Colored Resource-Oriented Petri Net Based Ladder Diagrams for PLC Implementation in Reconfigurable Manufacturing Systems

HUSAM KAID, ABDULRAHMAN AL-AHMARI, and ZHIWU LI

ABSTRACT A typical example of discrete event systems (DESs) is a reconfigurable manufacturing system (RMS) whose can be changed and become more complicated during its operation. Thus, an effective tool needs to be developed to create both high- and low-level DES implementation. The ladder diagram (LD) is a common programming method for controlling DESs. Petri nets (PNs) represent the most important graphical and mathematical tool that can be used to provide an integrated solution for the design, modeling, analysis, control, and implementation of DESs. Various types of PN-based LDs are developed for DESs, which have high structural complexity of the LDs. Therefore, an approach that can facilitate the elimination of the LDs’ structural complexity needs to be developed. The main objective of this paper is to develop a two-step approach for LDs implementation for RMSs with dynamic changes. First, a colored resource-oriented Petri net (CROPN)-based algorithm is proposed for a rapid and valid configurations of an RMS. Second, a ladder diagram colored resource-oriented Petri net (LDCROPN) is developed to convert the CROPN into an LD. The developed approach is evaluated using examples in the literature. Moreover, the results demonstrate the effectiveness of the developed approach for LD implementation under RMS specification changes.

INDEX TERMS Reconfigurable manufacturing system, modeling, simulation, colored Petri net, ladder diagram.
PLCs have become the cornerstone in executing the automation tasks. According to IEC 61131-3, the International Electrotechnical Commission has defined five PLC languages: the structured text (ST) [21], the ladder diagram (LD) [22], the functional block chart (FBD) [23], the instruction list (IL) [24], and the Grafcet or sequential function chart (SFC) [25]. Among them, the ladder diagram is the most popular PLC language, primarily because it can be heuristically performed. PLC programs can be easily written for simple systems using ladder diagram. However, the consideration of multiple systems complicates the problem. Indeed, if the system is individual in type and multiproduct in heuristic approaches, writing down the ladder logic programs is impossible [26], [27]. The most successful solutions for this problem are the use of PNs and the heuristic transformation of PN into LD.

Various methods developed using PNs in LD have been employed in the modeling, simulation, analysis, and control of transformation algorithms [28]–[42]. Using the token pass ladder logic (TPLL), Jones et al. [28], [29] design a concept for the conversion of PN into LD. In this method, the mechanism for primary control in PNs is considered as the token. The control of the program is performed using flags or auxiliary relay. The technique is extended by using delay timers in various contexts to include timed-transition PNs. The basic structure and related ladder logic equivalents of the timed-transition PNs are established. However, the number of the basic elements of PNs, which has significantly increased, has rendered the implementation of the PN difficult.

Uzam et al. [30]–[33] present the TPLL method for the conversion of the colored PN controllers into an LD in both the methodology and application. This method can also provide a simple mapping between the basic information on sequencing and programming steps. The basic structures and related LD equivalents of the timed-transition PNs are described. In addition, a heuristic PN controller is developed for DESS and a structured LD methodology. The method is also extended to include the formal structures of actuators and sensors. Uzam et al. have listed a set of additional features for ordinary PNs, thus enabling the sensor readings to be inserted during transitions and the control of the impulse and level of actuators, which results in an automation PN (APN). The work in [34] proposes the technique of bottom-up synthesis including the structure of a system controlled APN reachability graph. A controller of the system that is obtained by using the token pass marking (TPM) rules and the uncontrolled APN model is associated with the prohibited state requirements. The TPM rules are directly derived from the prohibited state requirements. In addition, the TPM rules are implemented using a mixture of enabling and inhibitor arcs that connect the appropriate places to the transitions of the corresponding controllers.

In Feldmann [36], logical control structures are designed and implemented to enable a combination of the features of traditional methods for designing the PLC code and to avoid drawbacks such as the lack of relevant validation. In combination with some extensions, they first present the ordered colored PNs (OCPNs) as a logic controller modeling tool. The OCPN’s logical control system model consists of two main structures: a coordination system model, and a submodel for some elements of the first model, obtained following a transitional refinement. Then, a structural analysis of the net will conduct the validation of both the model and the logic structure control requirements. They then propose an automated technique for creating a PLC code from a validated OCPN textual description. Park et al. [37] develop a method based on Petri nets for the development of control logic for machining systems. Two Petri nets are used to represent the control logic: one for mode decision and the other for the sequence. The Petri net control logic for each process can be created by replacing the I / O for the corresponding sensors, actuators, and operator commands from the operation module templates. The proposed Petri net control logic is still under study and it is anticipated to automatically generate the verified IEC1131–3 control logic code. Frey [38] defines a special type of Petri nets, which includes more information, named the signal interpreted Petri net (SIPN) and develops a transformation technique from SIPN to LD in which programmable logic controller input and output signals are defined for each transition and event. In addition, for each place there is a Boolean function value assigned. In the study of Borges and Lima the SIPN is used to design two educational techniques in laboratory classes: to transform the SIPN into ladder diagram for PLC and to transform the SIPN into C language for Arduino microcontroller. In [40], the JAVA application is developed for flexible manufacturing systems (FMS) to model a Petri net supervisor based on the signal interpreted Petri net and theory of regions, and is then converted into the corresponding LIST code approved by the PLC. Then, the obtained code is verified using the SIEMENS STEP7 compiler.

In [41], a technique is developed for creating the related LD for the PLC using Petri net models, which has been demonstrated to be efficient by the application to a sequence controller of a small-scale robotic system. The experimentation is carried out with Allen Bradley PLC for programming in an educational system. The developed method considers the LD rows to be two parts the “conditional part” and the “action part” as shown in the PLC’s memory, and then assigns a bit for a place in the conditional part of the Petri net, and another bit for a transition in the action part of the Petri net. Luo et al. [42] introduce an ordinary Petri net-based method for modeling and race-detection issues of programs for PLCs. In the first step, a PLC program is generated by a graph to reflect contacts and coils. The second step is to convert this graph into an ordinary Petri net using a conversion algorithm. Third, a technique is developed to determine whether a race occurs in a system using a reachability graph approach, to locate a race by adding the race path, and to fix a race by evaluating its subnet.

Feio et al. [43] propose a set of rules for the conversion of input-output place-transition (IOPT) Petri net
models \[44\], \[45\] into an LD. These rules are integrated in a programming tool that reads the IOPT model represented in the PNML format, and creates the LD. Kaid et al. \[35\] suggest a Petri net-based methodology for the prevention of deadlocks and create PLC codes for an automated manufacturing system (AMS). First, the controlled Petri net model is constructed using a deadlock-prevention method based on strict minimal siphons. Second, the controlled Petri net model is transformed into an automation-controlled Petri net model that is further transformed into a controlled token-passing logic (CTPL) model. The obtained CTPL model is then utilized to create the LDs for the automated manufacturing system (AMS). The suggested methodology is verified and evaluated using a real-world system.

Wu and Zhou have been active in the field of deadlock avoidance by using the formalism of resource-oriented Petri nets (ROPN) \[46\], \[47\]. The ROPN model is developed to describe the dynamic resources competition by parts in an AMS \[46\], \[48\]–\[64\]. In an ROPN, a type of resources is modeled by only one place. It is a type of colored Petri nets with a finite capacity for each place and there is a one-to-one mapping between the places in the AMS model and the resources in the plant, leading to a very compact Petri net model. Colors are used to differentiate between various jobs and the progress of their processing in the AMS \[46\], \[61\], \[63\]. Tokens and directed edges are marked with ROPN colors and the token color can change as it transfers from one place to another. In addition, a deadlock can appear in ROPN only in a circuit and the circuits’ number is limited.

Wu \[47\] develops colored resource-oriented Petri nets (CROPNs), and necessary and sufficient conditions under which a maximally permissive control law is obtained to ensure the deadlock-free operations. In the CROPN, the deadlock freedom of its interactive subnets is verified by the control law. If the control law is applied on each interactive subnet, then the overall CROPN of the system can be shown to be live. Deadlock starvation and blocking have a significant influence on the utilization of AMSs. To resolve this problem, Wu and Zhou \[50\] develop a novel deadlock-control policy based on CROPN to avoid deadlock completely, which decreases starvation and blocking problems through releasing a number of appropriate jobs into the system and controlling the order of the utilization of a resource based on state information embedded in the system model.

Wu and Zhou \[55\] present a CROPN modeling approach to handle deadlock avoidance and conflict resolution in automated guided vehicles (AGV) systems with unidirectional and bidirectional routes. Wu and Zhou \[54\] use the CROPN modeling paradigm to model the AGV system and part processing processes. Then, the two models are integrated through macro transitions. Based on the integrated model, a deadlock-avoidance policy is developed. In addition, Wu and Zhou \[61\] propose a CROPN based on an approach to minimize the bidirectional AGV traveling time, while both deadlock and blocking are avoided. The proposed approach can provide better solutions compared with other techniques. In addition, its performance evaluations are performed by experimental studies.

Wu and Zhou \[63\] propose a novel way for modeling robots in an automated manufacturing system using the CROPN modeling approach. A modeling paradigm leads to the appealing and interesting result, i.e., robots have no contribution to deadlocks. A new deadlock control policy is proposed, which is more permissive than the existing ones when robots are treated as both material handling devices and buffers. Wu et al. \[49\] propose ROPN to solve the problem of deadlock avoidance in flexible assembly systems (FAS). Based on ROPN, a deadlock control policy is proposed. The proposed control policy is shown to be better than the one in the literature and to be computationally efficient.

The work in \[65\] is extended by Wu et al. \[64\]. Wu and Zhou consider sequential problem allocation of resources in AMS, where each processing step requires a single type of resource. The necessary and sufficient conditions are proposed under which a one-step look-ahead maximally permissive deadlock-avoidance policy exists. In addition, a method is established to propose such a policy if it exists. Note that the system in \[64\] is modeled by the ROPN, and the system in \[65\] is modeled by traditional process-oriented Petri nets (POPNs). The ROPN is considered to be more effective and compact for deadlock avoidance than the POPN \[57\].

The works in Chen et al. \[59\] and \[48\] use the ROPN and the theory of regions to design a deadlock prevention controller for automated manufacturing systems. The relationship between structural properties and bad markings in an ROPN is demonstrated. Then, an efficient deadlock prevention approach is developed. Next, a controller is obtained by simple calculation and only one monitor is required for each strongly connected subnet in an ROPN and there is no redundant monitor at all. Thus, it is structurally very simple. Chen et al. \[56\] use an ROPN modeling approach to model a class of AMS to overcome the deadlock control problem. Several types of illegal markings are explored, which can be prevented by structural analysis of ROPN models. For these markings, a deadlock prevention policy is developed for illegal markings in an algebraic method without solving a notorious integer linear programming (ILP) problem. Approaches based on linear programming (LP) are designed to prevent other markings optimally, instead of ILP. Chen et al. \[62\] use an ROPN modeling approach to model the disjunctive and strict single-type automated manufacturing systems (DS-AMS) in which multiple units of one resource type are needed to support a processing stage. They propose effective techniques to achieve the liveness of an ROPN of a DS-AMS by only enforcing some specific transitions to be controlled using its structural properties.

It is well-known that various kinds of PN-based LDs are developed for DESs. However, these approaches have disadvantages, including their growing complexity when a system is large and the complication that arise when checking and changing the model completely. Therefore, an approach that can reduce the structural complexity of...
LDs needs to be developed. Thus, this paper is an extension of the work in [2] and aims to propose a novel two-step approach for the implementation of LDs for RMSs with dynamic changes. First, a colored resource-oriented Petri net (CROPN) modeling approach is proposed for a rapid and valid reconfiguration of RMS, and necessary and sufficient conditions under which a maximally permissive control law is obtained to ensure the deadlock-free of RMS operations. Second, a ladder diagram colored resource-oriented Petri net (LDCROPN) is developed to convert the CROPN model into an LD.

The main contributions are summarized as the following aspects.

1. A new method for RMS is developed in this paper for the first time to convert a CROPN into LDs, which has the ability to handle any complex RMS change.
2. A sufficient and necessary condition for liveness of CROPN is presented, which is more succinct than that in [2].
3. The GPenSIM code is developed for simulation, performance comparison, and validation; the results show that the proposed method has accurate results and low-overhead computation compared with the studies in [1], [66]–[68].
4. A single common transportation resource place in a CROPN is developed to transport all part types, which has polynomial complexity and has a simpler structure compared with the existing studies.

The rest of the paper is organized as follows. In Section II, the synthesis of CROPN is presented. In Section III, how to model the configurations of RMSs by the CROPN is described. The quantitative and qualitative properties of the CROPN are presented in Section IV. The LDCROPN is defined in Section V, and a conversion algorithm for CROPN into LD is presented. Section VI provides the conclusions, advantages, limitations, and future work of this paper.

II. COLORED RESOURCE-ORIENTED PETRI NET

A resource-oriented Petri net (ROPN) considers a part production operation as a part visiting the resources one by one based on its specified processing paths. In ROPN, machines and their buffers are represented as the same type resource, called H-resources [46], [49], [51], [57]. Each resource is designed by a single place. The production operations of the part are designed by the routes on which the parts visit the resources. When each part production operation is designed in an effectively way, the model is very compact, and some useful structural features can be demonstrated. In addition, the ROPN treats the material handling devices as another type of resources, called G-resources [46], [49], [51], [57]. Due to the limited resources in manufacturing systems, a finite capacity Petri net is used with its corresponding transition enabling and firing rules. In an ROPN, a transition models that a part is moved from a resource to another resource. Two steps can be used to model the RMS by the ROPN: (1) model the part production operations irrespective of the material transportation, and (2) model the material handling processes based on step 1 model.

Definition 1: Let $N = (P, T, I, O, C, K, M_0)$ be a finite-capacity colored resource-oriented Petri net (CROPN), where

1. $P = \{p_0\} \cup \{p_t\} \cup P_R$, where $p_0$ is a common idle place, indicating a load/unload station, $p_t$ is a common transportation resource, indicating the system’s transportation resources, and $P_R = \{p_1, p_2, p_2, \ldots, p_m\}, m > 0$, is a finite set of resource places in the system.
2. $T = \{t_1, t_2, \ldots, t_n\}, n > 0$, is a finite set of transitions with $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.
3. $C(p)$ and $C(t)$ are the sets of colors associated with place $p_i \in P$ and transition $t_j \in T$, respectively, where
   a. $C(p_i) = \{a_{i1}, a_{i2}, \ldots, a_{in}\}, u_i = |C(p_i)|$
   b. $C(t_j) = \{b_{j1}, b_{j2}, \ldots, b_{jn}\}, v_j = |C(t_j)|$
4. $I(p, t); C(p) \times C(t) \rightarrow \text{IN}$ and $O(p, t); C(p) \times C(t) \rightarrow \text{IN}$ are input and output functions, respectively, where $\text{IN} = \{0, 1, 2, \ldots\}$ is the set of nonnegative integers.
5. $K : P \rightarrow \text{IN}$ is the function of capacity that assigns to each place $p$ the maximal number of tokens $K(p)$.
6. $M : P \rightarrow \text{IN}$ is the function of the marking of $N$ that assigns to each place $p_i$ the number of the tokens. $M = (M(p_1), M(p_2), \ldots, M(p_m))^T$. $M(p_i)$ is the number of the tokens in $p_i$ irrespective of the tokens colors, and $M(p_i, a_{ij})$ is the number of tokens with color $a_{ij}$ in $p_i$. $M_0$ represents the initial marking of $N$.

Definition 2: Let $N = (P, T, I, O, C, K, M_0)$ be a CROPN and nodes $p, t \in P \cup T$ are a place and a transition in $N$, respectively. Let $F = I \cup O$, and the preset (postset) of $p$ is the set of all input (output) transitions of $p$, i.e., $p^* = \{t \in T | (t, p) \in F\}$ ($p^* = \{t \in T | (p, t) \in F\}$). The preset (postset) of $t$ is the set of all input (output) places of $t$, i.e., $t^* = \{p \in P | (p, t) \in F\}$ ($t^* = \{p \in P | (t, p) \in F\}$).

Definition 3: Let $N = (P, T, I, O, C, K, M_0)$ be a CROPN. $N$ is an ordinary CROPN if $p_i \in P, t_j \in T, \forall(p_i, t_j) \in F$, and $I(p_i, t_j) = 1$.

Definition 4: Let $N = (P, T, I, O, C, K, M_0)$ be a CROPN. $N$ is a weighted CROPN if $\exists p_i \in P, \exists t_j \in T, (p_i, t_j) \in F$, and $I(p_i, t_j) > 1$.

Definition 5: Let $N = (P, T, I, O, C, K, M_0)$ be a CROPN. $N$ is acceptably marked if (1) $M_0(p, a_{ij}) \geq 1$, (2) $M_0(p, a_{ij}) \geq 1$, and (3) $M_0(p_i, a_{ij}) = 0, \forall p_i \in P_R$.

Definition 6: Let $N = (P, T, I, O, C, K, M_0)$ be a CROPN. $N$ has a self-loop if for all $p_i, t_j \in P \cup T; I(p_i, t_j)(a_{ih}, b_{jk}) > 0$ implies that $O(p_i, t_j)(a_{ih}, b_{jk}) > 0$.

Definition 7: Let $N = (P, T, I, O, C, K, M_0)$ be a CROPN. $N$ is self-loop free if for all $p, t \in P \cup T; I(p, t)(a_{ih}, b_{jk}) > 0$ implies that $O(p, t)(a_{ih}, b_{jk}) = 0$.

Definition 8 [46]: Let $N = (P, T, I, O, C, K, M_0)$ be a CROPN. A transition $t_j \in T$ is enabled with respect to color $b_{jk}$ at marking $M$ if

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

\[ K(p_i) \geq M(p_i, a_{ib}) - I(p_i, t_j)(a_{ijh}, b_{jk}) + O(p_i, t_j)(a_{ijh}, b_{jk}), \]

\[ \forall p_i \in P, \forall p_t \in \mathbb{T}_t^*, \text{ and } a_{ijh} \in C(p_i) \]  

(2)

In Definition 8, \( t_j \) is process-enabled if condition (1) is satisfied and it is resource-enabled if condition (2) is satisfied. Therefore, \( t_j \) is said to be enabled if it is both process and resource-enabled.

**Definition 9** [46]: Let \( N = (P, T, I, O, C, K, M_o) \) be a CROPN. The marking \( M' \) resulting from the firing of an enabled transition \( t \) at marking \( M \) with respect to color \( b_{jk} \) is expressed by \( M(t)M' \) and can be updated from \( M \) to \( M' \) as follows:

\[ M'(p_i, a_{ib}) = M(p_i, a_{ib}) - I(p_i, t_j)(a_{ijh}, b_{jk}) + O(p_i, t_j)(a_{ijh}, b_{jk}), \]

\[ \forall p_i \in P \text{ and } a_{ijh} \in C(p_i) \]  

(3)

**Definition 10**: Let \( N = (P, T, I, O, C, K, M_o) \) be a CROPN. \( R(N, M) \) is a set of reachable markings from \( M \) in \( N \), which is denoted by vertices and arcs; vertices reflect markings that are labeled with \( M_k \) and arcs reflect firing of transitions that are labeled with \( t \). If \( t \) fires, then there is an arc from marking \( M_k \) to marking \( M_l \) and \( M_l \) is reached.

**Definition 11**: Let \( N = (P, T, I, O, C, K, M_o) \) be a CROPN. \( A \) is called the incidence matrix of net \( N \), where \( A \) is a \( m \times n \) matrix with \( A = [W_{ij}] = O(p_i, t_j) - I(p_i, t_j) \), where \( i = 1, 2, \ldots, m \), and \( j = 1, 2, \ldots, n \).

**Definition 12**: Let \( N = (P, T, I, O, C, K, M_o) \) be a CROPN. Let \( h \) be a nonempty set of colors with cardinality \( |H| = h \). Let \( X \) be a function of a set of places with respect to \( h \), expressed on \( p \) as \( X(p) : H \times C(p) \rightarrow \mathbb{Z} \), where \( \mathbb{Z} = \{-2, -1, 0, 1, 2, \ldots\} \).

**Definition 13**: Let \( N = (P, T, I, O, C, K, M_o) \) be a CROPN. Let \( X = [X_1, X_2, \ldots, X_m]^T \) be a weighted matrix that represents a set of places, where \( X_i = 1, 2, \ldots, m \), is a matrix of integers of dimension \( h \times u_i \). \( X \) is said to be a Place invariant of a CROPN if

\[ A^TX = 0 \]  

(4)

**Definition 14**: Let \( N = (P, T, I, O, C, K, M_o) \) be a CROPN. Let \( M(C) = (M(p_1, a_{11}), M(p_1, a_{12}), \ldots, M(p_1, a_{1u}), \ldots, M(p_2, a_{21}), \ldots, M(p_m, a_{m1}), \ldots, M(p_m, a_{mu}))^T \). If \( X \) is a Place invariant of a CROPN, then we have

\[ M(C)^TX = M_0(C)^TX, \forall M \in R(N, M_0) \]  

(5)

**Definition 15**: Let \( N = (P, T, I, O, C, K, M_o) \) be a CROPN. Let \( y = [y_1, y_2, \ldots, y_m] \) be a positive integer vector. The CROPN is conservative if

\[ M^Ty = M_0^Ty, \forall M \in R(N, M_0) \]  

(6)

**Definition 16**: Let \( N = (P, T, I, O, C, K, M_o) \) be a CROPN. \( N \) is said to be bound if there exists \( q \in \mathbb{N} \), \( \forall M \in R(N, M_0), \forall p_t \in P, M(p_t, a_{ij}) \leq q \).

**Definition 17**: Let \( N = (P, T, I, O, C, K, M_o) \) be a CROPN. \( N \) is called safe if \( \forall M \in R(N, M_0), \forall p_t \in P, M(p_t, a_{ij}) \leq 1. (N, M_0) \) is \( q \)-safe if it is \( q \)-bounded.

**Figure 1. (a) Example of an AMS and (b) The sequence of operation.**

**Figure 2. The CROPN model of system presented in Figure 1.**

**Definition 18**: Let \( N = (P, T, I, O, C, K, M_o) \) be a CROPN. A marking \( M_0 \) is called reversible if for each marking \( M' \in R(N, M_0) \), \( M_0 \) is reachable from \( M' \).

**Definition 19**: Let \( N = (P, T, I, O, C, K, M_o, OR) \) be a CROPN, where \( OR = \{OR_1, OR_2, OR_3, \ldots, OR_β \} \) are all feasible routes and can be produced in \( N \) for all part types \( β \). \( OR_i \) is \( R_o \rightarrow R_{i1} \rightarrow R_{i2} \rightarrow \ldots \rightarrow R_{iri} \rightarrow R_o \), \( r_i = \{1, 2, 3, \ldots\} \). \( R_o \) is called an infinite capacity load/unload station of \( N \), and \( R_i(i \neq 0) \) is called a machine or a buffer that is called an resource. The route of the operation begins at \( R_o \) and finishes at \( R_o \). For each part, in the presence of more than one route of operation such that \( R_{i1} \rightarrow R_{i(j\neq j)} \), a path from \( R_{i1} \) to \( R_{ij} \) is said to exist. If the part moves from \( R_{i1} \) to \( R_{ij} \), a material handling device, such as a robot or AGV, is used to move the part and is called a transportation resource.

Based on the operation routes and transportation requirements of the system and colored Petri net [1], the design of the CROPN is represented as follows:

Consider an example of an automated manufacturing system presented in Figure 1(a) to demonstrate the formulation of a CROPN. The system includes a machine (M1) and a robot (Ro1). M1 (Ro1) produces (holds) one part at a time. For loading/unloading parts, buffers are allocated into the system. Furthermore, the system is used to process part type A and the process sequence of part A is displayed in Figure 1(b). The transportation and operation resources model is displayed in Figure 2. For operation resource, \( p_1 \) models M1, and
the part A operation sequence is $p_0 \to t_{010A} \to p_1 \to t_{100A} \to p_0$. For transportation resource, place $p_0$ models $R_0$. To model the transportation processes of part types, $R_{O1}$ is required for loading part A to $p_1$ by transition $t_{010A}$. $R_{O1}$ is required for unloading part A from $p_1$ by transition $t_{100A}$. These processes can be conducted by place $p_r$. Therefore, place $p_r$ and arcs $(p_r, t_{010A})$, $(t_{010A}, p_r)$, $(p_r, t_{100A})$, and $(t_{100A}, p_r)$ are added into the model. Then, to define the initial marking of the developed model, add colored token into place $p_0$ such that $M_0(p_0) = \{c_{p0}\}$, where $c_{p0}$ denotes part type $A$; add colored token into place $p_r$ such that $M_0(p_r) = \{c_{11}\}$, where $c_{11}$ denotes transportation resource $R_{O1}$ in the system. The model presented in Figure 2 demonstrates the behavior of the CROPN model. When the requirements in Definition 8 are met, we may say that $t$ is enabled by $c_{pj}$ and $c_{ri}$. The CROPN marking can be modified in the following way by Definition 9. If transition $t_{010A}$ fires, it chooses only one colored token $c_{p1}$ and one colored token $c_{11}$ from input places $p_0$ and $p_r$, respectively. If $t_{010A}$ fires, it disposes only one colored token $c_{p1}$ and one colored token $c_{11}$ from input places $p_1$ and $p_r$, respectively. Finally, if transition $t_{100A}$ fires, it chooses only one colored token $c_{p1}$ and one colored token $c_{11}$ from input places $p_1$ and $p_r$, respectively. If $t_{100A}$ fires, it disposes only one token with color $c_{p1}$ and one colored token $c_{11}$ to output places $p_0$ and $p_r$, respectively.

III. DYNAMIC CHANGES DESIGNING BY CROPN

A. ADDING A NEW MACHINE

Suppose a new machine M2 is added to the system presented in Figure 1(a) to process part A after M1, and a robot is required to load/unload a part from/to M2. The part A operation sequence is stated as load/unload station $\rightarrow$ M1$\rightarrow$ M2$\rightarrow$ load/unload station. To model the new machine with the use of the synthesis procedure of Algorithm 1, new transitions and places can be inserted based on the original CROPN. Reconsider the CROPN model presented in Figure 2(b). $R_{O1}$ is required to transport part A between the load/unload station and M1; to move part A between M1 and M2; and to transport part A between M2 and load/unload station. Therefore, consider the following procedures in order to model the reconfigured system:

1. Add operation route of part A as: $p_0 \to t_{010A} \to p_1 \to t_{120A} \to p_2 \to t_{200A} \to p_0$.
2. Add the transportation operation of part A. $R_{O1}$ is required for the transportation of part A to $p_2$ by transition $t_{120A}$; these operations can be conducted by place $p_r$.
3. Add the transitions $t_{120A}$ and $t_{200A}$ to the model.

The reconfigured model by using the CROPN is presented in Figure 3.

B. ADDING A NEW PRODUCT

Assume a new product part B is added to the system presented in Figure 3, which means adding a new operation path and modifying CROPN structure. The part B operation route is designed by load/unload station $\rightarrow$ M1 $\rightarrow$ load/unload station. To transport part B between the load/unload station and M1, and to transport part B between M1 and the load/unload station, $R_{O1}$ is required. To model the newly added product by using the procedure of the CROPN synthesis, consider the following procedures:

1. Add a new route: $p_0 \to t_{010B} \to p_1 \to t_{100C} \to p_0$.
2. Add the transportation operation of part B. $R_{O1}$ is required for loading part B to $p_1$ by transition $t_{010B}$, and $R_{O1}$ is required for unloading part B from $p_1$ by transition $t_{100B}$.
3. Add the transitions $t_{010B}$ and $t_{100B}$ to the model.
4. Set the initial colored token $c_{p2}$ into place $p_0$ that denotes part B.

The reconfigured model by using the CROPN is presented in Figure 4.
C. REWORK

Suppose an inspection machine M3 is added to the system presented in Figure 4. Part B can be inspected after some of its operations have been completed. If it is produced accurately without defects, the system continues based on its original operation sequence. Otherwise, if defects exist in part B, rework is required. Note that the reworked part production operations can be modeled precisely and easily with alternative rework sequences. As a result, there are two possible sequences for part B: (1) Load/unload station → M1 → M3 → load/unload station or (2) Load/unload station → M1 → M3 → M1 → M3 → load/unload station.

To model the rework operation for part B by using the procedure of the CROPN synthesis, consider the following procedures:

1) Add two operations sequences as route 1: $p_0 \rightarrow t_{010B} \rightarrow p_1 \rightarrow t_{130B} \rightarrow p_3 \rightarrow t_{300B} \rightarrow p_0$ or route 2: $p_0 \rightarrow t_{010B} \rightarrow p_1 \rightarrow t_{130B} \rightarrow p_3 \rightarrow t_{310B} \rightarrow p_1 \rightarrow t_{130B} \rightarrow p_3 \rightarrow t_{300B} \rightarrow p_0$.

2) Add the transitions $t_{130B}$, $t_{310B}$, and $t_{300B}$.

IV. BEHAVIORAL AND QUANTITATIVE ANALYSIS OF CROPN

A. LIVENESS

A colored resource-oriented Petri net contains a number of circuits due to its strong connectedness. A special class of circuits in a CROPN is the production process circuit (PPC), which plays a crucial role for the liveness of the CROPN. Due to RMS routing complexity, the CROPN may have several PPCs, but only a few PPCs need to be identified in a CROPN.

**Definition 20** [46]: Let $N = (P, T, I, O, C, K, M_0, OR)$ be a CROPN. A circuit in a CROPN is said to be a PPC, denoted by $e$, if it does not include place $p_0$.

**Definition 21** [46]: Let $N = (P, T, I, O, C, K, M_0, OR)$ be a CROPN. A PPC $e$ is said to be elementary if it flows from a node $x$, through a number of nodes, back to $x$, and no node is repeated.

**Property 1**: In CROPN, a nonelementary PPC may consist of multiple elementary PPCs.

**Proof**: see [46].
Definition 22 [46]: Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. Let $P(e) = \{p_1, \ldots, p_n\}$ and $T(e) = \{t_1, \ldots, t_k\}$ be the sets of places and transitions on PPC $e$ with $p_i \in \bullet t_i, t_i \in P(e)$, and $|P(e)| = |T(e)|$, respectively. The tokens number of place $p_i$ that enable $t_i$ in $e$ is represented by $M(p_i, e)$. The tokens in a PPC $e$ are called cycling tokens of $e$ if $t_i$ is fired and the tokens remain in $e$. The tokens in a PPC $e$ are called leaving tokens of $e$ if $t_i$ is fired and the tokens leave $e$.

Definition 23 [46]: Let $(N, M_o)$ be a CROPN with $N = (P, T, I, O, C, K, M_o, OR)$. The number of cycling tokens of PPC $e$ at marking $M$ is defined by

$$M(e) = \sum M(p_i, e), \quad p_i \in P(e). \quad (7)$$

Definition 24 [46]: Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. A circuit in a CROPN is called an interactive PPC, formed by a number of PPCs, if each PPC in the circuit shares places and transitions with at least another PPC in the circuit and the circuit is strongly connected.

Definition 25 [46]: Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. Let $e^o$ denote an interactive circuit formed by $n$ PPCs in a CROPN, where $n$ is the number of PPCs that form the circuit. Let $P(e^o)$ and $T(e^o)$ be the set of places and transitions in $e^o$. For $p_i \in P(e^o)$, let $T_i = \{t \in p_i^* \cap T(e^o)\}$. The tokens in an $e^o$ are called cycling tokens of $e^o$ if the transitions (in $T_i$) enabled by a token are fired and the tokens remain in $e^o$. The tokens in $e^o$ are called leaving tokens of $e^o$ if the transitions (in $T_i$) enabled by a token are fired and the tokens leave $e^o$.

Definition 26 [46]: Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. The number of cycling tokens in $e^o$ at marking $M$ is defined by

$$M(e^o) = \sum M(p_i, e^o), \quad p_i \in P(e^o). \quad (8)$$

Property 2: Suppose that a CROPN with $N = (P, T, I, O, C, K, M_o, OR)$ be a marked net. A deadlock appears in a CROPN if there is a set of transitions $T_f$ in marking $M \in R(N, M_o)$ such that every $t \in T_f$ is process-enabled and none is resource-enabled.

Definition 27 [46]: Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. A PPC $e$ at marking $M$ is said to be full (no spaces in the places) if

$$\sum M(p_i) = \sum K(p_i) = K(e), \quad p_i \in P(e). \quad (9)$$

It is easy to verify that some PPCs are full when a deadlock appears. Even when one PPC is full, the system can be deadlock-free. Therefore, the following results are generated:

Lemma 1 [46]: If a CROPN is not live, then the net includes a PPC.

Lemma 2 [46]: The following two statements are similar:

1. A CROPN is not deadlock-free.
2. A CROPN is not live.

Lemmas 1 and 2 demonstrate that if deadlocks appear in a CROPN, then they appear in certain PPCs. Therefore, the following result is generated:

Theorem 1: The necessary condition for the existence of deadlocks in a CROPN is that at least one PPC $e$ exists such that

$$M_o(p_o) \geq K(e). \quad (10)$$

Proof: see [46].

Definition 28 [46]: Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. A transition $t$ in the CROPN is called a controlled transition if $t$ is both process and resource-enabled under the CROPN enabling rule.

Definition 29 [46]: Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. A CROPN is called a controlled transition if one transition or more is controlled in the net.

Definition 30 [46]: Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. Let $R_{C}(N, M_o)$ be the set of reachable markings of a CROPN under control.

Definition 31 [46]: Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. A PPC $e$ in a CROPN is called a live transition if every transition $t \in T(e)$ is live.

Definition 32 [46]: Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. A transition $t$ is called an input transition of interactive subnet $e^o$ if $t \notin T(e^o)$ and $t^* \in P(e^o)$. A transition $t$ is called an output transition of interactive subnet $e^o$ if $t \notin T(e^o)$ and $t^* \in P(e^o)$. Let $T_i(e^o)$ be an input transition set of interactive subnet $e^o$ and $T_o(e^o)$ be an input transition set of interactive subnet $e^o$.

Definition 33 [46]: Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. Let $S(e_i)$ be the number of free spaces in PPC $e_i$ in a CROPN at marking $M$, denoted by

$$S(e_i) = \sum_{p_j \in P(e_i)} (K(p_j) - M(p_j)) \quad (11)$$

and let $S'(e_i)$ be the number of current spaces available in PPC $e_i$ in a CROPN at marking $M$, denoted by

$$S'(e_i) = K(e_i) - M(e_i) \quad (12)$$
Theorem 2: Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. A subnet PPC $e$ in a CROPN is live if for any marking $M \in R_e(N, M_o)$,

$$S'(e) \geq 1.$$ (13)

**Proof:** see [46].

**Definition 34 [46]:** Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. A PPC $e$ in a CROPN is called an enabled if it is both process-enabled and resource-enabled.

**Corollary 1 [46]:** A PPC $e$ is an enabled PPC if the condition in Theorem 2 is satisfied.

Theorem 3: Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. An interactive subnet PPC $e^n$ in the CROPN is live if for any marking $M \in R_e(N, M_o)$ reachable from $M_o$, $S'(e_i) \geq 1$, for any $e_i$, (14)

and

$$\eta(e^n, M) \geq 1$$ (15)

where, $\eta(e^n, M)$ represents the number of enabled PPCs in $e^n$ at $M$.

**Proof:** see [46].

Theorem 4: Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. The CROPN is always live if there is no PPC in the net.

**Proof:** see [46].

We call a transition $t$ the input transition of multiple PPCs. Let $V_{en}(t)$ represent the set of these PPCs. Let $T_d$ be the set of transitions in CROPN such that if $t \in T_d$, then $V_{en}(t) \neq \emptyset$.

Theorem 5: Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. An interactive subnet $e^n$ in $N$ is live if (1) every transition $t \in T_f(e^n)$ is controlled; (2) before $t$ fires at marking $M$, $S'(e_i) \geq 2$, for any $e_i \in V_{en}(t)$; and (3) after $t$ fires, the marking of CROPN is changed from $M$ to $M'$ such that $\eta(e^n, M') \geq 1$.

**Proof:** see [46].

To demonstrate the liveness of the CROPN, let us consider the CROPN of the reconfigured manufacturing system shown in Figure 6. It contains one PPC: $e_1 = \{p_{1}, t_{304B}, p_{3}, t_{310B} \}$. Figure 7 displays the reachability graph of the CROPN illustrated in Figure 6. We suppose that all places other than $p_{2}$ and $p_{3}$ have a capacity of one with a machine and a buffer space. Based on the control law given in Theorem 2, the available spaces in PPC $e_1$ are computed as follows:

At the initial marking $M_0$,

$$K(e_1) = \sum K(p_i, e_1) = K(p_1, e_1) + K(p_3, e_1) = 2.$$ $M(e_1) = \sum M(p_i, e_1) = M(p_1, e_1) + M(p_3, e_1) = 0.$

$$S'(e_1) = K(e_1) - M(e_1) = 2 - 0 = 2.$$ $S'(e_1) \geq 1.$

At marking $M_1$

$$M(e_1) = \sum M(p_i, e_1) = M(p_1, e_1) + M(p_3, e_1) = 1 + 0 = 1.$$ $S'(e_1) = K(e_1) - M(e_1) = 2 - 1 = 1.$$ $S'(e_1) \geq 1.$

At marking $M_2$

$$M(e_1) = \sum M(p_i, e_1) = M(p_1, e_1) + M(p_3, e_1) = 0 + 0 = 0.$$ $S'(e_1) = K(e_1) - M(e_1) = 2 - 0 = 2.$$ $S'(e_1) \geq 1.$

At marking $M_3$

$$M(e_1) = \sum M(p_i, e_1) = M(p_1, e_1) + M(p_3, e_1) = 0 + 1 = 1.$$ $S'(e_1) = K(e_1) - M(e_1) = 2 - 1 = 1.$$ $S'(e_1) \geq 1.$

From the example, we can see that a subnet PPC $e_1$ in the CROPN is live and the condition in Theorem 2 is satisfied for any marking $M \in R_e(N, M_o)$. Thus, if the control law given in Theorem 2 is applied in real time, then the deadlock in a CROPN of the reconfigured manufacturing system presented in Figure 6 can be totally avoided.

**B. COMPUTATIONAL COMPLEXITY**

Theorem 6: Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN, and let $(N, M_o)$ be a live, bounded, and reversible net. Then the time complexity of Algorithm 1 is polynomial.

**Proof:** Let $N = (P, T, I, O, C, K, M_o, OR)$ be a CROPN. Since any route OR in a CROPN has $R_o \rightarrow R_{i1} \rightarrow R_{i2} \rightarrow \ldots \rightarrow R_{i\Theta} \rightarrow R_o$, each subroute $R$ needs place $p\sigma$ that relates to $R_{(j-1)}$, place $p_b$, which relates to $R_{ij}$, transition $t_i$, an arc $(p_s, t)$, an arc $(t, p_b)$, an arc $(p_s, t)$, and an arc $(t, p_s)$. Thus, let $\sigma$ be the cardinality of OR, i.e., $|OR| = \sigma$. Let $\zeta$ be the cardinality of $R_i$, i.e., $|R_i| = \zeta$. Let $\Theta$ be the number of corresponding places, transitions, and arcs to each subroute, and $\theta$ be the cardinality of $\Theta$, i.e., $|\Theta| = \theta$. The “FOR loop” is executed $(\sigma\theta)$ times to build the CROPN. Thus, the Algorithm 1 computational complexity is represented as...
O(σζθ). Therefore, the Algorithm 1 computational complexity is polynomial.

### C. GPENSIM CODE AND VALIDATION OF CROPN

To verify and validate Algorithm 1, the GPenSIM tool [1], [69] is employed to develop a code for the ICPN model. The developed code is implemented on MATLAB R2015a under the following conditions: Windows 10, 64-bit and Intel(R) Core (TM) i7-4702MQ CPU @ 2.20 GHz, 16 GB RAM. First, three GPenSIM files are generated, which are main simulation file (MSF), the PN definition file (PDF), and common processor file (COMMON_PRE file). The PDF is proposed to represent the sets of places, transitions, and arcs of the PN model. The COMMON_PRE file is developed to represent the conditions to enable fire transitions. The MSF is proposed to represent the simulation results. Then, the time performance is conducted, including the machines and robots, total throughput, and total throughput time. The results of the simulation are compared with those of the works by [1], [66]–[68]. Reconsider the model presented in Figure 3. The simulation was conducted for 480 minutes and Table 1 summarizes the simulation results of MATLAB, which presents the results in terms of the aforementioned criteria for time performance. It can also be observed that the proposed Algorithm 1 can obtain better utilization, greater throughput, and less throughput time than the other approaches, as presented in Figure 8. Thus, Algorithm 1 is effective and can accommodate other cases. Moreover, the results are accurate.

### V. LADDER DIAGRAM COLORED RESOURCE-ORIENTED PETRI NET

In an LD program, the input variable demonstrates the status of a switch or sensor that is sampled on regular intervals and inputs it to a PLC system; conversely, the output variable describes the status of an actuator such as a relay or motor, that is regularly delivered to an actuator by a PLC; and no external equipment, including the actuator or the sensors, is presented by an internal variable, but is used for an LD computation. The LD has nine element types, which include the left power bar, the right power bar, the contact, the coil, timer, counter, the horizontal line and the vertical line. The left and right power rails are represented as |. The main LD symbols for the Siemens Step 7-1200 PLC are presented in Table 2. The LD is a set of runs from the left power rails to the right power rail. Each rung is considered as a collection of contacts and coils linked by vertical and horizontal lines. In the same rung, contact and coil, which are represented by the same Boolean variable, may occur. In this article, it is assumed that coils are directly connected to the right power rail. The PLC is implemented on a periodic scanning. In each scan, the LD program is executed in the same way as the relay electrical connector chart. The execution flows from left to right and then from top to bottom; therefore, each output is calculated and updated.

**Definition 35:** Let \( N = (P, T, I, O, C, K, M_0, OR, I_s, Q, M, \Phi, \delta, \Xi) \) be a ladder diagram CROPN (LDCROPN), where \( P, T, F, W, M_0, K, \) and \( OR \) are defined in Definitions 1–19. Other symbols are defined as follows:

1. \( I_s = \{I_{S0}, I_{S1}, I_{S2}, I_{S3}, \ldots, I_l\} \) is a finite set of physical input signals denoting discrete physical inputs with \( |I| > 0 \).
2. \( Q = \{Q_0, Q_1, Q_2, Q_3, \ldots, Q_d\} \) is a finite set of places denoting physical output signals with \( |Q| = 0, |Q| > 0 \).
3. \( M = \{M_0, M_1, M_2, M_3, \ldots, M_k\} \) is a finite set of places denoting discrete memory signals with \( |M| > 0 \).
4. \( \Phi \) denotes the firing condition that maps each transition \( t \in T \) to a guard expression \( \Phi(\delta) \) that has a Boolean firing function,
5. \( \delta \) denotes the output function that maps each place \( p \in P \) to a guard expression \( \delta(p) \), which has a Boolean place function,
6. \( \Xi \) denotes the output function combining all the outputs \( \delta \) of all marked places \( p \in P \), \( \Xi: \{0, 1\}^p \rightarrow \{0, 1, -, c\} \), and \( c \) denotes the contradiction in the case where the same signal is set to 0 in a given place and one in another place; to compute \( \Xi(m_a) \), \( m_a: \Xi(m_a) \prod_i \delta(p_i) \) can be used.

**Definition 36:** Let \( N = (P, T, I, O, C, K, M_0, OR, I_s, Q, M, \Phi, \delta, \Xi) \) be an LDCROPN. When the left and right power rails are removed from an LD, a collection of contacts and coils satisfies the condition whose any two symbols are linked. Then it is said to be a rung \( g, g \in IZ, IZ = \{1, 2, 3, \ldots\} \).
Definition 37: Let \( N = (P, T, I, O, C, K, M_o, OR, I_s, Q, M, \Phi, \delta, \Xi) \) be an LDCROPN and \( k \in IZ \) coils, \( g \in IZ \) rungs, the LD graph of the \( (N, M_o) \) is expressed as \( \Omega_{LD} = (\Sigma, \gamma) \) if

1. \( \Sigma = \Sigma_L \cup \Sigma_R \cup \Sigma_v \) is a vertex set;
2. \( \Sigma_L = \{ \Sigma_{l1}, \Sigma_{l2}, \ldots, \Sigma_{lg} \} \) where \( \Sigma_{li}, 1 \leq i \leq g \) is called the junction of the \( i \) th rung and the left power rail in the LD;
3. \( \Sigma_R = \{ \Sigma_{r1}, \Sigma_{r2}, \ldots, \Sigma_{rg} \} \) where \( \Sigma_{ri}, 1 \leq i \leq g \) is called the junction of the \( i \) th rung and the right power rail in the LD;
4. \( \Sigma_v \) is called a set of components representing the LD symbols and \( v_{y1,y2,y3} \) is its element if
   a) \( yi \) is a character string variable of the component representing the name of the LD symbols such as contact, counter, timer, or coil;
   b) \( y2 \) is expressed as \( y2 = \{0, 1, 2, 3, 4, 5\} \) where 0 denotes a normally open contact, 1 denotes a normally closed contact, 2 denotes a coil, 3 represents a negated coil, 4 denotes a set coil, and 5 denotes a reset coil;
   c) \( y3 \) is expressed as \( y3 = \{0, i\} \), where 0 denotes a contact named by \( y/1 \) and \( i \), and 1 \( \leq i \leq g \) denotes the \( i \) th coil from top to bottom, named by \( yl \).

\( \gamma \) is a collection of edges. Let \( v_x \) and \( v_y \) be two components. For all \( v_x, v_y \in \Sigma, v_x, v_y \in \gamma \) if there exists a connect line between two symbols in the LD, which are corresponding to \( v_x \) and \( v_y \), respectively.

Definition 38: Let \( N = (P, T, I, O, C, K, M_o, OR, I_s, Q, M, \Phi, \delta, \Xi, \Omega_{LD}) \) be an LDCROPN. The vertex \( v_{y1,y2,y3} \) that is directly linked to \( \Sigma_{ri}, \Sigma_{ri} \in \Sigma_R, 1 \leq i \leq g, g \in IZ \) is said to be an assigned-component of the \( i \) th rung and represents the output variables in an LD graph \( \Omega_{LD} \).

Definition 39: Let \( N = (P, T, I, O, C, K, M_o, OR, I_s, Q, M, \Phi, \delta, \Xi, \Omega_{LD}) \) be an LDCROPN. Let \( v_{y1,y2,y3} \in \Sigma_v, (k = y3, 1 \leq k \leq g, g \in IZ) \) be an assigned-component if a path between \( \Sigma_{ri} \) and \( \Sigma_{ri} \) that includes \( v_{y1,y2,y3} \) exists; this path is expressed as \( \Delta(v_{y1,y2,y3}) \). Let \( \Delta(v_{y1,y2,y3}) \) be an instruction path of \( v_{y1,y2,y3} \) if \( \Sigma_{ri} \) and \( \Sigma_{ri} \) are removed from \( \Delta(v_{y1,y2,y3}) \) and represented by \( \Delta_{ki}, 1 \leq i \leq |\Delta(v_{y1,y2,y3})| \); \( k = y3 \) relates to the \( k \) th coil in the \( \Omega_{LD} \) and represents the \( k \) th path of \( \Delta(v_{y1,y2,y3}) \).

Definition 40: Let \( (N, M_o) \) be an LDCROPN with \( N = (P, T, I, O, C, K, M_o, OR, I_s, Q, M, \Phi, \delta, \Xi, \Omega_{LD}) \). The marking \( M' \) resulted from the firing of an enabled transition \( t \in T \) at marking \( M \) can be changed from \( M \) to \( M' \) as follows (16), as shown at the bottom of the previous page:

Definition 41: Let \( N = (P, T, I, O, C, K, M_o, OR, I_s, Q, M, \Phi, \delta, \Xi, \Omega_{LD}) \) be an LDCROPN. The marking \( M' \) resulted from the firing of an enabled transition \( t \in T \) at marking \( M \) that resets outputs places can be changed from \( M \) to \( M' \) as follows (17), as shown at the bottom of the previous page:

Theorem 7: Let \( N = (P, T, I, O, C, K, M_o, OR, I_s, Q, M, \Phi, \delta, \Xi, \Omega_{LD}) \) be an LDCROPN. The LDCROPN is live (all rungs work without errors).
TABLE 2. A sample of ladder diagram symbols for Siemens Step 7-1200 PLC.

| LD Element          | LD Symbol | Description                                                                                                                                 |
|---------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------|
| Normally open contact | ![Image] | It is often used in a PLC system to denote the variables input, internal, and output. It is closed (on) when it has signal state "1".               |
| Normally closed contact | ![Image] | It is often used in a PLC system to denote the variables input, internal, and output. It is closed (on) when it has signal state "0".             |
| Scan operand for the negative signal edge | ![Image] | This instruction determines whether the signal state of a given n1 operand is changed from "1" to "0". It compares the current n1 signal state with the previous scanning signal state which is saved in the edge memory bit n2. If a change in the result logic result (RLO) of "1" to "0" is detected by the instruction, the edge will fall negatively. |
| Scan operand for the positive signal edge | ![Image] | This instruction determines whether the signal state of a given n1 operand is changed from "0" to "1". It compares the current n1 signal state with the previous scanning signal state which is saved in the edge memory bit n2. If a change in the result logic result (RLO) of "0" to "1" is detected by the instruction, the edge will fall positively. |
| Coils are often used in a PLC system to denote the variables internal and output. The output coil is set when the input of the coil has signal state "1", if the signal state is "0" at the input of the coil, the output coil is reset. | ![Image] | Coils set instruction is used to set the signal state of a reset output to "1".                                                           |
| Coils set instruction is used to reset the signal state of a reset output to "0". | ![Image] |                                                                                                                                          |
| Comparators         | ![Image] | "Equal", "Greater or equal", or "Less or equal" instruction is used to query whether the value at input n1 is "equal", "Greater or equal", or "Less or equal" to the value at input n2, respectively. Both values to be compared must have the identical data type. |
| MOVE                | ![Image] | This instruction transfers the content of the operand at the IN input to the operand at the OUT output. The operands at input IN and at output OUT must be of the same data type. |
| ADD                 | ![Image] | This instruction adds the value at input IN1 and the value at input IN2 and query the sum at output OUT (OUT: = IN1+IN2).                        |
| SUB                 | ![Image] | It subtracts the value at input IN2 from the value at input IN1 and query the difference at output OUT (OUT: = IN1-IN2).                        |
| TON                 | ![Image] | "Generate on-delay" instruction delays the setting of a Q output by the programmed time PT. It starts with the input IN changing from "0" to "1". If the instruction is started, the programmed time PT begins. The output Q has the signal state "1", if the PT expires. |
| TP                  | ![Image] | In the "Pulse Timer" instruction, the output of TP Timer bit is on if the TP Timer is working and off at any other times.                     |
| CTUD                | ![Image] | "Count up and down" instruction is used to decrement and increment the value of counter at the CV output. The current counter value will be incremented by one, when the signal state at CU input is changed from "0" to "1" and the new value is stored at the CV output. The current counter value will be decremented by one, when the signal state at CD input is changed from "0" to "1" and the new value is stored at the CV output. The current counter value at the CV output remains the same, when there is a signal state "1" at the CU and CD inputs in one program cycle. |
Proof: It must be proven that all rungs $g, g \in \mathbb{Z}$ in $\Omega_{LD}$ are live and work from left to right, and from top to bottom. Let $v_{y_1,y_2,y_3} \in \Sigma_v$, ($k = y_3, 1 \leq k \leq g, g \in \mathbb{Z}$) be an assigned-component. Moreover, let $\Delta_k(\{v_{y_1,y_2,y_3}\})$ be an instruction path of $v_{y_1,y_2,y_3}$ represented by $\Delta_k, 1 \leq i \leq |\Delta_k(\{v_{y_1,y_2,y_3}\})|; k = y_3$ relates to the $k$th coil in the $\Omega_{LD}$ and represents the $k$th path of $\Delta_k$. For all $\Delta_k, i \in \Delta_k$, if $\Delta_k, i$ is working without error in any case, then it can be inferred that the LDCROPN $(N, M_0)$ is live. □

Theorem 8: Given an LDCROPN with $N = (P, T, I, O, C, K, M_o, \text{OR}, I_s, Q, M, \Phi, \delta, \Xi, \Omega_{LD})$, the time complexity of Algorithm 2 is polynomial.

Proof: Let $(N, M_o)$ be an LDCROPN with $N = (P, T, I, O, C, K, M_o, \text{OR}, I_s, Q, M, \Phi, \delta, \Xi, \Omega_{LD})$. A rung with its LD components needs to be added to each transition $t \in T$. Therefore, let $j$ be the cardinality of $T$, i.e., $|T| = j$. Each transition $t$ has input place $p \in \tau^*$, and each output place is represented by an LD component. Thus, let $E$ be the set of input places of transition $t$ and $e$ be the cardinality of $E$, i.e., $|E| = e$. Moreover, each transition $t$ has output place $p \in \tau^*$, and each output place is represented by an LD component. Thus, let $\Pi$ be the set of the output places of transition $t$ and $\pi$ be the cardinality of $\Pi$, i.e., $|\Pi| = \pi$. Also, physical output signal/s may exist that correspond to the output place/s of transition $t$, which can be represented by an LD component. Therefore, let $\Psi$ be the set of physical output places of transition $t$ and $\psi$ be the cardinality of $\Psi$, i.e., $|\Psi| = \psi$. To construct an $\Omega_{LD}$, the “FOR loop” is executed $(j \times \pi \times \psi)$ times. Thus, the Algorithm 2 computational complexity is represented as $O(j \times \pi \times \psi)$, which is polynomial. □

Based on Definitions 35–41 and Theorems 7 and 8, the developed algorithm for converting a CROPN into an LD is presented as follows:

**Algorithm 2 (Converting the CROPN Into an LD)**

**Input:** An LDCROPN $(N, M_o)$ with $N = (P, T, I, O, C, K, M_o, \text{OR}, I_s, Q, M, \Phi, \delta, \Xi)$.

**Output:** Output the LD graph $\Omega_{LD}$.

**Initialization:** $j = 0$

1. /* Start button to operate the whole system*/
2. $j = j + 1$;
3. Start a new rung $j$;
4. Add $v_{\text{ST},0,0}$ component;
5. Add $v_{\text{End},2,j}$ component;
6. /* End button to stop the whole system*/
7. $j = j + 1$;
8. Start a new rung $j$;
9. Add $v_{\text{L},0,0}$ component;
10. Add $v_{\text{Mend},2,j}$ component;
11. for each $t \in T$, do
12. $j = j + 1$;
13. Start a new rung $j$;
14. if there is a physical input signal $I_{\text{Si}} \in I_s$, then
15. Add $v_{\text{Si},0,0}$ component;
16. end if
17. end for
18. for each $p \in \tau^*$, do
19. $j = j + 1$;
20. Start a new rung $j$;
21. if an arc $(p, \pi)$ is normal, then
22. Add $v_{\text{Mp},0,0}$ component;
23. else
24. if an arc $(p, \pi)$ is inhibitor, then
25. Add $v_{\text{Mp},1,0}$ component;
26. end if
27. end if
28. end for
29. /* select case p type*/
30. case $p \in P_A$
31. $j = j + 1$;
32. Start a new rung $j$;
33. Add $v_{\text{Mp},0,0}$ component;
34. Add on delay time component;
35. branch the rung $j$;
36. Add $v_{\text{Op},2,j}$ component;
37. case $p \in P_T$
38. $j = j + 1$;
39. Start a new rung $j$;
40. Add $v_{\text{Ml},0,0}$ component;
41. Add $v_{\text{Mp},0,0}$ component;
42. Add on delay time component;
43. branch the rung $j$;
44. Add $v_{\text{Op},2,j}$ component;
45. end select
46. end for
47. end for
48. Output the LD graph $\Omega_{LD}$;
49. End

To demonstrate the LDCROPN synthesis, consider a CROPN presented in Figure 6. The obtained CROPN lacks recognition and identification of the two part types at the input buffer and machine 1. To enable recognition of parts A and B in the input buffer area, sensor 1 (an infrared reflective sensor) and sensor 2 (a proximity sensor) are added, respectively. Sensor 3 (an infrared reflective sensor) and sensor 4 (a proximity sensor) are added to enable recognition of parts A and B in machine 1, respectively. Moreover, sensor 5 (an infrared reflective sensor) and sensor 6 (a proximity sensor) are added to enable recognition of part A in machine 2 and part B in machine 3, respectively. Based on Definition 35,
TABLE 3. PLC’s physical input and output signals and memory addresses.

| Type | LDCCROPN | Address | Description       | Type | LDCCROPN | Address | Description |
|------|-----------|---------|-------------------|------|-----------|---------|-------------|
| Input| I_{i0}    | I.0     | Start pushbutton  | Output| Q_{i}     | Q0.0    | M1 operation |
|      | I_{i1}    | I.1     | Stop pushbutton   |      | Q_{i}     | Q0.1    | M2 operation |
|      | I_{i2}    | I.2     | Part A detection sensor 1 |      | Q_{i}     | Q0.2    | M3 operation |
|      | I_{i3}    | I.3     | Part B detection sensor 2 |      | Q_{i}     | Q0.3    | R1 operation |
|      | I_{i4}    | I.4     | Part A detection sensor 3 |      | Q_{i}     | Q0.4    | R2 operation |
|      | I_{i5}    | I.5     | Part B detection sensor 4 |      | -        | -       | -            |
|      | I_{i6}    | I.6     | Part A detection sensor 5 |      | -        | -       | -            |
|      | I_{i7}    | I.7     | Part B detection sensor 6 |      | -        | -       | -            |

TABLE 4. Boolean firing, place functions, and memory addresses for an LDCCPN.

| LDCCROPN | Address | Boolean firing function | LDCCROPN | Address | Boolean place function |
|----------|---------|-------------------------|----------|---------|------------------------|
| \Phi(t_{0108}) | M0.0  | (I_{i0}&M_{i12}&M_{i0.7})\text{OR} \text{M}_{i0.0}\text{\&}T_{i1} | \Phi(t_{3100}) | M0.6  | (M_{i1.8}&M_{i1.9})\text{OR} \text{M}_{i0.0}\text{\&}T_{i1} |
| \Phi(t_{0108}) | M0.1  | (I_{i0}&M_{i12}&M_{i0.8})\text{OR} \text{M}_{i0.1}\text{\&}T_{i1} | \Phi(t_{3100}) | M0.7  | (M_{i1.8}&M_{i1.9})\text{OR} \text{M}_{i0.0}\text{\&}T_{i1} |
| \Phi(t_{3100}) | M0.2  | (M_{i1.8}&M_{i1.9})\text{OR} \text{M}_{i0.0}\text{\&}T_{i1} | \Phi(t_{3100}) | M0.3  | (M_{i1.8}&M_{i1.9})\text{OR} \text{M}_{i0.0}\text{\&}T_{i1} |
| \Phi(t_{3100}) | M0.4  | (M_{i1.8}&M_{i1.9})\text{OR} \text{M}_{i0.0}\text{\&}T_{i1} | \Phi(t_{3100}) | M0.5  | (M_{i1.8}&M_{i1.9})\text{OR} \text{M}_{i0.0}\text{\&}T_{i1} |

FIGURE 9. The developed LDCCPN of the CROPN model presented in Figure 6.

The developed LDCCPN is presented in Figure 9. The physical input and output signals of the PLC are presented in Table 3, and the Boolean firing and place functions are listed in Table 5. The start and stop pushbuttons for the whole system are realized by input registers I\_0.0 and I\_0.1, respectively. Moreover, sensor readings are realized by input registers I\_0.2, I\_0.3, I\_0.4, I\_0.5, I\_0.6, and I\_0.7. The output bits, Q\_0.0, Q\_0.1, Q\_0.2, Q\_0.3, and Q\_0.4, are added to the action places. Conversely, the memory bits (M\_x.x) are assigned to places and transitions. It is noteworthy that we can assign two memory bits to the place whose capacity has two tokens. For the representation of the time delays, an on-delay timer is associated with timed transitions and places.

The LDCCPN model has the following specifications.

1. The loading operation of robot 1 is switched on with a level action (Q\_0.3) through transition I_{0108} if a part B exists in the input buffer area, i.e., M(p_{0}) = \{c_{p2}\} and sensor 2 detects it (with I\_0.3), Robot 1 is idle, i.e., M(p_{r}) = \{c_{t1}\}, and there is no operation in machine 1, i.e. M(p_{1}) = 0. The loading operation of robot 1 is switched off through transition I_{0108}, and occurs when a part B exists in machine 1 is detected. The operation of machine 1 is switched on with a level action (Q\_0.0) if a part B exists in machine 1, i.e., M(p_{1}) = \{c_{p2}\}, and sensor 4 detects it (with I\_0.5) and if the loading time of robot 1 has elapsed.

2. The unloading operation of the robot 1 is switched on with a level action (Q\_0.3) through transition I_{3100} if the operation time of machine 1 has elapsed and Robot 1 is idle, i.e., M(p_{r}) = \{c_{t1}\}. The unloading operation of robot 1 is switched off through transition I_{3100}, and occurs when a part B exists in inspection machine 3 is detected. Inspection machine 3’s operation is switched on with a level action (Q\_0.2) if a part B exists in the machine 3, i.e., M(p_{3}) = \{c_{p2}\}, and if the loading time of robot 1 has elapsed. The operation of inspection machine 3 is switched off through transition I_{3100} (rejected part) or transition I_{3000} (accepted part), and occurs when the inspection time has elapsed.

3. The unloading operation of the robot 1 for an accepted part B is switched on with a level action (Q\_0.3) through transition I_{3100} if the operation time of machine 3 has elapsed and robot 1 is idle, i.e., M(p_{r}) = \{c_{t1}\}. The unloading operation of robot 1 for a rejected part B is switched on with a level action (Q\_0.3) through transition I_{3100} if the operation time of machine 3 has elapsed and robot 1 is idle, i.e., M(p_{r}) = \{c_{t1}\}.
FIGURE 10. Ladder diagram for the LDCROPN model illustrated in Figure 9.
4. The loading operation of robot 2 is switched on with a level action (Q0.4) through transition \( t_{010A} \) if a part A exists in the input buffer area, i.e., \( M(p_b) = \{c_{p1}\} \) and sensor 1 detects it (with I0.2), Robot 2 is idle, i.e., \( M(p_r) = \{c_{t2}\} \), and there is no operation in machine 1, i.e. \( M(p_1) = 0 \). The loading operation of robot 2 is switched off through transition \( t_{010A} \), and occurs when a part A exists in machine 1 is detected. The operation of machine 1 is switched on with a level action (Q0.0) if a part A exists in machine 1, i.e., \( M(p_1) = \{c_{p1}\} \) and sensor 3 detects it (with I0.4), and if the loading time of robot 2 has elapsed.

5. The unloading operation of robot 2 is switched on with a level action (Q0.4) through transition \( t_{120A} \) if the operation time of machine 1 has elapsed and robot 2 is idle, i.e., \( M(p_r) = \{c_{t2}\} \). The unloading operation of robot 2 is switched off through transition \( t_{120A} \), and occurs when a part A exists in machine 2 is detected. The operation of machine 2 is switched on with a level action (Q0.1) if a part A exists in machine 2,
i.e., \( M(p_2) = [c_p] \), and the loading time of robot 2 has elapsed.

6. The unloading operation of the robot 2 for part A is switched on with a level action (Q0.4) through transition \( t_{200B} \) if the operation time of machine 2 has elapsed and robot 2 is idle, i.e., \( M(p_r) = [c_p] \).

VI. CONCLUSION

In this paper, a novel two-step approach is proposed for the implementation of ladder diagrams (LDs) for reconfigurable manufacturing systems (RMSs) with dynamic changes. First, a CROPN-based algorithm is developed to enable a rapid and valid reconfiguration of an RMS. Second, an LDCROPN is established to convert the CROPN into an LD. The developed approach is validated using examples in the literature. The effectiveness of the proposed approach for the implementation of LDs under RMS configuration changes is validated by the simulation results.

The main advantages of this paper are summarized as follows. (1) The proposed approach has the ability to easily handle any complex RMS change compared with the studies in [70]–[72]. (2) The obtained LD has a simpler structure and has low overhead computation compared with the approaches proposed in [30]–[33], [35].

The main limitations of the proposed approach are that the obtained result is restricted to discrete data types, including bit strings, integer values, and Boolean, as the CROPN is based on discrete data types. Thus, our future research will extend the proposed approach to use the continuous CROPN in order to tackle the continuous data type generally used by the programmable logic controller systems. Moreover, automatic transformation programs based on the proposed algorithms need to be developed, and systematic analysis techniques for user programs that are proposed based on the commercially available Siemens Step 7–1200 PLC need to be presented. Another interesting future research topic is to conduct a study on an LD control algorithm with more input and output signals, and define new transforming structures for additional control logic. Finally, the proposed method needs to be extended to such complex flexible assembly/disassembly systems. The proposed method will be applied to multiagent systems and social systems [73]–[76].

VII. ACKNOWLEDGMENT

The authors are grateful to the Raytheon Chair for Systems Engineering for funding.

REFERENCES

[1] H. Kaid, A. Al-Ahmari, Z. Li, and R. Davidjahj, “Single controller-based colored Petri nets for deadlock control in automated manufacturing systems,” *Processes*, vol. 8, no. 1, p. 21, Dec. 2019.

[2] H. Kaid, A. Al-Ahmari, Z. Li, and R. Davidjahj, “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.

[3] Z. Li and M. Zhou, *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*. Cham, Switzerland: Springer, 2009.

[4] 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.

[5] 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.

[6] 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.

[7] Q. Chen, L. Yin, N. Wu, M. A. El-Meligy, M. A. F. Sharaf, and Z. Li, “Diagnosability of vector discrete-event systems using predicates,” *IEEE Access*, vol. 7, pp. 147143–147155, 2019.

[8] 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. 103321–103323, 2020.

[9] 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.

[10] 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.

[11] M. P. Cabasino, A. Giua, M. Poci, 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.

[12] Z. Ma, G. Zhu, 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.

[13] C. Lagartinho-Oliveira, F. Moutinho, and L. Gomes, “GPGPUs applied to support the construction of the state-space graphs of IOTP Petri net models,” in *Proc. 45th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2019, pp. 5862–5867.

[14] C. R. Vázquez, D. Gómez-Gutiérrez, and A. Ramírez-Treviño, “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.

[15] J. A. V. Garnica, E. R. Beltran, V. H. H. Garcia, A. R. Trevino, and I. R. Leon, “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.

[16] A. Ramírez-Treviño, I. Rivera-Rangel, and E. Lopez-Mellado, “Observability of discrete event systems modeled by interpreted Petri nets,” *IEEE Trans. Robot. Autom.*, vol. 19, no. 4, pp. 557–565, Aug. 2003.

[17] 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.

[18] I. Grobelna, R. Wisniewski, M. Grobelny, and M. Wisniewska, “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.

[19] R. Wisniewski, I. Grobelna, and A. Karateevich, “Determinism in cyber-physical systems specified by interpreted Petri nets,” *Sensors*, vol. 20, no. 19, p. 5565, Sep. 2020.

[20] R. Wisniewski, L. Stefanowicz, A. Bukowiec, and J. Lipiński, “Theoretical aspects of Petri net decomposition based on invariants and hypergraphs,” in *Multimedia and Ubiquitous Engineering*, Cham, Switzerland: Springer, 2014, pp. 371–376.

[21] J. Asensio, F. Ortúñoz, M. Darnas, 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.

[22] G. Fan and W. Ning, “Transformation algorithm between ladder diagram and instruction list based on AOV diagram and binary tree,” *J. Nanjing Univ. Aeronaut. Astronaut.*, vol. 6, no. 38, pp. 754–758, 2006.

[23] L. Blocks, “Functional block diagram,” *Matrix*, vol. 8, no. 8, p. 8, 1998.

[24] A. T. Azar and S. Vaidyanathan, *Advances in Chaos Theory and Intelligent Control*, vol. 337, Cham, Switzerland: Springer, 2016.

[25] V. Vyatkin, *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design*, 3rd ed. Pittsburgh, PA, USA: International Society of Automation, 2007.

[26] 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 Automation, (SEIKEN)* Symp.-Novel Disciplines Next Centur., Nov. 1994, pp. 428–435.

[27] 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, 1994.
M. Uzam and A. Jones, “A rule-based methodology for supervi-
sion of regions,” Int. J. Mech. Eng. Educ., vol. 46, no. 4, pp. 302–314, Oct. 2018.

R. Feio, J. Rosas, and L. Gomes, “Translating IOPT Petri net models into PLC ladder diagrams,” in Proc. 5th IEEE Int. Conf. Mechatronics Control (ICMC), Mar. 2017, pp. 1211–1216.

L. Gomes, F. Moutinho, F. Pereira, J. Ribeiro, A. Costa, and J.-P. Barros, “Extending input-output place-transition Petri nets for distributed controller systems development,” in Proc. Int. Conf. Mechatronics Control (ICMC). Jul. 2014, pp. 1099–1104.

L. Gomes, J. P. Barros, A. Costa, and R. Nunes, “The input-output place-transition Petri net class and associated tools,” in Proc. 5th IEEE Int. Conf. Ind. Informat., Jul. 2007, pp. 509–514.

N. Wu and M. Zhou, System Modeling and Control With Resource-Oriented Petri Nets. New York, NY, USA: CRC Press, 2009.

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.

H. Chen, N. Wu, and M. Zhou, “A novel method for deadlock preven-
tion of AMS by using resource-oriented Petri nets,” Inf. Sci., vol. 363, pp. 178–189, Oct. 2016.

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.

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.

N. Wu, “Avoiding deadlocks in automated manufacturing systems with shared material handling system,” in Proc. Int. Conf. Robot. Autom., 1997, pp. 2427–2432.

N. Wu and M. Zhou, “Deadlock avoidance in semiconductor track sys-
tems,” in Proc. IEEE Int. Conf. Robot. Autom., May 2002, pp. 193–198.

Z. Xiang, “Deadlock avoidance of flexible manufacturing systems by color-
ed resource-oriented Petri nets with novel colored capacity,” EasyChair, Manchester, U.K., Tech. Rep., 2020.

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.

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.

H. Chen, N. Wu, Z. Li, and T. Qiu, “On a maximally permissive deadlock prevention policy for automated manufacturing systems by using resourc eof-oriented Petri nets,” ISA Trans., vol. 89, pp. 67–76, Jun. 2019.

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.

N. Wu, M. Zhou, and G. Hu, “On Petri net modeling of automated manufacturing systems,” in Proc. IEEE Int. Conf. Netw., Sens. Control, 2007, pp. 228–233.

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.

N. Wu and M. Zhou, “Resource-oriented Petri nets in deadlock avoid-
ance of AGV systems,” in Proc. ICRA, IEEE Int. Conf. Robot. Autom., May 2001, pp. 64–69.

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.

H. Chen, N. Wu, Z. Li, T. Qiu, and H. Xiao, “Liveness of disjunctive and strict single-type automated manufacturing system: An ROPN approach,” IEEE Access, vol. 7, pp. 17760–17771, 2019.

N. Wu and M. Zhou, “Deadlock resolution in automated manufactur-
ing systems with robots,” IEEE Trans. Autom. Sci. Eng., vol. 4, no. 3, pp. 474–480, Jul. 2007.

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.

K. Xing, M. Chu Zhou, H. Liu, and F. Tian, “Optimal Petri-net-based polynomial-complexity deadlock-avoidance policies for automated manufac-
turing systems,” IEEE Trans. Syst., Man, Cybern., A, Syst. Humans, vol. 39, no. 4, pp. 188–199, Jan. 2009.

Z. Li and M. Zhou, “Elementary siphons of Petri nets and their appli-
cation to deadlock prevention in flexible manufacturing systems,” IEEE Trans. Syst., Man, Cybern., A, Syst. Humans, vol. 34, no. 1, pp. 38–51, Jan. 2004.

H. Kaid, A. Al-Ahmari, Z. Li, and R. Davidrajuh, “Automatic supervisory controller for deadlock control in reconfigurable manufacturing systems with dynamic changes,” Appl. Sci., vol. 10, no. 15, p. 5270, Jul. 2020.

J. Ezpeleta, J. M. Colom, and J. Martinez, “A Petri net based deadlock prevention policy for flexible manufacturing systems,” IEEE Trans. Robot. Autom., vol. 11, no. 2, pp. 173–184, Apr. 1995.

R. Davidrajuh, Modeling Discrete-Event Systems With GPenSIM: An Intro-
duction. Cham, Switzerland: Springer, 2018.

S. Lee and D. M. Tilbury, “Deadlock-free resource allocation control for a reconfigurable manufacturing system with serial and parallel configu-
ration,” IEEE Trans. Syst., Man Cybern., C, Appl. Rev., vol. 37, no. 6, pp. 1373–1381, Nov. 2007.

J. Li, X. Dai, and Z. Meng, “Automatic reconfiguration of Petri net controllers for reconfi gurable manufacturing systems with an improved net rewriting system-based approach,” IEEE Trans. Autom. Sci. Eng., vol. 6, no. 1, pp. 156–167, Jan. 2009.

N. Wu and M. 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, Feb. 2011.

L. Yang, Z. Yu, M. A. El-Meligy, A. M. El-Sherbeeny, and N. Wu, “On multiplexity-aware influence spread in social networks,” IEEE Access, vol. 8, pp. 106705–106713, 2020.

X. Li, Z. Yu, Z. Li, and N. Wu, “Group consensus via pinning control for a class of heterogeneous multi-agent systems with input constraints,” Inf. Sci., vol. 542, pp. 247–262, 2020.
HUSAM KAID received the B.S. degree in industrial engineering from the University of Taiz, Taiz, Yemen, in 2010, and the M.S. degree in industrial engineering from King Saud University, Saudi Arabia, in 2015, where he is currently pursuing the Ph.D. degree with the Industrial Engineering Department, College of Engineering. He is also a Researcher with the Industrial Engineering Department, College of Engineering, King Saud University. His research interests include design and analysis of manufacturing systems, deadlock control in manufacturing systems, supply chain, simulation, operations research, optimization techniques, and bibliometric network analysis.

ABDULRAHMAN AL-AHMARI (Member, IEEE) received the Ph.D. degree in manufacturing systems engineering from The University of Sheffield, Sheffield, U.K., in 1998. He worked as the Dean of the Advanced Manufacturing Institute and the Chairman of Industrial Engineering Department. He is currently a Professor of industrial engineering with King Saud University, Riyadh, Saudi Arabia. He led a number of funded projects from different organizations in Saudi Arabia. He has published articles in leading the Journal of Industrial and Manufacturing Engineering. His current research interests include advanced manufacturing technologies, Petri nets, analysis and design of manufacturing systems, computer integrated manufacturing, optimization of manufacturing operations, flexible manufacturing systems and cellular manufacturing systems, and applications of decision support systems in manufacturing.

ZHIWU LI (Fellow, IEEE) received the B.S. degree in mechanical engineering, the M.S. degree in automatic control, and the Ph.D. degree in manufacturing engineering from Xidian University, Xi’an, China, in 1989, 1992, and 1995, respectively. He joined Xidian University, in 1992. He was a Visiting Professor with the University of Toronto, Technion (Israel Institute of Technology), Martin Luther University, Conservatoire National des Arts et Métiers (Cnam), Meliksah Universities, University of Cagliari, University of Alberta, and King Saud University. He is currently with the Macau University of Science and Technology. His current research interests include Petri net theory and application, supervisory control of discrete event systems, workflow modeling and analysis, system reconfiguration, game theory, and data and process mining. He serves as a reviewer for more than 90 international journals. He is listed in Marquis Who’s Who in the world, 27th Edition, 2010. He was a recipient of an Alexander von Humboldt Research Grant, Alexander von Humboldt Foundation, Germany, and Research in Paris, France. He is the Founding Chair of the Xi’an Chapter of IEEE Systems, Man, and Cybernetics Society.