

國立交通大學

電控工程研究所

博士論文

建構新型 Petri Nets 模式與其應用

Construction and Applications of Novel Petri Nets Models



研究生：蔡瑞益

指導教授：鄧清政教授

中華民國九十九年七月

建構新型 Petri Nets 模式與其應用

Construction and Applications of Novel Petri Nets Models

研究生：蔡瑞益

Student: Jui-I Tsai

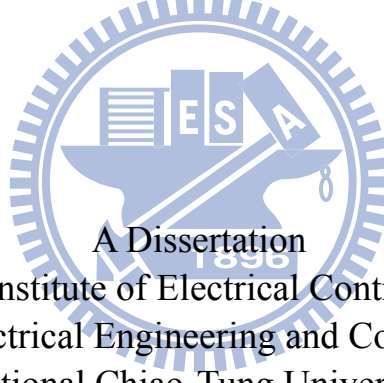
指導教授：鄧清政教授

Advisor: Ching-Cheng Teng

國立交通大學

電控工程研究所

博士論文



A Dissertation

Submitted to Institute of Electrical Control Engineering
College of Electrical Engineering and Computer Science

National Chiao-Tung University

In Partial Fulfillment of the Requirement

For the degree

of Doctor Philosophy

in

Electrical Control Engineering

July 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年七月十九日

建構新型 Petri Nets 模式與其應用

研究生：蔡瑞益

指導教授：鄧清政教授

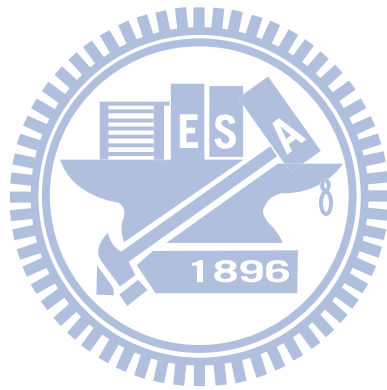
國立交通大學
電控工程研究所
摘要

本論文利用 Petri nets 可撓性，建構出 Logical Petri nets 和 Boolean Petri nets，分別應用到電子領域中的積體電路測試(IC testing)與電機領域中的階梯圖(ladder diagram)測試、診斷和設計領域。

積體電路測試中，Logical Petri nets 是根據真值表(true table)之臨界值(critical value)所建構而成，具有布林代數(Boolean algorithm)和 collapsing fault 性質，使 Petri nets 具有清晰物理觀念。本文所提出前進演算法(forward algorithm)與後退演算法(backward algorithm)，即為了在組合電路(combinational circuit)中，求得測試圖樣(test pattern)、故障點位置(site of fault) 和激發邏輯值(firing logic value)。

在階梯圖上，提出 Boolean Petri nets (BPNs)建構的抽象模式(abstract model)，可直接從 BPNs 的 transition 時序，產生測試事件序列(test event sequence)和提供製作出客製積體電路(application-specific integrated circuits)。此外，在設計可程式控制器方面，也可依系統規格直接建構 BPNs 抽象模式或利用 IDEF0 建構支援 BPNs 抽象模式可完成達到診斷、測試和控制器實現。最後經由一郵票打印程序(stamping process)提供一階梯邏輯圖設計、測試和實

現，證實所提出方法有用的，另由與 simplified Petri net controller (SPNC) 比較，
證實 BPNs 是一簡潔模式。



Construction and Applications of Novel Petri Nets Models

Student: Jui-I Tsai

Advisor: Prof. Ching-Cheng Teng

Institute of Electrical Control Engineering

National Chiao-Tung University

Due to the flexibility of Petri nets (PNs) and their ability to construct various types of clear, readable and suitable plane models, PNs have been recently employed in industrial applications. In this thesis, a Logic Petri nets (LPNs) and a Boolean Petri nets (BPNs) were applied to test, diagnose, and design ladder diagrams (LDs) and to test integrated circuits (ICs).

In IC tests, the proposed LPNs model possesses the properties of a Boolean algorithm including collapsing fault and clear physical concepts because the LPNs model was constructed according to the critical truth table of combinatorial circuits. To solve generated test patterns and determine fired logical values at the site of fault in combinatorial circuits, the proposed approach contains a site of fault and fired logical value reasoning algorithm and a test pattern generation reasoning algorithm.

In existing LDs, the proposed BPNs was used to construct an abstract model that can directly generate test events from the transition sequence of the BPNs and can support the implementation of application-specific integrated circuits (ASIC). Moreover, in the design of programmable logic controllers (PLCs), the proposed abstract BPNs model can be constructed according to the specifications of the system or by employing the integration definition for function modeling (IDEF0). The abstract model developed in this thesis can directly generate a testing event sequence for PLC testing and diagnosing. Finally, an example of a stamping process is provided to illustrate the design, implementation, testing and troubleshooting process.

Comparison of the basic elements (i.e., number of places, transitions, and arcs) of simplified Petri net controller (SPNC) (Lee, 2004) and BPNs are also given to demonstrate the usefulness of this approach.

Key Words: Logic Petri nets, Boolean Petri nets, Petri nets, Abstract model, Ladder diagram, Diagnosis, Testing, Fault model.



ACKNOWLEDGMENT


博士論文的完成得感謝很多人，首先要感謝的是指導教授鄧清政在課業與研究上的指導，以及樹立生活態度典範，在此表達最深的感謝與敬意。感謝口試委員王德勝博士、洪丈力副教授（萬能科技大學電子系）、李慶鴻副教授（元智大學電機系）、以及本系徐保羅教授、梁耀文副教授等師長在論文上指導。

感謝學長李慶鴻博士研究方法、英文寫作技巧與潤稿指導，才能踏出英文寫作之第一步，學長林保童博士協助完成論文臨門一腳，才能趕上口試時程，另要感謝老同事默默支持與長官廖泰杉博士協助、陳永富教授(交通大學電物系)、陳建人博士(精儀中心主任)支持，更要感謝師長林心宇教授在 Petri nets 和黃錫瑜教授（清華大學電機系）在 IC Testing 啓蒙教導，也要感謝投稿期間編審諸多包容協助，尤其 Meng-Chu Zhou 在建構 Petri nets model 建議，另學長李俊賢博士論文相助諸多與陳炯偉中醫師身體調理。謝謝系辦陳英芝和林滿足在事務上協助。

謹將此論文獻給我敬愛的先父 蔡金水先生，母親 蔡許玉女士、娘子章文芳和姊姊哥哥弟弟，因為有您們的支持與關懷，才能夠無後顧之憂，往前邁進。

感謝所有曾經幫助過我與默默祝福我的朋友，感謝您們。

TABLE OF CONTENTS

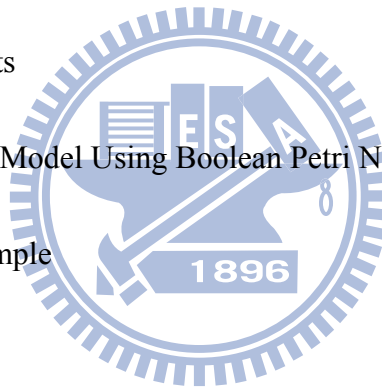
| | Page |
|--|-------------|
| ABSTRACT (CHINESE) | i |
| ABSTRACT (ENGLISH) | iii |
| ACKNOWLEDGMENT | v |
| TABLE OF CONTENTS | vi |
| LIST OF TABLES | ix |
| LIST OF FIGURES | xi |
|  | |
| <i>CHAPTER 1</i> | |
| INTRODUCTION | 1 |
| 1.1. General Review | 2 |
| 1.2. Problem Statement | 4 |
| 1.3. Proposed Approach | 6 |
| 1.4. Organization of Thesis | 8 |
| <i>CHAPTER 2</i> | |
| TEST GENERATION AND FAULT IDENTIFICATION IN COMBINATIONAL CIRCUITS USING LOGIC PETRI NETS | 10 |
| 2.1. The Model and Properties of Logic Petri nets | 10 |

| | |
|---|----|
| 2.2. A Fault Logic Reasoning Algorithm | |
| for Sites and Fired Logic Value | 14 |
| 2.3. Forward and Backward Reasoning Algorithm | 15 |
| 2.4. Summary | 21 |

CHAPTER 3

CONSTRUCTING AN ABSTRACT MODEL FOR LADDER DIAGRAM DIAGNOSIS USING PETRI NETS **22**

| | |
|---|----|
| 3.1. Boolean Petri nets | 23 |
| 3.2. Ladder Diagram Model Using Boolean Petri Net | 27 |
| 3.3. Application Example | 31 |
| 3.4. Summary | 40 |



CHAPTER 4

AN ASIC IMPLEMENTATION FOR TESTING OF A LADDER DIAGRAM USING A BOOLEAN PETRI NET **41**

| | |
|---|----|
| 4.1. Boolean Petri Net and Ladder Diagram Model | 41 |
| 4.2. Testing and Troubleshooting of ladder Diagrams | 47 |
| 4.3. Application Example | 53 |

| | |
|--------------|----|
| 4.4. Summary | 57 |
|--------------|----|

CHAPTER 5

DESIGN FOR TESTING AND IMPLEMENTATION OF LOGIC CONTROLLERS USING BOOLEAN PETRI NETS 59

| | |
|-------------------------|----|
| 5.1. Boolean Petri nets | 60 |
|-------------------------|----|

| | |
|--|----|
| 5.2. Constructing the Boolean Petri Net and Implementation | 62 |
|--|----|

| | |
|----------------------------------|----|
| 5.3. Testing and Troubleshooting | 65 |
|----------------------------------|----|

| | |
|-------------------------------------|----|
| 5.4. An example of stamping Process | 68 |
|-------------------------------------|----|

| | |
|--------------|----|
| 5.5. Summary | 76 |
|--------------|----|

CHAPTER 6

CONCLUSIONS 77

| | |
|-------------------------------|----|
| 6.1. Summary of Contributions | 77 |
|-------------------------------|----|

| | |
|----------------------|----|
| 6.2. Future Research | 78 |
|----------------------|----|

REFERENCES 80

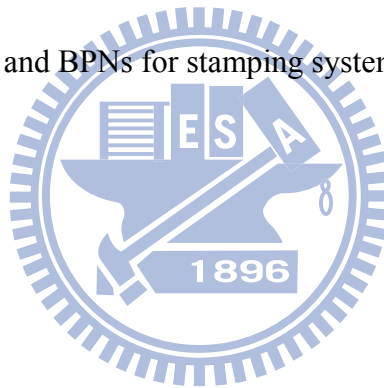
VITA 84

PUBLICATION LIST 85

LIST OF TABLES

| | |
|--|----|
| 2.1. True table and logic Petri nets model | 12 |
| 2.2. Boolean algorithm respect to LPNs | 13 |
| 2.3. Fault Collapsing respond to LPNs | 13 |
| 2.4. The transition state for forward and backward of places | 15 |
| 2.5. Immediate reachability set, reachability set, immediate backward incidence set and backward incidence set for each place P_i . | 20 |
| 2.6. Set of adjacent place Ap_{jk} for each place P_j . | 20 |
| 3.1. Some LDs models and their corresponding BPNs models. | 28 |
| 3.2. Composite and Decomposite of Boolean Petri nets. | 29 |
| 3.3. The descriptions of symbol. | 33 |
| 4.1. Simplified BPNs. | 44 |
| 4.2. Some LDs modules and their corresponding BE and BPNs models. | 45 |
| 4.3. Composites and decomposites of Boolean Petri nets. | 46 |
| 4.4. True table of a simple ladder diagram circuit | 49 |
| 4.5. Test event sequence. | 51 |
| 4.6. The HDL code of a basic ladder diagram | 52 |

| | |
|--|----|
| 4.7. Descriptions of symbols | 53 |
| 4.8. Test event sequence and troubleshooting of motor starting LDs | 56 |
| 4.9. The HDL code of motor start action. | 57 |
| 5.1. The information from the BPNs to PLC code. | 65 |
| 5.2. True table of a simple ladder diagram circuit. | 66 |
| 5.3. Test event sequence and troubleshooting. | 68 |
| 5.4. Test event sequence and troubleshooting of stamping process. | 74 |
| 5.5. Comparison of SPNC and BPNs for stamping system | 74 |



LIST OF FIGURES

| | |
|---|----|
| 2.1. Logic Petri nets model for NOT gate. | 12 |
| 2.2. (a) Combinational circuit, (b)LPNs circuit | 14 |
| 2.3. (a) Petri nets for immediate reachability, reachability, immediate backward incidence, and backward incidence sets, (b) Petri Net for adjacent place. | 16 |
| 2.4. (a) Combinational circuit, (b) LPNs equivalent circuit. | 18 |
| 2.5. The comparison between LPNs model and traditional method: (a) A sample good circuit; (b) A faulty circuit; (c) The search graph for locating the fault; (d) A faulty circuit of LPNs; (e) The search graph for locating the fault of LPNs | 25 |
| 3.1. Proposed hierarchical control. | 22 |
| 3.2. (a) An example Petri nets, (b) A token moving from A to B in Fig. 3.2. (a) After t_i fire. | 26 |
| 3.3. A simple example. | 26 |

| | |
|---|----|
| 3.4. The reduction result of Fig.3.3. | 26 |
| 3.5. (a) A simple example and (b) the composite result of Fig.3.5. (a). | 27 |
| 3.6. (a) A LDs of possessed fault and (b) A Petri model of possessed fault. | 30 |
| 3.7. Control circuit of a Y- Δ starting motor. | 32 |
| 3.8. BPNs model of a LDs controller. | 33 |
| 3.9. Equivalent diagram of Fig. 3.8. | 33 |
| 3.10. Abstract model of Fig.3.9. | 33 |
| 3.11. The reachability tree of the proposed BPNs. | 35 |
| 3.12. Petri nets model: (a) with fault f_1 , (b) with fault f_2 , (c) with fault f_3 in Fig. 3.7. | 37 |
| 3.13. Simulated fault free model and fault model. | 37 |
| 3.14. The faulty area in case 1. | 39 |
| 4.1. Framework of LDs functional testing | 41 |
| 4.2. (a) An example Petri nets, (b) A token moving from A to B in Fig. 4.2. (a) After t_i fire. | 42 |
| 4.3. (a) A basic ladder diagram circuit, (b) corresponding to <i>AND</i> logic, (c) Corresponding to the BPNs model. | 50 |
| 4.4. (a) Self-hold of a ladder diagram, (b) Corresponding to BPNs model and | |

| | |
|---|----|
| (c) Simplified BPNs model. | 51 |
| 4.5. Control circuit of a Y- Δ starting motor. | 53 |
| 4.6. (a) BPNs model of a LDs controller, (b) equivalent diagram of Fig.5 (a), (c)Simplified model of Fig. 5(b), (d) ASIC diagram, (e) Simulation result. | 56 |
| 5.1. Implementation scheme of PN-based controllers. | 59 |
| 5.2. Proposed hierarchical control. | 60 |
| 5.3. (a) Filling tank, (b) BPNs model of filling tank, (c) Material flows of IDEF0 and (d) Information flows of IDEF0. | 63 |
| 5.4. The IDEF0 scheme | 64 |
| 5.5. BPNs model, (b) Corresponding to a self-hold of a ladder diagram | 67 |
| 5.6. (a) Structure diagram of the stamping system (from Lee 2005), (b) Corresponded BPNs of the stamping system, (c) Corresponded BPNs of add safe designed stamping system, (d) Mapped LLD. | 71 |
| 5.7. Abstract BPNs model of stamping system. | 72 |
| 5.8. Corresponding SPNC of the stamping system (from Lee, 2004) | 74 |
| 5.9. LLD implementation of stamping system (from Lee, 2004) | 75 |

Chapter 1

Introduction

In industry, programmable logic controllers (PLCs) are often programmed with ladder diagrams (LDs), and the overall design and testing of the LDs are based on operator experience. Recently, Petri nets (PNs) have become popular tools for the design and implementation of logic controllers. Compared to LDs, Petri nets establish a system controller for various PLCs in a more flexible and understandable manner. Previous studies on the design of LDs and Petri nets have focused on the characteristics of both models and the conversion between LDs and Petri nets for the analysis, validation, design, and implementation of PLCs (Peng, 2004).

The objective of this thesis was to achieve the following goals:

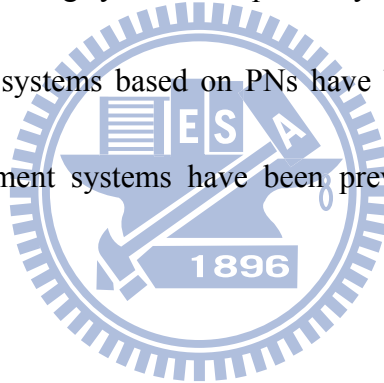
- 1) To develop a novel Petri nets for the construction of an abstract model of a logic controller.
- 2) To develop a testing and diagnosis procedure for existing logic controllers.
- 3) To develop a clear approach for the design of logic controllers.

The models and approaches developed in the thesis were applied to $Y-\Delta$ starting

motor and stamping processes.

1.1. General Review

Petri nets (PNs) theory was developed in 1962 by Carl A. Petri (1962). PNs are a theoretical, visual and graphical tool for the modeling, analysis, validation and control of discrete event systems. Moreover, PNs are excellent tools for modeling asynchronous concurrent systems. Due to the flexibility of PNs, they can be used to construct models of various systems, including information flow management, computer systems, manufacturing systems and power systems (Lan, 2009), (LO, 1997). Recently, video streaming systems based on PNs have been developed (Hu, 2009), and supply chain management systems have been previously constructed (Dotoli, 2009).



1.1.1. Development of an abstract model

Modeling plays an essential role in the design, fabrication, and testing of a digital system (Abramovici, 1990). Moreover, many techniques have been developed for the identification of faults in combinational circuits (Looney, 1987), (David, 1995); however, most of these methods are based on functional modeling at the logic level. A Logic Petri nets (LPNs) model of combinational circuits is alternation modeling approach; thus, the LPNs model can transfer logic circuit problems into a local,

adjacent place, resulting in a transition relational problem.

Traditionally, ladder diagrams (LDs) have been applied to programmable logic controllers. For instance, Jackman et al. (1995) proposed a conceptual model and working equation for converting relay ladder logic into a PNs model. Lee et al. (2000) presented a method for obtaining an augmented PNs from a LDs, and applied the Petri nets state equation to validate the corresponding flow mechanism in the PNs. Venkatesh et al. (1994) and Peng et al. (2004) modeled the conversion of a LDs contact to a PNs place, and increased the rate of virtual transitions. Lee et al (2004) modeled the conversion of a LDs connect to a PNs transition, and increased the position in the resulting PNs. However, the total number of nodes and links in the generated Petri nets were relatively high, and the complexity of the system increased. To reduce the complexity and increase the readability of the sequence control system in the construction of an abstract model, a Boolean Petri nets that introduces composite transitions, composite places, and relevant states was employed in this thesis.

1.1.2. Diagnosis and testing of the ladder diagram

In industry, LDs are used to program logic controllers. The LDs allow plant maintenance personnel to troubleshoot and maintain the system (Peng, 2001);

however, the overall troubleshooting method is often experience-based. Given the complexity of control programs and manufacturing systems, verification is time consuming, and the systems are difficult to troubleshoot. The proposed BPNs are the first model to introduce the concept of integrated circuit testing for solving experience-based testing and troubleshooting problems in sequence controllers in manufacturing systems.

1.1.3. The design of the logic controller

In industry, programmable logic controllers (PLCs) are often programmed using LDs, and the testing of PLCs is often experience-based. Moreover, verification is typically conducted through experiments or simulation. PNs focus on the design and implementation of logic controllers; however, tools for the design, implementation (Uzam et al. 1998), (Lee et al., 2005], and diagnosis of logic controllers are required. To achieve this goal, the Boolean Petri nets was employed, which supplies an integrated design tool for sequence control systems.

1.2. Problem Statement

LDs are a common method used to control discrete events in the programmable controller of an automated system. Researchers are constantly pursuing integrated

tools that overcome the current limitations of LDs. The objectives of these tools are to control the automated system, and to analyze, evaluate, and simulate the sequence control system. Over the past several decades, PNs have emerged as an important tool for the production of integrated solutions for the modeling, analysis, simulation, and control of automated systems. The construction of abstract models in existing circuits or specifications is not straightforward; thus, different types of PN-based models have been proposed and applied to diagnosis and test automated systems. However, all novel PNs must contain the following requirements:



1.2.1. An alternation model for the testing of combinational circuit

In practice, many techniques for the identification of faults and test patterns have been proposed. However, most of these methods have been developed through functional modeling at the logic level.

1.2.2. An abstract model for existing LDs

Although LDs have been converted to PNs for analysis and validation (Peng, 2004), PNs are usually more complex, and the construction of abstract models of LDs is not straightforward.

1.2.3. Systematic testing approaches for existing LDs

Systematic LDs testing is important; however, experience-based testing is still relatively common.

1.2.4. A sequence controller design for the testing, diagnosis and implementation of programmable controllers

Although PLC engineers prefer to use LLD for the implementation of programmable controllers, and straightforward designs have been constructed with LLD models, these designs only focus on implementation while testing and diagnosis of the system are neglected.

1.3. The proposed approach

To overcome the aforementioned problems, the following approaches are proposed in this thesis:



1.3.1. Improved logic fault efficiency

The transitions of the PNs are modified according to the critical truth table to produce a model called the Logic Petri Nets (LPNs). The LPNs model can transfer a complex circuit problem into a local, adjacent place and a transition relational problem, which simplifies the identification of the fault sites and fired logical values. The LPNs model possesses the properties of a Boolean

algorithm, including collapsing fault with clear physical concepts, fast calculation speed, and high veracity.

1.3.2. Constructing an abstract model of the ladder diagram

In this thesis, a Boolean Petri nets (BPNs) is introduced, and the approach used to transfer a LDs to a BPNs converts normal open (NO) and normal close (NC) contacts in the LDs into PNs transitions and converts devices (e.g., relay coils) in the LDs into PNs places. Moreover, the BPNs introduce the concepts of composite transitions, composite places, and relevant states to reduce the complexity of the system and to increase the readability of PNs in the construction of abstract models. The abstract model can be applied to the analysis and diagnosis of local controllers for the support of network-based monitoring and the supervision of automated systems.

1.3.3. Systematic testing of sequence controllers.

In this thesis, the concept of integrated circuit testing was introduced for the construction of a fault-free model and the generation of a test events sequence for LDs based on a BPNs. The fault-free BPNs model can directly convert hardware description languages (HDLs) and can implement application-specific integrated

circuits (ASICs). The comparison of the response of a fault-free circuit (i.e., ASIC circuit) and a fault circuit (i.e., LDs circuit) leads to the detection of fault occurrence, which aids in troubleshooting.

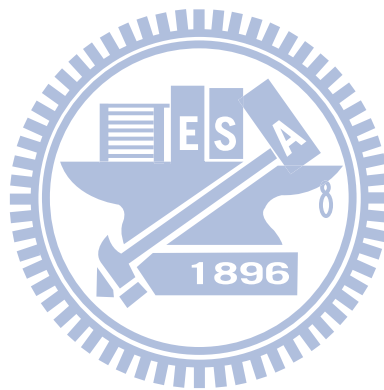
1.3.4. Design for the testing, diagnosis and implementation of the sequence controller

In this thesis, a design scheme for the testing, diagnosis and implementation of logic controllers based on BPNs are proposed. The abstract BPNs model can be constructed according to the specifications of the system or by employing the integration definition for function modeling (IDEF0). The abstract model can directly generate a testing event sequence for the testing and diagnosis of PLCs. Moreover, the model can also support network-based monitoring and supervision, and can be directly mapped into relay ladder logic (RLL), ladder logic diagrams (LLDs), or hard description language (HDL) for implementation in a system controller.

1.4. Organization of the thesis

This thesis is organized as follows: in Chapter 2, the LPNs model used to generate the testing pattern of the combinational circuit is introduced. Chapter 3 introduces the

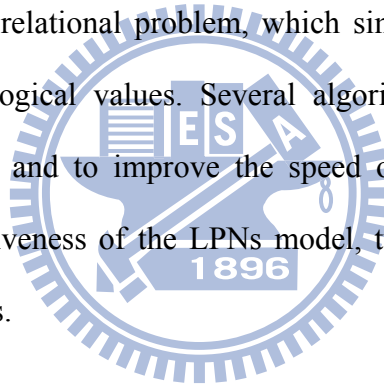
BPNs model used to construct the abstract model and to diagnosis the LDs. In Chapter 4, the BPNs model is used to generate testing event sequences and to implement ASIC for LDs testing. In Chapter 5, an integrated IDEF0/BPN/PLC approach for the testing, diagnosis and implementation of the sequence controller design is proposed. Finally, conclusions and recommendations for further research are provided in Chapter 6.



CHAPTER 2

Test Generation and Fault Identification in Combinational Circuits Using Logic Petri Nets

PNs are an excellent tool for modeling asynchronous concurrent systems. In this chapter, the proposed PNs are modified to solve test generations and sites of fired values based on the truth table of combinational circuits. To develop the LPNs, critical truth tables were embedded into the transitions of the PNs. Thus, the LPNs model can transfer a complex circuit problem into a local, adjacent place and a transition relational problem, which simplifies the identification of fault sites and fired logical values. Several algorithms were implemented to obtain the test pattern and to improve the speed of calculation. Moreover, to demonstrate the effectiveness of the LPNs model, two different processes were modeled with the LPNs.



2.1. The Model and Properties of LPNs

The purpose of the development of LPNs model is that the LPNs model holds clear logical property in IC testing. Firstly, the simplest way to represent a combinational circuit is by its truth table. Assuming binary input variable, a circuit realizing a function $X(x_1, x_2, \dots, x_n)$ of n variables requires a table with 2^n entries. The data structure representing a truth table is usually an array U of dimension 2^n . We arrange the input combinations in their increasing binary order. Then, we obtain $U(0) = X(0,0,\dots,0)$, $U(1) = X(0,0,\dots,1)$, ..., $U(2^n - 1) = X(1,1,\dots,1)$. The truth table can be divided into critical and no-critical part. For AND gate, the corresponding critical value is $x_1 \in 1$, $x_2 \in 1$, and $U(2^2 - 1) = X(1,1) = 1$. That is, if $X(x_1, x_2) = 1$ then $x_1 \in 1$ and

$x_2 \in 1$; no-critical value of AND gate is $x_1 \notin 1$ or $x_2 \notin 1$ and $X(x_1, x_2) \neq 1$, i.e., if $X(x_1, x_2) \neq 1$ then $\{x_1, x_2\} \not\subset \{1, 1\}$.

In this section, we embed the critical value of truth table into transition of PNs to develop LPNs model. This special transition is called “logic transition”. Table 2.1 describes the LPNs model corresponding to the truth table. Clearly, the LPNs model is matched properties of Boolean algorithm and fault collapsing. Based on the embed critical value of truth table in LPNs model, the Boolean algorithm and fault collapsing in LPNs representation are shown in Table 2.2 and Table 2.3.

In general used representation, the LPNs model structure can be defined as follows:

$$LPN = (P, T, D, I, O, i, o, f, b, \alpha, m_0)$$

Where

$p = \{p_1, p_2, \dots, p_m\}$: finite set of places,

$T = \{t_1, t_2, \dots, t_n\}$: finite set of logic transitions by critical value of truth table,

$D = \{d_1, d_2, \dots, d_m\}$: finite set of propositions,

$$P \cap T \cap D = \Phi,$$

$$|P| = |D|,$$

$I : T \rightarrow P^\circ$: an input function (a mapping from transitions to bags of places),

$O : T \rightarrow P^\circ$: an output function (a mapping from transitions to bags of places),

$i : T \rightarrow \{\bullet, \circ\}$: logical value of a input transitions,

$o : T \rightarrow \{\bullet, \circ\}$: logical value of a output transitions,

$f : p \rightarrow i(t_j)$: input logical value of a transitions (a forward mapping from place p to input critical value $i(t_j)$),

$b : p \rightarrow o(t_j)$: output logical value of a transitions (a backward mapping from place p to output critical value $o(t_j)$),

$\alpha : P \rightarrow \{\bullet, \circ\}$: logic value of place (a mapping from place to logic value, $\alpha(p_i) = \{\bullet, \circ\}$,

i.e., \bullet denotes logic 1 and \circ denotes logic 0).

$m_0 : P \rightarrow \{-, \bullet, \circ\}$: Initial mark.

Example 1: Herein, the description of LPNs model for NOT gate is introduced, as the following Fig. 2.1.

p_1, p_2 : place, t_k : transition, d_1 : stuck-at-1, $I(t_k) = p_1$, $O(t_k) = p_2$, $i(t_k) = \bullet$, $o(t_k) = \circ$,
 $f : p_1 \rightarrow i(t_k) = \bullet$, $b : p_2 \rightarrow o(t_k) = \circ$, $\alpha : P_1 \rightarrow \bullet$.

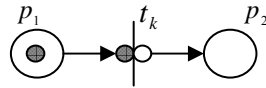


Fig. 2.1. Logic Petri Net model for NOT gate.

Table 2.1: Truth table and Logic Petri Nets model

| TYPE | True Table | Logic gate | Logic Petri Nets | | | | | | | | | | | | | | | |
|---|---|------------|------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|--|--|
| NOT gate | <table border="1"> <tr> <td>A</td> <td>C</td> </tr> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </table> | A | C | 0 | 1 | 1 | 0 | | | | | | | | | | | |
| | A | C | | | | | | | | | | | | | | | | |
| 0 | 1 | | | | | | | | | | | | | | | | | |
| 1 | 0 | | | | | | | | | | | | | | | | | |
| <table border="1"> <tr> <td>A</td> <td>B</td> <td>C</td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </table> | A | B | C | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | | | |
| A | B | C | | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | | | | | | | | | | | | | | | | |
| 0 | 1 | 1 | | | | | | | | | | | | | | | | |
| 1 | 0 | 1 | | | | | | | | | | | | | | | | |
| 1 | 1 | 1 | | | | | | | | | | | | | | | | |
| OR gate | <table border="1"> <tr> <td>A</td> <td>B</td> <td>C</td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </table> | A | B | C | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | | |
| | A | B | C | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | | | | | | | | | | | | | | | | |
| 0 | 1 | 1 | | | | | | | | | | | | | | | | |
| 1 | 0 | 1 | | | | | | | | | | | | | | | | |
| 1 | 1 | 1 | | | | | | | | | | | | | | | | |
| <table border="1"> <tr> <td>A</td> <td>B</td> <td>C</td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </table> | A | B | C | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | | |
| A | B | C | | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | | | | | | | | | | | | | | | | |
| 0 | 1 | 0 | | | | | | | | | | | | | | | | |
| 1 | 0 | 0 | | | | | | | | | | | | | | | | |
| 1 | 1 | 1 | | | | | | | | | | | | | | | | |
| AND gate | <table border="1"> <tr> <td>A</td> <td>B</td> <td>C</td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </table> | A | B | C | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | |
| | A | B | C | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | | | | | | | | | | | | | | | | |
| 0 | 1 | 0 | | | | | | | | | | | | | | | | |
| 1 | 0 | 0 | | | | | | | | | | | | | | | | |
| 1 | 1 | 1 | | | | | | | | | | | | | | | | |
| <table border="1"> <tr> <td>A</td> <td>B</td> <td>C</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </table> | A | B | C | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | | | |
| A | B | C | | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | | | | | | | | | | | | | | | | |
| 0 | 1 | 0 | | | | | | | | | | | | | | | | |
| 1 | 0 | 0 | | | | | | | | | | | | | | | | |
| 1 | 1 | 0 | | | | | | | | | | | | | | | | |
| NOR gate | <table border="1"> <tr> <td>A</td> <td>B</td> <td>C</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </table> | A | B | C | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | | |
| | A | B | C | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | | | | | | | | | | | | | | | | |
| 0 | 1 | 0 | | | | | | | | | | | | | | | | |
| 1 | 0 | 0 | | | | | | | | | | | | | | | | |
| 1 | 1 | 0 | | | | | | | | | | | | | | | | |
| <table border="1"> <tr> <td>A</td> <td>B</td> <td>C</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </table> | A | B | C | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | | | |
| A | B | C | | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | | | | | | | | | | | | | | | | |
| 0 | 1 | 1 | | | | | | | | | | | | | | | | |
| 1 | 0 | 1 | | | | | | | | | | | | | | | | |
| 1 | 1 | 0 | | | | | | | | | | | | | | | | |
| NAND gate | <table border="1"> <tr> <td>A</td> <td>B</td> <td>C</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </table> | A | B | C | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | | |
| | A | B | C | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | | | | | | | | | | | | | | | | |
| 0 | 1 | 1 | | | | | | | | | | | | | | | | |
| 1 | 0 | 1 | | | | | | | | | | | | | | | | |
| 1 | 1 | 0 | | | | | | | | | | | | | | | | |
| <table border="1"> <tr> <td>A</td> <td>B</td> <td>C</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </table> | A | B | C | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | | | |
| A | B | C | | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | | | | | | | | | | | | | | | | |
| 0 | 1 | 1 | | | | | | | | | | | | | | | | |
| 1 | 0 | 1 | | | | | | | | | | | | | | | | |
| 1 | 1 | 0 | | | | | | | | | | | | | | | | |

Table 2.2: Boolean algorithm respect to LPNs

| NO | Boolean algorithm | Before Logic Petri Nets | After Logic Petri Nets |
|------|--|-------------------------|------------------------|
| (1) | $\overline{\overline{A}} = A$ | | |
| (2) | $A+A=A$ | | |
| (3) | $A \cdot A=A$ | | |
| (4) | $A+0=A$ | | |
| (5) | $A \cdot 1=A$ | | |
| (6) | $A+1=1$ | | |
| (7) | $A \cdot 0=0$ | | |
| (8) | $A+\overline{A}=1$ | | |
| (9) | $A \cdot \overline{A}=0$ | | |
| (10) | $\overline{(A+B)} = \overline{A} \cdot \overline{B}$ | | |
| (11) | $\overline{(A \cdot B)} = \overline{A} + \overline{B}$ | | |

Table 2.3: Fault collapsing respect to LPNs

| Type of gate | Logic gate | Logic Petri Nets |
|--------------|------------|------------------|
| Not gate | | |
| OR gate | | |
| AND gate | | |
| NOR gate | | |
| NAND gate | | |

2.2. A Fault Logic Reasoning Algorithm for Sites and Fired Logic Value

Using the LPNs model, we proposed an algorithm to determine sites of a fault fired logical value at combinational circuits.

Algorithm 1

Step 1: Transfer the circuit into the LPNs circuit.

Step 2: List the table for transitional state of forward of place $f(p_i)$ and backward of place $b(p_i)$.

Step 3: If $b(p_i) = \phi$ and $f(p_i) \neq \phi$ then place p_i is the primary input, while line of a primary input is fired logical value $f(p_i)$, and it is denoted by $D(p_i) = s-a-f(p_i)$.

Step 4: If $b(p_i) \neq \phi$ and $f(p_i) = \phi$ then place p_i is the primary output, while line of a place of primary output is fired logical value $\overline{b(p_i)}$, and it is denoted by $D(p_i) = s-a-\overline{b(p_i)}$,

Step 5: If $b(p_i) \neq \phi$, $f(p_i) \neq \phi$, and $b(p_i) \neq f(p_i)$ then line of a place p_i is fired logical value $f(p_i)$, and it is denoted by $D(p_i) = s-a-f(p_i)$, else no site of fault for test generation.

Using Algorithm 1, the site of fault and fired logic values can be found. An example of simple circuit is described below.

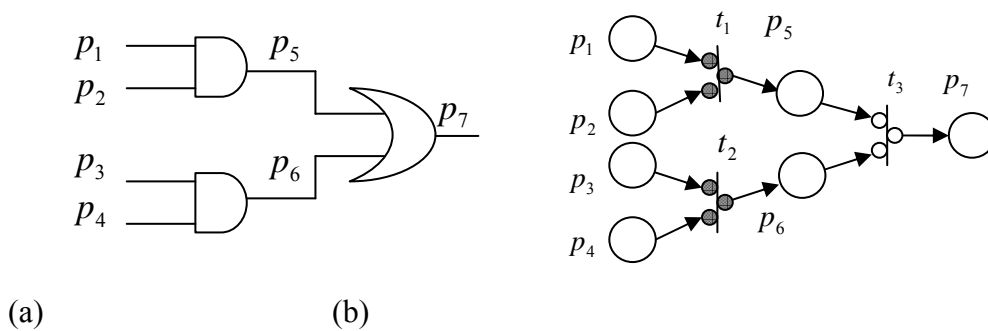


Fig. 2.2. (a) Combinational circuit; (b) LPNs circuit

Example 2: A simple combinational circuit with AND and OR gates are used here (as shown in Fig. 2.2 (a)).

Step 1: Transfer the combinational circuits to LPNs circuit, as Fig. 2.2 (b).

Step 2: List the transitional state as Table 2.4.

Step 3: Place p_1, p_2, p_3 and p_4 are primary inputs since $b(p_1)=b(p_2)=b(p_3)=b(p_4)=\phi$.

$D(p_1), D(p_2), D(p_3)$ and $D(p_4)$ are stuck-at-1 by $f(p_1)=f(p_2)=f(p_3)=f(p_4)=1$.

Step 4: Place p_7 is primary output since $f(p_7)=\phi$. $D(p_7)$ is stuck-at 1 since $\overline{b(p_7)}=1$.

Step 5: p_5, p_6 are not terminal place since $b(p_i) \neq \phi$, $f(p_i) \neq \phi$ and $b(p_i) \neq f(p_i)$, then $D(p_5)$ and $D(p_6)$ are stuck-at 0 since $f(p_5)=f(p_6)=0$.

By the results of above discussion, we can determine the fired logical values (struck-at-fault) of places $p_1 \dots p_7$ as Table 2.4.

Table 2.4: The transitional state for forward and backward of place

| Place p_i | $b(p_i)$ | $f(p_i)$ | $D(p_i)$ | Sign of stuck-at-fault |
|-------------|--------------------|--------------------|------------|------------------------|
| p_1 | ϕ | $i(t_1) = \bullet$ | Stuck-at-1 | ↑ |
| p_2 | ϕ | $i(t_1) = \bullet$ | Stuck-at-1 | ↑ |
| p_3 | ϕ | $i(t_2) = \bullet$ | Stuck-at-1 | ↑ |
| p_4 | ϕ | $i(t_2) = \bullet$ | Stuck-at-1 | ↑ |
| p_5 | $o(t_1) = \bullet$ | $i(t_3) = \circ$ | Stuck-at-0 | ↓ |
| p_6 | $o(t_2) = \bullet$ | $i(t_3) = \circ$ | Stuck-at-0 | ↓ |
| p_7 | $o(t_3) = \circ$ | ϕ | Stuck-at-1 | ↑ |

2.3. Forward and Backward Reasoning Algorithm

By the definitions of literature (Chen et al., 1990), (Chen et al., 2000), immediate reachability set, reachability set, immediate backward incidence set, backward incidence set, and adjacent place, a forward and backward reasoning algorithm is proposed for test generation of combinational circuits.

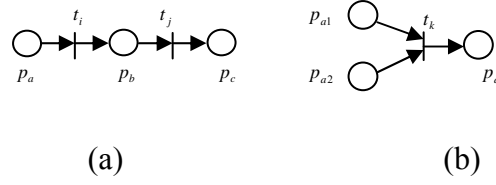


Fig. 2.3. (a) Petri Net for immediate reachability, reachability, immediate backward incidence, and backward incidence sets (b) Petri Net for adjacent place.

Firstly, the PNs model for describing the definitions is shown in Fig. 2.3. For Fig. 2.3 (a), let t_i and t_j be general transitions, and p_a, p_b, p_c be three places. If $p_a \in I(t_i), p_b \in O(t_i), p_b \in I(t_j)$ and $p_c \in O(t_j)$, then we have

- (1) Place p_b is immediately reachable from place p_a ,
- (2) Place p_c is immediately reachable from place p_b ,
- (3) Place p_a is an immediately backward incidence place of place p_b ,
- (4) Place p_b is an immediately backward incidence place of place p_c ,
- (5). Places p_b and p_c are reachable from place p_a ,
- (6) Places p_a and p_b are backward incidence places of place p_c .

The reachability relationship is the reflexive closure of the immediately reachable relationship. The backward incidence relationship is the reflexive closure of the immediately backward incidence relationship.

The set of places that is immediately reachable from a place p_a is called the immediately reachability set of p_a and is denoted by $IRS(p_a)$. The set of places that is reachable from a place p_a is called the reachability set of p_a and is denoted by $RS(p_a)$. The set of places that contains immediate backward places of p_b is called the immediate backward set of p_b and is denoted by $IBIS(p_b)$. The set of places which contains backward incidence places of p_c is called the backward incidence set of p_c and is denoted by $BIS(p_c)$.

For Fig. 2.3(b), let t_k be a transition, p_{a1} and p_{a2} be places. If place $p_{a1} \in I(t_k)$ and place $p_{a2} \in I(t_k)$ then p_{a1} and p_{a2} are called adjacent places with respect to t_k .

Next, we have the following forward and backward reasoning algorithm.

Algorithm 2

Step 1: Transfer the combinational circuits to LPNs circuit.

Step 2: List the table for immediate reachability set, reachability set, immediate backward incidence set, backward incidence set, and the table for set of adjacent places Ap_{jk} for each place p_j .

Step 3: Find the primary inputs p_i ($IBIS(p_i) = \phi$) and primary outputs ($IRS(p_j) = \phi$).

Step 4: Select a site of fault and fired logic value from Table 2.4, activate it and propagate to primary output, i.e., generate a fault effect and sensitized path. Initial mark m_0 are comprised by logical value of fault effect and logical value of a propagation of all adjacent place of sensitized path (i.e., $f : Ap_{jk} \rightarrow i(t_i)$ is logical value of an input transitions of all adjacent place of sensitized path).

Step 5: Find the test pattern by initial mark backtracing path and hold the fault effect as below.

(1) Proposition of place $p_j - D(p_j)$ generates a fault effect and forward propagates the error through t_i to proposition of immediate reachability place $p_k - D(p_k)$ until to the primary output p_o . The change of the state of $D(p_k)$ is depended on the input value $i(t_i)$ and output value $o(t_i)$ of transition relation. If $i(t_i) = o(t_i)$ then $D(p_k) = D(p_j)$. Otherwise, $D(p_k) = \overline{D(p_j)}$. Details of $i(t_i)$ and $o(t_i)$ can be found in Table 2.4.

(2) At the same time, the proposition of place p_j possesses a fault effect. The token of adjacent place Ap_{jk} is equal to a forward mapping from Ap_{jk} to $i(t_j)$, i.e., $\alpha(AP_{jk}) = i(t_j)$, the sensitized path is hold. Then we select a back path of immediate backward incidence place Ap_{jk} through transition t_b ($IBIS(AP_{jk})$) to primary input

p_m . If $\alpha(Ap_{jk}) = o(t_b)$ then $\{\alpha(p_b)\} = \{i(t_b)\}$. Otherwise, $\{\alpha(p_b)\} \not\subset \{i(t_b)\}$.

- (3) Find the test generation of back path. Place p_j propagate back through transition t_b to p_i until to primary input p_m . If $\alpha(p_j) = o(t_b)$ then $\{\alpha(p_b)\} = \{i(t_b)\}$. Otherwise, $\{\alpha(p_b)\} \not\subset \{i(t_b)\}$.

Step 6: If we can find a token of primary input $\alpha(p_m)$ set and generate a fault effect then fault f is detectable and test generation is set of a primary input token $\alpha(p_m)$.

Finally, we use an example to illustrate the LPNs reasoning process for test generation.

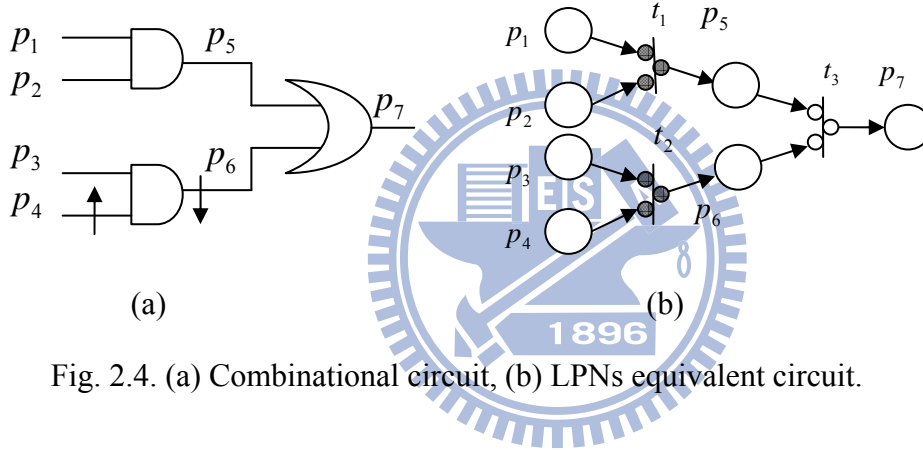


Fig. 2.4. (a) Combinational circuit, (b) LPNs equivalent circuit.

Example 3: Determine test generation of sat-at-1 at p_4 and sat-at-0 at p_6 in combinational circuit, as shown in Fig. 2.4(a).

Case (a) $D(p_6)$: sat-at-0.

Step 1: Transfer the combinational circuits to LPNs circuits as shown in Fig. 2.4 (b).

Step 2: List the table for immediate reachability set, reachability set, immediate backward incidence set, and backward incidence set table and the table for set of adjacent places Ap_{jk} , as Table 2.5 and Table 2.6.

Step 3: Find the primary input $p_m = \{p_1, p_2, p_3, p_4\}$ and the primary output $p_o = \{p_7\}$.

Step 4: Select a $D(p_6)$ (which is sat-at-0), $\alpha(p_6) = 1$ is generate a fault effect and $\alpha(p_5) = 0$ is logical value of a propagation of all adjacent place of sensitized path. So

$$m_0 = \{\alpha(p_6) = b(p_6) = o(t_2) = 1, \alpha(AP_{67}) = \alpha(p_5) = i(t_3) = 0\}.$$

Step 5:

(1) $D(p_6)$ propagates the error through t_3 to $D(p_7) = 1/0$ since $i(t_3) = o(t_3)$.

(2) $\alpha(AP_{67}) = \alpha(p_5) = i(t_3) = 0$, sensitized path is hold. $\alpha(p_5) \neq o(t_1) = 0$ implies

$$\{\alpha(p_1), \alpha(p_2)\} \neq \{i(t_1), i(t_1)\} = \{1,1\}, \text{ i.e., } \{\alpha(p_1), \alpha(p_2)\} = \{0,0\} \text{ or } \{0,1\} \text{ or } \{1,0\}.$$

(3) $\alpha(p_6) = b(p_6) = o(t_2) = 1$ implies $\{\alpha(p_3), \alpha(p_4)\} = \{i(t_2), i(t_2)\} = \{1,1\}$.

Step 6: $D(p_6)$ sat-at-0 is detectable. Then, the test generation is

$$\alpha(p_{in}) = \{\{\alpha(p_1), \alpha(p_2)\} \neq \{1,1\}, \{\alpha(p_3), \alpha(p_4)\} = \{1,1\}\}.$$

Case (b) $D(p_4)$: sat-at-1

Step 1: Transfer the combinational circuits to LPNs circuit as Fig. 2.4 (b).

Step 2: List the table for immediate reachability set, reachability set, immediate backward incidence set and backward incidence set table and the table for table set of adjacent places Ap_{jk} , as Table 2.5 and Table 2.6.

Step 3: Find the primary input $p_{in} = \{p_1, p_2, p_3, p_4\}$ and the primary output $p_o = \{p_7\}$.

Step 4: Select $D(p_4)$ (sat-at-1) and $\alpha(p_4) = 0$ generate a fault effect. $\alpha(p_3) = 1$, $\alpha(p_5) = 0$ are logical value of a propagation of all adjacent place of sensitized path.

$$\text{So } m_0 = \{\alpha(p_4) = \overline{f(p_4)} = \overline{i(t_2)} = 0, \alpha(AP_{46}) = \alpha(p_3) = i(t_2) = 1, \\ \alpha(AP_{67}) = \alpha(p_5) = i(t_3) = 0\}$$

Step 5:

(1) $D(p_4)$ (sat-at-1) and $\alpha(p_4) = 0$ since $i(t_2) = o(t_2)$, $D(p_4)$ propagates the error through t_2 to $D(p_6) = 0/1$. and $i(t_3) = o(t_3)$, $D(p_4)$ propagates the error through t_2 to $D(p_7) = 0/1$.

(2) $\alpha(AP_{46}) = \alpha(p_3) = i(t_2) = 1$.

(3) $\alpha(AP_{67}) = \alpha(p_5) = i(t_3) = 0$, sensitized path is hold. The result is similar to (2) of case (a)- Step 5. Thus, $\{\alpha(p_1), \alpha(p_2)\} \neq \{i(t_1), i(t_1)\} = \{1,1\}$, i.e., $\{\alpha(p_1), \alpha(p_2)\}$

$$= \{0,0\} \text{ or } \{0,1\} \text{ or } \{1,0\}.$$

Step 6: $D(p_4)$: sat-at-1 is detectable and test generation is

$$\alpha(p_m) = \{\{\alpha(p_1), \alpha(p_2)\} \neq \{1,1\}, \{\alpha(p_3), \alpha(p_4)\} = \{1,0\}\}.$$

Table 2.5: Immediate Reachability Set, Reachability Set, Immediate Backward Incidence Set and Backward Incidence Set for each place p_i

| Place | IRS(p_i) | RS(p_i) | IBIS(p_i) | BIS(p_i) |
|-------|--------------|----------------|----------------|--|
| p_1 | $\{p_5\}$ | $\{p_5, p_7\}$ | ϕ | ϕ |
| p_2 | $\{p_5\}$ | $\{p_5, p_7\}$ | ϕ | ϕ |
| p_3 | $\{p_6\}$ | $\{p_6, p_7\}$ | ϕ | ϕ |
| p_4 | $\{p_6\}$ | $\{p_6, p_7\}$ | ϕ | ϕ |
| p_5 | $\{p_7\}$ | $\{p_7\}$ | $\{p_1, p_2\}$ | $\{p_1, p_2\}$ |
| p_6 | $\{p_7\}$ | $\{p_7\}$ | $\{p_3, p_4\}$ | $\{p_3, p_4\}$ |
| p_7 | ϕ | ϕ | $\{p_5, p_6\}$ | $\{\{p_5, p_6\}, \{p_3, p_4\}, \{p_1, p_2\}\}$ |

Table 2.6: Set of Adjacent Places Ap_{jk} for each place p_j

| Place p_j | Transition t_i | $i(t_i)$ | $o(t_i)$ | Place p_k | Ap_{jk} |
|-------------|------------------|----------|----------|-------------|-----------|
| p_1 | t_1 | 1 | 1 | p_5 | p_2 |
| p_2 | t_1 | 1 | 1 | p_5 | p_1 |
| p_3 | t_2 | 1 | 1 | p_6 | p_4 |
| p_4 | t_2 | 1 | 1 | p_6 | p_3 |
| p_5 | t_3 | 0 | 0 | p_7 | p_6 |
| p_6 | t_3 | 0 | 0 | p_7 | p_5 |

The comparison between LPNs model and traditional method by Kirkland et al., (1988) in test generation for combinational circuit is shown in Fig. 2.5. The major differences are described below. (1) LPNs approach is parallel processing, i.e., LPNs approach has less operational time than (Kirkland et al., 1988); (2) every back tracing path of LPNs is shorter than (Kirkland et al., 1988), i.e., complexity of determining test generation LPNs is easier; (3) LPNs approach needs larger memory than (Kirkland et al., 1988), i.e., cost using LPNs approach will increase.

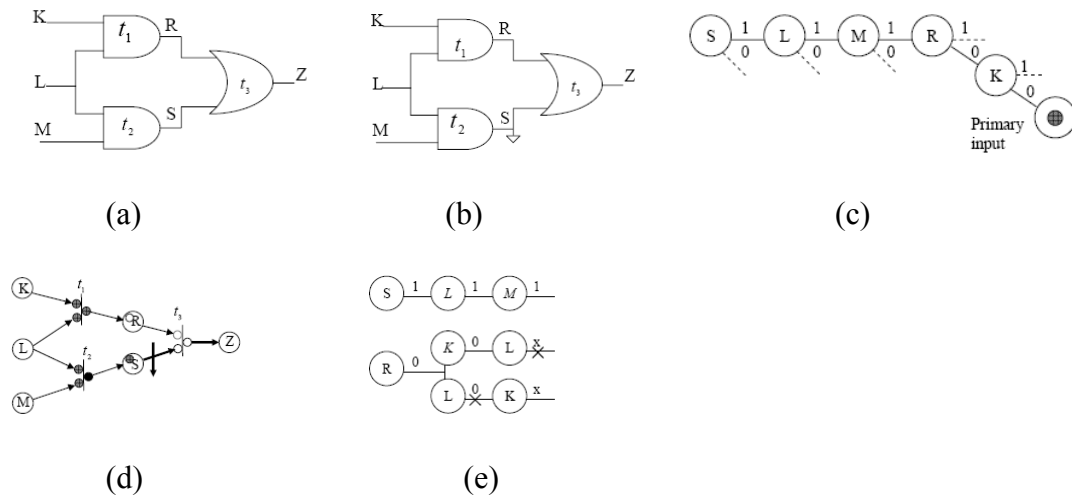


Fig. 2.5. The comparison between LPNs model and traditional method (Kirkland 1988): (a) A sample good circuit; (b) A faulty circuit; (c) The search graph for locating the fault; (d) A faulty circuit of LPNs; (e) The search graph for locating the fault of LPN.

2.4. Summary

For solving test generation and site of fault in combinational circuits, we have proposed a so-called Logic Petri Net model. The LPNs model embeds critical of truth table into transition of Petri Net with clear physical concepts, fast calculation speed and high veracity. It first transfers a complexity circuit problem to a local adjacent place and transition relational one. Thus, the site of fault and fired logical value problem is simplified. Both algorithms were presented for obtaining the test pattern and improved the calculation speed. Two examples were shown to demonstrate the effectiveness of LPNs model.

CHAPTER 3

Constructing an Abstract Model for the Diagnosis of Ladder Diagrams Using Boolean Petri Nets

As shown in Fig. 3.1, hierarchical control is an approach for the design of large-scale discrete event systems that are used to deduce complexity (Lee et al., 2004). In a manufacturing system, a LDs controller may use a local controller, which allows the LDs controller to be diagnosed and monitored remotely. In this chapter, the local controller (i.e., LDs controller) and abstract model (i.e., corresponding to the LDs model) are modeled with BPNs. The LDs controller model employed in this thesis is a structural model that is similar to the original LDs architecture, and the abstract model is a behavioral model. The behavioral model is simplified by the structural model; however, the behavioral model matches the functions of the LDs controller.

To construct an abstract model from a simplified BPNs model, a BPNs module was constructed from a table of LDs rungs based on a Boolean equation.

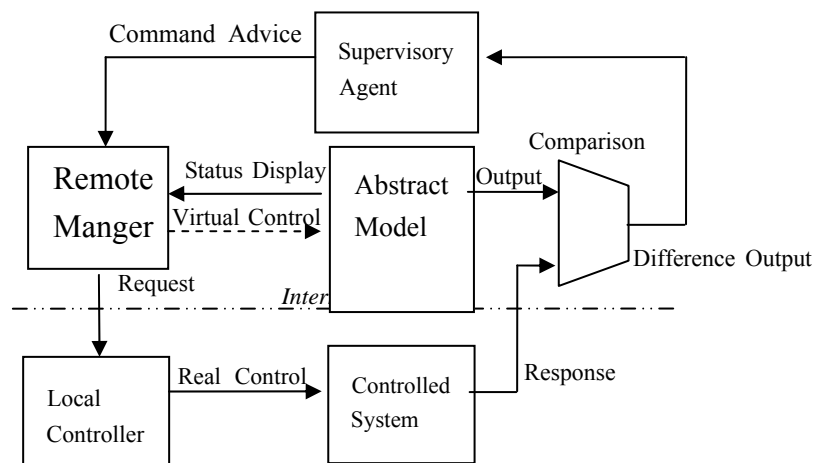


Fig. 3.1. Proposed hierarchical control (by Lee (2004))

3.1. Boolean Petri nets

Carl Adam Petri proposed the Petri nets theory. Fig. 3.2 shows the structure of Petri nets in a directed bipartite graph that consists of places, transitions, and arcs. A circle with a token represents the places. A bar indicates the flow of tokens when firing condition is satisfied, which represents the transition. Finally, a straight line that connects the place to the transition, or the transition to the place denotes the arc, which indicates the flow of tokens in the direction of the arrow.

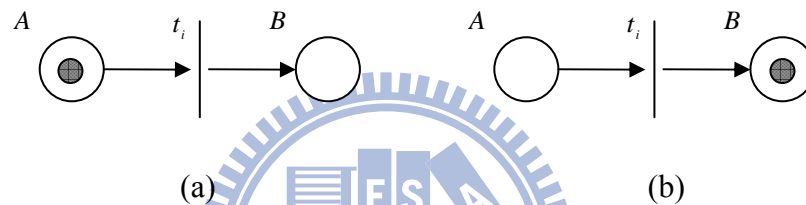


Fig. 3.2(a) An example Petri nets, (b) A token moving from A to B in Fig. 3.2(a) after t_i fire.

3.1.1 Definition of Boolean Petri nets

The purpose of developing the BPNs model is that this model exhibits the imply logic property in a LDs. The simplest way to represent LDs is by its Boolean equation. The approach proposed in this chapter embeds the Boolean equation in a PN transition to develop the BPNs model. This special transition is called the “Boolean transition.” Table 3.1 describes the BPNs model corresponding to the Boolean equation. Clearly, the BPNs model also matches the LDs. To map LDs into a Petri nets, the Petri nets must be extended. This extended Petri nets is called a Boolean Petri nets, which can be defined formally as

$$PN = (P, T, A, I, O, in, out, M_0) \quad (3-1)$$

Where $P = \{p_1, p_2, \dots, p_m\}$, $m \geq 1$, is a finite set of places representing the LDs action state. The places are associated with a component or a set of components (i.e., a compound component) such as the actuator output, relay coil, timer, counter solenoid, or source; $T = \{t_1, t_2, \dots, t_n\}$, $n \geq 1$, is a finite set of transitions representing event whether occurs or not. These transitions are always associates with a switch or a set of switches and represented by Boolean equations or variables.

The switch can be a normal open (NO) switch or normal closed (NC) switch. The NO switch is also called an “a” contact and the NC switch is also called a “b” contact, where $P \cap T = \Phi$ and $P \cup T \neq \Phi$.

$A \subset (P \times T) \cup (T \times P)$ is a set of arcs (\rightarrow) consisting of input arcs $A_i(P \times T)$ and output arcs $A_o(T \times P)$. The weight of each directed arc in this chapter is 1, and $A_i(P \times T)$ is defined as directed arcs from a place to a transition. Places are called input places and transitions are called output transitions, and the input arc is represented by a connected line as channel of token. $A_o(T \times P)$ is defined as directed arcs from a transition to a place, the transition is called the input transition and the place is called the output place, and the output arc is represented by a connected line as channel of token. The arc may be preservation arc ($\bullet \rightarrow$) that a input arc and a output arc exist simultaneously between same place and transition (Lee et al., 2000); $I : T \times P \rightarrow N$ is an input function that defines as number of output arcs $A_o(T \times P)$, where $N = \{0, 1, 2, \dots\}$, $O : P \times T \rightarrow N$ is an output function that are defined as number of input arcs $A_i(P \times T)$, where $N = \{0, 1, 2, \dots\}$; $in = \{in_1, in_2, \dots, in_n\}$ is a set of input switch, which is represented by Boolean function or variable. A set of input switches is associated with a transition t_j and is denoted by $t_j = \{in\}$. The Boolean function or variable can be ‘1’, in which case the related transition t_j is allowed to fire if it is enabled, or it can be ‘0’, in which case the related transition t is not allowed to fire; $out = \{out_1, out_2, \dots, out_m\}$ is a set of output actuator which is associated with a p_i and is denoted by $p_i = \{out\}$; $M_o(P)$ is the

initial marking that uses a token to represent the place status.

A transition is **enabled** if the number of tokens at the place is larger than or equal to the number of input arcs. A transition is **firing** if the enabled transition is fired and its transition states are true (i.e., the Boolean equation is true). When a transition fires, it moves the tokens from input places to output places along the input arcs and output arcs, as Fig. 3.2 illustrates. This moves the token of place *A* to place *B* along directed arcs if transition t_i is firing. A marking is denoted as an m -vector, where m is the total number of place P , while $m(p_i)$ is represented by the number of tokens at place p_i (Murata et al., 1989).

For the marking m_0 , there is an enabled transition t_1 . If there is a firing of transition t_1 , then the marking is immediately **reachable** to m' from m_0 , denoted by $m_0[t_1 > m']$. A marking m_i is said reachable from m_0 if there exists a sequence of firings that transforms m_0 to m_i . $R(m_0)$ is defined as the set of all reachable markings from m_0 . $F(m_0)$ is defined as the set of all firing sequences from m_0 . A place p_i is said to be **bounded** for an initial marking m_0 if $\exists k > 0, \forall m \in R(m_0), m(p_i) \leq k$, and $\forall m \in R(m_0)$. Specifically, it is said to be **safe** if $k=1$. A marking m_0 is said to be **live** for a Petri nets if every marking has been reached from m_0 , which indicates it is possible to fire any transition of the nets by some firing sequence (Murata et al., 2007), (Zhou et al., 1998). If m_0 may be reached from any marking, The Petri nets is said to be **reversible**.

To simulate the behavior of LDs, this approach changes a state or marking according to defined firing rules for the Boolean Petri nets model.

3.1.2. State equation

The firing definition easily shows that the token moves from state M_{k-1} to another state M_k by the k th firing, and U_k is a firing vector which can be given in terms of the following matrix state equation for Petri nets (Murata et al., 1977)

$$M_k = M_{k-1} + A^T U_k \quad (3-2)$$

Where U_k is called firing vector, and A^T is called the incidence matrix for any given topological structure of Petri nets, defined by

$$A^T(p_i, t_j) = \begin{cases} -O_{ij}(p_i, t_j) \\ 0 \\ I_{ji}(t_j, p_i) \end{cases}, \text{ where } 1 \leq i \leq m, 1 \leq j \leq n. \quad (3-3)$$

Note that M_k must be a vector of nonnegative integers (Murata et al., 1997). The firing vector will then select an appropriate column of A^T such that

$$M_{k-1} + A^T U_k \geq 0 \text{ for each } k \quad (3-4)$$

3.1.3. Definition of action dominance and equivalence

Dominance. An action p_1 is said to dominate another action p_2 in an irredundant place iff every exist of token for p_2 is also exist of token for p_1 . i.e., the life of a token of p_1 is longer than p_2 , denoted as $m(p_1) > m(p_2)$. The reduction of the place p_1 to be analyzed is based on the dominance relation.

Example 1: Fig. 3.3 shows a PN in which p_1 is dominated by p_2 and p_3 , i.e., $m(p_1) > m(p_2)$ and $m(p_1) > m(p_3)$. Fig. 3.4 shows the reduction result.

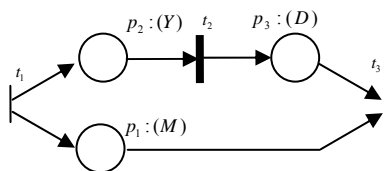


Fig. 3.3 A simple example

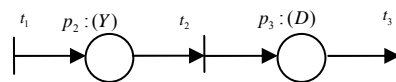


Fig. 3.4 The reduction result of Fig. 3.3

Equivalence. The actions p_1 and p_2 are equivalent if exist of token is same condition for p_1 and p_2 , i.e., $m(p_1) > m(p_2)$ and $m(p_2) > m(p_1)$. The composite of

the place p_1 and p_2 to be analyzed is based on the equivalent action.

Example 2: Fig. 3.5(a) shows a PN in which p_2 is equivalent to p_3 , i.e., $m(p_3) > m(p_2)$ and $m(p_2) > m(p_3)$. Fig. 3.5(b) shows the composite result.

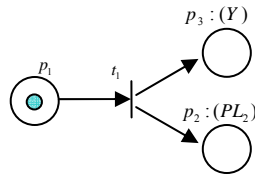


Fig. 3.5(a) A simple example

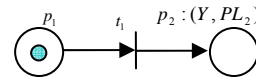


Fig. 3.5(b) The composite result of Fig. 3.5(a)

3.2. Ladder Diagram Model Using Boolean Petri Net

3.2.1 Model of basic modules

In ladder diagrams, the horizontal line (rung) and the associated elements represent Boolean equations. Similarly, in Boolean Petri nets, the associated transitions represent Boolean equations. In ladder diagrams, the symbol “○” represents the dependent element of the equation (coil). Similarly, in Boolean Petri nets, the symbol “○” represents the dependent element of the equation (place). In ladder diagrams, “|” represents the independent element (normal open contacts), while in Boolean Petri nets, “| or |” represents the independent element (input transitions). A diagonal line placed in the middle of these symbols (i.e., “|/|”) represents normal closed contacts, which indicate that the negated value of the variable is used. Similarly, bar “| or |” represents the output transition. In ladder diagrams, variables (contacts) placed in a series represent the *AND* Boolean function, while contacts placed in parallel represent the *OR* Boolean function. The rungs are executed in order from top to bottom. Therefore, the Type 8 ladder diagram in Table 3.1 represents Boolean equations $M = (A + M)\bar{B}$ and $N = M$ (Bender et al., 2008). In Boolean Petri nets, a similar

input transition represents $(A + M)$, denoted as $t_1 : (A + M)$, which is a composite transition. Conversely, output transitions represent \bar{B} , denoted as $t_2 : (B)$, and output places are denoted as $p_2 : (M, N)$ which are composite places. Finally, Table 3.1 summarizes some typical LDs modules and their corresponding Boolean Petri nets models, where S is a pseudo source and the composite and decomposite of Boolean Petri nets are as shown in Table 3.2.

TABLE 3.3: Some LDs modules and corresponding models

| Modules | Ladder Diagrams | Boolean Equations | Boolean Petri Nets | Modules | Ladder Diagrams | Boolean Equations | Boolean Petri Nets | |
|---------|-----------------|--------------------------|--------------------|---------|-----------------|--|---|--|
| Type 1 | | $M = A$ | | Type 3 | | $M = AB$ | | |
| | | $M = T_{\Delta}$ | | | Type 4 | | $M = A + B$ | |
| | | $M = A$ | | | Type 5 | | $M = \bar{A}\bar{B}$ $\bar{M} = A + B$ | |
| | | $M = A + M$ $M = A$ | | | Type 6 | | $M1 = A$ $M2 = A$ $M1 = M2$ | |
| Type 2 | | $\bar{M} = A$ | | Type 7 | | $M = (A + M)\bar{B}$ $= A\bar{B}$ | | |
| | | $\bar{M} = T_{\Delta}$ | | Type 8 | | $M = (A + M)\bar{B} = A\bar{B}$ $N = M$ | | |
| | | $\bar{M} = A$ | | Type 9 | | Timer = A $M = T_{\Delta}$ | | |
| Type 10 | | $\bar{M} = A$ $N = A$ | | Type 11 | | $\bar{B} = T_{\Delta}$ $C = T_{\Delta}$ | | |

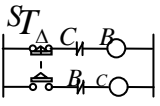
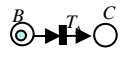
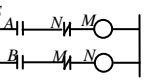
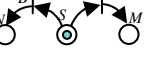
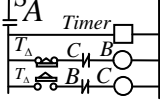

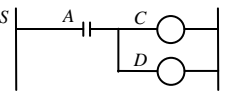
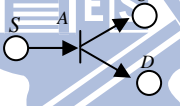
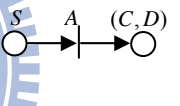
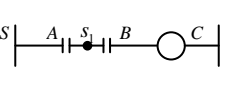
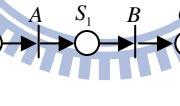
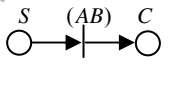
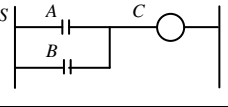
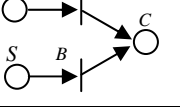
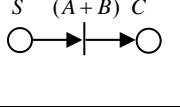
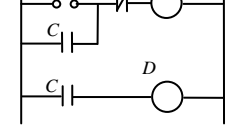
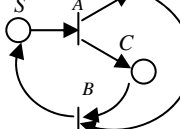
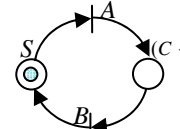
| | | | | | | | |
|---------|---|---|---|---------|--|---|---|
| Type 12 |  | $\bar{B} = T_{\Delta} + C$ $C = T_{\Delta} \bar{B}$ $\bar{B} = T_{\Delta} + C$ $= T_{\Delta} + T_{\Delta} \bar{B}$ $(1 + T_{\Delta}) \bar{B} = T_{\Delta}$ $\Rightarrow \bar{B} = T_{\Delta}$ $C = T_{\Delta} \bar{B}$ $= T_{\Delta} (T_{\Delta} + C)$ $(1 + T_{\Delta}) C = T_{\Delta}$ $\Rightarrow C = T_{\Delta}$ |  | Type 13 |  | $M = A \bar{N}$ $N = B \bar{M}$ $\bar{M} = A \bar{N} = \bar{A} + N$ $= \bar{A} + B \bar{M}$ $(1 + B) \bar{M} = \bar{A}$ $\bar{M} = \bar{A}$ $\Rightarrow M = A$ $\bar{N} = B \bar{M} = \bar{B} + M$ $= \bar{B} + A \bar{N}$ $(1 + A) \bar{N} = \bar{B}$ $\bar{N} = \bar{B}$ $\Rightarrow N = B$ |  |
| Type 14 |  | $Timer = A$ $B = \bar{A}$ $C = T_{\Delta}$ |  | | | | |

Table 3.2: Composite and Decomposite of Boolean Petri nets

| LDs | Boolean Equation | BPNs | |
|---|-------------------------|---|--|
| | | Decomposite | Composite |
|  | $C=A$ $D=A$ $C=D$ |  |  |
|  | $C=AB$ |  |  |
|  | $C=A$ $C=B$ $C=A+B$ |  |  |
|  | $C = A \bar{B}$ $D = C$ |  |  |

3.2.2. Model of faulty ladder diagram

A LDs circuit fault may generally be classed as both stuck-at 0 (s-a-0) and stuck-at 1 (s-a-1) type; the stuck-at 0 fault is like a NO switch and the stuck-at 1 fault is like an NC switch. Hence the fault model of the ladder diagram can be modeled as a Petri

nets. LDs of the possessed faulty example are illustrated in Fig. 3.6(a) and the Petri nets model is illustrated in Fig. 3.6(b). Where fault f_1 are represented s-a-0 to represent the switch A is struck at open, fault f_2 is represented s-a-1 to represent the switch B is stuck at close. According to Eq. (3-3), the incidence matrix

$$\text{is } A^T = \begin{matrix} & t_1 & t_2 \\ \begin{matrix} p_1 \\ p_2 \end{matrix} & \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \end{matrix}.$$

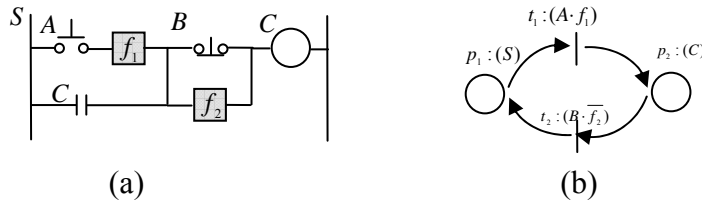


Fig. 3.6. (a) A LDs of possessed fault (b) A Petri nets model of possessed fault

The faults classified in the two cases are interpreted as below.

Case 1: Assume f_1 is s-a-0 fault and initial marking is $M_0 = \begin{matrix} p_1 \\ p_2 \end{matrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

When the bottom is pushed $A=1$.

$$U_1 = \begin{matrix} t_1:(A) \\ t_2 \end{matrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}, U_{f_1} = \begin{matrix} t_1:(A \cdot f_1) \\ t_2 \end{matrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix}, U_{1/f_1} = \begin{matrix} t_1:(A/A \cdot f_1) \\ t_2 \end{matrix} \begin{bmatrix} 1/0 \\ 0 \end{bmatrix}, \text{ where } U_1 \text{ and } U_{f_1} \text{ are}$$

represented as the fault free and faulty firing vector, respectively. U_{1/f_1} is represented as the fault free/faulty firing vector.

According to Eq. (3-2)

$$M_{1/f_1} = M_0 + A^T U_{1/f_1} = \begin{matrix} p_1 \\ p_2 \end{matrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{matrix} t_1 & t_2 \\ \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \end{matrix} \begin{bmatrix} 1/0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1/0 \\ 1/0 \end{bmatrix} = \begin{bmatrix} 0/1 \\ 1/0 \end{bmatrix}, \text{ i.e., } M_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; M_{f_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

where M_1 and M_{f_1} are represented as the fault free and faulty marking vector, respectively. M_{1/f_1} is represented as the fault free/faulty making vector.

In the LDs circuit, f_1 fault means the coil C is not active since the switch A is stuck at open.

Case 2: Assume f_2 is s-a-1 fault and current marking is $M_1 = \begin{matrix} p_1 \\ p_2 \end{matrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

When the bottom is pushed $B=1$

$$U_2 = \begin{matrix} t_1 \\ t_2 : (B/B \cdot \bar{f}_2) \end{matrix} \begin{bmatrix} 0 \\ 1/0 \end{bmatrix}$$

$$M_{2/f_2} = M_1 + A^T U_2 = \begin{matrix} p_1 \\ p_2 \end{matrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{matrix} t_1 & t_2 \\ -1 & 1 \\ 1 & -1 \end{matrix} \begin{bmatrix} 0 \\ 1/0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1/0 \\ -1/0 \end{bmatrix} = \begin{bmatrix} 1/0 \\ 0/1 \end{bmatrix} \text{ i.e., } M_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad M_{f_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

In the LDs circuit, f_2 fault means the coil C is maintain action since the switch B is stuck at close.

3.3. Application Example

This section illustrates a practical example of hierarchical control system in Fig. 3.1. The local controller is a LDs circuit. This circuit can be modeled by BPNs and is simplified to obtain an abstract model using Table 3.1 of the preceding section. The system fault can be diagnosed by the difference between the LDs response and abstract model response. The differences are as decision of supervisor agent.

3.3.1. Constructing an abstract model using a Boolean PNs

To start a three-phase motor, a LDs controller use type of Y- Δ starting to limit starting current, as shown in Fig. 3.7 and symbol descriptions in Table 3.3. In the LDs controller, the bottom Pb_1 is control relay coil M , Y and timer coil active. The motor enters the starting state when NO contacts of M and Y are turned on. Next, the relay coil Y turns off after delay time T_Δ , and the motor returns to the normal state when the relay coil Y turns off and relay coil D turns on. Finally, the motor stops if the bottom Pb_2 is pushed or the current is overload. This LDs controller can be specified as follows:

Step 1) The motor is commanded to start (Pb_1).

Step 2) The motor starting time is T_Δ .

Step 3) The motor is commanded to stop (Pb_2).

Step 4) The motor will stop if the current is overloaded.

The implicit specification is as following:

Spec) The relay coil D and relay coil Y are mutually exclusive.

The transformation from the ladder diagram (in parallel) to the abstract model (in series) is based on the following steps:

Step 1) A rung or compound rung of LDs is converted to a Boolean Petri nets module using Table 3.1 or the Boolean equation. LDs controller then assembles Boolean Petri nets modules, as Fig. 3.8 shows, where (1), (2) ... and (9) correspond to the number of LDs rungs.

Step 2) A Boolean Petri nets can be given after eliminating the redundant or pseudo places (i.e., the S place), as Fig. 3.9 illustrates.

Step 3) An abstract model can be obtained according to dominance relation reduce some places (in this case, an abstract model reduce place p_2), and eliminating some redundant elements (i.e., the coil of time or auxiliary relay), as Fig. 3.10 shows.

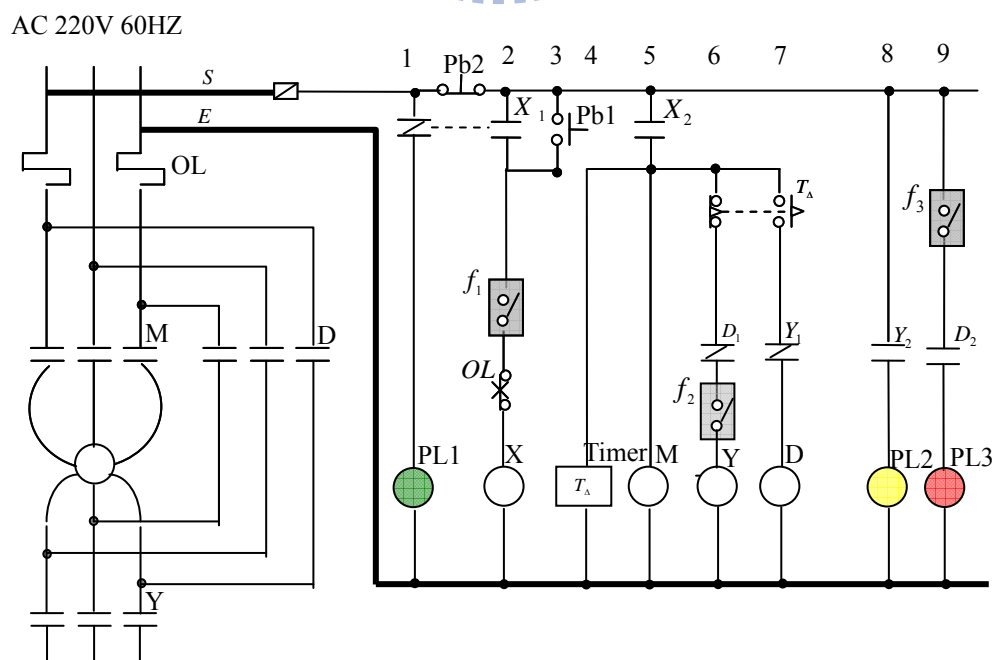















Fig. 3.7. Control circuit of a Y- Δ starting motor.

Table 3.3: The Descriptions of symbol

| Symbol | Description | Symbol | Description |
|---|--|---|----------------------------|
|  | Indicator light of green |  | “b” contact of Push bottom |
|  | Indicator light of yellow |  | “a” contact of Push bottom |
|  | Indicator light of red |  | “a” contact of relay |
|  | Relay |  | “b” contact of relay |
|  | Timer |  | “a” contact of timer |
|  | Stuck at 0 (s-a-0) switch for simulate fault (Abramovici et al., 1990) |  | “b” contact of timer |
| 1~9 | rung number |  | “b” contact of over load |

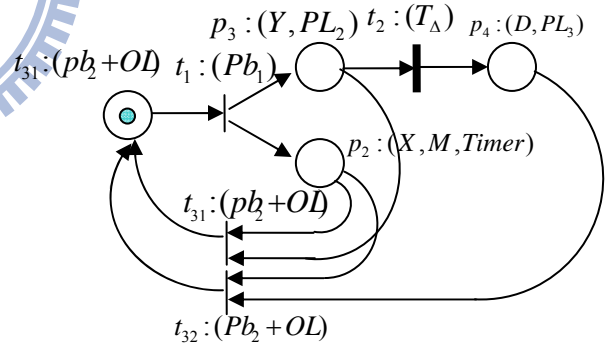
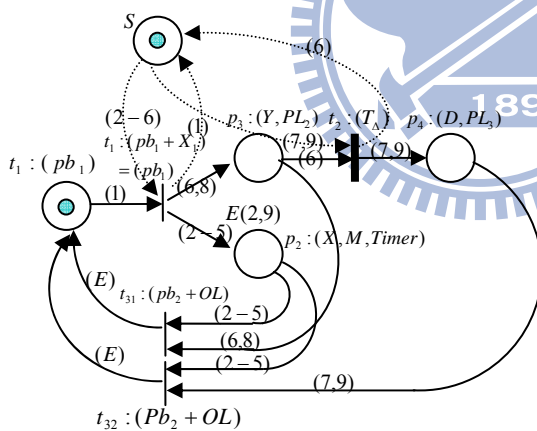


Fig. 3.8. BPNs model of a LDs controller.

Fig. 3.9. Equivalent diagram of Fig. 3.8.

3.8.

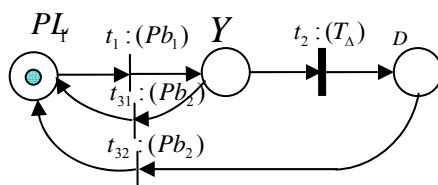


Fig. 3.10 Abstract model of Fig. 3.9

3.3.2 Properties of the proposed Boolean Petri nets

The reachability of PNs is a tree, which uses states as nodes and transitions as arcs (David et al., 1992). The construction of this tree starts from the root node. The root node is represented as the initial state, and the arcs outgoing from the root node are marked by the corresponding enabled transition. The arc will outgo to a new node (state) from the firing of the corresponding transition (arc). The above procedures are repeated until they produce duplicate nodes. Terminal nodes are identical to existing nodes, which have no any enable transitions are met. In the reachability tree, dash lines indicate nodes.

Due to the similar processes of PNs reachability tree, this study presents the reachability tree of the proposed BPNs in Fig. 3.9. For the sake of simplicity in representing the node (node) in the reachability tree, define the state variable vector in the reachability tree as $[p_1 \ p_2 \ p_3 \ p_4]$, and allow the initial state to be $[1 \ 0 \ 0 \ 0]$. If transition $t_1 : (Pb_1)$ fires when $Pb_1 = 1$ (i.e. Pb_1 is active), the state moves to $[0 \ 1 \ 1 \ 0]$. If transition $t_{31} : (Pb_2)$ fires when $Pb_2 = 1$ (i.e. Pb_2 is active), then the state moves to $[1 \ 0 \ 0 \ 0]$. Subsequently, if transition $t_2 : (T_\Delta)$ is enabled and fires, then the state moves to $[0 \ 1 \ 0 \ 1]$, while if transition $t_{32} : (Pb_2)$ is enabled and fires, the state moves to $[1 \ 0 \ 0 \ 0]$.

Proposition 1: The proposed BPN is live.

Proof: Consider a case based on the reachability tree in Fig. 3.11. This figure shows that there is no terminal node. Therefore, there always exists some sample path such that any transitions can eventually fire to reach any states from the initial state p_1 , i.e. $R(m_0) = \{p_2, p_3, p_4\}$, $F(m_0) = \{t_1, t_2\}$. According to this definition, the proposed

BPN is live.

Proposition 2: The proposed BPN is reversible.

Proof: The reachability tree in Fig. 3.11 indicates that there is no terminal node. Therefore, there always exist some sample path such that any transitions can eventually fire to reach the initial state p_1 from any states (i.e. p_2, p_3, p_4), $p_1 \in R(p_2), p_1 \in R(p_3), p_1 \in R(p_4)$. According to this definition, the proposed BPN is reversible.

Proposition 3: The proposed BPN is bound.

Proof: In stable PNs, the number of tokens in any place will not grow infinitely. The reachability tree in Fig. 3.11 indicates that one and only one marked token corresponds to any specific state. Therefore, the number of marked tokens in p_1, p_2, p_3 and p_4 is bounded above by 1. According to this definition, the proposed BPN is bounded and safe.

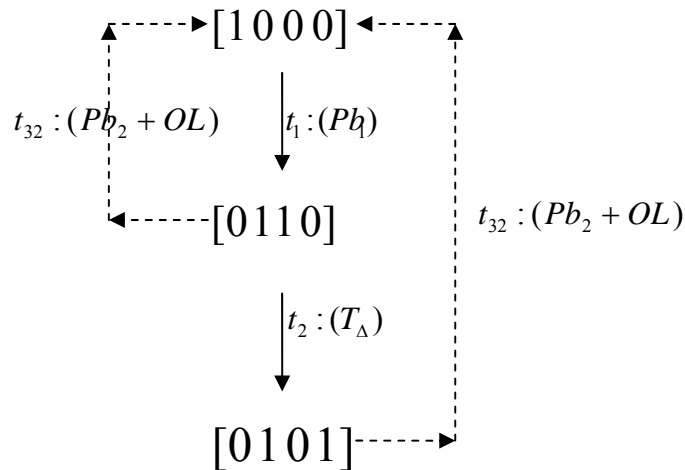


Fig. 3.11 The reachability tree of the Proposed BPNs

Similarly, the proposed abstract model in Fig. 3.10 is live, reversible, and safe. In the

Petri nets model, the live is represented as a reachable starting state (i.e. Y state) and running state (i.e. D state) from the ideal state (i.e. PL_1 state), the safe is represented as only existing in one state, the reversible is represented as returning to the ideal state (i.e. PL_1 state) from any other state (i.e. Y state and D state).

3.3.3 State equation

According to Eq. (3-2), Fig. 3.9 shows that the state equation is $M_k = M_{k-1} + A^T U_k$

$$\begin{aligned}
 A^T &= \begin{matrix} & t_1 & t_2 & t_{31} & t_{32} \\ p_1 & \begin{bmatrix} -1 & 0 & 1 & 1 \end{bmatrix} \\ p_2 & \begin{bmatrix} 1 & 0 & -1 & -1 \end{bmatrix} \\ p_3 & \begin{bmatrix} 1 & -1 & -1 & 0 \end{bmatrix} \\ p_4 & \begin{bmatrix} 0 & 1 & 0 & -1 \end{bmatrix} \end{matrix}, & M_0 &= \begin{matrix} p_1 & \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ p_2 & \\ p_3 & \\ p_4 & \end{matrix}, & U_1 &= \begin{matrix} t_1 & \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ t_2 & \\ t_{31} & \\ t_{32} & \end{matrix}, \\
 U_2 &= \begin{matrix} t_1 & \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ t_2 & \\ t_{31} & \\ t_{32} & \end{matrix}, & U_{31} &= \begin{matrix} t_1 & \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \\ t_2 & \\ t_{31} & \\ t_{32} & \end{matrix}, & U_{32} &= \begin{matrix} t_1 & \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\ t_2 & \\ t_{31} & \\ t_{32} & \end{matrix} \\
 M_1 &= M_0 + A^T U_1 = \begin{matrix} p_1 & \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ p_2 & \\ p_3 & \\ p_4 & \end{matrix} + \begin{matrix} t_1 & \begin{bmatrix} -1 & 0 & 1 & 1 \end{bmatrix} \\ t_2 & \\ t_{31} & \\ t_{32} & \end{matrix} \begin{matrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \end{matrix} = \begin{matrix} p_1 & \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \\ p_2 & \\ p_3 & \\ p_4 & \end{matrix} \\
 M_2 &= M_1 + A^T U_2 = \begin{matrix} p_1 & \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \\ p_2 & \\ p_3 & \\ p_4 & \end{matrix} + \begin{matrix} t_1 & \begin{bmatrix} -1 & 0 & 1 & 1 \end{bmatrix} \\ t_2 & \\ t_{31} & \\ t_{32} & \end{matrix} \begin{matrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \end{matrix} = \begin{matrix} p_1 & \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \\ p_2 & \\ p_3 & \\ p_4 & \end{matrix} \\
 M_{31} &= M_1 + C^T A^T U_{31} = \begin{matrix} p_1 & \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \\ p_2 & \\ p_3 & \\ p_4 & \end{matrix} + \begin{matrix} t_1 & \begin{bmatrix} -1 & 0 & 1 & 1 \end{bmatrix} \\ t_2 & \\ t_{31} & \\ t_{32} & \end{matrix} \begin{matrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \end{matrix} = \begin{matrix} p_1 & \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \\ p_2 & \\ p_3 & \\ p_4 & \end{matrix} \\
 M_{32} &= M_2 + A^T U_{32} = \begin{matrix} p_1 & \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \\ p_2 & \\ p_3 & \\ p_4 & \end{matrix} + \begin{matrix} t_1 & \begin{bmatrix} -1 & 0 & 1 & 1 \end{bmatrix} \\ t_2 & \\ t_{31} & \\ t_{32} & \end{matrix} \begin{matrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \end{matrix} = \begin{matrix} p_1 & \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \\ p_2 & \\ p_3 & \\ p_4 & \end{matrix}
 \end{aligned}$$

In reality, places $p_2 : (X, M, Timer)$, $p_3 : (Y, PL_2)$ and $p_1 : (D, PL_3)$ are compounded

places in Petri nets. Therefore, the state of $P = \{p_1, p_2, p_3, p_4\}$ can be decomposed into the $P = \{PL_1, X, M, Timer, Y, PL_2, D, PL_3\}$ state, so $M_0^T(P) = [1 \ 0 \ 0 \ 0]$ can be transferred into $M_0^T(p) = [1 \ (0 \ 0 \ 0) \ (0 \ 0) \ (0 \ 0)]$. Similarly, $M_1^T(P)$, $M_2^T(P)$, $M_{31}^T(P)$ and $M_{32}^T(P)$ can be decomposed into places $[0 \ (1 \ 1 \ 1) \ (1 \ 1) \ (0 \ 0)]$, $[0 \ (1 \ 1 \ 1) \ (0 \ 0) \ (1 \ 1)]$, $[1 \ (0 \ 0 \ 0) \ (0 \ 0) \ (0 \ 0)]$ and $[1 \ (0 \ 0 \ 0) \ (0 \ 0) \ (0 \ 0)]$, respectively.

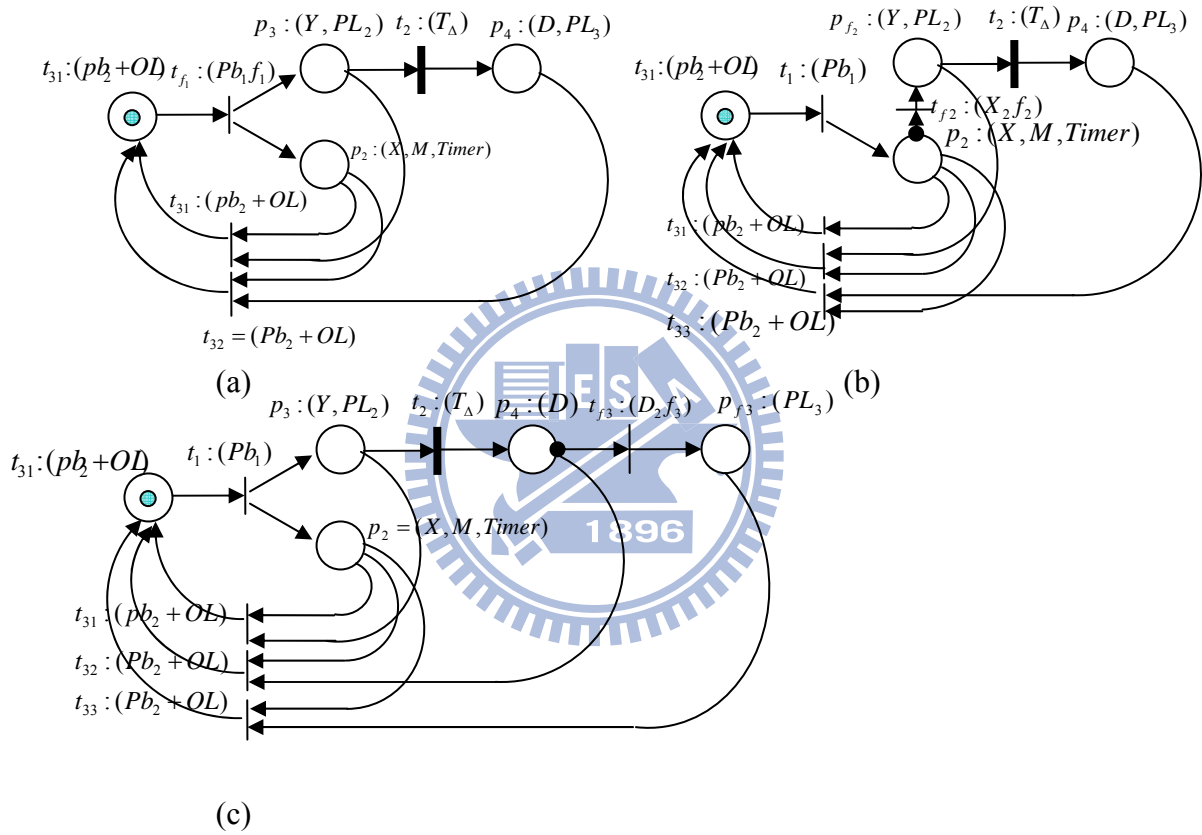


Fig. 3.12 Petri nets model (a) with fault f_1 , (b) with fault f_2 , (c) with fault f_3 in Fig.7.

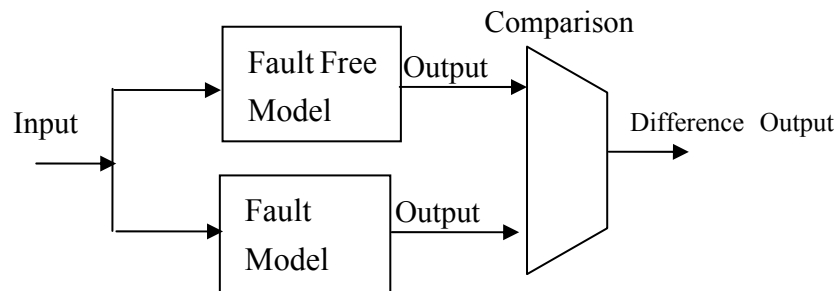


Fig. 3.13 Simulated fault free model and fault model.

3.3.4 Analysis and diagnosis of fault modeling

Assume that the faults f_1, f_2 and f_3 in the LDs are stuck at 0 (s-a-0), as Fig. 3.7 illustrates. The fault can then be modeled into Petri nets as shown in Fig. 3.12 (a), (b), and (c), respectively. In case 1, a transition $t_1:(pb_1)$ is fired since pb_1 is active. However, the transition $t_{f_1}:(pb_1f_1)=0$ cannot be fired since f_1 is stuck at 0. Similarly, in case 2, $t_{f_2}:(X_2f_2)=0$, $t_2:(X_2)=1$. In case 3, $t_{f_3}:(D_2f_3)=0$, $t_3:(D_2)=1$. For simple calculation of state equation, a control vector U_k contains the fault free Boolean equation and faulty Boolean equation of transitions, as denoted by $t_{f_i}:(\text{fault free} / \text{fault})=1/0$. Fig. 3.13 shows fault free model and fault model simulated structure.

A difference output vector (DOV) = fault free output vector - fault output vector. If a fault occurs then the difference output vector $\neq 0$. The fault is covered from a place of negative value to a place of positive value, and the faulty path flows through transition t_i in DOV.

Case1: Assume f_1 is s-a-0.

$$A_{f_1}^T = \begin{matrix} & t_1 & t_2 & t_{31} & t_{32} \\ \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{matrix} & \begin{bmatrix} -1 & 0 & 1 & 1 \\ 1 & 0 & -1 & -1 \\ 1 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \end{matrix}, M_0 = \begin{matrix} & p_1 \\ \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{matrix} & \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{matrix}, U_1 = \begin{matrix} & t_{f_1}:(pb_1 / pb_1f_1) \\ \begin{matrix} t_2 \\ t_{31} \\ t_{32} \end{matrix} & \begin{bmatrix} 1/0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{matrix}$$

$$M_{f_1} = M_0 + A_{f_1}^T U_1 = \begin{matrix} & t_1 & t_2 & t_{31} & t_{32} \\ \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{matrix} & \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 1 & 1 \\ 1 & 0 & -1 & -1 \\ 1 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1/0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1/0 \\ 1/0 \\ 1/0 \\ 0 \end{bmatrix} = \begin{matrix} & p_1 \\ \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{matrix} & \begin{bmatrix} 0/1 \\ 1/0 \\ 1/0 \\ 0 \end{bmatrix} \end{matrix}$$

DOV = $[-1 \ 1 \ 1 \ 0]^T$. The faulty area is covered from p_1 to p_2 and p_3 as Fig. 3.14

indicates, and the fault path flows through $t_1 : (pb_1)$. Thus, the fault is located between rung 1 and rung 3 in Fig. 3.7. In physical terms, this means the motor cannot start rotation.

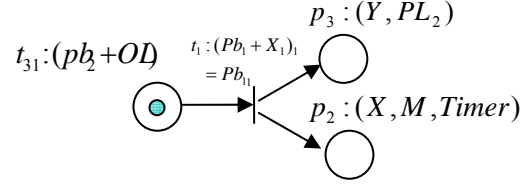


Fig. 3.14 the faulty area in case 1

Case2: Assume f_2 is s-a-0.

$$A_{f_2}^T = \begin{matrix} & t_1 & t_{f_2} \\ p_1 & \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \\ p_2 \\ p_{f_2} \end{matrix}; M_0 = \begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}; U_1 = \begin{matrix} t_1:(pb_1) \\ t_{f_2} \end{matrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}; U_1 = \begin{matrix} t_1 \\ t_{f_2}:(X_2/X_2f_2) \end{matrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$M_1(f_2) = M_0 + A_{f_2}^T U_1 = \begin{matrix} p_1 \\ p_2 \\ p_{f_2} \end{matrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{matrix} p_1 \\ p_2 \\ p_{f_2} \end{matrix} \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{matrix} p_1 \\ p_2 \\ p_{f_2} \end{matrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{matrix} p_1 \\ p_2 \\ p_{f_2} \end{matrix} \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} = \begin{matrix} p_1 \\ p_2 \\ p_{f_2} \end{matrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$M_2(f_2) = M_1 + C_{f_2}^T U_2 = \begin{matrix} p_1 \\ p_2 \\ p_{f_2} \end{matrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \begin{matrix} p_1 \\ p_2 \\ p_{f_2} \end{matrix} \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1/0 \end{bmatrix} = \begin{matrix} p_1 \\ p_2 \\ p_{f_2} \end{matrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \begin{matrix} p_1 \\ p_2 \\ p_{f_2} \end{matrix} \begin{bmatrix} 0 \\ 0 \\ 1/0 \end{bmatrix} = \begin{matrix} p_1 \\ p_2 \\ p_{f_2}:(Y, PL_2) \end{matrix} \begin{bmatrix} 0 \\ 1 \\ 1/0 \end{bmatrix}$$

DOV = $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. The faulty area is covered only in p_{f_2} as Fig. 3.12(b) indicates, and

the fault path flows through $t_{f_2} : (X_2)$. Thus, the fault is located between rung 2 and

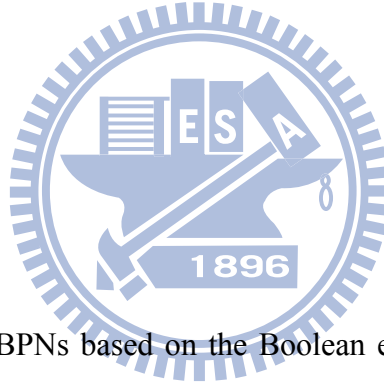
rung 5 in Fig. 3.7. In physical terms, this means the motor cannot run.

Case3: Assume f_3 is s-a-0.

$$A_{f_3}^T = \begin{matrix} & t_1 & t_2 & t_{31} & t_{32} & t_{33} & t_{f_3} \\ p_1 & \begin{bmatrix} -1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & -1 & -1 & -1 & 0 \\ 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \\ p_2 \\ p_3 \\ p_4 \\ p_{f_4} \end{matrix}; U_1 = \begin{matrix} t_1:(pb_1) \\ t_2 \\ t_{31} \\ t_{32} \\ t_{33} \\ t_{f_3} \end{matrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}; U_2 = \begin{matrix} t_1 \\ t_2:(T_\Delta) \\ t_{31} \\ t_{32} \\ t_{33} \\ t_{f_3} \end{matrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}; U_{f_3} = \begin{matrix} t_1 \\ t_2 \\ t_{31} \\ t_{32} \\ t_{33} \\ t_{f_3}:(D_2/D_2f_3) \end{matrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1/0 \end{bmatrix}$$

$$\begin{aligned}
M_0 = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_{f_3} \end{matrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} & ; & M_1 = M_0 + A_{f_3}^T U_1 = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_{f_3} \end{matrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} ; & & M_2 = M_1 + A_{f_3}^T U_2 = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_{f_2} \end{matrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} ; \\
M_{f_3} = M_2 + A_{f_3}^T U_{f_3} = & & \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_{f_3} : (PL_3) \end{matrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1/0 \end{bmatrix}
\end{aligned}$$

DOV = $[0 \ 0 \ 0 \ 0 \ 1]^T$. The faulty area is covered only in p_{f_3} as Fig. 3.12(c) indicates, and the fault path flows through $t_{f_3} : (D_2)$. Thus, the fault is located between rung 7 and rung 9 in Fig. 3.7. In physical terms, this means the indicator light PL_3 cannot light.



3.4. Summary

This chapter proposes the BPNs based on the Boolean equation, constructs a ladder diagram module and develops an abstract model to diagnose local faults in the LDs. The diagnostic process employs simple matrix manipulation and DOV to determine the faulty area for diagnosing the ladder diagram. This study also provides an example using composite transition, composite place, and relevant state to reduce complexity and increase readability of the Petri nets. The proposed methodology is useful and clear.

CHAPTER 4

Implementation of an ASIC for the Testing of a Ladder

Diagram

As described in the previous chapter, a BPNs model and an abstract model were constructed for the diagnosis of ladder diagrams. In this chapter, BPNs were used to solve experience-based testing and troubleshooting problems of sequence controllers in manufacturing systems. To describe the basic LDs and to propose a framework for LDs testing, the concept of integrated circuit testing was introduced during the construction of a fault-free model of LDs based on BPNs, as shown in Fig. 4.1. The developed model can directly generate test event sequences of LDs from the transition sequence of BPNs and can support the implementation of application-specific integrated circuits (ASICs). The BPNs constructs a model that aides in the troubleshooting of LDs and can be simulated using the state equation.

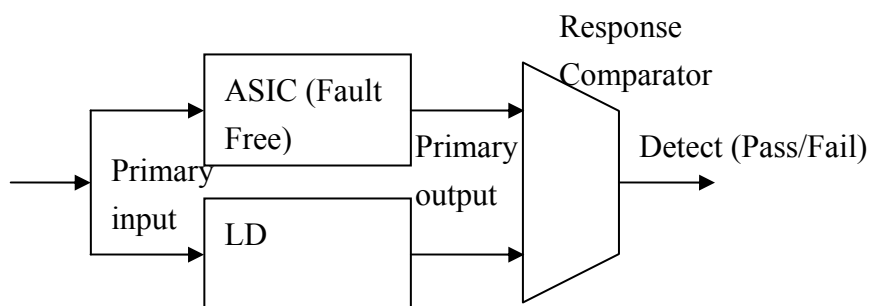


Fig. 4.1. Framework of LDs functional tests

4.1. Boolean Petri Net and Ladder Diagram Model

Carl Adam Petri proposed the Petri nets theory. Figure 2 shows the structure of Petri nets in a directed bipartite graph that consists of places, transitions, and arcs. A circle with a token represents the places. A bar that indicates the flow of tokens when the firing condition is satisfied, which represents the transition. Finally, a straight line that connects the place to the transition, or the transition to the place, denotes the arc, which indicates the flow of tokens in the direction of the arrow.

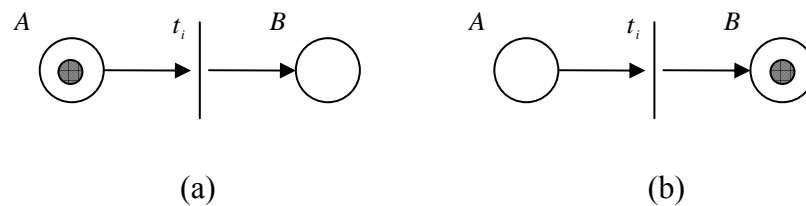


Fig. 4.2. (a) An example Petri nets. (b) A token moving from A to B in Fig. 4.2(a) after t_i firing.

4.1.1. Definition of Boolean Petri nets

The purpose of developing the BPNs model is that this model exhibits the implied logic property in LDs. The simplest way to represent LDs is by its Boolean equation. The approach proposed in this paper embeds the Boolean equation in a PN's transition to develop the BPNs model. This special transition is called the “Boolean transition.” Table 4.1 describes the BPNs model corresponding to the Boolean equation. Clearly, the BPNs model also matches the LDs. To map LDs into a Petri nets, the Petri nets must be extended. This extended Petri nets are called a Boolean Petri nets, which can be defined formally as

$$PN = (P, T, A, I, O, in, out, M_0) \quad (4-1)$$

Where $P = \{p_1, p_2, \dots, p_m\}$, $m \geq 1$, is a finite set of places representing the LDs action state. The places are associated with a component or a set of components (i.e., a compound component) such as the actuator output, relay coil, timer, counter solenoid, or source;

$T = \{t_1, t_2, \dots, t_n\}$, $n \geq 1$, is a finite set of transitions representing whether an event occurs or not. These transitions are always associated with a switch or a set of switches and are represented by Boolean equations or variables.

The switch can be a normal open (NO) switch or normal closed (NC) switch. The NO switch is also called an “a” contact, and the NC switch is also called a “b” contact, where $P \cap T = \Phi$ and $P \cup T \neq \Phi$.

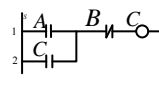
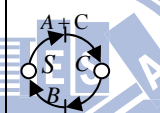
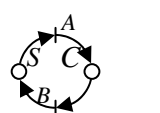
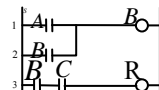
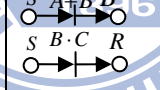
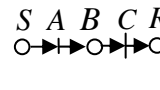
$A \subset (P \times T) \cup (T \times P)$ is a set of arcs (\rightarrow) consisting of input arcs $A_i(P \times T)$ and output arcs $A_o(T \times P)$. The weight of each directed arc in this paper is 1, and $A_i(P \times T)$ is defined as directed arcs from a place to a transition. Places are called input places, and transitions are called output transitions. The input arc is represented by a connected line as a channel of a token. $A_o(T \times P)$ is defined as directed arcs from a transition to a place, where the transition is called the input transition and the place is called the output place, and the output arc is represented by a connected line as a channel of a token. The arc may be a preservation arc ($\bullet \rightarrow$), where an input arc and an output arc exist simultaneously between the same place and transition; $I: T \times P \rightarrow N$ is an input function that defines the number of output arcs $A_o(T \times P)$, where $N = \{0, 1, 2, \dots\}$. $O: P \times T \rightarrow N$ is an output function that is defined as the number of input arcs $A_i(P \times T)$, where $N = \{0, 1, 2, \dots\}$; $in = \{in_1, in_2, \dots, in_n\}$ is a set of input switch, which is represented by a Boolean function or variable. A set of input switches is associated with a transition t_j and is denoted by $t_j = \{in\}$. The Boolean function or variable can be ‘1’, in which case the related transition t_j is allowed to fire if it is enabled, or it can be ‘0’, in which case the related transition t is not allowed to fire. $out = \{out_1, out_2, \dots, out_m\}$ is a set of output actuators, which is associated with a p_i and is denoted by $p_i = \{out\}$. $M_0(P)$ is the initial marking that uses a token to represent the place status.

A transition is **enabled** if the number of tokens at the place is larger than or equal to the number of input arcs. A transition is **firing** if the enabled transition is fired and

its transition states are true (i.e., the Boolean equation is true). When a transition fires, it moves the tokens from input places to output places along the input arcs and output arcs, as Fig. 4.2 illustrates. This action moves the token of place A to place B along directed arcs if transition t_i is firing. A marking is denoted as an m -vector, where m is the total number of place P , while $m(p_i)$ is represented by the number of tokens at place p_i (Murata et al., 1989).

To simulate the behavior of LDs, this approach changes a state or marking according to defined firing rules for the Boolean Petri nets model.

Table 4.1: Simplified BPNs

| Type | LDs | Boolean Equation | Before PNs | Simplified Boolean Equation | After PNs |
|------|---|------------------------------------|---|---|---|
| 1 |  | $C = (A + C)\bar{B}$ |  | $C = (A + C)\bar{B}$ $C(1 + \bar{B}) = A\bar{B}$ $C = A\bar{B}$ |  |
| 2 |  | $B = (A + B)$ $R = (B \cdot C)$ |  | $B = (A) = (A) \cdot (S)$ $R = (C) \cdot (B)$ |  |

4.1.2. Model of a Ladder Diagram

In a ladder diagram, a rung corresponding to a Boolean equation (BE) was introduced by David in 1995 (David et al., 1995). The Boolean equation associates every input variable (e.g., switch) and output variable (e.g., relay coil). In a Petri nets, the input variables are represented as an event. The output variables are represented as an event state, while an output variable is dependent on input variables. LDs rung can correspond to a Petri nets model, and the input variable of LDs may be simplified by its Boolean property. For example, $C = (A + C)\bar{B}$ can be simplified to $C = A\bar{B}$, as shown in Type 1 of Table 4.1, and a Petri nets corresponding to LDs can be simplified

by properties of enabling and firing. For example, in Type 2 of Table 4.1, a marking of place S is an enabled condition and a Boolean equation $B = A$ is the equivalent firing condition of rungs 1 and 2. A Boolean equation $R = (B \cdot C)$ is a firing condition of rung 3. Thus, the Boolean equation $B = A$ merges with the enable condition S , which is then given $B = (A) \cdot (S)$, and the Boolean equation $R = (B \cdot C)$ is separated into $R = (C) \cdot (B)$ and regarded as associated firing conditions with the enabling condition. Therefore, the property of firing and enabling can apply to the parallel LDs corresponding to series of Petri nets. Table 4.2 summarizes some typical LDs modules and their corresponding BPNs, where S is a pseudo source that can represent the ideal state of a relay coil, and composites and decomposites of BPNs are as shown in Table 4.3.

Table 4.2: Some LDs modules and their corresponding BE and BPNs models

| Modules | LDs | BE/BPNs | Modules | LDs | BE/BPNs |
|---------|-----|-------------------|---------|-----|---|
| Type1 | | $M = X$ | Type2 | | $M = X_1 + X_2$ $M = X_1 \cdot X_2$ $M_1 = M_2 = X$ |
| Type3 | | $\bar{M} = X$ | Type4 | | $\bar{M} = X_1 + X_2$ |

| | | | |
|--------------|---|---------------|---|
| <p>Type5</p> | <p>$M = A \cdot \bar{B}$</p> <p>$M = N = A \cdot \bar{B}$</p> | <p>Type6</p> | <p>$\bar{M} = X$ $N = X$</p> <p>$\bar{M} = T_\Delta$ $N = T_\Delta$</p> |
| <p>Type7</p> | <p>$\bar{M} = T_\Delta$ $N = T_\Delta$</p> | <p>Type8</p> | <p>$\bar{M} = T_\Delta$ $N = T_\Delta$</p> |
| <p>Type9</p> | <p>Timer = A $\bar{B} = T_\Delta$ $C = T_\Delta$</p> | <p>Type10</p> | <p>Timer = A $\bar{B} = T_\Delta$ $C = T_\Delta$</p> |

Table 4.3: Composites and decomposites of Boolean Petri nets

| LDs | Boolean Equation | BPNs | | |
|-----|---|-------------|-----------|-----------|
| | | Decomposite | Composite | Composite |
| | <p>$C=A$ $D=A$ $C=D$</p> | | | |
| | <p>$C=AB$</p> | | | |
| | <p>$C=A$ $C=B$ $C=A+B$</p> | | | |
| | <p>$C = A\bar{B}$ $D = C$</p> | | | |

4.1.3. State Equation

The firing definition easily shows that the token moves from state M_{k-1} to another state M_k by the k th firing, and U_k is a firing vector, which can be given in terms of the following matrix state equation for Petri nets (Murata et al., 1977).

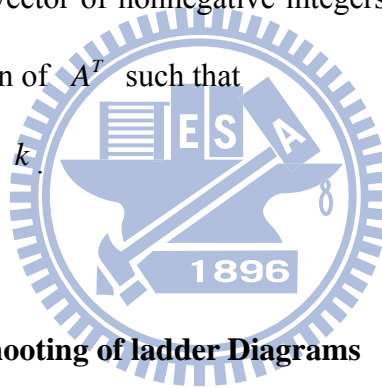
$$M_k = M_{k-1} + A^T U_k \quad (4-2)$$

Where U_k is called the firing vector, and A^T is called the incidence matrix for any given topological structure of Petri nets, defined by

$$A^T(p_i, t_j) = \begin{cases} -O_{ij}(p_i, t_j) \\ 0 \\ I_{ji}(t_j, p_i) \end{cases}, \text{ where } 1 \leq i \leq m, 1 \leq j \leq n. \quad (4-3)$$

Note that M_k must be a vector of nonnegative integers. The firing vector will then select an appropriate column of A^T such that

$$M_{k-1} + A^T U_k \geq 0 \text{ for each } k. \quad (4-4)$$



4.2. Testing and Troubleshooting of ladder Diagrams

In this section, we first introduce the concept of integrated circuit testing to describe a basic LDs and to generate the testing event sequence of a LDs using a BPNs model. The generated test event sequence can be applied to the testing and troubleshooting of the LDs, while we can use the BPNs to program the free-fault model and ASIC implementation.

4.2.1. Introduction of LDs testing (Lala et al., 2009)

A *failure* is said to have occurred in a ladder diagram circuit or system if it deviated from its specified behavior. A *fault* refers to a physical defect in a ladder diagram circuit. For example, a short in a normally open contact or a break in a normally closed contact is a physical defect. An *error* is usually the manifestation of a fault in

the ladder diagram circuit; thus, a fault may change the signal of a current in a ladder diagram circuit from the open (correct) to closed (erroneous) state or vice versa.

The most common model used for ladder diagram faults is the single stuck-at fault. It assumes that a fault in a ladder diagram rung results in one of its input or the output being fixed at either on (i.e., *stuck-at-on*) or off (i.e., *stuck-at-off*). A stuck-at-on fault implies the permanent closing of a rung in the ladder diagram circuit. A stuck-at-off fault implies the permanent opening of a rung in the ladder diagram circuit.

The inputs to the ladder diagram circuit are called the primary input. They are the only inputs can be applied to events in a Petri nets. This ability to apply an input event to the primary inputs of a Petri nets is known as *controllability*. The outputs from the ladder diagram are called primary outputs. The outputs can be observed in the effect of events occurring in the Petri nets. The ability to observe the response of a fault on an internal node via the primary outputs of a ladder diagram circuit is called *observability*.

In general, a test can detect more than one fault in a ladder diagram circuit, and when many tests in a set detect the same fault, it can be called a *dominance fault*. When many faults in the same set detect the tests, it can be called an *equivalent fault*. Thus, a major objective in test generation is to reduce the total number of faults to be considered by dominance and equivalent. For example, in a simple ladder diagram circuit shown in Fig. 4.3 (a) and its Boolean equation $C = A \cdot \bar{B}$ (Bender et al., 2008), its Boolean equation can be viewed with *AND* logic in an integrated circuit (IC), as shown in Figure 3(b), and with a BPNs model, as shown in Figure 3(c). Its true table is shown in Table 4.4. The equivalent sets for the simple ladder diagram circuit is $\{ A s-a-0, \bar{B} s-a-0, C s-a-0 \}$, and its fault dominance relations are $\{ C s-a-1, A s-a-0 \}$ and $\{ C s-a-1, \bar{B} s-a-1 \}$. The fault can be ignored if $\{ A s-a-0, \bar{B} s-a-0, C s-a-1 \}$. In other words, these test sets $\{A, \bar{B}\}$ are reduced $\{0,$

1}, {1, 1}, {1, 0}. Similarly, in the self-hold of the ladder diagram circuit, shown in Fig. 4.4(a), because its Boolean equation $C = (A+C) \cdot \bar{B}$ is equivalent to $C = A \cdot \bar{B}$ and because sometimes contact switches C are uncontrollable, these test sets $\{A, \bar{B}\}$ are reduced $\{0, 1\}, \{1, 1\}, \{1, 0\}$ as well. The $\{0, 1\}, \{1, 1\}, \{1, 0\}$ of the LDs test pattern correspond to no event occurrence (i.e., switch A and B is not pressed), and switch A event occurs (i.e., switch A is pressed) and switch B event occurs (i.e., switch B is pressed) for the BPNs, respectively.

Table 4.4: True table of a simple ladder diagram circuit

| A | \bar{B} | C (coil) | A s-a-1 | \bar{B} s-a-1 | C s-a-1 | A s-a-0 | \bar{B} s-a-0 | C s-a-0 |
|-----|-----------|------------|-----------|-----------------|-----------|-----------|-----------------|-----------|
| 0 | 0 | 0 | | | 1 | | | |
| 0 | 1 | 0 | 1 | | 1 | | | |
| 1 | 0 | 0 | | 1 | 1 | | | |
| 1 | 1 | 1 | | | | 0 | 0 | 0 |

4.2.2. Testing Event Sequence of a Ladder Diagram

Fault detection in a basic ladder diagram circuit, as shown in Fig. 4.3(a) and 4(a), is transferred by a Boolean Petri nets, and its test event sequence can be generated from the transition sequence of the transferred BPNs; thus, it is carried out by applying a sequence of test events and observing the resulting outputs. If the observed response is different than the expected response, a fault is present in the LDs. The aim of testing is to verify that functions in the ladder diagram are true or false using Fig. 4.1, which corresponds to troubleshooting, as shown in Table5.

In an m -input, there can be $2(m+1)$ stuck-at faults in the ladder diagram, but it can be an $(m+1)$ event sequence generated in the BPNs. Thus, the total number of single

stuck-at faults in a basic ladder diagram circuit is 6 ($=2 \times 3$), but the test event sequence can be $3(=2+1)$, generated using a BPNs. The test event sequence is *no event, an event occurs and B event occurs*, as shown in Table 4.5. The test event sequence can be calculated and verified using state equation as well:

$p_2 = t_{1-c} \cdot p_1 \Rightarrow 0/1 = t_{1-c} \cdot 1 \Rightarrow t_{1-c} = 0/1$, where $t_{1-c} = 0/1$ is represented as *A no event occur*, but *A* is fault-at-on.

$p_2 = t_{1-o} \cdot p_1 \Rightarrow 1/0 = t_{1-o} \cdot 1 \Rightarrow t_{1-o} = 1/0$, where $t_{1-o} = 1/0$ is represented as *an event occurs*, but from primary input *A* to primary output *C* is *fault-at-off*.

$p_1 = t_{2-c} \cdot p_2 \Rightarrow 1/0 = t_{2-c} \cdot 1 \Rightarrow t_{2-c} = 1/0$, where $t_{2-c} = 1/0$ is represented as *B event occurs*, but *B* is *fault-at-on*.

$$D^T = p_1 \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}, M_0 = p_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix}, U_{1-c} = t_{1-c} \begin{bmatrix} 0/1 \\ 0 \end{bmatrix}, U_{1-o} = t_{1-o} \begin{bmatrix} 1/0 \\ 0 \end{bmatrix}, U_{2-c} = t_{2-c} \begin{bmatrix} 0 \\ 1/0 \end{bmatrix};$$

$$M_{1-c} = M_0 + D^T \cdot U_{1-c} = p_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0/1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0/-1 \\ 0/1 \end{bmatrix} = \begin{bmatrix} 1/0 \\ 0/1 \end{bmatrix}$$

$$M_{1-o} = M_0 + D^T \cdot U_{1-o} = p_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1/0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1/0 \\ 1/0 \end{bmatrix} = \begin{bmatrix} 0/1 \\ 1/0 \end{bmatrix}$$

$$M_{2-c} = M_1 + D^T \cdot U_{2-c} = p_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1/0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1/0 \\ -1/0 \end{bmatrix} = \begin{bmatrix} 1/0 \\ 0/1 \end{bmatrix}$$

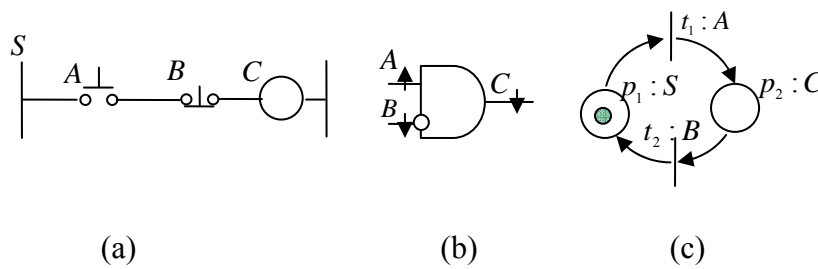


Fig. 4.3. (a) A basic ladder diagram circuit, (b) Corresponding to *AND* logic and (c) Corresponding to the BPNs model

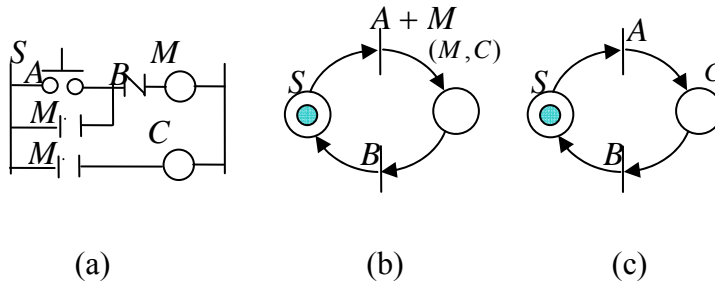


Fig. 4.4. (a) Self-hold of a ladder diagram, (b) Corresponding to BPNs model and (c) Simplified BPNs model

Table 4.5: Test events sequence

| Initial state | fault free | Fault | $A\bar{B}C$ | Firing effect | Test Event Sequence | Troubleshooting |
|---------------|------------|-------|-------------|---|----------------------|---|
| | | | 0 1 0/1 | The token was propagated to next position before A is firing. | <i>No event</i> | Please check push bottom A whether stuck at on or not. |
| | | | 1 1 1/0 | The token can not deposited to next position when A is firing (i.e., A is pressed event occur). | <i>A event occur</i> | Please check one of element A, B, C and interconnected line whether stuck at off. |
| | | | 1 0 0/1 | The token can be not propagated to next position when B is firing (i.e., B is pressed event occur). | <i>B event occur</i> | Please check push bottom B whether stuck at on or not. |

4.2.3. HDL Program

The basic ladder diagram can be implemented by HDL. The HDL program is carried out according to positions, transitions and token flowing in the BPNs, as shown in Table 4.6.

Table 4.6: The HDL code of a basic ladder diagram

| Number | I. THE HDL CODE OF BASIC LADDER DIAGRAM |
|--------|--|
| 1 | library ieee; |
| 2 | use ieee.std_logic_1164.all; |
| 3 | entity PLC is |
| 4 | port (Reset,CLK,A, B : in std_logic; |
| 5 | C : out std_logic); |
| 6 | end PLC; |
| 7 | architecture behave of LD is |
| 8 | type STATE_TYPE is (p1, p2); |
| 9 | signal present_state, next_state : STATE_TYPE; |
| 10 | begin |
| 11 | token_flow: process (Reset,present_state) begin |
| 12 | if Reset = '1' then next_state <= p1; |
| 13 | C = '0'; |
| 14 | case present_state is |
| 15 | when p1 => |
| 16 | if A = '1' then next_state <= p2; |
| 17 | else next_state <= p1; |
| 18 | end if; C <= '1'; |
| 19 | when p2 => |
| 20 | if B = '1' then next_state <= p1; |
| 21 | else next_state <= p2; |
| 22 | end if; C <= '0'; |
| 23 | end case; |
| 24 | end process token_flow; |
| 25 | state_clocking: process (CLK) begin |
| 26 | if CLK'EVENT and CLK = '1' then present_state <= next_state; |
| 27 | end if; |
| 28 | end process state_clocking; |
| 29 | end behave; |

4.3. Application Example

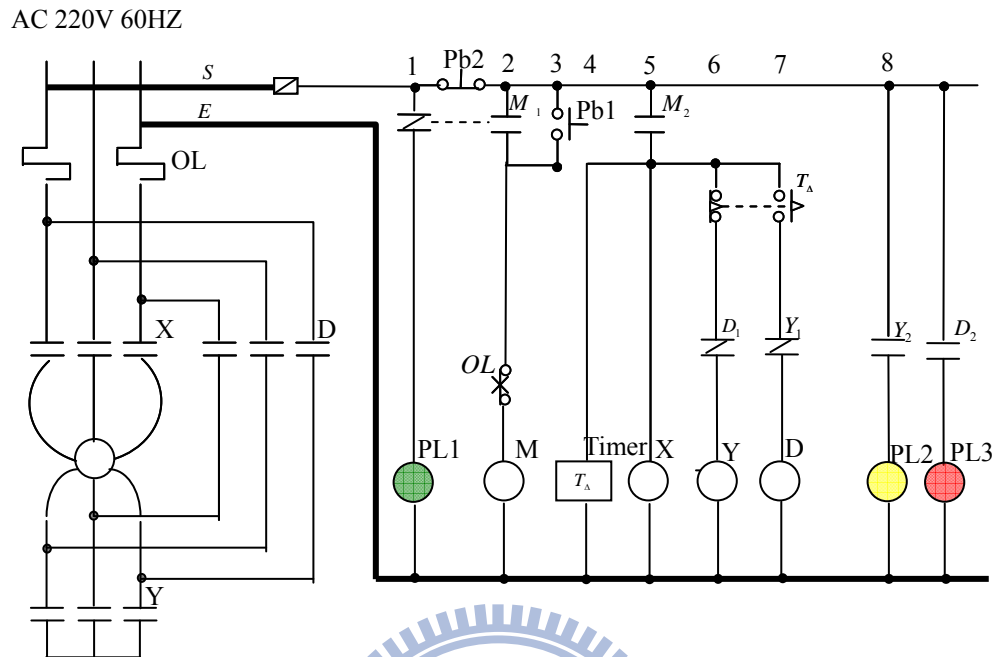














Fig. 4.5 Control circuit of a Y-Δ starting motor

Table 4.7: Descriptions of symbols

| Symbol | Description | Symbol | Description |
|--|---------------------------|---|----------------------------|
|  PL1 | Indicator light of green |  | “b” contact of Push bottom |
|  PL2 | Indicator light of yellow |  | “a” contact of Push bottom |
|  PL3 | Indicator light of red |  | “a” contact of relay |
|  | Relay |  | “b” contact of relay |
|  T _Δ | Timer |  | “a” contact of timer |
| 1~9 | rung number |  | “b” contact of timer |
| | |  | “b” contact of over load |

This section illustrates a practice example. The controller is a LDs circuit. This

circuit can be modeled by BPNs and is simplified to obtain a fault-free model using Table 4.1 of the preceding section. The system fault can be diagnosed by the difference between the LDs response and the ASIC response (i.e., the fault-free model).

To start a three-phase motor, a LDs controller uses a type of Y- Δ starting to limit the starting current, as shown in Fig. 4.5 and the symbol descriptions in Table 4.7. In the LDs controller, the bottom Pb_1 is a control relay coil M , Y and active timer coil. The motor enters the starting state when NO contacts of M and Y are turned on. Next, the relay coil Y turns off after delay time T_{Δ} , and the motor returns to the normal state when the relay coil Y turns off and relay coil D turns on. Finally, the motor stops if the bottom Pb_2 is pushed or the current is overload. This LDs controller can be specified as follows:

Step 1) The motor is commanded to start (Pb_1).

Step 2) The motor starting time is T_{Δ} .

Step 3) The motor is commanded to stop (Pb_2).

Step 4) The motor will stop if the current is overloaded.

The implicit specification is as follows:

The relay coil D and relay coil Y are mutually exclusive.

4.3.1 Fault Free Model

The transformation from the ladder diagram (in parallel) to the fault free model (in series) is based on the following steps:

Step 1) A rung or compound rung of the LDs is converted to a Boolean Petri nets module using Table I or the Boolean equation. The LDs controller then assembles Boolean Petri nets modules, as Fig. 4.6(a) shows, where (1), (2) ... and (9) correspond to the number of LDs rungs.

Step 2) A Boolean Petri nets can be given after eliminating the redundant or pseudo places (i.e., the S place), as Fig. 4.6(b) illustrates.

Step 3) A fault-free model can be obtained according to dominance relation reducing some places (in this case, an abstract model reduces place p_2) and eliminating some redundant elements (i.e., the coil of time or auxiliary relay), as Fig. 4.6(c) shows. Using the fault-free model to implement the ASIC circuit is shown in Fig. 4.6(d), and simulated results are shown in Fig. 4.6(e). The generated test pattern is $t_1(\text{push } pb_1)$ - $t_2(\text{push } T_\Delta)$ - $t_3(\text{push } Pb_2 + OL)$ and $t_1, t_2, t_3(\text{ideal state})$.

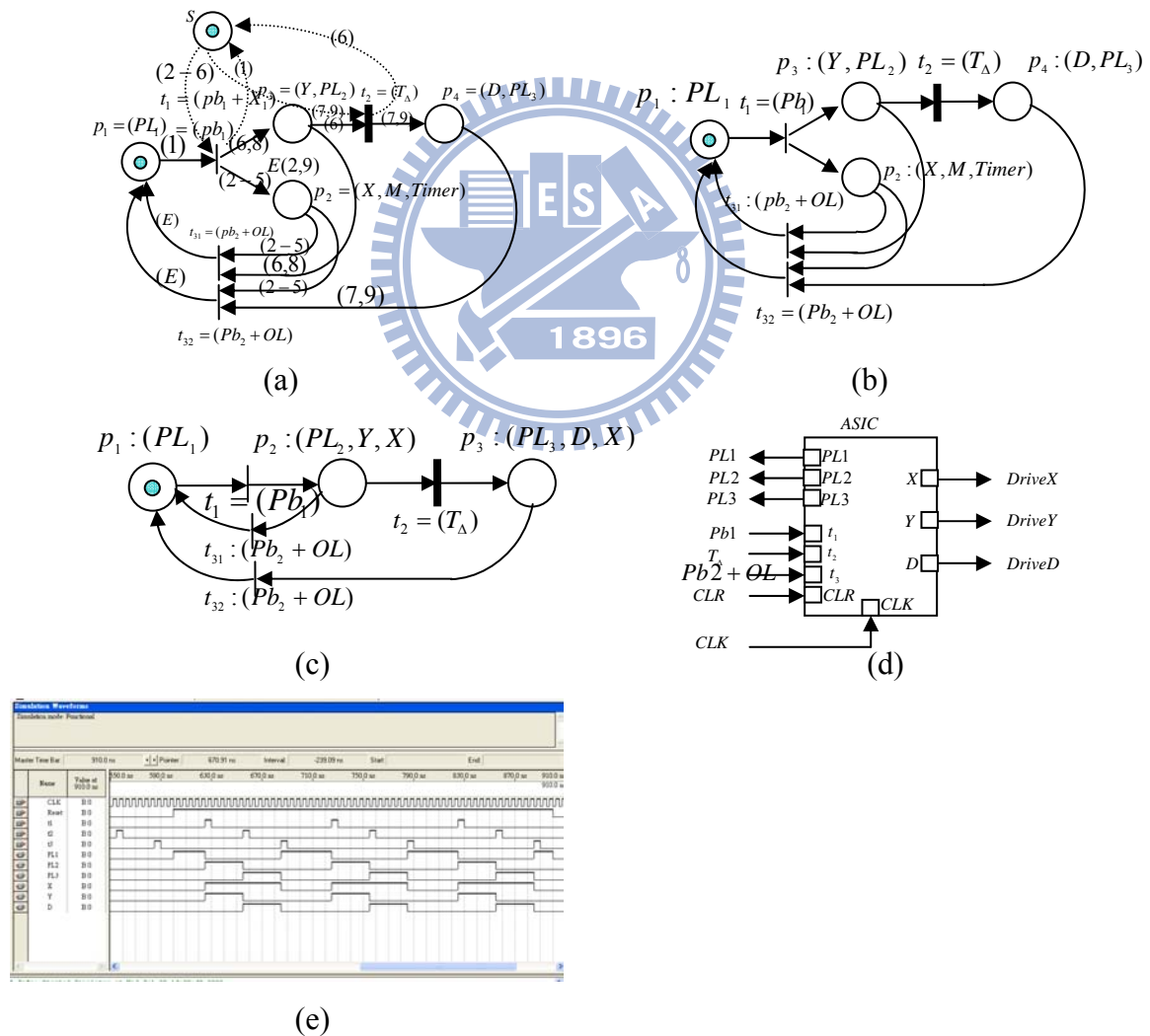


Fig. 4.6(a) BPNs model of a LDs controller, (b) Equivalent diagram of Fig.6 (a), (c) Simplified model of Fig. 4.6(b), (d) ASIC diagram, and (e) Simulation result.

4.3.2. Testing Event Sequence of a Motor Starting LDs and Troubleshooting

The test event sequence of the control ladder diagram can be generated in turn with the BPNs diagram, as shown in Table 4.8. The test event sequence is applied to the ladder diagram and ASIC to detect true or false of the LDs, as shown in Fig. 4.1, and it corresponds to troubleshooting, as shown in Table 4.8.

Table 4.8: Test event sequence and troubleshooting of motor starting LDs

| Test Event Sequence | No event occur | Pb_1 event occur | T_A event occur | Pb_2 event occur |
|---------------------|---|--|---|--|
| Troubleshooting | Please check push bottom Pb_1 whether stuck at on or not. | Please check element <i>from</i> Pb_1 to Y interconnected line whether stuck at off. | Please check element <i>from</i> T_A to D and interconnected line whether stuck at off. | Please check push bottom B whether stuck at on or not. |

4.3.3. HDL Program

The control ladder diagram of Y- Δ starting can be implemented by HDL. The HDL program is shown in Table 4.9, and its implementation and simulation are shown in Fig. 4.6(d) and (e), respectively.

Table 4.9: The HDL code of motor start action

| Number | II. THE HDL CODE OF MORTOR STARTING LADDER DIAGRAM |
|--------|--|
| 1 | library ieee; |
| 2 | use ieee.std_logic_1164.all; |
| 3 | entity PLC is |
| 4 | port (Reset,CLK,t1, t3 : in std_logic; |
| 5 | PL1, PL2, PL3, X, Y, D : out std_logic;); end PLC; |
| 6 | architecture behave of piston is |
| 7 | type STATE_TYPE is (p0, p1, p2); |
| 8 | signal present_state, next_state : STATE_TYPE; |

```

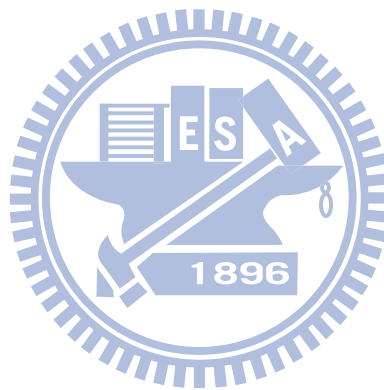
9      signal t2 : BIT;
10     begin
11     token_flow: process (Reset,present_state) begin
12     if Reset = '1' then next_state <= p0;
13         PL1='1'; PL2='0'; PL3='0'; X='0';Y='0'; D='0';
14     case present_state is
15     when p0 =>
16         if t1 = '1' then next_state<= p1;
17         else next_state<= p0;
18         end if; PL1<='0'; PL2<='1'X<='1';Y<='1';
19         wait for 5sec;
20         t2 <= '1';
21     when p1 =>
22         if t2 = '1' then next_state <= p2;
23         else next_state <= p1;
24         end if; PL2<='0'; PL3<='1'; Y<='0';X<='1';D<='1';
25     when p2 =>
26         if t3 = '1' then next_state <= p0;
27         else next_state <= p2;
28         end if; PL3<='0'; PL1<='1'; PL3<='0';
29         X<='0'; D<='0';
30     end case;
31     end process token_flow;
32     state_clocking: process (CLK) begin
33         if CLK'EVENT and CLK = '1' then present_state <= next_state;
34         end if;
35     end process state_clocking;
36 end behave;

```

4.4. Summary

This paper shows a solution for the experience-based testing and troubleshooting problem of LDs. We proposed a method for constructing a fault-free model, supporting the implementation program of ASIC and testing event sequences from BPNs. The testing problem was transferred to the determination of whether an event occurred or not. If an event does not occur in the primary input then the primary output is a have not response; likewise, if a have event occurs in the primary input,

then the primary output corresponds to response and the LDs detects a fault. Finally, an example of a motor start LDs was represented graphically as a fault-free model, providing a direct way to convert LDs to HDL and generate test an event sequence, while demonstrating this usable approach. In the future, we plan to apply this approach to more complicated systems and develop BPNs directly applied to the design of PLC implementation.



CHAPTER 5

The Testing, Diagnosis and Implementation of Logic Controllers

In previous chapters, a BPNs application that can be remotely diagnosed and monitored was developed. Although the proposed model can solve experience-based testing and troubleshooting problems in sequence controllers of manufacturing systems, sequence controllers are often designed with different types of LDs. Thus, the transfer of LDs to BPNs is difficult, and a systematic approach for the design of sequence controllers based on BPNs must be developed. Moreover, the BPN-directed application must be able to be remotely diagnosed and monitored. In this thesis, a method based on IDEF0, BPNs and TPL was developed to validate the system and to implement traditional PLCs. The proposed method can generate test event sequences for the solution of experience-based testing and troubleshooting problems in sequence controllers.

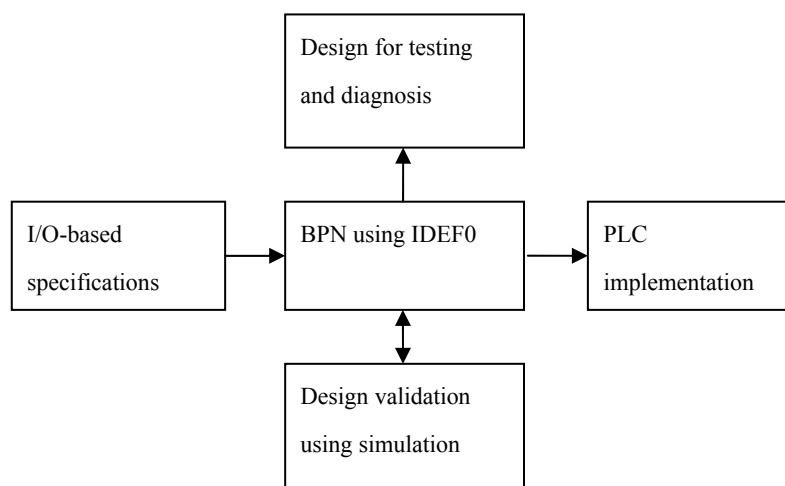


Fig. 5.1. Extension of the implementation scheme for Petri net-based controllers by

Taholakian et al. (1997).

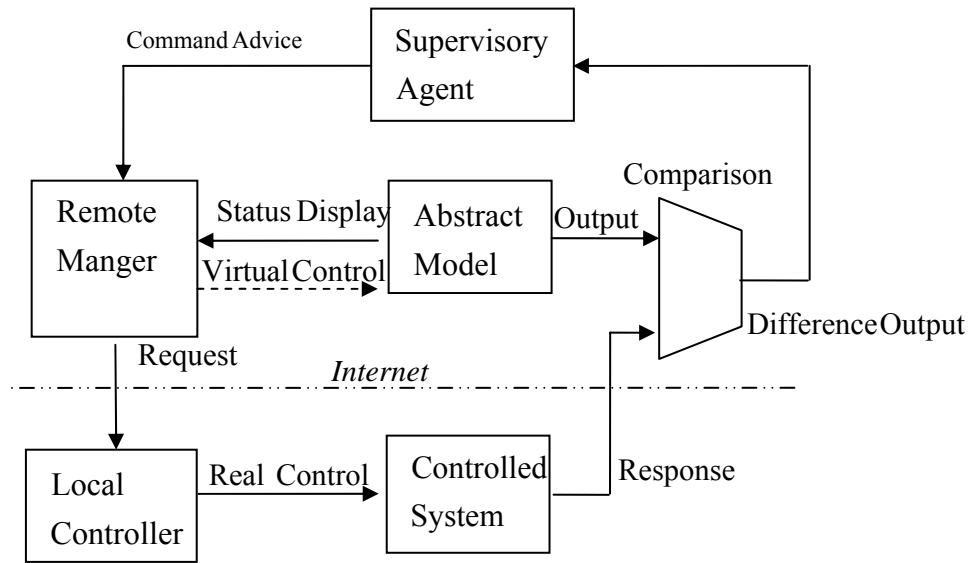


Fig. 5.2. Proposed hierarchical control (by Lee et al., (2004)).

5.1. Boolean Petri nets

In this section, define BPNs and describe the state equation of BPNs. The BPNs can be directly modeled from a specification of the logic controller or by employing IDEF0. The state equation can be used to simulate design validation and the constructed abstract model (as shown in Fig. 5.2) via an incidence matrix.

5.1.1 Definition of BPNs

The BPNs (Tsai et al., 2010) can be defined formally as

$$PN = (P, T, A, I, O, in, out, M_0) \quad (5-1)$$

Where $P = \{p_1, p_2, \dots, p_m\}$, $m \geq 1$ is a finite set of places that are associated with the output actuator; $T = \{t_1, t_2, \dots, t_n\}$, $n \geq 1$, is a finite set of transitions that are associated with input sensors; $P \cap T = \Phi$; and $P \cup T \neq \Phi$. $A \subset (P \times T) \cup (T \times P)$ is a set of arcs (\rightarrow) consisting of input arcs $A_i(P \times T)$ and output arcs $A_o(T \times P)$. The weight of each directed arc in this chapter is 1, and $A_i(P \times T)$ is defined as a directed arc from a place

to a transition. Places are called input places, and transitions are called output transitions. The input arc is represented by a connected line as a channel of a token. $A_o(T \times P)$ is defined as a directed arc from a transition to a place, where the transition is called the input transition and the place is called the output place. The output arc is represented by a connected line as a channel of a token. $I: T \times P \rightarrow N$ is an input function that defines the number of output arcs $A_o(T \times P)$, where $N = \{0,1,2,\dots\}$. $O: P \times T \rightarrow N$ is an output function that is defined as the number of input arcs $A_i(P \times T)$, where $N = \{0,1,2,\dots\}$; $in = \{in_1, in_2, \dots, in_n\}$ is a set of input sensors that is associated with a transition t_j and is denoted by $t_j: in$. The term $out = \{out_1, out_2, \dots, out_m\}$ is a set of output actuators that is associated with a p_i and is denoted by $p_i: out$. $M_0(P)$ is the initial marking that uses a token to represent the place status.

A transition is **enabled** if the number of tokens at the place is equal to or larger than the number of input arcs. An enabled transition is **firing** when an input sensor event occurrence associated with the enabled transition moves the tokens from input places to output places along the input arcs and output arcs. A marking is denoted as an m -vector, where m is the total number of places P , while $m(p_i)$ is represented by the number of tokens at place p_i .

5.1.2 State equation

The firing definition easily shows that the token moves from state M_{k-1} to another state M_k by the k th firing, and U_k is a firing vector that can be given in terms of the following matrix state equation for Petri nets (Lee et al., 2000):

$$M_k = M_{k-1} + A^T U_k, \quad (5-2)$$

Where U_k is the firing vector and A^T is the corresponding abstract model called the incidence matrix for any given topological structure of Petri nets, defined by

$$A^T(p_i, t_j) = \begin{cases} -O_{ij}(p_i, t_j) \\ 0 \\ I_{ji}(t_j, p_i) \end{cases}, \text{ where } 1 \leq i \leq m, 1 \leq j \leq n. \quad (5-3)$$

Note M_k must be a vector of nonnegative integers. The firing vector will then select an appropriate column of A^T such that

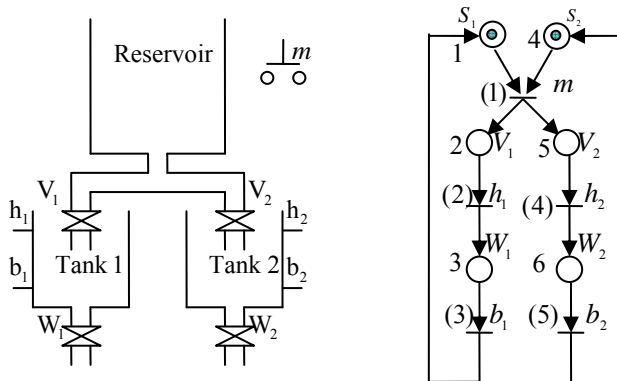
$$M_{k-1} + A^T U_k \geq 0 \text{ for each } k \quad (5-4)$$

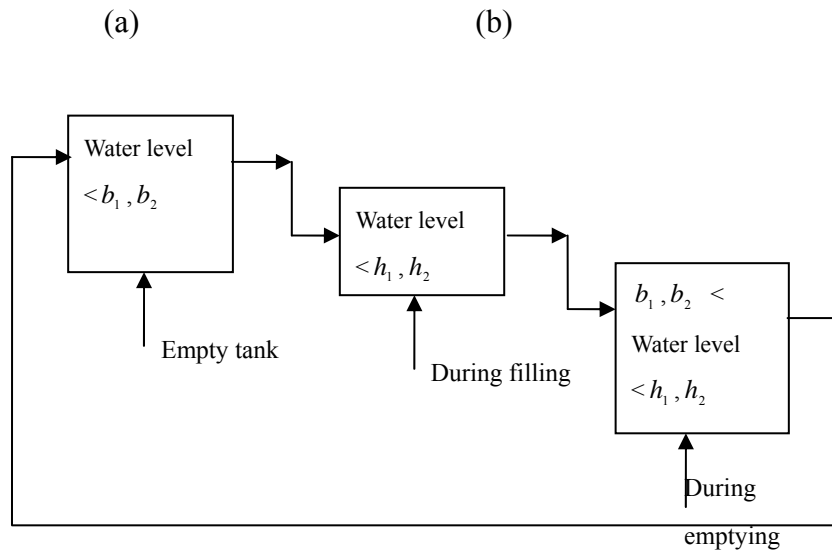
5.2. Constructing Boolean Petri Net and Implementation

In this section, construct a BPNs model from a specification of the system and map it to the PLC code based on the RLL, the LLD or the HDL.

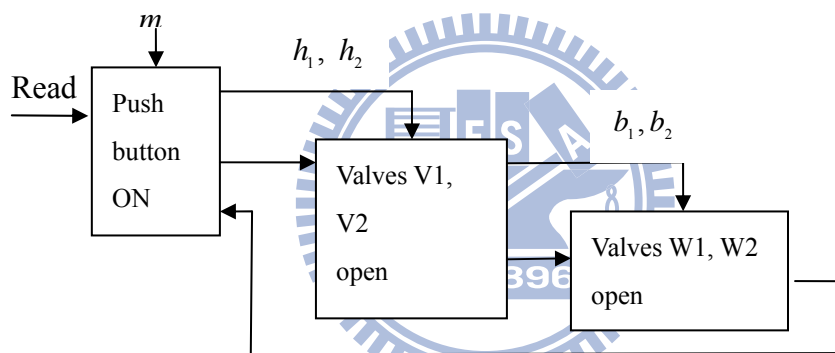
5.2.1 System description

An example of a tank filling is provided to describe the BPNs design stage for directly constructing a model from system specifications or IDEF0. The tank filling shown in Fig. 5.3(a) is redrawn from David's paper (David 1995). A reservoir provides water to tank 1 and tank 2. The tanks are modeled in three states: empty, during filling and during emptying. The initial state of the model is an empty tank (i.e., the water level of the tank is lower than b_1 and b_2). Valves V_1 and V_2 will be open when push button m is pressed. Water from the reservoir flows into tank 1 and tank 2 until the tanks are full of water (i.e., the water levels of the tanks are higher than h_1 and h_2). Valves W_1 and W_2 are then opened after the tank is filled until both tanks are empty.





(C)



(d)

Fig. 5.3. (a) Filling tank, (b) BPNs model of filling tank, (c) Material flows of IDEF0 and (d) Information flows of IDEF0.

5.2.2 Constructing the BPNs model

This BPNs model is explained in Fig. 5.3(b). The labels 1 to 6 represent steps, i.e., components of states. At the initial time, the steps in the set $\{1, 4\}$ are active. Next, transition (1), which follows these steps, can be fired as soon as event m associated with (1) occurs. After this firing, steps 2 and 5 are active. When step 2 is active, the output $v_1 = 1$. When step 2 is active, transition (2) can be fired if the h_1 event has occurred, and so on. The concurrency is explicitly represented in this model. Steps 1,

2 and 3 correspond to the states of tank 1 (empty, during filling, and during emptying, respectively), and steps 4, 5 and 6 correspond to the states of tank 2.

IDEF0 (FIPS 183) is an activity-oriented model approach that represents the activities performed in a system using ordered sets of boxes, as shown in Fig. 5.4. The boxes are input-control-output mechanisms. The activity may be a decision-marking, a material-conversion, or an information-conversion activity (Santarek et al., 1998). The information flow represents system activities and their interrelationships. It is transformed into a dynamic BPNs model based on the following steps:

1. The input and output commands of the activity box in the information flow diagram are transformed into input and outputs places in the BPNs, respectively.
2. The control signals of the sensor reading are transformed into transitions in the BPNs.
3. The initial token of the BPNs are set according to the initial condition of the system.

The IDEF0 approach can be used to design the tank filling system, as shown in Figs. 3(c) and (d). The BPNs model result is shown in Fig. 5.3(b).

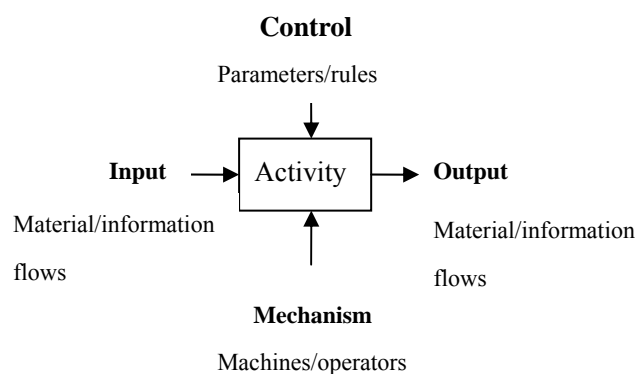


Fig. 5.4. The IDEF0 scheme (Lee et al. 2005).

5.2.3 BPNs mapping to implementation

To convert the BPNs model into PLC code for controller implementation, a direct mapping was used and is shown in Table 5.1. In the initial conditions, a token is located at place p_1 , which is represented as p_1 active. The token then flows to place p_2 when the sensor input is *on*; the sensor input *on* is associated with transition $t_1 : X_1$ firing. The active output device is assigned to the place p_2 active. According to the properties of being active and firing, the BPNs can be represented by the Boolean equation $p_2 = t_1 \cdot p_1$, which is equivalent to PLC code (Lee et al. 2005).

Table 5.1: Mapping the BPNs to PLC code

| BPNs | RLL | LLD | HDL |
|------|-----|-----|---|
| | | | <pre> when p1 => if t1 = '1' then next_state <= p2; else next_state <= p1; end if; Y1<='0', Y2<='1'; </pre> |
| | | | <pre> if t2 = '1' then next_state <= p0; else next_state <= p_k; end if; Y<='0'; </pre> |

5.3. Testing and Troubleshooting

In this section, introduce the concept of integrated circuit testing (Lala 2009) to describe basic LDs for the corresponding BPNs and to generate the testing event sequence of a BPNs model. The generated test event sequence can be applied for the testing and troubleshooting of the designed controller (i.e., local controller, as in Fig. 5.2).

5.3.1. Introduction of LDs testing

For example, for a BPNs model like the one shown in Fig. 5.5(a), with its corresponding LDs circuit shown in Fig. 5.5(b) and the Boolean equation $C = A \cdot \bar{B}$ (Bender et al. 2008), the Boolean equation can be viewed with *AND* logic in an integrated circuit (IC). Its true table is shown in Table 5.2. The equivalent set for the LD circuit is $\{A_{s-a-0}, \bar{B}_{s-a-0}, C_{s-a-0}\}$, and its fault dominance relations are $\{C_{s-a-1}, A_{s-a-0}\}$ and $\{C_{s-a-1}, \bar{B}_{s-a-1}\}$. The fault can be ignored if $\{A_{s-a-0}, \bar{B}_{s-a-0}, C_{s-a-1}\}$. In other words, these test sets $\{A, \bar{B}\}$ are reduced to $\{0, 1\}$, $\{1, 1\}$, and $\{1, 0\}$. The $\{0, 1\}$, $\{1, 1\}$, and $\{1, 0\}$ of the LDs test pattern correspond to no event occurrence (i.e., switch *A* and *B* are not pressed), switch *an* event occurs (i.e., switch *A* is pressed) and switch *B* event occurs (i.e., switch *B* is pressed) for the BPNs, respectively.

Table 5.2: True table of a simple LDs circuit

| <i>A</i> | \bar{B} | <i>C</i> (coil) | <i>A</i> s-a-1 | \bar{B} s-a-1 | <i>C</i> s-a-1 | <i>A</i> s-a-0 | \bar{B} s-a-0 | <i>C</i> s-a-0 |
|----------|-----------|-----------------|----------------|-----------------|----------------|----------------|-----------------|----------------|
| 0 | 0 | 0 | | | 1 | | | |
| 0 | 1 | 0 | 1 | | 1 | | | |
| 1 | 0 | 0 | | 1 | 1 | | | |
| 1 | 1 | 1 | | | | 0 | 0 | 0 |

5.3.2. Testing event sequence

Fault detection in a self-holding LDs, as shown in Fig. 5.5(b), it is mapped by a BPNs as shown in Fig. 5.5(a) and its test event sequence can be generated from the transition sequence of the BPNs model; thus, it is carried out by applying a sequence of test events and observing the resulting outputs. If the observed response differs from the expected response, a fault is present in the LDs. The aim of testing is to verify that functions in the LDs are true or false, as shown in Fig. 5.2, which

corresponds to troubleshooting, as shown in Table 5.3.

In an m -input system, there can be $2(m+1)$ stuck-at faults in the LDs, but there can be an $(m+1)$ event sequence generated in the BPNs. Thus, the total number of single stuck-at faults in a basic LDs circuit is $6 (=2 \times 3)$; however, the test event sequence can be equal to $3 (=2+1)$, generated using a BPNs. The test event sequence is *no event, an event occurs and B event occurs*, as shown in Table 5.3. The test event sequence can be calculated and verified using the Boolean equation and the state equation, respectively.

$p_2 = t_{1-c} \cdot p_1 \Rightarrow 0/1 = t_{1-c} \cdot 1 \Rightarrow t_{1-c} = 0/1$, where t_{1-c} (fault free/fault) = $0/1$ is represented as *A no event occurs*, but *A* is fault-at-on.

$p_2 = t_{1-o} \cdot p_1 \Rightarrow 1/0 = t_{1-o} \cdot 1 \Rightarrow t_{1-o} = 1/0$, where $t_{1-o} = 1/0$ is represented as an *event occurs*, but from primary input *A* to primary output *C* is *fault-at-off*.

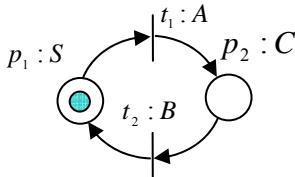
$p_1 = t_{2-c} \cdot p_2 \Rightarrow 1/0 = t_{2-c} \cdot 1 \Rightarrow t_{2-c} = 1/0$, where $t_{2-c} = 1/0$ is represented as *B event occurs*, but *B* is *fault-at-on*.

$$D^T = p_1 \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}, M_0 = p_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix}, U_{1-c} = t_{1-c} \begin{bmatrix} 0/1 \\ 0 \end{bmatrix}, U_{1-o} = t_{1-o} \begin{bmatrix} 1/0 \\ 0 \end{bmatrix}, U_{2-c} = t_{2-c} \begin{bmatrix} 0 \\ 1/0 \end{bmatrix};$$

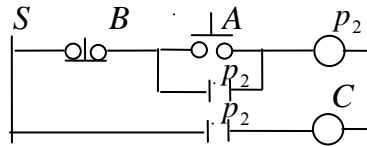
$$M_{1-c} = M_0 + D^T \cdot U_{1-c} = p_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0/1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0/-1 \\ 0/1 \end{bmatrix} = \begin{bmatrix} 1/0 \\ 0/1 \end{bmatrix},$$

$$M_{1-o} = M_0 + D^T \cdot U_{1-o} = p_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1/0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1/0 \\ 1/0 \end{bmatrix} = \begin{bmatrix} 0/1 \\ 1/0 \end{bmatrix},$$

$$M_{2-c} = M_1 + D^T \cdot U_{2-c} = p_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1/0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1/0 \\ -1/0 \end{bmatrix} = \begin{bmatrix} 1/0 \\ 0/1 \end{bmatrix}.$$



(a)



(b)

Fig. 5.5. (a) BPNs model (b) Corresponding to a self-hold on LDs.

Table 5.3: Test event sequence and troubleshooting

| Initial state | fault free | Fault | $A \bar{B} C$ | Firing effect | Test Event Sequence | Troubleshooting |
|---------------|------------|-------|---------------|---|-----------------------|---|
| | | | 0 1 0/1 | The token was propagated to the next position before A fires. | <i>No event</i> | Please check push button A to determine whether it is stuck at on or not. |
| | | | 1 1 1/0 | The token cannot be deposited in the next position when A is firing (i.e., A is pressed event occurs). | <i>A event occurs</i> | Please check one of the elements A, B, or C and the interconnected line to determine whether they are stuck at off. |
| | | | 1 0 0/1 | The token cannot be propagated to the next position when B is firing (i.e., B is pressed event occurs). | <i>B event occurs</i> | Please check push button B to determine whether it is stuck at on or not. |

5.4. An example of stamping Process

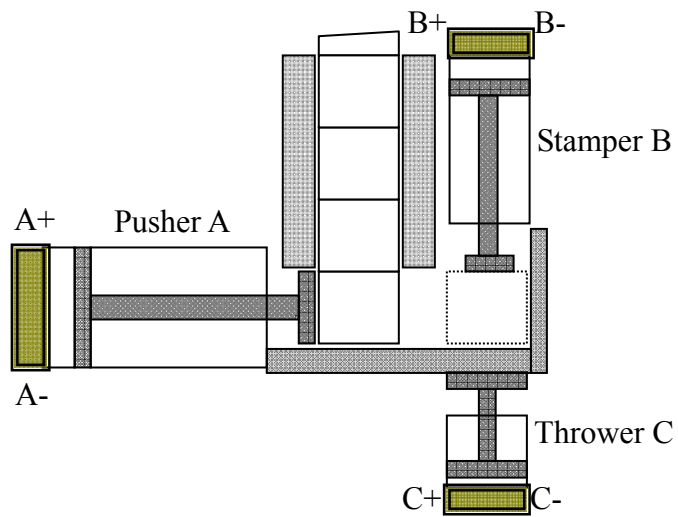
To demonstrate the viability of the developed approach, a stamping process application was investigated.

5.4.1. System description

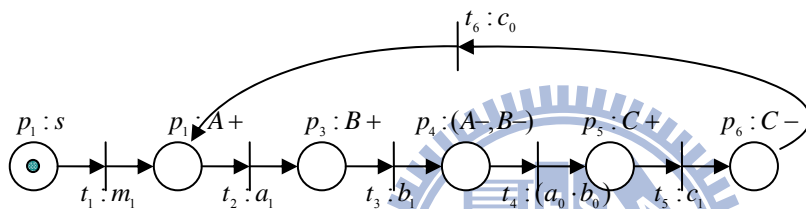
As shown in Fig. 5.6(a), a stamping system (Lee et al. 2005) consists of three cylinders. Each cylinder has two normal open limit switches. In terms of input sensors, the stamping system have a push button m_1 and 6 limit switches: $a0, a1, b0, b1, c0$ and $c1$. For output actuators, there are 6 solenoid valves: $A+, A-, B+, B-, C+$ and $C-$, where the + and – signs indicate a piston performing forward strokes and return strokes, respectively. In the stamping process, pusher A moves the work piece onto the worktable from a store. The work piece is then stamped by stamp B and afterwards is ejected by a thrower C . Thus, the work process sequence of the system is $A+, B+, \{A-, B-\}, C+$ and $C-$, where $\{A-, B-\}$ represents two concurrent actions as the pistons of both pusher A and stamper B perform return strokes simultaneously.

5.4.2 Construction of the BPNs model and mapping of LLD

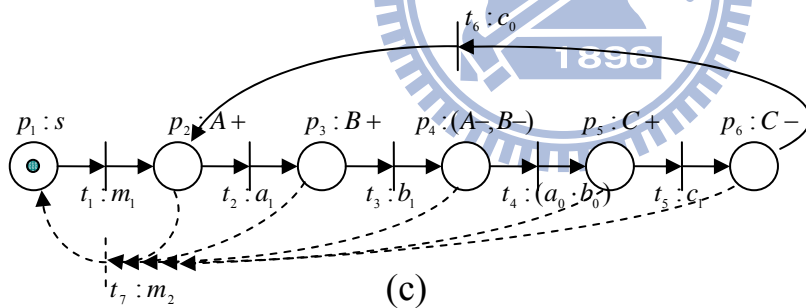
According to the sequence of the stamping system associate input sensor, the corresponding BPNs are shown in Fig. 5.6(b). One basic safety specification assumption is that in any case in which the system must be shut off, this should be done via the protruding switch m_2 . A BPNs model for this specification constructed using the reversible concept of a Petri nets, which is designed to add an *OR* transition m_2 (denoted as a dotted bar in Fig. 5.6(b)), is shown in Fig. 5.6(c). The proposed mapping LLD approach is shown in Fig. 5.6(d). Both RLL and HPL can be used in a similar way.



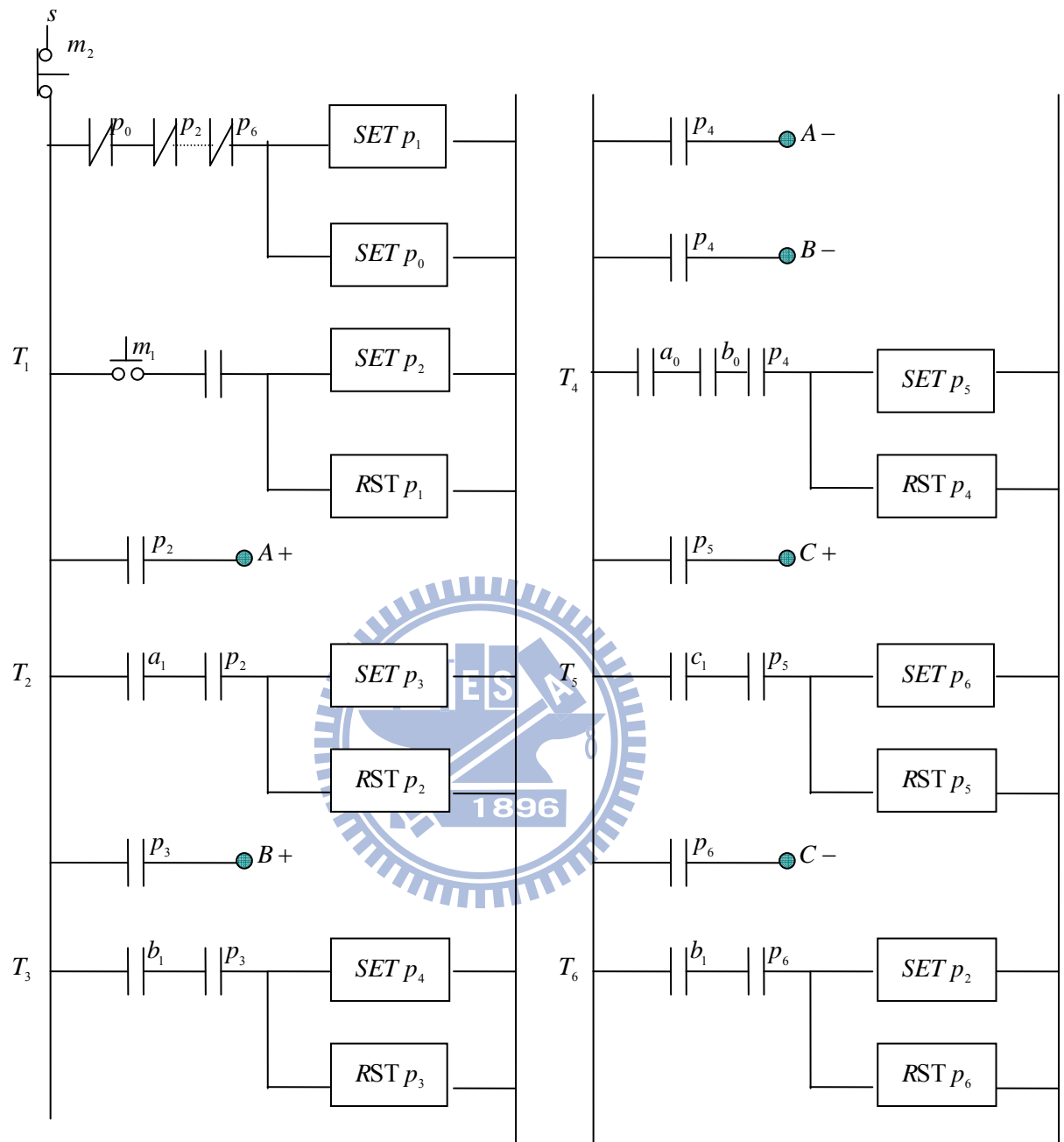
(a)



(b)



(c)



(d)

Fig. 5.6. (a) Structure diagram of the stamping system (from Lee 2005), (b) Corresponding BPNs of the stamping system, (c) Corresponding BPNs with the added safety design for the stamping system, (d) Mapped LLD using BPNs.

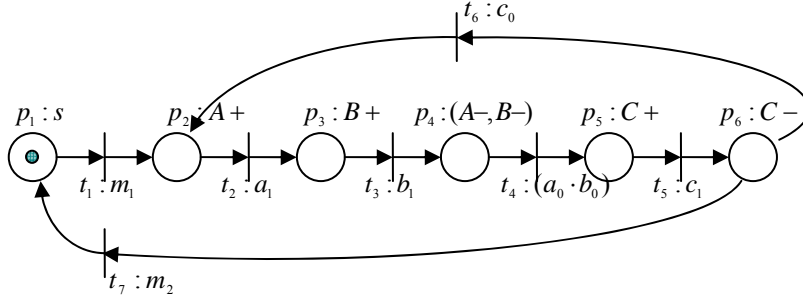


Fig. 5.7. Abstract BPNs model of the stamping system.

5.4.3 Abstract model and state equation

The abstract BPNs model shown in Fig. 5.7 is a behavioral model. The behavioral model is simplified by the given BPNs model but matches the function of the controller. According to Eq. (5-2), the state equation $M_k = M_{k-1} + A^T U_k$ can be used to analyze and simulate the stamping system, where A^T is the incidence matrix used to represent an abstract model, U_k is the firing vector (fault free/fault), “1” represents the occurrence of an event and “0” represents no event in the firing vector 1/0 (0/1).

$$A^T = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \\ \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 \end{bmatrix} \end{matrix} \succ M_0 = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{matrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{matrix} \succ U_1 = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \end{matrix} \begin{matrix} \begin{bmatrix} 1/0(0/1) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{matrix} \succ U_2 = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \end{matrix} \begin{matrix} \begin{bmatrix} 0 \\ 1/0(0/1) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{matrix} \succ \\
 U_3 = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \end{matrix} \begin{matrix} \begin{bmatrix} 0 \\ 0 \\ 1/0(0/1) \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{matrix} \succ U_4 = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \end{matrix} \begin{matrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1/0(0/1) \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{matrix} \succ U_5 = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \end{matrix} \begin{matrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1/0(0/1) \\ 0 \\ 0 \end{bmatrix} \end{matrix} \succ U_6 = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \end{matrix} \begin{matrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1/0(0/1) \\ 0 \end{bmatrix} \end{matrix} \succ U_7 = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \end{matrix} \begin{matrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1/0(0/1) \end{bmatrix} \end{matrix} \succ$$

$$M_1 = M_0 + A^T U_1 = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{bmatrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1/0(0/1) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{bmatrix} 0/1 \\ 1/0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ or } = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{bmatrix} (1/0) \\ (0/1) \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Similarly,

$$M_2 = M_1 + A^T U_2 = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{bmatrix} 0 \\ 0/1 \\ 1/0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ or } = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{bmatrix} 0 \\ (1/0) \\ (0/1) \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad M_3 = M_2 + A^T U_3 = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{bmatrix} 0 \\ 0 \\ 0/1 \\ 1/0 \\ 0 \\ 0 \end{bmatrix} \text{ or } = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{bmatrix} 0 \\ 0 \\ (1/0) \\ (0/1) \\ 0 \\ 0 \end{bmatrix},$$

$$M_4 = M_3 + A^T U_4 = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0/1 \\ 1/0 \\ 0 \end{bmatrix} \text{ or } = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ (1/0) \\ (0/1) \\ 0 \end{bmatrix}, \quad M_5 = M_4 + A^T U_5 = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0/1 \\ 1/0 \end{bmatrix} \text{ or } = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ (1/0) \\ (0/1) \end{bmatrix},$$

$$M_6 = M_5 + A^T U_6 = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{bmatrix} 0 \\ 1/0 \\ 0 \\ 0 \\ 0 \\ 0/1 \end{bmatrix} \text{ or } = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{bmatrix} 0 \\ (0/1) \\ 0 \\ 0 \\ 0 \\ (1/0) \end{bmatrix}, \quad M_7 = M_6 + A^T U_7 = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{bmatrix} 1/0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0/1 \end{bmatrix} \text{ or } = \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \begin{bmatrix} (0/1) \\ 0 \\ 0 \\ 0 \\ 0 \\ (1/0) \end{bmatrix}.$$

5.4.4 Generating the testing event sequence and troubleshooting

The test event sequence of the local controller and the abstract model shown in Fig. 5.2 can be generated from the BPNs model for PLC testing and diagnosis. It can also be used to support network-based monitoring and supervision. Faulty diagnosis of the local controller according to the switch type of the local controller leads to 3 simplified types, as shown in Table 5.4, where the test event sequence is $m_1 \rightarrow a_1 \rightarrow b_1 \rightarrow a_0 \rightarrow b_0 \rightarrow c_1 \rightarrow c_0 \rightarrow m_2$.

Table 5.4: Test event sequence and troubleshooting of the stamping process

| Test Event Sequence | Type of switches | Initial state | Fault free | Fault | Troubleshooting |
|--|----------------------|---------------|------------|-------|--|
| $t_i : (m_1, a_1, b_1, a_0, b_0, c_1, c_0)$ no event occurs | Normal open switch | | | | Please check normal switches to determine whether they are stuck at on or not. |
| $t_i : (m_1, a_1, b_1, a_0, b_0, c_1, c_0)$ event occurs | Normal open switch | | | | Please check relational switches and their interconnected line to determine whether they are stuck at off. |
| $t_7 : m_2$ event occurs | Normal closed switch | | | | Please check switch m_2 to determine whether it is stuck at on or not. |

TABLE 5.5: Comparison of SPNC and BPNs for stamping system

| Comparison measures | SPNC | BPNs |
|---------------------|--------------|--------------|
| Basic elements | Place 15 | Place 6 |
| | Transition 8 | Transition 6 |
| | Arc 25 | Arc 12 |
| | Total 48 | Total 24 |

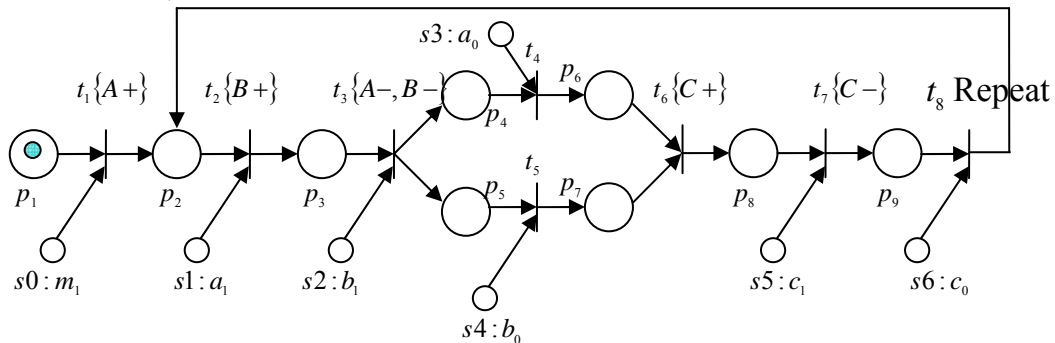


Fig. 5.8. Corresponding SPNC of the stamping system (from Lee, 2004)

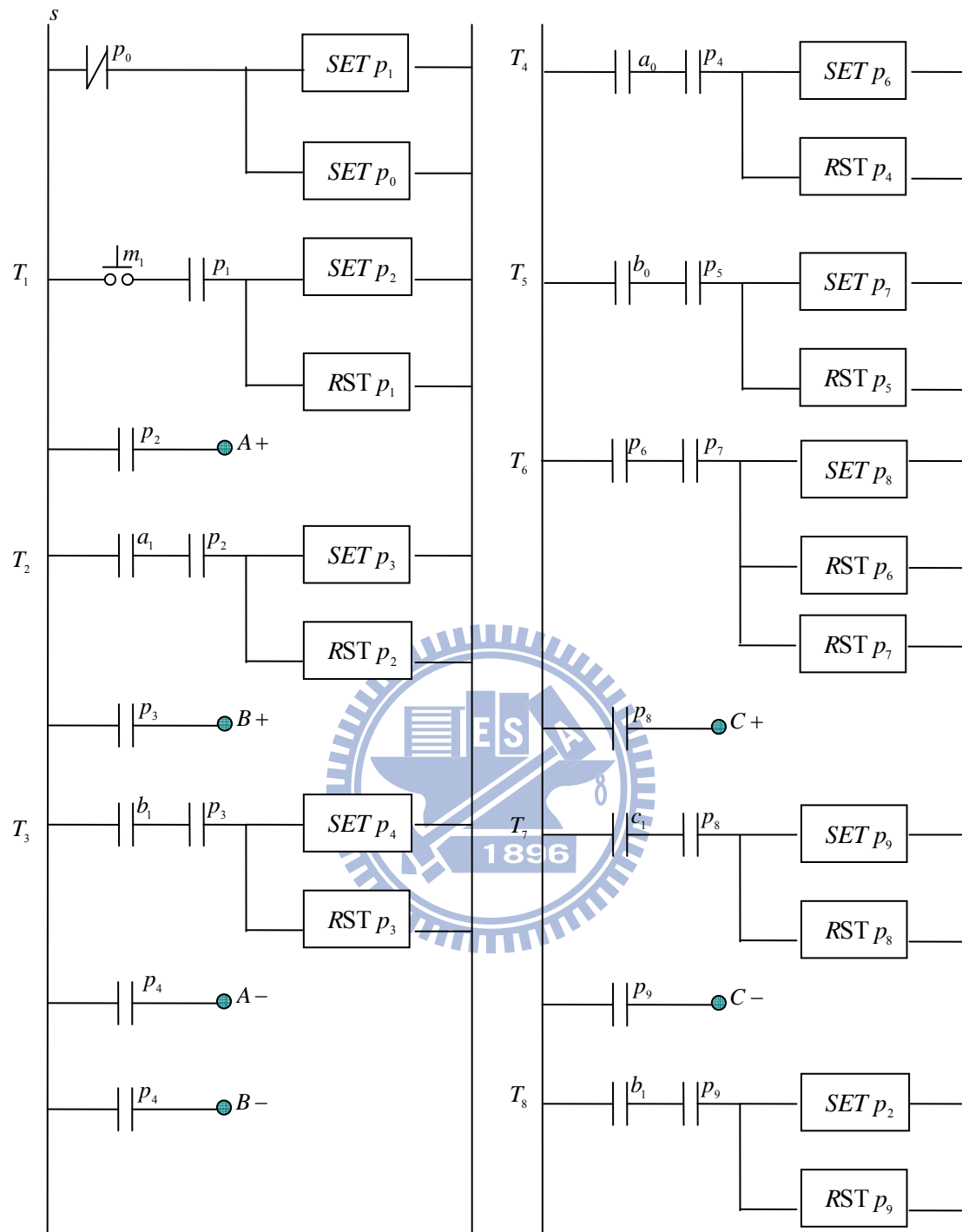


Fig 5.9. LLD implementation of stamping system (from Lee, 2004)

5.4.5. Comparison of SPLC, BPNs and Corresponding LLD

The IDEF0/SPNC/TPL/LLD (Lee, 2004) is a systematic implementation approach; however, the BPNs introduced composite transition and place to reduce the

complexity of BPNs and simplified controller implementation. Therefore, the comparison based simply on the number of Basic element for SPNC (Lee, 2004) and BPNs of stamping system, is shown in Table 5.5. The SPNC needs place 15, transition 8 and arc 25, while BPNs only needs place 6, transition 6 and arc 12 from Fig 5.8 and Fig. 5.6 (b), respectively. Furthermore, the mapping unit is from SPNC and BPNs to LLD are 8 and 6 from Fig. 5.9 and Fig. 5.6 (d), respectively. Thus the BPNs are a simple approach.

5.5. Summary

In this chapter, a clear design approach is proposed for the testing, diagnosis and implementation of logic controllers using BPNs. The BPNs model is a core approach, a bridge between a system specification and PLC code. The abstract model can directly generate a testing event sequence to solve the experience-based testing and diagnosis problems of controllers. It also supports network-based monitoring and supervision, and it can be directly mapped into three types of PLC code to support different implementations. Finally, an example of a stamping process is provided to illustrate the design, implementation, testing and troubleshooting process as well as to demonstrate the usefulness of this approach.

CHAPTER 6

CONCLUSIONS

6.1. Summary of Contributions

In this thesis, a method for the design, testing, diagnosis, and implementation of a sequence controller for remotely monitored and controlled processes were proposed. The model and techniques developed in thesis are useful for industrial applications of automated systems. The contributions of this thesis to the design of automated systems can be summarized as follows:

1) Test generation and determination of fault sites in combinational circuit

To improve the efficiency of logic faults, the transitions of general Petri nets were modified according to a local critical true table, known as the Logic Petri nets (LPNs). The LPNs model transferred complex circuit problems into a local, adjacent place and a transition relational problem, which simplified the site of fault and fired logic value problems (Tsai, Lee and Teng 2006).

2) Construction of an abstract model of a ladder diagram

To diagnose the local fault of a ladder diagram on-line, a Boolean Petri nets model was proposed. The model introduces the concepts of composite transitions, composite places, and relevant states to reduce the complexity of the system and to increase the readability of Petri nets. To determine faulty areas of the ladder diagram, the proposed diagnostic process employs simple matrix manipulations and a difference output vector (DOV) (Tsai and Teng, 2010).

3) Implementation of ASIC for the testing of a ladder diagram

To solve experience-based testing and troubleshooting problems in LDs, the developed method introduces a procedure that compares fault circuits and fault-free circuits into integrated circuit testing. Moreover, to achieve the proposed method, a fault-free model was constructed and application-specific integrated circuits (ASICs) and testing event sequences were implemented. As a result, the testing problem was transformed into the determination of event occurrence. For instance, if an event did not occur in the primary input, then a response is not obtained from the primary output. Likewise, if an event occurs in the primary input, then the primary output responds accordingly, which results in the detection of faults (Tsai, Lin and Teng, accepted).

4) Design for the testing and implementation of logic controllers

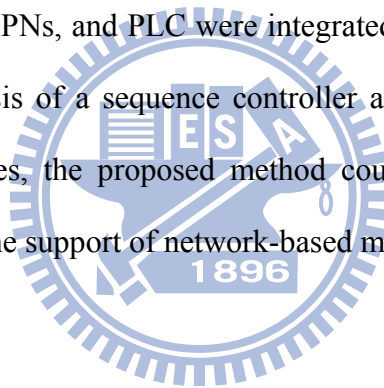
To solve experience-based testing and diagnosis problems in the design of sequence controllers from system specifications, a BPNs model that acts as a bridge between system specifications and PLC codes was developed. The abstract model can directly generate a testing event sequence to solve experience-based testing and diagnosis problems in sequence controllers. The model also supports network-based monitoring and supervision, and can be directly mapped into three different types of PLC code to support a variety of implementations. Finally, an example of a stamping process was provided to illustrate the design, implementation, testing, and troubleshooting of a sequence controller and to demonstrate the usefulness of the proposed approach (Tsai, Liao and Teng, submitting).

6.2. Future Research

The applications of PNs for the testing of circuit systems can be extended in the

following directions:

- 1) In this thesis, the LPNs were used to generate automatic test patterns (ATPGs) in combinational circuits. By applying an extended D-algorithm (Putzolu 1971), the present model could be extended to sequence ATPG applications.
- 2) By employing BPNs, the testing and diagnosis of sequence controllers designed from system specifications or existing PLCs was achieved. In future studies, the BPNs map could be applied to Java language because Java technology was used to implement the intelligent agent for on-line supervision (Lee 2004, thesis).
- 3) In this thesis, IDEF0, BPNs, and PLC were integrated to develop an approach for the testing and diagnosis of a sequence controller and the generation of testing events. In future studies, the proposed method could be extended to different *IF-THEN* systems for the support of network-based monitoring and supervision.



REFERENCES

- Abramovici, M., Breuer, M. A. and Friedman, A. D. (1990), *Digital systems testing and testable design*, Computer Science Press: New York.
- Bender, D. F., Combemale, B., Crégut, X., Farines, J., Berthomieu, M. B. and Vernadat, F. (2008), “Ladder metamodeling & PLC program validation through time Petri nets,” *ECMDA Lecture notes computer science*, Springer Berlin / Heidelberg, pp. 121-136.
- Chen, S. M., Ke, J. S. and Chang, J. F. (1990), “Knowledge representation using fuzzy Petri nets,” *IEEE Trans. Knowl. Data Eng.*, vol. 2, pp. 311-319.
- Chen, S. M. (2000), “Fuzzy backward reasoning using fuzzy Petri nets,” *IEEE. Trans. Systems, Man, and Cybernetics. – Part B*, vol. 30, no. 6, pp. 846-856.
- David, R. and Alla, H. (1992), *Petri Nets and Grafcet: Tools for Modeling Discrete-Event Systems*, London: Pretrice-Hall.
- David, R. (1995), “Grafcet: a powerful tool for specification of logic controllers,” *IEEE Transactions on Control Systems Technology*, pp. 253 – 268.
- Dotoli, M. P., Fanti, G. M., Iacobellis, G. G. and Mangini, A. M. (2009), “A First-Order Hybrid Petri Net Model for Supply Chain Management,” *IEEE Trans. Automat. Sci. Eng.*, vol. 6, no. 4, pp. 744–758.
- FIPS 183 (1993), *National Institute of Standards and Technology, Integration Definition*

for Function Modeling (IDEF0). NIST, USA.

Hu, H. S., Zhou, M. C. and Li, Z. W. (2009), “Liveness enforcing supervision of video streaming systems using nonsequential Petri nets,” *IEEE Trans. Multimedia*, vol. 11, no. 8, pp. 1457–1465.

Jackman, J., Linn, R. J. and Hyde, D. (1995), “Petri net modeling of relay ladder logic,” *J. Design & Manuf.*, vol. 5, pp. 143-151.

Kirkland, T. and Mercer, M. R. (1988), “Algorithms for Automatic Test-pattern Generation,” *IEEE Design and Test of Computers*, vol. 5, no. 3, pp. 43-55.

Lala, P. K. (2009), *An Introduction to logic circuit Testing*, Morgan & Claypool.

Lan, J. C. and Ma, M. (2009), “Fault diagnosis method of power system based on the adaptive fuzzy Petri net,” *Test and Diagnosis, IEEE Circuits and Systems International Conference on 28-29*, pp. 1-4.

Lee, G. S. and Lee, J. S. (2000), “The state equation of Petri net for the LD program,” *Proc. IEEE Trans. Int. Conf. Systems, Man, and Cybernetics*, pp. 3051-3056.

Lee, J. S. (2004), *Design of the remote supervision system for automated processes via the Petri nets approach*, Ph.D. Dissertation, Department of Electrical and Control Engineering, National Chiao- Tung University, Taiwan, ROC, July.

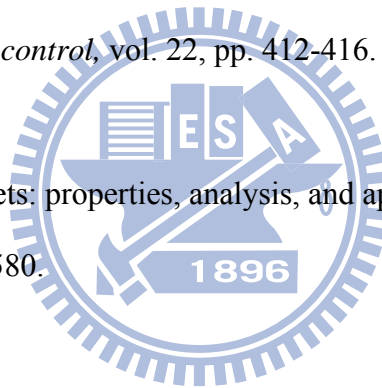
Lee, J. S. and Hsu, P. L. (2005), "A systematic approach for the sequence controller design in manufacturing systems," *Int. J. Adv. Manuf. Technol.*, pp. 754-760.

LO, K. L., Ng, H. S. and Trecat, J. (1997), "Power systems fault diagnosis using Petri nets," *IEE proceeding C. generation, Transmission and Distribution*, vol. 144, pp. 231-236.

Looney, C. G. (1987), "Logical Control via Boolean Rule Matrix Transformations," *IEEE. Trans. Systems, Man, and Cybernetics*, vol. SMC-17, no. 6, pp. 1077-1082.

Murata, T. (1977), "State equation, controllability, and maximal matching of Petri nets," *IEEE Trans. automatic control*, vol. 22, pp. 412-416.

Murata, T. (1989), "Petri nets: properties, analysis, and application," *Proceedings of IEEE*, vol. 77, no. 4, pp. 541-580.



Peng, S. S., and Zhou, M. C. (2001), Conversion between Ladder diagram and Petri-net in discrete-event control design- A survey, *IEEE Trans. Int. Conf. Systems, Man, and Cybernetics*, pp. 2682-2687.

Peng, S. S. and Zhou, M. C. (2004), "Ladder diagram and Petri-net-based discrete-event control design methods," *IEEE. Trans. Systems, Man, and Cybernetics -Part C: Applications and Review*, vol. 34, no. 4, pp. 523-531.

Santarek, K. and Buseif, I. M. (1998), "Modeling and design of flexible manufacturing systems using SADT and Petri nets tools," *J. Mater Process Tech.*, vol. 76, pp.

212-218.

Taholakian, A. and Hales, W. M., "PN->PLC: a methodology for designing, simulation and coding PLC based control system using Petri nets," *int. J. product. Res.*, vol. 35, no. 6 pp. 1743-1762, 1997.

Tsai, J. I. and Teng, C. C. (2010), "Constructing an abstract model for ladder diagnosis using Petri nets", *A Special Issue of Asian Journal of Control*, vol. 12, no. 3, pp. 309-318.

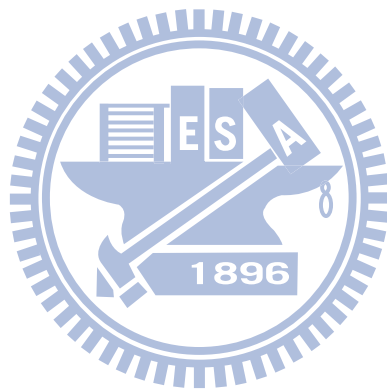
Uzam, M. and Jones, A. H. (1998), "Discrete even control system design using automation Petri nets and their ladder diagram implementation," *Int. J. Adv. Manuf. Tech.*, vol. 14, pp. 716-728.

Venkatesh, K., Zhou, M. C. and Caudill, R. J. (1994), "Comparing ladder logic diagrams and Petri nets for sequence controller design through a discrete manufacturing system," *IEEE Trans. on Industrial Electronics*, vol. 41, no. 6, pp. 611-619.


Venkatesh, K., Zhou, M. C. and Caudill, R. J. (1994a), "Evaluating the complexity of Petri nets and ladder logic diagrams for sequence controllers design in flexible automation," *Proc. of IEEE Workshop on Emerging Technologies and Factory Automation*, Tokyo, Japan, pp. 428-435.

Zhou, M.C. and Twiss, E. (1998), "Design of industrial automated systems via relay ladder logic programming and Petri nets," *IEEE Transactions on Systems, Man, and*

Cybernetics, Part C: Applications and Reviews, pp. 137 – 150.



VITA

| | |
|---|--|
| July 19, 2010 |  |
| PERSONAL DATA | |
| | |
| Name: 蔡瑞益 Jui-I Tsai | |
| Date of Birth: April 19, 1958 | |
| E-mail: tsai.ece90g@nctu.edu.tw | |

EDUCATION

| | |
|---------------|---|
| 2001/9-2010/7 | Receive the Ph.D. degree in the Institute of Electrical Control Engineering at National Chiao-Tung University, Taiwan, R.O.C. |
| 1992/9-1994/7 | Receive the M.S. degree in the Institute of Medical Engineering from National Cheng- Kung University, Taiwan, R.O.C. |
| 1981/9-1985/7 | Receive the B.S. degree in the Department of Electronic Engineering from Feng-Chia university, Taiwan, R.O.C. |

ATTENDED CONFERENCES

- **International Conferences**

| |
|--|
| 2006 Oct., IEEE Intl. Conf. Systems, Man and Cybernetics, Taipei, R.O.C. |
|--|

- **Domestic Conferences**

| |
|---|
| 2006 Nov., Chinese Automatic Control Conference, Taipei, R.O.C. |
|---|

PUBLICATION LIST

July 19, 2010

JOURNAL PAPERS

- | |
|---|
| 1. J.I. Tsai , J. M. Shieh, T. S. Liao and C. C. Teng, "High-voltage amplifier uses simplified circuit," <i>EDN Design Ideas</i> , pp.110, 2004. |
| 2. J.I. Tsai , W. W. Pai, F. C. Hsu, P. J. Chen, J. M. Shieh, C. C. Teng and T. S. Liao, "Relays eliminate high-voltage noise," <i>EDN Design Ideas</i> , pp.66-68, 2007. |
| 3. J.I. Tsai and C. C. Teng, "Constructing an abstract model for ladder diagnosis using Petri nets," <i>A Special Issue of Asian Journal of Control</i> , vol. 12, no. 3, pp. 309-318, 2010. |
| 4. B. T. Lin, J.I. Tsai and C. C. Teng, " <i>QFT / H_∞</i> Controller Design of a MIMO Suspension System," accepted by <i>the Advances in Differential Equations and Control Processes</i> , vol. 5, no. 1, pp. 49-63, 2010. |
| 5. J.I. Tsai , B. T. Lin and C. C. Teng "An ASIC Implementation for testing of a ladder diagram using a Boolean Petri net," accepted by <i>Far East Journal of Experimental and Theoretical Artificial Intelligence</i> , 2010. |
| 6. J.I. Tsai , T. S. Liao, B. T. Lin and C. C. Teng "Design for the testing and implementation of logic controllers using Boolean Petri net," <i>Far East Journal of Experimental and Theoretical Artificial Intelligence</i> , 2010 (revise). |

INTERNATIONAL CONFERENCE PAPER

- | |
|--|
| 1. J.I. Tsai , C. C. Teng and C. H. Lee, "Test Generation and Site of Fault for Combinational Circuits Using Logic Petri Nets," <i>IEEE International Conference on Systems, Man, and Cybernetics</i> , Taipei, pp. 91-96, Oct. 2006. |
|--|

DOMESTIC PAPER

- | |
|--|
| 1. J.I. Tsai , and C. C. Teng, "Fuzzy reasoning based on logical Petri nets," <i>Automatic Control Conference</i> , Taipei, pp. 1231-1236, Nov. 2006. |
|--|