Journal of Intelligent Manufacturing, 15, 245–252, 2004

Automatic generation of ladder diagram with control Petri Net

GI BUM LEE[1], HAN ZANDONG[2] and JIN S. LEE[3]

[1]*Research Institute of Industrial Science and Technology, Pohang, 790-600, Korea*
E-mail: gblee@rist.re.kr
[2]*Tsinghua University, Beijing 100084, People's Republic of China*
E-mail: hanzd@mail.tsinghua.edu.cn
[3]*Pohang University of Science and Technology, Pohang, 790-784, Korea*
E-mail: jsoo@postech.ac.kr

Received February 2002 and accepted May 2003

This paper presents a design method to generate a ladder diagram (LD) automatically with the control Petri Net (CPN) for control of discrete event system. This method describes the specification of a practical system with the CPN that associates operations with places and conditions with transitions. Based on the firing regulation of transition, the relationship of places, conditions, and events are formulated with Boolean functions. These functions can be easily converted into LD and implemented on a programmable logic controller (PLC). An application example in a liquid mixture system shows that the proposed method is effective and has the advantages of ease of understanding, modification, and maintenance.

Keywords: Automated system, programmable logic controller, control Petri Net, Boolean equation, ladder diagram

1. Introduction

Competition in today's world market is so stiff that product lifecycles have become shorter and shorter. Factories are pushed to produce small batches of diverse goods. Therefore, these factories are required to develop flexible and agile control systems (Zhou, 1995). At present, most automated industrial systems are controlled by programmable logic controllers (PLCs), and most of the control programs are developed by using a ladder diagram. These diagrams do not capture the underlying sequential, synchronous and concurrent events that drive and perform the process of the control system. So the ladder diagram (LD) program is difficult to modify and reuse (Venkatesh *et al*., 1994).

As a graph theoretic and visually graphical tool, Petri Nets (PN) are widely recognized as appropriate for modeling and analyzing discrete event systems, especially systems characterized by sequence, synchronization, concurrency and parallelism processes (Lewis *et al*., 1998; Murata, 1989). Besides being used as a modeling and analyzing tool, PN is capable of controlling and simulating discrete-event processes (Lee *et al*., 2000; Miyazawa *et al*., 1996). Several PN-based control design methods are presented in Venkatesh *et al*. (1994), Ferrarini (1992), and Zhou *et al*. (1992). Burns *et al*. (1994) have used the concept of transition variables to translate safe PNs into sequential Boolean equations. This modeling provided a basic methodology for translating industrial automation systems into ladder logic diagrams. Uzam *et al*. (1998) have introduced automation Petri Net (APN) to provide a method for design and implementation of discrete event control systems. They reported that the APN model could be converted into LD via token passing logic (TPL).

This paper presents a method for formally transferring a control Petri Net (CPN) into an equivalent LD. The proposed method is system-

atically developed for the CPN. The control process of the real system is represented by the evolution of markings. A logic description with CPN is developed according to the control specification and transferred into Boolean functions. It can be facilely converted into a LD. The proposed method has the advantages of being easy to understand, modify and reuse.

The remainder of the paper is arranged as follows: Definitions and notations of the CPN used in this paper are given in Section 2. In Section 3, an approach that transfers CPN to LD automatically will be described. An illustrative example is given in Section 4 and also in Sections 2 and 3. Conclusions are drawn in the final section.

2. Control Petri Net

A PN can be used to describe the controller of an automated process if it is deterministic, i.e. for every input sequence, its output sequence is determined. Before presenting the definition of CPN, we first give some definitions and notations of the PN that are required in this paper. For a more detailed description, please refer to David (1992) and Peterson (1981).

Definition 2.1
A marked PN is a 5-tuple $\mathrm{PN} = (P, T, I, O, M_0)$ *where:* $P = \{p_1, p_2, \ldots, p_n\}$ *is a finite set of places,* $T = \{t_1, t_2, \ldots, t_m\}$ *is a finite set of transitions, I is the input function, O is the output function,* M_0 *is the initial marking.*

Definition 2.2
A marked PN is said to be safe for an initial marking M_0 *if each place contains at most one token for all reachable markings.*

Definition 2.3
For a marked PN, $^\circ t = \{p | (p,t) \in I\}$ *and* $t^\circ = \{p | (p,t) \in O\}$ *are the input places set and output places set of t.* $^\circ p = \{t | (p,t) \in O\}$ *and* $p^\circ = \{t | (p,t) \in I\}$ *are the input transitions set and output transitions set of p.*

Definition 2.4
$C = \langle p, T_c \rangle$ *is called a conflict structure if it satisfies the following condition:* $T_c = \{t | t \in p^\circ\}$ *and* $|T_c| \geq 2$, *where* $|T_c|$ *is the cardinality of* T_c.

Definition 2.5
For a marked PN, a transition t is enabled under a given marking M if the following condition is satisfied: $M(p) \geq I(p,t)$, *for any* $p \in {}^\circ t$. *Firing the enabled transition t under M results in a new marking* M': $M'(p) = M(p) + O(p,t) - I(p,t)$, *for any p.*

Definition 2.6
A PN is said to be a CPN if it satisfies the following conditions:

(1) It is safe and deterministic.
(2) It is T-timed and synchronized.
(3) It has an operation part and a condition part. The operation part is associated with places. It includes outputs and actions. The condition part is associated with transitions.

Definition 2.7
A safe PN is deterministic if it satisfies the following conditions:

(1) For every conflict structure $\langle p, \{t_1, t_2, \ldots\} \rangle$, *the conditions associated with* $t_1, t_2, \ldots$, *must be mutually exclusive.*
(2) For every pair of places p_i *and* p_j, *such that their operations are incompatible, we have* $M(p_i) + M(p_j) \leq 1$.

The marking of places represents one of the system states, and the firing of transitions represents the changing between states. The output corresponds to a signal that acts on the environment of the system, represented as a Boolean variable. The output is active when its corresponding place is marked. When the place is unmarked, the output becomes inactive. The action is not connected with the environment but changes internal conditions and controls the process evolutions. The actions are carried out only once when the corresponding place changes from unmarked to marked. The action is represented as an operation with square brackets.

The condition is a Boolean function (Katz, 1994) of external and internal variables. The change of conditions may result in the firing of transitions, and consequently cause an operation.

The above characteristics fit for the control demands of a practical control system. So the CPN describes a controller. For every input sequence, its output sequence is determined. The following example is given to illustrate how to describe a control system with CPN.

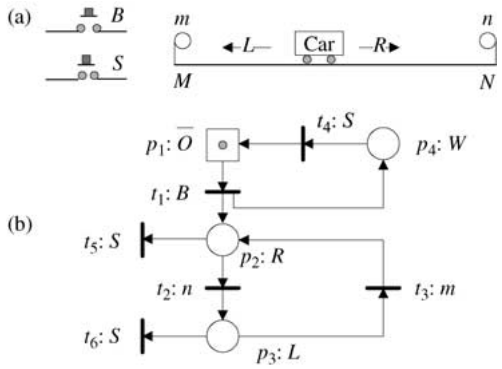

Fig. 1. (a) Function description of system. (b) Logic description with CPN.

Example 2.1
Let us consider a transport system in Fig. 1(a). At the initial state, the car is on an arbitrary place on the track. When the begin button B is pressed, the car moves to the right and returns to the left when it arrives at N, when the car goes left, it returns to the right immediately. The car repetitively moves between M and N. When the stop button S is pressed, the car stops where it is.

In this system, there are two states: initial (car is stopped) and working (car is moving). The working state includes two sub-states: left movement and right movement. There are two synchronized events: pressing the B button and pressing the S button (marked as B and S respectively). There are external conditions: arriving at the left-hand side and arriving at the right-hand side (marked as Boolean variable m and n respectively). Since the initial place at the beginning has a token, we assign $\overline{O}$ in the initial place. $\overline{O}$ is the complement of O. Here, the initial place p_1 is depicted with a square containing a token.

The logic description with CPN is shown in Fig. 1(b). The meaning of each place is explained in Table 1.

There is a token in place $p_1 : \overline{O}$ that represents the initial state. When B is pushed, t_1 fires and the token moves to $p_4 : W$ and $p_2 : R$, which means that the system is in a working state and the car moves right. When the car arrives at the right-hand side, n becomes true, therefore t_2 fires and the token leaves from p_2 and resides in p_3, which means that the car stops moving right and begins to move left. After arriving at the left-hand side, t_3 fires and the car moves to the right hand side again. When S is pushed, p_1 obtains a token and all other places lose their token, and the system returns to its initial state.

Table 1. Place meaning in Fig. 1

Place	*Parameter*	*Meaning*
p_1	O	System is in initial state
p_2	R	Car moves to right
p_3	L	Car moves to left
p_4	W	System is in working state

3. Transfer of CPN to LD

This section will describe the method for formally transferring a CPN into an equivalent LD. This method begins with the analysis of PN markings evolution.

The markings of CPN reflect a working stage of a controlled system. If the CPN markings remain constant, the working state will remain unchangeable. Therefore, the evolution of markings can be used to reflect the control process of a real system.

The firing of a transition may change the marking, so we first analyze the firing condition of a transition. We know, for a transition t_j in a safe PN, it is enabled if all its input places are marked. For a synchronized PN, an enabled transition fires right away if its corresponding condition is true and its associated event occurs. We can represent the idea with an expression.

First, we define some Boolean variables as follows:

$$P_i(\tau) = \begin{cases} 1 & \text{if } p_i \text{ is marked at time } \tau \\ 0 & \text{if } p_i \text{ is unmarked at time } \tau \end{cases}$$

$$T_j(\tau) = \begin{cases} 1 & \text{if } t_j \text{ is fired at time } \tau \\ 0 & \text{if } t_j \text{ is not fired at time } \tau \end{cases}$$

$$C_j(\tau) = \begin{cases} 1 & \text{if } C_j \text{ is true at time } \tau \\ 0 & \text{otherwise} \end{cases}$$

$$E_j(\tau) = \begin{cases} 1 & \text{if } E_j \text{ occurs at time } \tau \\ 0 & \text{otherwise} \end{cases}$$

C_j and E_j are the conditions and events associated with t_j. According to the enabling condition of the transition in PN, $T_j(\tau)$ can be algebraically defined by $P_i(\tau)$, $C_j(\tau)$ and $E_j(\tau)$ as follows:

$$T_j(\tau) = \prod_{\{p_i | p_i \in {}^\circ t_j\}} P_i(\tau) \cdot C_j(\tau) \cdot E_j(\tau) \qquad (1)$$

After the enabled transitions fire, the Boolean function associated with p_i, at time $\tau + \Delta\tau$ can be calculated as follows:

$$P_i(\tau + \Delta\tau) = (P_i(\tau) + \sum_{\{t_j | t_j \in {}^\circ p_i\}} T_j(\tau)) \cdot \prod_{\{t_j | t_j \in p_i^\circ\}} \overline{T_j(\tau)} \qquad (2)$$

Where $\Delta\tau$ is the scan time of PLC, Σ implies the ORing of terms and Π stands for the ANDing of terms.

When Equation 1 is substituted into Equation 2 and the index is made harmoniously, we get the following equation:

$$\begin{aligned} P_i(\tau + \Delta\tau) &= (P_i(\tau) + \sum_{\{t_j | t_j \in {}^\circ p_i\}} \left(\prod_{\{p_k | p_k \in {}^\circ t_j\}} P_k(\tau) \cdot C_j(\tau) \cdot E_j(\tau) \right)) \\ &\cdot \prod_{\{t_j | t_j \in p_i^\circ\}} \left(\overline{\prod_{\{p_k | p_k \in {}^\circ t_j\}} P_k(\tau) \cdot C_j(\tau) \cdot E_j(\tau)} \right) \end{aligned} \qquad (3)$$

According to Equation 3, we can calculate the marking of each place step by step. This process is similar to the working fundamentals of PLC. Because the marking states present the working stage of the controlled system, we can easily transfer Equation 3 into LD by the following operations:

(1) Transfer the left part of Equation 3 into output in LD.
(2) Transfer all elements of the right parts of Equation 3 into input in LD.

Because the time parts such as τ and $\tau + \Delta\tau$ do not appear in the LD, suppressing it in Equation 3 will not result in confusion when Equation 3 is transferred to LD. We can simplify Equation 3 as follows:

$$\begin{aligned} P_i &= \left(P_i + \sum_{\{t_j | t_j \in {}^\circ p_i\}} \left(\prod_{\{p_k | p_k \in {}^\circ t_j\}} P_k \cdot C_j \cdot E_j \right) \right) \\ &\cdot \prod_{\{t_j | t_j \in p_i^\circ\}} \left(\overline{\prod_{\{p_k | p_k \in {}^\circ t_j\}} P_k \cdot C_j \cdot E_j} \right) \end{aligned} \qquad (4)$$

Let us use the proposed approach to transfer the PN in Fig. 1 to LD shown in Fig. 3. In order to make the variable containing a physical meaning, we replace the place with its corresponding operation. For example, variable P_2 is replaced by R.

$$\overline{O} = (\overline{O} + W \cdot S) \cdot \overline{\overline{O} \cdot B} \quad \Rightarrow \quad O = O \cdot (\overline{W} + \overline{S}) + \overline{O} \cdot B$$

$$R = (R + \overline{O} \cdot B + L \cdot m) \cdot \overline{R \cdot n} \cdot \overline{R \cdot S}$$

$$L = (L + R \cdot n) \cdot \overline{L \cdot m} \cdot \overline{L \cdot S}$$

$$W = (W + \overline{O} \cdot B) \cdot \overline{W \cdot S}$$

Fig. 2. Boolean output functions.

According to Equation 4, the Boolean function of output can be represented with input as in Fig. 2.

From Fig. 2, we can easily get the LD program in Fig. 3. The generated LD program can be implemented in the PLC. However, each rung must be computed with the present values of input and output respectively, and the calculated values of output must be simultaneously updated to the output unit of the PLC.

Equation 3 is useful to develop LD for many industrial processes, but it does not include timers and counters that are very important elements for industrial processes. Thus, we have to consider how to model them by CPN. An example will be given in the following paragraph.

Example 2.2
Let us consider a similar transport system of Fig. 1(a). At the initial state, the car is on arbitrary place on the

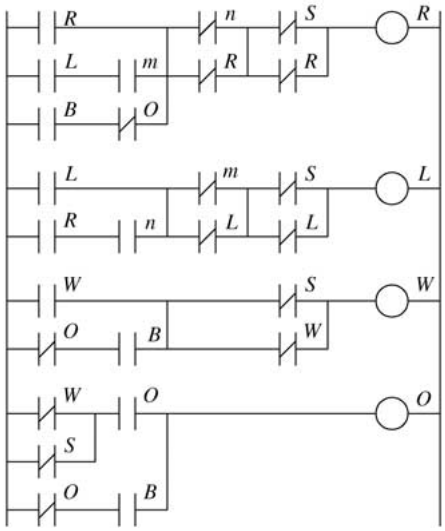

Fig. 3. LD program generated from Fig. 1.

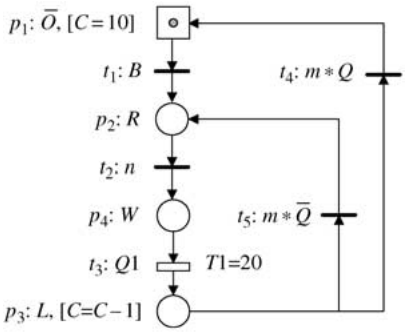

Fig. 4. CPN logic description.

track. When the begin button B is pressed, the car moves to the right. When the car arrives at the right hand side, it stays at position N for 20 s in order to load and unload the passengers, then moves to the left hand side. After the car repeatedly arrives at the left hand side 10 times, it stops at the left hand side until the begin button B is pressed again.

This system can be modeled by the CPN logic description of Fig. 4. Here, timed transition is represented by a box and delay time $T1$ is 20 s.

Besides the expressions of R, L, $\overline{O}$, W we should give the expressions of Q, $Q1$ associated with C, $T1$. The Boolean output functions are represented with inputs as in Fig. 5.

Where, CTD is the down counter and Reset: $\overline{O}$ is the reset input of the counter and In: L is the count input of the counter. The initial counter value is $C = 10$. When $C = C - 1$ becomes zero, the output contact Q goes ON, otherwise Q goes OFF. The CTD can be used to signal when a count has reached zero, on counting down from a present value. The CTD counts down the number of ''rising edges'' detected at input L. When the counter reaches zero, the Q output is set true and the counting stops.

$$\overline{O} = (\overline{O} + L \cdot m \cdot Q) \cdot \overline{\overline{O} \cdot B} \quad \Rightarrow \quad O = O \cdot (\overline{L} + \overline{m} + \overline{Q}) + \overline{O} \cdot B$$

$$R = (R + \overline{O} \cdot B + L \cdot m \cdot \overline{Q}) \cdot \overline{R \cdot n}$$

$$L = (L + W \cdot Q1) \cdot \overline{L \cdot m \cdot \overline{Q}} \cdot \overline{L \cdot m \cdot Q} = (L + W \cdot Q1) \cdot \overline{L \cdot m}$$

$$W = (W + R \cdot n) \cdot \overline{W \cdot Q1}$$

Q = CTD (Initial value: C=10; In: L; Reset: $\overline{O}$)

$Q1$ = TON (Initial value: $T1$=20 s; In: W)

Fig. 5. Boolean functions of output.

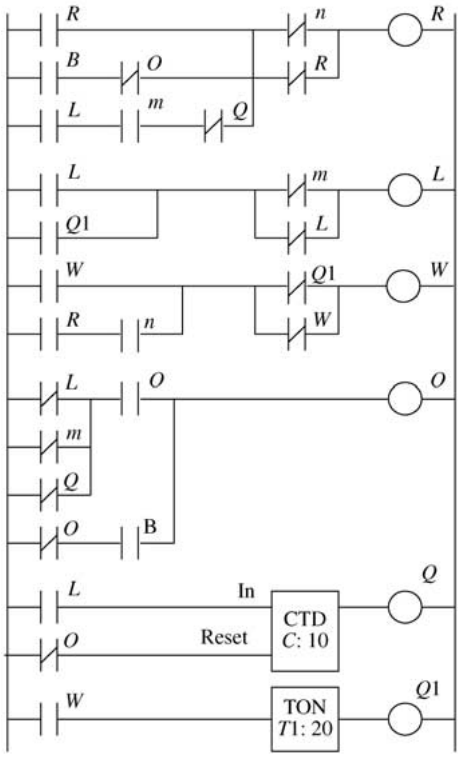

Fig. 6. LD generated from Fig. 4.

TON is the on-delay timer with the delay time ($T1 = 20$ s) and In: W describes the input of TON. The timer can be used to delay setting an output true, for a fixed period after an input becomes true. Since an input W becomes true, the elapsed time starts to increase. When the elapsed time reaches the delay time (20 s), the output $Q1$ becomes true and the elapsed time is held. The output $Q1$ remains true until the input W turns false. If the input is not true longer than the delay time (20 s), the output remains false.

According to the expressions, we can easily obtain the LD in Fig. 6.

When a practical system is modeled by CPN, still other important and useful tools to consider are inhibitor arcs and enabling arcs. An inhibitor arc is marked at the end by a small circle. The enabling arc is denoted by an empty arrow (Uzam *et al.*, 1998). In Fig. 7(a), the arc (p_1, t_1) is an inhibitor arc, (p_5, t_2) is an enabling arc. The inhibitor arc between p_i and t_j means that transition t_j is only enabled if place p_i is unmarked and the firing of t_j does not take away a token from p_i. The enabling arc between p_i and t_j means that transition t_j is enabled if all of its input

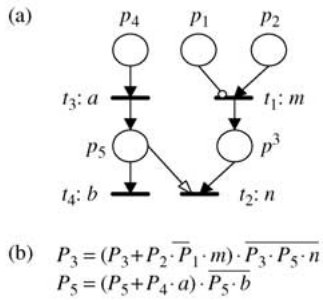

Fig. 7. Example of inhibitor arc and enabling arc.

places including p_i are marked and the firing of t_j does not remove the token from p_i. In the case of inclusion of inhibitor arcs and enabling arcs, Equation 4 is modified as follows:

$$P_i = \left(P_i + \sum_{\{t_j | t_j \in {}^\circ p_i\}} \left(\prod_{\{p_k | (p_k, t_j) \in A_N \cup A_E\}} P_k \cdot \prod_{\{p_k | (p_k, t_j) \in A_I\}} \overline{P_k} \cdot C_j \cdot E_j \right) \right) \cdot \prod_{\{t_j | (p_i, t_j) \in A_N\}} \cdot \left(\overline{\prod_{\{p_k | (p_k, t_j) \in A_N \cup A_E\}} P_k \cdot \prod_{\{p_k | (p_k, t_j) \in A_I\}} \overline{P_k} \cdot C_j \cdot E_j} \right) \tag{5}$$

Where A_I, A_E and A_N represent the set of inhibitor arcs, enabling arcs, and normal arcs, respectively.

By Equation 5, we can obtain the Boolean functions of outputs (P_3, P_5) considering an inhibitor arc and an enabling arc as shown in Fig. 7(b).

4. Application: liquid mixture system

An example will be given to illustrate how to use the proposed method.

Example 4.1
Let us consider a liquid mixture system in Fig. 8(a). There are two buttons: B and S. The operations are described as follows:

(1) Initial state: V1, V2, V3, and V4 valves are closed, and M is OFF.

(2) The start button B is pushed, then goes to the next stage 3.

(3) V1 opens first, when the liquid level arrives at L, V2 opens. After the liquid level arrives at H, both V1 and V2 are closed, and M begins to stir for 120 s. Then the process is repeated (open V4 50 s, close 10 s) 10 times, viz. by filling 10 bottles with materials. If the stop button S is pushed, then go to stage 4. Otherwise, go to stage 3.

(4) After the process (open V4 for 50 s, close it for 10 s) repeats 10 times, V3 opens until H level, and M begins to stir for 180 s. Then V4 opens for 600 s, and goes to stage 1.

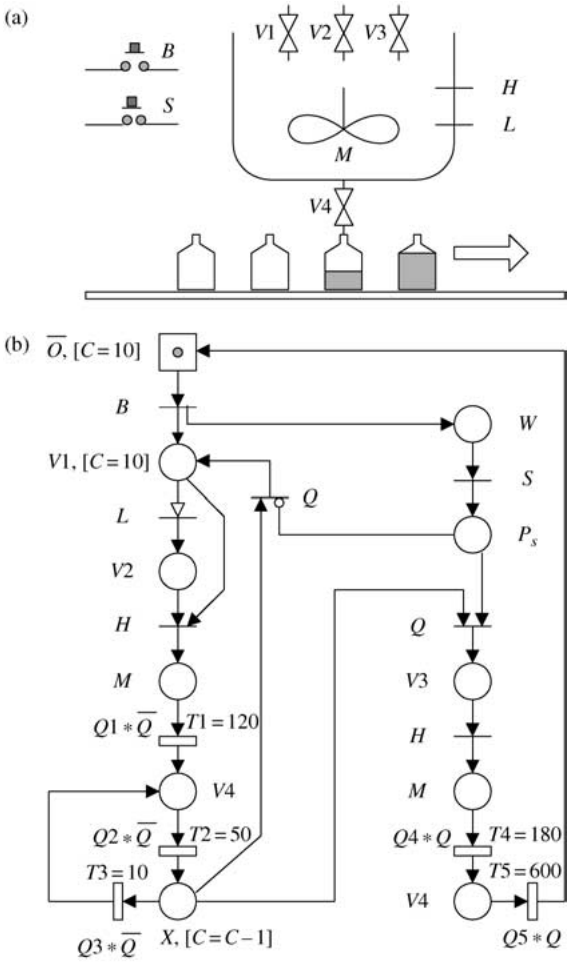

Fig. 8. (a) Function description of system. (b) Logic description with CPN.

According to the stages 1–4, we present the logic description with the CPN in Fig. 8(b). In order to make the CPN diagram clear, we suppress the place name, i.e., p_1, p_2, etc. However, to describe the flow process clearly, we separate stage 3 (process *V*1–*V*2–*M*–*V*4–*X*) and stage 4 (process *V*3–*M*–*V*4) in Fig. 8(b), i.e., operation *M* is assigned to two different places and operation *V*4 is also assigned to two

$$\overline{O} = (\overline{O} + V4 \cdot Q5 \cdot Q) \cdot \overline{\overline{O} \cdot B} \quad \Rightarrow \quad O = O \cdot (\overline{V4} + \overline{Q5} + \overline{Q}) + \overline{O} \cdot B$$

$$V1 = (V1 + \overline{O} \cdot B + X \cdot \overline{Ps} \cdot Q) \cdot \overline{V1 \cdot V2 \cdot H}$$

$$V2 = (V2 + V1 \cdot L) \cdot \overline{V1 \cdot V2 \cdot H}$$

$$V3 = (V3 + X \cdot Ps \cdot Q) \cdot \overline{V3 \cdot H}$$

$$V4 = (V4 + M \cdot Q1 \cdot \overline{Q} + X \cdot Q3 \cdot \overline{Q} + M \cdot Q4 \cdot Q) \cdot \overline{V4 \cdot Q2 \cdot \overline{Q}} \cdot \overline{V4 \cdot Q5 \cdot Q}$$

$$M = (M + V1 \cdot V2 \cdot H + V3 \cdot H) \cdot \overline{M \cdot Q1 \cdot \overline{Q}} \cdot \overline{M \cdot Q4 \cdot Q}$$

$$X = (X + V4 \cdot Q2 \cdot \overline{Q}) \cdot \overline{X \cdot Q3 \cdot \overline{Q}} \cdot \overline{X \cdot \overline{Ps} \cdot Q} \cdot \overline{X \cdot Ps \cdot Q} = (X + V4 \cdot Q2 \cdot \overline{Q}) \cdot \overline{X \cdot Q3 \cdot \overline{Q}} \cdot \overline{X \cdot Q}$$

$$W = (W + \overline{O} \cdot B) \cdot \overline{W \cdot S}$$

$$Ps = (Ps + W \cdot \overline{S}) \cdot \overline{Ps \cdot X \cdot Q}$$

Q = CTD (Initial value: $C = 10$; In: X; Reset: $\overline{O}$ or $V1$)

$Q1$ = TON (Initial value: $T1 = 120$ s; In: M)

$Q2$ = TON (Initial value: $T2 = 50$ s; In: $V4$)

$Q3$ = TON (Initial value: $T3 = 10$ s; In: X)

$Q4$ = TON (Initial value: $T4 = 180$ s; In: M)

$Q5$ = TON (Initial value: $T5 = 600$ s; In: $V4$)

Fig. 9. Boolean output functions.

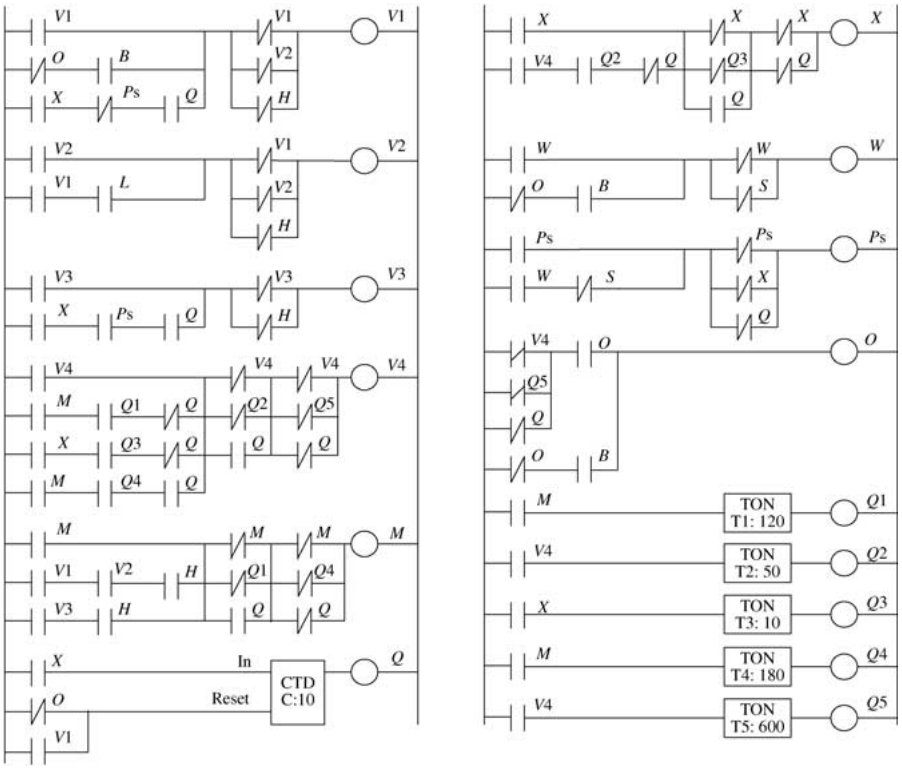

Fig. 10. LD program generated from Fig. 8.

different places. But when processing to make Boolean functions, the separated places are merged in an operation (M or $V4$), i.e., M, that is assigned to two places, is merged into one place and $V4$ is also merged into one place. The timed transitions ($T1$, $T2$, etc.) are used for the time delay. In order to fill 10 bottles with materials every time, the down counter with an initial value of 10 is utilized for the CPN modeling.

From Fig. 8, the Boolean output functions based on the places are represented as shown in Fig. 9.

From Fig. 9, we can easily obtain the LD program of Fig. 10.

5. Conclusions

This paper has presented a practical method to develop an automated system with the CPN. The CPN can give a clear specification for discrete event systems with sequence, synchronization, concurrency, and parallelism processes. The developed PN is easy to understand for electronic engineers and software engineers. A methodology for translating the developed CPN into Boolean functions has been proposed as an approach for systematically generating LD. The proposed method has been proven to be effective through application in a liquid mixture system. It can also be used for more complex industry systems.

References

Burns, G. L. and Bidanda, B. (1994) The use of hierarchical Petri nets for the automatic generation of ladder logic programs, *Proceedings of ESD IPC Conference and Exposition*, Detroit, Michigan, pp. 169–179.

David, R. and Alla, H. (1992) *Petri Net and Grafcet [M]*, Prentice-Hall International (UK) Ltd, London.

Ferrarini, L. (1992) An incremental approach to logic controller design with Petri nets. *IEEE Transactions on Systems, Man, Cybernetics*, **22**(2), 461–473.

Katz, R. H. (1994) *Contemporary Logic Design*, The Benjamin/Cummings Publishing Company, Inc.

Lee, G. B. and Lee, J. S. (2000) The state equation of Petri Net for the LD program. *IEEE International Conference on Systems, Man and Cybernetics*, **4**, 3051–3056.

Lewis, F. L., Gurel, A., Bogdan, S., Doganalp, A. and Pastravanu, O. C (1998) Analysis of deadlock and circular waits using a matrix model for flexible manufacturing systems. *Automatica*, **34**(9), 1083–1100.

Miyazawa, I., Tanaka H. and Sekiguchi, T. (1996) Classification of solutions of matrix equation related to parallel structure of a Petri Net, *IEEE Conference on Emerging Technologies and Factory Automation*, pp. 446–452.

Murata, T. (1989) Petri Nets: Properties, analysis and applications. *Proceedings of the IEEE*, **77**(4), 541–580.

Peterson. (1981) *Petri Net Theory and the Modeling of Systems*, Prentice-Hall International (UK) Ltd, London.

Uzam, M. and Jones, A. H. (1998) Discrete event control system design using automation Petri Nets and their ladder diagram implementation, *International Journal of Advanced Manufacturing Technology*, 716–728.

Venkatesh, K., Zhou, M., Caudill, R. J. (1994) Comparing ladder logic diagrams and Petri Nets for sequence controller design through a discrete manufacturing system. *IEEE Transactions on Industrial Electronics*, **41**, 611–619.

Zhou, M. (1995) *Petri Nets in Flexible and Agile Automation [M]*, Kluwer Academic Publishers Group, Boston.

Zhou, M., DiCesare, F. and Rudolph, D. (1992) Design and implementation of a Petri net based supervisor for a flexible manufacturing system. *Automatica*, **28**(6), 1999–2008.