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

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

LIST OF TABLES

LIST OF FIGURES

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	*Project Variables* dialog	50
3-11	Dialog to update variable's attributes	51
3-12	*Transition Attributes* dialog	52
3-13	*Place Attributes* dialog	53
3-14	*Place Settings* dialog	54
3-15	*Port Setting* 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

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

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 ~~a~~ 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 devices (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 is employed, or directly convertible to a machine understandable one when a high-level programming language is used.

It has been realized from early on, a user-friendly means of controller specifications-writing is a graphical-based CAD tools which provide all the basic functions required by a user to draw the PN model and define its attributes. To this end, a number of customized diagram editors have been developed which allow the user to draw and edit the PN models on the computer screen. A notable example is DiaGen, the diagram editor in (Frey and Minas, 2000) and (Minas and Frey, 2002). DiaGen allows the user to specify PLC from SIPN specifications by providing the necessary drawing tools and facilities, such the free-hand editing facility, which thus allows the user to freely create, delete and modify diagram components (places, transitions and tokens for the SIPN). Other examples of PN diagram editors, such as SOPHIA is found in (Fernandes, Adamski and Proenca, 1997) and *EnVisAge* (Extended Coloured Petri-Net Based Visual Application Generator Tool) in (Kurdthongmee, 2003). SOPHIA is used to specify shobi-PN while *EnVisAge* is used to construct Color Petri Net (CPN) to specify a microcontroller.

In all the tools reported in the literature, conversion of the PN model into object code for the target system is carried by a customized compiler. In the work reported by (Frey and Minas, 2000) and (Minas and Frey, 2002), for example, a customized compiler was used to generate the PLC's Instruction List (IL) code directly from the SIPN model. In SOPHIA (Fernandes, Adamski and Proenca, 1997), VHDL code for PLD was automatically generated by the compiler from the shobi-PN model. In (Melzer, 1997), a code generator is specifically developed to translate the B(PN)^2 notations into C language for a UNIX multiprocessor platform.

The extensions added have all been inspired by the applications intended for the PN. Extended PNs such as SIPN and Grafcet, have been derived from the need to design and specifiy PLC programs and also hybrid systems (Guillemaud and Gueguen, 1999). Both Grafcet and SIPN are naturally suited for modeling and simulating the PLC hardware and control characteristics. In particular, the functionalities available in Grafcet have a very important industrial application in the area of PLC programs specifications and design while the peculiar structure of

SIPN allows it to be directly translated to the instruction list (IL) code of a PLC (Frey, 2000). Another type of extended PN, namely the CPN, has been used to develop a code generator from the given specifications of a security access system (Mortenson, 1999). Similar work adopting CPN such as Color Timed Petri Nets (CTPN) (Gau and Hsiung, 2002) and Extended Color Petri Nets (ECPN) (Kurdthongmee, 2003) had contributed to synthesis of software code for embedded system.

Research works mentioned above have shown a wide applicability of PN and its extensions in specifying controllers. Most of the work carried out aim to provide solutions in programming controllers such as PLCs, ASICs, ASIPs, PLDs and even general microprocessor platform such as UNIX. There is without doubt that the methodologies developed resulted in *specification* and *implementation* tools to generate a piece of code for such controllers. However, the application of PNs in specifying microcontrollers is not widely studied. Motivated by the fact that PN is a useful modeling tool, this work proposes the usage of PN to specify a type of microcontroller.

1.2 Problem statement

In the area of microcontroller modeling and program design, however, extended PNs such as Grafcet and SIPN are simply inapplicable. Microcontrollers, because of their very different architecture from PLCs, are predicated on a platform that strictly executes programs in a sequential manner. Because of this characteristic, Grafcet and SIPN are simply incompatible for microcontroller modeling and program specifications; the main limitation being their tendency to over-describe (or under-describe) the characteristics of a microcontroller's program. For instance, SIPN, which allows explicit description of input/output signals, though having the ability to fully describe a microcontroller's output/input behavior, is incapable of describing characteristics such as subroutines, interrupts, time delays and arithmetic operations.

Approaches such as CCPN in (Hsiung and Su, 2002), TFCPN in (Hsiung, Lee and Su, 2002) and CTPN (Hsiung and Gau, 2002) have proposed methods to generate embedded software with multiple threads, which can be processed for dispatch by a real-time operating system. These PN extensions are also used to solve memory size constraint and concurrent task requirements in embedded systems (e.g. PLDs, ASICs and ASIPs). The methods developed suited devices with "hardwired" architecture and microprocessors that operate with an operating system. This again hinders direct applications of these extensions to specify microcontroller as the microcontroller possesses a single-threaded architecture. Multi-threading activities are not supported in microcontrollers due to the absence of an operating system.

In (Kurdthongmee, 2003), CPN has been used to specify MCS-51 family of microcontrollers. The work had achieved a few envisaged end-points such as the ability to perform model execution analysis and generation of C code for the MCS-51 microcontrollers. Some drawbacks of the work are found in its inability to describe hierarchy functions which makes the model more difficult to read and interpret. Besides, the method introduced does not indicate how interrupts are handled. The author himself stated that the user needs to go through a steep learning curve in using CPN to specify MCS-51. This will somehow affect the user-friendliness of the prototype developed.

In the light of the deficiencies highlighted above, a different type of PN is needed which can satisfactorily address the following requirements:

- It must be able to describe control flow characteristics of a microcontroller program.
- It must be capable in handling routines such as subroutines, timing routines and interrupt handling routines.
- It must be capable of specifying input/output signals of a microcontroller, and
- It must be capable of describing arithmetic operations.

1.3 Research objectives

Based on requirements highlighted in section 1.2, this work has sought to develop a PN model which would satisfactorily address the above-listed modeling requirements. Literature survey carried out at the start of the work by the author has ~~per chance~~ introduced him to the work of (Jorg and Ezparza, 1995) on S-System PN. By imposing the requirement of having only one input and one output arc at every transition, the authors have enabled the PN to model asynchronous control systems.

The objective of the research is to further enhance the S-System PN of (Jorg and Ezparza, 1995) by using some of the extensions introduced by others elsewhere in the literature. We report in this work the details of the enhancements added to the S-System model of (Jorg and Ezparza, 1995) and benefits brought about by the extension in modeling the hardware and software characteristics of the chosen microcontroller.

Another objective of this work is the development of a prototype tool that comprises of a CAD tool (diagram editor) that allows specification of extended S-System PN models and a compiler that implements the specified model to automatically generate the assembly code for the target microcontroller.

1.4 Research scope

The target microcontroller used in this work is PIC 16F84 which is an 8-bit, RISC type, Harvard architecture mid-range microcontroller from the PIC micro family. Sample application of the extended S-System PN model to selected control problems and the method of their specifications and solutions are shown as a guide to its application procedures. These include~~s~~ modeling of input/output signals, arithmetic operations, interrupts, serial peripheral interface communication and delays in the PIC 16F84.

Towards this end a basic prototype tool comprising of a diagram editor and a compiler has been developed. The design of the diagram editor is on graphs and graph grammars (Minas 1998) (Minas 1999) (Minas 2002) and it allows on-screen model entry and editing. Also provided in this tool are facilities for text parsing with context-free string grammars; diagram parsing with attributed graph grammars and syntax-directed assembly code generation. Also explained in this work is the simulation method used to verify and validate the assembly code generated by the compiler.

1.5 Thesis outline

This thesis comprises of six chapters. Chapter 2 provides reviews of related works to substantiate the details of the functionalities added to the S-System PN for describing a microcontroller program. Besides, it also reviews works related to the development of the prototype tool.

Chapter 3 is dedicated to explain the modeling and design of the prototype tool which comprises of a diagram editor and a compiler in detail. This chapter explains the function of the graphical editor and model editor that helps to construct the PN model of the controller graphically and to define the properties or attributes of the model. Also explained is the design and working principle of the compiler's parsing and code generation operations.

Chapter 4 describes the implementation of the prototype tool. Here, methods on code generation are presented in detail. Chapter 5 serves as a testing platform for the prototype tool. It provides details of the simulation work conducted on samples of assembly code generated by the code generator. Finally, chapter 6 provides the conclusion and discussions on the limitations identified with the model and modeling approaches adopted in this work. Also discussed here are recommendations for future work.

REFERENCES

Aho, A.V., Sethi, R. and Ullman, J.D. (1986). "Compilers: Principles, Techniques and Tools." Addison-Wesley.

Ammeraal, L. (1998). "Computer Graphics For Java Programmers." England. John Wiley & Sons Ltd.

Amroun, A. and Bolton, M. (1990). "Synthesis Of Controllers From Petri Net Descriptions And Application Of ELLA." In Claesen, L.J.M. (Ed.). "Formal VLSI Specification And Synthesis." VLSI Design Methods I. North -Holland.Elsevier Science Publishers. pp 291-308.

Bates, M. (2000) "The PIC 16F84 Microcontroller." Arnold.

David, R. (1995) "Grafcet: A Powerful Tool for Specification of Logic Controllers." IEEE Transactions on Control System Technology Vol. 3. No. 3.

Desroches, A.A. and Al-Jaar, R.Y. (1995). "Application of Petri Nets in Manufacturing Systems." IEEE Press marketing.

Drewes, F., Hoffmann, B. and Plump, D. (2000). "Hierarchical graph transformation. "in *Foundation of Software Science and Computation Structures (FOSSACS 2000)*,

Duesterwald, E. (2003). "Dynamic Compilation." In Srikant, Y.N. and Shankar, P.(Eds). " The Compiler Design Handbook: Optimizations and Machine Code Generation." USA. CRC Press LLC. pp. 739-762.

Fernandes, J.M., Adamski, M. and Proenca, A.J. (1997)."VHDL Generation From Hierarchical Petri Net Specifications of Parallel Controllers." IEEE Proceedings-E Computers and Digital Techniques. pp 127-137.

Frey, G. (2000). "Automatic Implementation of Petri Net based Control Algorithms on PLC." Proceedings of the American Control Conference ACC2000, Chicago.

Frey, G. and Litz, L. (2000). "Formal methods in PLC programming." Proceedings of the IEEE Conference on Systems Man and Cybernatics SMC 2000, Nashville.

Frey, G. and Minas, M. (2000) "Editing,Visualizing, and Implementing Signal Interpreted Petri Nets." *Proceedings of the AWPN 2000.* pp 57-62.

Ganapathi , M., Fischer, C.N. and Henessy, J.L. (1982) "Retargetable compiler code generation", *Computing Surveys 14.* pp 573-592.

Gau, C.H. and Hsiung, P.A. (2002) "*Time-memory scheduling and code generation of real-time embedded software*," In Proc. of the 8th International Conference on Real-Time Computing Systems and Applications (RTCSA'2002).

Graham, S.L. and Glanville, R.S. (1978). "A new method for compiler code generation." *Fifth Symposium on Principles of Programming Languages.* 231-240.

Guillemaud, L. and Gueguen, H. (1999). "Extending Grafcet for the specification of control hybrid systems." *IEEE SMC'99*, Tokyo.

Hoffmann, B. and Minas, M. (2000) "A generic model for diagram syntax and semantics." Proceedings of the Satellite Workshops of the 27th International Colloqium on Automata, Languages and Programming. No.8. pp 443-450.

Hsiung, P.A., Lee, T.Y. and Su, F.S. (2002). "Formal Synthesis and Code Generation of Real-Time Embedded Software using Time-Extended Quasi-Static scheduling," *apsec*, p. 395, Ninth Asia-Pacific Software Engineering Conference (APSEC'02).

Jorg, D. and Esparza, J. (1995). "Free Choice Petri nets." Cambridge University Press.

Lee, G.B., Han, Z. and Lee, J.S. (2004)."Automatic generation of ladder diagram with control Petri Net." Journal of Intelligent Manufacturing, Vol. 15, No. 2, pp.245-252.

Machado, R.J., Fernandes, J.M. and Proenca, A.J. (1997) "Specification of Industrial Digital Controllers with Object-Oriented Petri Nets", IEEE International Symposium on Industrial Electronics (ISIE '97). pp 78-83.

Katzen, S. (2003). "The Quintessential PIC Microcontroller." 2nd Edition. London. Springer-Verlag Ltd.

Kurdthongmee, W. (2003). "ertCPN: The adaptations of the coloured Petri-Net theory for real-time embedded system modeling and automatic code generation." Songklanakarin Journal of Science & Technology, 2003, 25(3): pp 381-394

Mak, R. (1996). "Writing Compilers and Interpreters: An Applied Approach Using C++." 2nd Edition. John Wiley & Sons.

Melzer, S. (1997) "Design and Implementation of a C-Code Generator for $B(PN)^2$." Institut fur Informatik, Univesitat Hildesheim.

Michel, G. (1990). "Programmable Logic Controllers: Architecture and Applications." England. John Wiley & Sons Ltd.

Minas, M. (1998). "Hypergraphs as a Uniform Diagram Representation Model", In Proc. 6th International Workshop on Theory and Application of Graph Transformations (TAGT '98), Germany.

Minas, M. (1999). "Creating Semantics Representation of Diagrams." Int. Workshop on Applications of Graph Transformation with Industrial Relevance (AGTIVE '99) At Monastery Rolduc, NL.

Minas, M. (2002). "Specifying Graph-like Diagrams with DIAGEN." Electronic Notes in Theoretical Computer Science 72. No. 2.

Minas, M. and Frey, G. (2002). "Visual PLC-Programming using Signal Interpreted Petri Nets." Proceedings of the American Conference 2002 (ACC2002), Anchorage, Alaska. pp. 5019-5024.

Minas, M. and Viehstaedt, G. (1995). "A Generator for Diagram Editor Providing Direct Manipulation and Execution of Diagrams." IEEE Proc.of VL'95.

Mortenson, K.H. (1999). "Automatic Code Generation from Coloured Petri Nets for an Access Control System." Second Workshop on Practical Use of Coloured Petri Nets and Design. pages 41-58.

Mirkowski, J. and Yakovlev, A. (1998). "*A Petri Net Model for Embedded Systems.*" Proceedings of the Workshop on Design and Diagnosis of Electronic Circuits and Systems Szrzyck. Poland.

Muchnick, S.S. (1997) "Advanced Compiler Design and Implementation." USA. Academic Press.

Oliveira, W.L.A. and Marranghello, N. (2000). "A High-level Petri Net for Digital Systems." Proceedings of the XV SBMicro – International Conference on Microelectronics and Packaging. pp.220–225.

Predko, M. (2002). "Programming and Customizing the PIC Microcontroller." McGraw-Hill.

Proth, J.M. and Xie, X. (1996). "Petri Nets: A tool for design and management of manufacturing systems." England. John Wiley & Sons Limited. ()

Qin, W. and Malik, S. (2003). "Architecture Description Languages for Retargetable Compilation." In Srikant, Y.N. and Shankar, P.(Eds). " The Compiler Design Handbook: Optimizations and Machine Code Generation." USA. CRC Press LLC. pp. 535-564.

Rekers, J. and Schurr, A. (1995). "A Graph Grammar Approach to Graphical Parsing." Proc. VL'95 – 11th Int. IEEE Symop. On Visual Languages, Darmstadt, Germany. 195-202. IEEE CS Press, Los Alamitos, USA.

Rekers, J. and Schurr, A. (1996). "A Graph Based Framework for the Implementation of Visual Environments." IEEE Symp. on Visual Languages.

Su, F.S. and Hsiung P.A. (2002), "Extended quasi-static scheduling for formal synthesis and code generation of embedded software." Proc. of the 10th IEEE/ACM International Symposium on Hardware/Software Codesign (CODES'2002).

Tanabe, J.M. (1997). "Timed Petri Nets and Temporal Linear Logic." In Azema, P. and Balbo, G. (Eds). " *Application and Theory of Petri Nets-97* " New York. Springer-Verlag, pp. 156-174.

Wallen, A. (1995). "Using Grafcet To Structure Control Algorithms." Proceedings of The Third European Control Conference, Rome, Italy.

Watt, D.A. and Brown, D.F. (2000). "Programming Language Processors in Java: Compilers and Interpreters." England. Pearsons Education Ltd.

Watt, D.A. and Brown, D.F. (2001). "Java Collections : An Introduction to Abstract Data Types, Data Structures and Algorithms." England. John Wiley & Sons Ltd.

Zhang, K.B., Orgun, M.A. And Zhang, K. (2002). " Visual Language Semantics Specification in the VisPro System." Pan Sydney Area Workshop on Visual Information Processing (VIP 2002).

Zhou, M. and Venkatesh, K. (1999). "Modeling, Simulation and Control of Flexible Manufacturing System." Singapore. World Scientific.

(1998) "MPLAB IDE, Simulator, Editor Users' Guide." Microchip Technology.