VOL. 4, NO. 10, DECEMBER 2009 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

FORMALISM FOR FUZZY AUTOMATION PETRI NETS TO LADDER LOGIC DIAGRAMS

P. R.Venkateswaran[1], Jayadev Bhat[2] and S. Meenatchisundaram[1]
[1]Department of Instrumentation and Control Engineering, Manipal Institute of Technology, Manipal, Karnataka, India
[2]Department of Chemical Engineering, Manipal Institute of Technology, Manipal, Karnataka, India
E-Mail: prv_i@yahoo.com

ABSTRACT

Process automation has been the default standard for industries since processors are prominently figured in the scheme of production. The significant characteristic of process automation is clarity in the definition of tasks, sequence of operation and concurrency. Programmable Logic Controllers (PLC) are a set of this morphing of processors prominently favored in the Process Automation due to its ease of use, ruggedness and low cost. PLC is preferred because of its ease of programming. Programming of PLC was started with Ladder Logic Diagrams (LLD) and in spite of other developed high level languages plays a prominent role even today. This problem however, is recognized and programming for DES is suggested via a modified approach called Grafcet [1]. In this paper, a fuzzy formalism is introduced into the modeling system as Fuzzy Automation Petri Nets (FAPN) and formal method for conversion of this FAPN into LLD is suggested.

Keywords: model, fuzzy automation petri nets, ladder logic diagrams, programmable logic controllers.

1. INTRODUCTION

The major problem with Ladder Logic Diagrams (LLD) is that programming is done heuristically. Heuristic methods get into difficulty with increasing order of complexity. When multiple systems are considered, the problem gets magnified. Even if the system is single and is of multiproduct type, the problem persists. The problem is well recognized [3, 14] and some of the successful solutions involve Petri nets in its design. Since Petri nets have been successfully used in representation of such systems, heuristic design is replaced by conversion of Petri nets into LLD. This is reflected in [4-9]. Token Passing Logic (TPL) methodology [10] improved on this logic by providing options to include timers, counters, and flags. This is used in the conversion of Fuzzy Automation Petri Net (FAPN) into LLD. The same technique is already used to deal with timed place Petri nets [10, 11], timed-transition Petri nets [11], Colored Petri nets [11]. The TPL methodology has also been developed to embrace statement lists [12], another recent addition to PLC programming languages. The common thread in these successful applications is that the Petri net formalism is adaptable to PLC programming languages with a valid translation mechanism.

The important feature of the TPL technique is that it facilitates the direct conversion of any Fuzzy Automation Petri Net (FAPN) into a Token Passing Logic Controller (TPLC). This is achieved by not disturbing the semantics of the TPL logic and at the same time extending it to refer to the Table of values associated with the level of truth with respect to a state or place. The TPLC is a generic form of control logic which may be implemented with low level languages such as machine languages, statement lists, LLD or with higher level languages like C. This is made possible by adopting the Petri net concept of using tokens as the main mechanism for controlling the flow of logic. The flow of logic, as an extension of vagueness is let through a series of subroutines before the decision. The simulated movement hence will have counters for each place in FAPN whose capacity is greater than 1. These counters will be incremented or decremented to simulate token flow. With every increment or decrement, the values of the markings are updated in the Table. Thus, each place within the FAPN has at least an associated counter in TPLC with a value defined in the Table. Since, the provision to show a Table is unavailable in TPLC; it remains hidden but is shown in the LLD diagram. The specific nature of counter needs to be understood from the hardware specifications. In general, the formalism in TPL will have the following characteristics: If the count value of the counter is greater than zero, then the status of the counter is ‘one’ and if the count value is zero, then the status of the counter is ‘zero’. The assignment of a counter to an FAPN place through the TPL is shown in the Figure-1. The decision to increment or decrement the count value will be ruled by the fuzzy logic. In case of single capacity places with Boolean operations, the counters can be replaced by flags. The assignment of flag in FAPN is same as that of APN and is shown in fig. 1.b. It is important to stress here that the fuzzy markings shown in places will be forwarded to a Table in the LLD this is because Petri nets do not have constructs to adequately deal with actuators and sensors. Automation Petri nets [13], which extends the ordinary Petri nets to deal with discrete event control applications. To accommodate uncertainty in this applications, Fuzzy Automation Petri nets (FAPN) is introduced in this work. The objective is to establish the idea as a well defined valid formalism capable of handling uncertainty in DES.

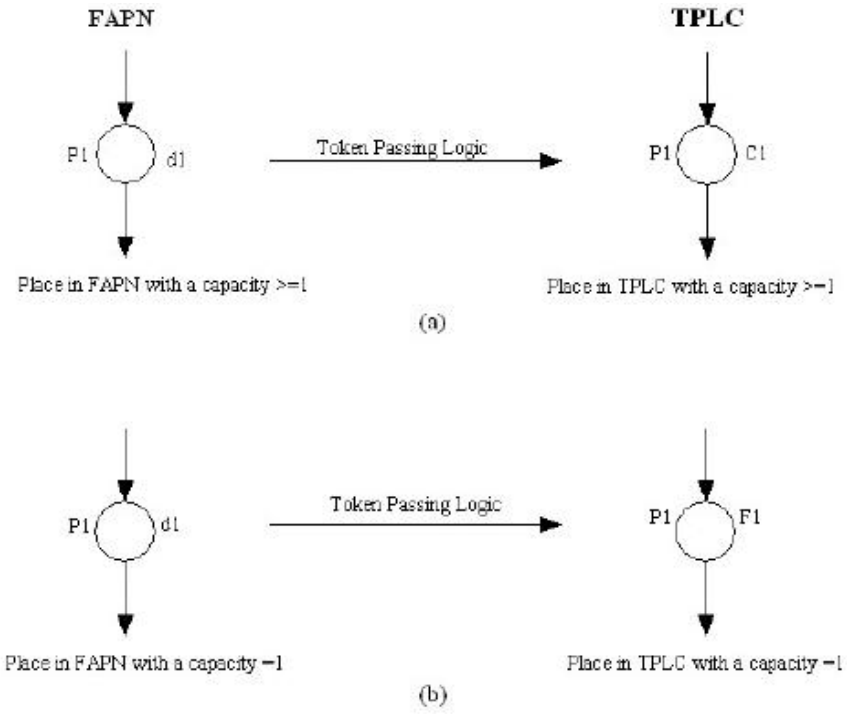

Figure-1. Equivalence between FAPN and TPLC.

2. FAPN TO LLD CONVERSION

A. Initial markings

Initial markings define the state of the system at start. This information is critical to the controllers as DES is based on the state based approach. This definitions needs to be incorporated into the LLD to ensure correct operation. The relative ease is to define the status of the places. This traditionally forms the first rung of the LLD. In order to define the initial states, an initialization flag can be used. The counters with their initial values and the flags are defined with respect to initial markings. One normally closed contact will ensure the initial settings. A subroutine is extended to define the fuzzy markings of the places. Finally, the initialization flag is reset. The LLD is illustrated in Figure-2. A subroutine is defined for defining the initial markings at more than one place.

B. FAPN without action

The primary condition for firing in any form of Petri net formalism is that there should be at least one token in the input place and the transition is enabled and the firing condition χ occurs. In addition, there will be a threshold defined in FAPN for the transition to fire. When the transition is fired it removes a token from the input places and puts a token to the output places. To convert a FAPN into a TPLC a counter or a flag is assigned to the places.

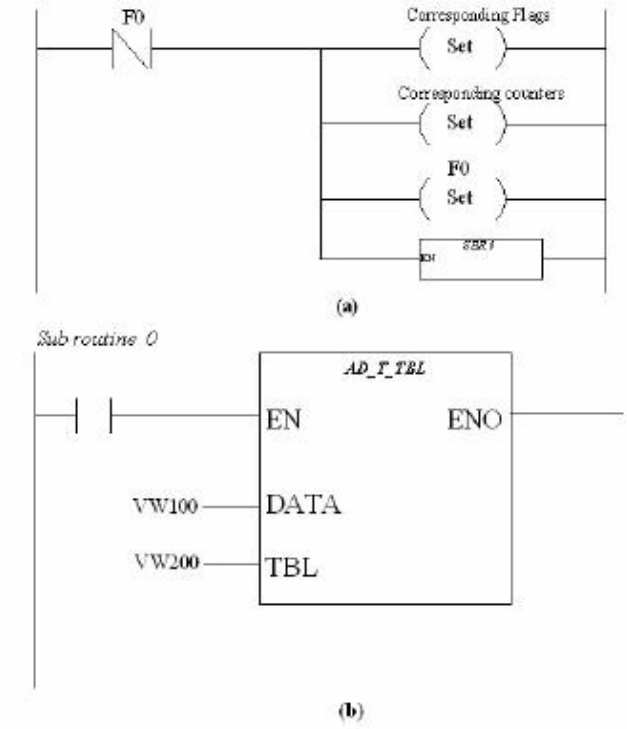

Figure-2. Initial markings in LLD.

In TPLC, each transition withdraws a token from the current logic place and adds a token to the next logic place. This is achieved by using a counter (or a flag) at each place to represent the tokens. A compare instruction is used to define the strength of the marking for the output place as against the threshold. When a transition is finally

fired, the counter for the input place will be decremented by one and the counter for the output place is incremented by one. The values of the Table are updated with new fuzzy markings. This is enabled through the subroutine. If flags are used, then the input flag is set and output flag is reset in the same way in addition to the update of the Table values. The process is illustrated in Figure-3.

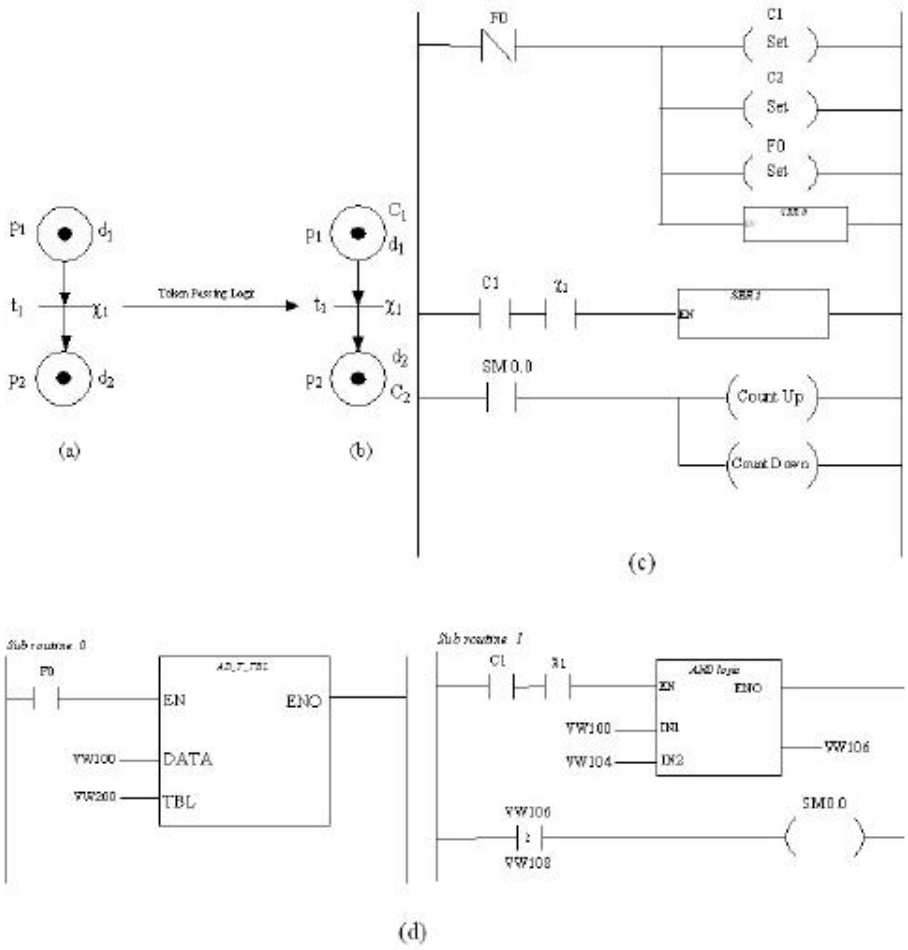

Figure-3. Flag settings in FAPN.

C. FAPN with action

Let us assign an action to the place in Figure-3. The primary requirement for an action at a place within a Petri net occurs only if the number of token at the place is non zero. Figure-4 illustrates the modification. Hence, the firing may happen either with respect to the presence of token (when the transition is already enabled) termed as impulse action or with respect to enabling of transition (when the place has a token) termed as level action. Since single tokens are associated with these places, flags instead of counters will make better option to be represented. The initial markings are not shown in the Figure-4. If the same action is assigned to more than one place, then the flags associated with the places have to be 'OR' ed together to activate the action. This implies that each action only appears once in the LLD code. If the count value of a counter at the control place is greater than zero or the related flag is set then any actions associated with the place are enabled. Associated markings are shown in terms of d_i.

D. Inhibitor arc FAPN

A simple inhibitor arc decision making is shown in Figure-5. The transition t_1 has two input places p_1 and p_2, where p_2 has an inhibitor arc, in (p_2, t_1). The transition t_1 is fired, when place p_1 has at least one token, place p_2 has no token and the firing condition $\chi_1$occurs. When it is fired, a token is removed from place p_1 and a token is deposited into the output place p_3. The change here is the marking at p_2 will not change. Only the token available at p_1 will be consumed. This operation is similar to AND operation except that one of the operands is inverted. Hence, AND construct is used for execution in the subroutine. LLD execution will take two steps once to check the logic and if the result exceeds the threshold, then the output is triggered to increment the counter at p_2 and decrement the counter at p_1. Each sub routine will have a result stored in the temporary register so that it can be used for increment or decrement of the counters.

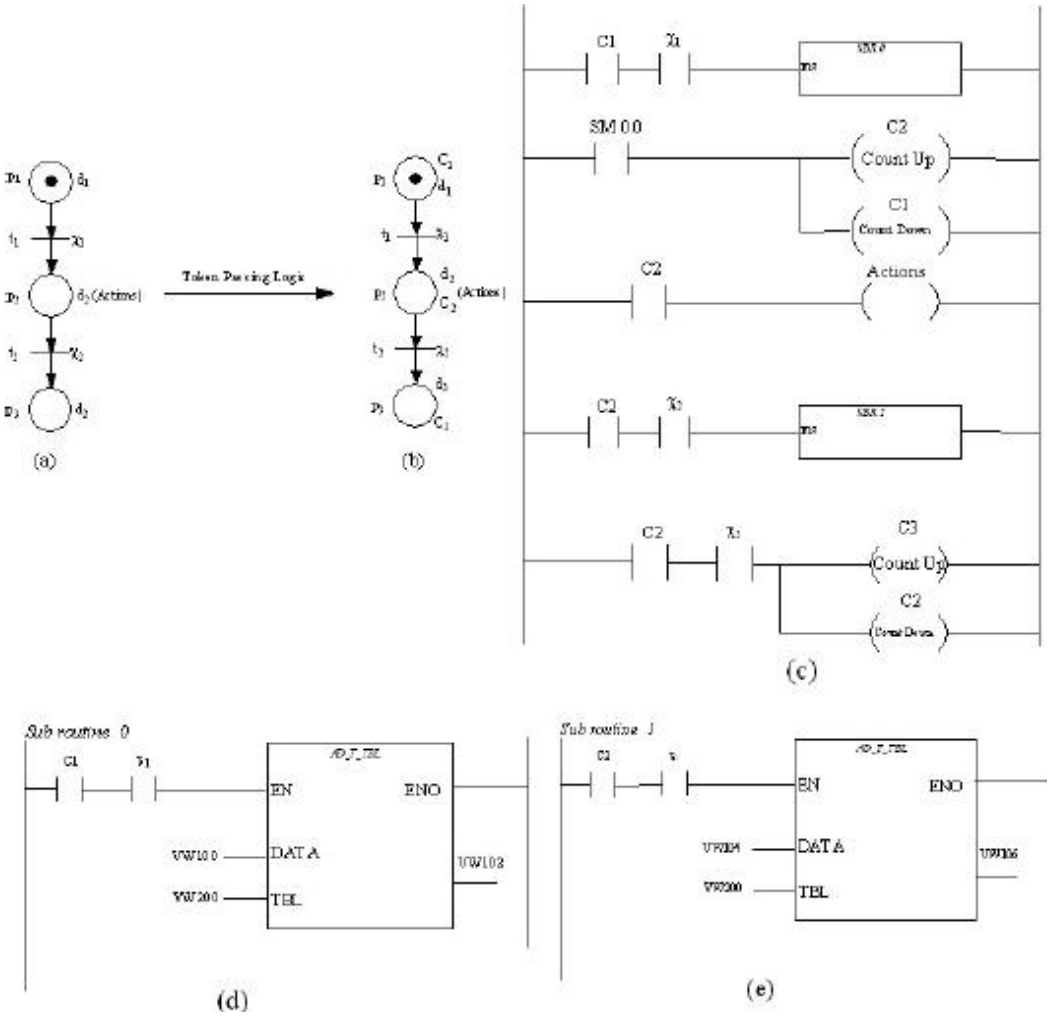

Figure-4. Tokens for actions in FAPN.

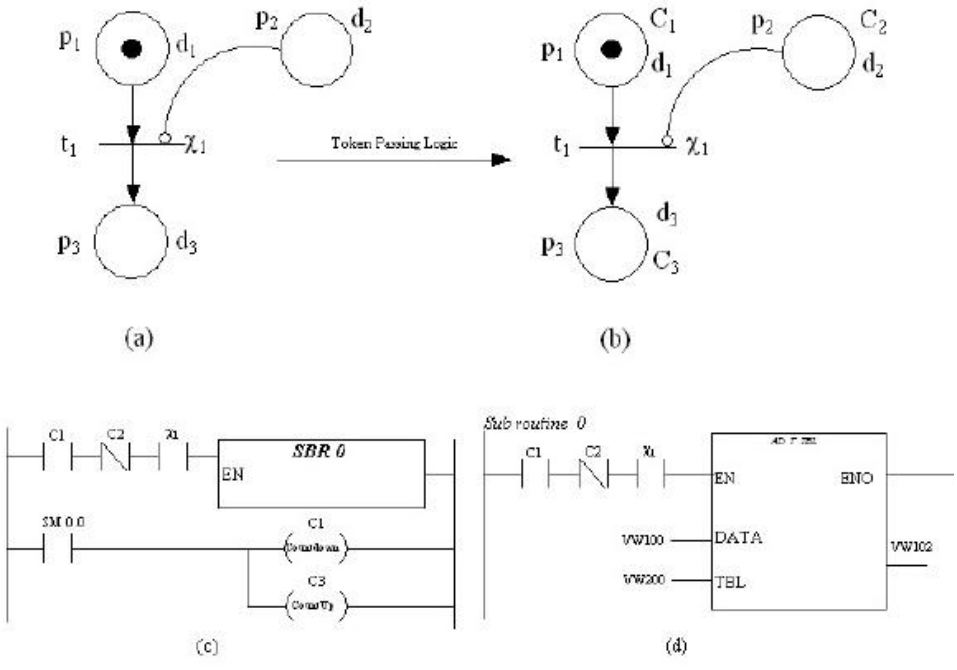

Figure-5. LLD equivalent for inhibitor arc FAPN.

E. Enabling arc FAPN

This concept is illustrated in Figure-6. Here, the places p_1 and p_2 are connected to a transition t_1, the difference being that the p_2 is having a enabling arc En (p_2, t_1) to t_1. The transition is fired if both place p_1 and p_2 have at least one token each and firing condition χ_1 occurs. On firing, one token from p_1 is removed and placed in p_3. The marking of p_2 does not change. If there are more than one output place, then tokens will be deposited at every output place on firing. The presence of a token at p_2 is essential for firing the transition. The TPLC equivalent has counters assigned to places with their strengths. Here transition t_1 fires when χ_1 occurs and counters C_1 and C_2 have at least one token each. When the transitions are fired, the value of C_1 is decremented and C_3 is incremented. The count of C_2 is left undisturbed. The marking of C_3 is decided through the subroutine and the Table is updated. For convenience, the initial markings are not shown.

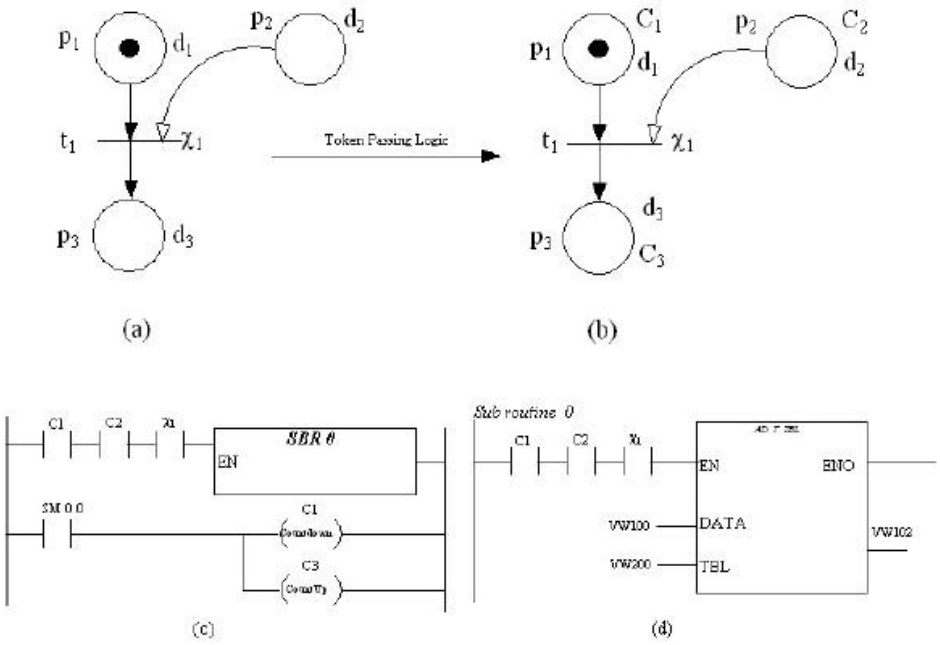

Figure-6. LLD markings for enabler arc transition.

F. AND transition FAPN

Figure-7 shows the AND transition in FAPN. A token is placed in p_3 when the transition t_1 fires. Transition t_1 fires when all the input places have a token (here, p_1 and p_2) and the firing condition χ_1 occurs. Either a counter or a flag can be assigned to represent the places here depending on the number of tokens in a place. Counters are used to handle more than one token. The logic is to remove tokens from input places and deposit in the output place. This is achieved by decrementing the count value at the input places and incrementing the count value at the output place. The marking will be decided on the least value between the two places p_1 and p_2. The resulting LLD is shown in Figure-7.c.

G. OR transition FAPN

There is a semantic problem with the definition of OR logic in Petri nets and therefore with all variants of Petri nets. As far as the LLD is concerned, Logic OR or Exclusive OR is a simple execution made available through a set of parallel constructs. The construct will not be valid in FAPN although the conceptual implementation is possible in TPLC and LLD. Hence, the representation in FAPN is indicative and is taken in by using a slightly convoluted method i.e., using (2n -1) transitions and n (2n-1) inhibitor arcs, where n is the number of input places. The case is illustrated in Figure-8.

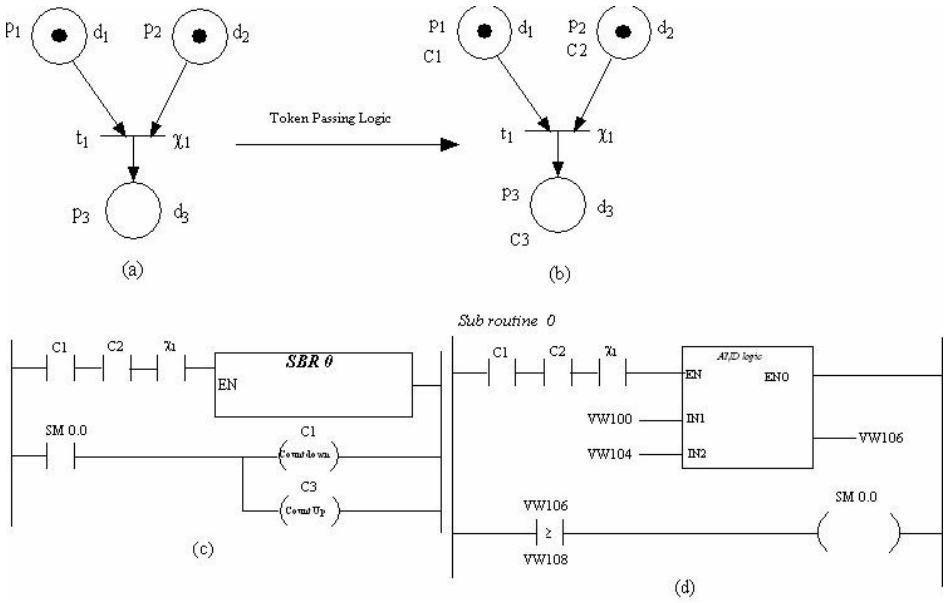

Figure-7. LLD equivalent for and transition FAPN.

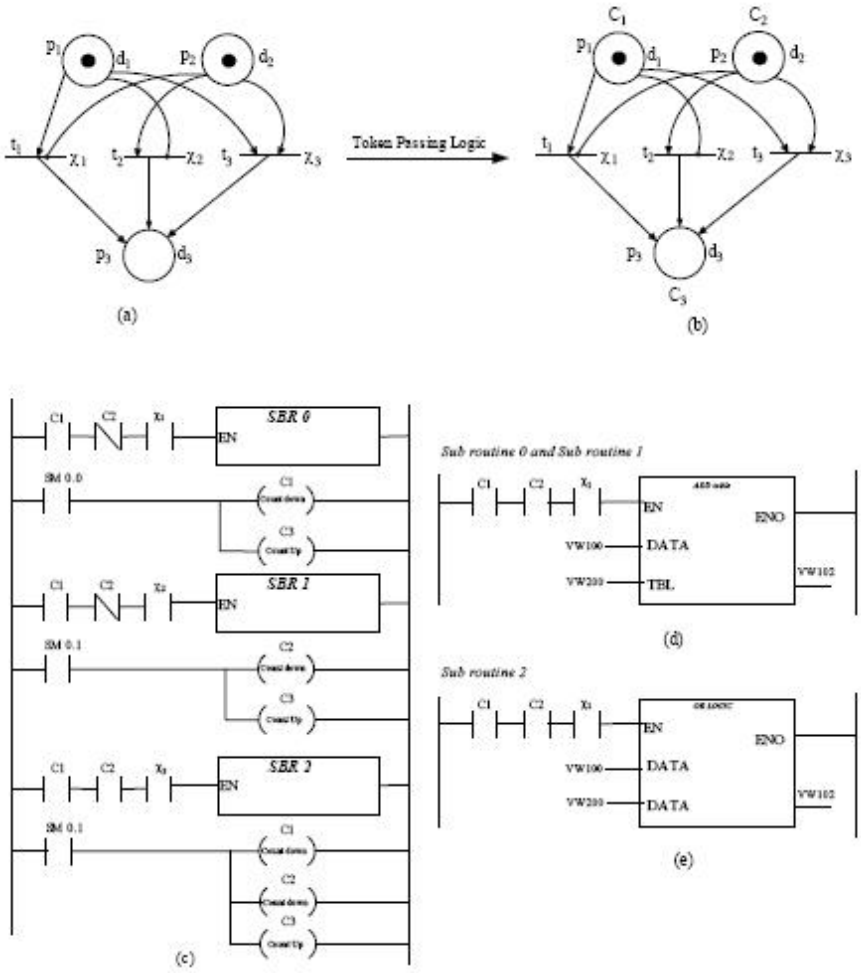

Figure-8. OR construct FAPN through LLD.

H. Weighted arc FAPN

A weighted arc FAPN is a variant visualized from weighted arc Petri nets. The concept of weighted Petri nets is to include more than one token from a place in decision making. The transition t_1 can be fired only when the number of tokens at the input place is greater than n and the firing condition χ_1 occurs. The number n is indicated in the arc from input place to the transition. Upon firing, n number of tokens from the input place will be removed and deposited at the output place. In order to accommodate for the number n a counter is introduced in the input place. When the count in the place equals or greater than n the transition t_1 will be enabled, provided the firing condition χ_1 occurs. When it fires, the value in counter C_1 is decremented n times and the value of counter C_2 is incremented by n times from the present value. The logic is illustrated in Figure-9.

I. Conflict in FAPN

Conflict arises when there is more than one valid location for a single resource. This is visualized in Petri nets as an input place which has at least two output transitions. According to the definition of Petri nets, only one output place can receive a token in the case of conflict. One simple way to resolve the conflict is to assign a priority to each place and solution is taken with respect to priority. In FAPN, this priority can be simply associated with the membership marking di associated with the place. When both transitions t_1 and t_2 leading to the conflict fire in concurrence with the firing conditions χ_1 and χ_2, then the strength of the marking leading to p_2 and p_3 is considered and whichever is higher is lead to the output. This is illustrated in the Figure-10. In case of traditional Petri net variations, the conflict can be resolved by the way in which the ladder logic diagrams are drawn. Since PLC scans from top to bottom, left to right, what is mentioned first as the output place will fire first followed by others. Once the token is exhausted in the input place, the successive rungs will automatically be disabled so that the other places do not fire. The only requirement is that the priority needs to be known clearly before the operation. In the case of FAPN, this is further refined to be understood not only as the place to be chosen but also the strength to be associated with the place. Accordingly, a conflict situation presents multiple solutions as per the requirements of the user. One such solution is given in Figure-10.

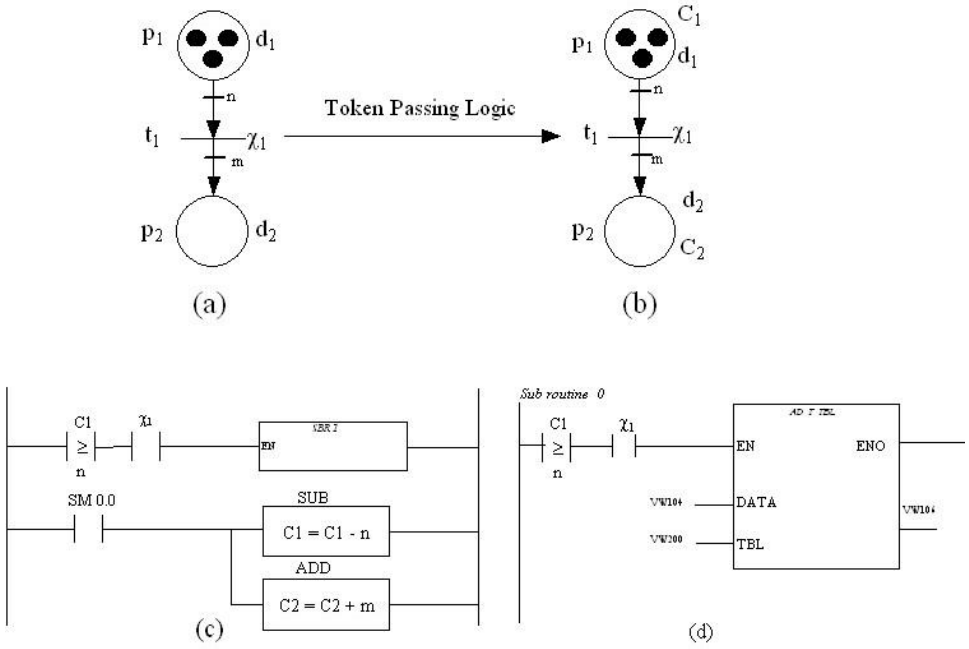

Figure-9. Weighted arc FAPN in LLD.

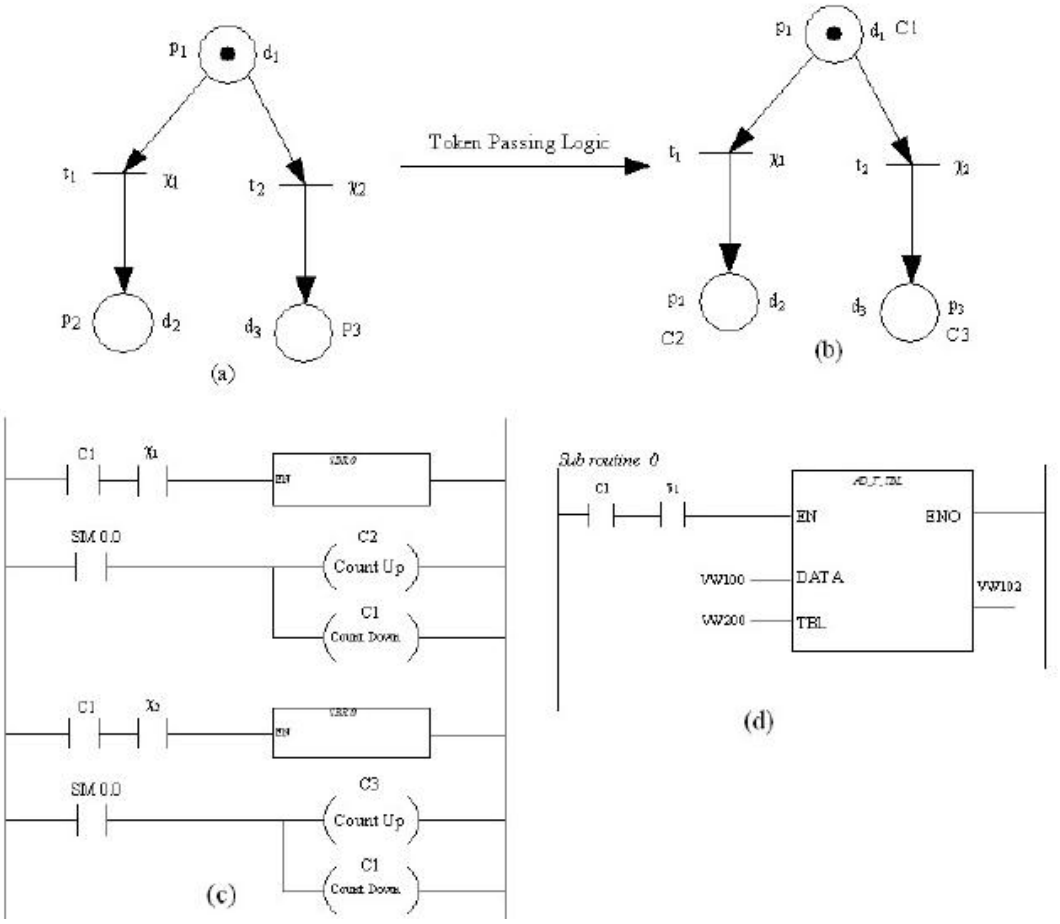

Figure-10. Representation of LLD for Weighed arc FAPN.

J. Timed transition FAPN

In the description of the systems so far, time is associated as a variable only with transitions so far. If presence of a token at a place triggers a time bounded event, then it becomes necessary that the place be associated with timer. Such places may be called timed-place FAPN. The different interpretation of timing sequences used in timers can be extended to such places. For example, once a place is initiated into timed state, the token can have two states: reserved to fire a timed transition t_1 or unreserved. When a timed transition is enabled, it is ready to fire. When the firing condition χ for the transition occurs, the token of the input place to this transition is said to be reserved for a specified amount of time (α_i). When the time αi has elapsed, the transition is effectively fired. The rest of the operation is as per the usual procedure. A timed FAPN is similar to FAPN with action, the difference being that the action happens after a delay α_i. Figure-11 shows a timed transition FAPN, where t_1 is being fired. Whenever t_1 is enabled, an unreserved token is deposited in place p_1. When firing condition χ_1 occurs, the sequence for firing is started and the token in place p_1 gets reserved. After a time lapse of α_i, the transition is effectively fired and the token is deposited to place p_2. To affect this time factor, an ON – delay timer is proposed in the TPLC equivalent and the same is called in the LLD program using timer instructions of PLC. To indicate the movement of tokens across the places, counters with increment or decrement functions are provided.

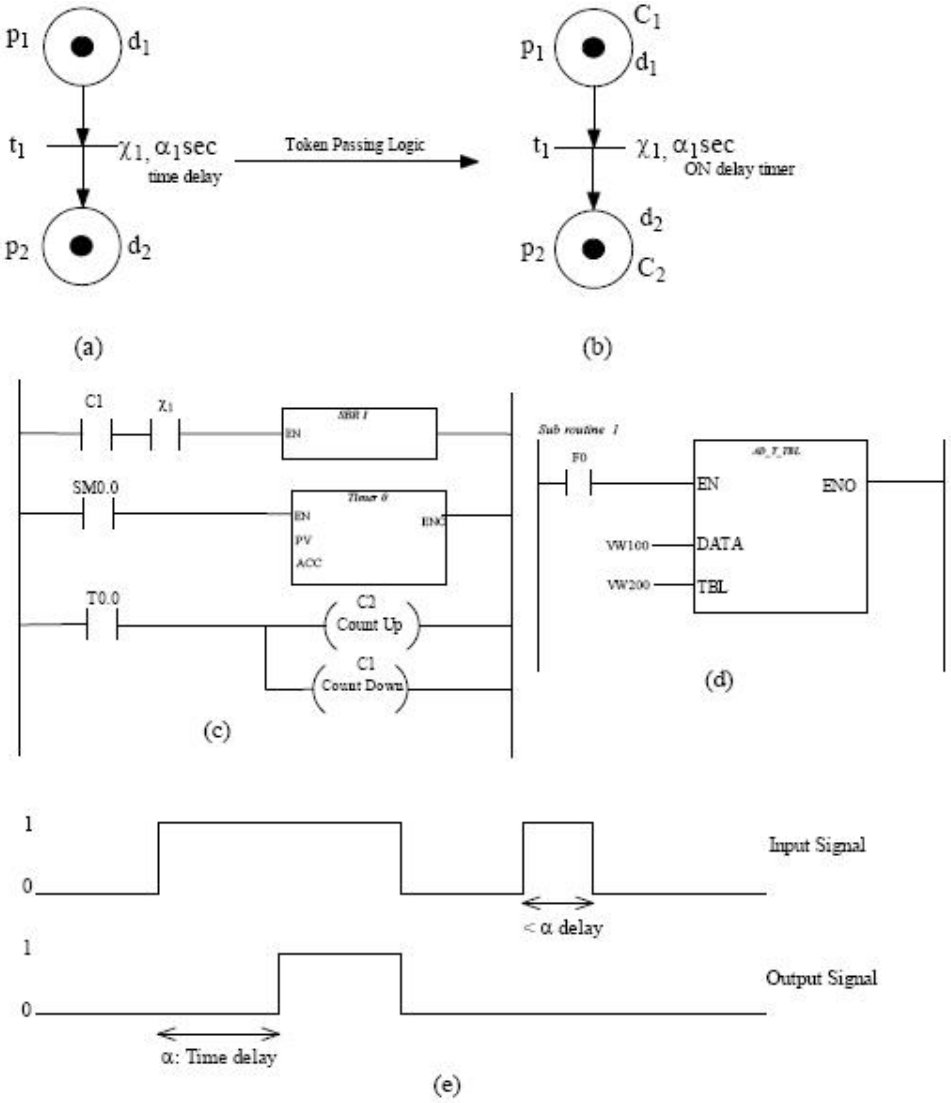

Figure-11. Timed transition FAPN through LLD.

3. CONCLUSIONS

A general methodology for converting Fuzzy Automation Petri net (FAPN) into LLD has been proposed. Since LLD is still widely a preferred programming language for PLC, this is made possible by referring the formalism through Token Passing Logic. The advantage of this routing is the concept is made clear and concise and a direct inference with both the FAPN and LLD is made possible. This mapping between the basic sequencing information and the programming steps is graphically portrayed for clarity. The formalism also adapts to counters, timers and fuzzy markings. Any changes need to be made in the programming language is possible by extension of the same into the semantics. The convenience of this translation also relies on the practical utility that many of the PLC programming software provide a conversion between LLD to any other PLC programming languages although the converse is not true. Hence, the purpose to maximally express the specifications and behaviour of the system into the modeling and control formalism will enable an efficient automation for the industry.

REFERENCES

[1] David R. and Alla H. 1992. Petri Nets and Grafcet: Tools for Modeling Discrete Event Systems. Prentice Hall International (UK) Ltd.

[2] Pollard J.R. 1994. Ladder Logic Remains the PLC Language of Choice. Control Engineering. April. pp. 77-79.

[3] Venkatesh K., Zhou M.C. and Caudill R. 1994. Evaluating the Complexity of Petri Nets and Ladder Logic Diagrams for Sequence Controllers Design in Flexible Automation. Proc. of IEEE Symposium on Emerging Technologies and Factory Automation. pp. 428-435.

[4] Greene J. 1989-1990. Petri Net Design Methodology for Sequential Control Measurement + Control. Vol. 22, December/January 22. pp. 288-291.

[5] Rattigan S. 1992. Using Petri Nets to Develop Programs for PLC Systems Lecture Notes in Computer Science. 616: Application and Theory of Petri Nets. pp. 368-372.

[6] Satoh T. Oshima H., Nose K. and Kumagai S. 1992. Automatic Generation System of Ladder List Program by Petri Net. Proc. of the IEEE Int. Workshop on Emerging Technologies on Factory Automation-Technology For The Intelligent Factory. pp. 128-133.

[7] Jafari M.A. and Boucher T.O. 1994. A Rule-Based System for Generating Ladder Logic Control Program from a High Level System Model. Journal of Intelligent Manufacturing. 5: 103-120.

[8] Burns G.L. and Bidanda B. 1994. The Use of Hierarchical Petri Nets for the Automatic Generation of Ladder Programs. In: Proc. of ESD IPC'94 Conference and Exposition-People Partnerships and Technology, Detroit, Michigan, April 11-14. pp. 169-179.

[9] Taholakian W.M. and Hales M. 1995. The Design and Modeling of PLC Programs Using Petri Nets. Proc. of the Int. Conf. on Planned Maintenance, Reliability and Quality Assurance, Cambridge, England, April 6-7. pp. 194-199.

[10] Jones A.H. and Uzam M. 1996. Towards a Unified Methodology for Converting Coloured Petri Net Controllers into Ladder Logic Using TPLL: Part II-An Application. Proc. of Int. Workshop on Discrete Event Systems, WODES'96, Edinburgh, UK, August 19-21. pp. 314-319.

[11] Uzam M. and Jones A.H. 1996. Conversion of Petri Net Controllers for Manufacturing Systems into Ladder Logic Diagrams. Proc. of the 5th IEEE International Conference on Emerging Technologies and Factory Automation, Hawaii, USA. November 18-21. pp. 649-655.

[12] Jones A.H., Uzam M. and Ajlouni N. 1996. Design of Discrete Event Control Systems for Programmable Logic Controllers Using T-Timed Petri Nets. Proc. of the IEEE International Symposium on Computer-Aided Control System Design-CACSD'96, Michigan, USA. September 15-17. pp. 212-217.

[13] Uzam M. and Jones A.H. 1998. Discrete Event Control System Design Using Automation Petri Nets and Their Ladder Diagram Implementation. Accepted for publication in the special issue of International Journal of Advanced Manufacturing Technology on Petri nets applications in advance manufacturing systems. 16(1): June 1998.

[14] Venkatesh K., Zhou M.C. and Caudill R. 1994. Comparing Ladder Logic diagrams and Petri Nets for Sequence Controller Design through a Discrete Manufacturing System. IEEE Trans. on Industrial Electronics. December. 41(6): 611-619.