in its pre- and post-conditions. We now formally define a Petri net.

Definition 2.1. Petri Net: [31, 32] A Petri net (PN) is a five tuple structure defined as

$$PN = (P, T, W_{pt}, W_{tp}, M(0)), \quad (\text{II.1})$$

where $P = \{p_1, p_2, \dots, p_l\}$ is a finite set of places, and $l \geq 0$.

$T = \{t_1, t_2, \dots, t_n\}$ is a finite set of transitions, and $n \geq 0$ (any arbitrary place is represented as p , and any arbitrary transition as t).

There are two weight functions, W_{pt} and W_{tp} , which attach a positive integer weight to each arc of the net connecting places to transitions (pt) and transitions to places (tp), respectively.

The initial marking is represented by $M(0) = [m_1(0) \ m_2(0) \ \dots \ m_l(0)]^T$ and is a function from the set of places to the non-negative integers (superscript T in this thesis always refers to transpose of a matrix). The marking at any arbitrary time instant k is represented as $M(k) = [m_1(k) \ m_2(k) \ \dots \ m_l(k)]^T$ and sometimes is referred to as the number of tokens in each place.

The other formal Petri net definition which is widely used [31, 49] defines PN as:

$$PN = (P, T, I, O, M(0)), \quad (\text{II.2})$$

where $P, T, M(0)$ are the same as defined in Definition 2.1. I is an input mapping $P \times T \rightarrow \{0, 1\}$ (input incidence application [49]) corresponding to the set of directed arcs from P to T . These arcs are called input arcs. O is an output mapping $T \times P \rightarrow \{0, 1\}$ (output incidence application [49]) corresponding to the set of directed arcs from T to P . These arcs are called output arcs.

The conversion between these two forms of definition is straightforward and is governed by the following equations.

$$W_{pt} = [W_{p_it_j}], \quad (\text{II.3})$$

where $W_{p_it_j} = I(P_i, T_j)$ and

$$W_{tp} = [W_{t_j p_i}], \quad (\text{II.4})$$

where $W_{t_j p_i} = O(T_j, P_i)$. W_{pt} and W_{tp} are called input and output incidence matrices.

II.2. Petri Net Graph

The graphical representation is one of the attractive features of Petri nets. It allows a precise and easily understood display of the formal theory. Places and transitions are represented by circles and bars, respectively. Arcs are shown by arrows, and tokens by small dots inside the places. Arc weights are represented by numbers placed close to the arcs. Absence of a number indicates a weight equal to one.

II.3. Petri net Dynamics

The state of a net is represented by the number of tokens in each place. The movement of tokens between places describes the dynamics of the net, and is accomplished by firing of the enabled transitions. Let $\bullet x$ and x^\bullet be called the preset and postset of x , where x is either a place or a transition.

Definition 2.2. Enabled Transition: [31, 32] Places p_i and p_o are called input and output places of transition t if they belong to the sets $\bullet t$ and t^\bullet , respectively. Then, transition $t \in T$ is called enabled under a marking $M(k)$ of a PN iff $\forall p_i \in \bullet t : M_{p_i}(k) \geq W_{pt}(p_i, t)$.

An enabled transition can be fired which yields a new marking, given by

$$M_{p_i}(k+1) = M_{p_i}(k) - W_{pt}(p_i, t) \quad \text{for all } p_i \in \bullet t, \quad (\text{II.5})$$

$$M_{p_o}(k+1) = M_{p_o}(k) + W_{tp}(t, p_o) \quad \text{for all } p_o \in t^\bullet, \quad (\text{II.6})$$

$$M_p(k+1) = M_p(k) \quad \text{otherwise.} \quad (\text{II.7})$$

II.4. An Example of a PN

The following example is meant to illustrate the above concepts and definitions. Figure II.3 shows a Petri net which models a printing process. There are five places representing various conditions, such as paper availability and printer queue, as defined by $P = \{p_1, p_2, p_3, p_4, p_5\} = \{ \text{Paper Available, Printer Queue, Printer Idle, Printing, Print Ready} \}$. There are also three transitions, given as $T = \{t_1, t_2, t_3\} = \{ \text{Print Request Arrives, Start Printing, Finish Printing} \}$. Printing can start when there is enough paper, a print request, and an idle printer.

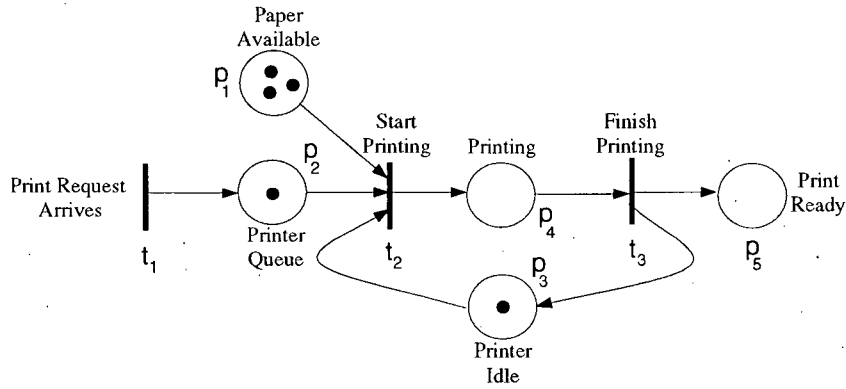


Figure II.3 A Simple Example of a Petri Net Modeling a Printing Process.

Arc weight matrices are written as:

$$W_{pt} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad W_{tp} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Each element of the arc weight matrices, W_{pt} and W_{tp} , represents an arc connecting a place to a transition or vice versa, respectively. For example $W_{pt}(1, 2) = 1$ indicates that there is an arc connecting place p_1 to transition t_2 with an arc weight equal to one.

The initial marking, with reference to the figure, is written as:

$$M(0) = [3 \quad 1 \quad 1 \quad 0 \quad 0]^T,$$

which indicates that there are three units of paper, one print request, and an idle printer. Under this condition, the transition representing “start printing” is enabled. This transition fires by removing a token from each of its input places (paper available, printer queue, and printer idle). It then adds a token to its output transition, showing that printing is taking place. The place marking vector then becomes:

$$M(1) = [2 \ 0 \ 0 \ 1 \ 0]^T.$$

The printing completion is modeled by the firing of transition t_3 . This transition removes a token from the place which is modeling the printing job (p_4) and adds a token to the print ready place (p_5). This state is presented by the following marking:

$$M(2) = [2 \ 0 \ 1 \ 0 \ 1]^T.$$

The next printing job starts once there is a new request in the printer queue, represented by a token in the place p_2 .

II.5. Why Petri Nets?

Many modeling and analysis methods, such as dataflow graphs [50, 51], higraph and statechart [52, 53], state machine design, constraint modeling [54, 29, 28], and structured design have been used to model and design various computing systems. Each of these methods has found success in the design of a particular type of system. However, if we compare the PN modeling with each of the above methods, it offers one or more of the following advantages over these methods. These advantages are listed to justify our selection of this method:

1. The model can represent concurrency and synchronization, which are integral parts of any computer system.
2. Both hardware and software can be represented by this model.
3. PN models can be developed and analyzed for various abstraction levels from system level transactions down to the circuit logic level [39, 55]. As noted in [56], a hierarchical

description is not only desirable, but is essential since it is impossible to describe any real system with a sufficient degree of detail in a single model.

4. There have been already many methods developed to analyze the Petri net model [57, 58]. Some of these methods are marking tree, reachability, liveness, boundedness, and invariance analysis [59, 60].
5. There is already a well-established interest in the PN theory which can benefit our model development efforts. There are many commercial PN tools which can be useful for modeling, analysis, and simulation of the model [61, 62]. There are at least two World Wide Web and FTP sites which provide the latest information on Petri nets and the latest tools [63, 64] available.
6. PNs are very easy to understand and work with, due to their graphical and precise representation scheme.
7. With the addition of temporal and stochastic specifications, the PN provides a structured framework for system simulation and performance evaluation.
8. PNs can be used in the various steps of system development and operation, such as requirement definition, design, testing, simulation and, on-line replication.

There are, however, some drawbacks to and trade-offs in using Petri nets which have to be considered. These are the following:

1. Only discrete-event (asynchronous) systems can be modeled with the basic Petri nets. In most real-time control systems, we encounter many elements which have to be modeled as time-driven systems.
2. Even though the basic Petri net has a mix of simplicity and power of expression for essential interactions, it needs further capabilities to be able to model the complexities of actual systems [56]. Moreover, useful extensions for real-time applications are still limited.
3. Modeling, analysis and simulation of a PN is very expensive and requires a large amount of processing power and time.

4. Some of the analysis methods are restricted to subclasses of PNs. This issue is addressed in Chapter IV on GPN analysis methods.

Our extension of the Petri net theory, which will be formally presented in the next chapter, is meant to address some of these shortcomings. Here we would like to point out the followings:

1. Hierarchical modeling and distributed detection ease the processing burden and allow more detailed modeling only when it is needed.
2. A new extension to the existing PN is presented which will greatly increase its modeling capabilities and ease of modeling for real-time systems.
3. This new extension has the capability of modeling hybrid systems with both discrete-time and event elements.

II.6. Assigning Time to Petri Nets

There are two approaches for adding temporal specifications to the basic PN. A stochastic PN (SPN) [65] is obtained by associating exponentially-distributed firing times to the transitions. Stochastic models are well-suited to performance evaluation but are not suitable for the modeling of real-time systems. In real-time systems, we are concerned with the worst case timing requirements because the system receives input from uncontrollable environments and not meeting the timing requirements may have catastrophic results. Thus, SPNs cannot be satisfactorily used for modeling real-time systems.

The other method for temporal specification is to assign time to transitions or, alternatively, to places [66, 48]. In the first extension, time is associated with each of the transitions of the original model. With this model, transition firing is no longer an instantaneous event as defined in the original PN model. Another extension associates a delay to each place in the classical net. It has been shown that assigning time to a transition or a place are equivalent, and any one of them can be transformed to the other one [67].

The two most important extensions are time PN and timed PN (TPN). The time PN extension [68] defines and associates an interval $[t_{min}, t_{max}]$, with each transition, which represents the minimum and maximum time during which an enabled transition should fire. Timed PNs are formed simply by associating a transition firing time to each transition [66].

II.7. Review of Some Relevant Petri nets

In this section we review some of the prominent research in Petri net modeling which has been aimed at modeling dynamic systems and/or fault detection and identification. We will briefly describe some of their salient features and discuss their advantages and shortcomings. The interested readers are referred [69–75] for further information about these issues.

II.7.1. Fault Detection by Petri Nets: Petri nets have been used to model a pressurized water reactor's cooling loop [76]. This model is used to detect faults such as leaks in vessels or temperature drifts in the sensors. This method is suitable for detecting failures with very long time constants. Fault detection is performed by checking whether the number of tokens in the places representing the cooler loop remains constant (conservativeness property). The actual number of tokens per place is compared with the initial token content of the total process. The method presented is applicable only to systems which have physical conservation qualities. In addition, this method can only detect fault as no attempt is made at recognizing its cause. In other words, it cannot localize the fault.

II.7.2. Digital Control Systems: The main purpose of this work [77] is to describe a programmer workbench. A workbench is a package which helps with the creation of a program, in which the code to be tested runs, together with a comprehensive diagnostic utility. A Petri net model is used to define how the utilities of the workbench and the different parts of the control system exchange data. Places and transitions represent data storage (mailboxes) and functions, respectively. Tokens are used to show if a place has all the expected data. This method does not model time-driven processes but instead represents them as a set of conditions and discrete-events.

II.7.3. Petri Nets as Discrete Controllers: This paper [78] presents an approach to the specification, modeling, and analysis of discrete manufacturing systems. Petri nets have been used for controller specification and implementation. The control computer uses a PN model of the controlled system, using the marking of the net, to determine control actions. In generating a PN model, state values in the systems specification are mapped into distinct PN places. For example, state X, with the possible state values a, b, c, and d, would be mapped onto four PN places. This sort of modelling can be used only for systems which have a limited number of states with just a few discrete values. Otherwise, the net size can explode even for small examples. The reduced reachability graphs are used which permit efficient evaluation of the state behavior of the system sub-components.

II.7.4. Failure Modeling and Analysis in a Material Handling System: Petri nets are used for modeling, simulation, and analysis of failures in a material handling system [47]. A new extension of Petri nets is used called Extended PN (EPN). Six types of places, such as action, sink, and switch are used to model different conditions which arise in the system [46]. Failures considered in the system include both hard failures (in conveyer belts, camera, or the robots) and soft failures (due to transient faults). Reachability graphs are used to analyze safeness, liveness and reversibility properties.

II.7.5. Continuous Petri Nets: The markings in an ordinary Petri net have either a binary or an integer value. A binary marking represents the validity of a condition whereas, an integer marking may signify the number of clients in a queue or the number of parts in a resource.

The continuous Petri net is a model in which the number of marks in the places are positive real numbers [79,49]. The inspiration for having real number markings comes from research in production systems in which the number of parts is modeled by real numbers.

In this model each mark is cut into infinitely smaller pieces. A transition is enabled if all of its input places have a marking greater than zero (in conventional Petri nets, the markings

should be greater than or equal to one). A quantity τ_j of transition T_j can be fired with $0 \leq \tau_j \leq \min(M(p_a), \dots, M(p_b))$, where $(M(p_a), \dots, M(p_b))$ is the set of input places of T_j . Then, τ_j is known as a firing quantity. In each transition firing, this quantity is subtracted and added to the input places and output places of the firing transition, respectively.

The difference between a discrete PN and a continuous PN is not a structural difference, since they differ only in their markings. Therefore, all structural properties which hold true for the former are true for the latter as well.

Timed continuous Petri nets are formed by assigning a firing speed to each transition. A transition can start firing when it is enabled. Firing frequency or speed is taken to be the inverse of transition delay defined as transition time for the timed Petri nets. Once a transition fires, tokens are transferred according to the firing speed of the transitions. In this way, the marking of a net changes continuously instead of in discrete steps, as in Petri nets. There are two approximations in calculating the firing speeds associated with each transition. The resulting models, corresponding to these approximations, are called constant speed continuous Petri nets (CCPN), and variable speed continuous Petri nets (VCPN) [49,80].

The main problem with this methodology is that the variable speed is dependent upon the markings of the net. Computation of the firing speed for transitions with variable speed becomes very expensive. In addition, construction of evolution graphs, which are used instead of the reachability trees, is more complicated. In these graphs one needs to include the firing speed of each transition. These speeds keep changing with changes in the markings. Finally, determination of enabled transitions, even in the case of CCPN, is not as straightforward as for simple PNs and this determination needs a complex algorithm to be implemented.

II.7.6. Hybrid Petri Nets: Hybrid Petri nets have been defined by the same group of researchers who have developed the continuous PN [80,41,49]. A hybrid PN is composed of both discrete PN places and transitions (D-places and D-transitions) and continuous PN places and transitions (C-places and C-transitions). Markings and arc weights of the continuous parts can take any positive real number value, just as with continuous PNs.

There are a few issues which should be addressed in evaluating the hybrid Petri net methodology. The first two of these issues have to do with the approximations which are made in modeling different processes. The first approximation is with regard to the markings. Each mark is divided into smaller pieces called tokens. For this approximation to approach the continuous space that it is intended for, the token size should be infinitely small. Since this division cannot be by infinity, any division is in fact a discretization but probably at smaller granularity.

The second issue to consider here is that the continuous process or firing is just an approximation of a discrete event [80], and is not a time-driven process. A transition cannot fire if its input conditions are not satisfied (if there is no token in its input places), clearly indicating that it is event-driven and not time-driven. In a hybrid PN, "continuous" really means breaking down a discrete-event which, for example, takes two seconds to occur into a large number of small instances of time. If we define hybrid systems as those which have both event-driven and time-driven processes [15], then hybrid PNs would be a misnomer, since in fact all transitions in a hybrid PN are event-driven.

The last issue is the limitations in terms of the markings which can be represented by this method. Since the marking is limited to positive real values only, we cannot have any negative markings. The developers of this method have not felt the necessity of having negative markings since their prime application area is the modeling of production systems. In these systems, positive real numbers would suffice for modeling resources, queues, and parts. But when one considers other application areas such as modeling of dynamic and control systems, the necessity for markings with negative values becomes evident.

Chapter III Global Petri Nets: Definitions and Examples

This chapter describes various features and attributes of a new extension of Petri nets called the GPN (Global Petri Net). We start with our formal definitions of timed Petri nets [66] and the GPN and then describe the GPN operation, using an example. The derivation of the GPN from a conventional PN and the derivation of the GPN dynamic equations are given in the next two sections. A comparison of the PN and GPN modeling is also presented and discussed. Next, various place and transition types allowed in the GPN formalism are described. The final section provides two modeling examples to demonstrate GPN modeling capabilities.

There are many extensions suggested by various researchers which are meant to increase the modeling power of PNs. These extensions usually are aimed at a particular application or area of interest and lack a generality which could benefit other classes of applications. Moreover, any increase in modeling power is accompanied by a decrease in the analytical power of the net [31]. In this section, we introduce our extensions to the original PN and call the resulting net a Global Petri Net (GPN). The global Petri net is developed to furnish a new methodology for modeling real-time control systems. This modeling tool has the advantage of preserving many of the PN analysis capabilities, as shown in the next chapter.

The Global Petri net (GPN) is a concise extension of the original Petri nets which enables one to model more complex systems. This new extension is developed for modeling hybrid systems which have both time-driven and event-driven parts. The GPN is very general and facilitates the modeling of any kind of digital system, including the digitized versions of analog plants and computer hardware and software. This class of systems covers a large area of applications, including systems in real-time control, robotics, and manufacturing.

Since the concept of time is essential in the definition of the GPN, we first present our method of adding temporal specifications to the original Petri net which was defined in the previous chapter.

III.1. Timed Petri Net Definition

In the present modeling methodology, we associate time with transitions. This approach is very close to the way time is represented in actual real-time control systems.

Definition 3.1. Timed Petri Net (TPN): Timed Petri net (TPN) can be defined formally as

$$TPN = (PN, TT), \quad (III.12)$$

where PN is as defined in Equation (II.1), and TT is an n-vector of $0 \leq tt < \infty$ transition times specified in k sampling times. Each transition, t_n , takes tt_n sample periods to complete. This time corresponds to the maximum time a transition would take to complete in an actual system, without causing any fault or noticeable performance degradation. A transition fires as soon as all its pre-conditions are met and takes a maximum of tt sampling time to finish. The firing starts by removing tokens from the firing transition's input places. The transition in this time period is busy and therefore cannot be fired unless the previous firing has ended. Firing ends by updating all of its output places.

III.2. Global Petri Net Definition

Places in a GPN correspond to system parameters, variables, or states. The system dynamics can be observed by looking at the places, which collectively describe the state which the system is in. Transitions represent processes or operation of various components. A process can be as simple as an addition operation of two numbers, or as complex as an entire system structure.

Tokens are the major source of departure of the GPN from the original PN. Tokens in the original Petri net have a binary value and therefore result in markings that can have

only positive integer values. In the global Petri net, markings can take any real number value, including negative values. The positive integer restriction on tokens limits the type of systems which can be modeled. The token in this case can only represent the validity of certain conditions (for example presence and number of parts in a manufacturing system). The inclusion of markings with any real number value enables us to model any state even those which take negative values.

Another major deviation from the conventional PN is the types of arcs which are allowed under GPN formalism. There are two types of arcs in a GPN. Event-driven or asynchronous arcs (whose weights are assigned by arc weight matrices W_{pt} and W_{tp}) are the same as those defined for the Petri net models (Equation II.1). In addition, there are the synchronous (time-driven) arcs, which are represented by A and B matrices. When A and B matrices have all zero elements, a GPN reduces to a PN.

Definition 3.2. Global Petri Net (GPN) : GPN is a triple structure defined as:

$$GPN = (TPN, A, B), \quad (III.13)$$

where

$$TPN = (P, T, W_{pt}, W_{tp}, M(0), TT).$$

P, T, W_{pt} , W_{tp} and TT are the same as those defined for the TPN (Equation III.12). $M(0)$ is an m-vector of initial markings of real numbers, A is an $l \times n$ matrix of real value arc weights drawn from places to transitions, and B is an $n \times l$ matrix of real value arc weights drawn from transitions to places.

III.3. GPN Graph

The GPN graph is very similar to the PN graph. The only difference is that tokens are represented as numbers instead of dots inside the places. Synchronous arcs with arc

weights A and B are shown by double arrow arcs. The transition time is scribed beside the transitions. The default value for transition time is one sampling period and may be omitted from the graph and the definition.

III.4. GPN Dynamics

In a GPN, firing is an execution of a transition by which the value of one or more markings in the corresponding places are changed. State evolution or dynamics of the net are represented by changes in the marking.

Every transition can have both input event (asynchronous) arcs, whose arc weights are given by W_{pt} and time (synchronous) arcs, with arc weights given by the A matrix. A transition which has no input event arc always is enabled, and fires according to its transition time, or in other words, it will fire as soon as it is not busy.

Let sets of all input and output places of a transition t , be denoted as $\bullet t$ and t^\bullet , respectively. Each individual place belonging to one of these sets can be written as p_i and p_o , respectively. Transitions which have input asynchronous arcs must meet the PN firing condition, which is $\forall p_i \in \bullet t : M_{p_i}(k) \geq W_{pt}(p_i, t)$.

Firing of the transition t results in the following changes in the current state (marking) of input places of the transition t :

$$M_{p_i}(k+1) = M_{p_i}(k) - A(p_i, t)M_{p_i}(k) - W_{pt}(p_i, t) \quad \text{for all } p_i \in \bullet t. \quad (\text{III.14})$$

$M_{p_i}(k)$ and $M_{p_i}(k+1)$ represent before and after firing markings of input place p_i . $A(p_i, t)$ and $W_{pt}(p_i, t)$ represent elements of the arc weight matrices A and W_{pt} , connecting place p_i and transition t . For transition output places, the change in marking can be represented as:

$$M_{p_o}(k+1) = M_{p_o}(k) + B(t, p_o)M_{p_i}(k) + W_{tp}(t, p_o) \quad \text{for all } p_o \in t^\bullet \text{ and } p_i \in \bullet t. \quad (\text{III.15})$$

Places, which do not belong to either of transition input or output places, are not affected by the firing; therefore, we can write:

$$M_p(k+1) = M_p(k) \quad \text{for all } p \notin \{\bullet t, t^\bullet\}. \quad (\text{III.16})$$

These concepts are illustrated through an example in the next section. The equations for the system dynamics are derived in Section III.7 of this chapter. Modeling advantages and the GPN capabilities are explored more fully through some other examples at the end of this chapter.

III.5. An Example of a GPN

In this section we present a simple example to show the dynamics of a GPN. Figure III.4 shows a net with three places and three transitions.

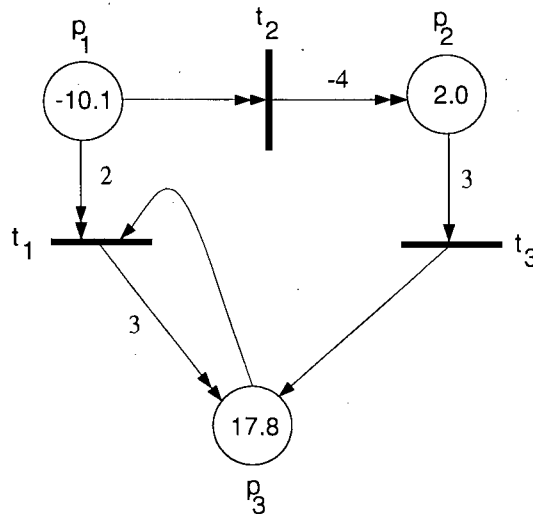


Figure III.4 An Example of a Simple GPN.

Net parameters, as defined in Equation (III.13), can be written from the figure as:

$$\begin{aligned}
P &= \{p_1, p_2, p_3\} \\
T &= \{t_1, t_2, t_3\} \\
W_{pt} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 3 \\ 1 & 0 & 0 \end{bmatrix} & W_{tp} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
A &= \begin{bmatrix} 2 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & B &= \begin{bmatrix} 0 & 0 & 3 \\ 0 & -4 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
M(0) &= \begin{bmatrix} m_1(0) \\ m_2(0) \\ m_3(0) \end{bmatrix} = \begin{bmatrix} -10.1 \\ 2.0 \\ 17.8 \end{bmatrix}
\end{aligned} \tag{III.17}$$

This example has three different types of transitions. t_2 is a synchronous transition since it has no asynchronous input arcs. This transition does not have a pre-condition for its firing and fires instantly when it completes the previous firing. t_3 is an asynchronous transition and fires only when its pre-condition is satisfied, that is when its input place marking is greater than the arc weight connecting them. The arc weight ($W_{pt}(2, 3)$), in this case, is equal to three. t_1 is a hybrid transition since it has both types of input arcs. A complete discussion of the GPN arc types is given in Section III.9. Now we can go through a series of firing to show how the state of the net changes due to the firing of various transitions.

At time instant $k = 0$, the initial marking is $M(0) = [M_1(0) \ M_2(0) \ M_3(0)]^T = [-10.1 \ 2.0 \ 17.8]^T$. Under this marking, transitions t_1 and t_2 are enabled. t_1 is enabled since $M_3(0) = 17.8$, which is greater than $W_{pt}(3, 1) = 1$. t_2 is a synchronous transition so it is always enabled. Transition t_3 is not enabled since the marking of place $p_2 = 2.0$ is smaller than the arc connecting this place to transition t_3 . The sets of input and output places for these enabled transitions can be written as:

$$\begin{aligned}
\bullet t_1 &= \{p_1, p_3\}, & t_1^\bullet &= \{p_3\}, \\
\bullet t_2 &= \{p_1\}, & t_2^\bullet &= \{p_2\}.
\end{aligned} \tag{III.18}$$

The firing of transitions t_1 and t_2 results in changes in their input place (p_1 and p_3) markings. By substituting appropriately from Equation (III.18) in Equation (III.14), we have

the following:

$$\begin{aligned} M_1(1) &= M_1(0) - A(1,1)M_1(0) - A(1,2)M_1(0) \\ M_3(1) &= M_3(0) - W_{pt}(3,1) . \end{aligned} \quad (\text{III.19})$$

Changes in the output places' (p_2 and p_3) markings due to these transitions firing can be found by substituting the arc weights and initial makings, in Equation (III.15):

$$\begin{aligned} M_2(1) &= M_2(0) + B(2,2)M_1(0) \\ M_3(1) &= M_3(0) + B(1,3)M_1(0) . \end{aligned} \quad (\text{III.20})$$

The overall effect of the firing of transitions t_1 and t_2 can be written by combining Equations (III.19) and (III.20).

$$\begin{aligned} M_1(1) &= M_1(0) - A(1,1)M_1(0) - A(1,2)M_1(0) \\ M_2(1) &= M_2(0) + B(2,2)M_1(0) \\ M_3(1) &= M_3(0) + B(1,3)M_1(0) - W_{pt}(3,1) . \end{aligned} \quad (\text{III.21})$$

The new markings can be obtained by substituting the net parameters (Equation III.17) in the above equations.

$$\begin{aligned} M_1(1) &= -10.1 - 2(-10.1) - 1(-10.1) = 20.2 \\ M_2(1) &= 2 - 4(-10.1) = 42.4 \\ M_3(1) &= 17.8 + 3(-10.1) - 1 = -13.5 \end{aligned}$$

or

$$M(1) = [20.2 \quad 42.4 \quad -13.5]^T . \quad (\text{III.22})$$

With this new marking transition, t_3 becomes enabled, but t_1 gets disabled since the marking of p_3 is negative. The sets of input and output places for the enabled transitions

t_2 and t_3 can be written as:

$$\begin{aligned} \bullet t_2 &= \{p_1\}, & t_2^\bullet &= \{p_2\}, \\ \bullet t_3 &= \{p_2\}, & t_3^\bullet &= \{p_3\}. \end{aligned} \quad (\text{III.23})$$

The firing of transitions t_2 and t_3 results in a new marking:

$$\begin{aligned} M_1(2) &= M_1(1) - A(1,2)M_1(1) = 20.2 - 20.2 = 0 \\ M_2(2) &= M_2(1) + B(2,2)M_1(1) - W_{pt}(2,3) = 42.2 + (-4)20.2 - 3 = -41.6 \\ M_3(2) &= M_3(1) + W_{tp}(3,3) = -13.5 + 1 = -12.5 \end{aligned}$$

or

$$M(2) = [0 \quad -41.6 \quad -12.5]^T. \quad (\text{III.24})$$

Under this new marking, only transition t_2 is enabled. However, its firing cannot change any of the markings since marking of its input place p_1 , at the present time instant is zero.

III.6. Derivation of the GPN from Conventional PN

In this section we show how the GPN structure can be derived from the conventional Petri nets. Rewriting the equation for Petri net dynamics, Equation (II.5):

$$M_{p_i}(k+1) = M_{p_i}(k) - W_{pt}(p_i, t) \quad \text{for all } p_i \in \bullet t. \quad (\text{III.25})$$

At time instants $k = 0, 1, 2, \dots, k$, while the transition is enabled, we have:

$$\begin{aligned} M_{p_i}(1) &= M_{p_i}(0) - W_{pt}(p_i, t) \quad \text{for all } p_i \in \bullet t \\ M_{p_i}(2) &= M_{p_i}(1) - W_{pt}(p_i, t) = M_{p_i}(0) - 2 \times W_{pt}(p_i, t), \\ &\vdots \\ M_{p_i}(k) &= M_{p_i}(k-1) - W_{pt}(p_i, t) = M_{p_i}(0) - k \times W_{pt}(p_i, t). \end{aligned} \quad (\text{III.26})$$

Rewriting the counterpart equation of GPN dynamics (only the synchronous elements), equation (III.14), and substituting for time instant $k=1$, we get

$$\begin{aligned}
M_{p_i}(k+1) &= M_{p_i}(k) - A(p_i, t)M_{p_i}(k) \quad \text{for all } p_i \in \bullet t, \\
M_{p_i}(1) &= M_{p_i}(0) - A(p_i, t)M_{p_i}(0).
\end{aligned}
\tag{III.27}$$

Comparing equations (III.26) and (III.27), if we choose

$$W_{pt}(t, p_i) = \frac{A(p_i, t)M_{p_i}(0)}{k}, \tag{III.28}$$

and run the net for k time samples, we get the same result as if we had run the equivalent GPN for only one sample time. This is one of the reasons it is more concise to model the dynamics of a complex system by a GPN.

Similarly if we compare equations (II.6) and (III.15), we need to choose

$$W_{tp}(t, p_o) = \frac{B(t, p_o)M_{p_i}(0)}{k}, \tag{III.29}$$

and run the PN for k time samples, to get the equivalent GPN.

III.7. Derivation of GPN Dynamics Equation

In this section we develop the equations which govern the dynamics of a GPN. This derivation is carried out for a general example and is extended for any given net. In the following derivation we need to define two functions. These two functions are used in the equations which describe GPN dynamics equations. Their importance and usefulness become clear later, when they are used in the derivation.

Definition 3.3. Diagonalizing (*Diag()*) Function: This function adds the elements of a matrix row and puts the result in the diagonal element of that row. It is denoted by *Diag()*.

The following shows its operation:

$$\begin{aligned} \text{If } \Theta &= \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \\ \theta_{31} & \theta_{32} \end{bmatrix}, \text{ then} \\ \text{Diag}(\Theta) &= \begin{bmatrix} \theta_{11} + \theta_{12} & 0 & 0 \\ 0 & \theta_{21} + \theta_{22} & 0 \\ 0 & 0 & \theta_{31} + \theta_{32} \end{bmatrix}. \end{aligned} \quad (\text{III.30})$$

Definition 3.4. One() Function: This function replaces all non-zero elements of a matrix with one. Zero elements remain the same. This function is written as *One()*. The following is an example of its use:

$$\begin{aligned} \text{If } S &= \begin{bmatrix} 2.1 & 0 \\ -3 & 1 \end{bmatrix}, \text{ then} \\ \text{One}(S) &= \text{One}\left(\begin{bmatrix} 2.1 & 0 \\ -3 & 1 \end{bmatrix}\right) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \end{aligned} \quad (\text{III.31})$$

The following is a general form of a two-place, two-transition net with all possible arcs among them included. Double and single arrow arcs represent time-driven (synchronous) and event-driven arcs, respectively.

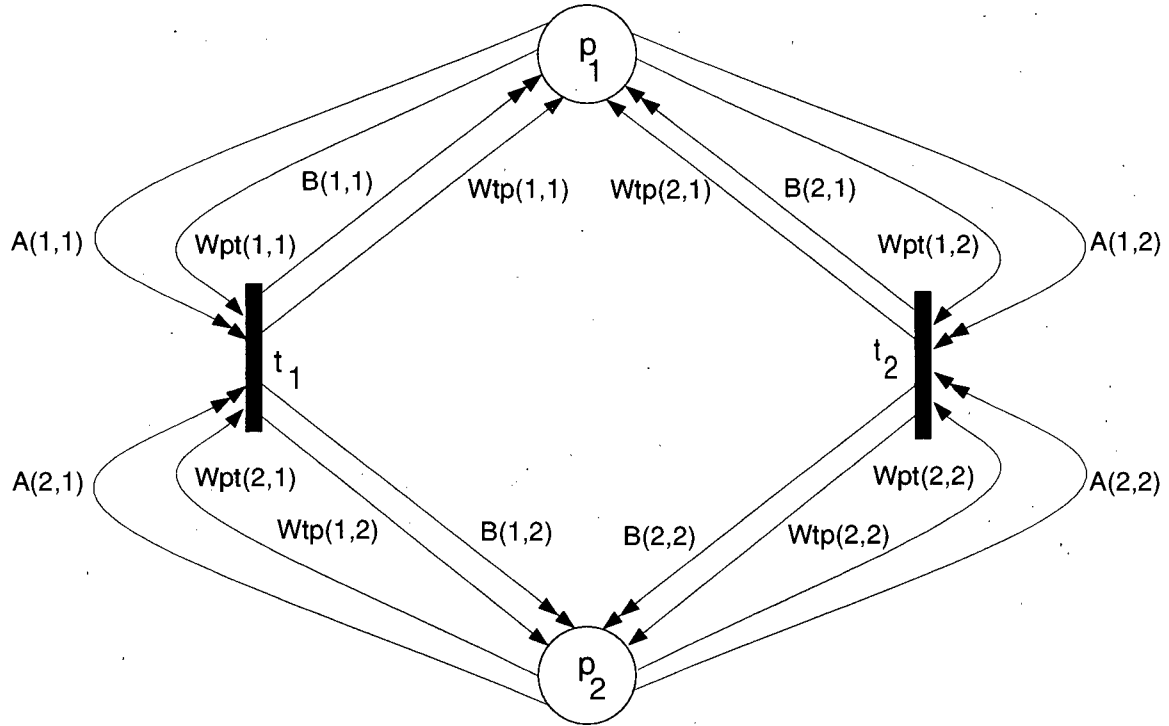


Figure III.5 A General GPN with all Possible Arcs

Net parameters for this example are written as

$$\begin{aligned}
 P &= \{p_1, p_2\} & T &= \{t_1, t_2\} \\
 W_{pt} &= \begin{bmatrix} W_{pt}(1,1) & W_{pt}(1,2) \\ W_{pt}(2,1) & W_{pt}(2,2) \end{bmatrix} & W_{tp} &= \begin{bmatrix} W_{tp}(1,1) & W_{tp}(1,2) \\ W_{tp}(2,1) & W_{tp}(2,2) \end{bmatrix} \\
 A &= \begin{bmatrix} A(1,1) & A(1,2) \\ A(2,1) & A(2,2) \end{bmatrix} & B &= \begin{bmatrix} B(1,1) & B(1,2) \\ B(2,1) & B(2,2) \end{bmatrix} & M(0) &= \begin{bmatrix} M_1(0) \\ M_2(0) \end{bmatrix}
 \end{aligned}$$

Definition 3.5. Petri Net Incidence Matrix N: N is called the Petri net incidence matrix and is defined as

$$N = W_{tp}^T - W_{pt} = \begin{bmatrix} W_{tp}(1,1) - W_{pt}(1,1) & W_{tp}(2,1) - W_{pt}(1,2) \\ W_{tp}(1,2) - W_{pt}(2,1) & W_{tp}(2,2) - W_{pt}(2,2) \end{bmatrix}$$

Definition 3.6. Petri Net Transition Firing Vector: In a Petri net, the firing of various transitions is represented by a firing vector with a size equal to the number of transitions. Every element of the vector corresponds to a transition. A non-zero entry in the vector shows the number of firings of the respective transition. For example, $f_k = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$ indicates that at time instant k , transition one is fired twice, whereas transition two is not fired at all.

Definition 3.7. Global Petri Net Transition Firing Matrix: In a GPN, the firing sequence is shown both by a vector and a matrix. The firing sequence vector, f_k , is used for the asynchronous part of the net. F_k which is used for the synchronous part, is a square matrix of size n (number of transitions). This matrix is diagonal, and each element of the main diagonal corresponds to a transition. The relation between the firing sequence vector and the GPN firing matrix is

$$F_k = \text{Diag}(f_k). \quad (\text{III.34})$$

In this thesis we use both $\text{Diag}(f_k)$ and F_k interchangeably to refer to the firing sequence matrix.

For the GPN in Figure III.5, assuming transition t_1 is enabled, the transition firing vector at time instant $k = 0$ will be $f_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. This transition firing will change the net marking to

$$\begin{aligned} M_1(1) &= M_1(0) - A(1,1)M_1(0) \\ &+ B(1,1)M_1(0) + B(1,1)M_2(0) - W_{pt}(1,1) + W_{tp}(1,1), \\ M_2(1) &= M_2(0) - A(2,1)M_2(0) \\ &+ B(1,2)M_1(0) + B(1,2)M_2(0) - W_{pt}(2,1) + W_{tp}(1,2), \end{aligned}$$

Or

$$\begin{aligned} M(1) &= \begin{bmatrix} M_1(0) \\ M_2(0) \end{bmatrix} - \begin{bmatrix} A(1,1) & 0 \\ 0 & A(2,1) \end{bmatrix} \begin{bmatrix} M_1(0) \\ M_2(0) \end{bmatrix} \\ &+ \begin{bmatrix} B(1,1) & B(1,1) \\ B(1,2) & B(1,2) \end{bmatrix} \begin{bmatrix} M_1(0) \\ M_2(0) \end{bmatrix} + \begin{bmatrix} W_{tp}(1,1) - W_{pt}(1,1) \\ W_{tp}(1,2) - W_{pt}(2,1) \end{bmatrix}. \end{aligned} \quad (\text{III.35})$$

Using the $Diag()$ and $One()$ functions, we can write the following:

$$\begin{aligned}
 \begin{bmatrix} A(1,1) & 0 \\ 0 & A(2,1) \end{bmatrix} &= Diag \left(\begin{bmatrix} A(1,1) \\ A(2,1) \end{bmatrix} \right) \\
 &= Diag \left(\begin{bmatrix} A(1,1) & A(1,2) \\ A(2,1) & A(2,2) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) = Diag(Af_0) , \\
 \begin{bmatrix} B(1,1) & B(1,2) \\ B(2,1) & B(2,2) \end{bmatrix} &= \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} B(1,1) & B(1,2) \\ B(2,1) & B(2,2) \end{bmatrix}^T \\
 &= \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} B(1,1) & B(1,2) \\ B(2,1) & B(2,2) \end{bmatrix}^T \\
 &= [One(A)Diag(f_0)B]^T .
 \end{aligned} \tag{III.36}$$

We can also write the following expression for the asynchronous part of the net:

$$\begin{aligned}
 &\begin{bmatrix} W_{tp}(1,1) - W_{pt}(1,1) \\ W_{tp}(1,2) - W_{pt}(2,1) \end{bmatrix} \\
 &= \begin{bmatrix} W_{tp}(1,1) - W_{pt}(1,1) & W_{tp}(2,1) - W_{pt}(1,2) \\ W_{tp}(1,2) - W_{pt}(2,1) & W_{tp}(2,2) - W_{pt}(2,2) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = Nf_0 .
 \end{aligned} \tag{III.37}$$

Substituting from expressions (III.36) and (III.37) in equation (III.35), we get

$$M(1) = M(0) - Diag(Af_0)M(0) + [One(A)Diag(f_0)B]^T M(0) + Nf_0 . \tag{III.38}$$

Now,

$$H_0 = \left(-Diag(Af_0) + [One(A)Diag(f_0)B]^T \right) \tag{III.39}$$

is called the hybrid transition matrix (H matrix for short). This matrix is defined formally in the next chapter. Substituting for the H matrix, we get

$$M(1) = M(0) + H_0M(0) + Nf_0 .$$

For any time instant k we can write

$$M(k+1) = M(k) + H_kM(k) + Nf_k , \tag{III.41}$$

where H_k is written as

$$H_k = \left(-\text{Diag}(Af_k) + [\text{One}(A)\text{Diag}(f_k)B]^T \right) \quad (\text{III.42})$$

or

$$H_k = \left(-\text{Diag}(AF_k) + [\text{One}(A)F_kB]^T \right). \quad (\text{III.43})$$

III.8. Comparison of the PN and the GPN Modeling

In this section we elaborate some of the structural and behavioral differences between the PN and the GPN modeling. Table III.1 summarizes the major modeling and structural differences between the PN and the GPN. It shows what their elements are and what they represent.

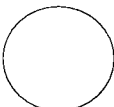
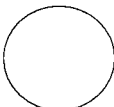



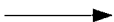


PN/GPN Element	GPN Representation	Graphical Representation (GPN)	PN Representation	Graphical Representation (PN)
Place (P)	Condition		Condition	
Transition (T)	Event, Change of State		Event	
Arcs (W's, A, B)	Relation		Relation	
Token, Marking (M)	Condition Validity, State Variable		Condition Validity	

Table III.1 Structural Differences between the PN and the GPN.

To demonstrate the GPN and the PN behavioral differences, we start by looking at the dynamics of each net.

Definition 3.8. GPN Dynamic Equation: The GPN dynamics, according to equation (III.41), can be defined by

$$M(k+1) = M(k) + H_k M(k) + N.f_k . \quad (\text{III.44})$$

If Both A and B matrices are zero, that is if there are no synchronous arcs, then

$$H_k = \left(-D(0.f_0) + [O(0).D(f_0).0]^T \right) = 0 . \quad (\text{III.45})$$

If Equation III.45 is substituted in Equation III.44, the PN dynamic equation can be found.

Definition 3.9. PN Dynamic Equation: is given as

$$M(k+1) = M(k) + N.f_k . \quad (\text{III.46})$$

As can be seen from equations (III.46) and (III.44), PN and GPN have different dynamics. In a PN, a new marking is an addition of the old marking and the token movement; in other words, it is an additive type of net. On the other hand for a GPN, a new marking in addition, has a term which is a multiplication of the previous marking. This part can be called the multiplicative one. A GPN, therefore, is a more generalized version of a conventional PN.

A Petri net is suitable for modeling discrete-event systems, whereas a GPN is geared towards modeling both discrete-time and discrete-event systems. GPN modeling gives us a tool which can be used to model and analyze a complete real-time system.

Another difference in the operation of the two nets is the way a change in the firing sequence affects the marking. Let us write the marking changes for a PN due to two transition firings f_0 and f_1 :

$$M(1) = M(0) + N.f_0 ,$$

$$M(2) = M(1) + N.f_1 .$$

Substituting for $M(1)$ in $M(2)$, we get

$$M(2) = M(0) + N(f_0 + f_1) .$$

If we change the order of the firings (first f_1 and then f_0), we get

$$M(1) = M(0) + N.f_1 ,$$

$$M(2) = M(1) + N.f_0 .$$

Then the final state is

$$M(2) = M(0) + N(f_1 + f_0) .$$

In the above case, irrespective of which transition firing (f_0 or f_1) we go through first, the final state $M(2)$ will be the same (assuming that the order of transition firing does not change the enabling of transitions). The reason for this equality is that

$$N(f_0 + f_1) = N(f_1 + f_0) .$$

Now we look at similar marking changes for a GPN:

$$M(1) = M(0) + H_0 M(0) + N.f_0$$

$$M(2) = M(1) + H_1 M(1) + N.f_1 .$$

Substituting for $M(1)$ in the equation for $M(2)$, we get

$$M(2) = M(0) + H_0 M(0) + H_1 M(1) + N(f_1 + f_0) . \quad (\text{III.53})$$

If we change the order of transition firing, we get a different marking,

$$M(2) = M(0) + H_1 M(0) + H_0 M(1) + N(f_0 + f_1) , \quad (\text{III.54})$$

which is not the same as the one in the previous firing order.

In a Petri net, it does not matter in what order transitions are fired unless they change the set of enabled transitions. In the above example, if we had fired Transition two before Transition one, we would still get the same result. In general, future states of a PN depend on its structure, initial condition, and the number of times each transition fires. For a GPN, however, the future states not only depend on the initial condition and structure of the net, but also on the order that transitions are fired.

III.9. Place and Transition Types in GPN

Any place in a GPN is either of real or integer type, depending on the type of arcs connected to it. Any transition, however, can be either of synchronous or asynchronous type. Hybrid places and transitions are also defined to be of one of these types. The type of a transition or a place depends on the type of arcs which are connected to them. These types easily can be determined by examining arc weight matrices A , B , W_{pt} , and W_{tp} .

III.9.1. Transition Types: Any transition in a GPN is of either synchronous or asynchronous type, depending on its input arcs.

Definition 3.10. Synchronous Transition: A transition is synchronous if it has no input asynchronous arc; that is, transition t_j is synchronous if all members of column j of weight matrix W_{pt} are zero. This type of transition has no input pre-condition. It fires as soon as its previous firing is completed and remains busy for a period equal to the transition time associated with it.

Definition 3.11. Asynchronous Transition: A transition is asynchronous if it has at least one asynchronous input arc; that is, transition t_j is asynchronous if at least one member of column j of weight matrix W_{pt} is non-zero. An asynchronous transition fires only when its input conditions are satisfied. It remains busy for a period equal to the transition time associated with it.

Figure III.6 shows all the possible configurations which a transition can have, and the resulting types. In row (a), all transitions are of the synchronous type since there are no asynchronous input arcs. The types of output arcs have no effect on the types of transitions as can be seen in the Figure III.6. Rows (b) and (c) show only asynchronous transitions; since all the transitions have at least one input asynchronous arc. Hybrid transitions such as those in row (c), also can be classified according to this rule. Items (b.1) and (b.3) are labelled as 'Not Defined', since it is not possible to have synchronous output arcs without having synchronous input arcs. Row (d) shows transitions with no input arcs. Items (d.1) and (d.3) are labelled as 'Not Defined' for the same reason that was stated for (b.1) and (b.3). However, (d.4) is also 'Not Defined' since it is an isolated transition. Finally, (d.2) is a synchronous transition since it has no input condition and fires in regular intervals determined by its transition time.

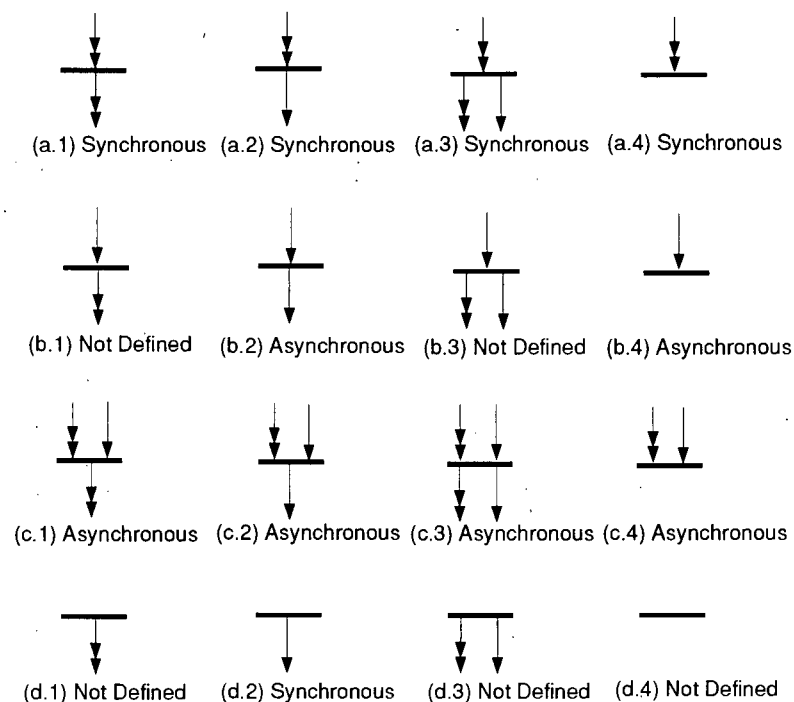


Figure III.6 Various Transition Types in the GPN.

III.9.2. Place Types: Places in GPNs are of either the real or the integer types. Their types also depend on the type of arcs which are connected to them.

Definition 3.12. Real Place: A place is real if it is connected to at least one synchronous input or output transition; that is, place p_i is real if at least one member of row i of weight matrix A or i column of weight matrix B is non-zero. A real place gets updated every time one of its input or output synchronous transitions fires. The marking of a real place can have any real value between minus infinity and plus infinity, unless a bound is specified.

Definition 3.13. Integer Place: A place is of the integer type if there is no synchronous transition connected to it; that is, place p_i is integer if all members of row i of weight matrix A or i column of weight matrix B are zeros. The marking of an integer place can take only positive integer values.

Figure III.7 shows all possible configurations of a place and its resulting type. Both input and output places, for each case, are drawn to show how they are affected by the type of transitions they are connected to. Places P_2 in Figure III.7(a.4) and P_2 in (b.4) are 'Not Defined', since they are isolated places which are not permitted either by PN or GPN definitions. The rest of the classifications are quite easy to figure out and are done according to the rules defined above.

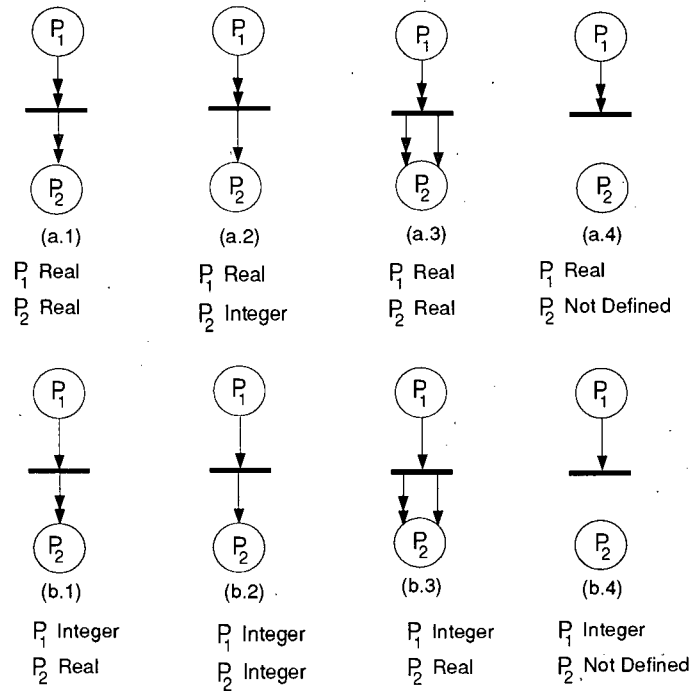


Figure III.7 Various Place Types in the GPN.

III.10. Advantages of Modeling with the GPN

In this section and the next we explore the GPN modeling capabilities and advantages through a series of examples. These examples are simple enough for illustration purposes but demonstrate the feature being discussed. The modeling of a simple flow control system is illustrated by the example in this section. The second example, in the next section shows how a six-transistor XOR gate can be modeled by a GPN. This section starts with modeling of nMOS and pMOS transistors. The modeling and simulation of XOR gates are performed by adding up these transistor models to form a larger GPN.

III.10.1. Hybrid Modeling; Water Tank Flow Control Example: The following example often is used to show a simple hybrid system [30, 81]. Figure III.8(a) shows a simple water tank with a closed loop flow control. The height of the water in the tank at any instant is represented by h . The water level is increased due to the flow into the tank, until the desired height H is reached. At this point water raises the float sufficiently to block the flow.

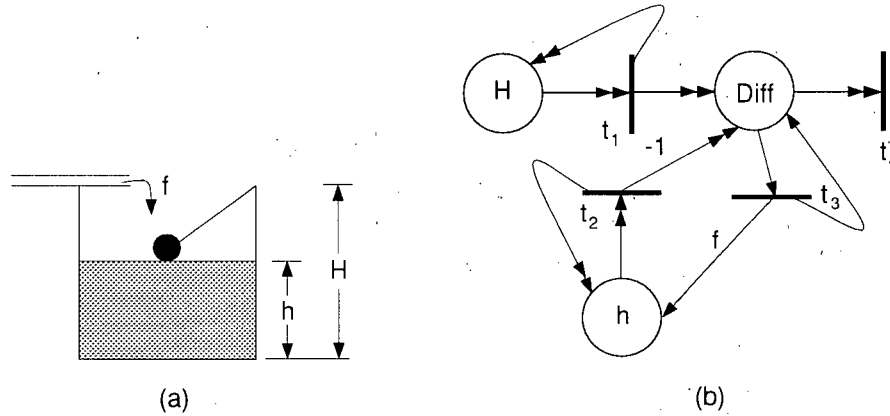


Figure III.8 (a) A Simple Flow control System. (b) The GPN Model of (a).

This system can be represented by the following relation:

$$h(k+1) = \begin{cases} h(k) + f & \text{if } h(k) < H \\ h(k) & \text{else} \end{cases}, \quad (\text{III.55})$$

where f is the height increase due to the constant flow into the tank when the valve is open. While h is smaller than H , its value gets increased by f . This system can be modeled by a GPN as shown in Figure III.8(b). This net has three places and four transitions. The net parameters can be written as:

$$\begin{aligned} P &= \{H, Diff, h\} & T &= \{t_1, t_2, t_3, t_4\} \\ W_{pt} &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} & W_{tp} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & f \\ 0 & 0 & 0 \end{bmatrix} \\ A &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} & B &= \begin{bmatrix} 1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \end{aligned} \quad (\text{III.56})$$

All transitions except t_3 are synchronous and fire at all instants. Place $Diff$ represents the difference between H and h . Transition t_3 fires when place $Diff = H - h$ marking is greater than zero, or $h < H$. Depending on this condition the dynamics of the system changes.

When $h < H$ or $Diff > 0$, all transitions fire and we get the following system equations:

$$H(k+1) = H(k) + H(k) - H(k) = H(k)$$

$$Diff(k+1) = Diff(k) - Diff(k) + H(k) - h(k) + 1 - 1 = H(k) - h(k) \quad (III.57)$$

$$h(k+1) = h(k) + h(k) - h(k) + f = h(k) + f.$$

When h exceeds H , the Diff place marking becomes negative, and transition t_3 is no longer enabled; therefore, only transitions t_1 , t_2 , and t_4 fire. The resulting dynamics equation then becomes:

$$H(k+1) = H(k) + H(k) - H(k) = H(k)$$

$$Diff(k+1) = Diff(k) - Diff(k) + H(k) - h(k) = H(k) - h(k) \quad (III.58)$$

$$h(k+1) = h(k) + h(k) - h(k) = h(k).$$

Figure III.9 shows how the place markings change. The desired input 'H', the error signal 'Diff', and the height of the water 'h' are plotted by '*', '—', and '+' signs, respectively. The desired input H is set at 20. The water is added at two units per sampling time ($f=2$) until the error signal becomes zero and stops the flow.

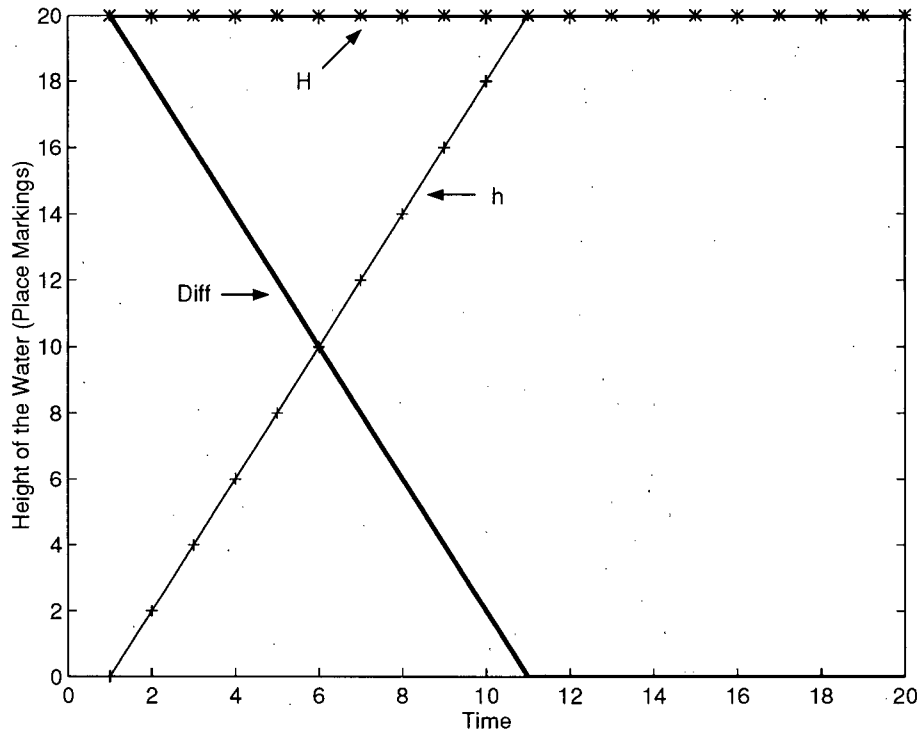


Figure III.9 Place Markings Versus Time Plots for the Flow Control Example.

III.11. GPN Modeling of a Six-Transistor XOR Gate

In this section we will show how a six-transistor XOR gate can be modeled by a GPN. Modeling of various logic gates by GPNs are described in Appendix A. Here, we will first show the modeling and simulation of a single MOS transistor. We will then proceed to model and simulate an XOR gate composed of six of these transistors. The results of the simulation are compared with those obtained by conventional VLSI analog simulation by HSPICE program.

11.1. MOS Transistor: The Metal Oxide Semiconductor (MOS) transistor has four terminals, called gate, drain, source, and substrate (body) [82]. The gate controls the flow of charge between the source and the drain. The fourth terminal, the body, cannot be used for performing useful logic and is not considered here. There are two types of MOS transistors: nMOS and pMOS. Figure III.10 shows the symbols for these types.

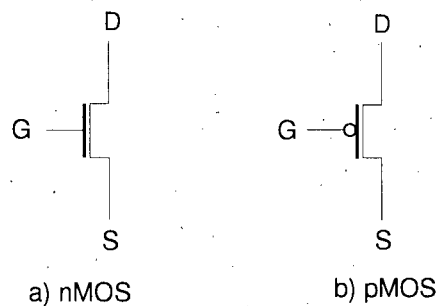


Figure III.10 nMOS and pMOS Transistor Symbols.

Hybrid Model of MOS Transistors:

The hybrid model of an nMOS transistor is shown in Figure III.11. There are five places in this net, representing the voltages at the three terminals and two voltage differences. The voltages at gate (V_G) and source (V_S) act as the input places, and the drain voltage (V_D) acts as the output place. In addition to these, the two internal places determine how the differences in the voltage levels of the terminal voltages control the output voltage.

All transitions in this net, except transition t_3 , are synchronous and fire in all sample periods no matter what their input place markings are. Transition t_3 is asynchronous and is controlled by the gate-to-source voltage, V_{gs} , which is defined as

$$V_{gs}(k+1) = V_G(k) - V_S(k) + 1.$$

Transition t_3 is enabled when V_{gs} is greater than 1, that is when V_G is larger than V_S . The drain-to-source voltage is computed as

$$V_{ds}(k+1) = V_S(k) - V_D(k).$$

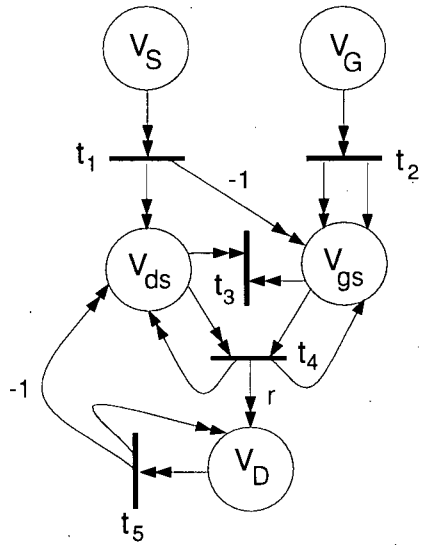


Figure III.11 Hybrid GPN Model of an nMOS Transistor.

Figure III.12 shows the simulation results for a set of inputs. The source voltage V_S (place p_2 marking) was assumed to be at five volts throughout the simulation. The markings of the remaining four places are shown as plots III.12(a)-(d). Gate voltage V_G is at zero volt initially, which keeps the transistor off. V_G is then changed to seven volts, which causes the transistor to conduct.

Plot III.12(d) shows the drain voltage (V_D). We have marked three regions in this plot, which mark the three stages that the transistor model goes through. The first region is called

the “non-conducting” region, which corresponds to the case when the gate voltage is not large enough to turn the transistor on. In this region, transition t_4 does not fire, and therefore the drain voltage (V_D) remains at zero volts.

The second region corresponds to the behavior which can be approximately described as Ohmic. In this region the drain voltage increases until it almost equals the source voltage. In this region, transition t_4 fires, and the drain voltage (V_D) starts increasing to the level set by the source voltage V_S . The rate of this increase can be controlled by the arc weight r , which connects transition t_4 to place V_D , according to the following equation:

$$V_D(k+1) = V_D(k) + V_D(k) - V_D(k) + rV_{ds}(k)$$

or

$$V_D(k+1) = V_D(k) + rV_{ds}(k) .$$

The last part is called the “saturation region”, and it corresponds to a situation where the voltages stabilize at their final values. This happens when the drain voltage nearly equals the source voltage. In this region transition t_4 is still enabled and firing, but it has no effect on the drain voltage. This is due to the reason that, the source-to-drain voltage difference, modeled by place V_{ds} , is nearly equal to zero.

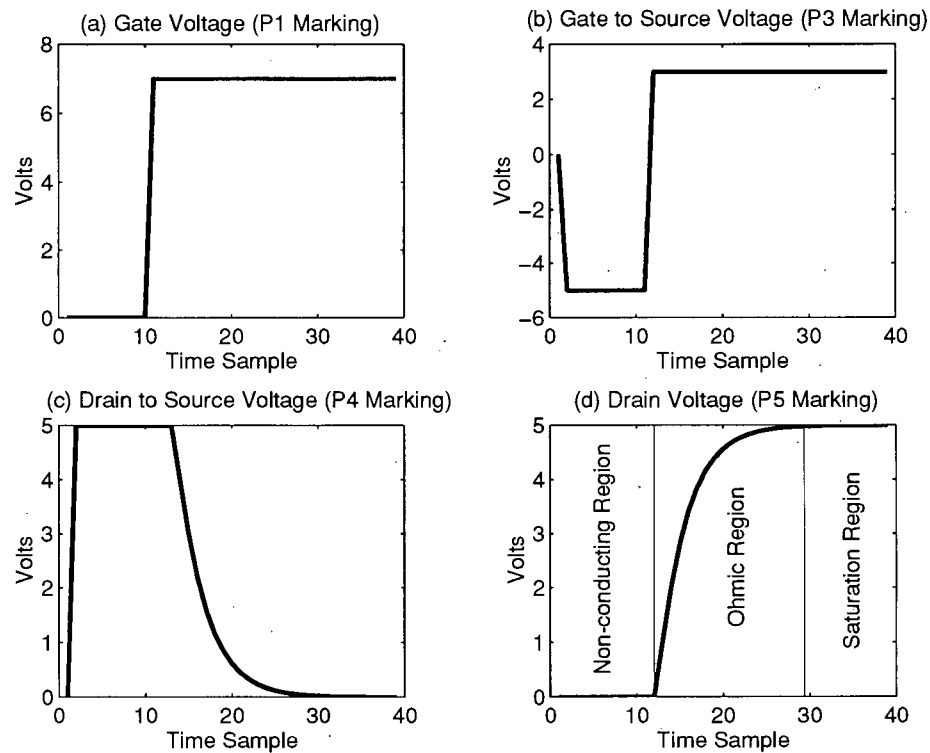


Figure III.12 Simulation Plots of Different Voltages in an Analog GPN Model of an nMOS Transistor.

III.11.2. GPN Modeling with MOS Transistors: In this section we will use the GPN models for the MOS transistors, developed in the previous sections, to model and simulate an XOR gate. Figure III.13 shows a six-transistor XOR gate. The first two transistors on the left, P_1 and N_1 , form an inverter which inverts the B input. The other four transistors work together to provide the logical operation needed for the XOR gate.

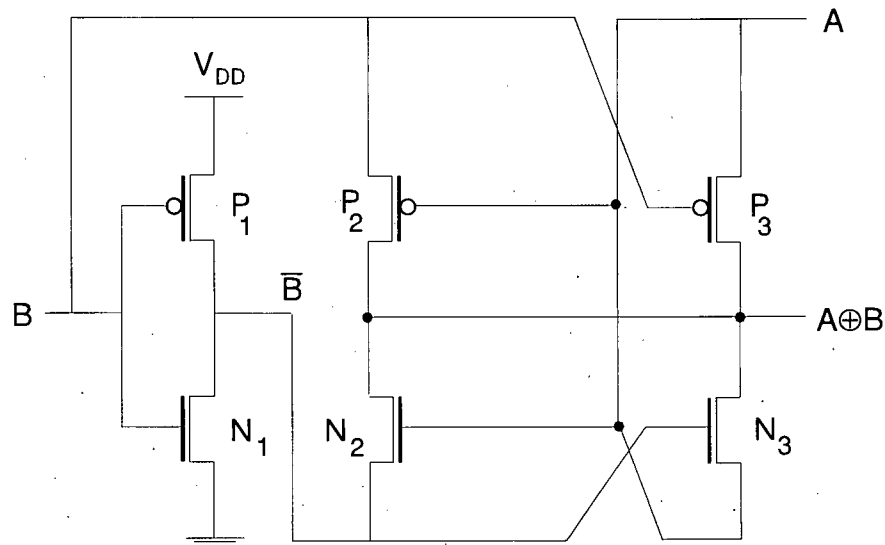


Figure III.13 The Six-Transistor XOR Gate.

A six-transistor XOR gate at the hybrid level can be built by connecting six hybrid MOS transistor models, which were developed in Section 11.1. This was done by replacing the transistors in Figure III.13 with their hybrid models, shown in Figure III.11. The modeling and simulation was carried out, and the results are summarized in Table III.2. The model was checked for a set of A and B inputs. The output entries show the final values of these signals before new inputs are introduced. The output of the inverter and the XOR gate are exactly as expected.

A	B	\bar{B}	$A \oplus B$
0	0	5	0
0	5	0	5
5	0	5	5
5	5	0	0

Table III.2 Simulation Results of the GPN Representing an XOR Gate.

Figure III.14 shows the plots for the input and output signals of the XOR model. Each plot shows how the signals change before settling at their steady state value. The input signals cover a range of all possible inputs.

The output of the XOR gate, shown in plot III.14(d) takes a few samples before reaching its final steady state value. The XOR output is formed by addition of the outputs of the four transistors on the right hand side of Figure III.13 (transistors P_2 , P_3 , N_2 , and N_3). After every change in the inputs A , and B , each of these transistors should produce its final output, before the XOR output can reach its final value.

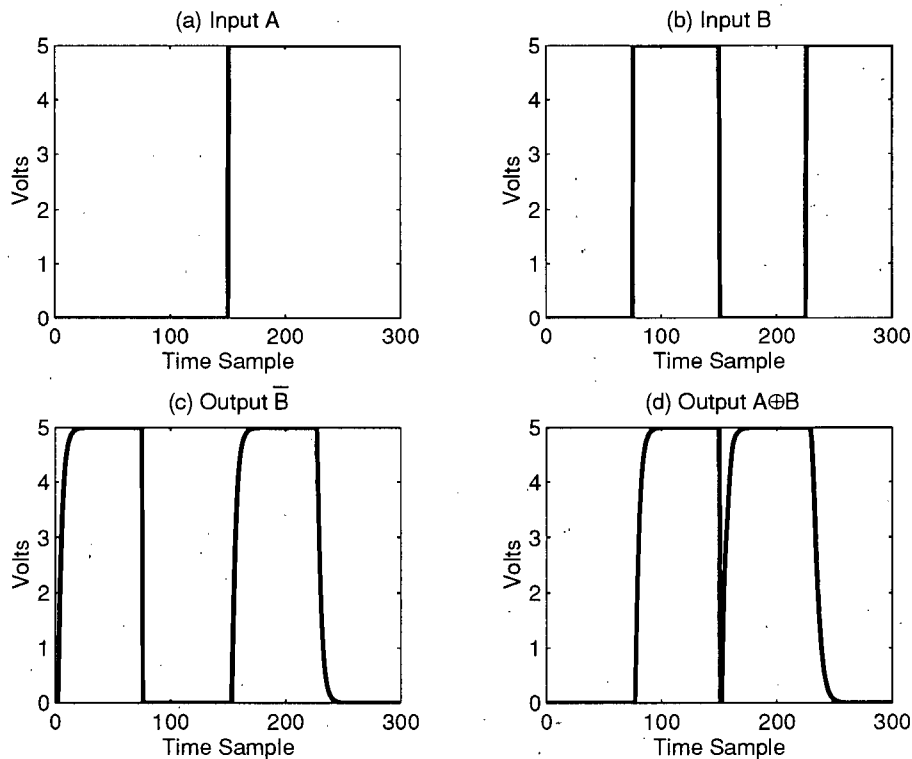


Figure III.14 Plots of the Input and Output Signals of the Hybrid Model of the XOR Gate.

Figure III.15 show the results of the simulation of the same gate by HSPICE program. The simulation is performed at analog level and using the same schematic circuit diagram as shown in Figure III.13. The inputs and outputs are shown as four separate plots and are as marked in Figure III.15.

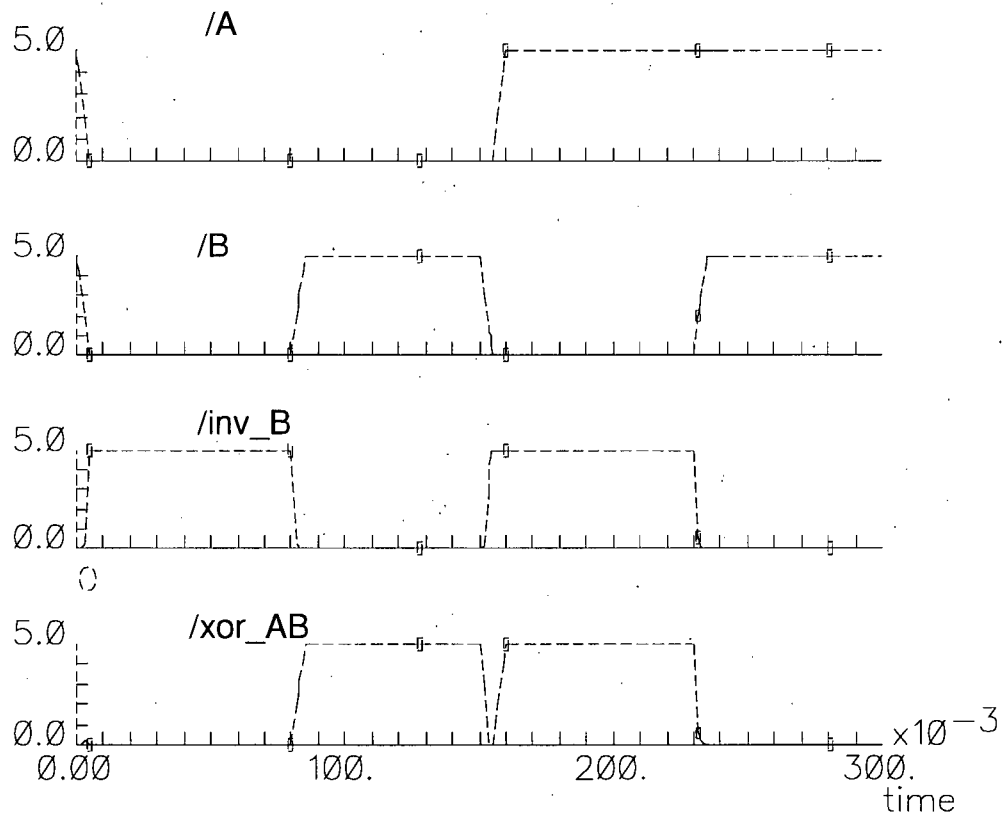


Figure III.15 HSPICE Simulation Results of the Six-transistor XOR Gate.

III.11.3. Simulation Conclusions In this section a modeling example by a GPN was presented. This example was chosen from an area (analog circuit simulation), quite different from the main focus of the thesis which is modeling and simulation of real time control systems. This example also illustrates how both digital and analog behavior can be modeled by GPNs. The simulation results and their comparison with a conventional simulation method (HSPICE) show the accuracy and correctness of modeling and simulation by GPNs.

Chapter IV GPN Properties and Analysis Methods

One of the main advantages of using a global Petri net for modeling systems is that the resulting model can be analyzed to find the system properties. The uniqueness of GPN analysis methods are in their ability to investigate the properties such as controllability and stability of hybrid systems. The controllability and stability, and other properties of discrete-event systems are analyzed by ordinary Petri net models. But, the extension of these analysis methods to hybrid systems is what sets GPN apart. Some of the original Petri net analysis techniques can be extended for GPN modeling. In this chapter, we investigate some of the important GPN properties and then develop the necessary tools to analyze them.

Modeling any actual system requires a large net with many places and transitions. This results in a large incidence and hybrid transition matrices which may be computationally difficult to check for GPN properties such as liveness and reachability. One way to handle such problems is by synthesis of GPNs. The basic idea is to start with a simple GPN which coarsely represents the system to be modeled, but is simple to analyze. Then more places and transitions are added to capture more modeling details. The additions are carried out in such a manner to preserve the properties which are proven for the more coarse models. In this thesis, we have not dealt with GPN synthesis explicitly, but the developments in Chapter VI are carried out in the top-down synthesis approach just described.

We start this chapter by discussing some of the modeling issues which are useful in later analysis. We continue with a discussion of the modelability of a given system by a GPN. We show what types of systems can be modeled with this net, and derive the condition for it. Then, hierarchy of GPNs is discussed. In the final two sections, analysis methods used for the GPN are introduced, and their application in finding system properties such as controllability, reachability, and stability is presented.

IV.1. GPN Modeling Issues

To put modeling by a GPN in a proper perspective, we first need to discuss the following modeling issues. Then, we will introduce a sub-class of GPNs which will ease the GPN analysis.

IV.1.1. Hybrid Transition Matrix: The next state of a GPN at any given instant k is determined by its dynamic equation, derived in the previous chapter (Equation III.41) as

$$M(k+1) = M(k) + H_k M(k) + N f_k, \quad (\text{IV.62})$$

where H_k is called the hybrid transition matrix, and is written (Equation III.42) as

$$H_k = \left(-\text{Diag}(A f_k) + [\text{One}(A) \text{Diag}(f_k) B]^T \right). \quad (\text{IV.63})$$

As can be seen from Equation (IV.63), corresponding to each transition sequence vector f_k we get a different H_k matrix. This is true only for transition sequence vectors which include a hybrid transition. To explain and prove this property, we first have to define a hybrid transition.

Definition 4.1 Hybrid Transition: A hybrid transition is one which has both asynchronous and synchronous type input arcs (at least one of each). Firing of a hybrid transition affects the H matrix. A hybrid transition is by definition an asynchronous transition, since its firing depends on its input place marking. Hybrid transitions do not fire in every sampling period. Of the transitions shown in Figure III.6, only four can be called hybrid transitions.



Figure IV.16 Hybrid Transitions.

Now let any given transition be classified either as a pure synchronous, a pure asynchronous, or a hybrid transition. Corresponding to each transition, is an element in the transition firing vector. If n is the total number of transitions, then

$$n = n_{ps} + n_{pa} + n_h, \quad (\text{IV.64})$$

where n_{ps} , n_{pa} and n_h , represent the number of pure synchronous, pure asynchronous, and hybrid transitions, respectively.

Theorem 4.1 : The number of different H matrices possible for a given GPN is equal to

$$\begin{cases} 0 & \text{if } n_{ps} = 0 \text{ and } n_h = 0 \\ 1 & \text{if } n_{ps} \geq 1 \text{ and } n_h = 0 \\ 2^{n_h} & \text{if } n_h \geq 1. \end{cases} \quad (\text{IV.65})$$

Proof : The proof is intuitive but can be explained as below:

1. When both n_{ps} and n_h are zero, it means we have only pure asynchronous transitions and our net reduces to a timed Petri net. In this case, we simply have no H matrix.
2. When there is at least one pure synchronous transition and no hybrid transition, all these pure synchronous transitions fire simultaneously and in every clock cycle, which gives us a single H matrix.
3. When there is one or more hybrid transitions, we get a different H matrix based on what combination of these transitions is fired. For example, for a net with two hybrid transitions; $n_h = 2$, there will be $2^{n_h} = 2^2 = 4$ different H matrices. These correspond to the cases where no hybrid transition fires, one of the two hybrid transitions fires and, when both fire simultaneously. In this case, the number of pure synchronous transitions does not matter since their firing coincides with one of the above 4 cases.

Having different H matrices and consequently different dynamic equations is one of the strong points of modeling by a GPN. This allows a single model to represent many different dynamics of a system which are switched by events (firing of transitions).

IV.1.2. A Sub-class of GPNs (When A is a Diagonal Matrix): An increase in the modeling power of a Petri net extension always is accompanied by a decrease in its analytical or decision power. Peterson in [31] states that "This has resulted in the development of sub-classes of PN with reasonable structural restrictions which will increase the decision power of the restricted net models while not overly restricting the modeling power". Some of these sub-classes mentioned in the literature are state machines, marked graphs, and simple PN [41].

One of the restrictions in the case of a GPN is with regard to the A matrix. A is the matrix of synchronous arc weights connecting places to transitions. If we consider the expression for the hybrid matrix

$$H_k = -\text{Diag}(A.F_k) + [\text{One}(A).F_k.B]^T, \quad (\text{IV.66})$$

when matrix A is a diagonal matrix, it means that there are equal number of transitions and places and there is just one synchronous arc connecting each place to a transition. In that case ($\text{One}(A)$) reduces to an identity matrix, and Equation (IV.66) reduces to

$$\begin{aligned} H_k &= -A.F_k + [F_k.B]^T, \\ H_k &= -A.F_k + B^T.F_k^T, \\ &\text{since } F_k \text{ is diagonal,} \\ H_k &= (B^T - A)F_k. \end{aligned} \quad (\text{IV.67})$$

As can be seen in Equation (IV.67), we can take the firing sequence matrix F_k out of the H matrix expression. This is a great help in the analysis of the GPN, as will be seen later.

This sub-class of GPNs will have the following restriction:

$$|P^\bullet| = |\bullet T| = 1. \quad (\text{IV.68})$$

That is, the set of output transitions of every place and the set of input places of every transition have exactly one member. Figure IV.17 shows a couple of cases which are not

allowed under this formulation. In Figure IV.17(a),

$$p_1^\bullet = \{t_1, t_2\} \text{ and therefore } |p_1^\bullet| = 2. \quad (\text{IV.69})$$

This is a violation of the restriction given in Equation (IV.68). In Figure IV.17(b),

$${}^\bullet t_1 = \{p_1, p_2\} \text{ and therefore } |{}^\bullet t_1| = 2. \quad (\text{IV.70})$$

This is another example of a violation of the restriction given in Equation (IV.68).

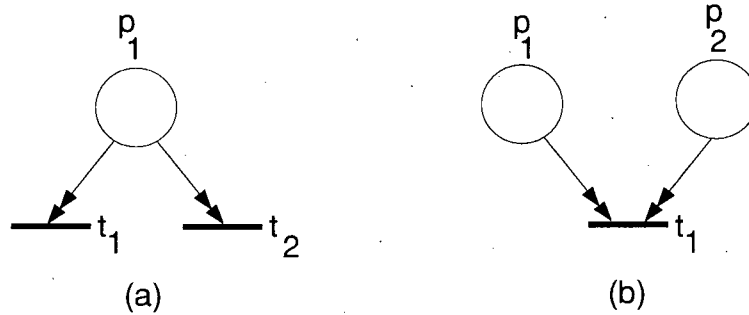


Figure IV.17 Two Examples of Nets not Allowed by Diagonal A Restriction.

The restriction, imposed to insure that the A matrix becomes diagonal, seems to curtail our ability to model any sort of conflicts. But we should point out that this restriction applies only to the synchronous arcs. For example in Figure IV.17(a), we can have an asynchronous arc instead of each of the synchronous arcs and not violate the restriction. In that case, we have modeled a conflict between t_1 and t_2 .

The above restriction does not affect our ability to model synchronization, since we can have

$$|{}^\bullet P| = |T^\bullet| = n. \quad (\text{IV.71})$$

Figure IV.18 shows two examples of nets which model synchronization and are allowed under the above restriction.

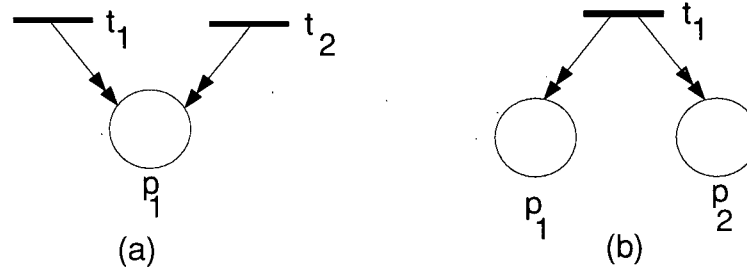


Figure IV.18 Two Examples of Nets Allowed by Diagonal A Matrix Restriction.

We have developed a procedure which converts GPNs with non-diagonal A matrices to GPNs which have diagonal A matrices. Appendix C illustrates this procedure through a general two-place, two-transition GPN. Originally, a GPN is considered which has a non-diagonal A matrix with all non-zero elements. Then, the GPN is transformed to one with a diagonal A matrix. The transformed GPN parameters (A , B , W_{pt} , W_{tp}) are obtained in terms of the first GPN parameters.

The resulting nets in all cases are larger than the original ones since either the number of places or transitions are increased to make these numbers equal. But, the reduction in analysis complexities very well justifies the conversion especially for analyzing system properties such as controllability and conservation which are analyzed by linear algebraic methods. This should become evident in later parts of this chapter when the analysis methods for GPNs with both diagonal and non-diagonal A matrices are described.

IV.2. Condition for GPN Modelability

Modelability of any given system by a GPN is defined as the condition which allows the system to be represented by the GPN parameters (a set of places, a set of transitions, appropriate weight matrices and a marking vector). As it was shown in Chapter III and Appendix A, many different types of systems can be modeled by GPNs. These include logic gates and functions, certain types of non-linearities (such as saturation and deadbands), analog and digital electronic devices, and dynamic control systems. The only limitation is to be able to specify the system behavior in terms of a set of states and a set of equations relating

those states. These equations are formed into a set of state-space matrices. These relations (matrices) can be time-variant even though only time-invariants ones are considered in this thesis.

Any such systems is modeled by one of the following two methods. First, for the synchronous part of the system, if the system dynamics are given in the form of a set of state space matrices, we can write the equivalent H matrix and subsequently set up the net by determining its A and B matrices. Then we have to find out how events affect these states and accordingly set up W_{pt} and W_{tp} matrices. Alternately, if the system is described by a set of relations among its parameters in terms of parallel operations or precedences, we can write the net arc weight matrices and then find the H and N matrices.

Here, we show the condition for finding the synchronous part of the H matrix by the first method, through an example. The second method is straightforward since we already have the required matrices.

Dynamics of GPN is described by

$$M(k+1) = M(k) + H_k M(k) + N f_k. \quad (\text{IV.72})$$

Since we are concerned only with the synchronous part of the system, we assume the N matrix is null. All time-invariant linear systems that can be represented by state space can also be represented by the above equation. H matrix is defined as

$$H_k = \left(-\text{Diag}(A.F_k) + [\text{One}(A).F_k.B]^T \right). \quad (\text{IV.73})$$

To construct the net we need to find the number of places and transitions and establish matrices A and B. The number of places is fixed by the dimension of the H matrix. In the simplest form, if we model the system such that it has an equal number of places and transitions, and let A and F be our identity matrices, then:

$$\text{Diag}(A.F) = \text{Diag}(I.I) = I,$$

and

$$[One(A).F.B]^T = [One(I).I.B]^T = B^T . \quad (IV.74)$$

If we substitute from Equation (IV.74) in Equation (IV.73), we will have

$$H = B^T - I . \quad (IV.75)$$

Equation (IV.75) above, gives the sufficient condition for any given H to have a representation by a GPN. In many cases we can find a smaller net, that is a net with fewer transitions, which can represent the H matrix. The following example illustrates this point.

IV.2.1. An Example: Let a given system and its corresponding H matrix be

$$\begin{bmatrix} m_1(k+1) \\ m_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} m_1(k) \\ m_2(k) \end{bmatrix} ,$$

or

$$H = \begin{bmatrix} 0 & 0 \\ 3 & 0 \end{bmatrix} . \quad (IV.76)$$

Now as shown above, if we take A=I and F=I, then

$$B^T = H + I = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix} ,$$

$$B = \begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix} .$$

This results in a net with two places and two transitions. This net also can be represented by a smaller net with only 2 places and one transition. In that case we have

$$A = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad B = [1 \quad 3] .$$

The H matrix for the second net is exactly the same as the first one.

IV.2.2. Modelability Condition for the Synchronous Part of GPNs: In this section we would like to show the general condition by which we can find a set of A and B matrices to model a given equation, where A is not necessarily an identity matrix. Assuming that every transition in a net fires exactly once (there are no hybrid transitions), we can write the state equations as

$$M(k+1) = M(k) + HM(k)$$

where

$$H = (-\Theta + \Phi) \quad (IV.79)$$

$$\Theta = \text{Diag}(A) \quad \text{and} \quad \Phi = [\text{One}(A).B]^T$$

Every H matrix member can be calculated as

$$\begin{aligned} H_{i,j} &= -\theta_{i,j} + \phi_{i,j} \\ H_{i,j} &= -\sum_{N=1}^n (A_{i,N}) + \sum_{N=1}^n O(A_{j,N}).B_{N,i} \quad \text{for } i = j \\ H_{i,j} &= \phi_{i,j} \\ H_{i,j} &= \sum_{N=1}^n O(A_{j,N}).B_{N,i} \quad \text{for } i \neq j, \end{aligned} \quad (IV.80)$$

where n is the number of equations and N is an arbitrary variable. According to the above we have p^2 equations corresponding to H matrix members, and $(p \times t)$ unknowns corresponding to A matrix and $(p \times t)$ corresponding to B matrix. Therefore, as long as the number of transitions is smaller than half the number of places, we have more equations than unknowns and can be assured of finding appropriate A and B matrices. This is only the sufficient condition for modelability of a given matrix. In most cases finding A and B matrices is much easier than solving these equations.

IV.3. GPN Hierarchy

In modeling any system with a high degree of complexity, we need to model it at various levels of abstraction since it reduces the analysis burden. A GPN can model different systems at various degrees of abstraction.

A GPN can be used for two purposes. In its general form it can model a dynamic system, with changes in its dynamics represented by a switching matrix which is event-driven. This switching takes place by the firing of enabled transitions, and it is modeled by firing sequence matrix F . In the special case when F is an identity matrix, the dynamics of the net are fixed. This special case can be used to model a dynamic plant in parts of or in its entirety.

The dynamics of a GPN are such that every place is a function of one or more of the other places including itself. In building a model, the number of places is in direct proportion to the number of parameters which should be monitored in the system. When a place is not an input place, it can be written as a function of other places. That way we can reduce the number of places.

Any general system can be represented by a net with an equal number of transitions and places. That is, for modeling any system we need a maximum of l places and l transitions. Now, if we eliminate a place by writing it as a function of other places, then we get fewer places as well as fewer transitions, and consequently a smaller net. The advantage of modeling with the GPN is that we can do all the hierarchical reduction from the H matrix and then translate the final result into a GPN model.

IV.4. Analysis Methods

The two main approaches to the analysis of a Petri net are reachability tree and linear algebraic methods. The first method is based on constructing the set of all reachable markings of the net. This analysis method is only useful for small nets since the reachability tree size explodes rapidly with an increase in the state space size. The linear algebraic method does not involve this problem but may be restricted to only a sub-class of nets [83, 84]. These two methods are described in the following sections. It is shown how they can be applied to analyze a given GPN.

IV.4.1. Reachability Tree: The First analysis method is based on constructing a complete reachability tree (RT) or a subset of all reachable states. The tree found by this method

depends on the initial marking of the net, therefore, this method is considered a behavioral analysis technique rather than a structural one. By looking at an RT we can discover many of the modeled system properties, such as boundedness, conservativeness, and liveness. These properties will be defined formally in the next section. Use of the RT method for system analysis is explored in the chapter on systems analysis.

The main advantage of RT analysis is that it can be used for any system, unlike the linear algebraic method, which might be restricted to a sub-class of nets. It should be noted that a combination of these two methods is necessary for a complete analysis. The main problem with the RT method is that the size of an RT can become very large for any non-trivial system.

The construction of an RT starts with assignment of the initial marking as the root node of the tree and then finds the other nodes by firing each enabled transition. Arcs represent the transition firings and show how a node is reached from other nodes. Each new state is either a middle node or a terminal node. A terminal node can be either an "OLD" or a "DEAD END" type. A tree is complete when all the branches end with a terminal node. Next, we present an algorithm for the construction of the GPN reachability tree (GPNRT) and then follow it with an illustrative example.

Algorithm for Construction of a GPNRT An algorithm for the construction of GPN reachability trees (GPNRT) has been developed and is given in Figure IV.19. This algorithm is based on a standard algorithm which is used for constructing Petri net reachability trees (PNRT), as in [85, 31, 49].

Figure IV.19 An Algorithm for Constructing GPN and PN Reachability Trees.

1. Start with the initial marking $M(0)$ as the root node and label it "NEW".
2. While there is a "NEW" node left, do the following:
 - 2.a. Select a "NEW" node and call it marking M .
 - 2.b. If M is identical to a marking already processed, then tag M "OLD" and go back to step 2.a. **For real places, if the marking is close enough to a marking already processed, it is marked "OLD".**
 - 2.c. Check if any transition is enabled under marking M ; if not, label it "DEAD END" and go back to step 2.a. (**Note: Synchronous transitions are by definition always enabled.**)
 - 2.d. While there are enabled transitions at M , do the following for each enabled **asynchronous** transition t_1, t_2, \dots, t_n **and all of the synchronous transitions** at M :
 - 2.d.i. Obtain the marking M' that results from firing **asynchronous** transition t_1, t_2, \dots, t_n **and all of the synchronous transitions** at M .
 - 2.d.ii. On the path from the root to M , if there exists a marking M'' such that $M'(p) \geq M''(p)$ for each **Integer** place and M' is not equal to M'' , then replace $M'(p)$ by ω for each **real** place such that $M'(p) > M''(p)$.
 - 2.d.iii. On the path from the root to M , if there exists a marking M'' such that $M'(p) > M''(p)$ for some real places while the sign (+/-) of all other place markings in M' and M'' remain the same, then replace $M'(p)$ by ω for each real place such that $M'(p) > M''(p)$.
 - 2.d.iv. On the path from the root to M if there exists a marking M'' such that $M'(p) < M''(p)$ for some real places while the sign (+/-) of all other place markings in M' and M'' remain the same, then replace $M'(p)$ by $-\omega$ for each synchronous place such that $M'(p) < M''(p)$.
 - 2.d.v. Introduce M' as a node and draw an arc with label t from M to M' and tag M' "NEW".
 - 2.d.vi. END
 - 2.e. END
3. END

The modifications to the basic algorithm are presented in boldface. Description of this algorithm is given by highlighting the main differences between this algorithm and the one for the PNRT.

1. We have two types of places in a GPN. Integer places have markings which only take positive integer values and real places which can have real number markings. In the latter case we can have an infinite number of possible states which may differ from each other only by the slightest margin. To limit the number of states that a GPNRT can have, we have introduced a closeness criterion for deciding if a marking is an "OLD" node or not. By this measure, if a marking is within a threshold from an "OLD" node, it is taken to be the same one. This is implemented as step 2.b. in Figure IV.19. The closeness measure is to be decided by the system designer since it may vary for different systems and depends on the magnitude of the variable that any particular marking represents. A threshold which is too small will increase the size of the RT, whereas a large one may result in missing some important dynamics in the way markings evolve. For markings modeled in this thesis, the threshold was taken to be anywhere from 0.1 to 0.01.
2. There are also two types (synchronous and asynchronous) of transitions in a GPN. Synchronous transitions always are enabled, and fire according to their transition time. In this GPNRT implementation, we have assumed that the transition time for all transitions is the same and is equal to one sampling period. Therefore, all synchronous transitions fire simultaneously and at all sampling periods. Asynchronous transitions are enabled once their input conditions are satisfied. Starting from the root node, we fire one of the asynchronous transitions along with all the synchronous ones to get to the next state. When there are no enabled asynchronous transitions, only synchronous ones are fired. Further, when there are no synchronous ones in the system, that node is marked "DEAD END". This step is implemented as step 2.c. in Figure IV.19.
3. An RT can become infinite if the markings are unbounded. To keep the tree finite, a special character ω is used which has the following properties. For any constant C,

$\omega \pm C = \omega$, $\omega > C$, $\omega \geq \omega$. A place marking in a PN becomes an ω when its value is increased by a transition firing, while all others remain the same or increase due to the same firing. The reasoning behind this is that the firing of such a transition can repeat an infinite number of times and increase the marking of the output place to infinity. This is possible since none of the markings is decreased which would disable this transition. This step is implemented as step 2.d.ii. Markings of a GPN, unlike those of a PN, can take negative values. Therefore, a place marking can go out of bound both in the negative and positive directions. In this algorithm, we mark the possibility of crossing a negative bound by $-\omega$. An ω or $-\omega$ appear in a real place marking, according to the rules shown in steps 2.d.iii. and 2.d.iv.

GPN Reachability Tree Examples: Figure IV.20(a) shows a GPN with four places and two transitions. Both transitions are asynchronous since their firing depends on the availability of tokens in place p_1 . Figure IV.20(b) shows another net which is almost the same as the one in Figure IV.20(a), with the exception of an extra synchronous transition t_3 . We will use these two nets to illustrate how their RTs differ due to this extra transition. The reachability trees for these two nets (a) and (b) are shown as Figures IV.21 and IV.22, respectively. These RTs are found by the algorithm described earlier.

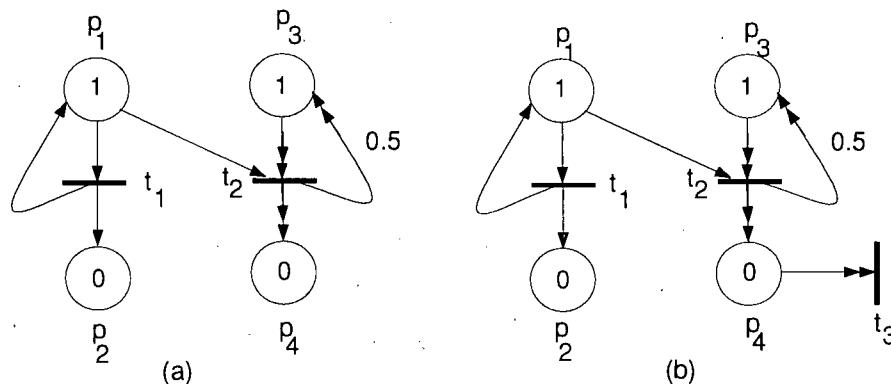


Figure IV.20 Examples of a GPN used for Construction of GPNRTs. (a) A GPN with only Asynchronous Transitions. (b) A GPN with an Extra Synchronous Transition.

We start by describing the tree found for net which is shown in Figure IV.21(a). The initial marking of the net is $M(1) = [1 \ 0 \ 1 \ 0]^T$, which is shown as the main (root) node of the tree. This node is written as $(1 \ 0 \ 1 \ 0)$. Under this marking condition, both t_1 and t_2 are enabled. Firing of transition t_1 results in node $(1 \ 1 \ 1 \ 0)$. This firing increases the second place marking while keeping all others the same. According to standard PNRT, with this condition we should get an infinity character, ω , in the place p_2 marking. But since p_2 is a synchronous place we should apply the GPNRT rule instead. By this rule, we do not get an ω since we have a change in the sign of the other place markings (a change from "0" to positive "1").

Firing transition t_2 from the root results in marking $(0 \ 0 \ 0.5 \ 1)$. This node is a "DEAD END" since no transition can be fired from this marking as is shown in Figure IV.21.

Starting from node $(1 \ 1 \ 1 \ 0)$, we can fire both transitions. The firing t_2 results in another "DEAD END" node in $(0 \ 0 \ 0.5 \ 1)$. Firing of transition t_1 gives us node $(1 \ \omega \ 1 \ 0)$. The infinity term ω , in place p_2 , indicates that this place is unbounded and can increase infinitely under the same firing condition.

Firing transition t_1 from node $(1 \ \omega \ 1 \ 0)$ does not produce a new node, and therefore this node is marked "OLD". Transition t_2 gets us another "DEAD END" node. This completes the GPNRT for this net.

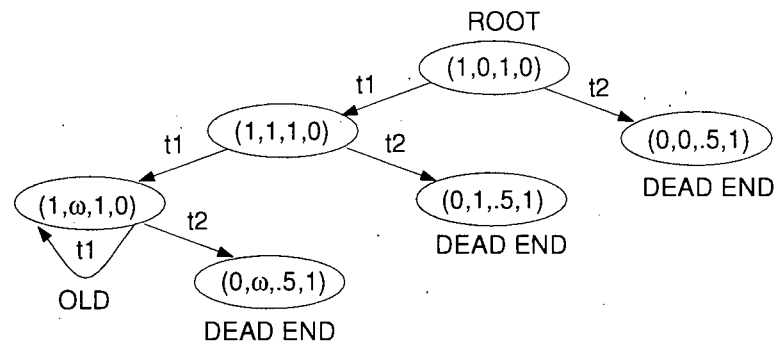


Figure IV.21 Reachability Tree of the GPN Given in Figure IV.20(a).

Next we construct the GPNRT for the net given in Figure IV.20(b). The only difference between this net and the previous one (Figure IV.20(a)) is an extra synchronous transition t_3 . This transition has no firing condition and fires every sample period, along with any other asynchronous transition which might be enabled and firing. Therefore, as can be seen in Figure IV.22, all arcs have t_3 as one of the transitions being fired. This extra transition produces some extra nodes. An interesting observation is that there is no “DEAD END” node since transition t_3 fire can under any condition. All the extra nodes in this RT are produced due to this fact.

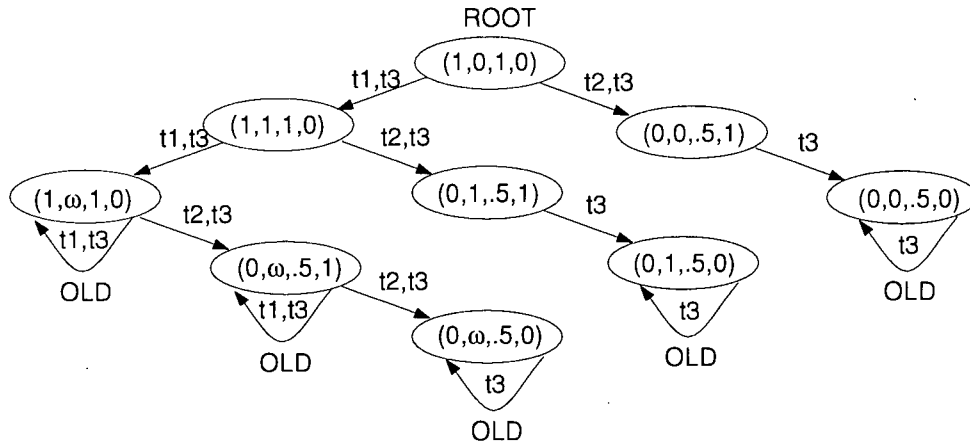


Figure IV.22 Reachability Tree of the GPN Given in Figure IV.20(b).

A program was written which computes all the nodes of a given GPN and produces a complete reachability tree. This program will be described in Chapter V.

IV.4.2. Linear Algebraic Method: The linear algebraic method is based on an analysis of the dynamic equations of a given net [86]. By checking whether these equations satisfy certain conditions, we can find out about the modeled system properties. Due to the difficulty of the analysis, these properties might be restricted to certain classes of nets as will be shown in the following sections. The dynamic equations for a PN are given as

$$M(k+1) = M(k) + Nf_k$$

$$\begin{aligned}
 M(k+2) &= M(k+1) + Nf_{k+1} \\
 &= M(k) + N[f_k + f_{k+1}]
 \end{aligned}$$

$$\vdots$$

$$\text{If } f = [f_k + \cdots + f_n],$$

$$M(k+n) = M(k) + Nf. \quad (\text{IV.81})$$

By analyzing equation (IV.81) we can prove certain properties for a given PN. These properties are given in the next section. The dynamic equations of GPNs which are used for the matrix equation analysis method are given below.

$$\begin{aligned}
 M(k+1) &= M(k) + H_k M(k) + Nf_k, \\
 M(k+2) &= M(k) + H_k M(k) + H_{k+1} M(k+1) + Nf_{k+1}, \\
 &\vdots \\
 M(k+n) &= M(k) + \sum_{j=1}^n H_{k+j-1} M(k+j-1) + Nf. \quad (\text{IV.82})
 \end{aligned}$$

Equation (IV.82) represents the dynamic equation, which can be used in this analysis method. If we consider the GPNs with a diagonal A matrix, we can substitute

$$H_k = (B^T - A)F_k$$

in Equation (IV.82) and get

$$\begin{aligned}
 M(k+1) &= M(k) + (B^T - A)F_k M(k) + Nf_k, \\
 M(k+2) &= M(k) + (B^T - A)F_k M(k) \\
 &\quad + (B^T - A)F_{k+1} M(k+1) + Nf_{k+1},
 \end{aligned}$$

$$M(k+n) = M(k) + \begin{pmatrix} B^T - A \end{pmatrix} \sum_{j=1}^n F_{k+j-1} M(k+j-1) + Nf. \quad (\text{IV.84})$$

The linear algebraic method does not depend upon the initial marking of the net and only takes into account the structure of the net. Many of the system properties can be proven by this method and are described in the following sections.

These two methods (linear algebraic and reachability tree), can be applied in the analysis of GPNs to investigate the modeled system properties. In the following section, we will examine each of these properties in detail and show how they can be investigated by any of these two methods.

IV.5. Controllability

The Controllability of a control system is defined as an answer to the question “Is it possible to steer a system from a given initial state to an arbitrary state in a finite time period?” [87]. A Petri net is said to be completely controllable if any marking is reachable from any other marking [85]. Controllability of GPNs is defined as follows: given an initial marking $M(k)$, is it possible to reach any desired marking $M(k+n)$, with the given H matrix, in a finite sample time n ?

IV.5.1. Controllability (When A is an Identity Matrix): As we have shown, to be able to take the firing matrix out of the H matrix, we need to transform the A into a diagonal matrix. When this transformation is made, we get time-invariant matrices, which are easier to analyze. In such a case, for a GPN we have

$$M(k+1) = M(k) + H_k M(k) + Nf_k. \quad (\text{IV.85})$$

When matrix A is diagonal, Equation (IV.85) can be written as

$$M(k+1) = M(k) + \begin{pmatrix} B^T - A \end{pmatrix} F_k M(k) + Nf_k \quad (\text{IV.86})$$

where H_k , F_k , and f_k are H and F matrices, and f vector at time instant k , respectively. Assuming initial marking $M(0)$, and substituting $k = 0, 1, 2, \dots$, we get

$$\begin{aligned} M(1) &= M(0) + (B^T - A)F_0M(0) + Nf_0, \\ M(2) &= M(0) + (B^T - A)F_0M(0) + Nf_0 + (B^T - A)F_1M(1) + Nf_1, \\ M(3) &= M(0) + (B^T - A)F_0M(0) + Nf_0 \\ &+ (B^T - A)F_1M(1) + Nf_1 + (B^T - A)F_2M(2) + Nf_2. \end{aligned} \quad (IV.87)$$

In the same manner, for $k = n$, we have

$$M(n) = M(0) + (B^T - A) \sum_{j=1}^n F_{j-1}M(j-1) + N \sum_{j=1}^n f_{j-1}.$$

If we take the initial marking $M(0)$ to the other side, we get

$$\begin{aligned} M(n) - M(0) &= \\ &= \left[(B^T - A) : \dots : (B^T - A) : N : \dots : N \right] \begin{bmatrix} F_{n-1}M(n-1) \\ \vdots \\ F_0M(0) \\ f_{n-1} \\ \vdots \\ f_0 \end{bmatrix}. \end{aligned} \quad (IV.89)$$

Then, the controllability condition is that the

$$\text{Rank} \left[(B^T - A) : \dots : (B^T - A) : N : \dots : N \right] = n. \quad (IV.90)$$

IV.5.2. Controllability of Time-variant Systems: When A is taken to be any given matrix (without restriction of being a diagonal matrix), then our H matrix becomes time-variant of the following general form:

$$M(k+1) = M(k) + H_kM(k) + Nf_k.$$

The controllability matrix of such a system is given as [88]

$$[P_0, P_1, \dots, P_{n-1}] ,$$

where

$$P_0 = [H_k \quad N] , \quad (\text{IV.92})$$

and

$$P_{l+1} = -P_l + P_l(k+1) .$$

IV.6. Conservation

When the total token count in a set of PN places remains constant, those places are called conservative. This property is useful in modeling resource allocation in a system. By showing that the number of tokens in a set of places remains constant, we are assured that no resource is created or destroyed. This property can also be used for systems with some conservation properties, for example, conservation of energy assuming no losses.

A GPN is said to be conservative if, starting from an initial marking $M(k)$ and for all reachable markings,

$$\sum_{p_i \in P} M_{p_i}(k) = \sum_{p_i \in P} M_{p_i}(k+1) . \quad (\text{IV.93})$$

Equation IV.93 is the condition for strict conservation, which rarely holds for systems. A more general form can be written as

$$M_{k+1}^T Y = M_k^T Y , \quad (\text{IV.94})$$

where $M_{k+1} = M(k+1)$ and $M_k = M(k)$ are the markings at instants $k+1$ and k , respectively. Y is an arbitrary m -dimensional vector. The weak conservation condition checks the weighted sum of the markings. We first derive the conservation condition for GPNs which have a diagonal A matrix, and later for any given GPN.

IV.6.1. Conservation of GPNs with Diagonal A Matrix: From the definition of the dynamics of a GPN with a diagonal A matrix, we have

$$M_{k+1} = M_k + (B^T - A)F_k M_k + Nf_k. \quad (\text{IV.95})$$

Transposing both sides and multiplying by a vector Y yields

$$M_{k+1}^T Y = M_k^T Y + M_k^T F_k (B^T - A)^T Y + f_k^T N^T Y.$$

Now the condition for conservativeness of a given net can be derived as

$$\begin{aligned} (B^T - A)^T Y &= 0, \\ \text{and } N^T Y &= 0, \end{aligned} \quad (\text{IV.97})$$

which gives us

$$M_{k+1}^T Y = M_k^T Y.$$

Vector Y is called a P-invariant, and its properties are discussed in the literature [49].

Theorem 4.1. Conservation Condition (When A is diagonal): A GPN with a diagonal A matrix is said to be (partially) conservative if there exists an arbitrary positive weighting integer $y(p)$ for every (some) place p , such that $M_{k+1}^T Y = M_k^T Y = \text{constant}$ and

$$(B^T - A)^T Y = 0 \quad \text{and} \quad N^T Y = 0. \quad (\text{IV.99})$$

IV.6.2. Conservation of any Given GPN: Any given GPN can be written as a series of nets with different H matrices. The conservation property should be ascertained for each of the H matrices. Let the dynamics of a net be given as

$$M_{k+1} = M_k + H_k M_k + Nf_k. \quad (\text{IV.100})$$

Now depending on the firing condition of the net we get different H matrices, which can be written as H_1, H_2, \dots, H_j . Then the condition for conservativeness of such a net will be

$$\begin{aligned} N^T Y &= 0 \quad \text{and} \\ H_1^T Y &= 0, \quad H_2^T Y = 0, \dots, \text{and} \quad H_j^T Y = 0. \end{aligned} \quad (\text{IV.101})$$

Theorem 4.2. Conservation Condition (Any GPN): A GPN with H matrices H_1, H_2, \dots, H_j is said to be (partially) conservative if there exists an arbitrary positive weighting integer $y(p)$ for every (some) place p , such that $M_{k+1}^T Y = M_k^T Y = \text{constant}$ and

$$N^T Y = 0 \text{ and} \quad (IV.102)$$

$$H_1^T Y = 0, \quad H_2^T Y = 0, \dots, \text{ and } H_j^T Y = 0.$$

The conservation property can also be analyzed by the R-tree method. For this we have to inspect the tree and see if the conservation condition holds for each node of the tree. Obviously a set of places with an infinity sign ω in one of them cannot be conservative.

IV.7. Boundedness and Stability

Boundedness of an event-driven system and stability of a time-driven system are very similar concepts. Boundedness examines whether one or more of the system states can grow beyond a limit or bound. Stability of a system checks whether for given bounded inputs to the system, its outputs would continue to remain bounded (Bounded Input Bounded Output, BIBO). These properties are the most important characteristics of any given system.

A place in a PN is called I-bounded if the marking of that place cannot exceed a positive integer number I . A PN is bounded if all its places are bounded. Boundedness of a GPN is defined in the same manner. A place in a GPN is called R-bounded if the marking of that place cannot exceed a real number R . A GPN is bounded if all its places are bounded. Since marking in a GPN can have both negative and positive values, we will have both negative and positive bounds.

We start this section by showing how the boundedness of a PN is determined by the linear algebraic method. We then show why this method is not sufficient to determine GPN boundedness and present our analysis method.

Theorem 4.2. PN Boundedness Condition: [85] A given PN is bounded if there exists a vector Y such that

$$\exists Y > 0 \text{ and } N^T Y \leq 0. \quad (IV.103)$$

Proof: [85] The equation for PN dynamics can be written as

$$M_{k+1} = M_k + N f_k . \quad (\text{IV.104})$$

If we take the transpose of each matrix and multiply both sides by Y , we get

$$M_{k+1}^T Y = M_k^T Y + f_k^T N^T Y . \quad (\text{IV.105})$$

Since M_k and f_k are both positive, then $f_k^T N^T Y$ becomes negative. Therefore, we can write

$$M_{k+1}^T Y \leq M_k^T Y . \quad (\text{IV.106})$$

The bound on each individual place marking can be written as

$$M_{k+1}(p) \leq \frac{M_k^T Y}{Y(p)} . \quad (\text{IV.107})$$

In a GPN, however, not all the places are of PN types and can have both negative and positive markings. Stability or boundedness of a GPN depends on both its synchronous and asynchronous parts. Either proof of stability of the H matrix or boundedness of synchronous places alone does not prove the overall system stability and boundedness.

Theorem 4.3. GPN Boundedness and Stability Condition: Any given GPN is both bounded and stable if there exists a vector Y such that

$$\exists Y > 0 \text{ and } N^T Y \leq 0 , \quad (\text{IV.108})$$

and all roots of its characteristic equations, given by the eigenvalues of $[zI - H(z) - I]$, lie inside the unit circle.

Proof: The dynamic equation of a GPN is

$$\begin{aligned} M(k+1) &= M(k) + H_k M(k) + N f_k , \\ M(k+1) &= (H_k + I) M(k) + N f_k . \end{aligned} \quad (\text{IV.109})$$

If we write the z-transform for this equation, we get

$$\begin{aligned}
 zM(z) &= (H(z) + I)M(z) + Nf(z) , \\
 M(z) &= [zI - H(z) - I]^{-1} Nf(z) .
 \end{aligned}
 \tag{IV.110}$$

The stability of this net then depends upon where the poles of the system or the roots of the system characteristic equation, given by the eigenvalues of $[zI - H(z) - I]$, fall. Places with eigenvalues of less than one (whose corresponding poles are within the unit circle) are stable and bounded no matter what the N matrix is. Places with eigenvalues greater than one are unstable and consequently unbounded. Places with their poles on the unit circle are critically stable, and their stability and boundedness depend on the N matrix.

Lemma 4.3.1. Boundedness and Stability Conditions of Critically Stable Places: Any place with its corresponding H matrix row or column equal to a zero vector has an eigenvalue equal to one. Such places are critically stable can become unbounded if their corresponding elements in vector $N^T Y$ are not less than or equal to zero.

Proof: The dynamic equations governing the changes in the place markings are given by the followings:

$$\begin{aligned}
 zM(z) &= (H(z) + I)M(z) + Nf(z) \\
 M(z) &= [zI - H(z) - I]^{-1} Nf(z) .
 \end{aligned}$$

The change in a place marking is a function of the corresponding H matrix elements and the previous markings. The following equation shows the H matrix in terms of the net

parameters A, B, and F matrices. Derivation of this expression is given in Appendix B.

$$H = -\text{Diag}(Af) + [\text{One}(A)\text{Diag}(f)B]^T =$$

$$\begin{bmatrix} -\sum_{j=1}^n a_{1j}f_j + \sum_{j=1}^n o_{1j}f_jb_{j1} & \sum_{j=1}^n o_{2j}f_jb_{j1} & \dots & \sum_{j=1}^n o_{lj}f_jb_{j1} \\ \sum_{j=1}^n o_{1j}f_jb_{j2} & -\sum_{j=1}^n a_{2j}f_j + \sum_{j=1}^n o_{2j}f_jb_{j2} & \dots & \sum_{j=1}^n o_{lj}f_jb_{j2} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{j=1}^n o_{1j}f_jb_{jn} & \sum_{j=1}^n o_{2j}f_jb_{jn} & \dots & -\sum_{j=1}^n a_{lj}f_j + \sum_{j=1}^n o_{lj}f_jb_{jn} \end{bmatrix} \quad (\text{IV.112})$$

The change in place p_i marking is governed by the i^{th} row of the H matrix at that instant. All the elements in the i^{th} row of the H matrix can become zeros, either because of weight matrices A and B, or the transition firing matrix F. In this case the characteristic root (poles of the system) due to that place will be of the form $(z-1)$. That is, the pole falls on the unit circle.

If the whole of the H matrix becomes a null matrix, then

$$zM(z) = M(z) + Nf(z)$$

$$M(z) = [zI - I]^{-1}Nf(z). \quad (\text{IV.113})$$

This is the dynamic equation of the PN. In this case the stability problem becomes a boundedness problem and is dealt with the same way we deal with the PNs.

Boundedness also can be analyzed by reachability tree analysis. Presence of an infinity symbol in a place marking indicates that that place is unbounded.

IV.8. Liveness

Another problem which is of interest in resource allocation is avoidance of deadlock [89, 90]. A deadlock in a Petri net is defined as a transition or a set of transitions which cannot fire [85]. A transition is potentially fireable if, starting from an initial marking, there is a reachable marking in which that transition is fireable. A net is called live if all transitions in that net are potentially fireable.

In a GPN, all pure synchronous transitions (those which have no pre-conditions for the firing), are live all the time since they can continue firing no matter what marking their input

places have. However we can have a situation where the fired transition does not change the markings. For example when the input place marking is zero, the transition firing can go on, but the input and output place markings are not altered by the firing. Therefore in liveness analysis of the GPNs, we consider only the synchronous transitions which can become unabled due to their input place markings.

The liveness of a net depends on its initial marking and that is why it is analyzed by the reachability tree method. The reachability tree analysis only proves the liveness of a net for a given initial marking. Structural liveness, in general, is difficult to show and only certain classes (such as free-choice and marked graph nets) can be handled [91] by linear algebraic methods.

IV.9. Safeness

Safeness is a special case of the boundedness property. In the Petri net theory, a safe place can have only zero or one token at a time or is 1-bounded. This way a place can be represented by a flip-flop. This property can be used in fault detection to check if a place has erroneously acquired more than one token when it was meant to be a safe one. This property can be defined only for the asynchronous global Petri net places and not for its synchronous ones.

The way to check for this property is the same as the one for checking the boundedness except that the bound is taken to be equal to 1.

IV.10. Reachability

Reachability is the most basic problem of a Petri net analysis. A marking is reachable if there exists a sequence of transition firing which, starting at the initial marking, results in that marking. The reachability problem is analyzed both by reachability tree and matrix-equation methods. Reachability of a set of places also can be analyzed and is referred to as sub-marking reachability.

Chapter V Global Petri Net Simulation and Analysis Tool (GPNSAT)

In this chapter some of the main issues regarding the development and use of a tool for simulation and analysis of hybrid systems are discussed. We start by showing how a net structure can be prepared from other types of information known about the system. We then continue by describing the tool structure and its various functions.

V.1. Modeling Procedure

Figure V.23 shows the GPN model development procedure. It shows how various system specifications can be translated into GPN parameters. Any system to be modeled can be specified as a set of events and conditions, state space representation, transfer function, or a set of logical statements or simultaneous equations. The modeling procedure changes these specifications into a set of GPN model parameters, such as number of places, transitions, and arc weight matrices. This gives us a number of small GPN models which, when combined by the GPN engine, can represent the complete system. The GPN engine is used for model simulation and analysis. At first we show a top view of the tool structure and then describe its various parts.

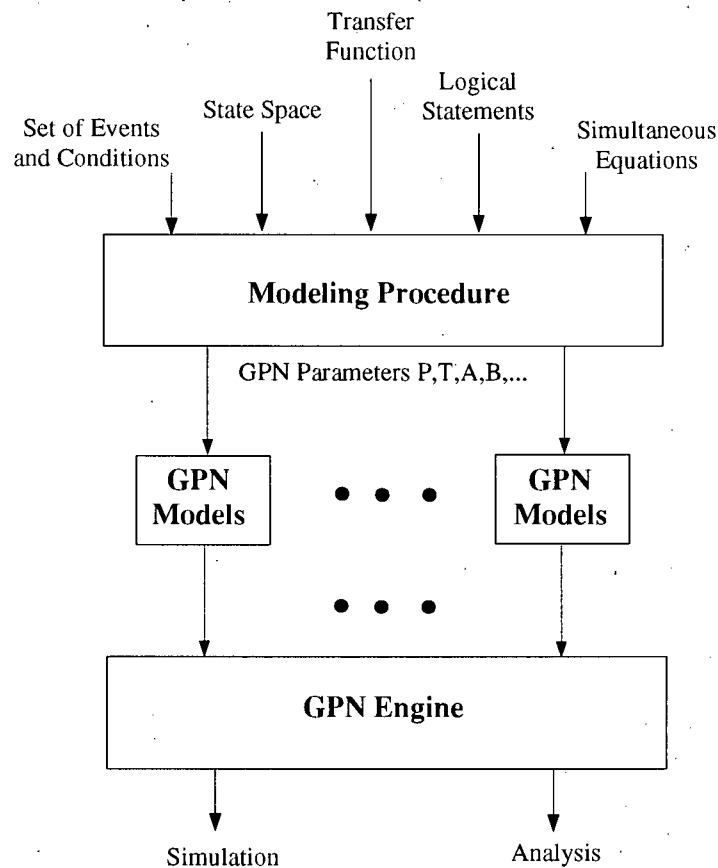


Figure V.23 Modeling Procedure Used for Translation of System Specifications into a Complete Net Model.

V.2. GPNSAT Top View Structure

The GPN simulation and analysis tool (GPNSAT) consists of many parts, as shown in Figure V.24. Rounded boxes represent different modules written for performing simulation and analysis of Petri and global Petri nets. The four main parts are the human interface, net utilities, simulators, and analysis tools.

All of these programs have been written in MATLAB [92]. The MATLAB package is very suitable for our purposes since many of the required matrix manipulation routines are already provided. Moreover, routines written in the MATLAB programming language or C can be used to provide a structured environment for program development.

At the heart of the GPNSAT is a human interface which takes the input parameters and passes them onto various programs as desired by the user. Inputs can be entered interactively through a keyboard or via an already composed file. Outputs are displayed both graphically and numerically. Input/output devices in Figure V.24 are represented by grey boxes.

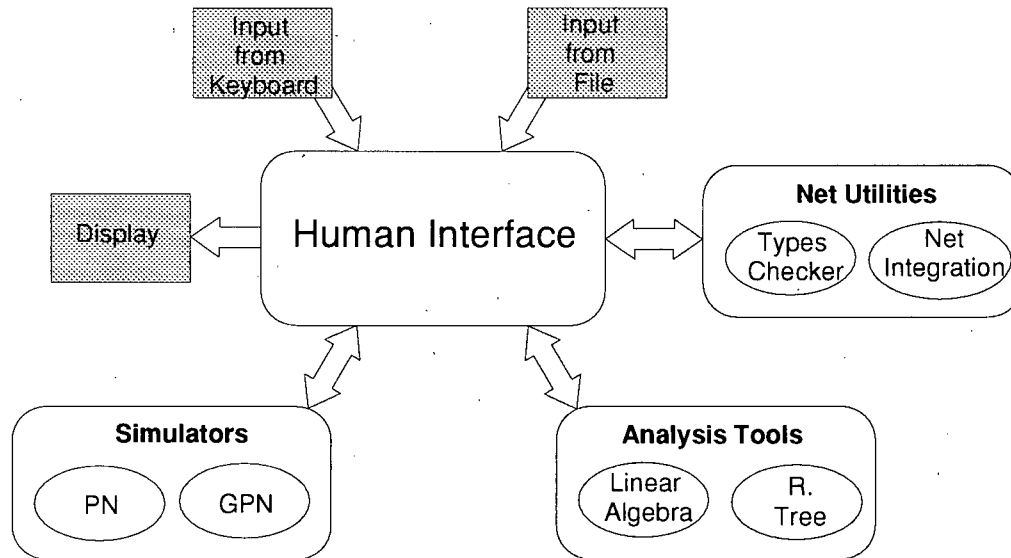


Figure V.24 The Overall Structure of the GPN Simulation and Analysis Tool (GPNSAT).

V.3. GPNSAT Substructures

In this section we describe some of the programs which have been written to model, simulate, and analyze various systems by the GPN formalism. Some important implementations issues are discussed, and the trade-offs are explained.

V.3.1. Human Interface: This module is used for interacting with the user to obtain net parameters. The user has two options for inputting the net parameters. S/he can do it interactively by answering a series of questions, or ask the program to access the required information, which is stored in a file. This module can also import directly the parameters from the net integrator described later on.

Once the parameters are input, the human interface checks to make sure that there are no inconsistencies in the data. These inconsistencies could be the result of entering arc weight

matrices with inappropriate sizes, forgetting to enter initial markings, or simply a violation of the PN/GPN rules as given by the PN/GPN formal definitions. The user is informed of any input inconsistencies which are detected by the HI, and is asked to check and re-enter the parameters.

V.3.2. Net Utilities: The net utilities module consists of a set of programs used for defining nets. A routine called type checker is used to determine what the types of different transitions and places are. The net integration module takes smaller nets as its input and puts them into a large net, which can then be used for simulation or analysis. In the following we describe each of these routines.

Type Checker: Types of places and transitions play an important role in the simulation of GPNs. Depending on what the types are, different firing rules are applied. The types of places or transitions can be defined by the user, and then there will not be any need to use this routine. But if the types are not known or need verification, this routine can check them. Definitions of types of GPN places and transitions were given in Section III.9. These definitions and the algorithm based on them, which determines these types from given arc weight matrices, are used in this routine.

Net Integration: As mentioned in Section V.1, modeling with GPN can be carried out by constructing a series of small nets representing various parts of the complete system. These smaller nets can then be juxtaposed to form a complete picture of the total system. The advantage of this kind of modeling is that smaller nets can be analyzed individually and their properties proved before being integrated into the system. Another advantage is that we can modularize the modeling and analysis processes. In this way a module can be used in many parts of the system and, if the need arises, it can be replaced by a new one.

Let two smaller nets be defined as:

$$GPN_1 = (P_1, T_1, W_{pt_1}, W_{tp_1}, A_1, B_1, M_1(0))$$

and

$$GPN_2 = (P_2, T_2, W_{pt_2}, W_{tp_2}, A_2, B_2, M_2(0)),$$

with their places given as

$$P_1 = \{p_1, p_2, \dots, p_l, p_{l+1}, \dots, p_{l+c}\} \text{ and}$$

$$P_2 = \{p_l, p_{l+1}, \dots, p_{l+c}, p_{l+c+1}, \dots, p_{c+l'}\}.$$

That is, GPN_1 has $l + c$ places, out of which the last c places are the same one as those of net GPN_2 . Net GPN_2 has $c + l'$ places, with first c places same as those of GPN_1 . The transitions of the two nets are given as

$$T_1 = \{t_1, t_2, \dots, t_n, t_{n+1}, t_{n+2}, \dots, t_{n+d}\} \text{ and}$$

$$T_2 = \{t_n, t_{n+1}, \dots, t_{n+d}, t_{n+d+1}, \dots, t_{d+n'}\}.$$

That is, GPN_1 and GPN_2 have $n + d$ and $n' + d$ transitions, respectively. Out of these transitions, they share d transitions. The transitions are numbered such that the first $n + d$ transitions belong to GPN_1 , and the last $d + n'$ transitions belong to GPN_2 .

The weight matrices for these two nets have the following dimensions:

GPN_1	Matrix	Size	Matrix	Size
	A_1	$l + c \times n + d$	B_1	$n + d \times l + c$
	W_{pt_1}	$l + c \times n + d$	W_{tp_1}	$n + d \times l + c$
GPN_2	A_2	$c + l' \times d + n'$	B_2	$d + n' \times c + l'$
	W_{pt_2}	$c + l' \times d + n'$	W_{tp_2}	$d + n' \times c + l'$

These two nets can be combined to form a larger net. This resulting net has a smaller number of places and transitions than the two nets put together, since we can eliminate the

redundant transitions and places which are common to both nets. The new net, called *GPN*, has the following structure:

$$GPN = (P, T, W_{pt}, W_{tp}, A, B, M(0)),$$

where

$$P = \{p_1, p_2, \dots, p_l, p_{l+1}, \dots, p_{l+c}, p_{l+c+1}, \dots, p_{l+c+l'}\},$$

$$T = \{t_1, t_2, \dots, t_n, t_{n+1}, \dots, t_{n+d}, t_{n+d+1}, \dots, t_{n+d+n'}\}.$$

The dimensions of the combined net weight matrices are:

<i>GPN</i>	<i>Matrix</i>	<i>Size</i>	<i>Matrix</i>	<i>Size</i>
	<i>A</i>	$l + c + l' \times n + d + n'$	<i>B</i>	$n + d + n' \times l + c + l'$
	<i>W_{pt}</i>	$l + c + l' \times n + d + n'$	<i>W_{tp}</i>	$n + d + n' \times l + c + l'$

The *A* and *B* matrices, given *A*₁, *B*₁, *A*₂, and *B*₂ can be written as:

$$A = \begin{matrix} & t_1 & t_2 & \dots & t_n & t_{n+1} & \dots & t_{n+d} & t_{n+d+1} & \dots & t_{n+d+n'} \\ \begin{matrix} p_1 \\ p_2 \\ \vdots \\ p_l \\ p_{l+1} \\ \vdots \\ p_{l+c} \\ p_{l+c+1} \\ \vdots \\ p_{l+c+l'} \end{matrix} & \begin{bmatrix} A_1 & & & & & & A_1 & 0 & 0 & 0 \\ & & & & & & & 0 & 0 & 0 \\ & & & & & & & 0 & 0 & 0 \\ & & & & A_2 & & & & & A_2 \\ & & & & & & & & & \\ & & & & & & & & & \\ A_1 & & & & & & A_1 & & & \\ 0 & 0 & 0 & & & & & & & \\ 0 & 0 & 0 & & & & & & & \\ 0 & 0 & 0 & A_2 & & & & & & A_2 \end{bmatrix} \end{matrix},$$

$$B = \begin{matrix} & p_1 & p_2 & \dots & p_l & p_{l+1} & \dots & p_{l+c} & p_{l+c+1} & \dots & p_{l+c+l'} \\ \begin{matrix} t_1 \\ t_2 \\ \vdots \\ t_n \\ t_{n+1} \\ \vdots \\ t_{n+d} \\ t_{n+d+1} \\ \vdots \\ t_{n+d+n'} \end{matrix} & \begin{bmatrix} B_1 & & & & & & B_1 & 0 & 0 & 0 \\ & & & & & & & 0 & 0 & 0 \\ & & & & & & & 0 & 0 & 0 \\ & & & & B_2 & & & & & B_2 \\ & & & & & & & & & \\ & & & & & & & & & \\ B_1 & & & & & & B_1 & & & \\ 0 & 0 & 0 & & & & & & & \\ 0 & 0 & 0 & & & & & & & \\ 0 & 0 & 0 & B_2 & & & & & & B_2 \end{bmatrix} \end{matrix}.$$

The *W_{pt}* and *W_{tp}* matrices also can be formed in a similar fashion.

V.3.3. Simulators: Simulation modules simulate many different types of PNs and GPNs. Both timed and untimed versions of these nets are simulated. The first version of this simulator was written for ordinary PNs. Later, timing specifications were added, which enabled modeling of timed Petri nets (TPN). The human interface, based on the net parameters, can decide which one of the simulators is needed. It then can call the appropriate simulator to perform the simulation.

Petri Net Simulator: The PN simulator can model any discrete-event system. After getting the net parameters from the human interface routine, it can run the simulation for a specified number of sample periods. At the end of the simulation run, the results are presented graphically by plots of the variations in token counts of each place.

The simulator consists of many smaller routines, with each performing a specific task. For example, one routine is used to decide which transition should be fired next when there are more than one enabled transitions. This selection can be decided by a random order assignment or can be prioritized. In the latter case, some transitions have a higher priority and are fired first.

The simulator program is a loop which runs for the number of time instants specified by the user. In each pass, the first thing to do is to find the transitions which are enabled. This is done by applying the PN enabling rules. Once all enabled transitions are determined, one of these transitions is selected to be the next one to be fired. The transitions are fired by subtracting tokens, specified by the arc weights connecting them to their input places, from the input places markings.

Once the firing is done in this way, a flag is set to mark the transition busy for the time specified by its transition time. Then the pass is continued by checking whether any more transitions are still enabled, taking into account the token movements after the first firing in that run. This is continued until there are no more enabled transitions left. Then the present pass ends and a new one starts by checking whether any transition has finished

its firing. Completion of a transition firing is completed by adding tokens to output places of the transition.

In the untimed version of this simulator, no time is specified for transitions. The firings in this case are atomic actions, and additions and subtractions of tokens are done simultaneously. Other details of the simulation are the same as for the timed version.

Global Petri Net Simulator: The GPN simulator was written based on the PN simulator. Its major difference is its capability in simulating synchronous and hybrid nets. The GPN simulation program also consists of many smaller modules. Some of these modules are the same as the ones in the PN simulator, but some are written specifically for GPN simulation.

Two routines new to this simulator are ones which implement functions *One()* and *Diag()*, defined in the GPN dynamic equations. These functions are necessary in every simulation pass to find the system H matrix.

Like the PN simulator, every simulation pass starts by deciding which asynchronous transitions to fire next. After firing the first enabled transition, the process continues until all enabled transitions are found. The synchronous transitions are fired once in every pass, unless they are busy completing their earlier firing.

V.3.4. Analysis Tools: Another important part of the GPNSAT is its analysis tools. This part also can be invoked by the human interface upon the user's request. The analysis tools module is dedicated to analyzing the given nets. Both reachability tree and linear algebraic analysis methods are available. These programs can check a variety of net properties such as boundedness, stability, and controllability. Depending upon which property one wants, one or both of the analysis methods are used.

Reachability Tree Analysis: The module for reachability tree construction is based on the algorithm developed in Figure IV.19. This algorithm can be used for construction of both PN and GPN reachability trees, using two different routines. The programs closely follow

the steps enumerated in the algorithm. In MATLAB, a number called Inf is used to denote an arbitrarily large number. This value is used in defining the infinity (ω) nodes of the tree. The RT program continues generating new nodes until all the branches of the tree are processed. At the end, each node is defined as one of the two types, DEAD END or OLD.

Once the RT is constructed, we can check for different net properties, such as boundedness, liveness, and conservation.

Linear Algebraic Analysis: The other analysis tool is based on the linear algebraic method. This method finds all possible H matrices for a given net by firing different combinations of hybrid transitions. These H matrices, along with the system incidence matrix N, form the basis for the analysis. This method finds the boundedness property by checking if the inequalities formed by

$$\exists Y > 0 \text{ and } N^T Y \leq 0$$

hold true. The stability of the system is checked by finding whether the roots of $[zI - H(z) - I]$ fall within or outside the unit circle.

Chapter VI Modeling and Analysis of a Hydraulic Control System

In this chapter we demonstrate how our extension of the Petri net theory, developed in the previous chapters, can be used to model and analyze a complete real-time hybrid control system. We also will show how this system is simulated and analyzed by the GPNSAT package.

We have chosen a real-time hydraulic control system, which is a very good example of a hybrid system. This system has many interacting parts with stringent fault detection and identification requirements. One of the advantages of using a Petri net is the ability to model a system at various levels of abstraction. This advantage becomes quite evident in the following presentation. We have modeled the system as a distributed computing system with nodes dedicated to various tasks, such as control, monitoring, and supervision. At the highest level, the system interactions are modeled by a conventional Petri net as an asynchronous system. Later in this chapter, parts of the system are modeled at a more detailed level as hybrid subsystems.

VI.1. Overall System Structure: Level 0

Figure VI.25 shows a block diagram representation of the hardware structure of a hydraulic machine control system [93]. This structure was used for the dynamic simulation and control of teleoperated hydraulic manipulators [94]. The system consists of a controller, a computing node (CPU and memory), and a controlled subsystem, which is an excavator. There are also many sensors and actuators which are used for detection and application of outputs and control inputs, respectively. All of the subsystems are connected to a system bus which also can be replaced with a communication channel when the system is implemented in a teleoperated mode.

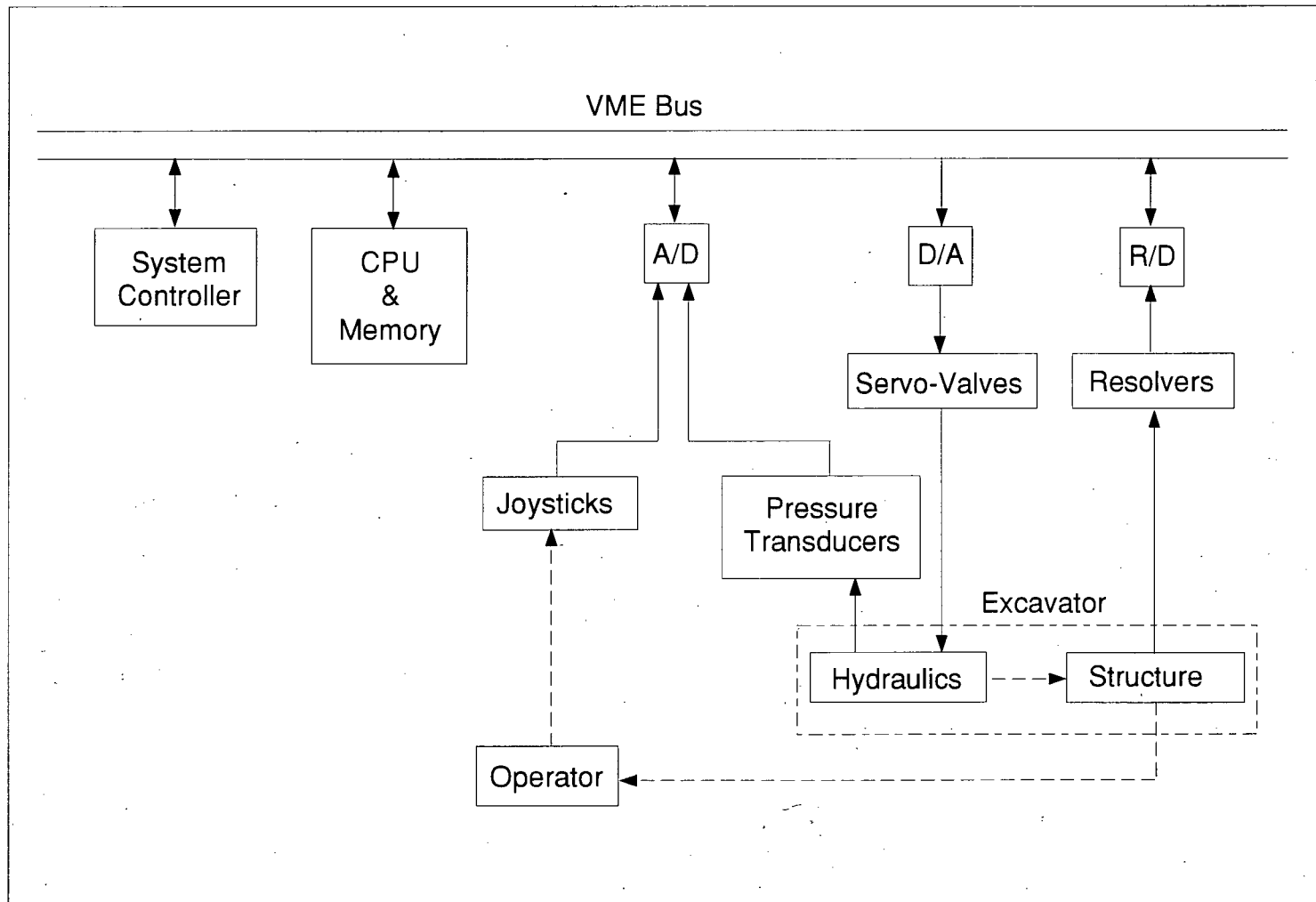


Figure VI.25 Block Diagram of a Hydraulic Control System Hardware Structure.[93]

Figure VI.26 shows a generalized and simpler representation of the previous block diagram. It shows a number of subsystems connected to and sharing a common system bus. We assume that the bus is the critical resource, which should be shared by the subsystems, and that its performance and utilization could affect the whole system. Any other part of the system, such as CPU or memory, could also be treated and studied as a resource. But in this study we consider only the bus.

The subsystems are numbered one through N , of which the first two and the last one are shown in Figure VI.26. Each subsystem can access and take hold of the bus and use it for its transactions, while the others wait for its release. At this level, the system can be modeled as an asynchronous system. We are not interested in the internal functions of each subsystem and model only the bus utilization.

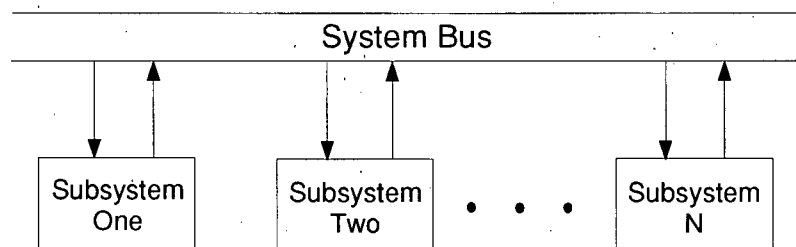


Figure VI.26 Block Diagram of a Distributed System at Level 0.

The above system is modeled by a Petri net, as shown in Figure VI.27. There are four places representing the three subsystems and the system bus as places $p_1 - p_4$, respectively. Arrival of a token in one of the first three places indicates that that particular subsystem has a task to perform and needs to access the system bus. Transitions $t_1 - t_3$ represent the arrival of tasks at subsystems one through three, respectively. These transitions represent external events and fire due to external conditions not modeled within this system. Transitions $t_4 - t_6$ represent the system bus accesses by subsystems $p_1 - p_3$, respectively.

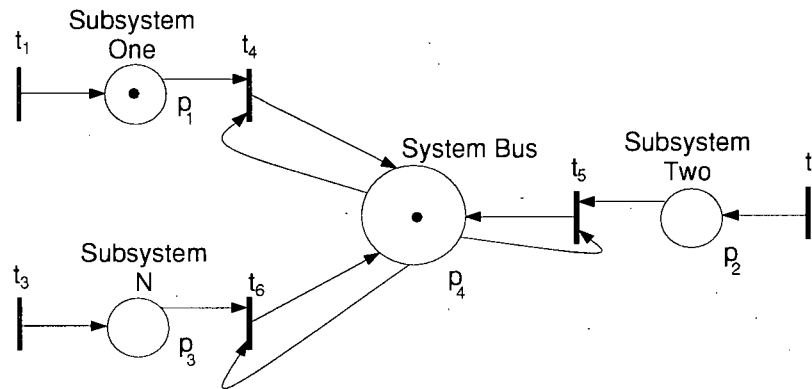


Figure VI.27 Petri Net Model of the System shown in Figure VI.26.

Net parameters for this example are defined as

$$\begin{aligned}
 P &= \{p_1, p_2, p_3, p_4\} & T &= \{t_1, t_2, t_3, t_4, t_5, t_6\} \\
 W_{pt} &= \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} & W_{tp} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 M(0) &= [1 \ 0 \ 0 \ 1]^T & & \\
 TT &= [5 \ 10 \ 15 \ 2 \ 4 \ 6]. & &
 \end{aligned} \tag{VI.126}$$

P and T are the sets of places and transitions, respectively. The W_{pt} and W_{tp} matrices give the weight of the arcs connecting places to transitions, and vice versa, respectively. For example, $W_{pt}(1,4) = 1$ indicates that place p_1 is connected to transition t_4 by an arc whose weight is 1. $M(0) = [1 \ 0 \ 0 \ 1]^T$ is the initial marking or the marking at time instant zero. It indicates that initially there is one token in places p_1 and p_4 , and none in other places. This is illustrated in Figure VI.27 by the presence and absence of dots (tokens) in the circles representing the respective places. TT is the transition time vector, which associates a time with each transition firing. $TT(4) = 2$ indicates that transition t_4 takes 2 sampling periods to complete (each system bus access by subsystem one lasts 2 sampling periods).

The net in Figure VI.26 was simulated by the simulation package described in the previous chapter. Figure VI.28 shows the results of a simulation run. Plots (a) through

(d) are the token count at places $p_1 - p_4$, respectively. Subsystems one through three get a new token, at the rate defined by their input transition firing time. For example, place p_1 receives a token every five sample periods as a result of the firing of its input transition t_1 . Subsystems two and three receive a new token every 10 and 15 sample periods, as determined by their input transition firing time. Ideally, each token should be consumed by the place output transitions before a new one arrives. That is, each subsystem should have a chance to access the bus frequently enough to avoid a pile-up of tokens at the place representing it. The token arrival time and bus access time for this example were selected so that the pile up situation happens. As can be seen in plots (a) and (b), token counts (marking) at places p_1 and p_2 get larger as time goes on. This is more severe in the first subsystem since it gets updated more often. On the other hand place p_3 marking, shown as plot (c), never exceeds one, indicating that the bus can keep up with the new token arrivals at this place. The marking at place p_4 is given as plot (d). This plot shows how the bus switches between idle and busy states. The bus utilization for this case is around 92%. This could be around 100% if it were not for the initial idle time in the beginning, when none of the subsystems needed the bus. The bus utilization and accumulation of tokens in subsystems can be used to decide when a faster bus is needed.

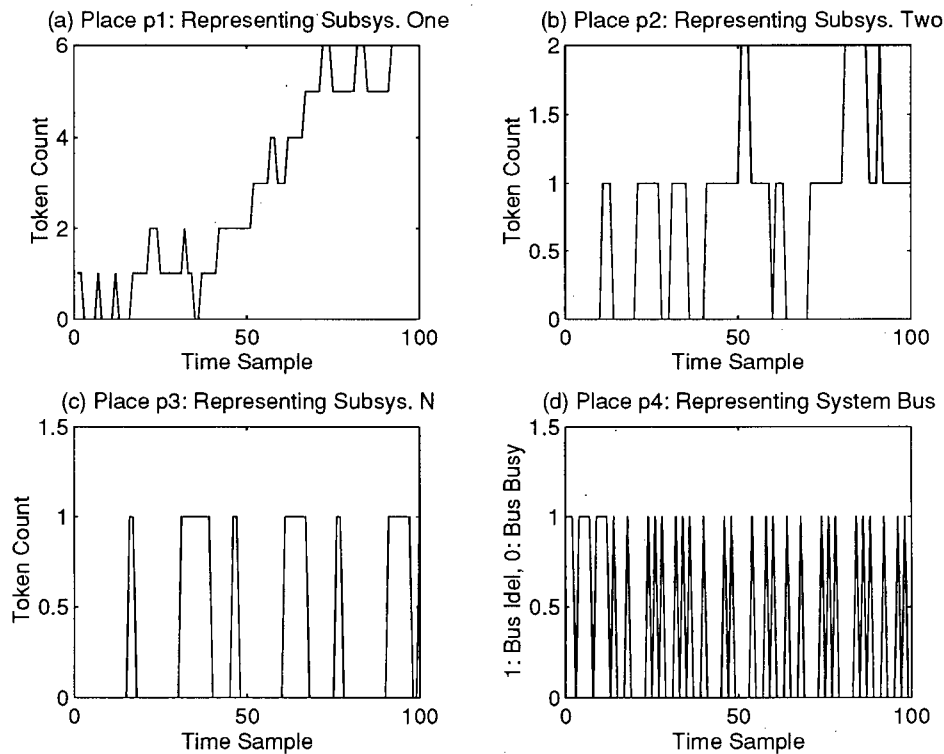


Figure VI.28 Simulation Results of the PN Model Given in Figure VI.27. (a) Place p_1 Marking. (b) Place p_2 Marking. (c) Place p_3 Marking. (d) Place p_4 Marking.

In the current example all subsystems were taken to have the same access priority. We can easily simulate many different operation scenarios to study the effect of having a faster bus, more than one bus, less frequent token arrivals, and various priority policies amongst the subsystems. All these are done simply by changing one or more of the initial settings for the net. For example, let us increase the number of bus channels from one to two by setting the initial marking to $M(0) = [1 \ 0 \ 0 \ 2]^T$. In this case, the bus can easily meet the transaction demands by the three subsystems. Figure VI.29 shows the plots of the same parameters as those already shown in Figure VI.28. As can be seen, all the bus accesses by the three subsystems are going through in time, and no pile-up of tokens takes place.

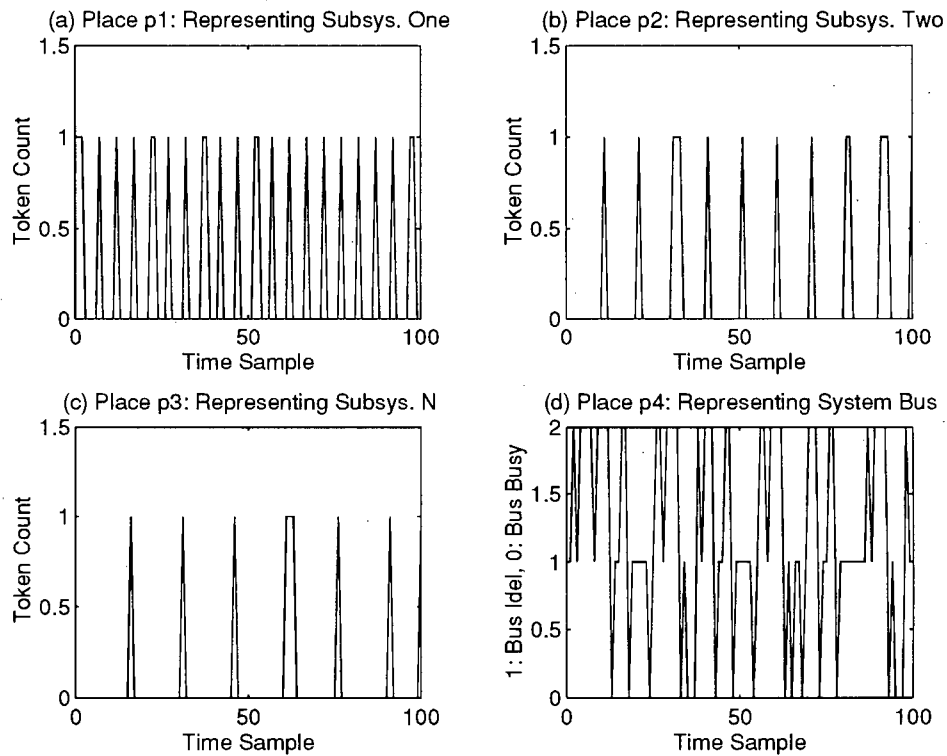


Figure VI.29 Simulation Results of the PN Model Given in Figure VI.27 when the Number of Bus Channels is Increased to Two. (a) Place p_1 Marking. (b) Place p_2 Marking. (c) Place p_3 Marking. (d) Place p_4 Marking.

VI.1.1. Petri Net Analysis of the Model at Level 0: We can use the given Petri net to analyze the system model for the various properties which were described earlier.

Reachability Tree Analysis: We start the analysis by constructing the reachability tree for the previous example. We use the RT construction tool described in Chapter V to obtain the reachability tree shown in Figure VI.30.

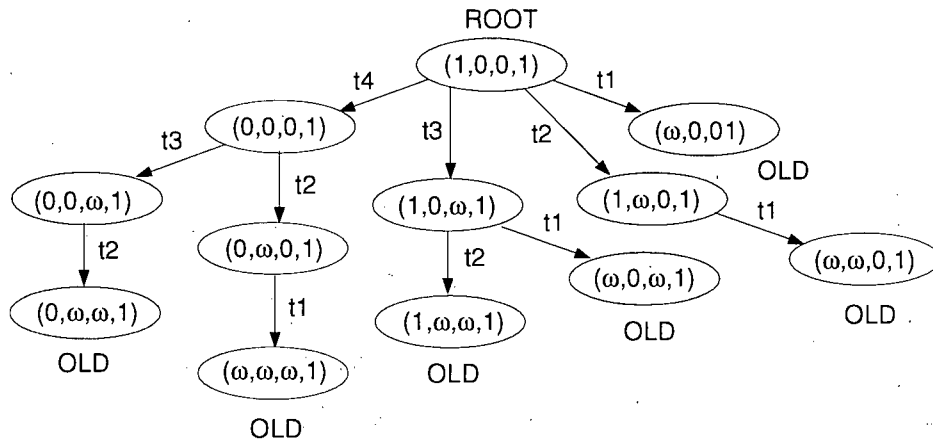


Figure VI.30 The Reachability Tree of the Petri Net Shown in Figure VI.27

Figure VI.30 can be used to investigate many of the modeled system properties. These are as follows:

1. **Boundedness:** The appearance of the infinity symbol ω indicates the possibility of unboundedness in a particular place. With reference to the tree, we conclude that the first three places, representing the subsystems, are unbounded. As we saw in the simulation results, this situation can happen when the subsystems cannot access the bus before the next token arrives. Place four, representing the system bus, is safe since its marking never exceeds one. We also constructed the RT for the case where there are two bus channels available. The only difference was in the number of tokens in place p_4 since it can now have some nodes with two tokens. All other markings remained the same. This shows that, with the current structure, the number of bus channels does not eliminate the possibility of a subsystem going unbounded.
2. **Liveness:** Another interesting fact deduced from the above analysis is the absence of deadlocks, since none of the nodes are marked "DEAD END". This and the fact that the fourth place marking never exceeds one, indicates the mutually exclusive use of the bus by subsystems without getting into a deadlock situation.
3. **Conservation:** Neither the complete net nor any set of its places is conservative, as the appearance of the ω symbol in the places' marking indicates.

4. **Reachability:** As the RT shows, only those states in which place four has one token are reachable. All other places may reach any marking, starting from the given initial marking.

Linear Algebraic Analysis: Other properties of the net can be investigated by the linear algebraic method. The incidence matrix for this net can be written as

$$N = W_{tp}^T - W_{pt} = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (\text{VI.127})$$

For this net to be bounded, given that Y is an arbitrary positive vector, the following inequalities should hold:

$$N^T Y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (\text{VI.128})$$

These inequalities can be written as

$$\begin{aligned} y_1 &\leq 0 \\ y_2 &\leq 0 \\ y_3 &\leq 0 \\ -y_1 &\leq 0 \\ -y_2 &\leq 0 \\ -y_3 &\leq 0. \end{aligned} \quad (\text{VI.129})$$

Since the first three above inequalities contradict the original assumption of Y being positive, we conclude that the net is not bounded. In fact, this is the same result obtained by the RT analysis.

Controllability of the net can be checked by the rank of the incidence matrix N . Since

$$\text{rank}(N^T) = 3 \quad (\text{VI.130})$$

is less than the number of places $m=4$, this net is not controllable. This is due to the fact that place four, which represents the system bus, cannot be made to have any arbitrary number of tokens.

VI.2. A More Detailed Model of the Overall System: Level 1

In the previous section we showed how a distributed computing system can be modeled and analyzed at its highest level as an asynchronous system by a Petri net. In this section we present the modeling of this system at a more detailed level of abstraction. Figure VI.31 shows part of the distributed system, consisting of three subsystems. The first two subsystems represent the field and computing subsystems, which are physically separate and communicate through the system bus. To enable study of the effects of all other subsystems, we have lumped everything else together into a subsystem called 'All Other Subsystems'. The field unit consists of the plant and its input-output interfaces with the world, such as sensors and actuators.

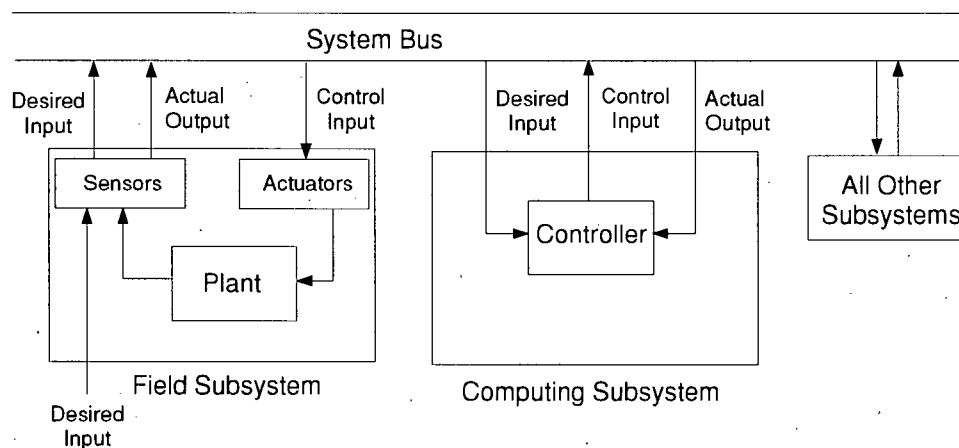


Figure VI.31 A More Detailed Model of the Overall System: Level 1.

This system at this level can be modeled by another Petri net, as shown in Figure VI.32. This net provides a more detailed description of the system under study by modeling the internal structure of each subsystem.

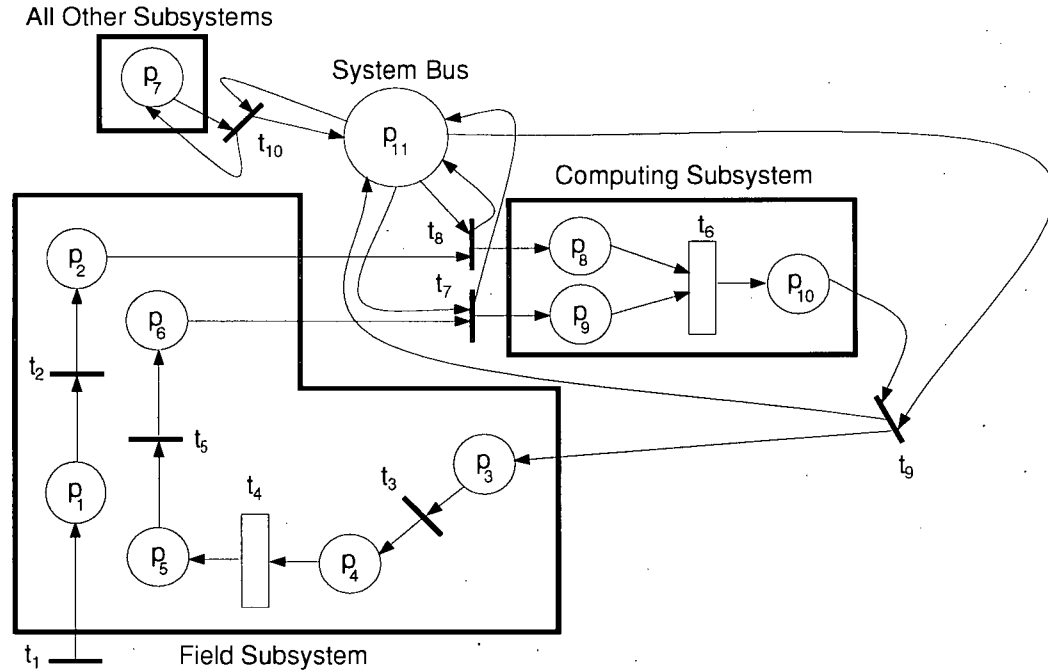


Figure VI.32 A Petri Net Model of the Distributed System Given in Figure VI.31.

Places

p_1 : User Input
 p_2 : Sensor Output
 p_3 : Control Input
 p_4 : Actuator Output
 p_5 : Plant Output
 p_6 : Sensor Output
 p_7 : All Other Subsys.
 p_8 : Desired Input
 p_9 : Actual Output
 p_{10} : Controller Output
 p_{11} : System Bus

Transitions

t_1 : User
 t_2 : Sensor
 t_3 : Actuator
 t_4 : Plant
 t_5 : Sensor
 t_6 : Controller
 t_7 : Bus Access
 t_8 : Bus Access
 t_9 : Bus Access
 t_{10} : Bus Access

The above net is still a conventional Petri net and does not have any of the GPN nodes. In a system represented by an event-driven model, we only know when an event starts and ends and have no idea what happens in between. For example when both places p_8 and p_9 have a token (both the desired input and the actual output are present), the controller routine

can run which is modeled by the firing of transition t_6 . The next information available to us is when this process is completed. The completion is modeled by the arrival of a token at place p_{10} . We have no idea what happens while the event is in progress.

The other information which is missing in the discrete-event model is the actual (quantitative) value of each signal. For example, all we know is that the control input is generated and available (a condition). But we have no information as to what its value is. In order to see the dynamics of the system in between events, we can model the process as a time-dependent element by a GPN.

In Figure VI.32, transitions t_4 and t_6 represent the plant and the controller, respectively. These transitions are shown as boxes instead of bars. Each of these transitions can be modeled by a GPN with both synchronous and asynchronous elements.

VI.3. A Hydraulic Control System

The parts of the system that we have chosen to model are the field and computing (controller) subsystems (Figure VI.31). The resulting model at this level is a global Petri net which represents a hybrid system consisting of both time and event-driven parts. The GPN modeling methodology used in this example can be applied to other hybrid systems, including those in flexible manufacturing systems, process control, robotics, and communication systems.

The system shown in Figure VI.33 is a two-stage electro-hydraulic valve along with its control circuitry [95, 96]. It consists of a pilot spool valve whose movement varies the differential pressure acting at the ends of the main spool. The output flow rate from the main spool is used to drive the piston, which is provided with a feedback loop.

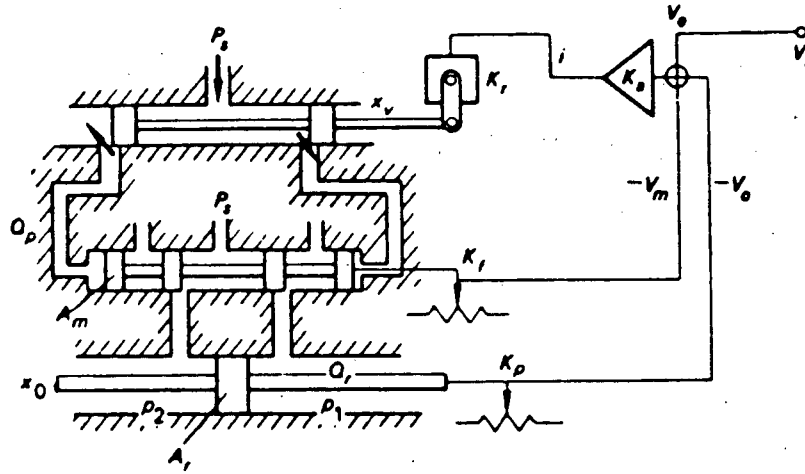


Figure VI.33 Schematic Diagram of a Two-stage Hydraulic Valve System [95].

By deriving the relevant equations which describe the hydraulic system (Figure VI.33), we can draw the block diagram shown in Figure VI.34. It is a second order system with feedback. All the parameters in the block diagram correspond to those in the schematic diagram represented in Figure VI.33.

We derived the transfer function for the block diagram in Figure VI.34 as

$$\frac{X_o}{V_i} = \frac{\frac{1}{K_p}}{\frac{2A_m A_r}{K_p K_a K_r K_1 K_1' P_s} S^2 + \frac{K_f A_r}{K_p K_1' \sqrt{\frac{P_s}{2}}} S + 1} \quad (\text{VI.132})$$

The above transfer function can be written in the state space form described by the following equations:

$$\begin{aligned} \dot{X}(s) &= AX(s) + BU(s) \\ Y(s) &= CX(s) \end{aligned} \quad (\text{VI.133})$$

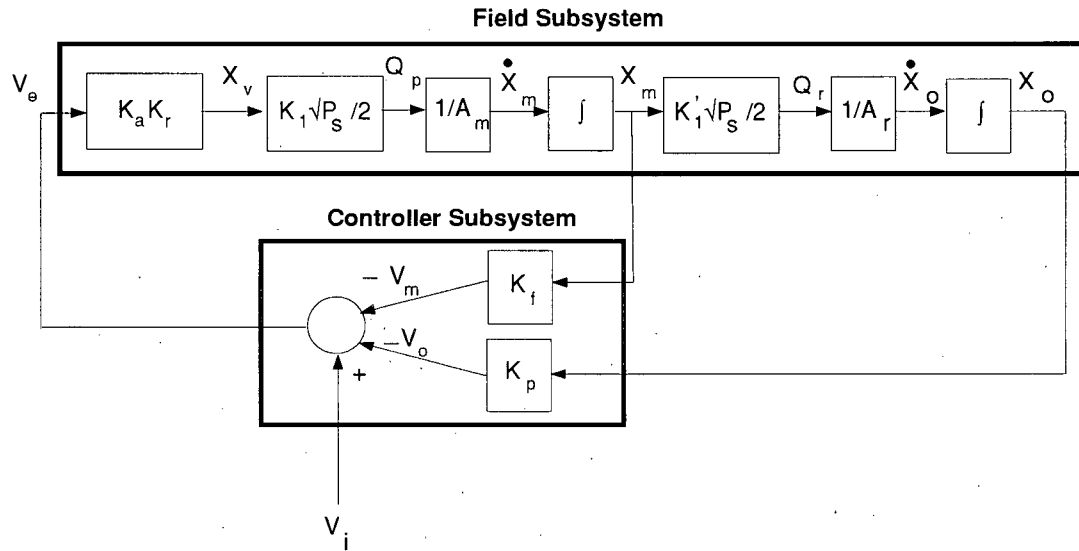


Figure VI.34 Block Diagram Representation of the Plant and the Controller Corresponding to Transitions t_4 and t_6 in Figure VI.32.

X_o and X_m which are output and main valve displacements, respectively, are selected as the system states. We have also selected the outputs to be the same as these two states. The control input (V_e) is the difference of the reference input (V_i) and the feedback from the system (X_o).

$$\begin{aligned} X(s) &= \begin{bmatrix} x_1(s) \\ x_2(s) \end{bmatrix} = \begin{bmatrix} X_o(s) \\ X_m(s) \end{bmatrix}, \\ Y(s) &= \begin{bmatrix} X_o(s) \\ X_m(s) \end{bmatrix} \quad U(s) = V_e(s). \end{aligned} \quad (\text{VI.134})$$

With this structure, we will have the following state space matrices in terms of the system variables:

$$\begin{aligned} A &= \begin{bmatrix} 0 & \frac{K_1' \sqrt{P_s}}{A_r} \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{K_a K_r K_1 \sqrt{P_s}}{A_m} \end{bmatrix}, \\ C &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \end{aligned} \quad (\text{VI.135})$$

The controller was designed by varying the feedback gains to attain a fast tracking of the system reference inputs and a small steady state error. The control input can be written as

$$U(s) = V_e(s) = V_i(s) - K_p Y_1(s) - K_f Y_2(s), \quad (\text{VI.136})$$

where V_i is the reference input, and K_p and K_f are the feedback gains.

The system was converted from continuous to discrete-time by assuming a zero-order hold on the inputs and a sample time equal to 0.02 seconds. The discretized system in state space form can be written as

$$\begin{aligned} X(k+1) &= AX(k) + BU(k), \\ Y(k) &= CX(k), \end{aligned} \quad (\text{VI.137})$$

where X , Y and U represent the same states, outputs, and inputs, respectively. The system parameters (A , B and C matrices) for the discrete model are

$$A = \begin{bmatrix} 0 & 2.5 \\ 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \quad (\text{VI.138})$$

VI.4. GPN Model of the Hydraulic System: Level 2

In this section we present the GPN model of the hydraulic system and then show how it can be integrated into the model developed at level 1. As a general procedure, we should represent each of the inputs, outputs and system states as distinct places. The relation among these states or places is modeled by transitions and arcs connecting them. Figures VI.35 and VI.36 represent the GPN models for the plant and the controller respectively.

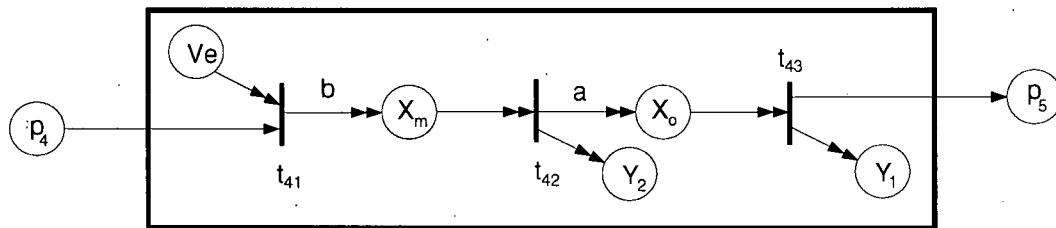


Figure VI.35 Global Petri Net Model of the Plant Shown in Figure VI.34.

The a and b in the above figure are given as

$$a = \frac{K_1' \sqrt{\frac{P_s}{2}}}{A_r}, \quad b = \frac{K_a K_r K_1 \sqrt{\frac{P_s}{2}}}{A_m} \quad (\text{VI.139})$$

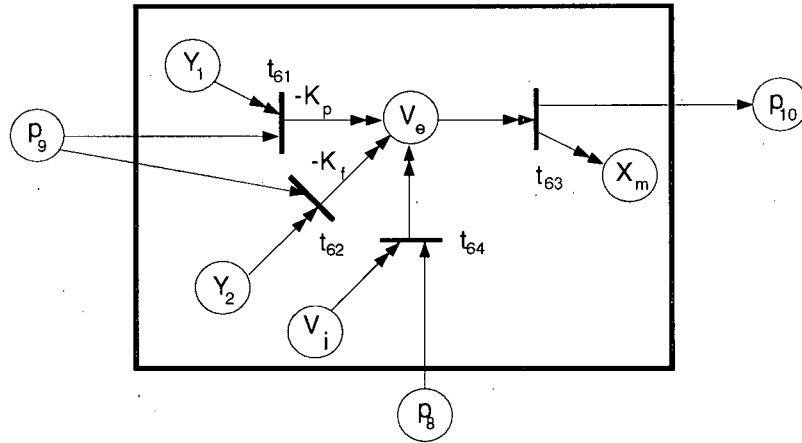


Figure VI.36 Global Petri Net Model of the Controller Shown in Figure VI.34.

The complete hybrid system consists of the net in Figure VI.32 with transitions t_4 and t_6 replaced by their equivalent GPNs (shown in Figures VI.35 and VI.36). The system at this level was simulated by the GPNSAT (Global Petri Net Simulation and Analysis Tool) package .

Figures VI.37(a) and (b) show the plant output and the control input, respectively, as simulated by the GPN model. These are the states modeled by places V_e and Y_1 in Figure VI.35. We would have obtained the same results if we had simulated the system as a conventional control system without any of the distributed communications (system bus).

The results of the simulation by a conventional control system simulation program (MATLAB), in continuous-time, are shown in Figures VI.37(c) and (d) for comparison. The plant was discretized at 0.02 second (50 Hz) as mentioned in the previous section. In each sample period, which takes 0.02 seconds, the outputs are sensed and control inputs are applied exactly once. The transition time of the system bus is chosen to be one third of the plant sampling time, i.e. $0.02/3 = 0.0067$ second. The reason for this selection is that the

bus has to be accessed at least three times during each plant sampling period. These accesses are by two output sensors (twice) and by the actuator input (once). In this way, the speed of the communication (bus access) is three times that of the plant sampling time. That is, transitions t_4 and t_6 take three times as long to fire as do transitions $t_7 - t_{10}$.

The GPN simulation was run for 100 sampling periods (each equal to one transition sampling time). Because of the reason just explained, the results that we get after 100 transition sampling periods ($100 \times 0.02/3 = .66$ second of real time operation) of simulation by the GPN model is the same as those obtained after 33 sampling periods ($33 \times 0.02 = .66$ second of real time operation) of simulation by the MATLAB program.

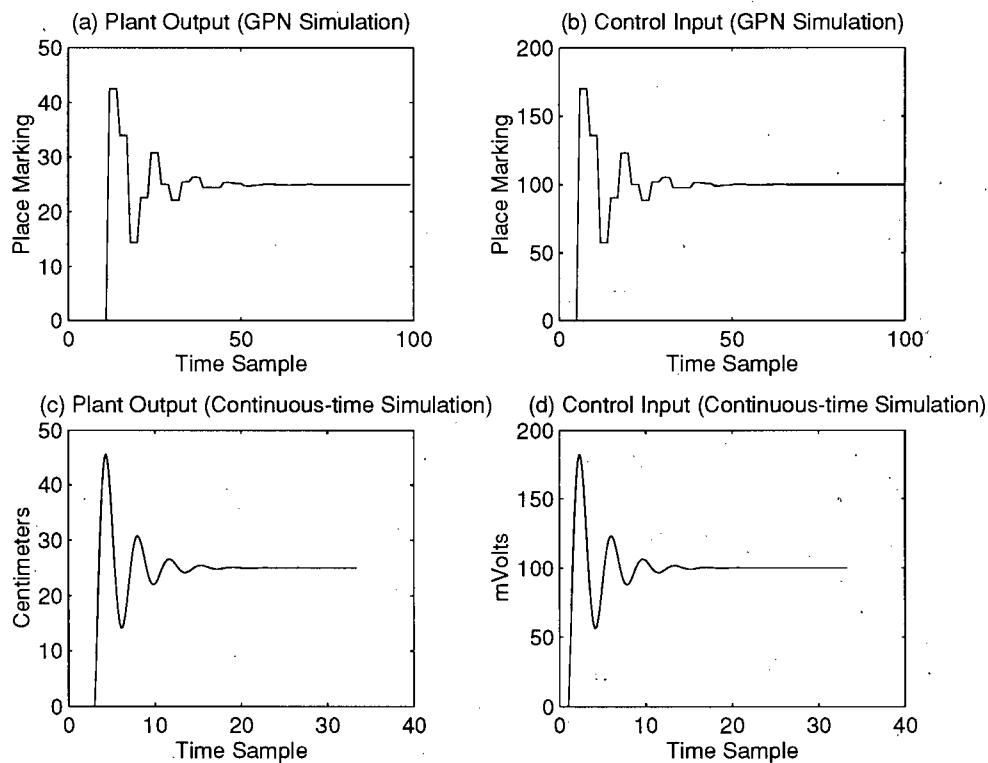


Figure VI.37 Comparison of Simulation Results of the Complete System by the GPN Model and a Conventional Control System Simulator. (a) Plant Output Place X_o in Figure VI.35, (b) Control Input Place V_e in Figure VI.36, (c) Plant Output X_o , (d) Control Input V_e .

Figure VI.38 shows the token count for places p_5 and p_{10} . A token in place p_5 indicates that the plant output has been read from the plant. A token in place p_{10} shows that the

control input is computed by the controller and can be used to access the system bus to pass the value to the plant. The token count for the first 100 samples is plotted here. As can be seen in Figure VI.38, there is a regular access at almost every third time sample. The token count becomes one when a new value of the control input is calculated, and drops to zero once it is transferred by the system bus. It should be mentioned that these places are of the asynchronous type, and their synchronous behavior shows that the system is working as expected. As a result, the bus accesses are going through very regularly and exactly at every third sample. In the next chapter we show what happens when there are timing faults and all the bus accesses cannot get through.

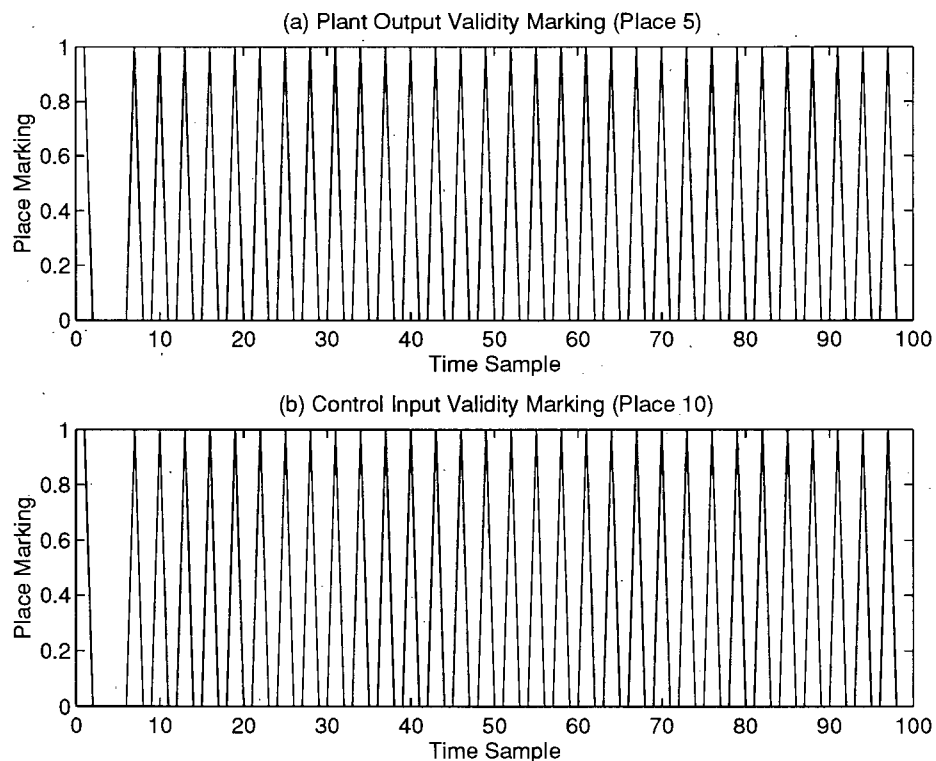


Figure VI.38 Result of the Simulation of the Complete System by the GPN Model (a) Plant Output Validity Place p_5 in Figure VI.32, (b) Control Input Validity Place p_{10} in Figure VI.32.

VI.5. GPN Analysis of the Hydraulic System

In this section, the complete GPN model is analyzed in order to check the properties of the hydraulic control system. A complete analysis requires both the RT and linear algebraic methods.

VI.5.1. Linear Algebraic Analysis: The complete system is modeled by a net consisting of the Petri net represented in Figure VI.32 and global Petri nets in Figures VI.35 and VI.36. The complete system GPN diagram and its parameters, such as weight matrices, are given in Appendix D.

For this system to be both stable and bounded, we need to have a bounded incidence matrix N and stable H matrices. We first look at the boundedness property. The system incidence matrix $N = W_{tp}^T - W_{pt}$ can be written as

$$N = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 4 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

N is bounded iff, for an arbitrary vector $Y > 0$, the inequalities resulting from $N^T Y \leq 0$ hold true. If we substitute N from above into $N^T Y \leq 0$ we get the following set of inequalities:

$$\begin{aligned} y_1 &\leq 0 & y_2 - y_1 &\leq 0 & 4y_4 - y_3 &\leq 0 \\ y_5 - y_4 - y_8 - y_9 &\leq 0 & y_6 - y_5 &\leq 0 & y_{10} - y_4 - y_8 - y_9 &\leq 0 \\ 4y_9 - y_6 &\leq 0 & 4y_8 - y_2 &\leq 0 & y_3 - y_{10} &\leq 0 \\ -y_4 - y_8 - y_9 &\leq 0 & & & & \end{aligned}$$

The above set of inequalities was run through the analysis tools of GPNSAT and could not be satisfied. As an example, the first inequality $y_1 \leq 0$ is clearly against the condition of Y being a positive vector and indicates that the first place is unbounded. We therefore, conclude that the net is unbounded.

There are all together four hybrid transitions ($n_h = 4$), in the net. These transitions are t_{41} in Figure VI.35, and t_{61}, t_{62} , and t_{64} in Figure VI.36. According to Theorem 4.1 in Chapter IV, we will have $2^{h_s} = 2^4 = 16$ H matrices. These matrices correspond to when one or more of these hybrid transitions fire.

t_{41}	t_{61}	t_{62}	t_{64}	Location of Roots of H+I Matrix not on the Unit Circle
0	0	0	0	None
1	0	0	0	0
0	1	0	0	-0.2
1	1	0	0	-0.1+0.7i, -0.1-0.7i
0	0	1	0	0
1	0	1	0	0,0
0	1	1	0	-0.2,0
1	1	1	0	-0.1+0.7i, -0.1-0.7i, 0
0	0	0	1	None
1	0	0	1	0
0	1	0	1	-0.2
1	1	0	1	-0.1+0.7i, -0.1-0.7i
0	0	1	1	0
1	0	1	1	0,0

Table VI.3 Roots of $H_k + I$ Matrices Formed by the Firing of Various Hybrid Transitions. (Continued) . . .

0	1	1	1	-0.2,0
1	1	1	1	-0.1+0.7i, -0.1-0.7i, 0

Table VI.3 Roots of $H_k + I$ Matrices Formed by the Firing of Various Hybrid Transitions.

For this system to be stable at all times, the roots of all characteristic equations formed by firing different hybrid transitions should have roots inside the unit circle. That is the eigenvalues of the various $H_k + I$ matrices should be less than one:

$$\begin{aligned}
 M(k+1) &= M(k) + H_k M(k) + N f_k \\
 M(k+1) &= (H_k + I)M(k) + N f_k .
 \end{aligned}
 \tag{VI.142}$$

Table VI.3 shows the location of the roots for all system H matrices, found by firing of various transitions. The first four columns show which of the four hybrid transitions are fired. The fifth column shows the location of the roots corresponding to those firings. For example in the first row, no transition is fired. This is the case when none of the hybrid transitions is able to fire. All roots of the H+I matrix in this case lie on the unit circle. The last row corresponds to the case where all transitions are fired and the system is working under normal or fault-free conditions. In this case, all roots are on the unit circle except three, which are located inside. This shows that all places are stable.

All rows in between the first and last rows represent cases where one or more transitions are not fired. They correspond to the situation in which one or more of the system level data transmission is not transferred. For example in row 15, transition t_{41} has not fired, which shows the case in which the control input is not applied to the plant.

A cursory look at the roots presented in the table reveals that the system is stable under all firing conditions. That is, no matter what hybrid transition fails to fire, the hydraulic system will remain stable.

VI.5.2. Reachability Tree Analysis: The reachability tree of this net was constructed by the GPNSAT package. The tree is quite large since there are about 15 places, each having

many different possible markings. Therefore, the tree cannot be drawn here, but, the results of the RT inspection can be summarized as follows:

1. **Boundedness:** The infinity symbol ω appears in places p_1 , p_2 , and p_8 , indicating that these places can become unbounded. Unboundedness of place p_1 is due to the transition which reads the operator's input. If these inputs are not processed fast enough, the place may go unbounded. Places p_2 and p_8 can also become unbounded due to the same reason (non-processing of input data).
2. **Liveness:** As with the higher level representation of the system, there are no deadlocks detected (there were no nodes labeled "DEAD END").
3. **Conservation:** The complete net is not conservative, due to the presence of infinite nodes. There might however be some subset of bounded places which are conservative.
4. **Reachability:** The RT showed that the only reachable states are those in which place p_{11} (which represents the system bus) has one token.

VI.6. Simulation and Analysis Performance

All the simulation and analysis works were performed by GPNSAT (GPN simulation and analysis tool) which was described in the previous chapter. All routines in GPNSAT are written as MATLAB [92] m-files. The net (at level-2) has 15 places and 15 transitions. The simulation of this net for 200 transition sampling periods takes about 115 seconds, running on a SUN SPARC 5 workstation. 200 transition sampling periods represent $200 \times 0.02/3 = 1.33$ second of real time operation. The simulation of the net at levels zero and one take significantly less time since they have fewer places and transitions. The time complexity of the simulation program is $O(p \times t)$ where p and t are the number of places and transitions respectively.

The linear algebraic analysis takes almost no time and produces virtually instantaneous results. The construction of the reachability tree for this net (at level-2) takes more than

two hours time since over 500,000 different states are to be constructed and checked for various properties.

Chapter VII Fault Modeling and Analysis by GPN

This chapter is dedicated to fault modeling, simulation, and analysis by global Petri nets. We only consider off-line analysis of faults and no attempt is made to use the GPN methodology for on-line detection and identification of faults. A fault detection, identification, and reconfiguration scheme is proposed in Appendix E. The scheme is based on recognition of fault conditions by comparing the actual system outputs with those simulated by a GPN based simulator. We shall use the system described in the previous chapter in order to show how faults in a hybrid system can be modeled and analyzed. We start with system level faults (bottlenecks) and see how they affect the system. We then get into how hydraulic system faults can be represented.

VII.1. System Level Faults

One of the most important factors in the design and operation of distributed discrete-event and hybrid systems is the utilization of critical resources. These resources become the system bottlenecks, and their management poses a serious challenge [97]. As described in the previous chapter, the bus in our hybrid system is modeled as a resource used by various subsystems. In the simulation results shown there, we had assumed that the subsystem marked "All Other" was dormant and did not need to access the bus (Figure VI.32). In that case, the bus was capable of meeting the demands. This manifested itself in all bus accesses going through on time and as per request [98].

To show how the system would behave if the system bus accesses did not go through, we simulated the system with some extra load. This is done by assigning a token to the initial marking of the place representing the "All Other" subsystem. When we do this, the system bus has to serve this client as well; and therefore, it cannot keep up with all the bus transactions which are requested.

The results of this simulation are plotted in Figure VII.39. The first two plots show the plant output and the control input. There is a noticeable difference in the plots when compared with the previous simulation results (Figure VI.37). Even though the system does not become unstable, the plant output takes a longer time to reach its desired value. The cause of this can be found in plot (c), which shows the token count at place p_2 . This place represents the validity of the data which the desired input sensor (e.g. input joystick) reads. When the system bus is able to meet all demands, there will be a regular bus access by this place. That means the marking can never exceed one, since the token is consumed before the next one arrives. But in the present simulation run, the bus access demands are not met. That is why we see two and even three tokens in place p_2 .

Plot VII.39(d) shows the token count at place p_7 . This place represents the "all Other" subsystem. We have chosen the arc weights so that one token is deposited in this place after every bus access. As is seen from the final token count at this place, there have been over thirty bus accesses by this subsystem. This represents the extra load on the bus, which has caused the difference in the hydraulic system behavior.

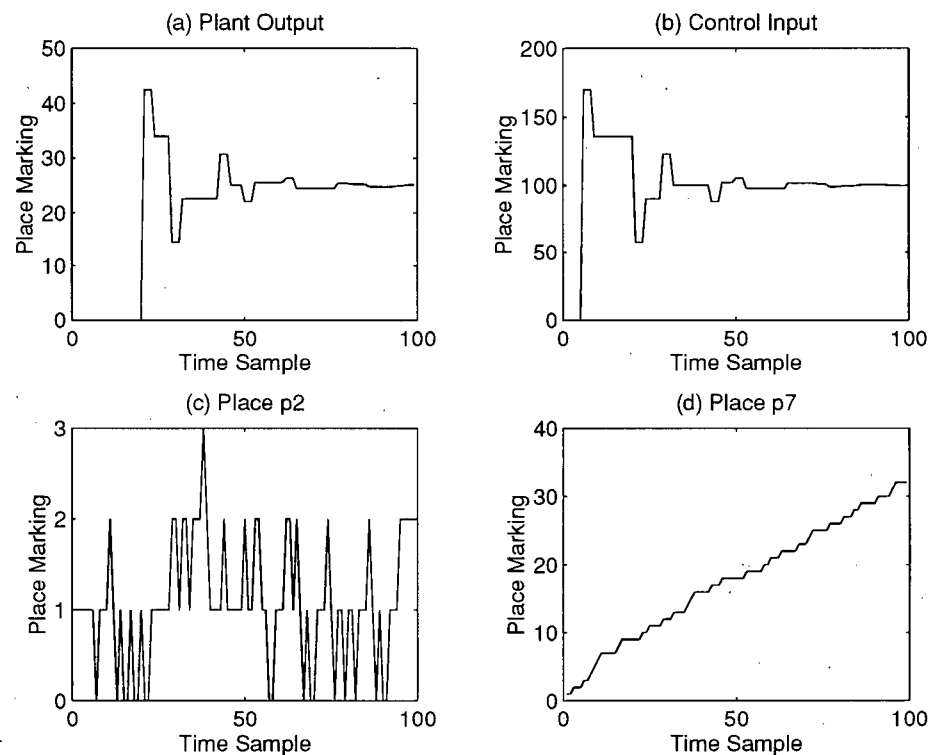


Figure VII.39 Simulation Results of the Hybrid System when the Bus Accesses are not Met. (a) Hydraulic System Output (Place X_o Marking). (b) Control Input (Place V_e Marking). (c) Desired Input Validity (Place p_2) Marking. (d) "All Other" Subsystem (Place p_7) Marking.

VII.2. Hydraulic System Faults

In this section we consider how different faults in the hydraulic system can be modeled by a GPN. In the later sections we take up the simulation and analysis of these faults. The faults considered are as follows:

1. Sensor faults
2. Actuator faults
3. Disturbances
4. System parameter changes.

These faults in the GPN model can be represented as events which happen asynchronously [99, 100]. Each fault condition is explicitly modeled by a place. Presence of a token in that

place indicates a potential fault. The fault occurs when the transition enabled by that fault place is enabled. In the following section we present the sensor fault modeling in detail and then apply the same technique to other fault types.

VII.2.1. Sensor Faults Model:

Let a sensor be modeled as a simple gain as described by the following equation [101]:

$$Y_s(k+1) = G_s Y(k), \quad (\text{VII.143})$$

where $Y(k)$, $Y_s(k)$, and G_s represent plant outputs (which are the sensor inputs), sensor outputs, and sensor gain, respectively [102]. We consider two types of faults and represent them with the following equation:

$$Y_s(k+1) = Y_s(k) - f_1 G_f Y(k) + f_2 G_s Y(k) + f_3 d - f_4 Y_s(k), \quad (\text{VII.144})$$

where G_f and d are a faulty gain change and a faulty bias, respectively. When all four transitions fire ($f_1 = f_2 = f_3 = f_4 = 1$), we will have

$$Y_s(k+1) = (G_s - G_f)Y(k) + d. \quad (\text{VII.145})$$

This system can be modeled with the global Petri net shown as Figure VII.40.

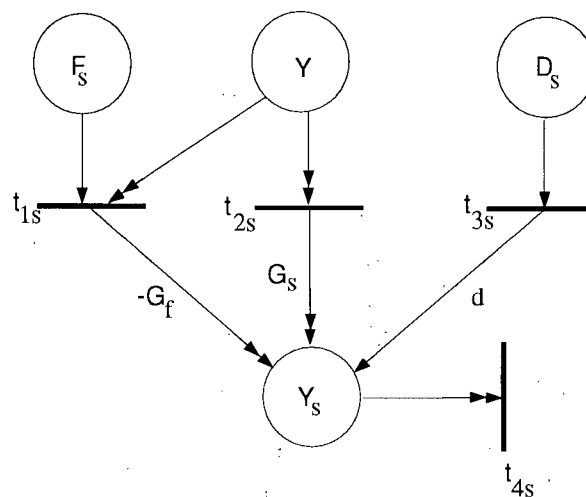


Figure VII.40 A Sensor Fault Model.

The GPN parameters can be written as

$$\begin{aligned}
 P &= \{F_s, Y, Y_s, D_s\} & T &= \{t_{1s}, t_{2s}, t_{3s}, t_{4s}\} \\
 W_{pt} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} & W_{tp} &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
 A &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} & B &= \begin{bmatrix} 0 & 0 & -G_f & 0 \\ 0 & 0 & G_s & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
 M(k) &= [F_s(k) \quad Y(k) \quad Y_s(k) \quad D_s(k)].
 \end{aligned} \tag{VII.146}$$

The above net can be used instead of the transitions representing sensors as shown in Figure VII.41. The sensor transitions in the net representing the hybrid system are transitions t_2 and t_5 (Figure VI.32). Places p_5 and p_6 here are the same places as in Figure VI.32. These two asynchronous places indicate whether the plant output (sensor input) and sensor output data are valid. The box representing transition t_2 models the sensor which is itself a smaller net; it was represented earlier in Figure VII.40.

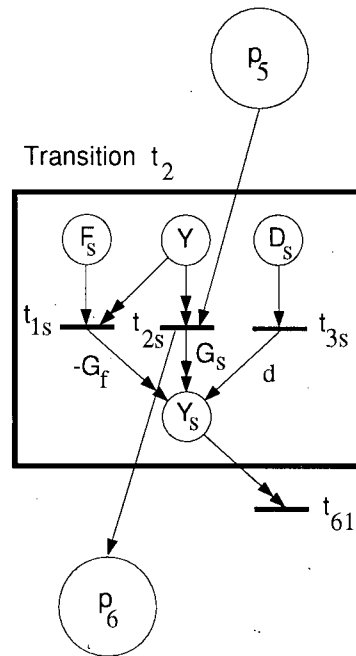


Figure VII.41 A Sensor Fault Model as a More Detailed Representation of Sensor Transition.

VII.3. Fault Scenarios: Simulation and Analysis

Fault simulation can be carried out by the GPN Simulation and Analysis Tool (GPNSAT). The net, modeling the system, is simulated as before. A fault is activated once the place which represents it gets a token. For example, in Figure VII.41, a gain change fault is introduced when place F_s gets a token. The presence of a token in this place enables transition t_{1s} . The firing of this transition effectively changes the sensor gain from a no fault value of G_s to a faulty value, $(G_s - G_f)$.

The sensor bias fault is activated by the presence of a token in place D_s . The firing of transition t_{3s} , which is enabled by this place, adds a bias to the eventual sensor output. The effects of these changes are reflected in the system simulation.

To give an illustration of the simulation results, let us start with the gain change faults in the first sensor. This sensor reads X_o and feeds it back to the controller. We introduced the fault with various gain values, G_f . The results are plotted in Figure VII.42. The simulation is run for the normal (no fault) system for 100 samples. This is sufficient for the system to reach the desired output and settle at its steady state values. At this point, faults are introduced by placing a token in the appropriate places.

VII.3.1. Sensor One Gain Change Fault Simulation: Figures VII.42 (a) and (b) show the hydraulic system output $y_1 = X_o$ and control input $U = V_e$ when there is a break in the feedback loop. This can be modeled by having $(G_s - G_f) = 0$ or $G_s = G_f = 1$. The value of G_s in the proceeding simulation runs is always taken to be one. Gain changes are achieved by changing G_f . As can be seen from the simulation results, the effect of this fault is to introduce a steady state error in the output. The reason for this can be seen in the analysis of these faults later on.

The next two plots in Figures VII.42 (c) and (d) represent the plant output, X_o and control input, V_e , when $G_f = -0.5$. This makes the faulty sensor gain $(G_s - G_f) = 1 - (-0.5) =$

1.5. The steady state effect of this fault is similar to the previous one with final steady state error of a different value. The transient behavior of this fault, however is, quite different.

The third fault introduced is a change of sensor gain to 2. This can be done by having $G_f = -1$. This fault causes an oscillatory type of behavior as can be seen in the plots in Figures VII.42 (e) and (f). The final fault in this series is a change of gain to 2.5 by setting $G_f = -1.5$. This fault makes the system unstable. The plant output and control input plotted in Figures VII.42 (g) and (h) keep increasing until they go out of bound.

VII.3.2. Sensor One Gain Change Fault Analysis: An analysis of the net representing the faulty system reveals many of its properties. The most important property is the stability of the whole system or its parameters. This can be analyzed by the linear algebraic method developed in the earlier chapters.

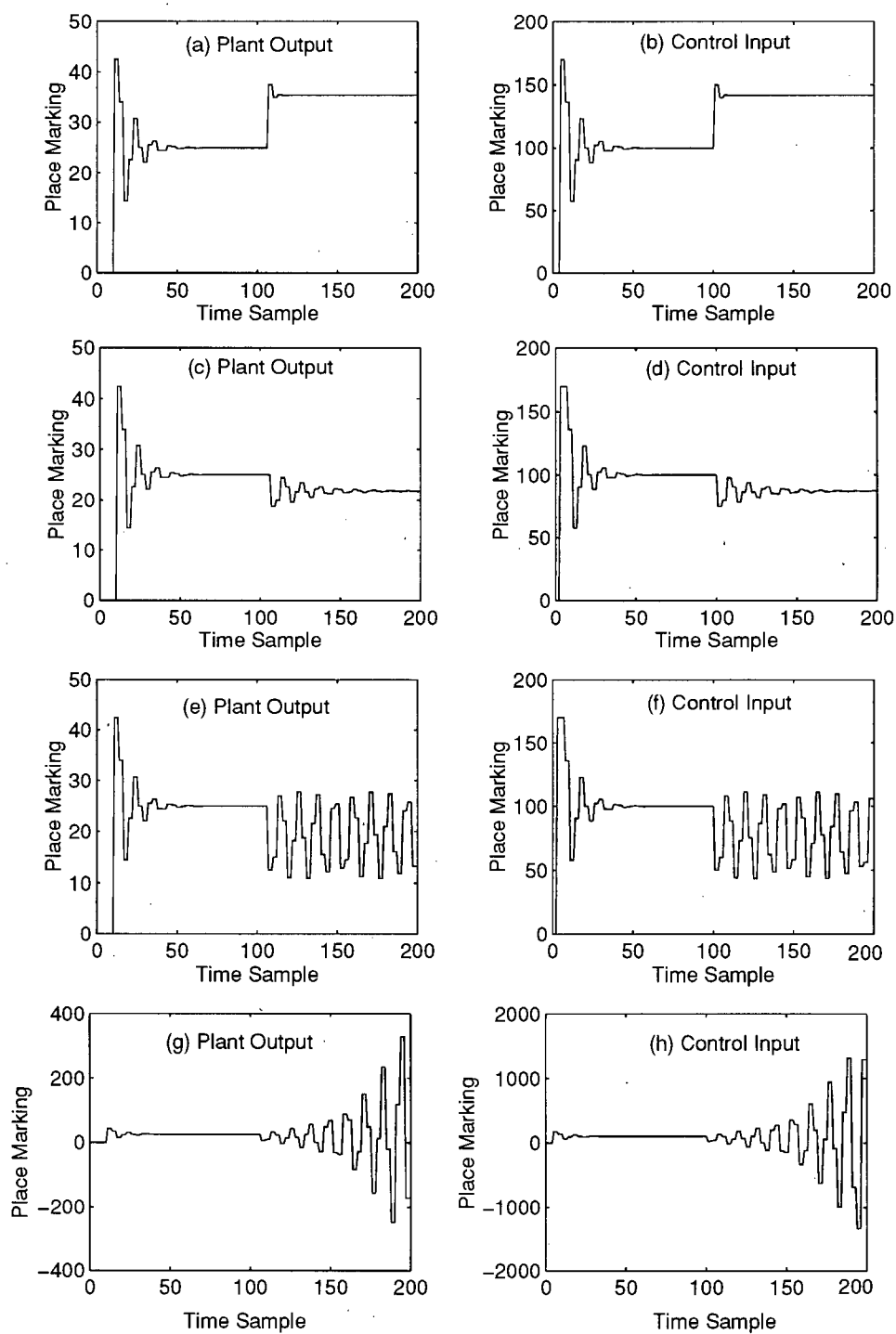


Figure VII.42 Simulation Plots of the First Sensor Gain Faults. a-b) Sensor Cut off, $G_f = 1$. c-d) Sensor Gain Change to $G_f = -0.5$. e-f) Sensor Gain Change to $G_f = -1$. g-h) Sensor Gain Change to $G_f = -1.5$.

Table VII.4 summarizes the type of faults introduced, the roots of their characteristic

equations found by checking the H matrices, and the effect of these faults. The first row shows the situation for the non-faulty system. As mentioned earlier, the two roots of the hydraulic system lie at $-0.1 + 0.7i$ and $-0.1 - 0.7i$. These roots are within the unit circle, which is why the system remains stable. We observe this behavior in all of the simulation plots in Figure VII.42, up to the 100th sample.

The gain change faults which do not force these roots out of the unit circle (row 2 and 3) of Table VII.4, have no effect on the system stability. Therefore, they only cause a steady state error, as seen in the simulation results.

The gain change fault of $G_f = -1$ (4th row), which makes the fault sensor gain of two, causes the roots to fall right on the unit circle. This makes the system behave in an oscillatory manner (critically stable), which is also evident in Figures VII.42 (e) and (f).

The faults in the next two rows are the ones which make the system unstable (any sensor gain greater than two). Again, this is because the roots found by an analysis of the H matrix fall outside the unit circle. This is the kind of system behavior depicted in Figures VII.42 (g) and (h).

Fault Type	First Sensor Overall Gain	Roots	Condition
No Fault $G_f=0$	1	$-0.1+0.7i$ $-0.1-0.7i$	Stable, No Error
Sensor cut off $G_f=1$	0	0, -0.2	Stable, Steady State Error
$G_f=-0.5$	1.5	$-0.1+0.8602i$ $-0.1-0.8602i$	Stable, Steady State Error
$G_f=-1$	2	$-0.1+0.995i$ $-0.1-0.995i$	Oscillatory

Table VII.4 Sensor One Gain Change Faults and Their Analysis Results. (Continued) ...

$G_f=-1.05$	2.05	-0.1+1.0075i -0.1-1.0075i	Unstable
$G_f=-1.5$	2.5	-0.1+1.1136i -0.1-1.1136i	Unstable

Table VII.4 Sensor One Gain Change Faults and Their Analysis Results.

VII.3.3. Sensor Two Gain Change Fault Simulation and Analysis: Figure VII.43 and Table VII.5 summarize the simulation and analysis results of faults caused by a change in gain of the second sensor. The results and conclusions of their analyses are similar to those for the first sensor. For the second sensor, any gain fault of $G_f = -6.5$ (overall sensor gain of 7.5), causes the system to become oscillatory. Any gain larger than that makes the system unstable.

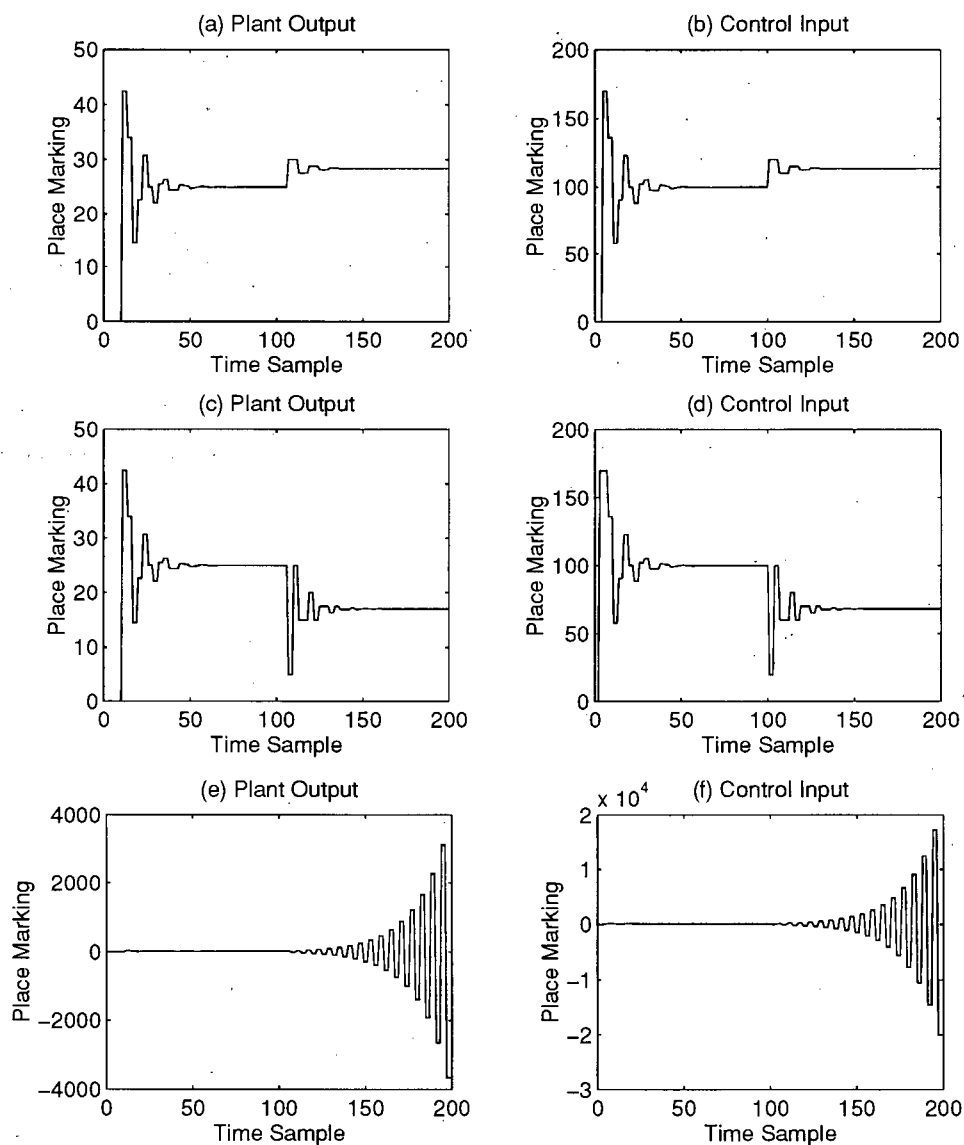


Figure VII.43 Simulation Plots of the Second Sensor Gain Faults. a-b) Sensor Cut off, $G_f = 1$. c-d) Sensor Gain Change to $G_f = -4$. e-f) Sensor Gain Change to $G_f = -7$.

An interesting observation in comparing the results of the two sensors analyses is that gain changes produce different roots and, consequently, different transient behavior. This difference can be a basis for the recognition of anticipated faults by different sensors in a

fault diagnostic procedures capable of capturing these distinguishing parameters.

Fault Type	Second Sensor Overall Gain	Roots	Condition
No Fault $G_f=0$	1	$-0.1+0.7i$ $-0.1-0.7i$	Stable, No Error
Sensor cut off $G_f=1$	0	$0+0.7i$ $0-0.7i$	Stable, Steady State Error
$G_f=-4$	5	$-0.5+0.5i$ $-0.5-0.5i$	Stable, Steady State Error
$G_f=-6.5$	7.5	-1, -0.5	Oscillatory
$G_f=-6.55$	7.55	-1.0196, -0.4904	Unstable
$G_f=-7$	8	-1.1742, -0.4258	Unstable

Table VII.5 Sensor Two Gain Faults and Their Analysis Results.

VII.3.4. Sensors Bias Faults Simulation and Analysis: The last set of sensor faults we consider include the disturbances which act as biases on the sensor output. In this particular system, since the outputs of the system sensors are added up by the control routine, the bias on each sensor has a similar effect and cannot be distinguished. In fact, all bias faults act as changes in the reference input.

The simulation results for two bias values are plotted in Figure VII.44. As can be seen, no matter what the amount of the bias is, the bias faults only introduce a steady state error. The reason for this can be found by referring to Table VII.6.

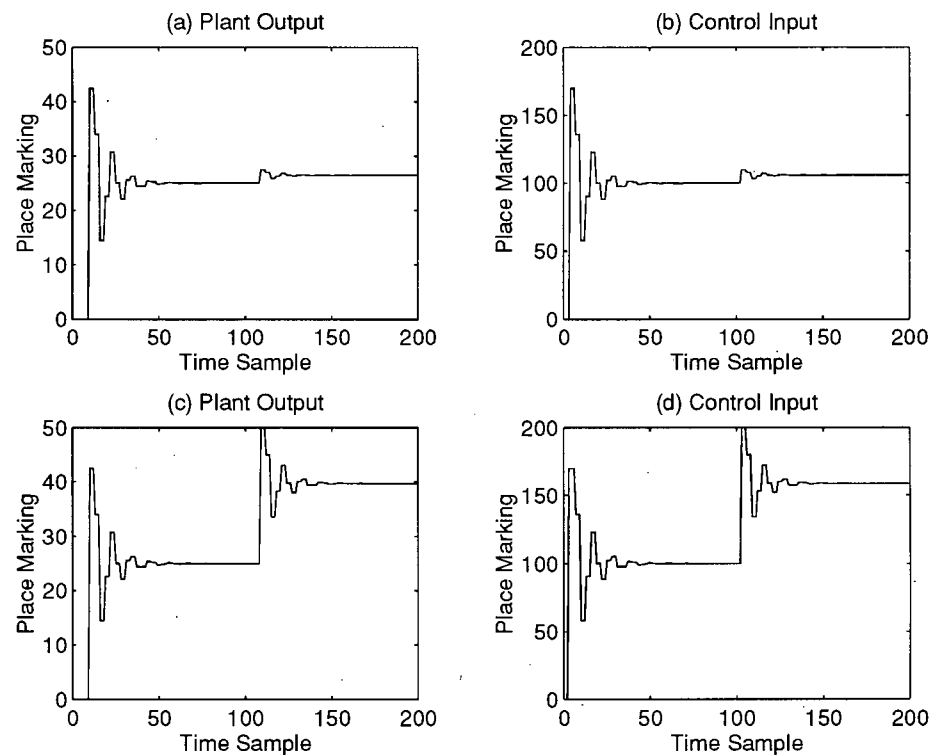


Figure VII.44 Simulation Plots of the Sensors Bias Faults. a-b) Sensor Bias of 10, $d = 10$. c-d) Sensor Bias of 100, $d = 100$.

These faults or any other bias faults do not change the H matrix, since they are modeled by the asynchronous places and transitions. Therefore, the bias faults do not affect the dynamics of the system, such as the roots of the H matrix. A stable system can absorb these kind of faults or load changes and stay stable.

Fault Type	Roots	Condition
No Fault	-0.1+0.7i -0.1-0.7i	Stable, No Error

Table VII.6 Sensor Bias Faults and Their Analysis Results. (Continued) . . .

d=10	-0.1+0.7i -0.1-0.7i	Stable, Steady State Error
d=100	-0.1+0.7i -0.1-0.7i	Stable, Steady State Error

Table VII.6 Sensor Bias Faults and Their Analysis Results.

VII.4. Other Fault Types

In this section we move to other types of hydraulic system faults. These faults, like the sensor faults modeled earlier, are common in all types of control systems. Here we briefly describe how these faults can be modeled. The same type of fault analysis that was carried out for sensor faults can be applied to these fault types as well.

VII.4.1. Actuator Faults Modeling: As the first step, for analysis of actuator faults, we need to model the actuators used in our system. Then these models which take the form of GPNs, can be integrated into the system GPN. In general and in its simplest form, actuators can be modeled as a gain which is described by the following equation:

$$U_a(k+1) = G_a U(k), \quad (\text{VII.147})$$

where $U(k)$ is the control input which is computed by the control subsystem and sent to the actuator; $U_a(k)$ is the actuator output which is applied to the plant to control it; and G_a is the actuator gain when there is no fault. The actuation process takes a finite amount of time (eg., one sampling period).

For this type of actuator we can consider two types of faults: a gain change fault and a bias fault. These faults can be modeled by the global Petri net shown in Figure VII.45.

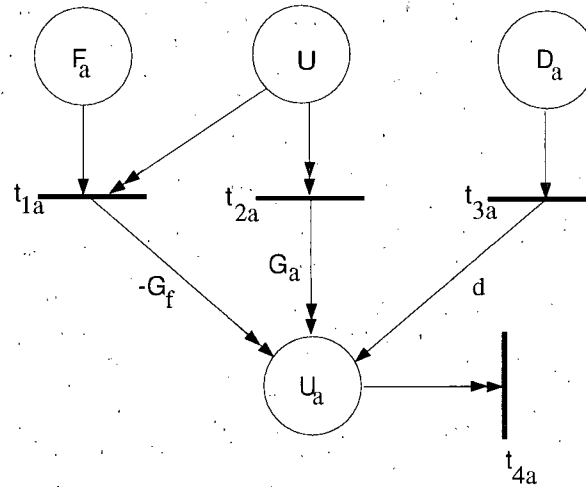


Figure VII.45 An Actuator Fault Model.

The operation of this net can be represented by the following equation:

$$U_a(k+1) = U_a(k) - f_1 G_f U(k) + f_2 G_a U(k) + f_3 d - f_4 U_a(k), \quad (\text{VII.148})$$

where G_f and d are a faulty gain change and a faulty bias, respectively. Gain change and bias faults are activated when asynchronous transitions t_{1a} and t_{3a} , respectively fire. When all four transitions fire ($f_1 = f_2 = f_3 = f_4 = 1$), we will have

$$U_a(k+1) = (G_a - G_f)U(k) + d. \quad (\text{VII.149})$$

VII.4.2. Actuator Faults Simulation and Analysis: The GPN model developed for the actuator can be integrated into the main net which represents the overall system. This can be done by replacing the transition which models the actuator (transition t_3 of Figure VI.32) by the net in Figure VII.45. This was done for the system we have been modeling in this thesis. This system was simulated, and various faults were introduced. Figure VII.46 shows the plot of three such fault cases.

The first two plots (Figures VII.46 (a) and (b)) represent the case when the actuator's connection to the rest of the system is broken. The next case represent an actuator gain change, which does not cause instability in the system, since the roots of the system

characteristic equations remain within the unit circle as can be seen in Table VII.7. The final two plots (Figures VII.46 (e) and (f)) show a gain change fault of $G_f = -1.2$. This fault makes the overall system unstable, as any actuator gain of greater than two would.

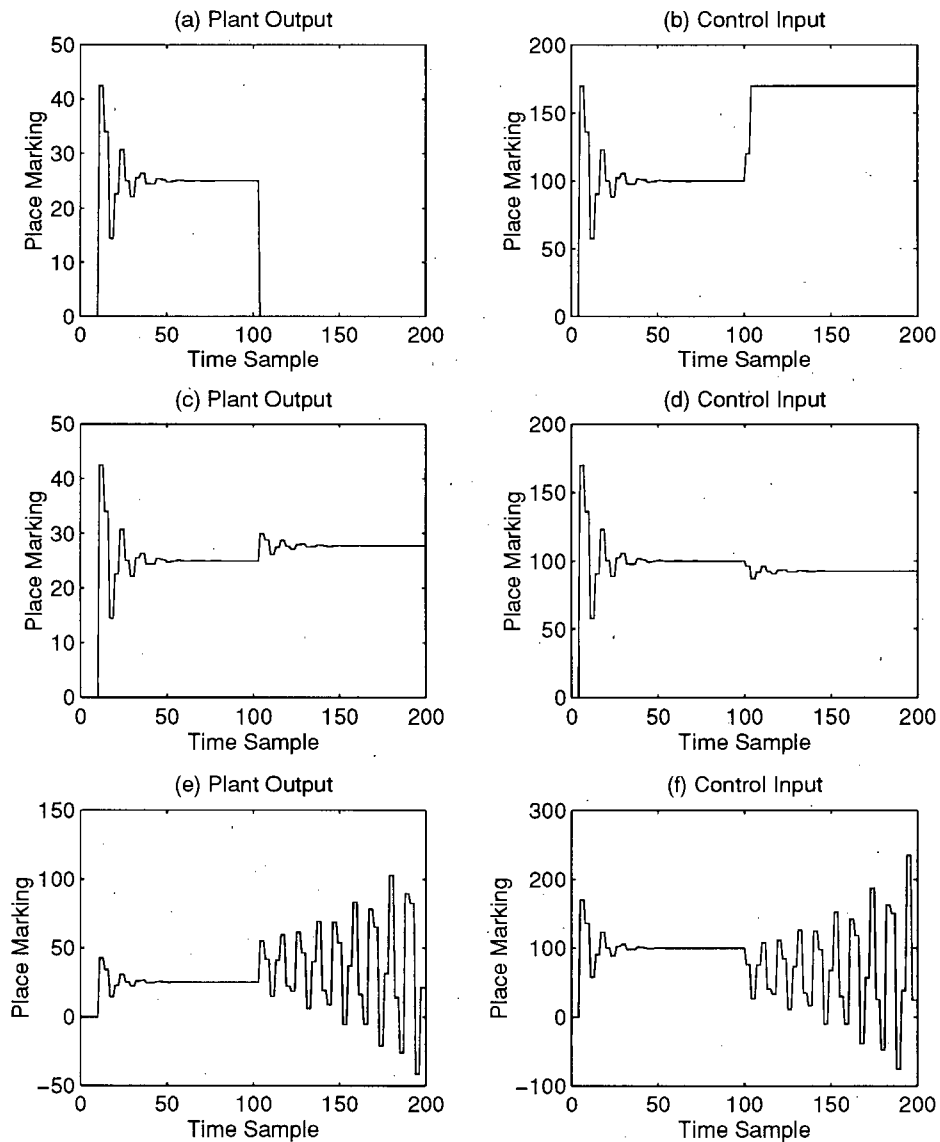


Figure VII.46 Simulation Plots of the Actuator Faults. a-b) Actuator Cutoff $G_f = 1$.
c-d) Actuator Gain Change to $G_f = -0.2$. e-f) Actuator Gain Change to $G_f = -1.2$.

Table VII.7 below summarizes some of the analysis results for the actuator faults. It shows what the effects of various actuator gain changes are, and what gains make the system unstable. These results are found by analyzing the faulty system H matrix.

Fault Type	Actuator Overall Gain	Roots	Condition
No Fault $G_f=0$	1	-0.1+0.7i -0.1-0.7i	Stable, No Error
$G_f=-0.2$	1.2	-0.12+0.7652i -0.12-0.7652i	Stable, Steady State Error
$G_f=-1$	2	-0.2+0.9798i 0.2-0.9798i	Oscillatory
$G_f=-1.2$	2.2	-0.22+1.0255i -0.22-1.0255i	Unstable
Actuator Cut off $G_f=0$	0	No Roots	Stable, no output

Table VII.7 Actuators Faults and Their Analysis Results.

VII.4.3. Disturbances and System Parameter Changes: Other types of faults in control system (such as the hydraulic system under study) which are commonly considered are system disturbances, and parameter changes. Disturbances could be due to change in the load, and can be modeled in exactly the same way as was done with sensor bias faults. They can be taken to originate from asynchronous places.

System parameter changes are reflected in changes in system state space matrices, (A, B, C, and D). These changes are usually very slow acting and occur gradually and over a long period of time. The cause of these can be a change in the environment, such as temperature or pressure. These changes can be modeled by changing the GPN model. The same type of analysis can be applied to the modified models.

VII.5. Fault Conditions Modeled by a GPN

In this final section we list some of the fault conditions which can be manifested in a GPN model. Simulation and analysis of these faults is the first step in detection and identification of them [99, 100]. The detection of faults can be accomplished by comparing the fault (actual) signals with the modeled (estimated) signals. The difference is called the fault residual and is used to detect a fault when it crosses certain threshold. The identification of faults can be done by analyzing the frequency characteristic of the fault residuals (poles and zeros of the signal) [99, 100].

The analysis techniques developed in this dissertation can help in analyzing the system for these fault conditions. A fault detection, identification, and reconfiguration scheme is proposed in Appendix E. This scheme is based on recognition of fault conditions by comparing the actual system outputs with those simulated by a GPN based simulator.

1. **Boundedness:** One of the main symptoms of any control system going unstable is that some of the system parameters go beyond a bound or threshold. By analyzing that if a system is capable of going or has gone out of bound, we can detect many of the faults.
2. **Reachability:** This property is useful in investigating if a faulty state is reachable with the given state of the system. By knowing that a faulty state is reachable, we can be ready to check for it when the system is actually running.
3. **Controllability:** Given a marking which corresponds to a faulty state, is it always possible to steer the system to a safe state in a finite number of samples? This question can be answered by the controllability property. This analysis can be used in recovering from faults and reconfiguring the system.
4. **Liveness:** Another important issue in the design of a fault-tolerant system is avoidance of deadlock states. Liveness property shows whether a system is capable of getting into a deadlock state from a given initial marking or not.
5. **Conservation:** This property can check if the total token content of a set of places should remain constant. When this holds for a non-faulty system, any violation of this

property is a symptom of a fault in the system. There are many examples of such properties in physical systems such as preservation of energy, and the amount of oil or coolant in a motor.

6. **Timing Faults:** Another set of faults which are very essential in our analysis are timing faults. These faults can be due to a delay in the system, or a breakdown. In many systems, tokens in a particular place should be consumed before a new token arrives. Accumulation of tokens is a symptom of a timing fault.

Chapter VIII Conclusions and Future Work

The main objective of this research was to develop a new methodology for the modeling, simulation, and analysis of hybrid systems. Petri nets have proven to be an extremely useful tool for modeling and analyzing discrete-event systems. Linear control system theory has solved many of the problems in the control and identification of dynamic and time-driven systems. Our new modeling methodology, which is called a global Petri net (GPN), successfully combines the capabilities of these two powerful tools into a single tool. Although, many additions have been made to the PN theory and practice over the years (including Continuous and Hybrid PNs), this is the first work, to the best of our knowledge, that allows modeling, simulation, and analysis of discrete-event and discrete-time dynamic systems to be performed within a unified PN framework.

Conventional Petri nets can model only discrete-event (asynchronous) systems. In a system represented by an event-driven model, we know only when an event starts and ends, and have no idea what happens in between events. The other information which is missing in a discrete-event model is the actual (quantitative) value of each signal. All we know is that a signal is generated and available (a condition), but, we have no information about what its value is. To be able to see the dynamics of the system in between events, we must model the process as a time-dependent element in a GPN. The GPN is very general and facilitates modeling of any kind of digital system, including the digitized versions of analog plants, computer hardware and software, and human interactions.

We formally defined the GPN and showed the structural and behavioral differences between the PN and GPN formalisms. Derivation of the GPN from the basic PN and the derivation of the GPN dynamic equations were also given. The resulting structure was very similar to both the Petri net and the state space representation of dynamic systems. This similarity makes it more appealing and intuitive for both Petri net and control communities.

We have also included a few examples, which were used to illustrate the scope of modeling by the GPN. These examples showed how various type of hybrid systems can be modeled by GPN. One example specifically dealt with modeling and simulation of logical gates at both analog and switch levels. These examples showed how easy and intuitive it was to model with the GPN.

We also discussed the structural and behavioral differences between the PN and the GPN models. One of the main differences is in the way markings are presented. Tokens in the conventional PN can only take binary values, resulting in positive integer markings. But in the GPN formulation, markings can have any real number value. The other important difference is the type of arcs which are allowed. There are two types of arcs allowed in the GPN structure. The first one, which is the same as the arcs in the PN model, is called the asynchronous arc. These arcs set the condition for the firing of transitions. The second type, which is exclusive to the GPN, is called the synchronous arc. These arcs do not impose any conditions on the transitions to which they are connected. These two major differences enable the GPN to model hybrid systems.

We have developed some techniques for analyzing the systems modeled by the GPN. These techniques are based either on the construction of a reachability tree or on linear algebra. The properties which can be analyzed by these methods are controllability, boundedness, stability, liveness, and conservation. Theorems regarding proof of these properties are stated and proven. Some of these properties are defined for a sub-class of GPNs to ease the analysis burden. This sub-class is defined in the thesis, and the reasons for reduction in the required analyses are explained. We have developed a theorem which enables us to find the number of hybrid transition matrices for a given net. We have also discussed some other modeling issues such as the GPN modelability and hierarchy.

To assist us in our research, a tool called GPNSAT (Global Petri Net Simulation and Analysis Tool) was developed. The tool's structure and its salient features are described in the thesis. The GPNSAT package was used for all of our PN and GPN modeling, simulation,

and analysis.

A distributed hybrid system was modeled by the GPN methodology. This system consisted of a hydraulic control system, along with its various parts, such as sensors, actuators, controller, and the communication links. The modeling was carried out at different levels of abstraction. At the highest level the system was modeled as a discrete-event system. At the lowest level, where we modeled the details of a hydraulic control system, a hybrid model was used. At each level the system was simulated and analyzed by the GPNSAT.

Fault modeling and analysis for the hybrid system also was performed. Various system level faults were modeled for a hydraulic control system, its sensors, and its actuators. These faults in the GPN model are represented as events which happen asynchronously. Each fault condition is explicitly modeled by a place. Presence of a token in that place indicates a potential fault. The fault happens when the transition enabled by that fault gets fired. The analysis of these faults by the GPN analysis methods showed that these faults can be distinguished from each other successfully in a GPN-based detection and recognition scheme.

The GPN analysis method can be used in model-based schemes for fault detection and recognition. These schemes monitor the operation, detect faults, and initiate the recovery process by comparing the system's actual outputs with the outputs estimated by the use of the system model. We have shown that there exists a single tool which can be used to model both the plant and the computer controlling it. In this fashion, a single detection scheme is sufficient to monitor the entire system and diagnose various forms of faults.

VIII.1. Future Work

There are many different routes that the present research can lead to. One can pursue this work in applying these theories to other real-life examples such as multimedia, manufacturing processes, robotics, and work cell scheduling. Another direction is to work in developing more analysis techniques or refining the ones presented in this dissertation. Here, we provide some pointers to the future directions.

The main limitation of modeling and analysis by GPNs is the growth in the number of states (places) when the system complexity increases. This specially becomes onerous in simulation and reachability tree analysis algorithms. This growth also limits the applicability of this method for on-line detection and identification of faults. Due to this reason only off-line fault analysis was attempted in this thesis. This shortcoming can be overcome when higher speed processors and more efficient algorithms are used.

Some system properties such as liveness and reachability were analyzed only by reachability tree method. There is still a need for development of techniques based on linear algebraic methods to deal with these properties.

One assumption made in developing the analysis methods was that the transition time of all synchronous transitions is the same. However, the GPN simulation program allows the specification of different transitions times. The analysis of hybrid systems with different synchronous transition times would be a very useful and interesting undertaking.

In the construction of reachability trees, we have not specified the transition time. The next version of RTs can take this into consideration. This might reduce the size of the tree, since some of the states may not be reachable when the transitions are timed.

On the implementation side, we do not have a graphical editor for the GPNSAT. There are many PN editors in shareware. These editors can be modified to accept GPN parameters as well. Another area to develop is to optimize the programs written for the GPNSAT. This can help reduce the running time and memory requirements of these programs.

Bibliography

- [1] R. Williams, B. Benhabib, and K. Smith, "A Hybrid Supervisory Control System for Flexible Manufacturing Workcells," in *Proc. of IEEE International Conf. on Robotics and Automation*, pp. 2551–2556, 1994.
- [2] J. A. Stiver and P. Antsaklis, "Modeling and Analysis of Hybrid Control Systems," in *Proc. of the 31st Conf. on Decision and Control, Tuscon, Arizona*, pp. 3748–3751, 1992.
- [3] M. Marsen, G. Balbo, and G. Bruno, "TOPNET: A Tool for the Visual Simulation of Communication Networks," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 9, pp. 1735–1747, 1990.
- [4] R. Valette, M. Courvoisier, and D. Mayeux, *Control of Flexible Production Systems and Petri Nets*, pp. 266–27. in *Application and Theory of Petri Nets, Selected Papers from the 3rd European Workshop*, Springer-Verlag, 1983.
- [5] R. J. Abbott, "Resourceful Systems for Fault Tolerance, Reliability and Safety," *ACM Computing Surveys*, vol. 22, no. 1, pp. 35–68, 1990.
- [6] M. D. Lemmon and P. Antsaklis, "Event Identification in Hybrid Control Systems," in *Proc. of the 32nd Conf. on Decision and Control, San Antonio, Texas*, pp. 2323–2328, 1993.
- [7] P. Peleties and R. DeCarlo, "Analysis of a Hybrid System Using Symbolic Dynamics and Petri Nets," *Automatica*, vol. 30, no. 9, pp. 1421–1427, 1994.
- [8] P. Zave, "An Operational Approach to Requirements Specification for Embedded Systems," *IEEE Trans. on Software Engg.*, vol. 8, no. 3, pp. 250–269, 1982.
- [9] J. R. Connet, E. J. Pasternak, and B. D. Wagner, "Software Defenses in Real-Time Control Systems," in *Proc. of the Fault Tolerant Computing Symposium FTCS-2*, pp. 94–99, 1972.

-
- [10] R. Doraiswami, M. Kaye, and M. Rezai, "Detection and Identification of Sensor, Actuator and Excessive Load Faults," *Proc. of Canadian Conf. on Elect. and Comp. Engg., Montreal*, 1989.
 - [11] J. H. Lala, R. E. Harper, and L. S. Alger, "A Design Approach for Ultrareliable Real-Time Systems," *IEEE Computer Magazine*, pp. 12–22, May 1991.
 - [12] G. Bruno, A. Castella, G. Macario, and M. P. Pescarmona, "Scheduling Hard Real Time Systems Using High-Level Petri Nets," *Application and Theory of Petri Nets 1992, Lecture Notes in Computer Science, Springer-Verlag*, vol. 616, pp. 93–112, 1992.
 - [13] H. Kopetz and W. Merker, "The Architecture of MARS," in *Proc. of Fault Tolerant Computing Symposium FTCS-15*, pp. 274–279, 1985.
 - [14] C. Cao, "Supervisory Control of a Class of Hybrid Dynamic Systems," in *Proc. of 1993 IEEE Midwest Symposium on Circuits and Systems*, pp. 967–970, 1993.
 - [15] Y. Ho, "Performance Evaluation and Perturbation Analysis of Discrete Dynamic Systems," *IEEE Trans. on Automatic Control*, vol. 32, no. 7, pp. 563–572, 1987.
 - [16] M. Heymann, "Concurrency and Discrete Event Control," *IEEE Control Systems Magazine*, pp. 103–112, June 1990.
 - [17] A. Ichikawa and K. Hiraishi, "Analysis and Control of Discrete Event Systems Represented by Petri Nets," *Discrete Event Systems: Models and Applications, IIASA Conf., Sopron, Hungary, Lecture Notes in Control and Inf. Systems, Springer-Verlag*, pp. 115–134, 1987.
 - [18] R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel(Eds.), *Hybrid Systems*. Springer-Verlag, New York, 1993.
 - [19] P. Antsaklis(Eds.), *Hybrid Systems II, Lecture Notes in Computer Science, Vol. 999*. Springer-Verlag, New York, 1995.
 - [20] R. Alur, T. A. Henzinger, and E. D. Sontag(Eds.), *Hybrid Systems III: Verification and Control, Lecture Notes in Computer Science, Vol. 1066*. Springer-Verlag, New York,

1996.

- [21] Z. Manna and A. Pnueli, "Verifying Hybrid Systems," in *Hybrid Systems, Lecture Notes in Computer Science, Springer-Verlag*, vol. 736, pp. 4–35, 1993.
- [22] P. Antsaklis, A. Stiver, and M. Lemmon, "Hybrid Systems Modeling and Autonomous Control Systems," in *Hybrid Systems, Lecture Notes in Computer Science, Springer-Verlag*, vol. 736, pp. 366–391, 1993.
- [23] R. Alur and D. Dill, "The Theory of Timed Automata," in *Real-Time: Theory and Practice, Lecture Notes in Computer Sc., Springer-Verlag*, vol. 600, pp. 45–73, 1991.
- [24] R. L. Grossman and R. Larson, "Viewing Hybrid Systems as Products of Control Systems and Automata," *Proceedings of the 31st Conference on Decision and Control, Tucson, Arizona*, pp. 2953–55, 1992.
- [25] Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, New York, 1992.
- [26] O. Maler, Z. Manna, and A. Pnueli, "From Timed to Hybrid Systems," in *Real-Time: Theory in Practice, Lecture Notes in Computer Science, Springer-Verlag*, vol. 600, pp. 447–484, 1991.
- [27] Y. Zhang, *A Foundation fro the Design and Analysis of Robotic Systems and Behaviors*. PhD thesis, , Ph.D. Thesis, Dept. of Comp. Sc., University of British Columbia, 1994.
- [28] Y. Zhang and A. K. Mackworth, "Will The Robot Do the Right Thing?," in *Proc. Artificial Intelligence 94, Banff, Alberta*, pp. 255–262, May, 1994.
- [29] Y. Zhang and A. K. Mackworth, "Design and Analysis of Embedded Real-Time Systems: An Elevator Case Study," Tech. Rep. 93–4, Dept. of Comp. Sc., University of British Columbia, 1993.
- [30] C. G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis*. Asken Associates Inc. Pub., 1993.

-
- [31] J. Peterson, *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Inc., N.J., USA, 1981.
 - [32] W. Reisig, *Petri Nets: An Introduction*. Springer-Verlag, 1985.
 - [33] W. Brauer, "Carl Adam Petri and Informatics," in *Concurrency and Nets: Advances in Petri Nets*, Springer-Verlag, pp. 13–21, 1987.
 - [34] J. Ghofraniha and M. Rezai, "Dynamic Scheduling of a Job Shop Using a Petri Net Model," in *Proc. of Production and Manufacturing Engg. Conference, Tehran, Iran*, 1993.
 - [35] M. Zhou and F. Dicesare, "Adaptive Design of Petri Net Controllers for Error Recovery in Automated Manufacturing Systems," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 963–973, 1989.
 - [36] M. D. Jeng and F. DiCesare, "A Review of Synthesis Techniques for Petri Nets with Applications to Automated Manufacturing Systems," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 23, no. 1, pp. 310–312, 1993.
 - [37] J. E. Coolahan and N. Roussopolous, "A Time Petri Net Methodology for Specifying Real-Time System Timing Requirements," in *Proc. International Workshop on Timed Petri Nets, Torino, Italy*, pp. 24–31, 1985.
 - [38] C. Ghezzi, D. Mandrioli, S. Morasca, and M. Pezze, "A Unified High-Level Net Formulation for Time-Critical Systems," *IEEE Trans. on Software Engg.*, vol. 17, no. 2, pp. 160–172, 1991.
 - [39] J. Baer, "Modelling Architectural Features with Petri Nets," in *Petri Nets: Applications to Other Models of Concurrency*, vol. Lecture Notes in Computer Sc. Springer-Verlag, no. 255, pp. 258–277, 1986.
 - [40] Y. Shieh, D. Ghosal, P. R. Chintamaneni, and S. K. Tripathi, "Modeling of Heirarchical Distributed Systems with Fault-Tolerance," *IEEE Trans. on Software Engg.*, vol. 16, no. 4, pp. 444–457, 1990.

-
- [41] R. David and H. Alla, "Petri Nets for Modelling of Dynamic Systems—A Survey," *Automatica*, vol. 30, no. 2, pp. 175–202, 1994.
- [42] C. L. Beck and B. Krogh, "Models for Simulation and Discrete Control of Manufacturing Systems," in *Proceedings of Conf. on Robotics and Automation*, pp. 305–310, 1986.
- [43] R. Sreenivas and B. H. Krogh, "On Petri Net Models of Infinite State Supervisors," *IEEE Trans. on Automatic Control*, vol. 37, no. 2, pp. 274–277, 1992.
- [44] A. Giua and F. DiCesare, "Petri Net Structural Analysis for Supervisory Control," *IEEE Trans. on Robotics and Automation*, vol. 10, no. 2, pp. 185–195, 1994.
- [45] D. Gracanin, P. Srinivasan, and K. Valavanis, "Parametrized Petri Nets and Their Application to Planning and Coordination in Intelligent Systems," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 24, no. 10, pp. 1483–1497, 1994.
- [46] S. Ramaswamy, K. Valavanis, and S. Landry, "Modeling, Analysis and Simulation of a Material Handling System with Extended Petri Nets," *Proc. of the 31st Conference on Decision and Control*, pp. 1665–1672, Dec. 1992.
- [47] S. Ramaswamy and K. Valavanis, "Modeling, Analysis and Simulation of Failures in a Material Handling System with Extended Petri Nets," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 24, no. 9, pp. 1358–1373, 1994.
- [48] P. Freedman, "Time, Petri Nets and Robotics," *IEEE Trans. on Robotics and Automation*, vol. 7, no. 4, pp. 417–433, 1991.
- [49] R. David and H. Alla, *Petri Nets and Grafcet Tools for modelling discrete event systems*. Prentice-Hall, 1992.
- [50] G. R. Gao, *A Code Mapping Scheme for Dataflow Software Pipelining*. Kulwer Academic Publishers, 1991.
- [51] E. A. Lee, "Dataflow Programming for Parallel Implementation of Digital Signal Processing Systems," in *Discrete event Systems: Models and Applications, Lecture Notes in Control and Information Systems*, Springer-Verlag, vol. 103, pp. 135–148, 1987.

-
- [52] D. Harel, "On Visual Formalisms," *Communications of the ACM*, vol. 31, no. 5, pp. 514–530, 1988.
- [53] N. Day, "A Model Checker for Statecharts (Linking Case Tools with Formal Methods)," Master's thesis, Dept. of Computer Science, University of British Columbia, 1993.
- [54] R. C. Waters, "System Validation via Constraint Modeling," *ACM SIGPLAN Notices*, vol. 26, no. 8, pp. 27–36, 1991.
- [55] Shapiro, "Validation of VLSI Chip Using Heirarchical Colored Petri Nets ," *Microelectronics and Reliability*, vol. 31, no. 4, pp. 607–625, 1991.
- [56] J. D. Noe, "Nets in Modeling and Simulation," in *Net Theory and Applications, Lecture Notes in Computer Science, Springer-Verlag*, vol. 84, pp. 347–367, 1980.
- [57] K. Jensen, "Coloured Petri Net : A High Level Language for System Design and Analysis," *Advances in Petri Nets 1990, Lecture Notes in Computer Science, Springer-Verlag*, no. 483, pp. 342–416, 1990.
- [58] K. Jensen, *Coloured Petri Nets: Basic Concepts, Analysis Methods, and Practical Use*. Springer-Verlag, 1992.
- [59] B. Berthomieu and M. Diaz, "Modeling and Verification of Time dependent Systems Using Time Petri Nets," *IEEE Trans. on Software Engg.*, vol. 17, no. 3, pp. 259–273, 1991.
- [60] T. Agerwala, "Putting Petri Nets to Work," *IEEE Computer*, pp. 85–94, Dec. 1979.
- [61] K. Jensen, "Computer Tools for Construction, Modification and Analysis of Petri Nets," in *Petri Nets: Applications and Relationships to other Models of Concurrency, Lecture Notes in Computer Science, Springer-Verlag*, vol. 255, Part II, pp. 4–19, 1986.
- [62] F. Feldbrugge, "Petri Net Tool Overview 1989," *Advances in Petri Nets 1989, Lecture Notes in computer Science, Springer-Verlag*, no. 424, pp. 151–178, 1989.
- [63] J. Goutal, "List of Tools Based on Petri Nets at CRIM (Centre de Recherche Informatique de Montreal) @ http://www.crim.ca/Domaines_Services/GL/PETRI/,"

1995.

- [64] "Petri Nets on World Wide Web (WWW) @ <http://www.daimi.aau.dk/~petrinet/>," 1995.
- [65] G. Florin, C. Fraize, and S. Natkin, "Stochastic Petri Nets: Properties, Application and Tools," *Microelectronics and Reliability*, vol. 31, no. 4, pp. 669–697, 1991.
- [66] W. M. Zuberek, "Timed Petri Nets, Definitions, Properties and Applications," *Microelectronics and Reliability*, vol. 31, no. 4, pp. 627–644, 1991.
- [67] J. Sifakis, "Performance Evaluation of Systems Using Nets," *Net Theory and Applications, Proc. of Advanced Course on General Net Theory of Processes and Systems, Lecture Notes in Computer Science, Springer-Verlag*, vol. 84, pp. 307–319, 1980.
- [68] P. Merlin and D. J. Faber, "Recoverability of Communication protocols- Implications of a Theoretical Study," *IEEE Transaction on Communications*, vol. 24, no. 9, pp. 1036–1043, 1976.
- [69] G. Bruno and G. Marchetto, "Process Translatable Petri Nets for Rapid Prototyping of Process Control Systems," *IEEE Trans. on Software Engg.*, vol. 12, no. 2, pp. 346–357, Feb. 1986.
- [70] K. Brand and U. Kopainsky, "Principles and Engineering of Process Control with Petri Nets," *IEEE Trans. on Auto. Control*, vol. 33, no. 2, pp. 138–149, Feb. 1988.
- [71] R. Valette, *Petri Nets and Reliable Real Time Systems*, pp. 222–227. in Application and Theory of Petri Nets, Springer-Verlag, 1982.
- [72] M. R. Zargham and K. Danhof, "Toward a Definition of Fault Analysis for Petri Net Models," *Information Processing Letters*, vol. 34, pp. 299–305, May 1990.
- [73] N. Levenson and J. Stolzy, "Safety Analysis Using Petri Nets," *IEEE Trans. on Software Engg.*, vol. 13, no. 3, pp. 386–397, 1987.
- [74] V. Krasnohaev and L. Krasnobaev, "Application of Petri Nets for Modeling of Detection and Location of Intermittent Faults in Computers," *Automation and Remote Control*, vol. 49, no. 9, pp. 1198–1204, 1989.

-
- [75] T. Murata, "Some Recent Applications of High-Level Petri Nets," in *Proc. of 1991 IEEE International Sym. on Circuits and Systems, Singapore*, pp. 818–821, 1991.
 - [76] J. Prock, "A New Technique for Fault Detection Using Petri Nets," *Automatica*, vol. 27, no. 2, pp. 239–245, 1991.
 - [77] P. Cofrancesco, A. Cristoforetti, M. Villa, R. Scattolini, and D. W. Clarke, "A Workbench for Digital Control Systems," *IEEE Control Systems Magazine*, pp. 102–105, 1991.
 - [78] R. G. Willson and B. H. Krogh, "Petri Net Tools for the Specification and Analysis of Discrete Controllers," *IEEE Trans. on Software Engg.*, vol. 16, no. 1, pp. 39–50, 1990.
 - [79] J. L. Bail, H. Alla, and R. David, "Asymptotic Continuous Petri nets: An Efficient Approximation of Discrete Event Systems," in *Proc. of the 1992 IEEE Inter. Conf. on Robotics and Automation*, pp. 1050–1056, 1992.
 - [80] N. Zerhouni and H. Alla, "Dynamic Analysis of Manufacturing Systems Using Continuous Petri Nets," in *Proc. of the 1990 IEEE Inter. Conf. on Robotics and Automation*, pp. 1070–1075, 1990.
 - [81] A. K. Martin and C. H. Seger, "Discrete Conservative Approximations of Hybrid Systems," Tech. Rep. 93–44, Dept. of Comp. Sc., University of British Columbia, 1994.
 - [82] L. A. Glasser and D. W. Dobberpuhl, *The Design and Analysis of VLSI Circuit*. Addison-Wesley Pub. Co., 1991.
 - [83] W. Reisig, "Combining Petri Nets and Other Formal Methods," *Application and Theory of Petri Nets 1992, Lecture Notes in Computer Science, Springer-Verlag*, vol. 616, pp. 24–44, 1992.
 - [84] K. Lautenbach, "Linear Algebraic Techniques for Place/Transition Nets," *Advances in Petri Nets in Lecture Notes in Comp. Science, Springer-Verlag*, pp. 142–167, 1987.
 - [85] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proc. of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

-
- [86] W. Reisig, "Petri Nets and Algebraic Specifications," *Theoretical Computer Science*, no. 80, pp. 1–34, 1991.
- [87] K. Ogata, *Discrete-Time Control Systems*. Prentice-Hall, N.J., 1987.
- [88] K. S. Tsakalis and P. Ioannou, *Linear Time-Varying Systems: Control and Adaptation*. Prentice-Hall, N.J., 1993.
- [89] N. Viswanadham, Y. Narahari, and T. L. Johnson, "Deadlock Prevention and Deadlock Avoidance in Flexible Manufacturing Systems Using Petri Net Models," *IEEE Trans. on Robotics and Automation*, vol. 6, no. 6, pp. 713–723, 1990.
- [90] N. N. Ivanov, "Algebraic Method of Determining Nonexistence of Deadlock Markings of Petri Nets," *Automation and Remote Control*, vol. 52, no. 17, part 2, pp. 986–989, 1991.
- [91] A. A. Desrochers and R. Y. Al-Jaar, *Application of Petri Nets in Manufacturing Systems*. IEEE Press, New York, 1995.
- [92] "MATLAB User Manual, Version 4.2 by the MathWorks, Inc., USA," March 1994.
- [93] N. Sepehri, *Dynamic Simulation and Control of Teleoperated Heavy-Duty Hydraulic Manipulators*. PhD thesis, Dept. of Mechanical Engg., University of British Columbia, 1990.
- [94] P. D. Lawrence, F. Sassani, N. S. and U. Wallersteiner, and J. Wilson, "Computer-Assisted Control of Excavator-Based Machines," in *1993 International Off-Highway & Powerplant Congress & Exposition, Milwaukee, Wisconsin*, 1993.
- [95] D. McCloy and H. Martin, *Control of Fluid Power: Analysis and Design*. John Wiley & Sons, 1980.
- [96] P. Dransfield, *Hydraulic Control Systems – Design and Analysis of their Dynamics*. Lecture Notes in Control and Information Sciences, Springer-Verlag, 1981.
- [97] R. Valette, J. Cardoso, and D. Dubois, "Monitoring Manufacturing Systems by Means of Petri nets with Imprecise Markings," *Proc. of IEEE Int. Symposium on Intelligent*

Control, pp. 233–237, 1989.

- [98] M. Rezai, M. R. Ito, and P. D. Lawrence, “Modeling and Simulation of Hybrid Control Systems by Global Petri Nets,” in *Proc. of 1995 IEEE International Symposium on Circuits and Systems, Seattle, USA*, vol. II, pp. 908–911, May 1995.
- [99] M. Rezai, M. R. Ito, and P. D. Lawrence, “Petri Net Based Fault Tolerance,” in *Proc. of Iranian Conference on Electrical Engineering ICEE-94, Tehran, Iran*, vol. on Control, pp. 199–208, 1994.
- [100] M. Rezai, P. D. Lawrence, and M. R. Ito, “Analysis of Faults in Hybrid Systems by Global Petri Nets,” in *Proc. of 1995 IEEE International Conference on Systems, Man and Cybernetics, Vancouver, Canada*, 1995.
- [101] M. Rezai, M. Kaye, and R. Doraiswami, “Fault Identification in Digital Control Systems,” in *Proc. of IASTED International Conf. on Control and Modeling, Tehran, Iran*, 1990.
- [102] J. C. Delaat and W. C. Merril, “A Real Time Microcomputer Implementation of Sensor Failure Detection for Turbofan Engines,” *IEEE Control Systems Magazine*, pp. 29–37, June 1990.

Appendix A Modeling of Logic Gates by GPNs

In this appendix we show how logic gates are represented by GPNs. We show the nets for simpler gates. More complicated gates and switches can be built up by putting these gates together. Table A.8 shows the truth table for various logic gates. We will make use of this table in our presentation in this appendix.

A	B	\overline{A}	$A.B$	$\overline{A.B}$	$A + B$	$\overline{A + B}$
0	0	1	0	1	0	1
0	1	1	0	1	1	0
1	0	0	0	1	1	0
1	1	0	1	0	1	0

Table A.8 The Truth Table for Various Digital Logic Gates.

A.1. Inverter Gate

We start with the inverter gate since the GPN model representing this gate is the simplest. The inverter gate inverts the logic sense of a binary signal. If we represent the input to this gate as A (the first column of Table A.8), the output is given by \overline{A} (the third column). This gate can be modeled by two places representing the input and output and a transition representing the logical operation.

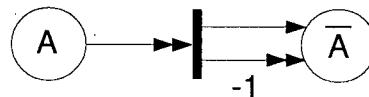


Figure A.47 The Inverter Gate Modeled by a GPN.

Figure A.47 shows the net modeling an inverter. The transition fires irrespective of the marking of place A . The output will just be the opposite of the input.

A.2. AND Gate

The logical operation of the AND gate is presented by the fourth column of Table A.8. The global Petri net model of this gate is given in Figure A.48 and consists of three places and one transition. The transition will fire only when both inputs are equal to one which results in changing the output to one. The output remains zero for all other combinations of inputs.

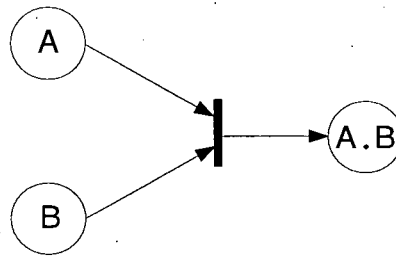


Figure A.48 The AND Gate Modeled by a GPN.

A.3. NAND Gate

The GPN representing the NAND gate can be constructed by putting the first two (AND and Inverter) nets described above together. The NAND gate logical operation is given by the fifth column of Table A.8.

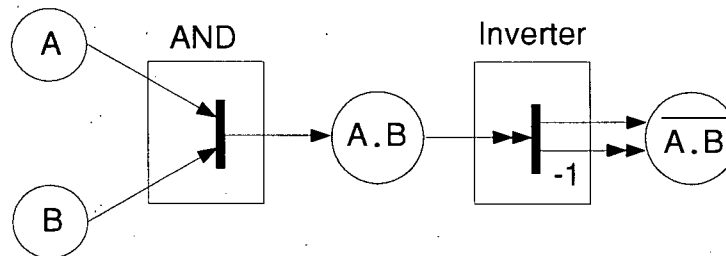


Figure A.49 The NAND Gate Modeled by a GPN.

A.4. NOR Gate

Since modeling a NOR gate by the GPN methodology is simpler than modeling an OR gate, we show the NOR model first. According to DeMorgan's theorem, we can write:

$$\overline{A + B} = \overline{A} \cdot \overline{B} \quad (\text{A.150})$$

where the left hand side is the NOR operation. Therefore, if we invert our inputs A and B and then AND them, the result will be the same if we had done a NOR operation on the inputs. Using this logic, the GPN modeling a NOR gate can be developed as shown in Figure A.50.

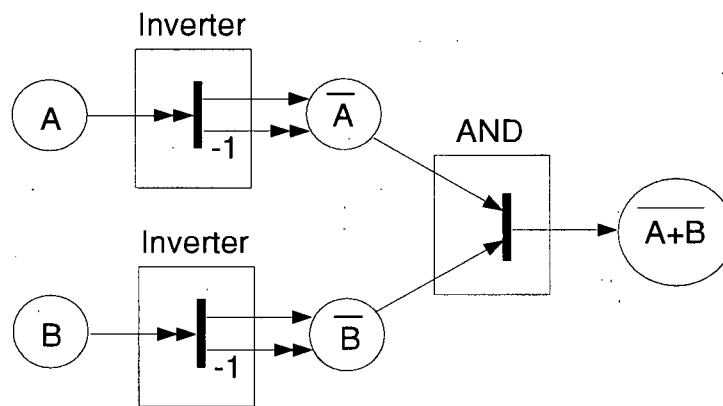


Figure A.50 The OR Gate Modeled by a GPN.

A.5. OR Gate

The GPN model of the OR gate can be constructed by adding an inverter to the output of the NOR gate in Figure A.50.

Appendix B Derivation of the Hybrid Transition Matrix

In this appendix we present the derivation of H matrix expression in terms of the global Petri net parameters such as the weighting matrices and the transition firing vector. H matrix is part of the GPN dynamic equation:

$$M(k+1) = M(k) + H_k M(k) + N f_k. \quad (\text{B.151})$$

H matrix at any instant k is defined by:

$$H_k = -(Diag(Af_k)) + [One(A)Diag(f_k)B]^T. \quad (\text{B.152})$$

We derive the expression for H matrix irrespective of the time instant k by substituting in:

$$H = -(Diag(Af)) + [One(A)Diag(f)B]^T. \quad (\text{B.153})$$

For an m place, n transition GPN, the weighting matrices (A and B) and transition firing vector (f) are given as:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{l1} & a_{l2} & \dots & a_{ln} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1l} \\ b_{21} & b_{22} & \dots & b_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nl} \end{bmatrix} \quad f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}. \quad (\text{B.154})$$

$Diag(Af)$ can be written by matrix multiplication of A and f , and then diagonalizing the resulting vector Af by $Diag$ function:

$$Af = \begin{bmatrix} \sum_{j=1}^n a_{1j} f_j \\ \sum_{j=1}^n a_{2j} f_j \\ \vdots \\ \sum_{j=1}^n a_{lj} f_j \end{bmatrix} \quad Diag(Af) = \begin{bmatrix} \sum_{j=1}^n a_{1j} f_j & 0 & \dots & 0 \\ 0 & \sum_{j=1}^n a_{2j} f_j & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sum_{j=1}^n a_{lj} f_j \end{bmatrix}. \quad (\text{B.155})$$

To find the second element on the right hand side of Equation (IV.73), we need to find the following:

$$One(A) = \begin{bmatrix} o_{11} & o_{12} & \dots & o_{1n} \\ o_{21} & o_{22} & \dots & o_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ o_{l1} & o_{l2} & \dots & o_{ln} \end{bmatrix} \quad Diag(f) = \begin{bmatrix} f_1 & 0 & \dots & 0 \\ 0 & f_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & f_n \end{bmatrix} \quad (B.156)$$

Multiplying these two matrices gives us:

$$One(A)Diag(f) = \begin{bmatrix} o_{11}f_1 & o_{12}f_2 & \dots & o_{1n}f_n \\ o_{21}f_1 & o_{22}f_2 & \dots & o_{2n}f_n \\ \vdots & \vdots & \vdots & \vdots \\ o_{l1}f_1 & o_{l2}f_n & \dots & o_{ln}f_n \end{bmatrix} \quad (B.157)$$

Multiplying the above matrix by B and then taking transpose:

$$[One(A)Diag(f)B] = \begin{bmatrix} \sum_{j=1}^n o_{1j}f_jb_{j1} & \sum_{j=1}^n o_{1j}f_jb_{j2} & \dots & \sum_{j=1}^n o_{1j}f_jb_{jl} \\ \sum_{j=1}^n o_{2j}f_jb_{j1} & \sum_{j=1}^n o_{2j}f_jb_{j2} & \dots & \sum_{j=1}^n o_{2j}f_jb_{jl} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{j=1}^n o_{lj}f_jb_{j1} & \sum_{j=1}^n o_{lj}f_jb_{j2} & \dots & \sum_{j=1}^n o_{lj}f_jb_{jn} \end{bmatrix}, \quad (B.158)$$

$$[One(A)Diag(f)B]^T = \begin{bmatrix} \sum_{j=1}^n o_{1j}f_jb_{j1} & \sum_{j=1}^n o_{2j}f_jb_{j1} & \dots & \sum_{j=1}^n o_{lj}f_jb_{j1} \\ \sum_{j=1}^n o_{1j}f_jb_{j2} & \sum_{j=1}^n o_{2j}f_jb_{j2} & \dots & \sum_{j=1}^n o_{lj}f_jb_{j2} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{j=1}^n o_{1j}f_jb_{jl} & \sum_{j=1}^n o_{2j}f_jb_{jl} & \dots & \sum_{j=1}^n o_{lj}f_jb_{jn} \end{bmatrix} \quad (B.159)$$

Finally substituting from Equations (B.155) and (B.159) in Equation (B.153), we find the expression for H matrix in terms of the net parameters.

$$H = -\text{Diag}(Af) + [\text{One}(A)\text{Diag}(f)B]^T =$$

$$\begin{bmatrix} -\sum_{j=1}^n a_{1j}f_j + \sum_{j=1}^n o_{1j}f_jb_{j1} & \sum_{j=1}^n o_{2j}f_jb_{j1} & \dots & \sum_{j=1}^n o_{lj}f_jb_{j1} \\ \sum_{j=1}^n o_{1j}f_jb_{j2} & -\sum_{j=1}^n a_{2j}f_j + \sum_{j=1}^n o_{2j}f_jb_{j2} & \dots & \sum_{j=1}^n o_{lj}f_jb_{j2} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{j=1}^n o_{1j}f_jb_{jl} & \sum_{j=1}^n o_{2j}f_jb_{jl} & \dots & -\sum_{j=1}^n a_{lj}f_j + \sum_{j=1}^n o_{lj}f_jb_{jn} \end{bmatrix} \quad (\text{B.160})$$

Appendix C Diagonal A Matrix Transformation

The analysis burden for any given global Petri net can be reduced if the net with a non-diagonal A matrix is transformed to a net with a diagonal A matrix which has an equivalent subnet. Two GPNs are said to have equivalent subnets if for a subset of their places, the changes in the markings of those places are exactly the same for any string of events. In this appendix, a procedure for diagonalization of an A matrix is presented through a general case example. Figure C.51 shows a two-place, two-transition GPN with all possible arcs.

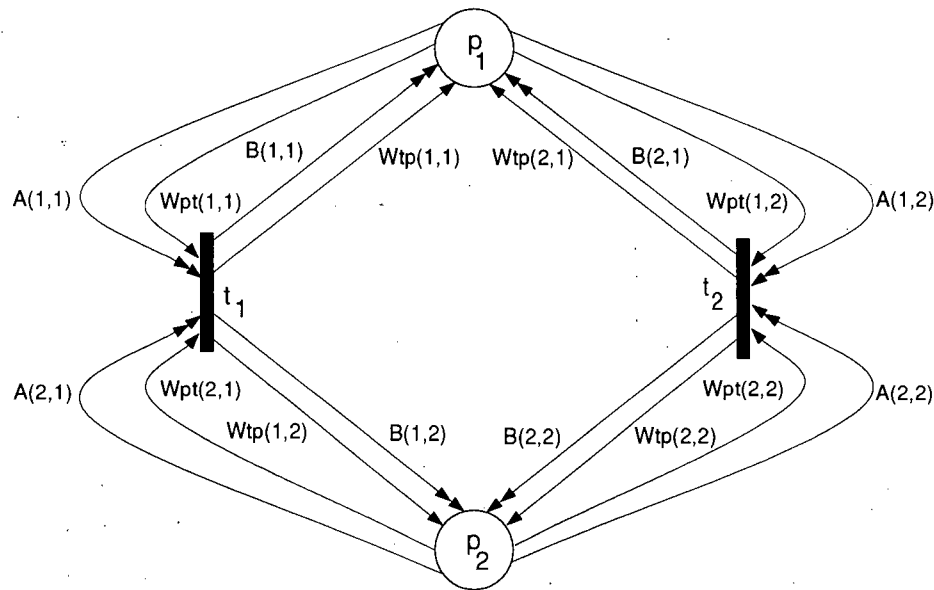


Figure C.51 A Two-Place, Two-Transition GPN with All Possible Arcs

The net (GPN_1) parameters can be written as:

$$GPN_1 = (P, T, A, B, W_{pt}, W_{tp}, M)$$

$$P = \{p_1, p_2\} \quad T = \{t_1, t_2\}$$

$$W_{pt} = \begin{bmatrix} W_{pt}(1,1) & W_{pt}(1,2) \\ W_{pt}(2,1) & W_{pt}(2,2) \end{bmatrix} \quad W_{tp} = \begin{bmatrix} W_{tp}(1,1) & W_{tp}(1,2) \\ W_{tp}(2,1) & W_{tp}(2,2) \end{bmatrix}$$

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \quad M(k) = \begin{bmatrix} M_1(k) \\ M_2(k) \end{bmatrix}.$$

The A matrix above is not a diagonal matrix. We need to transform this matrix into a diagonal one. In the following, we show how this transformation will affect other net parameters. The GPN_1 dynamic equation is given as:

$$M(k+1) = M(k) + H_k M(k) + N f(k).$$

We start by finding the incidence matrix N , and the hybrid matrix H . The incidence matrix N can be written as:

$$N = W_{tp}^T - W_{pt} = \begin{bmatrix} W_{tp}(1,1) - W_{pt}(1,1) & W_{tp}(2,1) - W_{pt}(1,2) \\ W_{tp}(1,2) - W_{pt}(2,1) & W_{tp}(2,2) - W_{pt}(2,2) \end{bmatrix}.$$

The state transition vector is

$$f(k) = f_k = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}.$$

The hybrid matrix H is defined as:

$$H_k = \left(-\text{Diag}(A f_k) + [\text{One}(A) \text{Diag}(f_k) B]^T \right). \quad (\text{C.165})$$

The overall dynamic of this net is governed by this equation. We need to find each element of the above equation and substitute in it.

$$A f_k = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} a_{11} f_1 + a_{12} f_2 \\ a_{21} f_1 + a_{22} f_2 \end{bmatrix}$$

$$\text{Diag}(A f_k) = \text{Diag} \left(\begin{bmatrix} a_{11} f_1 + a_{12} f_2 \\ a_{21} f_1 + a_{22} f_2 \end{bmatrix} \right) = \begin{bmatrix} a_{11} f_1 + a_{12} f_2 & 0 \\ 0 & a_{21} f_1 + a_{22} f_2 \end{bmatrix}$$

$$One(A) = \begin{bmatrix} O(a_{11}) & O(a_{12}) \\ O(a_{21}) & O(a_{22}) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

Since we are assuming all elements of A matrix are nonzero.

$$\begin{aligned} Diag(f_k) &= Diag\left(\begin{bmatrix} f_1 \\ f_2 \end{bmatrix}\right) = \begin{bmatrix} f_1 & 0 \\ 0 & f_2 \end{bmatrix} \\ One(A)Diag(f_k)B &= \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} f_1 & 0 \\ 0 & f_2 \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \\ One(A)Diag(f_k)B &= \begin{bmatrix} f_1 b_{11} + f_2 b_{21} & f_1 b_{12} + f_2 b_{22} \\ f_1 b_{11} + f_2 b_{21} & f_1 b_{12} + f_2 b_{22} \end{bmatrix} \\ [One(A)Diag(f_k)B]^T &= \begin{bmatrix} f_1 b_{11} + f_2 b_{21} & f_1 b_{11} + f_2 b_{21} \\ f_1 b_{12} + f_2 b_{22} & f_1 b_{12} + f_2 b_{22} \end{bmatrix}. \end{aligned}$$

Substituting from above equations in Equation (C.165), we find the hybrid transition matrix to be:

$$H_k = \begin{bmatrix} -a_{11}f_1 - a_{12}f_2 + f_1 b_{11} + f_2 b_{21} & f_1 b_{11} + f_2 b_{21} \\ f_1 b_{12} + f_2 b_{22} & -a_{21}f_1 - a_{22}f_2 + f_1 b_{12} + f_2 b_{22} \end{bmatrix}. \quad (C.173)$$

We want to transform the above net so that we have a new net with a diagonalized A matrix. Let the net which is obtained by this transformation be represented as:

$$GPN'_1 = (P', T', A', B', W'_{pt}, W'_{tp}, M').$$

There are four distinct synchronous output arcs, corresponding to the A matrix elements a_{11}, a_{12}, a_{21} and a_{22} . Therefore, the size of the new A (denoted as A') has to be 4×4 so that we have only one element on each row and each column of the new $A = A'$ matrix. We will have a net which has four places and four transitions.

$$\begin{aligned} P' &= \{p'_1, p'_2, p'_3, p'_4\} = \{p_1, p_2, p_{1d}, p_{2d}\}, \\ T' &= \{t'_1, t'_2, t'_3, t'_4\} = \{t_1, t_2, t_{2d}, t_{1d}\}. \end{aligned}$$

The first two places and transitions are the same as the ones in the previous net and the other two are dummy places and transitions. The net parameters are selected such that the marking of places $p'_3 = p_{1d}$ and $p'_4 = p_{2d}$ are always equal to p_1 and p_2 respectively. The other net parameters are selected as:

$$\begin{aligned}
 W'_{pt} &= \begin{bmatrix} W_{pt}(1,1) & W_{pt}(1,2) & 0 & 0 \\ W_{pt}(2,1) & W_{pt}(2,2) & 0 & 0 \\ 0 & 0 & W_{pt}(1,2) & W_{pt}(1,1) \\ 0 & 0 & W_{pt}(2,2) & W_{pt}(2,1) \end{bmatrix} \\
 W'_{tp} &= \begin{bmatrix} W_{tp}(1,1) & W_{tp}(1,2) & 0 & 0 \\ W_{tp}(2,1) & W_{tp}(2,2) & 0 & 0 \\ 0 & 0 & W_{tp}(2,1) & W_{tp}(1,1) \\ 0 & 0 & W_{tp}(2,2) & W_{tp}(1,2) \end{bmatrix} \\
 A' &= \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ 0 & 0 & a_{12} & 0 \\ 0 & 0 & 0 & a_{21} \end{bmatrix} \quad B' = \begin{bmatrix} b'_{11} & b'_{12} & b'_{13} & b'_{14} \\ b'_{21} & b'_{22} & b'_{23} & b'_{24} \\ b'_{31} & b'_{32} & b'_{33} & b'_{34} \\ b'_{41} & b'_{42} & b'_{43} & b'_{44} \end{bmatrix} \\
 M'(k) &= \begin{bmatrix} M'_1(k) \\ M'_2(k) \\ M'_3(k) \\ M'_4(k) \end{bmatrix} = \begin{bmatrix} M_1(k) \\ M_2(k) \\ M_1(k) \\ M_2(k) \end{bmatrix} \quad f'(k) = f'_k = \begin{bmatrix} f'_1(k) \\ f'_2(k) \\ f'_3(k) \\ f'_4(k) \end{bmatrix} = \begin{bmatrix} f_1(k) \\ f_2(k) \\ f_2(k) \\ f_1(k) \end{bmatrix}
 \end{aligned}$$

All net parameters for GPN'_1 in the above are written in terms their of counterpart parameters in GPN_1 , except B' matrix. We need to find this matrix and show that by this transformation, we do not change the net behavior. The incidence matrix for GPN'_1 can be written as:

$$\begin{aligned}
 N' &= W'^T_{tp} - W'_{pt} = \\
 &\begin{bmatrix} W_{tp}(1,1) - W_{pt}(1,1) & W_{tp}(2,1) - W_{pt}(1,2) & 0 & 0 \\ W_{tp}(1,2) - W_{pt}(2,1) & W_{tp}(2,2) - W_{pt}(2,2) & 0 & 0 \\ 0 & 0 & W_{tp}(2,1) - W_{pt}(1,2) & W_{tp}(1,1) - W_{pt}(1,1) \\ 0 & 0 & W_{tp}(2,2) - W_{pt}(2,2) & W_{tp}(1,2) - W_{pt}(2,1) \end{bmatrix}
 \end{aligned}$$

If we compare Equations (163) and (178), we see that contribution to changes in the marking of equivalent places in GPN_1 and GPN'_1 due to these two equations are the same.

Next we will find the hybrid transition matrix for GPN'_1 .

$$A' f'_k = \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ 0 & 0 & a_{12} & 0 \\ 0 & 0 & 0 & a_{21} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_2 \\ f_1 \end{bmatrix} = \begin{bmatrix} a_{11}f_1 \\ a_{22}f_2 \\ a_{12}f_2 \\ a_{21}f_1 \end{bmatrix}$$

$$Diag(A' f'_k) = Diag \left(\begin{bmatrix} a_{11}f_1 \\ a_{22}f_2 \\ a_{12}f_2 \\ a_{21}f_1 \end{bmatrix} \right) = \begin{bmatrix} a_{11}f_1 & 0 & 0 & 0 \\ 0 & a_{22}f_2 & 0 & 0 \\ 0 & 0 & a_{12}f_2 & 0 \\ 0 & 0 & 0 & a_{21}f_1 \end{bmatrix}$$

$$One(A') = \begin{bmatrix} O(a_{11}) & 0 & 0 & 0 \\ 0 & O(a_{22}) & 0 & 0 \\ 0 & 0 & O(a_{12}) & 0 \\ 0 & 0 & 0 & O(a_{21}) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Diag(f'_k) = \begin{bmatrix} f_1 & 0 & 0 & 0 \\ 0 & f_2 & 0 & 0 \\ 0 & 0 & f_2 & 0 \\ 0 & 0 & 0 & f_1 \end{bmatrix}$$

$$One(A') Diag(f'_k) B' = Diag(f'_k) B' = \begin{bmatrix} f_1 & 0 & 0 & 0 \\ 0 & f_2 & 0 & 0 \\ 0 & 0 & f_2 & 0 \\ 0 & 0 & 0 & f_1 \end{bmatrix} \begin{bmatrix} b'_{11} & b'_{12} & b'_{13} & b'_{14} \\ b'_{21} & b'_{22} & b'_{23} & b'_{24} \\ b'_{31} & b'_{32} & b'_{33} & b'_{34} \\ b'_{41} & b'_{42} & b'_{43} & b'_{44} \end{bmatrix}$$

$$\begin{bmatrix} One(A') Diag(f'_k) B' \end{bmatrix}^T = \begin{bmatrix} Diag(f'_k) B' \end{bmatrix}^T = \begin{bmatrix} f_1 b'_{11} & f_2 b'_{21} & f_2 b'_{31} & f_1 b'_{41} \\ f_1 b'_{12} & f_2 b'_{22} & f_2 b'_{32} & f_1 b'_{42} \\ f_1 b'_{13} & f_2 b'_{23} & f_2 b'_{33} & f_1 b'_{43} \\ f_1 b'_{14} & f_2 b'_{24} & f_2 b'_{34} & f_1 b'_{44} \end{bmatrix}$$

The hybrid transition matrix for GPN'_1 can then be written as:

$$H'_k = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{34} & h_{44} \end{bmatrix}$$

$$H'_k = \begin{bmatrix} f_1 b'_{11} - a_{11} f_1 & f_2 b'_{21} & f_2 b'_{31} & f_1 b'_{41} \\ f_1 b'_{12} & f_2 b'_{22} - a_{22} f_2 & f_2 b'_{32} & f_1 b'_{42} \\ f_1 b'_{13} & f_2 b'_{23} & f_2 b'_{33} - a_{12} f_2 & f_1 b'_{43} \\ f_1 b'_{14} & f_2 b'_{24} & f_2 b'_{34} & f_1 b'_{44} - a_{21} f_1 \end{bmatrix} \quad (C.186)$$

The changes in the marking of the net GPN_1 (before transformation) can be written by reference to Equation C.173. The change in marking of the first place is given as:

$$M_1(k+1) = [-a_{11} f_1 - a_{12} f_2 + f_1 b_{11} + f_2 b_{21}] M_1(k) \\ + [f_1 b_{11} + f_2 b_{21}] M_2(k)$$

Similarly, the changes in the second place markings is given by:

$$M_2(k+1) = [f_1 b_{12} + f_2 b_{22}] M_1(k) \\ + [-a_{21} f_1 - a_{22} f_2 + f_1 b_{12} + f_2 b_{22}] M_2(k)$$

For the transformed net GPN'_1 to be equivalent of the original net (GPN_1), the changes in its marking should be exactly the same as the changes in the marking of the original net.

The changes in the marking of GPN'_1 is governed by the dynamic equation given in Equation (C.186). The changes in the first place marking of GPN' is given as:

$$M'_1(k+1) = [f_1 b'_{11} - a_{11} f_1] M'_1(k) \\ + f_2 b'_{21} M'_2(k) + f_2 b'_{31} M'_3(k) + f_1 b'_{41} M'_4(k).$$

Now, since we are taking the dummy places p_3 and p_4 to be same as p_1 and p_2 respectively, their markings also should be taken to be equivalent. The changes in marking of the first place in GPN'_1 given in Equation (189) can be written as:

$$\begin{aligned} M'_1(k+1) = & \left[-a_{11}f_1 + f_1b'_{11} + f_2b'_{31} \right] M'_1(k) \\ & + \left[f_1b'_{41} + f_2b'_{21} \right] M'_2(k) \end{aligned}$$

Comparing Equations (190) and (190), we can find the values of B' matrix (of GPN'_1) in terms of elements of A and B matrices (of GPN_1) so that the behavior of these two nets becomes equivalent. These values are found to be:

$$\begin{aligned} b'_{11} &= b_{11} & b'_{21} &= b_{21} \\ b'_{31} &= b_{21} - a_{12} & b'_{41} &= b_{11} \end{aligned}$$

Similarly, we can write the changes in marking for the other three places in GPN'_1 and find the values of remaining elements in B' matrix.

$$\begin{aligned} M'_2(k+1) = & \left[f_1b'_{12} + f_2b'_{32} \right] M'_1(k) \\ & + \left[-a_{22}f_2 + f_2b'_{22} + f_1b'_{42} \right] M'_2(k) \end{aligned}$$

$$\begin{aligned} M'_3(k+1) = M'_1(k+1) = & \left[-a_{12}f_2 + f_2b'_{33} + f_1b'_{13} \right] M'_1(k) \\ & + \left[f_2b'_{23} + f_1b'_{41} \right] M'_2(k) \end{aligned}$$

$$\begin{aligned} M'_4(k+1) = M'_2(k+1) = & \left[f_1b'_{14} + f_2b'_{34} \right] M'_1(k) \\ & + \left[-a_{21}f_1 + f_1b'_{41} + f_2b'_{24} \right] M'_2(k) \end{aligned}$$

In this fashion the B' matrix is found to be:

$$B' = \begin{bmatrix} b'_{11} & b'_{12} & b'_{13} & b'_{14} \\ b'_{21} & b'_{22} & b'_{23} & b'_{24} \\ b'_{31} & b'_{32} & b'_{33} & b'_{34} \\ b'_{41} & b'_{42} & b'_{43} & b'_{44} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} & b_{11} - a_{11} & b_{12} \\ b_{21} & b_{22} & b_{21} & b_{22} - a_{22} \\ b_{21} - a_{12} & b_{22} & b_{21} & b_{22} \\ b_{11} & b_{12} - a_{21} & b_{11} & b_{12} \end{bmatrix}$$

Therefore, if we select the second net (GPN'_1) parameters according to the following, any general two-place, two-transition non-diagonal net can be transformed to a diagonal one.

$$GPN'_1 = (P', T', A', B', W'_{pt}, W'_{tp}, M').$$

$$P' = \{p'_1, p'_2, p'_3, p'_4\} = \{p_1, p_2, p_{1d}, p_{2d}\},$$

$$T' = \{t'_1, t'_2, t'_3, t'_4\} = \{t_1, t_2, t_{2d}, t_{1d}\}.$$

$$A' = \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ 0 & 0 & a_{12} & 0 \\ 0 & 0 & 0 & a_{21} \end{bmatrix} \quad B' = \begin{bmatrix} b_{11} & b_{12} & b_{11} - a_{11} & b_{12} \\ b_{21} & b_{22} & b_{21} & b_{22} - a_{22} \\ b_{21} - a_{12} & b_{22} & b_{21} & b_{22} \\ b_{11} & b_{12} - a_{21} & b_{11} & b_{12} \end{bmatrix}$$

$$W'_{pt} = \begin{bmatrix} W_{pt}(1,1) & W_{pt}(1,2) & 0 & 0 \\ W_{pt}(2,1) & W_{pt}(2,2) & 0 & 0 \\ 0 & 0 & W_{pt}(1,2) & W_{pt}(1,1) \\ 0 & 0 & W_{pt}(2,2) & W_{pt}(2,1) \end{bmatrix}$$

$$W'_{tp} = \begin{bmatrix} W_{tp}(1,1) & W_{tp}(1,2) & 0 & 0 \\ W_{tp}(2,1) & W_{tp}(2,2) & 0 & 0 \\ 0 & 0 & W_{tp}(2,1) & W_{tp}(1,1) \\ 0 & 0 & W_{tp}(2,2) & W_{tp}(1,2) \end{bmatrix}$$

$$M'(k) = \begin{bmatrix} M'_1(k) \\ M'_2(k) \\ M'_3(k) \\ M'_4(k) \end{bmatrix} = \begin{bmatrix} M_1(k) \\ M_2(k) \\ M_1(k) \\ M_2(k) \end{bmatrix} \quad f'(k) = f'_k = \begin{bmatrix} f'_1(k) \\ f'_2(k) \\ f'_3(k) \\ f'_4(k) \end{bmatrix} = \begin{bmatrix} f_1(k) \\ f_2(k) \\ f_2(k) \\ f_1(k) \end{bmatrix}$$

Appendix D The Hybrid System Global Petri Net Parameters

In this appendix we include the GPN parameters for the hybrid developed in Chapter VI. These are parameters which were used for all the simulation and analysis carried out in that chapter.

$$GPN = \{P, T, W_{pt}, W_{tp}, A, B, M(0), TT\}, \quad (D.201)$$

where $P=15$, $T=15$ and

$$W_{pt} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (D.202)$$

$$W_{tp} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (D.203)$$

Appendix E Fault Detection, Identification and Reconfiguration (FDIR) Scheme

Figure E.52 shows the configuration for a fault detection, identification and reconfiguration (FDIR) scheme. This scheme can be used for the excavator system along with all of its peripheral and input/output devices. This system consists of a set of sensors in form of joysticks. These are used to enter the desired input as Cartesian coordinates. These are converted to joint angles which in turn are the inputs of the controller. The actuation is provided through pilot valves and hydraulic subsystem. The position of them arm is read and feedback to the system through joint angle sensors.

The fault detection, identification and reconfiguration (FDIR) scheme can written in MATLAB. This scheme receives the system parameters and compares these with the estimated parameters provided by the global Petri net (GPN).

The actions taken by the FDIR scheme will include an alarm with announcement of the fault type and logging of the appropriate data. The system may also be configured in an attempt to reach a safe and acceptable state by changing the controller parameters.

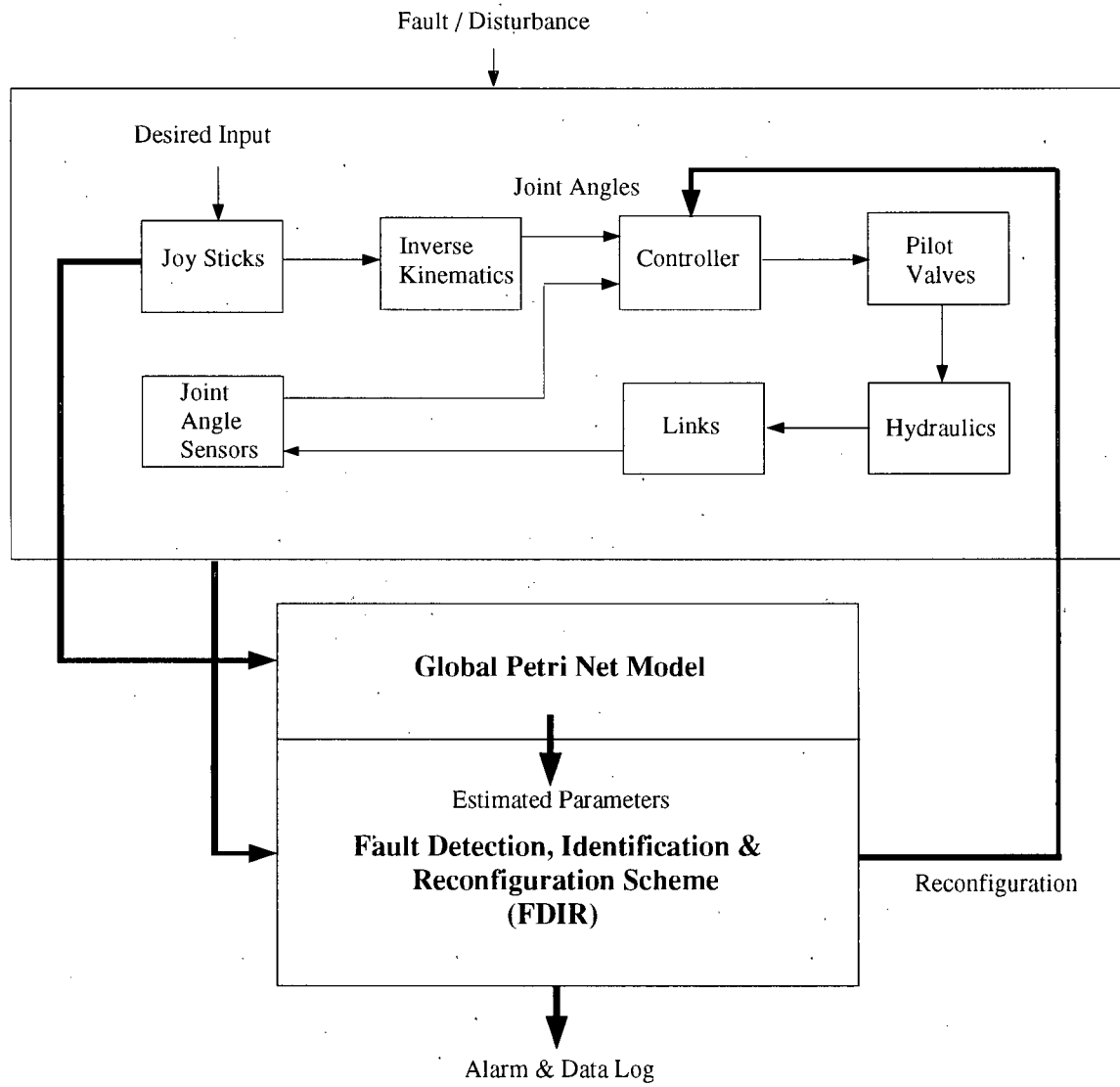


Figure E.52 The FDIR Scheme Block Diagram