

A GRAPHICAL METHOD FOR AUTOMATIC CODE GENERATION FROM EXTENDED S-SYSTEM PETRI NET MODELS

NG KOK MUN

KOLEJ UNIVERSITI TEKNOLOGI TUN HUSSEIN ONN

PERPUSTAKAAN KUITTHO



3 0000 00180731 0

KOLEJ UNIVERSITI TEKNOLOGI TUN HUSSEIN ONN

PENGESAHAN STATUS TESIS

A GRAPHICAL METHOD FOR AUTOMATIC CODE GENERATION
FROM EXTENDED S-SYSTEM PETRI NET MODELS

SESI PENGAJIAN : 2005/2006

Saya NG KOK MUN mengaku membenarkan Tesis Sarjana ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hak milik Kolej Universiti Teknologi Tun Hussein Onn.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. ** Sila tandakan (✓)



SULIT

(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)



TERHAD

(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)



TIDAK TERHAD

Disahkan oleh



(TANDATANGAN PENULIS)



(TANDATANGAN PENYELIA)

Alamat Tetap:

NO 6, LORONG PJS 7/15 B,
BANDAR SUNWAY, 46150
PETALING JAYA, SELANGOR

PM. DR. ZAINAL ALAM BIN HARON

Tarikh:

24/7/06

Tarikh: 24/7/06

Nama Penyelia

CATATAN:

- ** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali sebab dan tempoh tesis ini perlu di kelaskan sebagai SULIT atau TERHAD.

**A GRAPHICAL METHOD FOR AUTOMATIC CODE GENERATION
FROM EXTENDED S-SYSTEM PETRI NET MODELS**

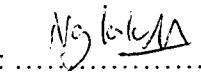
NG KOK MUN

A thesis submitted in fulfillment of the requirements for the award of the Degree of
Master of Electrical Engineering

Department of Electrical and Electronics
Faculty of Electrical Engineering
Kolej Universiti Teknologi Tun Hussein Onn

JULY, 2006

"I hereby declare that the work in this thesis is my own except for quotations and summaries which have been duly acknowledged"

Signature : 

Name of Candidate : NG KOK MUN

Date : 24/7/06

Supervised by :

Supervisor : 

: PM DR. ZAINAL ALAM BIN HARON

“I/We declare that I/We have read through this thesis and in my/our view, this thesis is sufficient in fulfilling the scope and quality for the purpose of awarding the Master of Electrical Engineering”

Examiner I : PROF. DR. MOHAMED KHALIL BIN MOHD. HANI
Universiti Teknologi Malaysia

Examiner II : PROF. MADYA DR. ROSZIATI BINTE IBRAHIM
Kolej Universiti Teknologi Tun Hussein Onn

Chairman : DR. CHRISTY A/L PATHROSE GOMEZ
Kolej Universiti Teknologi Tun Hussein Onn

Date : 20/6/06

Dedicated to my parents and siblings

ACKNOWLEDGEMENT

I would like to convey my appreciation to my supervisor, Associate Professor Dr. Zainal Alam bin Haron for introducing Petri Net and also providing the necessary guidance in this work. The constructive criticism, feedbacks and advice provided encouraged me to put more effort in this write up.

I am also touched by the encouragement and prayers given by my church friends during this process of doing this research. Not to forget, the financial support and help given when I am in financial need during the initial stage of my course. Special thanks to my siblings Yee Fong, Yee Fun and Kok Kee for being my scholarship's guarantors.

Finally, all praises and glory be to my Lord Jesus Christ for His love and provision of finance, wisdom and strength to complete this work. To God be the glory!

ABSTRACT

This work has introduced a fast and reliable method for graphical modeling of discrete systems control problems using extended S-system Petri Net. By adding new functionalities to the extended S-System Petri Net, dynamic quantities such as microcontroller signals transitions, system timing, interrupts, subroutines and arithmetic operations could now be modeled by software. A graphical-based diagram editor has been developed in this work to handle the model entry, editing and visualization. The diagram editor contains all the basic facilities required for entering, editing, visualization and syntax analysis of the S-System Petri Net model. A compiler has also been built to compile the graphical model and generate the assembly code automatically. Together, the diagram editor and model compiler forms an integrated design and development tool called S-PNGEN. Seamless data binding between the diagram editor and the model compiler is achieved by using a common directed-graph framework to internally represent the model diagrams. Diagram syntax checking was implemented using *attributed graph grammar*. Also introduced in this work is an efficient method for implementing the control solutions on a microcontroller. This involves the development of a procedure for automatically mapping S-System Petri Net models constructed in the diagram editor to control flow graphs. The procedure uses a notion called graph nesting to help the design tool read and understand S-System model diagrams and transform them into control flow graphs. Conversion of an S-System Petri Net model into a control flow graph is an innovative approach introduced in this work for automatic code generation as it guarantees the production of the correct code layout and information for use by the compiler. By applying a syntax-directed translation on the control flow graph constructed, the built-in compiler then automatically generates the assembly code for the target microcontroller.

ABSTRAK

Penyelidikan ini memperkenalkan kaedah yang effisien untuk membentuk model bagi sistem kawalan diskrit secara grafikal dengan menggunakan *extended S-System Petri Net*. Dengan menambahkan fungsi-fungsi baru ke atas suatu *S-System Petri Net*, kuantiti dinamik suatu pengawal mikro seperti isyarat, masa, subrutin, *interrupts* dan operasi aritmetik dapat dimodelkan oleh *software*. Satu *diagram editor* telah dibina untuk membolehkan pelukisan dan pengubahsuaian model. *Diagram editor* ini mempunyai kemudahan asas yang membolehkan pembentukan dan pengubahsuaian model serta melaksanakan analisis sintaks ke atas model *S-System Petri Net* yang dibina. Satu pengkompil telah dibangunkan untuk mengkompil model grafikal yang dibina dan juga untuk menjana kod *assembly* secara otomatik. Kedua-dua *diagram editor* dan pengkompil diintegrasikan sebagai suatu alat rekabentuk model dipanggil *S-PNGEN*. Kedua-dua *diagram editor* dan pengkompil berkongsi data dengan menggunakan rangka struktur data graf yang sama bagi mewakili model yang dilukis. Sintaks model diimplementasikan melalui *attributed graph grammar*. Hasil kerja ini juga memperkenalkan suatu prosedur yang memetakan model *S-System Petri Net* yang dibina dalam *diagram editor* kepada *control flow graphs*. Prosedur ini menggunakan suatu konsep *graph nesting* yang membolehkan alat rekabentuk kami membaca dan memahami model *S-System Petri Net* dan mengubahnya kepada *control flow graphs*. Pertukaran model kepada *control flow graphs* merupakan satu inovasi di dalam kerja ini untuk menjana kod secara otomatik kerana ia dapat memberikan bentangan kod yang betul dan maklumat untuk kegunaan pengkompil. Dengan mengaplikasikan *syntax-directed translation* ke atas *control flow graphs* yang dibina, pengkompil dapat menjanakan kod *assembly* untuk suatu pengawal mikro secara otomatik.

TABLE OF CONTENTS

CHAPTER	ITEMS	PAGE
	TITLE	i
	STUDENT'S DECLARATION	ii
	ESAMINERS' DECLARATION	iii
	DEDICATION	iv
	ACKNOWLEDEMENT	v
	ABSTRACT	vi
	ABSTRAK	vii
	TABLE OF CONTENTS	viii
	LIST OF TABLES	xiii
	LIST OF FIGURES	xiv
	LIST OF ABBREVIATIONS	xviii
	LIST OF APPENDIXES	xix
I	INTRODUCTION	1
	1.1 Background	1
	1.2 Problem statement	4
	1.3 Research objectives	6
	1.4 Research scope	6
	1.5 Thesis outline	7
II	LITERATURE REVIEW	8
	2.1 Introduction	8
	2.2 A brief overview of the S-System PN	8

CHAPTER	ITEMS	PAGE
2.2.1	Adding extended functionalities to the S-System PN	9
2.2.1.1	Practical implications of the S-System PN components	9
2.2.1.2	Describing hierarchy and subroutines using macro places	11
2.2.1.3	Describing logic states and arithmetic operations	11
2.2.1.4	Handling interrupts in microcontroller	13
2.2.1.5	Describing timing in microcontrollers with timed transitions	15
2.2.1.6	Token in the S-System PN	16
2.2.2	Application of the extended S-System PN	17
2.3	Control flow graph and the S-System PN	20
2.4	A brief review of specification tools	27
2.4.1	Directed graphs	28
2.5	A brief review of diagram parsers	32
2.6	A brief review on text parsing and code Generation methods	33
2.7	Summary	34
III	MODELING THE PROTOTYPE TOOL	36
3.1	Introduction	36
3.2	An overview of prototype tool Development	37
3.3	Conceptual framework adopted for prototype tool	42

CHAPTER ITEMS	PAGE
3.3.1 Diagram editor	43
3.3.2 Compiler	43
3.3.2.1 Parser	44
3.3.2.2 Code generator	45
3.4 Diagram editor design	46
3.4.1 The graphical editor	47
3.4.2 The model editor	49
3.4.2.1 Variables declaration	50
3.4.2.2 Updating place or transition properties or attributes	52
3.4.2.3 I/O pins direction	55
3.4.2.4 EEPROM settings	55
3.4.3 Data structures	56
3.4.3.1 Directed graph construction via diagram drawing	57
3.4.3.2 Symbol table (Hash-table)	59
3.4.3.3 Other data structures	60
3.4.4 Other functions in the diagram editor	61
3.5 Parser design	62
3.5.1 Text and diagrams syntax parsing	63
3.5.1.1 Parsing text language with context-free grammars	63
3.5.1.2 Using context-free grammars to parse text into parse trees	66
3.5.2 Parsing diagrams	71
3.5.2.1 Parsing diagrams with attributed graph Grammar	74

CHAPTER	ITEMS	PAGE
	3.5.2.2 Detecting subnets for macro places	79
	3.5.3 Creating abstract representations from graph transformation	82
	3.6 Summary	87
IV	IMPLEMENTATION OF THE PROTOTYPE	89
	4.1 Introduction	89
	4.2 The target machine architecture	91
	4.3 Code selection	91
	4.3.1 Code selection for I/O port and EEPROM initialization	93
	4.3.2 Code selection for expressions and assignments	95
	4.3.3 Overall code generation	101
	4.3.3.1 Code generation for port variables	104
	4.3.3.2 Subroutines code generation	105
	4.3.3.3 Code generation for control and branching instructions for the main program	109
	4.4 Registers and memory allocation	110
	4.5 Discussion	114
	4.6 Summary	115

CHAPTER ITEMS	PAGE
V APPLICATIONS OF THE PROTOTYPE	117
5.1 Introduction	117
5.2 Results	118
5.3 The assembler / simulation tool	119
5.4 Assembly code verification and validation	121
5.4.1 Assembling the assembly code	121
5.4.2 Simulation on assembly code	123
5.4.2.1 Simulating the mixing cycle	125
5.4.2.2 Simulating the wash cycle	128
5.5 Further simulations	129
5.6 Summary	130
VI CONCLUSION AND RECOMMENDATIONS	131
6.1 Introduction	131
6.2 Benefits	131
6.3 Drawbacks	132
6.4 Recommendations	135
6.4.1 Code optimizations	135
6.4.2 Retargetable compiler	137
6.4.3 Upgrading the prototype tool	138
6.5 Conclusion	139
REFERENCES	141
APPENDIXES	147

LIST OF TABLES

TABLE NO.	TITLE	PAGE
2.1	Places in the main net	19
2.2	Transitions in the main net	19
2.3	Places in the subnet	20
2.4	Transitions in the subnet	20
2.5	Representations by PN for a <i>block</i> node	23
2.6	Representations by PN for a <i>switch</i> node	24
2.7	Methods in the <i>ASDigraph</i> class	30
2.8	Accessor methods	30
2.9	Iteration methods	31
3.1	Input sentences	68
3.2	Evaluation conditions	76
3.3	Semantic rules violation	78
3.4	Attributes in <i>RegionNode</i> objects that describe the main net	86
3.5	Attributes of <i>RegionNode</i> associated to a subnet	87
4.1	Nodes of parse tree of figure 4-7 and the associated code templates	96
4.2	Nodes of parse tree of figure 4-11 and their associated code templates	100
5.1	Inputs and outputs of the mixing operation	123
6.1	Optimizations methods (adapted from (Muchnick, 1997))	137

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
2-1	Macro place	11
2-2	S-System PN model for a switch press mechanism	12
2-3	Handling PORT B bit 0 interrupt	14
2-4	Timed transition	16
2-5	A mixing operation	17
2-6	S-System PN model for the mixing operation	18
2-7	Control flow graph of a microcontroller program	22
2-8	Basic blocks terminology	23
2-9	Derivation of the S-System model into control flow graph	25
2-10	Control flow graph with the respective algebraic assignments and expressions	25
2-11	Assembly code generated	26
2-12	Adjacency-set representation of a directed graph	29
3-1	Design flow of the prototyping tool	38
3-2	An overview of the diagram editor	39
3-3	The Parser module	40
3-4	MPLAB IDE	41
3-5	Absolute assembly-level code translation	42
3-6	Assembly code arrangement within basic blocks	45

FIGURE NO.	TITLE	PAGE
3-7	S-PNGEN's diagram editor	47
3-8	Graphical editor menus and sub-menu functions	48
3-9	Model editor's menus and functions	49
3-10	<i>Project Variables</i> dialog	50
3-11	Dialog to update variable's attributes	51
3-12	<i>Transition Attributes</i> dialog	52
3-13	<i>Place Attributes</i> dialog	53
3-14	<i>Place Settings</i> dialog	54
3-15	<i>Port Setting</i> dialog	55
3-16	EEPROM Settings dialog	56
3-17	EEPROM contents	56
3-18	A Petri Net model drawn on the drawing area	58
3-19	Associated adjacency set of the model in figure 3-18	58
3-20	Symbol table (hash-table)	59
3-21	Array structure to represent I/O pin direction	60
3-22	Backus-Naur Form production rules	64
3-23	PN model with assignments and expressions	67
3-24	Parse tree for the assignment: PORTA = b`01001100'	69
3-25	Parse tree for expression b*c < x-y	69
3-26	Parse tree for assignment y = m*x+c	70
3-27	Parse tree for expression x+y-c > b+	70
3-28	A sample text parsing report displayed by S-PNGEN	71
3-29	Attributed graph grammars	73
3-30	A PN model in the S-PNGEN	77
3-31	Diagram parsing report	77

FIGURE NO.	TITLE	PAGE
3-31	Macro place p201 and its related subnet	79
3-33	A Subnet associated with 2 identical macro places	80
3-34	A PN model	81
3-35	Parsing report of the PN model in figure 3-34	81
3-36	Graph transformation rules	83
3-37	Graph transformation on a PN model of a mixing operation	85
4-1	Code Generator module	89
4-2	Organization of instructions in a standard template	92
4-3	Array structure and pin directions	93
4-4	A code template for I/O pins direction initialization	94
4-5	Hash-table structure that organizes an EEPROM	94
4-6	A code template for EEPROM initialization	95
4-7	Visiting nodes of a parse tree	96
4-8	Structure of a simulated stack	97
4-9	Parse tree for expression $b*c < x-y$	98
4-10	Evaluation on the left hand side of the expression	99
4-11	Evaluation of the right hand side of the expression	99
4-12	PN model of mixing operation and its abstract representation	102
4-13	Places and transitions with their associated assignments and expressions	103
4-14	Port variables declaration	105
4-15(a)	Assembly code for mixing operation	106

FIGURE NO.	TITLE	PAGE
4-15(b)	Assembly code for mixing operation	107
4-15(c)	Assembly code for mixing operation	108
4-16	Mid-range PIC memory map	111
4-17	Organization of a symbol table	112
4-18	Registers allocation	113
4-19	Registers allocation for mixing operation	114
5-1	Assembler / simulator tool	118
5-2	Code buffer	118
5-3	MPLAB IDE	120
5-4	Assembly code in text editor of the MPLAB	122
5-5	Build Results window	122
5-6	Asynchronous stimulus dialog and watch window in the MPLAB IDE	124
5-7	Input stimulus buttons	124
5-8	Bits status of PORTA and PORTB	125
5-9	Valve 1 and valve 2 activated	126
5-10	Liquid level reaches sensor2 and motor starts spinning	126
5-11	Motor stops spinning and valve 3 activated	127
5-12	Controller back to the initial logic state	127
5-13	Valve 4 activated	128
5-14	Water level reaches sensor2 and motor starts spinning	128
5-15	Motor stops spinning and valve 3 activated	129
6-1	Assembly code for assignment <i>counter = counter + 1</i>	133
6-2	Assembly code for assignment <i>counter = counter + 1</i>	133
6-3	Assembly code for expression <i>count > 2</i>	134
6-4	Code optimizer module	136

LIST OF ABBREVIATIONS

ABBREVIATIONS	MEANING
B(PN) ²	Basic Petri Net Programming Notations
CAD	Computer-aided Design
DLL	Doubly Linked-List
EEPROM	Electrically Erasable Programmable Read Only Memory
IDE	Integrated Development Environment
lhs	Left-Hand Side
OOP	Object-Oriented Approach
PLC	Programmable Logic Controllers
PLD	Programmable Logic Devices
PN	Petri Net
rhs	Right-Hand Side
RISC	Reduced Instruction Set Computer
SDK	Standard Development Kit
SFC	Sequential Function Chart
SIPN	Signal Interpreted Petri Net
SLL	Singly Linked-List

LIST OF APPENDIXES

APPENDIX NO.	TITLE	PAGE
A	The PIC Microcontroller Architecture	147-152
B	The PIC Microcontroller's Instruction Set	153-154
C	Simulation Results	155-199

CHAPTER I

INTRODUCTION

1.1 Background

Since its introduction by Carl Petri in 1962, Petri net (PN) has found a number of important applications in the areas of modeling sequential behavior and process concurrency. In the manufacturing environment, the net-theoretic approach of PN has made it especially valuable in the modeling and design of discrete control and manufacturing systems (Desrochers and Al-Jaar, 1995) (Zhou and Venkatesh, 1999). In the area of control system modelling, the ordinary PN of Carl Petri has been subject to numerous simplifications on the one hand, and extensions on the other hand, tailored according to the level of sophistication expected of the formal model. In general, the simplifications have been intended to result in a simple formal model for the complex systems studied, while the extensions have been used to add functionalities to the net which would widen the scope of applications, such as for developing a full model of the hardware and software characteristics of logic and digital controllers. Also, the extended PNs have been used as formal models for developing more systematic and structured controller programs (Frey and Litz, 2000).

Grafcet which is a subset of PN is a notable example of an extended PN. Drawing its inspiration from PN, Grafcet has been used as the basis for the international standard Sequential Function Chart (SFC), a graphical language for specifying programmable logic controllers (PLC) (David, 1995). Another example

of extended PN is Signal Interpreted PN (SIPN), which allows explicit description of input/output in a well defined way; its application in specifying PLC is found in (Frey and Minas, 2000) and (Minas and Frey, 2002).

Encouraging results have been obtained in the areas of specifying digital controllers and automatic code generation by extending the features of PN. (Machado, Fernandes and Proenca, 1997) for example, has used shobi-PN (a PN extension approach based on SIPN) to specify logic control in programmable logic device⁵(PLD). Their work resulted in automated VHDL code for PLDs. Petri Net for Digital Systems (PNDS) proposed by (Oliveira and Marranghello, 2000) contains most of the features needed for a methodical modeling of digital systems.

An Extended Quasi-Static Scheduling (EQSS) method for formally synthesizing and automatically generating code for embedded software using the Complex-Choice Petri Nets (CCPN) models has been proposed in (Su and Hsiung, 2002). Their work resulted in generation of POSIX based multi-threaded embedded software program in the C programming language. The C code generated is applicable for hardware platform such as Application Specific Integrated Circuits (ASICs), Application Specific Instruction Set Processors (ASIPs) and PLDs. Another approach using PN extension called Timed Free Choice Petri Nets (TFCPN) to model embedded real time software (ERTS) is found in (Hsiung, Lee and Su, 2002). The objective of the work is to synthesize complex ERTS to meet up limited embedded memory requirements and to satisfy hard real-time constraints.

In the area of control system design, current design requirements and practices have reached a high degree of complexity that prevents their efficient realization without sophisticated computer-aided specification and implementation tools (Fernandes, Adamski and Proenca, 1997) (Frey and Minas, 2000) (Minas and Viehstaedt, 1995). *Specification* here is concerned with the description of the PN model and its attributes, which can be specified either textually through textual editors or graphically through CAD tools. The word *implementation* in the software context refers to the generation of a piece of code written in some computer programming language. This code should be ready to use when a low-level language