PETRI NETS BASED DESIGN OF LADDER LOGIC DIAGRAMS

J-L Chirn and D. C. McFarlane

Institute for Manufacturing
University of Cambridge
Mill Lane, Cambridge, CB2 1RX, UK

Keywords: Ladder logic diagram (LLD), Programmable logic controller (PLC), Petri net (PN)

Abstract: In this study, a systematic approach to designing a ladder logic diagram (LLD) for a PLC is proposed which uses a Petri Net (PN) based modelling approach. This approach is applied to the programming of sequential logic in order to improve design efficiency and to reduce the test and maintenance effort required for complicated systems. A general method for mapping PN models to LLDs is developed which can be implemented in a wide range of applications using a variety of PLCs. Besides the sequential logic, the combinational logic of action outputs to the external environment is also considered to complete a total design of a LLD for an industrial application. Next, a design example is presented to clarify the full procedure. Finally, this approach is compared with other PN based LLD design in order to evaluate its performance.

1. INTRODUCTION

Programmable Logic Controllers (PLCs) have been the mainstays in the execution of automation tasks in a modern factory for more than 20 years. The main programming language of the PLC, a graphical symbolic language based on so-called "*ladder logic diagrams*" (LLDs), is therefore widely used in automation. Many industrial users of PLCs prefer to program LLD using heuristic methods. For simple systems, it is easy to write down PLC programs using the heuristic method. However, as a system gets more complex it becomes very difficult to handle problems effectively [Chirn, J-L, 1999]. These problems have been recognised ever since LLD has been widely used. Some higher-level design tools have been proposed to help resolve these problems [IEC, 1992; David, R., 1995]. Petri Nets (PNs) [Peterson, J. L., 1981] are a common tool used in this aspect because of their success in discrete event control systems (DECS) design. Some researchers have compared PN and LLD design and have suggested that PNs possess superior properties than LLDs in terms of design complexity and response time [Venkatesh, K., et al, 1998, Zhou, M. et al, 1995]. Thus, the PN model is recommended new design tool instead of LLDs.

Because of the planning and organisational advantages of PNs, a number of researchers have attempted to develop methods to transfer PNs which have been developed in the design phase to LLDs implementations [Sato, T. et al, 1995; Jafari M. A. et al, 1994; Burns, G. L. et al, 1994; Taholakian, A. et al, 1997]. Furthermore, Uzam, et al [Uzam, M., et al, 1996] have proposed a token passing ladder logic methodology (TPLL) to directly convert more general PNs into LLDs, including the coloured and timed PNs. These approaches, however, typically focus only on the design phase rather than the other phases of the control system development. Their methodologies aim simply to translate the behaviour of PNs into the syntax of LLDs. However, problems in the test phase and later maintenance phase are still outstanding. For example (see Figure 1), when the bugs or faults occur during real-time operations, it is difficult for engineers to trace the LLDs. Therefore, a means to make the program more readable in order to locate the fault in LLDs is a vital issue at this stage. Not only design time but also debugging and maintenance time can be reduced if an appropriate approach is available. This issue, which has not yet been focused before, will be taken into account in this study, through a transparent relationship between LLD and the PN it represents.

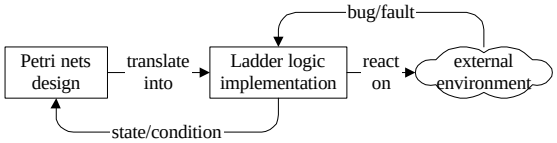

Figure 1. Relationships among PN, LLD and real world

Furthermore, this study intends to develop a unified approach for LLD development which applies to a wide range of PLCs despite very little standard syntax existing in PLC programming. Moreover, the more general topology of PNs is considered in this study to cover wider applications. Apart from generating the LLDs which can implement the sequential control behaviour of a PN model, the combinational logic of output actions to the external environment in a complex system is also taken into account to achieve a complete PLC application. This has not been established in previous work, yet is a critical element in developing an appropriate maintenance and monitoring strategy. The details are explained in the following sections.

2. DESIGN METHOD

The proposed conversion method from PN to LLD is developed in this section. The basic concept of PN is introduced first. Then, a mapping method from a primitive PN to a LLD is presented. This method is then modified to make sure that it can be applied to various PLCs, and extended to the more general topological structure of PN. Next, a simplified mapping is developed which simplifies the application in most cases. Finally, the issue of internal conditions and the action outputs is considered to develop a complete program for the PLC.

2.1 A Primitive Mapping Rung

Before developing this approach, the basic definition of PN is introduced first. PNs consist of four basic symbols (see Figure 2) [Peterson, J. L., 1981] :

(1) a *place*, which denotes the state of system, and is represented by a circle.
(2) a *transition*, which denotes the events or conditions that can occur in the system, and is represented by a bar.
(3) an *arc*, which denotes the connection between places and transitions, and is represented by an arrow between a place and a transition.
(4) a *token*, which denotes the currently active place, and is drawn as a black dot within a place.

As soon as a place obtains a token, the state of this place becomes active. This place is called a *marked place*. At this time, the following transition of this place would be firing. After firing, this place returns to its idle status again and the token is transferred to the following place.

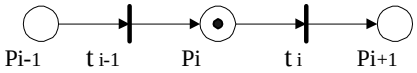

Figure 2. A simple PN model

There is generally no unique way to describe the sequential behaviour of a system. A state/condition model is frequently suggested to describe the discrete and concurrent control system. PNs and LLDs are both suitable tools to design a state/condition model. In PNs, each place can be regarded as one of the states in the system. A marked place means its respective state is progressing. The transition can represent the condition of changing one state to another state. Likewise, in LLD, the output coil can represent the status of the place in a PN, and the combinational logic of input contacts can represent the transition of a PN. Following relatively simple rules, PN models can be easily mapped to LLDs.

A LLD rung shown in Figure 3 is a self-holding circuit, which has been widely applied in LLD programming. It can be derived to implement the basic behaviour of a place in a PN: The output coil P in the LLD rung represents the status of place P in a PN. When the coil is energised, it means that its corresponding place is marked; otherwise it is idle. The contacts L and U represent the marking and unmarking conditions of the place P respectively. When contact L is turned on, the marking status of P is "latched on". Even thought L is turned off afterwards, P still remains on because of the self-holding property of the circuit. The status of P does not return to the off state until the normal-close contact U is turned off.

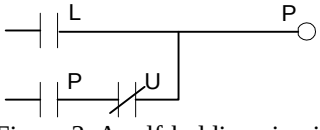

Figure 3. A self-holding circuit

The basic rung in Figure 3 is now used to illustrate the conversion of the basic structure shown in Figure 2 into LLD. Firstly, considering the latched condition of P_i in Figure 2, we note that P_i can be marked if and only if its previous place P_{i-1} is marked and its input transition t_{i-1} is fired. The *latched condition* can be written in Boolean notation as

$$L = P_{i-1} \cdot t_{i-1} \tag{1}$$

In the same manner, the unlatched condition of place P_i will occur when t_i is fired. The *unlatched condition* can be expressed as

$$U = P_i \cdot t_i \tag{2}$$

The *unlatched rung* is defined as the lower part of the self-holding circuit, which comprises both unlatched and self-holding properties. Its combinational logic can be expressed as

$$P \cdot \overline{U} \tag{3}$$

After substituting equation (2) into expression (3), the combinational logic of unlatched rung becomes

$$P \cdot \overline{U} = P_i \cdot \overline{(P_i \cdot t_i)} = P_i \cdot (\overline{P_i} + \overline{t_i}) = P_i \cdot \overline{t_i} = P_i \cdot \overline{t_i} \tag{4}$$

Therefore, the mapping rung can be obtained to represent the behaviour of P_i in Figure 2 and shown in Figure 4.

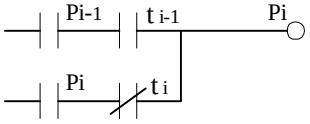

Figure 4. A primitive mapping rung

However, as we will show in the following section, this mapping method can result in a critical implementation error.

2.2 A Modified Mapping Rung

The mapping method developed above was applied to different PLC systems to test its performance. Two scanning modes are usually employed in PLC systems to evaluate LLDs during operation. The first is rung by rung scanning, or *horizontal scanning*, and the other is column by column scanning, or *vertical scanning* [Kissell, T. E., 1986]. In our test, the mapping method developed in Section 2.1 can run well in vertical scanning type PLC, but, unfortunately, cannot perform the expected control sequence in horizontal scanning type PLC. The reason is explored below and then a modified mapping method is developed to solve this problem.

The converted rungs for places P_i and P_{i+1} of Figure 2 are shown in Figure 5. The purpose is to examine its performance in horizontal scanning type PLCs. P_i is initially assumed to be a marked place. That is, the coil of P_i is energised. Coil P_i is turned off after contact t_i turns

on. In theory, P_{i+1} will be marked afterwards because contact t_i has been turned on in line C. In fact, the coil P_{i+1} cannot be latched in line C because P_i has been turned off after execution of line B. Due to the asynchronous characteristics of the LDD execution, the token-passing function of PNs cannot be repesented by this mapping method. The control sequence can be depicted by Figure 6 to describe why the control sequence cannot continue.

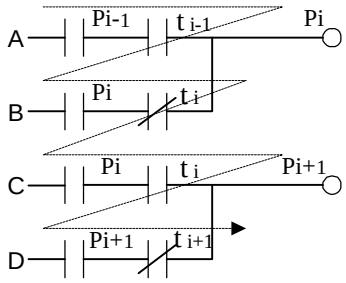

Figure 5. A mapping result of Figure 2

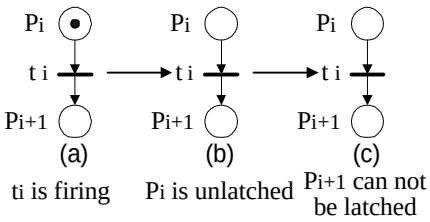

Figure 6. The sequence of token-passing in Figure 5

A revised control sequence to implement the token-passing property in an asynchronous condition can be found by swapping the unlatched and latched sequences as shown in Figure 7. After t_i is triggered, P_{i+1}'s latched condition occurs earlier than P_i's unlatched condition. As a result, the token can be passed to the following place successfully by this means.

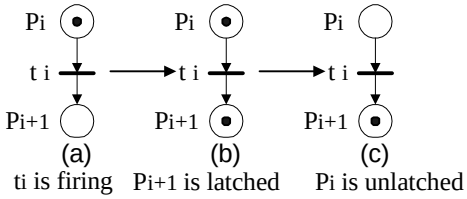

Figure 7. Revised sequence for Figure 5

One method to execute the sequence of Figure 7 is to modify the original unlatched condition in Equation (4). Observing Figure 7, if the unlatched condition of P_i is revised to become

$$U = P_i \cdot t_i \cdot P_{i+1} \tag{5}$$

then the condition will not hold until the status of P_{i+1} becomes true. Therefore, the condition corresponds to the sequence of Figure 7.

To implement the modified mapping rung, the unlatched rung can be changed to

$$P \cdot \overline{U} = P_i \cdot \overline{(P_i \cdot t_i \cdot P_{i+1})} = P_i \cdot (\overline{t_i} + \overline{P}_{i+1}) \tag{6}$$

The final modified mapping rung is shown in Figure 8. It is applicable to both horizontal and vertical scanning types of PLCs.

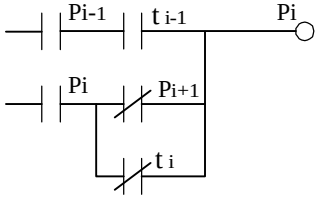

Figure 8. A revised mapping rung

The result can be further extended to a more general structure of PNs. This generalisation can be divided into four additional types of element and are shown in Figure 9. Types I and II represent for the generalised mapping of latched rungs, while types III and IV represent the generalised mapping of unlatched rungs. The derivation of type IV is further explained below. Type IV contains a place P which has multiple output transitions. The unlatched condition of the place P will hold after any one of the output transitions fires. It can be expressed as

$$\begin{aligned} U &= P \cdot t_1 \cdot P_1 + P \cdot t_2 \cdot P_2 + \cdots + P \cdot t_n \cdot P_n \\ &= P \cdot \sum_{i=1}^{n} t_i \cdot P_i \end{aligned} \tag{7}$$

The unlatched rung can be obtained by substituting equation (7) into (3). It can be written as

$$P \cdot \overline{U} = P \cdot \prod_{i=1}^{n} (\overline{P_i} + \overline{t_i}) \tag{8}$$

2.3 A Simplified Unlatched Rung

A simplified set of mapping rungs for PN types III and IV is now introduced to decrease the use of contacts as well as to simplify the structure of rungs, especially in more complicated control systems. Recalling from the previous modified mapping method that equation (5) can be used to implement the expected control sequence of Figure 7, an alternative modified unlatched condition can be written as

$$U = P_i \cdot P_{i+1} \tag{9}$$

This is a looser condition compared with Equation (5). Observing Figure 7, if t_i is not firing, P_{i+1} will never be marked. Whereas P_{i+1} will be marked when t_i is firing. That is, one of the conditions of t_i and P_{i+1} is redundant. Hence, removing the condition t_i from Equation 5 does not change the function we expect.

This revised mapping can be applied to simplify the unlatched rungs of Figure 9 and the result is shown in Figure 10.

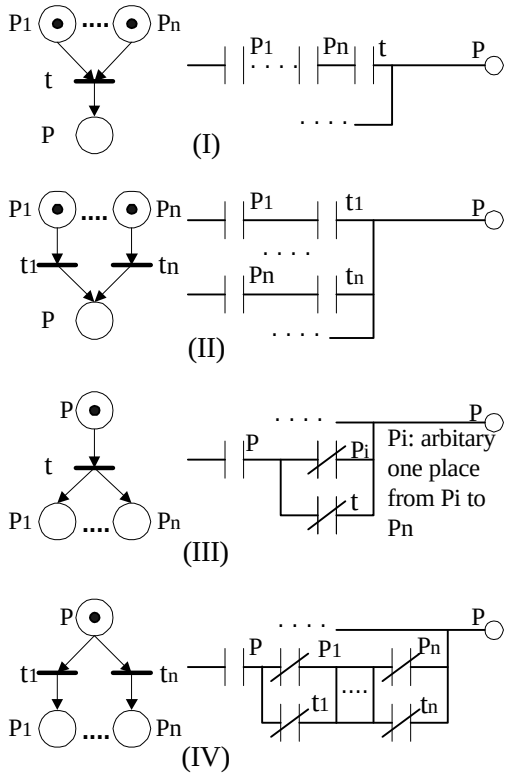

Figure 9. Generalised mapping for a place P

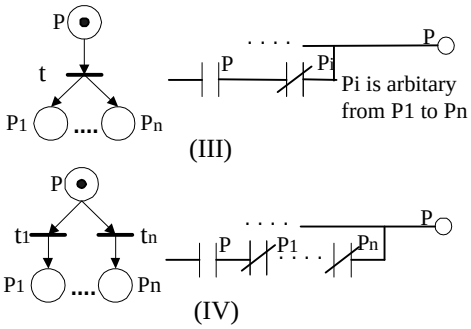

Figure 10. Simplified unlatched rungs

Remark: We note here that this conversion is similar to the concept used in [Jafari M. A. et al, 1994]. However, the use of this simplified mapping should be avoided in some cases. Figure 11 is an example of failure in the use of this simplification. This is because P_i's looser unlatched condition (see Equation (9)). P_i will lose its token when P_{i+1} is marking, although it is not triggered by its own transition t_i. This mapping leads to a malfunction in control flow. Therefore, the stricter unlatched condition of equation (5) should be used in place of P_i and P_{i+1} to avoid wrong behaviour. In general, the simplified mapping method can be applied only in a place whose output transition is the only input transition of its following place.

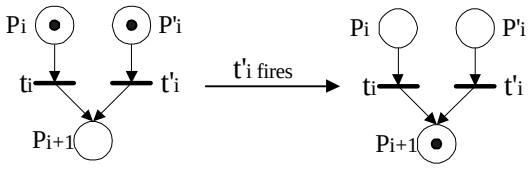

Figure 11. A failure example of simplified method

3. INTERFACING THE PLC TO THE EXTERNAL ENVIRONMENT

In this section we briefly discuss the connection between the PN based LLDs and the external environment that the logic sequences in the PLC is controlling.

3.1 Auxiliary Tables

An "*external condition table*" linking external input data to the PLC is also proposed to aid PNs to synthesise a complete structure of LLD in sequential control flow. The transitions of PNs represent these external conditions, such as the status of a limit switch, or a complex combinational logic. Therefore, the external condition table is built to list all the conditions in the system. It can assist the designers in developing their applications during the programming and testing period. An example of an external condition table is presented in the next section.

3.2 Action Rungs Implementation

The method to design action rungs which specify commands to the external environment is developed next. So far, all the topological structure of PNs (including their external conditions), has been investigated to build up the equivalent sequential behaviour in LLD form. However, another important issue is to design the output action which connects to the external environment at each state. (For example, to turn on/off a lamp, to activate/deactivate an actuator)

An important issue in developing successful LLDs should be emphasised here. *The output coil of an action must occur only once in the program*. Otherwise, a malfunction of the action could be activated by the interaction of other rungs which have the output coil of the same address. An action table is therefore proposed to aid the design of the action rungs. A suggested form of an action table is shown in Table 1.

The columns for "Switch on" and "Switch off" describe the conditions when the action is activate and deactivate respectively. Two types of trigger signals, level trigger and edge trigger, are often used to start or stop an action. A symbol with superscript asterisk denotes an edge-triggering signal; otherwise, a symbol without superscript asterisk designates a normal level-triggering signal. Normally, if an action is started by an edge-triggering signal, it will be stopped by another edge-triggering signal. Some simple examples of actions are given in Table 1 and their corresponding rungs illustrated in Figure 12.

Coil No	Action name	Switch On	Switch Off	Remarks
1	Operating light	P_1+P_2		light on when P_1 or P_2 is marking
2	Motor rotating	P_1^*	P_2^*	motor starts at P_1 and stops at P_2
....				

Table 1. An example of an action table

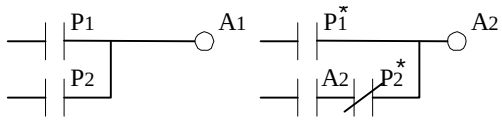

Figure 12. An example of action rungs

4. A DESIGN EXAMPLE

An example is now outlined to clarify the full PN to LLD design procedure. A test station in a transport line for detecting defective products is used as the design example. The condition of a product is checked when the product enters this test station. If a product defect is found, it is expelled; otherwise it is passed through and goes to the next station. The mechanism for this example is illustrated in Figure 13.

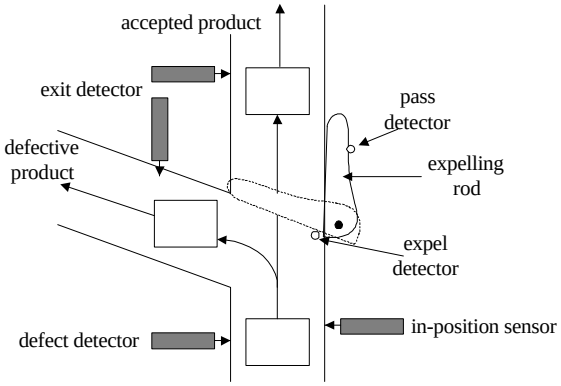

Figure 13. Layout of a design example

The control sequence for the test station is described by the PNs diagram in Figure 14. In this figure, five places (P1 to P5) represent the progressive states for operating the test station, while six transitions (T1 to T6) decide the sequence of these states. Four actions (A1 to A3) are activated in state P2 to P5 respectively, to perform the external actions.

An Omron PLC C-200H [Omron, 1994] is employed to implement the function of the PNs. The descriptive tables of transition (external condition) and action for the PNs are tabulated in Table 2 and Table 3. The converted LLD of the PN model and corresponding actions are shown in Figure 15 and Figure 16 respectively.

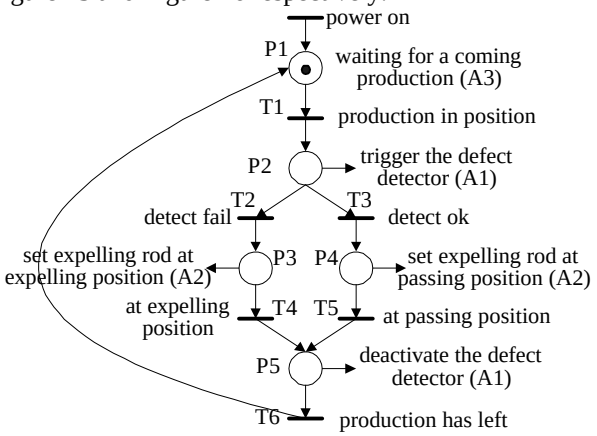

Figure 14. A PN model of the case study

Trans.No	t_i	$\overline{t_i}$ (optional)	Remarks
T1	0000.00		Product in position
T2	0000.02		Detect fail
T3	0000.01		Detect ok
T4	0000.04		At expelling position
T5	0000.03		At passing position
T6	0000.05+0000.06	$\overline{0000.05} \cdot \overline{0000.06}$	Deactivate the defect detector

Table 2. An external condition table

Action No	Coil No	Action name	Switch On	Switch Off
A1	0001.00	Activate defect detector	P_2	P_5
A2	0001.01	Set the position of the expelling rod	P_3	P_4
A3	0001.02	Turn on the busy indicator	The others	P_1

Table 3. An action table for the case study

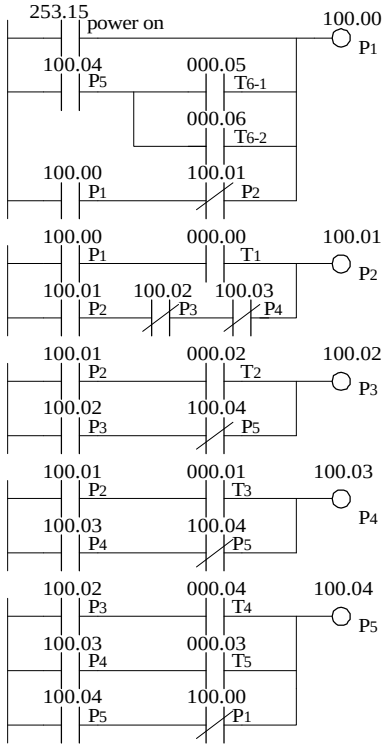

Figure 15. Converted LLD for sequential control

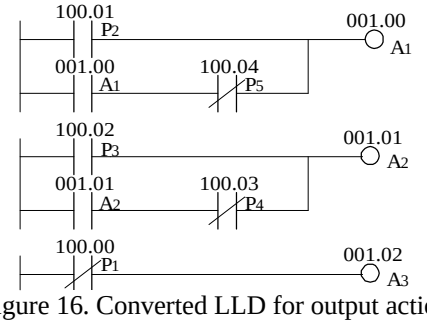

Figure 16. Converted LLD for output actions

Note that the simplified conversion can still apply to places P3 and P4 because these two places will not be marked at the same time. Therefore, the failed situations described in Figure 11 will not occur in this case.

5. EVALUATION

In this section, the characteristics of this approach is evaluated. Five criteria are proposed as a means of comparison with five other approaches:

(1) Simplicity. This index is used to assess how easy a method converts PN to LLD. The criteria include (i) Whether a direct mapping can be achieved without intermediate steps, (ii) Whether there is no need of extra knowledge.
(2) Portability. This index is used to assess how applicable a method is to a range of commercially available PLCs. The criteria include (i) The range of LLD instruction types employed, (ii) The ability to operate with different scanning types.
(3) Completeness. This index is used to assess how completely the conversion addresses the range of requirements needed for a fully operating LLD. The criteria include (i) sequential logic implementation in control flow, (ii) combinational logic implementation in action outputs.
(4) Generalisation. This index is used to assess the range of applicability to different PN model types, such as single state machine, ordinary, timed and colour PN.
(5) Testability. This index is used to assess the ease with which the design can be tested. The criteria include (i) The topologies matching between the PN and its LLD, (ii) whether the status of PN's places and actions in the PLC is traceable (iii) Ease of documentation.

Comparison results and comments for each approach are presented below and also in tabular form (see Table 4).

I: The approach in the study - apply to Boolean PNs (maximal capacity of each place is equal to one)
II: Burns [Burns, G. L. et al, 1994] - need intermediate boolean expressions, no considerations for output action implementation, apply to Boolean PNs, no action output traceable
III: Jafari [Jafari M. A. et al, 1994] - no considerations for output action implementation, apply to event graph PNs, no action output traceable
IV: Sato [Sato, T. et al, 1995] - need to design a redefined PN and sequence table, apply to Boolean PNs , not complete mapping from PN to LLD
V: Taholakian [Taholakian, A. et al, 1997] - redefine PNs and symbols, information of sequential status is lost, state can not be traced, not consider the general PNs, output action rungs are not available
VI: [Uzam, M., et al, 1996] - special LLD instructions are employed, no considerations for combinational logic of output action

The results in Table 4 indicate that the proposed developing method compares favourable with existing methods in all areas apart from its ability to handle extended PNs, such as timed PNs and coloured PNs. This is a subject of ongoing research.

Indexes	I	II	III	IV	V	VI
simplicity	●	❍	●			●
portability	●	●	●	●	●	
completeness	●			●		❍
generalisation	❍	❍		❍		●
testability	●	❍	❍			●

●: good ❍: fair blank: bad

Table 4. Comparison results for each approach

6. REFERENCES

Burns, G. L. and Bidanda, B., "The Use of Hierarchical Petri Nets for the Automatic Generation of Ladder Logic Programs", *Proceedings ESDIPC 94 Conference & Exposition*, pp 169-179, 1994.

Chirn, J-L and D. C. McFarlane, "Petri Nets Based Design of Ladder Logic Diagram", *Internal Report*, Institute for Manufacturing, Cambridge University, 1999.

David, R., "Grafcet: A Powerful Tool for Specification of Logic Controlles", *IEEE Transactions on Control Systems Technology*, Vol. 3, No. 3, 1995.

IEC, International Electrotechnical Commission. International standard IEC 1131-3, Programmable Controllers, Part 3: Programming Languages. Geneva, 1992.

Jafari, M., and Boucher, T., "A rule-based system for generating a ladder logic control program from a high-level systems model", *Journal of Intelligent Manufacturing*, 1994, Vol. 5, No. 2, pp. 103-120.

Kissell T. E., *Understanding and Using Programmable Controllers*, Prentice-Hall International Editions, Englewood Cliffs, New Jersey, 1986.

Omron, *C200H Programmable Controllers (CPU01-E/03-E/11-E) Operation Manual*, Omron Co., 1994.

Peterson, J. L., *Petri net theory and the modelling of systems*, Prentice Hall, 1981.

Sato, T., and Nose, K., "Automatic Generation of Sequence Control Program via Petri Nets and Logic Tables for Industrial Applications", *Petri Nets in Flexible and Agile Automation*, Kluwer Academic Publishers, 1995, pp. 93-108.

Taholakian, A. and Hales, W.M.M., "PN⇔PLC: A methodology for designing, simulating and coding PLC based control systems using Petri nets", *International Journal of Product Research*, Jun. 1997, Vol. 35, No. 6, pp. 1743-1762.

Uzam, M., Jones, A.,and Ajlouni, N., "Conversion of Petri nets controllers for manufacturing systems into ladder logic diagrams", *IEEE Symposium on Emerging Technology and Factory Automation, ETFA*, 1996, Vol. 2, pp. 649-655.

Venkatesh K., et al, "Discrete-Event Control Design for Manufacturing Systems via Ladder Logic Diagrams and Petri Nets: A Comparative Study", *Petri Nets in Flexible and Agile Automation*, Kluwer Academic, 1995, pp. 265-304.

Zhou, M. and Twiss, E., "A comparison of relay ladder logic programming and petri net approach for sequential industrial control systems", *IEEE Conference on Control Applications*, pp. 748-753, 1995.