

Received 8 March 2023, accepted 14 March 2023, date of publication 22 March 2023, date of current version 28 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3259900

## RESEARCH ARTICLE

# A Novel Method for Converting Colored Petri Nets to Ladder Diagram in the Automation of Automated Manufacturing Systems

HUSAM KAID<sup>1</sup>, ABDULRAHMAN AL-AHMARI<sup>1,2,3</sup>, KHALED N. ALQAHTANI<sup>1</sup>,  
FAHAD ALASIM<sup>2</sup>, EMAD H. ABUALSAUD<sup>1</sup>, ABDULMAJEED DABWAN<sup>1</sup>,  
AND MUSTAFA M. NASR<sup>1</sup>

<sup>1</sup>Industrial Engineering Department, College of Engineering, Taibah University, Medina 41411, Saudi Arabia

<sup>2</sup>Industrial Engineering Department, College of Engineering, King Saud University, Riyadh 11421, Saudi Arabia

<sup>3</sup>Raytheon Chair for Systems Engineering (RCSE Chair), Advanced Manufacturing Institute, King Saud University, Riyadh 11421, Saudi Arabia

Corresponding authors: Husam Kaid (hkaid@taibahu.edu.sa) and Abdulrahman Al-Ahmari (alahmari@ksu.edu.sa)

This work was supported by the Raytheon Chair for Systems Engineering.

**ABSTRACT** The Ladder Diagram (LD) is an industry-standard programming language for constructing automated manufacturing systems (AMSs) control algorithms. Petri net (PN) theory, on the other hand, is a mathematical and graphical modeling tool for AMSs. Multiple types of PN-based LDs are designed for AMSs with highly complicated LD structures. Thus, it is necessary to propose a technique that can assist in the minimization of the structural complexity of LDs. The main purpose of this study is to propose a methodology for the implementation of LDs in AMSs. First, a colored resource-oriented Petri net (CROPN) is developed for modeling and guaranteeing the deadlock-free behavior of AMS. Second, a ladder diagram CROPN (LDCROPN) is constructed in order to transform the CROPN into an LD. The proposed LDCROPN is assessed using instances from the literature. The results show that the LDCROPN is effective, has a simpler structure, and has less computational overhead than existing techniques.

**INDEX TERMS** Petri net, programmable logic controllers, industrial automation, model, ladder diagram.

## I. INTRODUCTION

Automated manufacturing systems (AMSs) are defined as a series of discrete states and a state evolution based on the events that characterize the system state [1], [2]. AMSs require a control program using mathematical and graphical tools with a variety of characteristics, including efficiency, accuracy, and adaptability [3], [4], [5]. Industrial automation typically use programmable logic controllers (PLCs). PLCs are applied in systems that require the execution of logical sequencing processes. The growing complexity of PLC technology and user-imposed operational and security requirements require the development of novel control mechanisms. Petri nets are a mathematical and graphical modeling tool for industrial systems [1], [4], [6], [7], [8], [9], [10], [11],

The associate editor coordinating the review of this manuscript and approving it for publication was Mouloud Denai<sup>1</sup>.

[12], [13], [14], [15], [16], [17], [18], [19], [20]. Petri nets have several advantages over other programming tools, such as simple schematic diagram, hierarchical algorithms, and simple algebraic expression [21], [22]. In addition, they allow for the implementation of synchronization, concurrency, and causality [23]. When a PN-based controlled system is developed, it must be translated into a PLC for its execution.

In the execution of automation activities, PLCs have become indispensable. The International Electrotechnical Commission has, according to IEC 61131-3, defined five PLC languages: the functional block chart [24], the ladder diagram [25], the Grafset or sequential function chart [26], the structured text [27], and the instruction list [28]. LD is the best language for PLCs since it can be executed heuristically. Using a ladder diagram, it is simple and clear to build PLC programs for simple systems. Moreover, the problem is complicated by the consideration of multiple systems. In fact,

it is difficult to construct a ladder diagram if the system is unique in class and multiproduct in heuristic method [29], [30]. Petri nets and the heuristic conversion of a Petri net to a ladder diagram are two of the most practical approaches to this problem.

Modeling of conversion techniques [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45] have been performed by applying several techniques based on PNs in ladder diagram. Jones et al. [31], [32] propose a method for the translation of PN to LD via token pass ladder logic (TPLL). In this approach, the token is the key control mechanism in PNs. Using flags or an auxiliary relay, the program is controlled. By applying delay timers in various settings, the method is extended to include timed-transition PNs (TTPNs). The TTPNs' primary ladder logic construction equivalents are presented. Moreover, the significant increase in the number of PNs' fundamental elements has considered their implementation challenging.

Uzam et al. [33], [34], [35], [36] propose both the methodology and application of the TPLL approach for the transformation of colored Petri net supervisors to ladder diagram. This approach can also enable a direct translation between sequencing information and programming procedures. The main features and ladder diagram equivalents of TTPN are presented. A structured LD methodology and a heuristic PN controller are formal sensor and actuator structures that are introduced into the proposed technique. The authors have developed a number of new characteristics for ordinary Petri nets (OPN), allowing the inclusion of sensor readings during transitions and the control of input and actuator levels, resulting in an automated Petri net (APN). The research presented in [37] introduces a bottom-up synthesis method, including the construction of the system's automated Petri net reachability graph. The forbidden state restrictions are associated with a system controller developed using token pass marking (TPM) rules and an uncontrolled automated Petri net model. The token pass marking rules are established immediately from forbidden state restrictions. Moreover, the token pass marking rules are applied using a combination of inhibitor and enabling arcs, which connect the proper places to the transitions of the respective supervisors.

In the work [39], logical control structures are developed and implemented to allow a synthesis of the characteristics of current techniques for developing PLC code and to prevent disadvantages, including the absence of necessary validation. They introduce ordinary colored Petri nets as a modeling tool for logic controllers in combination with other extensions. The logical control system model of the ordered colored Petri net includes two primary constructions: a system model coordination and a submodel for certain aspects of the coordination system model, which is achieved through transitional refinement. A structural analysis of the net is then performed to validate both the model and the logic structure control requirements. The authors next present an approach for automatically generating a PLC code from a validated ordinary

colored Petri net textual description. Park et al. [40] propose a PN-based approach for the construction of control logic for operations machining. The control logic is represented by two Petri nets: one for mode selection and the other for sequence. Each process's PN control logic can be generated by substituting the input and output for the associated actuators, sensors, and operator instructions in the operation module designs. Frey [41] proposes the signal interpreted PN (SIPN) and provides a mechanism for translation of the proposed SIPN to a ladder diagram that provides input and output signals for each event and transition for PLCs. Moreover, a Boolean function value is allocated to each place, which is TRUE when the place is marked and FALSE otherwise. The work of [46] presented a procedure for implementing an instance of a "hierarchical Petri net" in LD using Siemens' TIA Portal software.

Wu and Zhou have been studying deadlock avoidance employing ROPNs to characterize the dynamic resource competition among components of an AMS [3], [4], [5], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56]. In ROPN, there is a "one-to-one mapping" between the resources and the places, resulting in a finite capacity colored Petri net model. In the AMS, colors are utilized to distinguish between different parts and their processing status [47], [49], [50], [51], [52], [53], [55], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67]. Tokens are designated with ROPN colors, with the token's color allowed to change as it moves between places. In particular, a deadlock can only exist within a circuit in ROPN, and the set of circuits is finite. Wu [48] develops colored ROPNs (CROPNs) as well as the "necessary and sufficient conditions" for achieving a "maximally permissive control law" that guarantees deadlock-free operations. The CROPN's control law guarantees that its interactive subnets are deadlock-free and the overall CROPN can be demonstrated to be live.

It is well known that different Petri net-based ladder diagrams are designed for AMSs. However, these methods have a number of drawbacks, including the fact that their complexity increases as a system's size increases and the complications that come with completely modifying the model. Therefore, a method for reducing the complexity of ladder diagrams should be developed. In addition, some complicated structures and blocks in Petri nets and their LD equivalents should be considered, such as "auto-loop," "interlocking," "precedence," "conflict state," "on-delay timer," "off-delay timer," "up-counter," and "down-counter." The main motivation for modeling and converting these complex structures and blocks to LD is that they were developed primarily based on the experience of programmers in industrial automation. Therefore, it is crucial to develop methods that assist in ensuring the effectiveness of conversion algorithms implemented in industrial processes or machines. This research is an extension of [3] and [5] and presents a novel approach to the development of ladder diagrams for AMSs. First, a CROPN modeling technique and control laws are

introduced for modeling and ensuring the prevention of deadlock in the operations of an AMS. Second, a ladder diagram of CROPNs (LDCROPNs) is proposed to consider all of the aforementioned challenges and transform the CROPN model into a ladder diagram.

The following is a summary of this study's main contributions.

1. This work develops a new approach to transforming a CROPN into LDs that can handle complicated structures and blocks in Petri nets compared with the study in [5] and [38].
2. The LDs are constructed for validation using the PLC Fiddle simulator; the results demonstrate that the developed approach provides low-overhead computation and has a simpler structure in comparison to the researches in [33], [34], [35], [36], and [38].
3. A global transportation resource place with a simpler structure and polynomial complexity is constructed to transfer all component types in a CROPN, compared to current researches [38].

The rest of the paper's structure is as follows: Section II presents the CROPN synthesis. Section III defines LDCROPN and provides a CROPN-to-LD conversion method. Section IV includes the verification and validation of the presented LDCROPN model. The typical application of the presented LDCROPN model is documented in Section V. The paper's conclusion, advantages, drawbacks, and future research are presented in Section VI.

## II. CROPN SYNTHESIS

A ROPN describes a component's manufacturing based on its stated processing routes as the component sequentially visits the resources. ROPN represents machines and buffers as "H-resources" [47], [58], [59], [63] of the same resource type and constructs each resource in a single place. The production procedures of a component are defined by the routes it takes to obtain its resources. If each component manufacturing process is constructed efficiently, the model is highly compressed and presents a number of essential structural properties. In addition, the ROPN defines material handling equipment as "G-resources" [47], [58], [59], [63]. The ROPN can be used to represent the AMS in two steps: (1) modeling the component manufacturing without considering how materials are moved; and (2) building the material handling processes based on the model developed in step 1 [68].

A CROPN can be defined as  $N = (P, T, C, I, O, K, D, M, In, En)$ , where

1.  $P = \{p_o\} \cup \{p_r\} \cup P_R$ , is a set of places, including a load/unload place  $p_o$ , a common resource place  $p_r$  that denotes material handling equipment, and a set of model resource places  $P_R, P_R = \cup_{i \in m} \{p_i\}, m > 0$ ;
2.  $T = \cup_{j \in n} \{t_j\}$ , is a set of net transitions with  $P \cap T = \emptyset, P \cup T \neq \emptyset$  and  $n > 0$ ;
3.  $C(p)$  and  $C(t)$  are the color sets related to net places and transitions, respectively, where

- a)  $\forall p_i \in P, C(p_i) = \cup_{i \in m} \{a_{iu}\}, u = |C(p_i)|$
  - b)  $\forall t_j \in T, C(t_j) = \cup_{j \in n} \{b_{jv}\}, v = |C(t_j)|$ ;
4.  $I(p, t): C(p) \times C(t) \rightarrow \mathbf{IN}$ , denotes the input function of the net, where  $\mathbf{IN} = \{0, 1, 2, \dots\}$ ;
  5.  $O(p, t): C(p) \times C(t) \rightarrow \mathbf{IN}$  denotes the output function of the net;
  6.  $K: P \rightarrow \mathbf{IN}$  represents the capacity function that assigns for each  $p_i \in P$  the maximum tokens and is expressed as  $K(p_i)$ ;
  7.  $D: T \rightarrow \mathbf{FT}$  is the firing time function that assigns the firing time  $D(t)$  to each transition  $t$ , where  $\mathbf{FT} > 0$ ;
  8.  $M: P \rightarrow N$  indicates the marking function, which adds the specified number of tokens to each place.  $M(p_i)$  denotes the number of tokens in place  $p_i$  irrespective of its tokens color.  $M(p_i, a_{ij})$  denotes the number of tokens in place  $p_i$  with color  $a_{ij}$ .  $M_o$  denotes the initial marking of the CROPN;
  9.  $In(p, t): C(p) \times C(t) \rightarrow \mathbf{IN}$  represents the inhibitor function of the net connecting an input place  $p$  to a transition  $t$ . The transition  $t$  is enabled if the number of tokens in  $p$  ( $M(p)$ ) is less than the inhibitor arc weight  $In(p, t)$ . If the transition  $t$  is fired, the marking of the place  $p$  will not change;
  10.  $En(p, t): C(p) \times C(t) \rightarrow \mathbf{IN}$  denotes the enabling function of the net connecting an input place  $p$  to a transition  $t$ . The transition  $t$  is enabled if the number of tokens in  $p$  ( $M(p)$ ) is at least equal to the enabling arc weight  $En(p, t)$ . If the transition  $t$  is fired, the marking of the place  $p$  will not change.

Assume that  $F = I(p, t) \cup O(p, t), \forall (p, t) \in F$ , the preset (input) and postset (output) transitions of place  $p$  can be expressed as " $\bullet p = \{t \in T | (t, p) \in F\}$ " and " $p^\bullet = \{t \in T | (p, t) \in F\}$ ," respectively. Similarly, the preset (input) and postset (output) places of transition  $t$  can be expressed as " $\bullet t = \{p \in P | (p, t) \in F\}$ " and " $t^\bullet = \{p \in P | (t, p) \in F\}$ ," respectively. A CROPN is said to be

1. an ordinary net if " $I(p_i, t_j)(a_{ih}, b_{jk}) = 1, \forall (p_i, t_j) \in F, p_i \in P, t_j \in T, a_{ih} \in C(p_i),$  and  $b_{jk} \in C(t_j)$ ."
2. a weighted net if " $I(p_i, t_j)(a_{ih}, b_{jk}) > 1, (p_i, t_j) \in F, \exists p_i \in P,$  and  $\exists t_j \in T, a_{ih} \in C(p_i),$  and  $b_{jk} \in C(t_j)$ ."
3. self-loop free if " $I(p_i, t_j)(a_{ih}, b_{jk}) > 0, O(p_i, t_j)(a_{ih}, b_{jk}) = 0, \forall (p_i, t_j) \in P \cup T, a_{ih} \in C(p_i),$  and  $b_{jk} \in C(t_j)$ ."
4. self-loop if " $I(p_i, t_j)(a_{ih}, b_{jk}) > 0, O(p_i, t_j)(a_{ih}, b_{jk}) > 0, \forall (p_i, t_j) \in P \cup T, a_{ih} \in C(p_i),$  and  $b_{jk} \in C(t_j)$ ."

In a CROPN, a  $t_j \in T$  called a "process-resource-enabled" [47] if

$$M(p_i, a_{ih}) \geq I(p_i, t_j)(a_{ih}, b_{jk}), \quad \forall p_i \in P, \forall p_i \in \bullet t_j, \\ a_{ih} \in C(p_i), \quad b_{jk} \in C(t_j) \quad (1)$$

and

$$K(p_i) \geq M(p_i, a_{ih}) + O(p_i, t_j)(a_{ih}, b_{jk}) - I(p_i, t_j)(a_{ih}, b_{jk}), \\ \forall p_i \in P, \quad \forall p_i \in t_j^\bullet, \quad a_{ih} \in C(p_i), \quad b_{jk} \in C(t_j) \quad (2)$$

If a transition  $t_j \in T$  is enabled at marking  $M$ , it can fire and the marking changes from  $M$  to  $M'$ , denoted as  $M[t_j]M'$  and stated as

$$\begin{aligned} & \forall p_i \in P, \quad a_{ih} \in C(p_i), \quad b_{ik} \in C(t_j), \quad M'(p_i, a_{ih}) \\ & = M(p_i, a_{ih}) + O(p_i, t_j)(a_{ih}, b_{jk}) - I(p_i, t_j)(a_{ih}, b_{jk}) \end{aligned} \quad (3)$$

CROPN has multiple circuits due to its high connectivity. Production process circuits (PPCs) are unique circuits in a CROPN that have a significant effect on the CROPN's liveness. There are no idle place in PPCs, which are denoted as  $PPCs = \{e_1, e_2, \dots, e_r\}$ ,  $r > 0$ , where  $e_r$  is a circuit in PPCs. When a circuit  $e_r$  flows from a node  $a$  to multiple nodes and then back to the originating node  $a$  without duplicating any nodes, it called an elementary circuit. A circuit  $e_r$  is non-elementary if it does not return to the initial node  $a$ . Moreover, in an  $e_r$ , the set of transitions must equal to the set of places, such that " $|P(e_r)| = |T(e_r)|$ ", where  $P(e_r)$  and  $T(e_r)$  are respectively the number of places and the number transitions in a circuit  $e_r$ , and the input places of transition  $t_j$  for a circuit  $e_r$  (" $\bullet t_j \in P(e_r)$ ,  $p_i \in \bullet t_j$ ") must be in a circuit  $e_r$ . If a transition  $t_j$  in a circuit  $e_r$  is fired and the tokens leave the  $e_r$ , the tokens are described as the "departing tokens" and the "cycling tokens" when they do not leave a circuit  $e_r$ , and can be formulated as

$$p_i \in P(e_r), M(e_r) = \Sigma M(p_i, e_r) \quad (4)$$

where  $M(p_i, e_r)$  indicates the number of tokens in  $p_i$  that enables  $t_j$  in an  $e_r$ .

It is said that a circuit  $e_r^c$  is an interactive subnet, which contains  $c$  PPCs,  $c > 0$  if its transitions and places are shared with at least one other PPC and its connections are strong. If a transition  $t_j$  in  $e_r^c$  ( $p_i \in P(e_r^c)$  and  $t_j \in (p_i \bullet \cap T(e_r^c))$ ) is fired and the tokens leave  $e_r^c$ , where  $P(e_r^c)$  and  $T(e_r^c)$  are respectively the sets of places and transitions in an  $e_r^c$ , the tokens in  $e_r^c$  are known as the "departing tokens;" otherwise, they are said to be the "cycling tokens," and can be formulated as

$$p_i \in P(e_r^c), M(e_r^c) = \Sigma M(p_i, e_r^c) \quad (5)$$

where  $M(p_i, e_r^c)$  denotes the number of tokens in  $p_i$  that enables  $t_j$  in an  $e_r^c$ .

In a CROPN, if

1.  $t_j$  satisfies conditions in Eqs. (1) and (2), it said to be a controlled transition;
2. there is at least one controlled transition  $t_j$ , then CROPN is called a controlled net;
3. a circuit  $e_r$  satisfies conditions in Eqs. (1) and (2), then it is said to be enabled;
4.  $t_j \in T(e_r)$  is live (deadlock-free), the  $t_j$  is called live;
5.  $t_j \notin T(e_r^c)$  and  $t_j \bullet \in P(e_r^c)$  and  $\bullet t_j \in P(e_r^c)$ , then  $t_j$  is called an input and output transition of an  $e_r^c$ , respectively.

The following are the conditions necessary for the deadlock-freeness in a CROPN:

$$M_o(p_o) \geq K(e_r) \quad (6)$$

1. **Condition 1:** A CROPN is not live if

*Proof:* See [47].

$$S'(e_r) \geq \quad (7)$$

2. **Condition 2:** At any marking  $M \in R(N, M_o)$ , a circuit  $e_r$  is live if

where  $R(N, M_o)$  represents the CROPN reachable markings, and  $S'(e_r)$  is the current number of spaces in an  $e_r$ , and can be expressed as

$$S'(e_r) = K(e_r) - M(e_r); \quad (8)$$

*Proof:* See [47].

Deadlock-free control policy (DFC-Policy) [69] can be implemented to avoid deadlock states if condition 2 is not satisfied, where DFC-Policy is defined as follows: At any marking  $M$ , transitions in  $T_I(e_r)$  and  $T(e_r)$  in the CROPN are controlled such that the following condition is fulfilled: when there is at most one place  $p_j \in P(e_r)$  such that  $|M(p_j)| \leq K(p_j)$ , and for any other  $p_i \in P(e_r)$ ,  $j \neq i$ , we have  $|M(p_i)| < K(p_i)$ . Thus, the job number condition is said to be satisfied for the  $e_r$ .

3. **Condition 3:** At any marking  $M$ , if the condition 2 is satisfied, then transitions in  $T(e_r)$  and  $T_I(e_r)$  are controlled, where  $T_I(e_r)$  represents the set of input transitions in an  $e_r$  (see DFC-Policy in [69]);
4. **Condition 4:** At any  $M \in R(N, M_o)$ , a circuit  $e_r^c$  is live if

$$S'(e_r) \geq 1, \forall e_r \quad (9)$$

and

$$\eta(e_r^c, M) \geq \quad (10)$$

where  $\eta(e_r^c, M)$  denotes the enabled PPCs in an  $e_r^c$

*Proof:* See [47].

5. **Condition 5:** At any marking  $M$ , an  $e_r^c$  is deadlock-free if
  - a) Any  $t_j \in T_I(e_r^c)$  is controlled, where  $T_I(e_r^c)$  denotes the set of input transitions in the  $e_r^c$ ;
  - b) before a controlled  $t_j$  fires, for any  $e_r \in V_{en}(t_j)$ ,  $S'(e_r) \geq 2$ , where  $V_{en}(t_j)$  represents the set of PPCs in the CROPN and the  $t_j$  denotes the input transition of these PPCs;
  - c) the marking  $M$  can be changed to  $M'$  after the firing of the  $t_j$ , such that  $\eta(e_r^c, M') \geq 1$ .

*Proof:* See [47].

6. **Condition 6:** If a CROPN has no PPC, then it is always live.

*Proof:* See [47].

Based on above conditions, Algorithm 1 shows the deadlock avoidance algorithm for a CROPN model.

It appears that to apply the control policy presented in Algorithm 1, we need to evaluate each PPC to check if conditions 1–6 are fulfilled. The number of PPCs may be

**Algorithm 1** : Deadlock Avoidance Policy for the CROPN

---

**Input:** The CROPN model;  
**Output:** The controlled CROPN model;

1. Compute PPCs =  $\{e_1, e_2, \dots, e_r\}$ ,  $r > 0$  of the CROPN model;
2. Compute all reachable markings  $R(N, M_0)$  of the CROPN model;
3. if the PPCs  $\neq \emptyset$ , then
4. for  $(1 \leq |PPCs| \leq r++)$ , do
5. if an  $e_r$  is not an interactive, then
6. for  $(0 \leq |R(N, M_0)| \leq k++)$ , do
7.  $K(e_r) = \Sigma K(p, e_r)$ ,  $p \in P(e_r)$ ;
8.  $M_k(e_r) = \Sigma M_w(p, e_r)$ ,  $p \in P(e_r)$ ;
9.  $S'(e_r) = K(e_r) - M_k(e_r)$ ;
10. if  $S'(e_r) \geq 0$ , then
11. The  $e_r$  has no deadlock states;
12. else if
13. The  $e_r$  has deadlock states;
14. Implement the DFC-Policy in [69] to avoid deadlock states;
15. end for
16. else if an  $e_r$  is an interactive, then
17. for  $(0 \leq |R(N, M_0)| \leq k++)$  do
18.  $K(e_r) = \Sigma K(p, e_r)$ ,  $p \in P(e_r)$ ;
19.  $M_k(e_r) = \Sigma M_k(p, e_r)$ ,  $p \in P(e_r)$ ;
20.  $S'(e_r) = K(e_r) - M_k(e_r)$ ;
21. if  $S'(e_r) \geq 1$  and  $\eta(e_r^c, M_k) \geq 0$ , then
22. The  $e_r$  has no deadlock states;
23. else if
24. The  $e_r$  has deadlock states;
25. Implement the condition 5 to avoid deadlock states;
26. end for
27. end if
28. end for
29. else if
30. The  $e_r$  is live based on condition 6;
31. end if
32. **Output** t: A controlled CROPN model
33. **End**

---

exponentially dependent on the number of machines. Therefore, it lacks computational efficiency. It is important to note that the CROPN must be deadlocked if a transition  $t$  is not fireable according to the control policy and firings of the transition  $t$ . To check if a controlled transition  $t$  is fireable, we can simply fire  $t$  to check if the CROPN is deadlocked. If it is, then it indicates that the control policy's conditions have been violated; if not, the transition  $t$  can be fired. In addition, if the material handling systems (MHSs) are shared resources in the manufacturing process, deadlock may occur due to the competition for the MHS. In the CROPN, we designed a common place  $p_r$  representing the MHSs with colored tokens in it and a number of arcs into the CROPN such that  $p_r$  and some transitions in the CROPN form a number of self-loops. We can now demonstrate that the CROPN is live if there are several self-loops or when place  $p_r$  has multiple self-loops with various circuit transitions. Assume  $T_r$  to be the set of these transitions. In the CROPN, when transition  $t \in T_r$  is enabled, then transition  $t$  can fire since  $p_r$  contains a token. If many transitions in the CROPN, let a transition set  $T_1 \subset T_r$  to be simultaneously enabled, then one of these transitions,

suppose transition  $t_1$ , can be selected to fire. After transition  $t_1$  fires, a token is returned to a place  $p_r$ , and another transition in  $T_1$  can be selected to fire. Consequently, in the CROPN, after a sequence of transition firings, each transition on the circuit can be enabled at any marking  $M$ . This indicates that the net is deadlock-free.

To show the synthesis of a CROPN, consider the AMS presented in Figure 1. It consists of three machines  $m_1$ – $m_3$  (operations resources), two robots  $r_1$  and  $r_2$  (material handling equipment resources), and a load/unload station (L/U station for short). The AMS produces two part types, A and B, and their processes are as follows:

1. part A: L/U station  $\rightarrow m_1 \rightarrow m_2 \rightarrow$  L/U station;
2. part B: L/U station  $\rightarrow m_1 \rightarrow m_3 \rightarrow$  L/U station or L/U station  $\rightarrow m_1 \rightarrow m_3 \rightarrow m_1 \rightarrow m_3 \rightarrow$  L/U station.

Robots  $r_1$  and  $r_2$  are employed for components A and B loading and unloading operations, as well as transporting components between machines  $m_1$  and  $m_2$ , and machines  $m_1$  and  $m_3$ , respectively. Each component can be processed at a time by machines  $m_1$ – $m_3$ . Figure 2 presents the developed CROPN for the AMS shown in Figure 1. As presented in Figure 2, the model for the operation's resources, i.e., machines  $m_1$ – $m_3$ , is constructed where  $p_i$ ,  $i = 1, 2, 3$  represents  $m_i$ . The model for the L/U station and two robots,  $r_1$  and  $r_2$ , is respectively represented by an idle place  $p_o$  and a common place  $p_r$ . The operation path of component A is modelled by  $p_o \rightarrow t_{010A} \rightarrow p_1 \rightarrow t_{120A} \rightarrow p_2 \rightarrow t_{200A} \rightarrow p_o$ , while component B is represented by  $p_o \rightarrow t_{010B} \rightarrow p_1 \rightarrow t_{130B} \rightarrow p_3 \rightarrow t_{300B} \rightarrow p_o$  or  $p_o \rightarrow t_{010B} \rightarrow p_1 \rightarrow t_{130B} \rightarrow p_3 \rightarrow t_{310B} \rightarrow p_1 \rightarrow t_{130B} \rightarrow p_3 \rightarrow t_{300B} \rightarrow p_o$ . Note that the form of the transition is  $t_{xyzk}$ , where  $x$  ( $x = 1, 2, \text{ or } 3$ ) denotes the current operation position of the colored token,  $y$  ( $y = 1, 2, \text{ or } 3, x \neq y$ ) represents the distention operation position of the colored token,  $z$  ( $z = 1, 2, 3, \dots$ ) denotes if there is an alternative processing path for component type, and  $k$  ( $k = A$  or B) represents two component types that can be produced by the system.

To construct transporting components operations, robot  $r_1$  is used for loading component A to  $p_1$  via  $t_{010A}$ , transporting component A from  $p_1$  to  $p_2$  via transition  $t_{120A}$ , and unloading component A from  $p_2$  via  $t_{200A}$ . Therefore, the model includes the arcs  $(p_r, t_{010A})$ ,  $(t_{010A}, p_r)$ ,  $(p_r, t_{120A})$ ,  $(t_{120A}, p_r)$ ,  $(p_r, t_{200A})$ , and  $(t_{200A}, p_r)$ . Robot  $r_2$  is used for loading component B to  $p_1$  via  $t_{010B}$ , transporting component B from  $p_1$  to  $p_3$  via  $t_{130B}$ , unloading component B from  $p_3$  via  $t_{300B}$  or transporting component B from  $p_3$  to  $p_1$  via  $t_{310B}$ . Thus, the model involves the arcs  $(p_r, t_{010B})$ ,  $(t_{010B}, p_r)$ ,  $(p_r, t_{130B})$ ,  $(t_{130B}, p_r)$ ,  $(p_r, t_{310B})$ ,  $(t_{310B}, p_r)$ ,  $(p_r, t_{300B})$ ,  $(t_{300B}, p_r)$ ,  $(p_r, t_{300B})$ , and  $(t_{300B}, p_r)$ . Finally, add the initial markings of the developed CROPN by placing two tokens with color  $c_{p1}$ ,  $c_{p2}$  into the idle place  $p_o$  where  $c_{p1}$  and  $c_{p2}$  represent components A and B, respectively; placing two tokens with color  $c_{r1}$  and  $c_{r2}$  into the common place  $p_r$  where  $c_{r1}$  and  $c_{r2}$  represent robots  $r_1$  and  $r_2$ , respectively. Thus, the initial markings of the idle place  $p_o$  and the common place  $p_r$  can be represented as  $M_o(p_o) = \{c_{p1}, c_{p2}\}$  and  $M_o(p_r) = \{c_{r1}, c_{r2}\}$ , respectively.

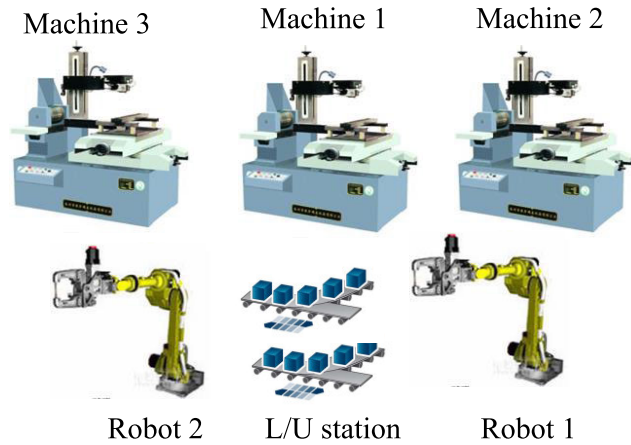


FIGURE 1. Example of an automated manufacturing system.

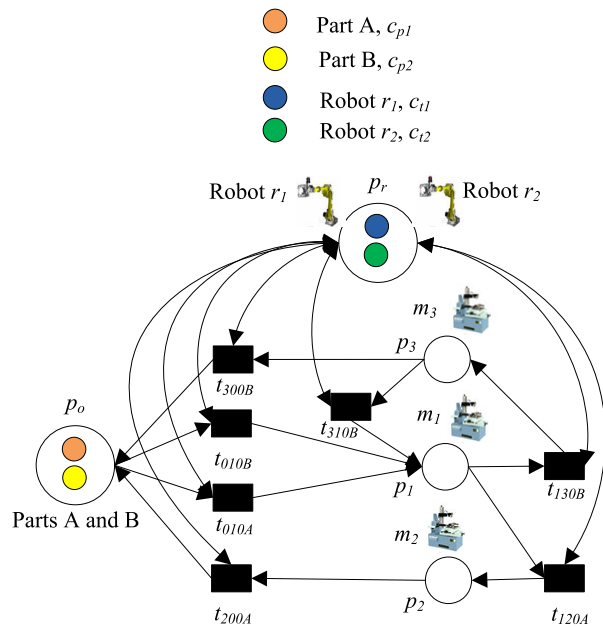


FIGURE 2. A CROPN of the AMS shown in Figure 1.

To present the behavior of the constructed CROPN shown in Figure 2, we assume that  $p_r$  and  $p_o$  have a capacity of two, while  $p_1, p_2,$  and  $p_3$  have a capacity of one. Figure 3 illustrates the reachability graph for the CROPN presented in Figure 2. As seen in Figure 3, for component A, if machine  $p_1$  and robot  $r_1$  are idle, the transition  $t_{010A}$  can be fired, and a token  $c_{p1}$  and a token  $c_{r1}$  are selected from  $p_o$  and  $p_r$ , respectively. If  $t_{010A}$  is fired, one token  $c_{p1}$  and one token  $c_{r1}$  are transferred to the places  $p_1$  and  $p_r$ , respectively. When machine  $p_2$  and robot  $r_1$  are idle and the processing time of machine  $p_1$  has elapsed, the transition  $t_{120A}$  can be fired, and a token  $c_{p1}$  and a token  $c_{r1}$  are selected from  $p_1$  and  $p_r$ , respectively. When  $t_{120A}$  is fired, one token  $c_{p1}$  and one token  $c_{r1}$  are transferred to places  $p_2$  and  $p_r$ , respectively. When robot  $r_1$  is idle and the processing time of machine  $p_2$  has elapsed, the transition  $t_{200A}$  can be fired, and a token  $c_{p1}$  and a token  $c_{r1}$  are selected from  $p_2$  and

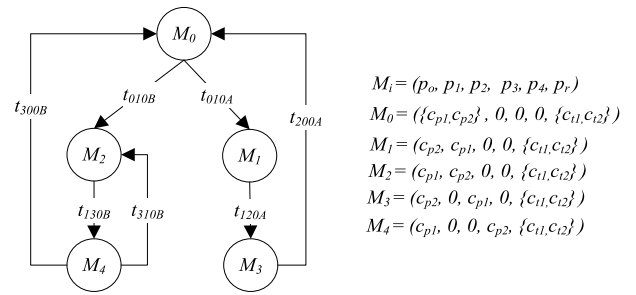


FIGURE 3. Reachable markings of a CROPN model shown in Figure 2.

$p_r$ , respectively. When  $t_{200A}$  is fired, one token  $c_{p1}$  and one token  $c_{r1}$  are transferred to places  $p_0$  and  $p_r$ , respectively.

Similarly, for component B, when machine  $p_1$  and robot  $r_2$  are idle, the transition  $t_{010B}$  can be fired, and a token  $c_{p2}$  and a token with color  $c_{r2}$  are selected from places  $p_o$  and  $p_r$ , respectively. If  $t_{010B}$  is fired, a token  $c_{p2}$  and a token  $c_{r2}$  are transferred to the places  $p_1$  and  $p_r$ , respectively. When machine  $p_3$  and robot  $r_2$  are idle and the processing time of machine  $p_1$  has elapsed, the transition  $t_{130B}$  can be fired, and a token  $c_{p2}$  and a token  $c_{r2}$  are selected from places  $p_1$  and  $p_r$ , respectively. If  $t_{130B}$  is fired, a token  $c_{p2}$  and a token  $c_{r2}$  are transferred to places  $p_3$  and  $p_r$ , respectively. If component B is produced properly and without defects, it is transferred to the L/U station. Otherwise, if component B contains defects, rework is required, and the defective component moves to machine  $p_1$ . In the first scenario, if robot  $r_2$  is idle and the processing time of machine  $p_3$  has elapsed, the transition  $t_{300B}$  can be fired, and a token  $c_{p2}$  and a token  $c_{r2}$  are selected from places  $p_3$  and  $p_r$ , respectively. If  $t_{300B}$  is fired, a color  $c_{p2}$  and a token  $c_{r2}$  are transferred to places  $p_0$  and  $p_r$ , respectively. In the latter scenario, if machine  $p_1$  and robot  $r_2$  are idle and the processing time of machine  $p_3$  has elapsed, the transition  $t_{310B}$  can be fired, and a token  $c_{p2}$  and a token  $c_{r2}$  are selected from places  $p_3$  and  $p_r$ , respectively. If  $t_{310B}$  is fired, a token  $c_{p2}$  and a token  $c_{r2}$  are transferred to places  $p_1$  and  $p_r$ , respectively, and then move to  $p_3$  and  $p_0$  as shown in the first scenario.

To illustrate if the CROPN is live based on Algorithm 1, reconsider the model depicted in Figure 2. It involves a PPC:  $e_1 = \{p_1, t_{130B}, p_3, t_{310B}\}$ . The control condition described in Eq. (8) is used to figure out how many spaces are free in PPC  $e_1$  and are presented in Table 1. As shown in Table 1, the subnet PPC  $e_1$  in the CROPN is deadlock-free and the condition in Eq. (8) is fulfilled for all markings depicted in Figure 3.

### III. LADDER DIAGRAM CROPN SYNTHESIS

As shown in [70], the ladder diagram programming language runs on the same idea as an electromechanical relay, with the flexibility that includes block diagrams. IEC-61131-3 characterized ladder diagram as “modeling networks of simultaneous functioning electromechanical elements, such as relay contacts and coils, timers, counters, etc. The logic

TABLE 1. The current spaces in PPC  $e_1$ .

| $M_k$ | $M(e_i) = \Sigma M_i(p_i, e_i)$ | $K(e_i) = \Sigma K(p_i, e_i)$ | $S'(e_i) = K(e_i) - M_k(e_i)$ | If $S'(e_i) \geq 1$ |
|-------|---------------------------------|-------------------------------|-------------------------------|---------------------|
| 0     | 0                               | 2                             | 2                             | Yes                 |
| 1     | 1                               | 2                             | 1                             | Yes                 |
| 2     | 1                               | 2                             | 1                             | Yes                 |
| 3     | 0                               | 2                             | 2                             | Yes                 |
| 4     | 1                               | 2                             | 1                             | Yes                 |

“AND,” “OR,” “AND–OR,” “mathematical calculations,” “precedence,” “interlocking,” “timers,” “auto-loop,” and “counters” are evaluated on the control lines. The inputs and outputs of “AND,” “OR,” “AND–OR,” “auto-loop,” “precedence,” and “interlocking” are discrete, while the inputs and outputs of “mathematical comparisons,” “counters,” and “timers” are analog discrete. In a PLC, the control algorithm operates in a cycle and typically carries out the following five operations [70]:

1. receiving physical inputs,
2. copying the state of physical inputs,
3. comparing the current copy to the previous copy,
4. copying the state of physical outputs,
5. and delivering these states to physical components.

Table 2 presents a part of the programming standard IEC 61131-3 for LD elements.

A ladder diagram CROPN (LDCROPN) can be expressed by  $N = (P, T, I, O, C, K, M_o, I_s, Q, \mathcal{M}, \Phi, \delta, \Xi)$ , where

1.  $P, T, F, W, M_o,$  and  $K$  are introduced in Section II;
2.  $I_s = \cup_{i \in k} \{I_i\}, k > 0,$  represents a set of discrete input physical signals;
3.  $Q = \cup_{j \in q} \{Q_j\}, q > 0, I_s \cap Q = \emptyset,$  denotes a set of places for physical output signals;
4.  $\mathcal{M} = \cup_{l \in b} \{\mathcal{M}_l\}, b > 0,$  represents a finite set of places for discrete memory signals;
5.  $\Phi$  indicates the firing rule, which adds a Boolean firing rule for each transition and can be represented as  $\Phi(t)$ ;
6.  $\delta$  represents the output function, which adds a Boolean place function for each place and can be represented as  $\delta(p)$ ;
7.  $\Xi$  indicates the output function integrating all marked places outputs  $\delta$ ;  $\Xi: \{0, 1\}^P \rightarrow \{0, 1, -, c\}$ , where  $c$  indicates the conflict when the same signal is assumed to be zero in one place and 1 in another; to calculate  $\Xi(m_a), m_a: \Xi(m_a) \prod_i \delta(p_i)$  is used.

If the left and right power rails are eliminated from a ladder diagram, a set of contacts and coils can satisfy the condition in which any two elements are connected. Then it is called a rung  $g, g = \{1, 2, 3, \dots\}$ . Let coils  $k$  and rungs  $g, k, g \in \{1, 2, 3, \dots\}$ , the ladder diagram graph of the LDCROPN can be represented as  $\Omega_{LD} = (\Sigma, \gamma)$  if

1.  $\Sigma = \Sigma_L \cup \Sigma_R \cup \Sigma_v$  is the vertex set, where
  - a)  $\Sigma_L = \cup_{1 \leq i \leq g} \{\Sigma_{li}\}, g > 0,$  is the intersection of the  $i$ th rung with the left power rail in the ladder diagram,

- b)  $\Sigma_R = \cup_{1 \leq j \leq g} \{\Sigma_{rj}\}, g > 0,$  is the intersection of the  $i$ th rung with the right power rail in the ladder diagram.
2.  $\Sigma_v$  represents a collection of components expressing ladder diagram symbols, and its elements are  $v_{y1,y2,y3}$ , where
  - a)  $y1$  denotes the LD symbol’s name, which including coil, counter, contact, or timer;
  - b)  $y2 = \{0, 1, 2, 3, 4, 5\}$  where 0, 1, 2, 3, 4, 5 indicates, respectively, a “normally open contact, a normally closed contact, a coil, a negated coil, a set coil, and a reset coil;”
  - c)  $y3 = \{0 i\}$ , where 0 and  $i (1 \leq i \leq g)$  are named by  $y1$  and represent a contact and the  $i$ th coil from top to bottom, respectively.
3.  $\gamma$  represents a set of edges. Assume that the  $v_x$  and  $v_y$  are two elements,  $v_x, v_y \in \gamma, \forall v_x, v_y \in \Sigma$  when there is a connected line between two elements in the ladder diagram that relate to  $v_x$  and  $v_y$ , respectively.

In a ladder diagram graph  $\Omega_{LD}$ ,

1. if the vertex  $v_{y1,y2,y3}$  is connected directly to  $\Sigma_{rj}, \Sigma_{rj} \in \Sigma_R, 1 \leq j \leq g,$  it is called an assigned-component of the  $i$ th rung and indicates the output variables.
2. if a path between  $\Sigma_{li}$  and  $\Sigma_{rj}$  that includes an assigned-component  $v_{y1,y2,y3} \in \Sigma_v, (k = y3, 1 \leq k \leq g)$  exists, then the path can be represented as  $\Delta(v_{y1,y2,y3})$ .
3. if a path between  $\Sigma_{li}$  and  $\Sigma_{rj}$  are removed from  $\Delta(v_{y1,y2,y3})$ , then the path can be represented as an instruction path  $\Delta(v_{y1,y2,y3})$ , and expressed by  $\Delta_{k,i}, 1 \leq i \leq |\Delta * (v_{y1,y2,y3})|, k = y3$  denotes the  $k$ th coil in the  $\Omega_{LD}$  and indicates the  $k$ th path of  $\Delta * (v_{y1,y2,y3})$ .
4. if the  $t_j$  is enabled at marking  $M$ , it can fire and the marking  $M$  changes to  $M'$ , and is formulated as
  - a) if  $t^\bullet \in P \cap Q$  and  $\bullet t \in P$ , then  $M'(t^\bullet) = (\prod M(\bullet t) = 1 \text{ AND } M(t^\bullet) = 0)$ ;
  - b) if  $t^\bullet \in P \cap M$  and  $\bullet t \in P$ , then  $M'(t^\bullet) = (\prod M(\bullet t) = 1 \text{ AND } M(t^\bullet) = 0)$ ;
  - c) if  $t^\bullet \in P \cap Q$  and  $\bullet t \in P$ , then  $M'(t^\bullet) = (\sum M(\bullet t) = 1 \text{ AND } M(t^\bullet) = 0)$ ;
  - d) if  $t^\bullet \in P \cap M$  and  $\bullet t \in P$ , then  $M'(t^\bullet) = (\sum M(\bullet t) = 1 \text{ AND } M(t^\bullet) = 0)$ ;
  - e) if  $t^\bullet \in P \cap Q$  and  $\bullet t \in P$ , then  $M'(t^\bullet) = (\sum (\prod M(\bullet t) = 1, \dots, \prod M(\bullet t) = 1) = 1 \text{ AND } M(t^\bullet) = 0)$ ;
  - f) if  $t^\bullet \in P \cap M$  and  $\bullet t \in P$ , then  $M'(t^\bullet) = (\sum (\prod M(\bullet t) = 1, \dots, \prod M(\bullet t) = 1) = 1 \text{ AND } M(t^\bullet) = 0)$ ;
  - g) Otherwise,  $M'(t^\bullet) = M(t^\bullet)$ .
5. due to the marking  $M'$ , reset output places is modified from  $M$  to  $M'$  and stated as
  - a) if  $t^\bullet \in P \cap Q$  and  $\bullet t \in P$ , then  $M'(t^\bullet) = (\prod M(\bullet t) = 0 \text{ AND } M(t^\bullet) = 1)$ ;
  - b) if  $t^\bullet \in P \cap M$  and  $\bullet t \in P$ , then  $M'(t^\bullet) = (\prod M(\bullet t) = 0 \text{ AND } M(t^\bullet) = 1)$ ;

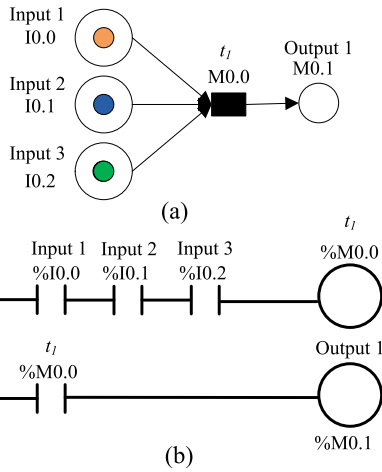


FIGURE 4. An example of AND logic: (a) CROPN model, and (b) Equivalent ladder diagram.

- c) if  $t^* \in P \cap Q$  and  $\bullet t \in P$ , then  $M'(t^*) = (\sum M(\bullet t) = 0 \text{ AND } M(t^*) = 1)$ ;
- d) if  $t^* \in P \cap M$  and  $\bullet t \in P$ , then  $M'(t^*) = (\sum M(\bullet t) = 0 \text{ AND } M(t^*) = 1)$ ;
- e) if  $t^* \in P \cap Q$  and  $\bullet t \in P$ , then  $M'(t^*) = (\sum (\prod M(\bullet t) = 1, \dots, \prod M(\bullet t) = 1) = 0 \text{ AND } M(t^*) = 1)$ ;
- f) if  $t^* \in P \cap M$  and  $\bullet t \in P$ , then  $M'(t^*) = (\sum (\prod M(\bullet t) = 1, \dots, \prod M(\bullet t) = 1) = 0 \text{ AND } M(t^*) = 1)$ ;
- g) Otherwise,  $M'(t^*) = M(t^*)$ .

6. if all rungs  $g$ , in  $\Omega_{LD}$  work without error in any case, then the LDCROPN is called live.

To demonstrate the AND logic, consider the CROPN model depicted in Figure 4(a). Based on above conditions, when the firing condition of the  $t_1$  is met, the contacts Input 1 (IO.0), Input 2 (IO.1), and Input 3 (IO.2) have tokens, they have a signal state of '1' and the transition  $t_1$  (M0.0) is enabled to fire. If transition  $t_1$  is fired,  $t_1$  is set on (signal state "1"), and then Output 1 coil (M0.1) is set on (signal state "1"). Figure 4(b) depicts the ladder diagram that corresponds to Figure 4(a).

Figure 5 illustrates the OR logic, as seen in Figure 5(a), if the firing condition of the  $t_1$ ,  $t_2$ , and  $t_3$  is met (all contacts Input 1 (IO.0), Input 2 (IO.1), and Input 3 (IO.2) have at least on token), the transitions  $t_1$  (M0.0),  $t_2$  (M0.1), and  $t_3$  (M0.2) are enabled to fire. If any transitions  $t_1$ ,  $t_2$ , or  $t_3$  is fired, it is set on (signal state "1"), and then Output 1 coil (M0.3) is set on (signal state "1"). Figure 5(b) depicts the ladder diagram that corresponds to Figure 5(a). Figure 6 displays AND-OR logic. As shown in Figure 6(a), Output 1 coil (M0.2) is set on (signal state "1") if contacts Input 1 (IO.0) and Input 2 (IO.1) allow transition  $t_1$  (M0.0) to fire, or if contact Input 3 (IO.2) allows transition  $t_2$  (M0.1) to fire. Figure 6(b) depicts the ladder diagram that corresponds to Figure 6(a).

In addition, Figure 7 shows the auto-loop. As presented in Figure 7(a), if the contacts Input 1 (IO.0) and Input 2 (IO.1)

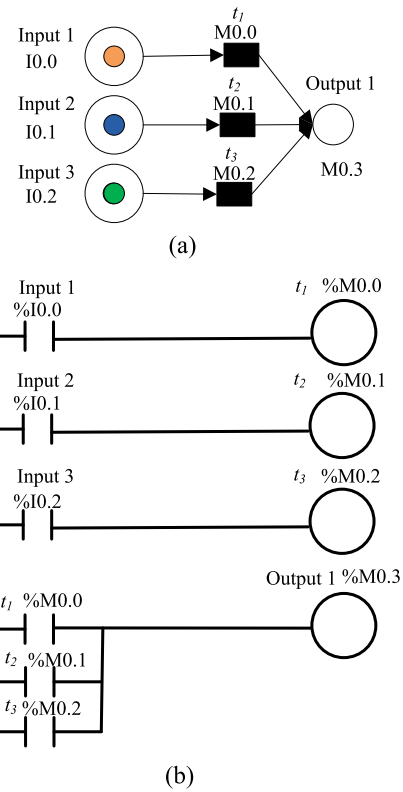


FIGURE 5. An example of OR logic: (a) CROPN model, and (b) Equivalent ladder diagram.

enable transition  $t_1$  (M0.0) to fire or the contact Output 1 (M0.2) enables transition  $t_2$  (M0.1) to fire, then Output 1 coil (M0.2) is set on (signal state "1"). Figure 7(b) presents the ladder diagram that corresponds to Figure 7(a).

Figure 8 shows the interlocking logic. As shown in Figure 8(a), if the contacts Input 1 (IO.0) and  $\sim$ Output 2 (M0.3) enable transition  $t_1$  (M0.0) to fire, then Output 1 coil (M0.2) is set on (signal state "1"), and it prevents the powering of Output 2 coil (M0.3). When Output 2 coil is powered first, it enables transition  $t_2$  (M0.1) to fire, and then Output 1 is set off (signal state "0"). Figure 8(b) presents the ladder diagram that corresponds to Figure 8(a). Note that the inhibitor arc shown in Figure 8(a) connects the place "Output 1" or "Output 2" to a transition " $t_2$ " or " $t_1$ ," and the transition " $t_2$ " or " $t_1$ " is enabled if the place "Output 1" or "Output 2" has tokens that are less than the inhibitor arc weight.

Figure 9 shows the precedence logic. As shown in Figure 9(a), if transition  $t_2$  (M0.1) is fired,  $t_2$  is set on (signal state "1"), and then Output 2 coil (M0.3) is set on (signal state "1"); if the contacts Input 1 (IO.0) and Output 2 (M0.3) enable transition  $t_1$  (M0.0) to fire, then Output 1 coil (M0.2) is set on (signal state "1"). Figure 9(b) presents the ladder diagram that corresponds to Figure 9(a). Note that the enable arc shown in Figure 9(a) connects the place "Output 2" to a transition " $t_1$ ," and the transition " $t_1$ " is enabled if the place "Output 2" has tokens that are equal to the enable arc weight. Note that the enable arc shown in Figure 9(a) connects the



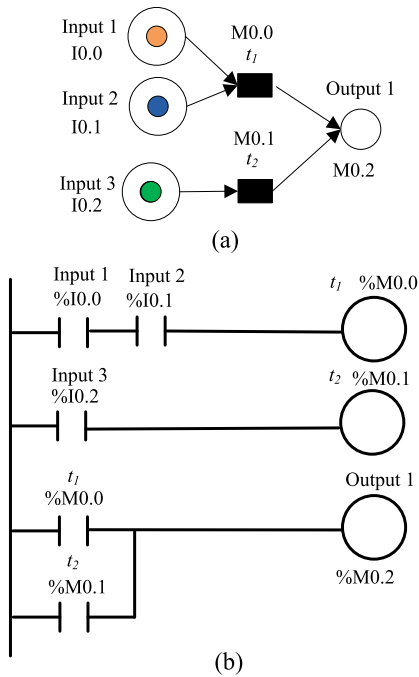


FIGURE 6. An example of AND-OR logic: (a) CROPN model, and (b) Equivalent ladder diagram.

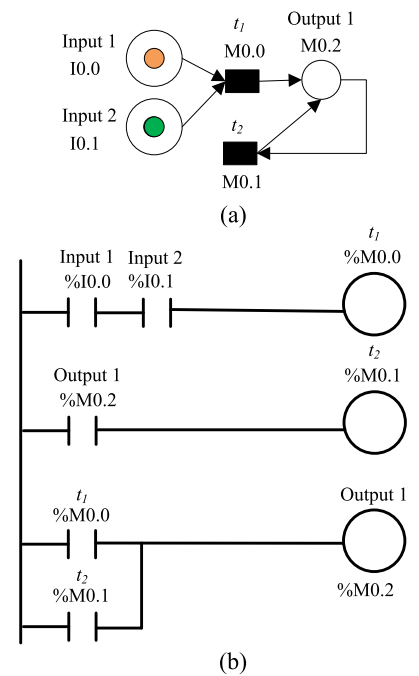


FIGURE 7. An example of auto-loop logic: (a) CROPN model, and (b) Equivalent ladder diagram.

place “Output 2” to a transition “ $t_1$ ,” and the transition “ $t_1$ ” is enabled if the place “Output 2” has tokens that are equal to the enable arc weight.

In the CROPN, a conflict appears if an input place has multiple output transitions. Consider the design depicted in Figure 10(a). The conflict can be identified if there is one

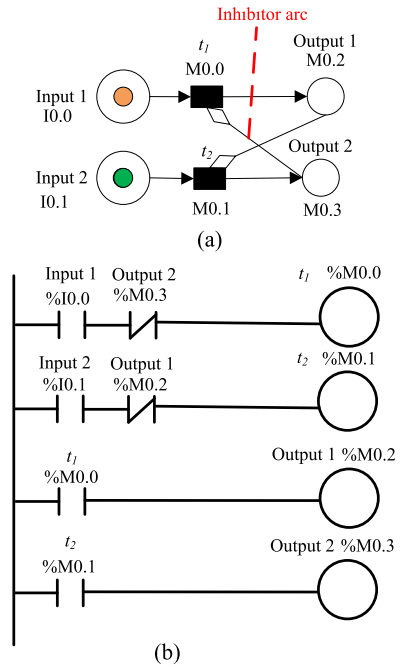


FIGURE 8. An example of interlocking logic: (a) CROPN model, and (b) Equivalent ladder diagram.

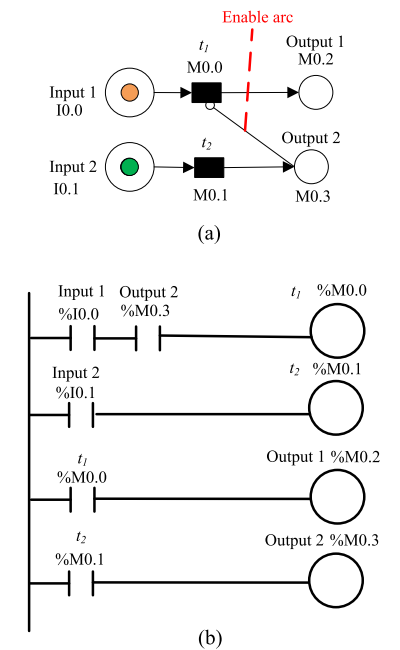


FIGURE 9. An example of precedence logic: (a) CROPN model, and (b) Equivalent ladder diagram.

token in Input 1 (IO.0) and firing rules of the  $t_1$ (M0.0),  $t_2$ (M0.1), and  $t_3$ (M0.2) exist simultaneously. In a ladder diagram, the conflict can be addressed by determining the order of priority for the conflicting transitions, which the PLC immediately reads the rungs from top to bottom, left to right. We assume that the  $t_1$  has precedence over the  $t_2$  and  $t_3$ , and the  $t_2$  has precedence over the  $t_3$ . Therefore, the Output 1 coil

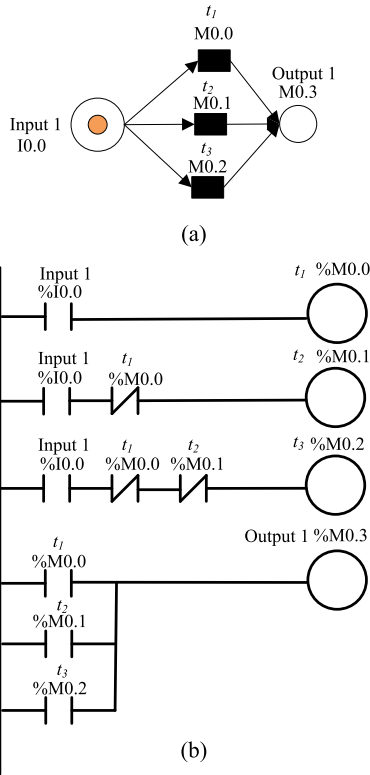


FIGURE 10. An example of a conflict state: (a) CROPN model, and (b) Equivalent ladder diagram.

(M0.3) is set on (signal state “1”) if contact Input 1 allows transition  $t_1$ ,  $t_2$ , or  $t_3$  to fire. Figure 10(b) depicts the ladder diagram that corresponds to Figure 10(a).

Figure 11 shows the on-delay timer (TON) block. As depicted in Figure 11(a), if the firing rule of the  $t_1$ (M0.0) is met (the contact Input 1 (I0.0) has a token), the  $t_1$  is enabled to fire. If the  $t_1$  is fired,  $t_1$  is set on (signal state “1”), and if the “elapsed time” (ET) adds a “base time” (BT),  $ET = ET + BT$ , and if it is equal to or greater than the “preset time” (PT) ( $PT = 5s$ ), the Output 1 coil (M0.1) is powered. Note that “elapsed time” and “preset time” are analog. Figure 11(b) depicts the ladder diagram that corresponds to Figure 11(a).

Figure 12 shows the off-delay timer (TOF) block. As depicted in Figure 12(a), if the firing rule of the  $t_1$ (M0.0) is met (the contact Input 1 (I0.0) has a token), the  $t_1$  is enabled to fire. If the  $t_1$  is fired and the  $t_1$  is set on (signal state “1”), then the ET is set to zero and the Output 1 coil (M0.1) is powered. If the signal state of the contact Input 1 is “0,” the ET variable adds BT ( $ET = ET + BT$ ), and if the ET is equal to or less than the PT ( $PT = 5s$ ), the Output 1 coil is de-energized. Figure 12(b) depicts the ladder diagram that corresponds to Figure 12(a).

Figure 13 presents the up-counter block. In the up-counter block, if Input 1 (I0.0) (the pulse to the counter) is set on (signal state “1”), then the current value (CV) variable of the up-counter is set to  $CV = CV + 1$ ; if Input 2 (I0.1) has signal state “0” and the CV is equal to or greater than PV, then the

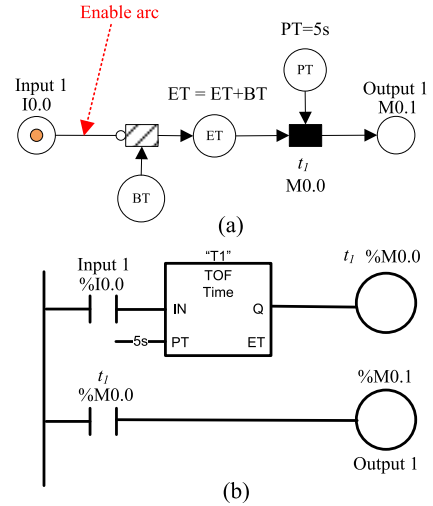


FIGURE 11. An example of on-delay timer block: (a) CROPN model, and (b) Equivalent ladder diagram.

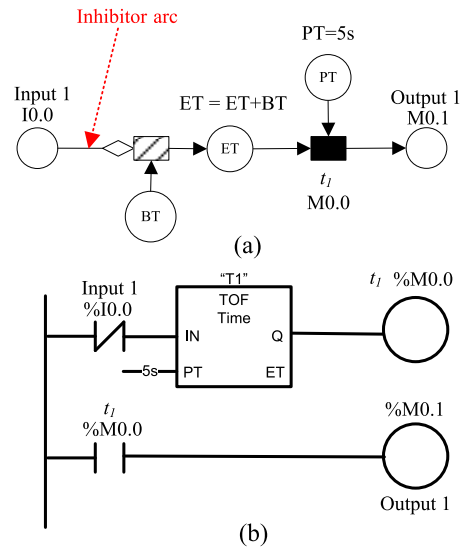


FIGURE 12. An example of off-delay timer block: (a) CROPN model, and (b) Equivalent ladder diagram.

$t_1$ (M0.0) is enabled to fire. If the  $t_1$  is fired and the  $t_1$  is set on (signal state “1”), then Output 1 coil (M0.1) is powered. If the contact Input 2 is set to “on” (signal state “1”), then the Output 1 coil is de-energized and the CV variable of the up-counter is set to zero. Figure 13(b) depicts the ladder diagram that corresponds to Figure 13(a).

Figure 14 presents the down-counter block. In the down-counter block, if Input 1 (I0.0) (the pulse to the counter) is set on (signal state “1”), then the current value (CV) variable of the down-counter is set to  $CV = CV - 1$ ; if Input 2 (I0.1) has signal state “0” and the CV is equal to or less than zero, then the  $t_1$  (M0.0) is enabled to fire. If the  $t_1$  is fired and the  $t_1$  is set on (signal state “1”), then Output 1 coil (M0.1) is powered. If the contact Input 2 is set to “on” (signal state “1”), then the Output 1 coil is de-energized and the CV variable of the

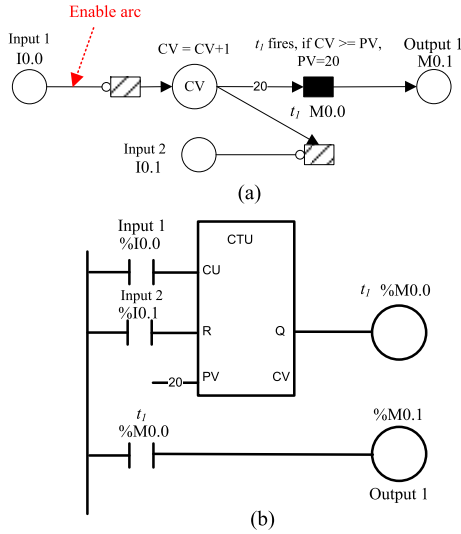


FIGURE 13. An example of up-counter block: (a) CROPN model, and (b) Equivalent ladder diagram.

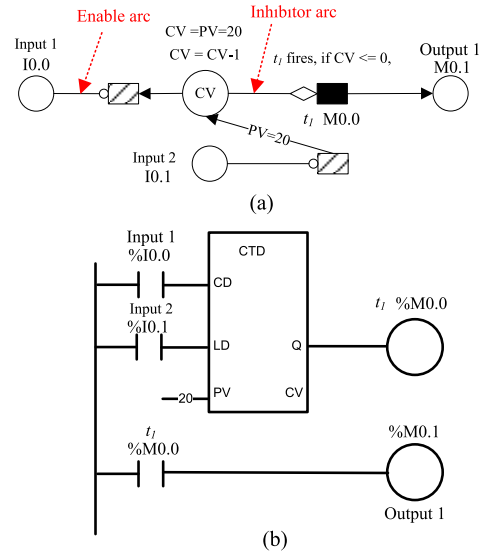


FIGURE 14. An example of down-counter block: (a) CROPN model, and (b) Equivalent ladder diagram.

down-counter is equal to PV. Figure 14(b) depicts the ladder diagram that corresponds to Figure 14(a).

We can use the CROPN depicted in Figure 2 to illustrate the LDCROPN synthesis. At the input buffer and machines  $m_1$ – $m_3$ , the resulting CROPN fails identification and recognition of the two components A and B. Seven infrared reflective sensors are added to the CROPN depicted in Figure 2. Sensor 1 and sensor 2 are respectively installed in the input buffer area to allow for the identification of components A and B. The addition of sensors 3 and 4 respectively enables machine  $m_1$  to recognize components A and B. Sensor 5 is added to machine  $m_2$  to enable the detection of component A. In addition, sensor 6 assists identification of accepted component B in machine  $m_3$ . The addition of sensor 7 enables machine  $m_3$  to identify defective component B. Figure 15 depicts the designed LDCROPN. As seen in Figure 15, inputs  $I_{0.0}$  and  $I_{0.1}$  respectively, represent the system's start  $I_1$  and stop  $I_2$  buttons. In addition, inputs  $I_{0.2}$ ,  $I_{0.3}$ ,  $I_{0.4}$ ,  $I_{0.5}$ ,  $I_{0.6}$ ,  $I_{0.7}$ , and  $I_{1.0}$  are used for the reading of sensors 1 ( $I_3$ ), 2 ( $I_4$ ), 3 ( $I_5$ ), 4 ( $I_6$ ), 5 ( $I_7$ ), 6 ( $I_8$ ), and 7 ( $I_9$ ), respectively.  $Q_{0.0}$ ,  $Q_{0.1}$ ,  $Q_{0.2}$ ,  $Q_{0.3}$ ,  $Q_{0.4}$ , and  $Q_{0.5}$  are applied to the action places as output bits to represent respectively  $Q_1$  (machine  $m_1$  operation for component A),  $Q_2$  (machine  $m_1$  operation for component B),  $Q_3$  (machine  $m_2$  operation for component A),  $Q_4$  (machine  $m_3$  operation for component B),  $Q_5$  (robot  $r_1$  operation), and  $Q_6$  (robot  $r_2$  operation).

In contrast, memory bits  $M_{0.0}$ ,  $M_{0.1}$ ,  $M_{0.2}$ ,  $M_{0.3}$ ,  $M_{0.4}$ ,  $M_{0.5}$ , and  $M_{0.6}$  are allocated to transitions  $\Phi(t_{010A})$ ,  $\Phi(t_{010B})$ ,  $\Phi(t_{120A})$ ,  $\Phi(t_{130B})$ ,  $\Phi(t_{310B})$ ,  $\Phi(t_{200A})$ , and  $\Phi(t_{300B})$ , respectively. In addition, memory bits  $M_{0.7}$ ,  $M_{2.0}$ ,  $M_{2.1}$ ,  $M_{1.0}$ ,  $M_{1.1}$ , and  $M_{1.2}$  are allocated to places  $\delta(p_1)$  (for component A),  $\delta(p_1)$  (for component B),  $\delta(p_2)$ ,  $\delta(p_3)$ ,  $\delta(p_r)$  (for component A), and  $\delta(p_r)$  (for component B), respectively. Notable is the fact that two memory bits can be assigned to a place whose capacity holds two tokens.

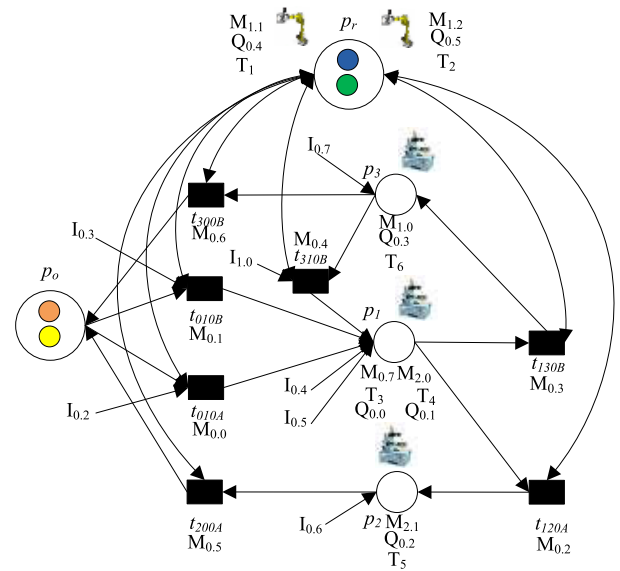


FIGURE 15. The LDCROPN of a CROPN model shown in Figure 2.

In order to model time delays, on-delay timers  $T_1$ ,  $T_2$ ,  $T_3$ ,  $T_4$ ,  $T_5$ , and  $T_6$  are assigned to places  $p_1$  (component A processing time in machine  $m_1$ ),  $p_1$  (component B processing time in machine  $m_1$ ),  $p_2$  (component A processing time in machine  $m_2$ ),  $p_3$  (component B processing time in machine  $m_3$ ),  $p_r$  (component A transporting time), and  $p_r$  (component B transporting time), respectively.

Following are the specifications for the LDCROPN model.

1. The loading process of the  $r_1$  is set on via  $t_{010A}$  when a component A is available in the L/U station, i.e.,  $M(p_o) = \{c_{p1}\}$  and sensor 1 recognizes it, the  $r_1$  is free, i.e.,  $M(p_r) = \{c_{r1}\}$ , and there is no process in the  $m_1$ , i.e.  $M(p_1) = 0$ . The loading process of the  $r_1$  is set off via  $t_{010A}$ , if a component A is available in the  $m_1$ .

The operation of the  $m_1$  is set on with Q0.0 if a component A is available in the  $m_1$ , i.e.,  $M(p_1) = \{c_{p1}\}$ , and sensor 3 recognizes it and when the loading process time of the  $r_1$  has finished. The Boolean firing functions of  $t_{011A}$ ,  $p_1$ , and  $p_r$  are stated as follows.

- $\Phi(t_{010A}) = M0.0 = I0.2 \& M1.2 \& \overline{M0.7} \& \overline{I0.1}$ .
- $\delta(p_1) = M0.7 = I0.4 \& \overline{M0.0} \& \overline{I0.1}$ .
- $\delta(p_r) = M1.2 = \overline{T_3} \& \overline{T_4} \& \overline{T_5} \& \overline{I0.1}$

2. The transporting process of the  $r_1$  is set on with Q0.5 via  $t_{120A}$  if the processing time of the  $m_1$  has finished and robot  $r_1$  is free, i.e.,  $M(p_r) = \{c_{r1}\}$ . The transporting process of robot  $r_1$  is set off through  $t_{120A}$  when a component A is available in the  $m_2$ . The operation of the  $m_2$  is set on when a component A is available in the  $m_2$ , i.e.,  $M(p_2) = \{c_{p1}\}$ , and the transporting process time of the  $r_1$  has elapsed. The processing of the  $m_2$  is set off via  $t_{200A}$ , and occurs when the processing time of the  $m_2$  has elapsed. The Boolean firing functions of  $t_{120A}$ ,  $t_{200A}$ ,  $p_2$ , and  $p_r$  are stated as follows.

- $\Phi(t_{120A}) = M0.2 = M1.6 \& M1.2 \& \overline{M2.1} \& \overline{I0.1}$ .
- $\Phi(t_{200A}) = M0.5 = (M1.7 \& M1.2) \& \overline{I0.1}$
- $\delta(p_2) = M2.1 = I0.6 \& \overline{M0.2} \& \overline{I0.1}$
- $\delta(p_r) = M1.2 = \overline{T_3} \& \overline{T_4} \& \overline{T_5} \& \overline{I0.1}$

3. The loading process of the  $r_2$  is set on via  $t_{010B}$  if a component B is available in the L/U station, i.e.,  $M(p_o) = \{c_{p2}\}$  and sensor 1 recognizes it, the  $r_2$  is free, i.e.,  $M(p_r) = \{c_{r2}\}$ , and there is no processing in the  $m_1$ , i.e.  $M(p_1) = 0$ . The loading process of the  $r_2$  is set off via  $t_{010B}$  when a component B is available in the  $m_1$ . The processing of the  $m_1$  is set on if a component B is available in the  $m_1$ , i.e.,  $M(p_1) = \{c_{p1}\}$ , and sensor 4 recognizes it and when the loading process time of robot  $r_2$  has elapsed. The Boolean firing functions of  $t_{010B}$ ,  $p_1$ , and  $p_r$  are stated as follows.

- $\Phi(t_{010B}) = M0.1 = I0.3 \& M1.1 \& \overline{M2.0} \& \overline{I0.1}$ .
- $\delta(p_1) = M2.0 = I0.5 \& \overline{M0.1} \& \overline{I0.1}$ .
- $\delta(p_r) = M1.1 = \overline{T_3} \& \overline{T_4} \& \overline{T_6} \& \overline{I0.1}$

4. The transporting process of robot  $r_2$  is set on via  $t_{130B}$  if the processing time of the  $m_1$  has finished and robot  $r_2$  is free, i.e.,  $M(p_r) = \{c_{r2}\}$ . The transporting process of robot  $r_2$  is set off through  $t_{130B}$  when a component B is available in the  $m_3$ . The processing of the  $m_3$  is set on if a component A is available in the  $m_3$ , i.e.,  $M(p_3) = \{c_{p2}\}$ , and the transporting process time of robot  $r_2$  has finished. The processing of the  $m_3$  is set off via  $t_{300B}$  (accepted component) or  $t_{310B}$  (rejected component), and occurs when the processing time of the machine  $m_3$  has elapsed. The Boolean firing functions of  $t_{130B}$ ,  $t_{300B}$ ,  $p_3$ , and  $p_r$  are stated as follows.

- $\Phi(t_{130B}) = M0.3 = M1.5 \& M1.1 \& \overline{M1.0} \& \overline{I0.1}$ .
- $\Phi(t_{310B}) = M0.4 = I1.0 \& M2.2 \& M1.1 \& \overline{M2.0} \& \overline{I0.1}$ .
- $\Phi(t_{300B}) = M0.6 = M2.3 \& M1.1 \& \overline{I0.1}$ .
- $\delta(p_3) = M1.0 = I0.7 \& \overline{M0.3} \& \overline{I0.1}$
- $\delta(p_r) = M1.1 = \overline{T_3} \& \overline{T_4} \& \overline{T_6} \& \overline{I0.1}$ .

5. The transporting process of robot  $r_2$  for a rejected component B is set on via  $t_{310B}$  when the processing time of the

$m_3$  has finished and robot  $r_2$  is free, i.e.,  $M(p_r) = \{c_{r2}\}$ . The unloading process of robot  $r_2$  is set off through  $t_{310B}$  if a component B is available in the  $m_1$ . The processing of the  $m_1$  is set on if a component B is available in the  $m_1$ , i.e.,  $M(p_1) = \{c_{p2}\}$ , and the transporting process time of robot  $r_2$  has finished. The Boolean firing functions of  $t_{310B}$ ,  $p_1$ , and  $p_r$  are stated as follows.

- $\Phi(t_{310B}) = M0.4 = I1.0 \& M2.2 \& M1.1 \& \overline{M2.0} \& \overline{I0.1}$ .
- $\delta(p_1) = M2.0 = I0.5 \& \overline{M0.1} \& \overline{I0.1}$
- $\delta(p_r) = M1.1 = \overline{T_3} \& \overline{T_4} \& \overline{T_6} \& \overline{I0.1}$ .

Figure 16 illustrates the ladder diagram of the LDCROPN model depicted in Figure 15. As seen in Figure 16, rungs 1 and 2 represent the system's start and stop buttons, respectively. The initial states of robots 1 and 2 are shown on rungs 3 and 4, respectively. Moreover, rungs 5 to 11 indicate transition firing conditions. Rungs 12 to 15 show the actions and delays of a machine's processes. The 16th and 17th rungs depict the actions and delays of the robot's activities. Finally, rungs 18 to 23 represent the set and reset of robots' and machines' operations, as well as the states of memories.

#### IV. VERIFICATION AND VALIDATION OF THE LDCROPN

Using a PLC Fiddle simulator [71], the validation and verification of the proposed LDCROPN are provided in this section. PLC-fiddle is advantageous for engineers and readers who need a simplified version of the PLC design application. As an instructional tool, PLC Fiddle provides its own benefits. The PLC Fiddle design approach will assist readers in learning the logic and thought process of PLCs; therefore, readers are not required to have experience developing PLCs. PLC Fiddle has the benefit of being simple to use for beginners. To start programming logic practice, readers do not need to complete a difficult installation process. In an educational environment, learning objectives will be fulfilled [72], [73], [74], [75]. To verify using PLC Fiddle, we must first analyze the other components required for a production control system to work. Consider then the on/off switches and several input/output sensors which monitor the arrival of components. Third, choose the PLC, which receives the information from the sensors and translates it into Boolean algebra so that it can interpret and make judgments. Figure 17 depicts a part of the PLC Fiddle diagrams for the LD shown in Figure 16. Based on these PLC Fiddle diagrams, the suggested LD is error-free.

#### V. TYPICAL APPLICATION OF THE LDCROPN

Consider the LDCROPN depicted in Figure 18 to illustrate the LDCROPN synthesis. As seen in Figure 18, Robots  $r_1$  and  $r_2$  are used to load and unload components A and B. At the input buffer and machines  $m_1$ – $m_4$ , the resulting LDCROPN fails identification and recognition of the two components A and B. Seven an infrared reflective sensors are added to the LDCROPN model. Sensor 1 and sensor 2 are respectively installed in the input buffer area to allow for the identification of components A and B. The addition of sensor 3 enables the

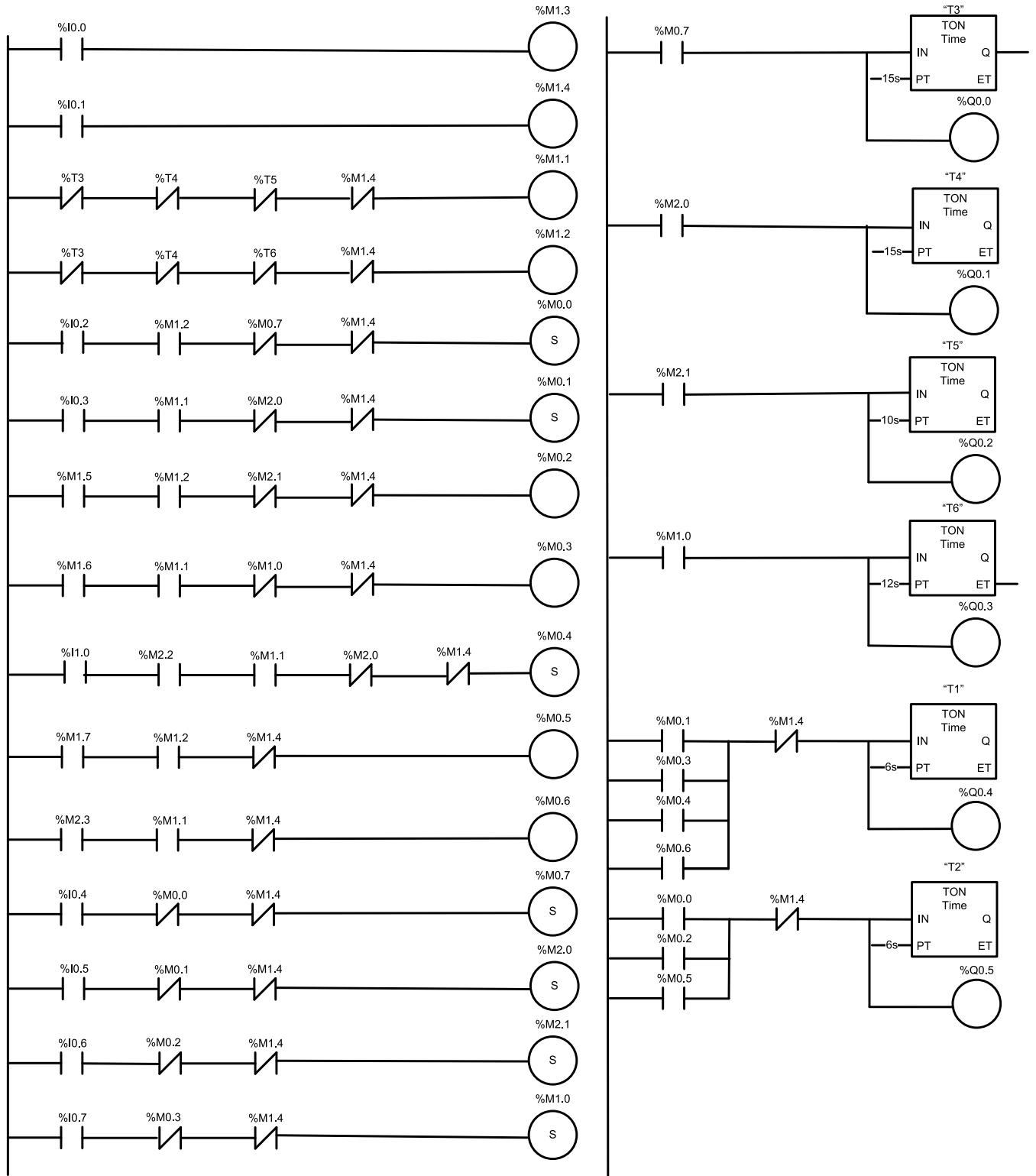


FIGURE 16. LD for the LDCROPN model presented in Figure 15.

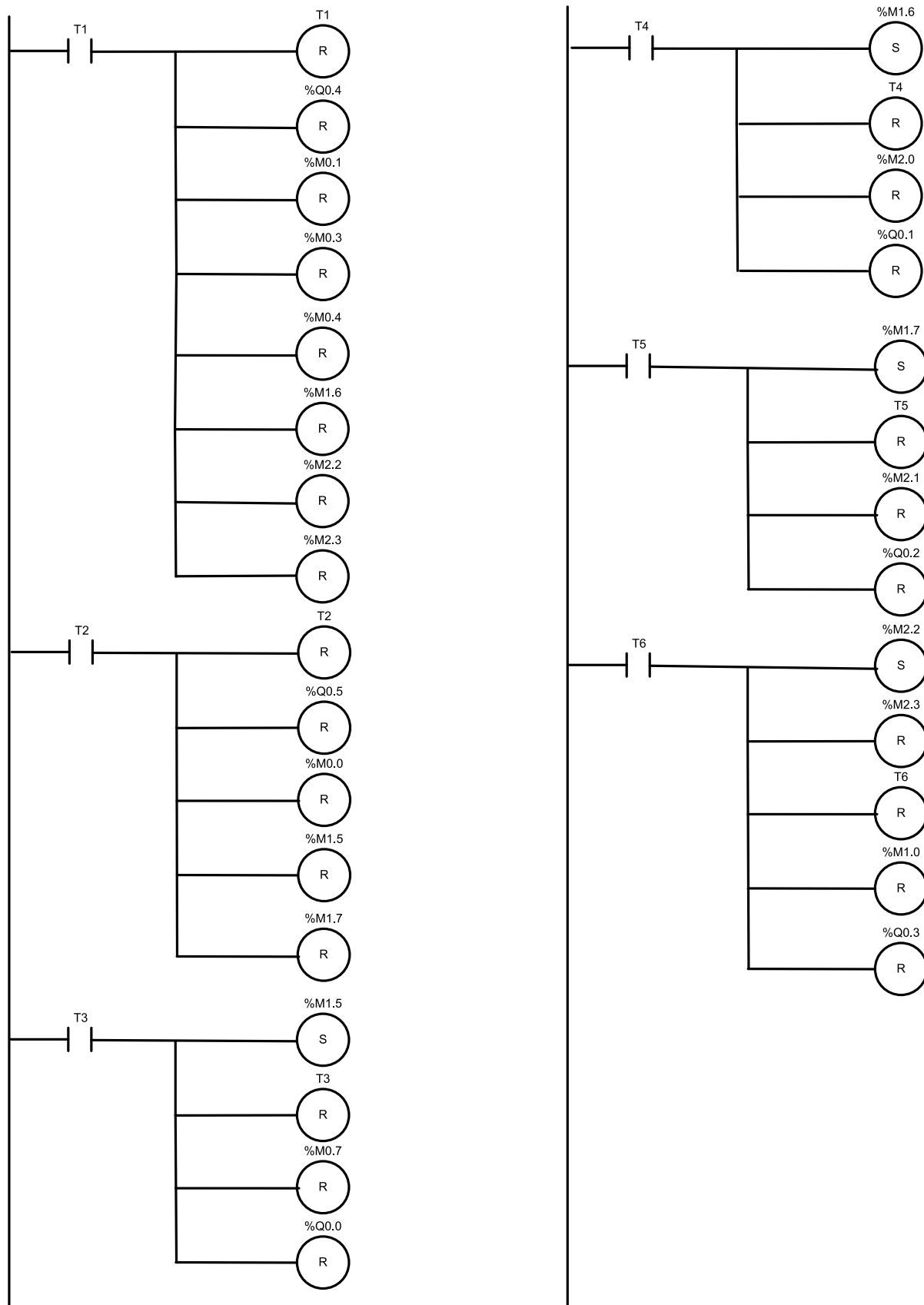



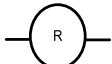

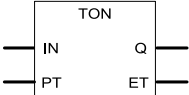

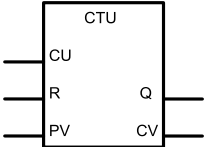
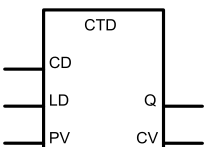


FIGURE 16. (Continued.) LD for the LDCROPN model presented in Figure 15.

TABLE 2. Sample of LD elements based on IEC 61131-3 programming standard.

| LD Symbol   | Description   |
|---|---|
|    | It represents the “normally open contact”, which is frequently used to represent the variables input, internal, and output. When the signal state is "1", it is closed (on).  |
|    | It indicates the “normally closed contact”. Input, internal, and output variables are typically represented by this symbol in PLC systems. When the signal state is "0", it is closed (on).   |
|    | It represents the “Coil”, which is frequently used to indicate the variables internal and output. When the input coil's signal is "1", the output coil is set; when the input coil's signal is "0", the output coil is reset.   |
|    | It indicates the “coil reset”, which is used to reset the output signal to be "0".  |
|    | It indicates the “coil set” that is used to set the output signal to be "1".  |
|    | It represents the “on-delay timer (TON)”. If the input IN signal is "1", and elapsed time (ET) adds base time and when $ET \geq$ preset time (PT), then the output Q's signal state is "1".   |
|    | It denotes the “off-delay timer (TOF)”. If the input IN signal state is "1", then the output Q's signal state is "1", and the ET variable is set to zero. If the input IN signal is "0", the ET variable adds base time, and if $ET \geq$ PT, the output Q's signal state is "0".   |
|   | It represents the up-counter block. It is an incrementing counter, which indicates it counts "up" for each off-to-on input change to its input "CU". If the input "CU" signal state changes from "0" to "1", the current value (CV) is incremented by one and stored at the CV output. The IEC 61131-3 programming standard dictates that the counter output "Q" should activate if $CV \geq PV$ . The "R" is the reset input; activating this input resets the counter's CV to zero. |
|  | It represents the down-counter block. It is used to decrement the counter value at the CV output. If the input "CD" signal state changes from "0" to "1", the CV is decremented by one and stored at the CV output. The IEC 61131-3 programming standard dictates that the counter output "Q" should activate if $CV \leq 0$ . The "LD" is the “load” input; activating this input causes the counter's CV to equal PV.   |

$m_1$  to recognize component A. Sensor 4 is added to the  $m_4$  to enable the detection of component B. In addition, sensor 5 and sensor 6 assist identification of components A and B, respectively, in the  $m_3$  and the  $m_4$ . The addition of sensor 7 enables the  $m_3$  to identify defective components A.

Table 3 depicts the PLC's I/O physical signals, while the Boolean transition firing function and the Boolean place function are presented in Table 4. I0.0 and I0.1 represent the system's start and stop buttons, respectively. In addition, I0.2, I0.3, I0.4, I0.5, I0.6, I0.7, and I1.0 are used for sensor readings. Q0.0, Q0.1, Q0.2, Q0.3, Q0.4, and Q0.5 are applied to the action places as output bits. In contrast, memory Addresses are allocated to transitions and places. Notable is the fact that two memory bits can be assigned to a place whose capacity holds two tokens. In order to model time delays, timed transitions and places are connected to an on-delay timer.

Following are the specifications for the LDCROPN model.

1. The loading process of robot  $r_1$  is set on via  $t_{011A}$  when a component A is available in the L/U station, i.e.,  $M(p_o) = \{c_{p1}\}$  and sensor 1 recognizes it, robot  $r_1$  is free, i.e.,  $M(p_r) = \{c_{r1}\}$ , and there is no processing in the  $m_1$ , i.e.  $M(p_1) = 0$ . The loading process of robot  $r_1$  is set off via  $t_{011A}$ , if a component A is available in the  $m_1$ . The processing of the  $m_1$  is set on when a component A is available in the  $m_1$ , i.e.,  $M(p_1) = \{c_{p1}\}$ , and sensor 3 recognizes it and when the loading process time of the  $r_1$  has finished.
2. The unloading process of the  $r_1$  is set on via  $t_{130A}$  if the processing time of the  $m_1$  has finished and the  $r_1$  is free, i.e.,  $M(p_r) = \{c_{r1}\}$ . The unloading process of robot  $r_1$  is set off through  $t_{130A}$  if a component A is available in the  $m_3$ . The processing of the  $m_3$  is set on if a component A is available in the  $m_3$ , i.e.,  $M(p_3) = \{c_{p1}\}$ , and the loading

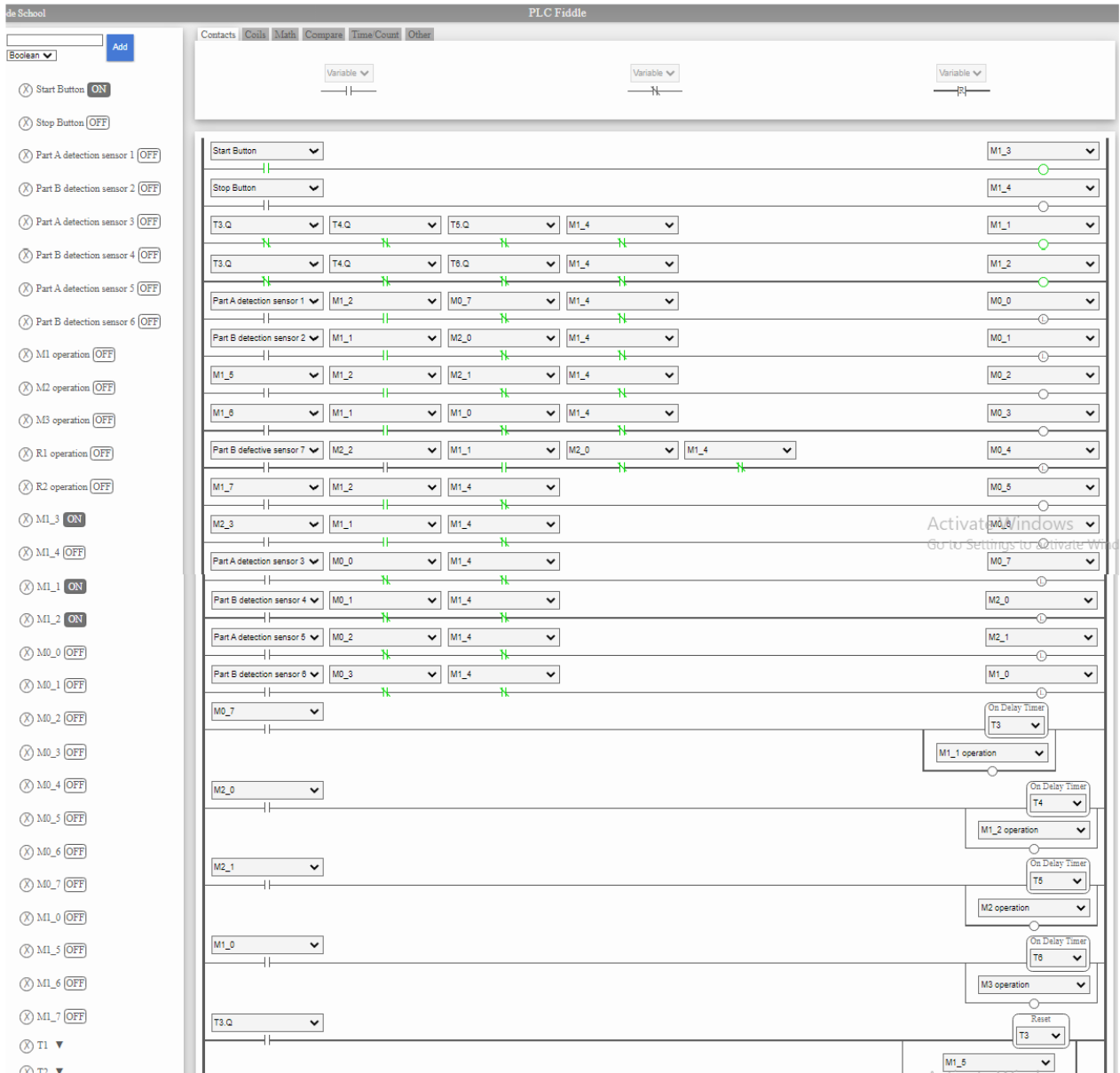


FIGURE 17. Part of PLC Fiddle diagrams of LD model presented in Figure 12.

TABLE 3. The addresses of the PLC’s I/O physical signals and memories for the LDCROPN model presented in Figure 18.

| Type  | LDCROPN | Address | Description                           | Type   | LDCROPN | Address | Description     |
|-------|---------|---------|---------------------------------------|--------|---------|---------|-----------------|
| Input | $I_1$   | I0.0    | On switch                             | Output | $Q_1$   | Q0.0    | $m_1$ operation |
|       | $I_2$   | I0.1    | Off switch                            |        | $Q_2$   | Q0.1    | $m_2$ operation |
|       | $I_3$   | I0.2    | Component A detection sensor 1        |        | $Q_3$   | Q0.2    | $m_3$ operation |
|       | $I_4$   | I0.3    | Component B detection sensor 2        |        | $Q_4$   | Q0.3    | $m_4$ operation |
|       | $I_5$   | I0.4    | Component A detection sensor 3        |        | $Q_5$   | Q0.4    | $r_1$ operation |
|       | $I_6$   | I0.5    | Component B detection sensor 4        |        | $Q_6$   | Q0.5    | $r_2$ operation |
|       | $I_7$   | I0.6    | Component A detection sensor 5        | -      | -       | -       |                 |
|       | $I_8$   | I0.7    | Component B detection sensor 6        | -      | -       | -       |                 |
|       | $I_9$   | I1.0    | Defect component A detection sensor 7 |        |         |         |                 |

process time of robot  $r_1$  has finished. The processing of the  $m_3$  is set off through  $t_{310A}$  (rejected component) or  $t_{300A}$  (accepted component) if its processing time has finished.

3. The unloading process of the  $r_2$  for a rejected component A is set on via  $t_{310A}$  when the processing time of the  $m_3$  has finished and the  $r_2$  is free, i.e.,  $M(p_r) = \{c_{r2}\}$ . The



TABLE 4. The Boolean firing and place functions, and memories addresses for the LDCROPN model presented in Figure 18.

| LDCROPN          | Address | Boolean firing function   | LDCROPN          | Address | Boolean place function   |
|------------------|---------|---|------------------|---------|--|
| $\Phi(t_{011A})$ | M0.0    | $I_3 \& M_{1,1} \& \overline{M_{0,7}} \& \overline{I_2}$            | $\Phi(t_{200B})$ | M0.6    | $M_{2,3} \& M_{1,1} \& \overline{I_2}$                                 |
| $\Phi(t_{040B})$ | M0.1    | $I_4 \& M_{1,2} \& \overline{M_{1,0}} \& \overline{I_2}$            | $\delta(p_1)$    | M0.7    | $I_5 \& \overline{M_{0,0}} \& \overline{I_2}$                          |
| $\Phi(t_{130A})$ | M0.2    | $M_{1,5} \& M_{1,1} \& \overline{M_{2,1}} \& \overline{I_2}$        | $\delta(p_2)$    | M2.0    | $I_6 \& \overline{M_{0,3}} \& \overline{I_2}$                          |
| $\Phi(t_{420B})$ | M0.3    | $M_{1,6} \& M_{1,1} \& \overline{M_{2,0}} \& \overline{I_2}$        | $\delta(p_3)$    | M2.1    | $I_7 \& \overline{M_{0,2}} \& \overline{I_2}$                          |
| $\Phi(t_{310A})$ | M0.4    | $I_9 \& M_{1,7} \& M_{1,2} \& \overline{M_{0,7}} \& \overline{I_2}$ | $\delta(p_4)$    | M1.0    | $I_8 \& \overline{M_{0,1}} \& \overline{I_2}$                          |
| $\Phi(t_{300A})$ | M0.5    | $M_{2,2} \& M_{1,2} \& \overline{I_2}$                              | $\delta(p_r)$    | M1.1    | $\overline{T_3} \& \overline{T_4} \& \overline{T_6} \& \overline{I_2}$ |
|                  |         |   |                  | M1.2    | $\overline{T_3} \& \overline{T_4} \& \overline{T_6} \& \overline{I_2}$ |

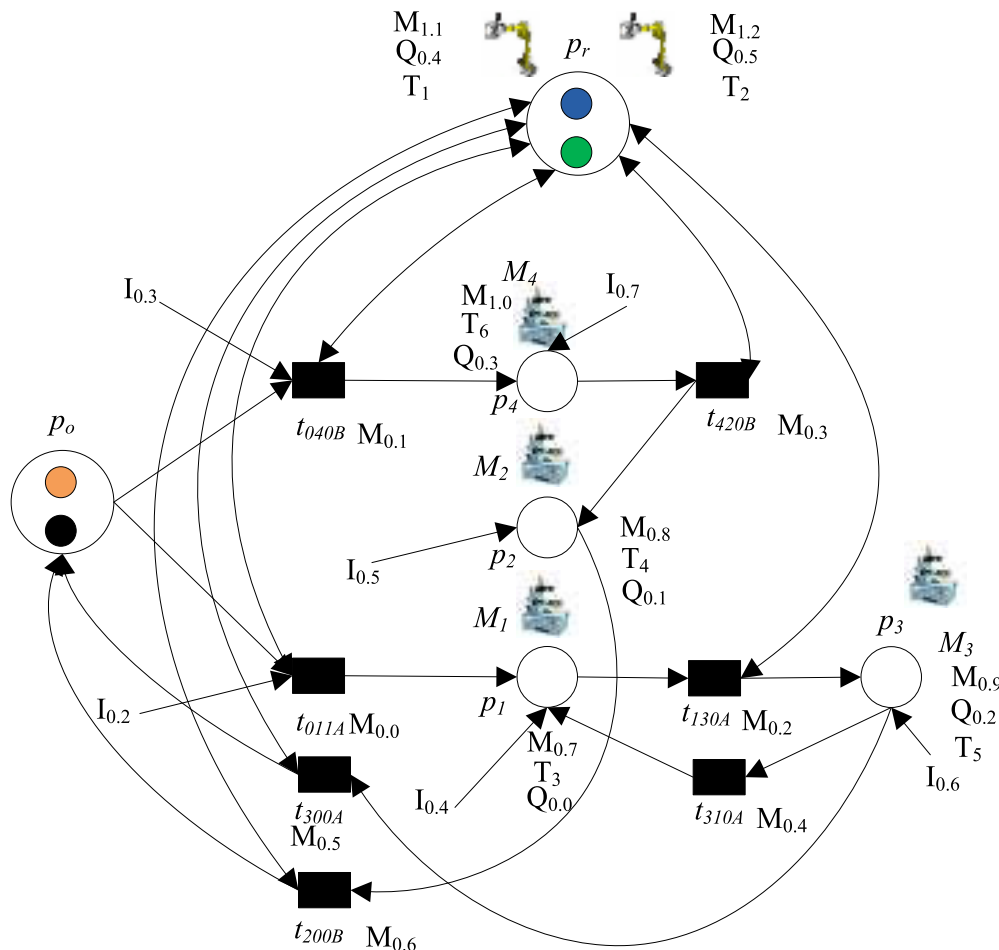


FIGURE 18. The developed LDCROPN for the typical application.

unloading process of the  $r_2$  is set off through  $t_{310A}$  if a component A that is available in the  $m_1$ . The processing of  $m_1$  is set on when a component A is available in the  $m_1$ , i.e.,  $M(p_1) = \{c_{p1}\}$ , and the loading process time of robot  $r_2$  has finished.

4. The unloading process of the  $r_2$  for an accepted component A is set on via  $t_{300A}$  when the processing time of the  $m_3$  has finished and the  $r_2$  is free, i.e.,  $M(p_r) = \{c_{r2}\}$ . The unloading process of the  $r_2$  is set off through  $t_{300A}$  when the loading process time of the  $r_2$  has finished.

5. If component B appears in the L/U station, i.e.,  $M(p_o) = c_{p2}$ , the  $r_2$  is free, i.e.,  $M(p_r) = c_{r2}$ , sensor 2 recognizes it, and the  $m_4$  is not operating, i.e.,  $M(p_4) = 0$ , then the loading process of the  $r_2$  is set on via  $t_{040B}$ . When component B is detected in the  $m_4$ , the loading process of the  $r_2$  is set off through  $t_{040B}$ . The operation of the  $m_4$  is set on when a component B is available in the  $m_4$ , i.e.,  $M(p_1) = c_{p2}$ , sensor 3 identifies it, and the  $r_2$ 's loading process time has finished.

6. If the operating time of the  $m_4$  has finished and the  $r_1$  is free, i.e.,  $M(p_r) = c_{r1}$ , the unloading process of robot  $r_1$

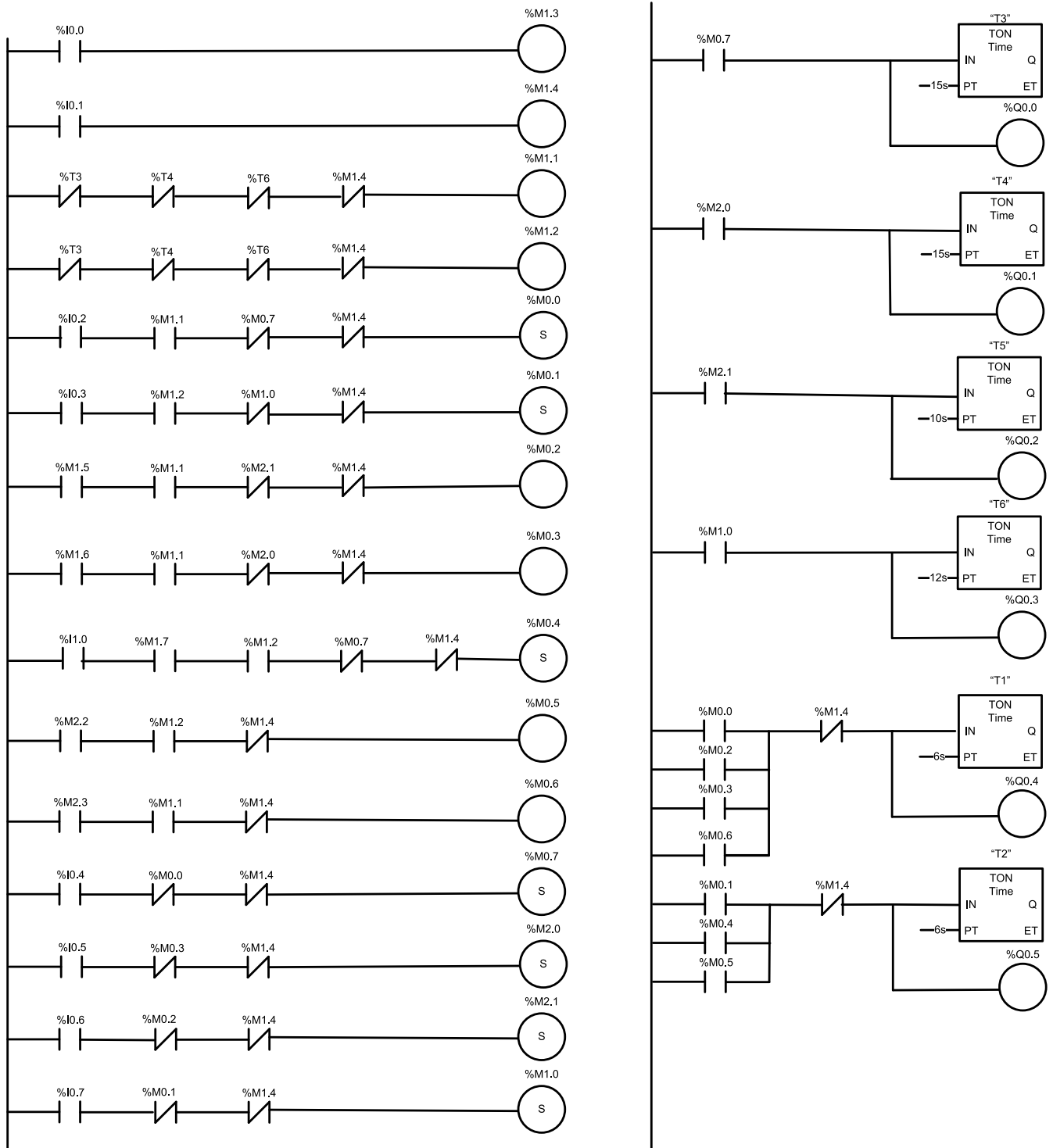


FIGURE 19. LD for the LDCROPN model presented in Figure 18.

is set on through  $t_{420B}$ . When component B is detected in the  $m_2$ , the unloading process of the  $r_1$  is set off through  $t_{420B}$ . The operation of the  $m_2$  is set on if component B is available in the  $m_2$ , i.e.,  $M(p_2) = c_{p_2}$ , and the  $r_1$ 's loading process duration has finished.

7. The unloading process of the  $r_1$  for component B is set on via  $t_{200B}$  when the processing time of the  $m_2$  has finished and the  $r_2$  is free, i.e.,  $M(p_r) = c_{t_2}$ . If the  $r_1$ 's loading process time is completed, the  $r_1$ 's unloading process is set off via  $t_{200B}$ .

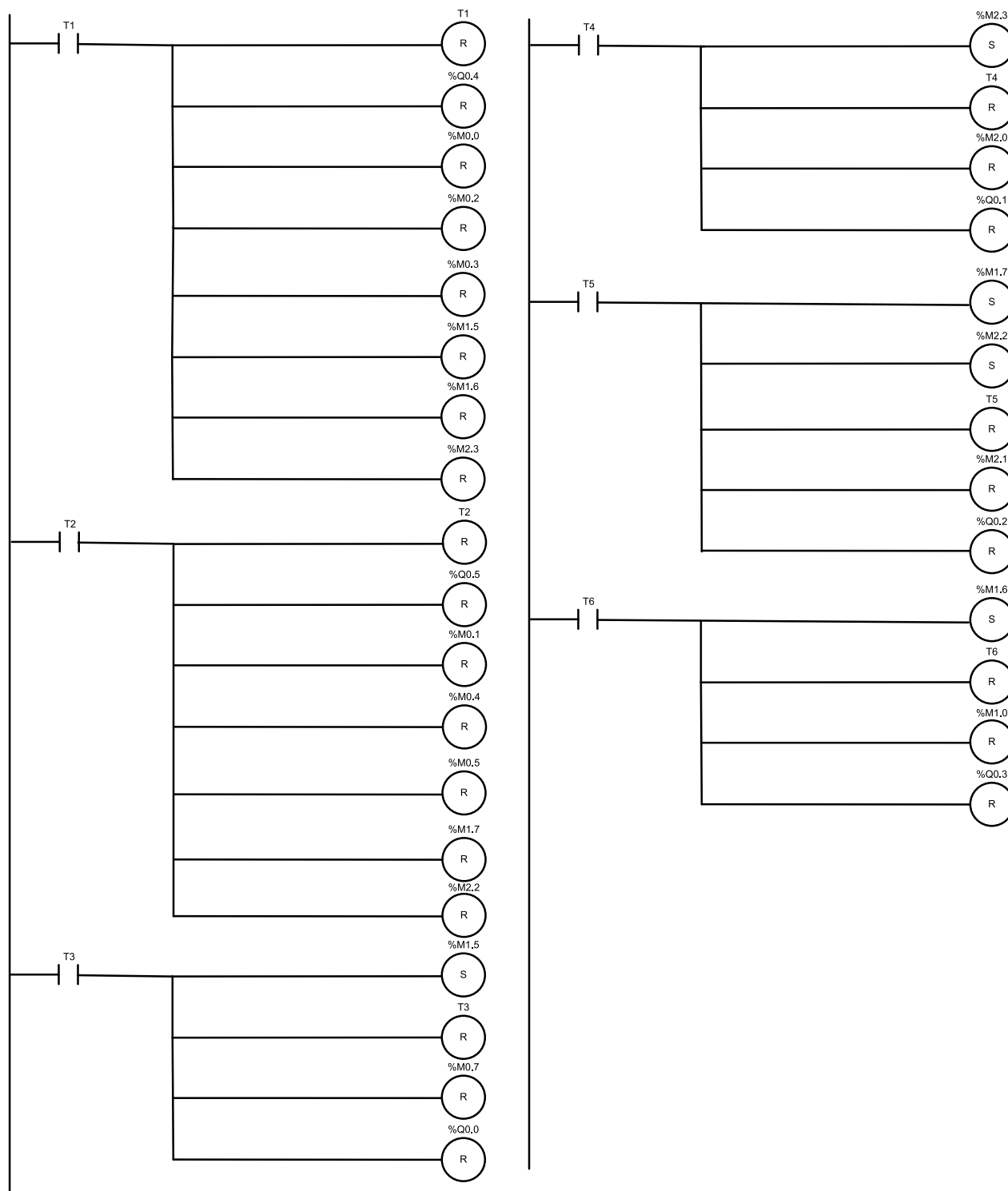


FIGURE 19. (Continued) LD for the LDCROPN model presented in Figure 18.

Figure 19 illustrates the ladder diagram of the LDCROPN model depicted in Figure 18.

**VI. CONCLUSION**

In this article, a new two-step implementation method for ladder diagrams for AMSs is proposed. First, a CROPN modeling method and control rules are applied to model

AMS operations and ensure that the AMS operates without deadlocks. Second, we propose an LDCROPN to translate the CROPN model into a LD. For the proposed LDCROPN model’s dynamic behavior, equations and constraints for firing transitions and marking places are provided for solving the difficulties of mark accumulation and the initiating condition of the LDCROPN structure.

Following is a summary of this paper's main advantages:

1. This work presents a novel approach for AMS to transform a CROPN into LDs that is prepared to handle any difficult AMS modification compared with the study in [5] and [38].
2. The LDs are constructed for validation using the PLC Fiddle simulator; the results demonstrate that the developed approach provides low-overhead computation and has a simpler structure in comparison to the researches in [33], [34], [35], [36] and [38].
3. A global transportation resource place with a simpler structure and polynomial complexity is developed to transport all component types in a CROPN, compared to current researches [38].

The main drawback of the presented methodology is that the generated LDCROPN is constrained to discrete events. Therefore, our future study will improve the current approach to include the continuous events in the LDCROPN.

### DATA AVAILABILITY STATEMENT

All the data is available in the manuscript.

### ACKNOWLEDGMENT

The authors would like to thank the Raytheon Chair for Systems Engineering for funding.

### REFERENCES

- [1] H. Kaid, A. Al-Ahmari, Z. Li, and R. Davidrajah, "Single controller-based colored Petri nets for deadlock control in automated manufacturing systems," *Processes*, vol. 8, no. 1, p. 21, Dec. 2019.
- [2] A. Al-Ahmari, H. Kaid, Z. Li, and R. Davidrajah, "Strict minimal siphon-based colored Petri net supervisor synthesis for automated manufacturing systems with unreliable resources," *IEEE Access*, vol. 8, pp. 22411–22424, 2020.
- [3] H. Kaid, A. Al-Ahmari, Z. Li, and W. Ameen, "Deadlock control and fault detection and treatment in reconfigurable manufacturing systems using colored resource-oriented Petri nets based on neural network," *IEEE Access*, vol. 9, pp. 84932–84947, 2021.
- [4] H. Kaid, A. Al-Ahmari, Z. Li, and R. Davidrajah, "Intelligent colored token Petri nets for modeling, control, and validation of dynamic changes in reconfigurable manufacturing systems," *Processes*, vol. 8, no. 3, p. 358, Mar. 2020.
- [5] H. Kaid, A. Al-Ahmari, and Z. Li, "Colored resource-oriented Petri net based ladder diagrams for PLC implementation in reconfigurable manufacturing systems," *IEEE Access*, vol. 8, pp. 217573–217591, 2020.
- [6] Z. Li and M. Zhou, *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*. London, U.K.: Springer, 2009.
- [7] X. Zan, Z. Wu, C. Guo, and Z. Yu, "A Pareto-based genetic algorithm for multi-objective scheduling of automated manufacturing systems," *Adv. Mech. Eng.*, vol. 12, no. 1, 2020, Art. no. 1687814019885294.
- [8] L. Li, F. Basile, and Z. Li, "An approach to improve permissiveness of supervisors for GMECs in time Petri net systems," *IEEE Trans. Autom. Control*, vol. 65, no. 1, pp. 237–251, Jan. 2020.
- [9] Y. Liu, K. Cai, and Z. Li, "On scalable supervisory control of multi-agent discrete-event systems," *Automatica*, vol. 108, Oct. 2019, Art. no. 108460.
- [10] Q. Chen, L. Yin, N. Wu, M. El-Meligy, M. Sharaf, and Z. Li, "Diagnosability of vector discrete-event systems using predicates," *IEEE Access*, vol. 7, pp. 147143–147155, 2019.
- [11] H. Kaid, A. Al-Ahmari, E. A. Nasr, A. Al-Shayea, A. K. Kamrani, M. A. Noman, and H. A. Mahmoud, "Petri net model based on neural network for deadlock control and fault detection and treatment in automated manufacturing systems," *IEEE Access*, vol. 8, pp. 103219–103235, 2020.
- [12] M. Zhao, "An integrated control method for designing non-blocking supervisors using Petri nets," *Adv. Mech. Eng.*, vol. 9, no. 6, 2017, Art. no. 1687814017700829.
- [13] M. P. Cabasino, A. Giua, and C. Seatzu, "Fault detection for discrete event systems using Petri nets with unobservable transitions," *Automatica*, vol. 46, no. 9, pp. 1531–1539, Sep. 2010.
- [14] M. P. Cabasino, A. Giua, M. Poggi, and C. Seatzu, "Discrete event diagnosis using labeled Petri nets. An application to manufacturing systems," *Control Eng. Pract.*, vol. 19, no. 9, pp. 989–1001, Sep. 2011.
- [15] Z. Ma, Z. Li, and A. Giua, "Computation of admissible marking sets in weighted synchronization-free Petri nets by dynamic programming," *IEEE Trans. Autom. Control*, vol. 65, no. 6, pp. 2662–2669, Jun. 2020.
- [16] C. Lagartinho-Oliveira, F. Moutinho, and L. Gomes, "GPGPU applied to support the construction of the state-space graphs of IOPT Petri net models," in *Proc. 45th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2019, pp. 5862–5867.
- [17] C. R. Vázquez, D. Gómez-Gutiérrez, and A. Ramírez-Tevino, "Observer design for linear hybrid systems with unknown inputs and Petri-net discrete dynamics," *Nonlinear Anal., Hybrid Syst.*, vol. 36, May 2020, Art. no. 100876.
- [18] J. A. V. Garnica, E. R. Beltrán, V. H. H. García, A. R. Treviño, and J. R. León, "Fault diagnosis for a Petri net class using reduced observable projections," in *Proc. IEEE Int. Autumn Meeting Power, Electron. Comput. (ROPEC)*, Nov. 2019, pp. 1–6.
- [19] A. Ramírez-Trevino, I. Rivera-Rangel, and E. López-Mellado, "Observability of discrete event systems modeled by interpreted Petri nets," *IEEE Trans. Robot. Autom.*, vol. 19, no. 4, pp. 557–565, Aug. 2003.
- [20] I. Rivera-Rangel, A. Ramírez-Treviño, and E. López-Mellado, "Building reduced Petri net models of discrete manufacturing systems," *Math. Comput. Model.*, vol. 41, nos. 8–9, pp. 923–937, Apr. 2005.
- [21] I. Grobelna, R. Wiśniewski, M. Grobelny, and M. Wiśniewska, "Design and verification of real-life processes with application of Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 11, pp. 2856–2869, Nov. 2017.
- [22] R. Wisniewski, I. Grobelna, and A. Karatkevich, "Determinism in cyber-physical systems specified by interpreted Petri nets," *Sensors*, vol. 20, no. 19, p. 5565, Sep. 2020.
- [23] R. Wiśniewski, Ł. Stefanowicz, A. Bukowiec, and J. Lipiński, "Theoretical aspects of Petri nets decomposition based on invariants and hypergraphs," in *Multimedia and Ubiquitous Engineering*. Berlin, Germany: Springer, 2014, pp. 371–376.
- [24] L. Blocks, "Functional block diagram," *Scanning*, vol. 1, p. 2, Oct. 1998.
- [25] G. Fen and W. Ning, "Transformation algorithm between ladder diagram and instruction list based on AOV digraph and binary tree," *J. Nanjing Univ. Aeronaut. Astronaut.*, vol. 38, no. 6, pp. 754–758, 2006.
- [26] V. Vyatkin, *Function Blocks for Embedded and Distributed Control Systems Design*, Standard IEC 61499, 2007.
- [27] J. Asensio, F. Ortuño, M. Damas, and H. Pomares, "Industrial automation programming environment with a new translation algorithm among IEC 61131–3 languages based on the TC6-XML scheme," *Int. J. Automat. Control Eng.*, vol. 2, no. 2, pp. 47–55, 2013.
- [28] A. T. Azar and S. Vaidyanathan, *Advances in Chaos Theory and Intelligent Control*, vol. 337. Cham, Switzerland: Springer, 2016.
- [29] K. Venkatesh, M. Zhou, and R. Caudill, "Evaluating the complexity of Petri nets and ladder logic diagrams for sequence controllers design in flexible automation," in *Proc. IEEE Symp. Emerg. Technol. Factory Autom. (ETFA)*, Nov. 1994, pp. 428–435.
- [30] K. Venkatesh, M. Zhou, and R. J. Caudill, "Comparing ladder logic diagrams and Petri nets for sequence controller design through a discrete manufacturing system," *IEEE Trans. Ind. Electron.*, vol. 41, no. 6, pp. 611–619, Dec. 1994.
- [31] A. Jones, M. Uzam, A. Khan, D. Karimzadgan, and S. Kenway, "A general methodology for converting Petri nets into ladder logic: The TPLL methodology," in *Proc. 5th Int. Conf. Comput. Integr. Manuf. Autom. Technol. (CIMAT)*, 1996, pp. 357–362.
- [32] A. H. Jones, M. Uzam, and N. Ajlouni, "Design of discrete event control systems for programmable logic controllers using T-timed Petri nets," in *Proc. IEEE Int. Symp. Comput. Aided Control Syst. Design*, 1996, pp. 212–217.
- [33] M. Uzam and A. Jones, "Design of ladder logic for an agile manufacturing system using TPLL," in *Proc. 1st Turkish Symp. Intell. Manuf. Syst. (IMS)*, Sakarya, Turkey, 1996, pp. 55–74.
- [34] M. Uzam, A. H. Jones, and N. Ajlouni, "Conversion of Petri net controllers for manufacturing systems into ladder logic diagrams," in *Proc. IEEE Conf. Emerg. Technol. Factory Autom. (ETFA)*, Nov. 1996, pp. 649–655.

- [35] M. Uzam and A. Jones, "Towards a unified methodology for converting coloured Petri net controllers into ladder logic using TPLL: Part I—Methodology," in *Proc. Int. Workshop Discrete Event Syst. (WODES)*, Edinburgh, U.K., 1996, pp. 178–183.
- [36] M. Uzam and A. Jones, "Toward a unified methodology for converting colored Petri nets TPLL: Part II—An application," in *Proc. Int. Workshop Discrete Event Syst. (WODES)*, 1996, pp. 314–319.
- [37] M. Uzam, A. Jones, and I. Yucel, "A rule-based methodology for supervisory control of discrete event systems modeled as automation Petri nets," *Int. J. Int. Cont. Syst.*, vol. 3, no. 3, pp. 297–325, 1999.
- [38] H. Kaid, A. Al-Ahmari, A. M. El-Tamimi, E. A. Nasr, and Z. Li, "Design and implementation of deadlock control for automated manufacturing systems," *South Afr. J. Ind. Eng.*, vol. 30, no. 1, pp. 1–23, 2019.
- [39] K. Feldmann, A. W. Colombo, C. Schnur, and T. Stockel, "Specification, design, and implementation of logic controllers based on colored Petri net models and the standard IEC 1131. I. Specification and design," *IEEE Trans. Control Syst. Technol.*, vol. 7, no. 6, pp. 657–665, Nov. 1999.
- [40] E. Park, T. Tilbury, and P. P. Khargonekar, "Control logic generation for machining systems using Petri net formalism," in *Proc. IEEE Int. Conf. Syst., Man Cybern. Cybern. Evolving Syst., Hum., Org., Complex Interact.*, Oct. 2000, pp. 3201–3206.
- [41] G. Frey, "Automatic implementation of Petri net based control algorithms on PLC," in *Proc. Amer. Control Conf. (ACC)*, 2000, pp. 2819–2823.
- [42] M. U. Borges and E. J. Lima, "Conversion methodologies from signal interpreted Petri nets to ladder diagram and C language in Arduino," *Int. J. Mech. Eng. Educ.*, vol. 46, no. 4, pp. 302–314, Oct. 2018.
- [43] S. Rezig, C. Ghorbel, Z. Achour, and N. Rezg, "PLC-based implementation of supervisory control for flexible manufacturing systems using theory of regions," *Int. J. Autom. Control*, vol. 13, no. 5, pp. 619–640, 2019.
- [44] S. G. Tzafestas, M. G. Pantelelis, and D. L. Kostis, "Design and implementation of a logic controller using Petri nets and ladder logic diagrams," *Syst. Anal. Model. Simul.*, vol. 42, no. 1, pp. 135–167, Jan. 2002.
- [45] J. Luo, Q. Zhang, X. Chen, and M. Zhou, "Modeling and race detection of ladder diagrams via ordinary Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 7, pp. 1166–1176, Jul. 2018.
- [46] G. Andrzejewski, W. Zając, K. Krzywicki, A. Karasiński, T. Królikowski, and B. Bałasz, "Implementation of an example of hierarchical Petri net (HPN) in LAD language in TIA portal," *Proc. Comput. Sci.*, vol. 192, pp. 3657–3666, 2021.
- [47] N. Wu and M. Zhou, *System Modeling and Control With Resource-Oriented Petri Nets*. New York, NY, USA: CRC Press, 2009.
- [48] N. Wu, "Necessary and sufficient conditions for deadlock-free operation in flexible manufacturing systems using a colored Petri net model," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 29, no. 2, pp. 192–204, May 1999.
- [49] N. Wu and M. Zhou, "Modeling and deadlock control of automated guided vehicle systems," *IEEE/ASME Trans. Mechatronics*, vol. 9, no. 1, pp. 50–57, Mar. 2004.
- [50] N. Wu and M. Zhou, "Modeling and deadlock avoidance of automated manufacturing systems with multiple automated guided vehicles," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 6, pp. 1193–1202, Dec. 2005.
- [51] N. Wu and M. Zhou, "Shortest routing of bidirectional automated guided vehicles avoiding deadlock and blocking," *IEEE/ASME Trans. Mechatronics*, vol. 12, no. 1, pp. 63–72, Feb. 2007.
- [52] N. Wu and M. Zhou, "Deadlock resolution in automated manufacturing systems with robots," *IEEE Trans. Autom. Sci. Eng.*, vol. 4, no. 3, pp. 474–480, Jul. 2007.
- [53] N. Wu and M. Zhou, "Avoiding deadlock and reducing starvation and blocking in automated manufacturing systems," *IEEE Trans. Robot. Autom.*, vol. 17, no. 5, pp. 658–669, Oct. 2001.
- [54] K. Xing, M. C. Zhou, H. Liu, and F. Tian, "Optimal Petri-net-based polynomial-complexity deadlock-avoidance policies for automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 39, no. 1, pp. 188–199, Jan. 2009.
- [55] N. Wu, M. Zhou, and G. Hu, "One-step look-ahead maximally permissive deadlock control of AMS by using Petri nets," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 1, pp. 1–23, Jan. 2013.
- [56] A. Al-Shayea, H. Kaid, A. Al-Ahmari, E. A. Nasr, A. K. Kamrani, and H. A. Mahmoud, "Colored resource-oriented Petri nets for deadlock control and reliability design of automated manufacturing systems," *IEEE Access*, vol. 9, pp. 125616–125627, 2021.
- [57] H. Chen, N. Wu, and M. Zhou, "A novel method for deadlock prevention of AMS by using resource-oriented Petri nets," *Inf. Sci.*, vol. 363, pp. 178–189, Oct. 2016.
- [58] N. Wu, M. Zhou, and Z. Li, "Resource-oriented Petri net for deadlock avoidance in flexible assembly systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 1, pp. 56–69, Jan. 2008.
- [59] N. Wu, "Avoiding deadlocks in automated manufacturing systems with shared material handling system," in *Proc. Int. Conf. Robot. Autom.*, 1997, pp. 2427–2432.
- [60] N. Wu and M. Zhou, "Deadlock avoidance in semiconductor track systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2002, pp. 193–198.
- [61] Z. Xiang, "Deadlock avoidance of flexible manufacturing systems by colored resource-oriented Petri nets with novel colored capacity," in *Proc. EasyChair*, 2020, pp. 2314–2516.
- [62] H. F. Chen, N. Q. Wu, Z. W. Li, and T. Qu, "On a maximally permissive deadlock prevention policy for automated manufacturing systems by using resource-oriented Petri nets," *ISA Trans.*, vol. 89, pp. 67–76, Jun. 2019.
- [63] N. Wu and M. Zhou, "Process vs resource-oriented Petri net modeling of automated manufacturing systems," *Asian J. Control*, vol. 12, no. 3, pp. 267–280, Apr. 2010.
- [64] N. Wu, M. Zhou, and G. Hu, "On Petri net modeling of automated manufacturing systems," in *Proc. IEEE Int. Conf. Netw., Sens. Control*, Apr. 2007, pp. 228–233.
- [65] H. Chen, N. Wu, and M. Zhou, "Resource-oriented Petri net-based approach to deadlock prevention of AMSS," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2015, pp. 515–520.
- [66] N. Wu and M. Zhou, "Resource-oriented Petri nets in deadlock avoidance of AGV systems," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2001, pp. 64–69.
- [67] H. Chen, N. Wu, Z. Li, T. Qu, and H. Xiao, "Liveness of disjunctive and strict single-type automated manufacturing system: An ROPN approach," *IEEE Access*, vol. 7, pp. 17760–17771, 2019.
- [68] H. Kaid, A. Al-Ahmari, and K. N. Alqahtani, "Fault detection, diagnostics, and treatment in automated manufacturing systems using Internet of Things and colored Petri nets," *Machines*, vol. 11, no. 2, p. 173, Jan. 2023.
- [69] M. N. Q. Wu and M. C. Zhou, "Intelligent token Petri nets for modelling and control of reconfigurable automated manufacturing systems with dynamical changes," *Trans. Inst. Meas. Control*, vol. 33, no. 1, pp. 9–29, May 2011.
- [70] J. C. Q. Quezada, E. F. García, J. M. Marín, J. B. López, and V. Q. Aguilar, "Ladder diagram Petri nets: Discrete event systems," in *Petri Nets in Science and Engineering*. London, U.K.: IntechOpen, 2018.
- [71] *PLC Fiddle*. Accessed: Nov. 1, 2022. [Online]. Available: <https://www.plcfiddle.com/>
- [72] R. Aurachman, "PLC fiddle software review as an instrumentation and automation learning tool," *J. Phys., Conf. Ser.*, vol. 1825, no. 1, Feb. 2021, Art. no. 012040.
- [73] M. M. Khan, O. Shawareb, R. Palaniappan, V. Vijejan, F. G. Nabi, A. Shawareb, and N. K. Khan, "Simulation of PLC ladder logic programming for an automated glass bottle molding and refilling plant," in *Proc. 4th Smart Cities Symp. (SCS)*. Bahrain: Institution of Engineering and Technology (IET), Nov. 2021.
- [74] A. Kurniawan, J. Prananda, E. S. Koenhardono, S. Sarwito, I. R. Kusuma, and A. A. Masroeri, "Pelatihan dasar programmable logic controller (PLC) berbasis daring menggunakan PLC fiddle untuk guru SMK di surabaya," *Sewagati*, vol. 5, no. 3, pp. 278–285, Aug. 2021.
- [75] A. Kurniawan, J. Prananda, S. Sarwito, E. S. Koenhardono, I. R. Kusuma, and A. A. Masroeri, "Pembuatan modul tutorial dasar ladder diagram programmable logic controller dalam jejaring," *Jurnal Abdimas PHB, Jurnal Pengabdian Masyarakat Progresif Humanis Brainstorming*, vol. 4, no. 2, pp. 140–146, 2021.

• • •