Transformation from Petri Nets Model to Programmable Logic Controller using One-to-One Mapping Technique

Devinder Thapa, Suraj Dangol, and Gi-Nam Wang

Department of Industrial & Information Systems Engineering
Ajou University Suwon 442-749, Korea
Email :{debu,suraj,gnwang}@ajou.ac.kr

Abstract- **Agile and flexible manufacturing system makes it mandatory that a control program should have a features such as agility, flexibility, and reusability in order to meet fast changing customer demands. Proper modeling and fast formation of the control logics using graphical and mathematical tools is one of the most peculiar ways to make the control logic more feasible for rapid product prototyping, concurrent engineering, flexible, and agile manufacturing. This paper illustrates efficient conversion of Petri nets model into programmable logic controller like LLD (Ladder Logic Diagram) in a very convenient way using one-to-one mapping technique. Furthermore, property analysis has been briefly explained to generate flawless control code for better operational performance. An example of painting robots has been demonstrated to prove the proposed method. Keywords: Petri Nets (PN), Agile and Flexible Manufacturing (AFM), Programmable Logic Controller (PLC), IEC 61131-3**

I INTRODUCTION

One of the most important factors of success in today's competitive market is to meet the customer need in time. The functional operation of any manufacturing unit should be smooth enough to run the operation without any work stoppage. A delay in generating controlling code of such devices may cause a big loss of revenue and goodwill. One of the major requirements of such industries is to use rapid modeling and compiling tools to generate ready-to-use logic controller codes. A logic controller is considered to be efficient if it precisely describe the functional, standard and non-functional specification [6].

The objective of our paper is to use robust modelling techniques, Petri nets for graphical and mathematical representation, to do property analysis, and to generate error free LLD (Ladder Logic Diagram). The reason for using Petri net is due to its concurrent, asynchronous, distributed, parallel, nondeterministic, and stochastic feature [1]. We can extent the model by adding some additional parameters but keeping the core requirements intact. Petri nets have the capability to model, analysis and visualization of the logic control algorithms. This paper has explained the one-to-one mapping technique to transform the purified model into LLD clearly. LLD is one of the most widely used logic controller programs over US and Asia. It is easy to design logic program, but as soon as the length of the program grows, it becomes complicate to check and modify the model completely. Formalization of informal specification using Petri nets model and converting it into IEC 61131-3 codes will be an efficient and effective way to reduce cost, time, and to improve quality of the agile and flexible manufacturing system. This paper has explained the complete transformation process in a very convenient and efficient way.

1) Related Work

After the origination of Petri nets much of the academic research has been done in this subject. Due to the complexity of the LLD, lot of industrial researchers have also started to explore the possibilities of using Petri nets tool as a modeling and analysis technique [5]. Many theses have been written about Petri nets but still it is hard to find a complete solution for the manufacturing systems [8]. Most of the previous work is suitable for small systems but the comprehensive result for flexible manufacturing system is yet to come. Reference [2] has made a comparative analysis of PLC programming using ladder logic and Petri nets. It proves the flexibility and usability of the Petri nets. References [2] and [3] have given a good explanation of transformation process but it is still hard to find practical implementation in real agile manufacturing environment. One of the main problems with the programming logic controller is lack of unanimity. In this paper, we tried to describe clearly a very fundamental technique of transformation from Petri nets model to PLC code. A focus is given to illustrate the Petri nets as a supporting tool to make PLC code more powerful and error free easily. Comparison to previous works is also presented in a convenient way the one-to-one mapping technique for easy transformation of Petri net model into ladder diagram.

2) Programmable Logic Controller (PLC)

PLC or programmable logical controller has replaced the hard wired relay logics which were difficult to reuse. The advent of the PLC began in the 1970s, and has become the

Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation, and International Confe
Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'05)
0-7695-2504-0/05 $20.00 © 2005 IEEE

most common choice for manufacturing controls. To maintain the homogeneity and portability IEC (International Electro Technical Committee) has defined some standard known as IEC 61131-3. It defines, as a minimum set, the basic programming elements, syntactic and semantic rules for the most commonly used programming languages. IEC 61131-3 defines 5 types of logic programs which are as follows [11]:

Table I
The IEC61131 Development Languages

Language	Description
Sequential Function Chart (SFC)	Petri-net derived means of describing sequential behaviour of a control program. Used for defining control sequences that those are time/event driven.
Structured Text (ST)	Pascal derived procedural language
Function Block Diagram (FBD)	Graphical language for depicting signal and data flows through re-usable software elements
Ladder Diagram (LD)	Graphical language based on relay ladder logic extended to permit the already established use of function blocks to support hierarchical program design
Instruction List (IL)	An assembly-like language for low-abstraction PLC programming

In this paper, LLD are used for the implementation purpose due to its popularity and wide availability in the automobile industries.

3) Formal Definition of Petri Nets

A Petri net is one of the robust modelling and analyzing tools for representations of discrete distributed systems. As a modelling language, it graphically describes the structure of a distributed system as a directed bipartite graph with explanation. A Petri net is a 5-tuple, PN= (P, T, F, W, M_0) where:

- P={P_1,P_2,P_3…..,P_m} is a finite set of places
- T={t_1,t_2,t_3,……..t_n} is a finite set of transitions
- F = set of arcs (Flow relation)
- W:F->{1,2,3…} is a weight function
- M_0:P->{0,1,2,3…} is the initial marking

At any one time during a Petri net's execution; each place can hold zero or more tokens. Unlike more traditional data processing systems that can process only a single stream of incoming tokens, Petri net transitions can consume tokens from multiple input places, act on them, and output tokens to multiple output places. Before acting on input tokens, a transition waits until the following two conditions are met:

- A required number of tokens appears in every one of its input places, and
- The number of tokens in each of its output places falls below some threshold.

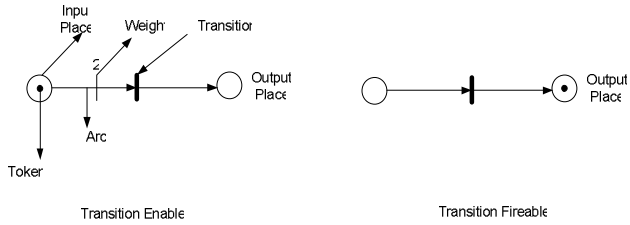

Fig. 1. Transition Enable/Firing rule

Transitions act on input tokens by a process known as firing. When a transition fires, it consumes the tokens from its input places, performs some processing task, and places a specified number of tokens into each of its output places. It does this atomically, namely in one single non-preemptible step. Since more than one transition on a net can be firing at any one time, Petri nets are well suited for modelling concurrent behaviour [1].

Different characteristics of Petri Net modelling and the relationship among them can be classified as follows:

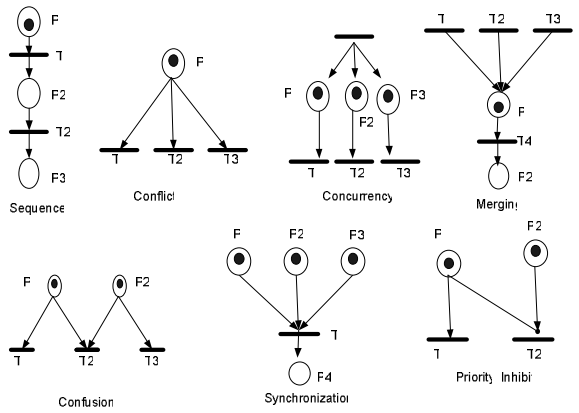

Fig. 2. Primitive structure of Petri Nets

Sequence: Two transitions cannot occur together. (P1[T1>P2 or P1[T1> and ¬ P2[T2>

Conflict: Two transitions can occurs independently but both cannot occur together P1 [T1> or P1 [T2 and ¬ P2 [T1, T2

Concurrence: Two transitions can fire together. P1 [T1> and P2 [T2> and P3 [T3>

Confusion: is another not so obvious construct. It is a combination of conflict and concurrency. P1 enables both T1 and T2, but if T1 fires, T2 is no longer enabled.

Merging: is not quite the same as synchronization, since there is nothing requiring that the three transitions fire at the same time, or that all three fire before T4; this simply merges three parallel processes

The priority/inhibit: construct uses the inhibit arc to control T1; as long as P1 has a token, T2 cannot fire.

Synchronisation: T1 will be enabled only when tokens from P1, P2, P3 arrives into the place currently without token.

The rest of the paper consists of the following topics. Section II explains about the designing of the transformation process, property analysis and methods. Concluding remarks and references are given in section III.

II. DESIGNING OF TRANSFORMATION PROCESS

To get a proper and effective controller logic algorithm we have to follow the following process:

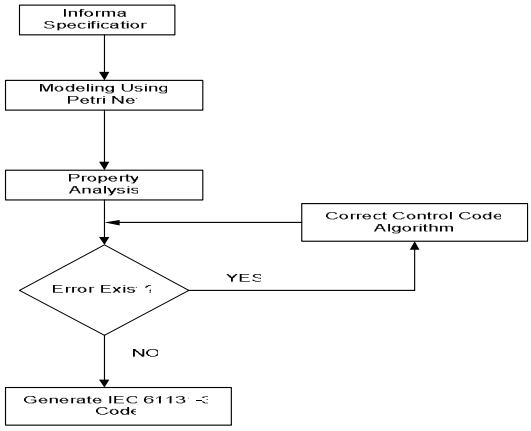

Fig. 3. Transformation process

1) Informal Specification

This paper has explained a scenario of the automated manufacturing system depicted in (fig. 4). A robot takes a piece from the input buffer (P1) and loads either machine 1 (P2) or machine 2 (P3). The (P2) and (P3) are equivalent and they paint pieces in different colors. Each machine unloads its painted part into buffer 1 (P6) or buffer 2 (P7) respectively, both of finite capacity. A third machine 3 (P8) loads the two painted parts from the buffers (P6 or P7) welds the parts and release the final composed project to an output buffer (P10).

The following set of specifications is considered for the desired operation of the manufacturing cell:

- If machine M_1 or M_2 is painting the robot can not load a piece in this machine
- When the robot finish loading a part in M_1 or M_2, then the robot comes back to idle position
- The machines M_1 and M_2 are painting only if the robot has loaded part

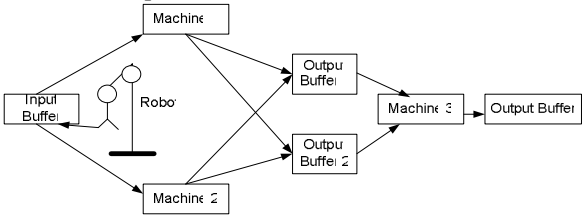

Fig. 4. Informal Specification of the scenario

2) Formalization of algorithm using Petri Nets Model

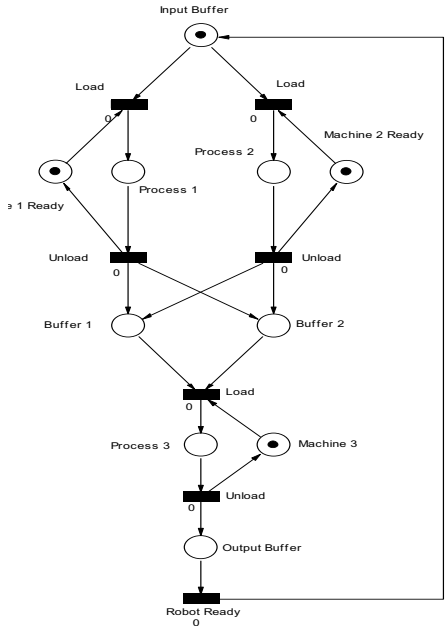

Fig. 5. Formalization of control algorithm

A detail of the places and transitions has been given in a tabular form.

Table II
Symbolic description of Petri nets model

Place	Description
P1	Input buffer
P2	Machine 1 Ready
P3	Machine 2 ready
P4	Machine 1 process
P5	Machine 2 process
P6	Output buffer 1
P7	Output buffer 2
P8	Machine 3 Ready
P9	Machine 3 processing
P10	Output Buffer
Transitions	Description
T1	Loading from P1 to P4
T2	Loading from P1 to P5
T3	Transferring from P4 to P6
T4	Transferring from P5 to P7
T5	Loading from P6 or P7 to P9
T6	Transferring from P9 to P10
T7	Robot idle and ready to load
Functional Specification	
If P2 or P3 is processing Robot cannot load a piece in the machine	
When machine P2 and P3 finish processing robot ready to load another part	
The machines P2 and P3 are processing only if the robot has loaded part	

3) Property Analysis

A. Reachability

Whether a system can reach a specific state or exhibit a particular functional behaviour. It is necessary to find such a sequence of firings of transitions which would transform a marking M_0 to M_i where mi represents the specific state, and the sequence of firings represents the required functional behaviour. A marking mi is said to be reachable from a marking mo if there exists a sequence of transitions firings which transforms a marking M_0 to M_i.

B. Boundedness and safeness

The Petri net property which helps to identify in the modeled system the existence of overflows is the concept of bounded ness. A place p is said to be k-bounded if the number of tokens in p is always less or equal to k (k is a non negative number) for every marking m reachable from the initial marking M_0, $M \in R(M_0)$.

C. Conservativeness

A Petri net is said to be conservative is there exist a weight vector W=(w1,w2,w3…….wn). If each transaction has the same number of input and output arcs, then the net will be strictly conservative.

D. Liveness

This implies that for all markings m, which are reachable from the initial marking m0, it is ultimately possible to fire any transition in the net by progressing through some firing sequence. A transition t in a Petri net is said to be:

- L0-live (or dead) if there is no firing sequence in L (M_0) in which t can fire.
- L1-live (potentially fireable) if t can be fired at least once in some firing sequence in L (M_0).
- L2-live if t can be fired at least k times in some firing sequence in L (M_0) given any positive integer k.
- L3-Live if t can be fired infinitely often in some firing sequence in L (M_0).
- L4-Live-(or Live) if t is L1-Live in every marking in R (M_0).

E. Reversibility and Home State

These systems are required to return from the failure states to the preceding correct states. A Petri net, for the initial marking mo, is said to be reversible if for each marking m in $R(M_0)$, M_0 is reachable from m..

4) Analysis Method

The existence of the one to one functional correspondence between an original requirements specification and its Petri net representation allows to project the analysis results, obtained fro the Petri net model, onto the original description. Due to a vague, incomplete, and frequently incorrect perception of the system functionality we need to do proper analysis of all the above mentioned properties. Brief discussion of the analysis methods has been given; however, detail explanation is out of the scope of this paper. The analysis methods are as follows:

1) Reachability (Coverability) Tree :

Karp-Miller algorithm

1.0) Let the initial marking M_0 be the root of the tree and tag it new

2.0) While "new" markings exist do the following:

3.0) Select a "new" marking m

- 3.1) if m is identical to another marking in the tree, then tag M old", and go to another "new"marking
- 3.2) if no transitions t enabled in marking m tag m "deadend"

4.0) For every transition t enabled in marking M do the following:

- 4.1) Obtain the marking m' which results from firing t in M
- 4.2) If on the path from the root to m, there exists a marking M '' such that M'(p)>= M"(p) for each place p, and M' != M", then replace M'(p) by ω for each p whenever M'(p)> M"(p)
- 4.3) Introduce M' as a node, draw an arc from M m to M' labelled t, and tag M m" "new".

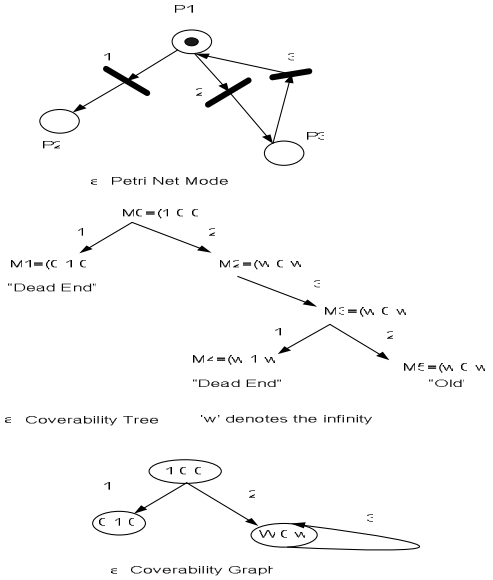

Fig. 6. Reachability tree and coverability graph

2) Invariant analysis

It describes the dynamic behaviour of Petri nets using some equations. Matrix equations are basically used to define the concurrent behavior of Petri nets model. However, the solvability of these equations is somewhat limited, partly because of the non deterministic nature inherent in Petri-net models and because of the constraint that solutions must be found as Non-negative integers [1]. Invariant analysis uses incidence matrix and state equation that govern the dynamic behaviour of concurrent systems modulated by Petri nets.

A. Incidence matrix

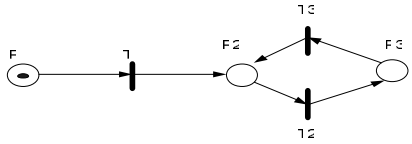

Fig. 7. Petri net Model

Table 3. Incidence matrix representation

T1	T2	T3	
-1	0	0	P1
1	1	-1	P2
0	-1	1	P3

B. State equation

Necessary condition for marking M to be reachable from initial marking M_0:

$$M=M_{0+Av}$$

Reach ability of $M=|0\ 0\ 1|^T$ from $M_0=|1\ 0\ 0|^T$

$$v1=\begin{vmatrix}1\\0\\1\end{vmatrix}\quad\begin{vmatrix}0\\0\\1\end{vmatrix}=\begin{vmatrix}1\\0\\0\end{vmatrix}+\begin{vmatrix}-1&0&0\\1&1&-1\\0&-1&1\end{vmatrix}\begin{vmatrix}1\\0\\1\end{vmatrix}$$

C. Reduction

By simplifying a subnet or structure while preserving the concerned properties, one is able to derive the properties of a complex net. The limitation lies in that reducible subnets may not exist or are difficult to find. Some reductions rules could be very hard to apply.

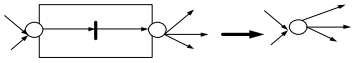

Fig. 8. Structure reduction method

D. Simulation

For complex Petri net models, discrete-event simulation is another way to check the system properties. The idea is simple, i.e. using the execution algorithm to run the net. Simulation is an expensive and time-consuming technique. The algorithm is given as follows:

a) Initialization: initial marking and the set of all enabled transitions at the marking

b) If the number of preset simulation steps or certain stopping criteria is met, stop. Otherwise, if there is no transition enabled, report a deadlock marking and either stop or go to step 1.

c) Randomly pick a transition to fire. Remove the same number of tokens from each of its inputs places as the number of arcs from that place to the transition and deposit the same number of tokens to each of its output places as the number of arcs from the transition to that place.

d) Remove enabled transitions which are modified at the new marking by checking the output transitions of the input places used in step 3. if the output transitions of the output places in step 3 become enabled, add those enabled ones. Go to step 2.

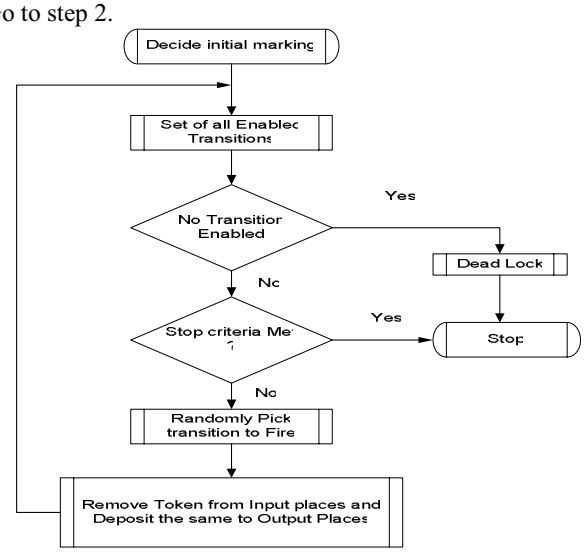

Fig. 9. Simulation algorithm

The advantage of this technique is to allow one to derive the temporal performance for a system under very realistic assumptions.

5) Implementation of Petri Nets model into Ladder Logic Diagram using one-to-one mapping technique

Due to the similarities in the construction of the Petri net and ladder logic diagram it is quite transparent and convenient to convert Petri net model into LD. The figure shows an example of the possible one to one mapping between Petri nets and ladder logic diagram.

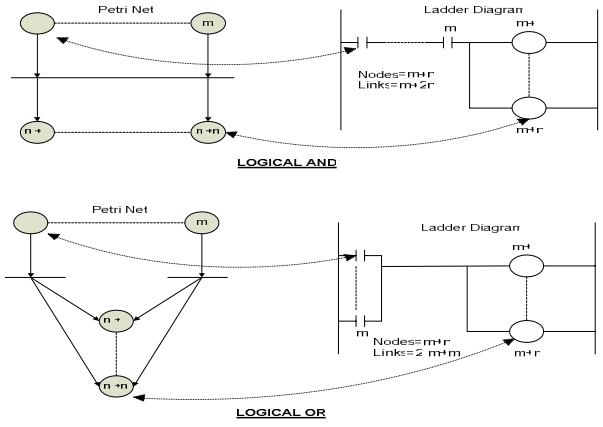

Fig. 10. One to one mapping illustration

A. Boolean equation of Petri Net model

Based on the (fig. 4) model we have developed a Boolean equation of Petri net is as follows:

P0 = P9.T7 [1]

-To activate the place P0

U = P0.(T1.P1 + T2.P2) [2]

- To deactivate place P0

We can simplify the Place and transition behaviour using combinatorial logic as follows:

$$\overline{P.U} = P0\{\overline{P0(T1P1 + T2P2)}\} \quad [3]$$

$$= P0\{(\overline{T1} + \overline{P1}x\overline{T1 + P1})\}$$

Combinatorial logic for output generation has been given below:

$$\overline{P.U} = P9\{\overline{(P9.T7.P0)}\} \quad [4]$$

$$= P9\{(\overline{T7} + \overline{P0})\}$$

This paper has explained the partial model while the method for the whole model can be generated using the similar process.

B. Transformation into ladder logic diagram

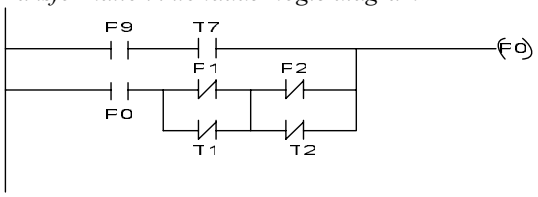

Fig. 11. One–to-one place transition behaviour mapping

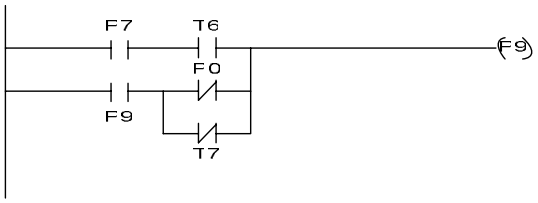

Fig. 12. One-to-one output mapping

Token parser logic can be used to map one-to-one parameter from Petri Nets model to ladder logic diagram [10]. The Petri net model has been created and tested using Visual Object ++. Visual object ++ is a free software tool for Petri net modelling and simulation. Ladder logic (LD) based on this model has been evaluated using WinTriologi (Educational version free software tool for designing ladder logic) for H-series PLC devices. The conversion process has been explained theoretically, we can use the modus operandi of lexical analysis (token creation), semantic check (Token parsing), and finally PLC code generation.

III. CONCLUSION

In a manufacturing unit one of the major concerns is uninterrupted processing and just in time production. To meet these criteria, fast transformation of the control logic code with well analysed model, is a mandatory job. Different types of modelling and verification tools have been available for this purpose. This paper has specially focused on the implementation of Petri nets based approach due to its concurrent, asynchronous, distributed, parallel, nondeterministic, and stochastic features. The only limitation of this approach is state explosion which can be solved by using hierarchical technique or reduction method. Fundamental technique of implementation of Petri net model into ladder logic diagram using one-to-one mapping technique has been explained in this paper. Our future work is related to develop tools for auto generation of IEC 61131-3 codes and integrates the verification and validations features and use XML visualization technique to meet non-functional requirements. It will be a complementary tool for virtual manufacturing system to help the automation and logic control.

REFERENCES

[1] Tadao Murata, Petri Nets: Properties, Analysis and Applicaiotns, *Proceedings of the Proc. IEEE, Vol. 77* No.4, r. 1989., pp. 541-*580

[2] Shih Sen Peng and Meng Chu Shou, Ladder Diagram and Petri-Net-Based Discrete-Event Control Design Methods, *IEEE Transactions of Systems,Man, and Cybernetics-Part C:Applications and reviews, Vol.34* No.4, 2004

[3] J-L Chirn and D.C. McFarlane, Petri Nets Based Design of Ladder Logic Diagrams, *Control 2000*, Cambridge, U.K. 2000

[4] Rene David, Grafecet: A Powerful Tool for specification of Logic Controllers, *IEEE Transactions on Control Systems Technology, Vol 3. No.3*, September 1995

[5] Richard Zurawski and MengChu Zhou,Petri Nets and Industrial Applications: A Tutorial, *IEEE Transactions on Industrial Electronics, Vol. 41 No.6*, December 1994

[6] George Frey and Lothar Litz,Formal Methods in PLC Programming,*Proceedings for the IEEE Conference on Systems Man and Cybernatics SMC 2000*,Nachville, Oct. 8-11, 2000

[7] Zoubek, Roussel and Kwiatkowska,Towards automatic verification of ladder logic programs, *IMACS Multiconference on Computational Engineering in Systems Applications (CESA)*, 2003

[8] MengChu Zhou,*Petri nets in agile and flexible manufacturing*,(Kluwer Academic Pulishers Group, 1995)

[9] J.S.Lee and P.L.Hsu, An improved evaluation of ladder logic diagrams and Petri Nets for the sequence controller design in manufacturing systems,*int J Adv Manuf Technol* (2004) 24: 279-287

[10] Thomas W. Parsons, *Introduction to compiler construction*, Hofstra University, NewYork, 1992.

[11] IEC 61131-3 (website), www.plcopen.org