Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2015, Article ID 813476, 14 pages
http://dx.doi.org/10.1155/2015/813476

Research Article

Modeling and Verification of Reconfigurable and Energy-Efficient Manufacturing Systems

Jiafeng Zhang,[1,2] Mohamed Khalgui,[3] Wassim Mohamed Boussahel,[2,4] Georg Frey,[2] ChiTin Hon,[5] Naiqi Wu,[5] and Zhiwu Li[5,6]

[1]*School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China*
[2]*Chair of Automation and Energy Systems, Saarland University, 66123 Saarbrücken, Germany*
[3]*University of Carthage, 1054 Carthage, Tunisia*
[4]*Zentrum für Mechatronik und Automatisierungstechnik, 66121 Saarbrücken, Germany*
[5]*Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau*
[6]*Faculty of Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia*

Correspondence should be addressed to Zhiwu Li; systemscontrol@gmail.com

Received 4 September 2014; Revised 4 March 2015; Accepted 5 March 2015

Academic Editor: Cengiz Çinar

This paper deals with the formal modeling and verification of reconfigurable and energy-efficient manufacturing systems (REMSs) that are considered as reconfigurable discrete event control systems. A REMS not only allows global reconfigurations for switching the system from one configuration to another, but also allows local reconfigurations on components for saving energy when the system is in a particular configuration. In addition, the unreconfigured components of such a system should continue running during any reconfiguration. As a result, during a system reconfiguration, the system may have several possible paths and may fail to meet control requirements if concurrent reconfiguration events and normal events are not controlled. To guarantee the safety and correctness of such complex systems, formal verification is of great importance during a system design stage. This paper extends the formalism reconfigurable timed net condition/event systems (R-TNCESs) in order to model all possible dynamic behavior in such systems. After that, the designed system based on extended R-TNCESs is verified with the help of a software tool SESA for functional, temporal, and energy-efficient properties. This paper is illustrated by an automatic assembly system.

1. Introduction

A reconfigurable manufacturing system (RMS) is designed at the outset for rapid change in structure, as well as in hardware and software components, in order to quickly adjust production capacity and functionality within a part family in response to sudden changes in market or in regulatory requirements [1]. A RMS should be designed with several configurations (behavior modes) to, respectively, meet different production requirements in various conditions. There are two types of reconfigurations: static and dynamic reconfigurations. Generally, a static reconfiguration is applied offline to modify a RMS extensively such as adjusting architecture of physical systems and removing obsoleted machines, whereas a dynamic system reconfiguration, to switch a RMS from one configuration to another at runtime, is applied with the aim of fault-tolerance or actively changing system behavior modes [2, 3]. This paper focuses on dynamic RMSs.

Traditionally, manufacturing is an energy-intensive process, using motors, steam, and compressed air systems to transform raw materials into durable goods and consumer products [4–6]. Recent research shows that switching machines of a manufacturing system into their energy-efficient modes when they are idle during production can make considerable contribution to the reduction of energy demand and thus can reduce carbon footprint as well as operating costs [7–15]. This paper takes the advantage of dynamic reconfigurations of machines of a RMS between their working modes and energy-efficient modes as a way of reducing system energy consumption. A RMS with such

energy-efficient operations is called a reconfigurable and energy-efficient manufacturing system (REMS).

REMSs can be abstracted as reconfigurable discrete event systems (DESs) when only their logic behavior properties are investigated. In this paper, a reconfiguration is called a local reconfiguration, if it is applied for switching a machine of a REMS between its working mode and energy-efficient mode. A reconfiguration is named a global reconfiguration if it is applied for switching a REMS between different configurations.

A REMS should be able to reconfigure itself smoothly due to changed inner/outer environments at runtime. Meanwhile, normal unreconfigured events should go on occurring whenever they meet their occurrence preconditions. However, uncontrolled concurrence of reconfiguration events and normal events may cause faults such as deadlocks and overflow [16–19]. Therefore, the formal verification is of great importance during design stages.

Petri nets [20, 21] have found an extensive application to discrete event systems [22, 23], including automated flexible manufacturing systems [24–28] and reconfigurable systems [29]. Reconfigurable timed net condition/event systems (R-TNCESs) [30, 31] are reconfigurable extensions of timed net condition/event systems (TNCESs) [32, 33]. TNCESs [34, 35] have a visual graph expression, a clear modular structure, and an exact mathematical definition inherited from Petri nets [36–40]. In addition, they have a strong analysis software tool: SESA (http://homepages.engineering.auckland.ac.nz/vyatkin/tools/modelchekers.html) [41]. System behavior properties, such as state/event trajectories and temporal requirements, can be specified by Computation Tree Logic (CTL), extended CTL (eCTL), and timed CTL (TCTL) [42–44] before being checked by SESA automatically. If a property is satisfied by the system, the model checker will return "true". Otherwise, a counterexample will be returned. Therefore, TNCESs have been widely applied in verification and validation of industry control systems especially for manufacturing systems [45–47]. The verification of a R-TNCES can be performed with the assistance of SESA [30, 31].

However, R-TNCESs cannot fully meet our requirements for a REMS. In a R-TNCES, reconfiguration functions model system reconfiguration events and transitions model normal events. However, the concurrence of reconfiguration functions and transitions is forbidden in a R-TNCES, which is in fact inconsistent with system requirements of REMSs. As a result, formal verification of such complex systems cannot be performed.

Motivated by the fact aforementioned, this paper extends R-TNCESs. First, the reconfiguration functions of R-TNCESs are assigned with action ranges and concurrent decision functions. After that, they are divided into two types according to their action ranges: major and minor reconfiguration functions. The major ones are used to model global reconfiguration events, whereas the minor ones are applied to model local reconfiguration events. Accordingly, the dynamics of R-TNCESs is updated for these extensions such that the concurrence of reconfiguration events and normal events can be conditionally allowed to guarantee the system correctness. Afterwards, an implementation method for an extended R-TNCES is developed. Finally, the software tool SESA is applied to check system functional, temporal, and energy-efficient properties. An automatic assembly system is used to illustrate this work.

The paper is organized as follows. The system specification of REMSs and the applied automatic assembly system are depicted in Section 2. The drawbacks of R-TNCESs on analyzing REMSs and the proposed extended R-TNCESs are described in Section 3. The formal verification of a REMS based on extended R-TNCESs is illustrated in Section 4. Finally, Section 5 concludes this paper and briefly presents further studies.

2. Reconfigurable and Energy-Efficient Manufacturing Systems

This paper treats a reconfigurable and energy-efficient manufacturing system (REMS) as a reconfigurable discrete event control system. This section presents system specification and interesting system dynamics before it illustrates them with an automatic assembly system.

2.1. System Specification. A REMS is designed with a set of configurations to meet various requirements in different execution environments. A configuration Con is defined as

$$\text{Con} = (\text{Com}, \text{Str}, \text{Dat}), \tag{1}$$

where Com is a set of all activated components in Con, Str defines the structure, that is, the connection relationship and the communication protocol among components of Com, and Dat denotes the set of all global variables and parameters of Con.

A REMS is denoted by

$$\text{Sys} = \left(\sum, R_c\right), \tag{2}$$

where $\sum$ is the set of n configurations and $R_c : \sum \rightarrow \sum$ is the reconfigurable controller dealing with system reconfigurations.

There are two types of system reconfigurations in a REMS: global and local reconfigurations. The former ones are applied for switching system configurations. The latter ones are applied for switching an activated component between its working mode and energy-efficient mode when the system is in a particular configuration.

A REMS starts running as described in one of these configurations. After that, it should be able to change into other configurations smoothly due to the detection of component faults or other well-defined conditions. In addition, in each configuration, local reconfigurations can be applied to components such that the components can reconfigure themselves into their energy-efficient modes to save energy when they are idle and turn back to their working modes when the system needs them.

Dynamics of a REMS can be described by the evolution of system states. The evolution is caused by the occurrences of events events. A REMS includes three types of normal events, local reconfiguration events, and global reconfiguration events.

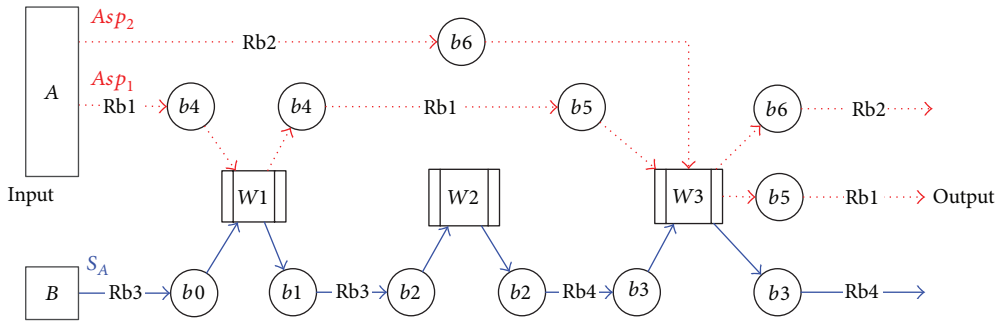

Figure 1: Default working process diagram of AAS.

(1) If a normal event occurs, the system changes its state within its current configuration.

(2) If a local reconfiguration event occurs, a component of current configuration switches into its energy-efficient mode or switches back into its working mode.

(3) If a global reconfiguration event occurs, the system switches into another configuration.

Meanwhile, during a global or local reconfiguration, if normal events meet their occurring conditions and they are not modified by the occurring reconfiguration events, they should go on occurring. However, this kind of concurrence brings safety threat to the system, since they may cause unboundedness, deadlocks, and even other functional or temporal failings.

2.2. Running Example. An automatic assembly system, denoted by AAS, is applied to illustrate works presented in this paper. AAS includes three workstations ($W1$, $W2$, and $W3$) and four robots (Rb1, Rb2, Rb3, and Rb4). It is assumed that robots are high energy consumption machines. The respective time consumption of $W1$, $W2$, and $W3$ to finish a machining task is 40 time units, 30 time unites, and 50 time unites. The time consumption of both Rb1 and Rb2 to finish a task is 20 time units. The time consumption of both Rb3 and Rb4 to finish a task is 25 time units. The default working process diagram of AAS is shown in Figure 1.

The main function of AAS is to assemble machine parts into a subassembly of a vehicle, to be marked by S_A. Robots Rb1 and Rb2 move machine parts from the input into AAS, transfer machine parts between workstations, and remove trashy machine parts to the output. Dotted arrows in Figure 1 are used to denote the movements of machine parts during an assembly process. On the other hand, S_A is shifted along $W1$, $W2$, and $W3$ by robots Rb3 and Rb4. Solid arrows in Figure 1 are used to denote the movement of S_A. To make it clear, $b0, b1, \ldots$, and $b6$ are used to denote positions where machine parts or S_A should be during an assembly process. The main assembly process is briefly described by the following three steps.

(1) The to-be-worked subassembly S_A is shifted from input B to $b0$ by Rb3. A machine part Asp_1 is delivered to $b4$ from the input A. After that, Asp_1 and S_A are preprocessed on $W1$. Then, the preprocessed S_A is moved to $b1$ automatically. The preprocessed Asp_1 is moved to $b4$ automatically before being moved to position $b5$ by Rb1.

(2) S_A is transported to $b2$ from $b1$ by Rb3. Then, a second preprocess for S_A is done by $W2$. After that, S_A is shifted to $b3$ from $b2$ by Rb4.

(3) A machine part Asp_2 is delivered to $b6$ by Rb2. Then, $W3$ starts the assembly after S_A is in $b3$, preprocessed Asp_1 is in $b5$, and Asp_2 is in $b6$. After the assembly, the machined S_A is moved out by Rb4. Two other trashy machine parts are removed out of AAS by Rb1 and Rb2, respectively.

It is assumed that four behavior modes are designed for AAS. Their work processes are illustrated as follows.

(i) *Mode* 1: *Mode* 1 is the default mode as depicted in Figure 1, where all the robots are applied.

(ii) *Mode* 2: *Mode* 2 is a responding mode when Rb2 breaks down, where Rb1 should update itself to perform the function of Rb2.

(iii) *Mode* 3: *Mode* 3 is applied when Rb4 breaks down during the execution of *Mode* 1, where the work of Rb4 has to be done by Rb3.

(iv) *Mode* 4: *Mode* 4 is applied when both Rb1 and Rb2 break down, where only Rb1 and Rb3 are applied. In this case, Rb1 should cover the function of Rb2 as in *Mode* 2 and Rb3 should cover the function of Rb4 as in *Mode* 3.

In each behavior mode, the applied robots should be able to reconfigure themselves into their energy-efficient modes when they are idle and reconfigure themselves back into their working modes when they have new tasks. A local reconfiguration for switching a robot from its working mode to its energy-efficient mode consumes one time unit. Likewise, a local reconfiguration for switching a robot from its energy-efficient mode back to its working mode consumes one time unit, as well.

To avoid the halt of a continuous production line, possible dynamic reconfigurations applied for switching AAS between these behavior modes are shown in Figure 2. The solid arrows denote global reconfigurations and dotted ones denote local reconfigurations.

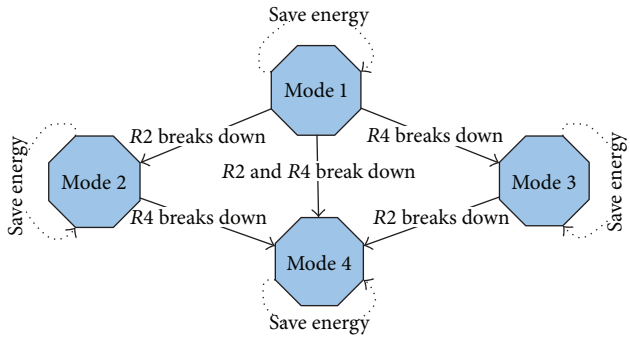

Figure 2: Possible reconfigurations in AAS.

It is assumed that a robot consumes one energy unit per time unit when it works in its working mode. However, it only consumes 30% energy units per time unit when it works in its energy-efficient mode. Note that the numerical value "30%" is an assumption by the authors to facilitate the quantitative analysis on energy-efficient operations. It does not come from any literature on industry systems.

Obviously, the possible reconfiguration events of AAS can occur simultaneously with many normal events in it. For example, when Rb1 is being modified by a global reconfiguration or being switched into its energy-efficient mode, only its own work needs to stop for a while and the workstations and other running robots should do their jobs unaffectedly.

3. Extended R-TNCES

Reconfigurable timed net condition/event systems (R-TNCESs) [30, 31] are extensions of timed net condition/event systems (TNCESs) [34, 35]. Reconfiguration functions of R-TNCESs can be used to model global reconfiguration events of REMSs. However, they are not proper to model local reconfiguration events of REMSs directly. In addition, the concurrence of normal events and reconfiguration events is currently not allowed in R-TNCESs. Therefore, in order to perform correct formal verification of a REMS, this paper extends R-TNCESs. This section briefly recalls basic conceptions of R-TNCESs, analyzes the drawbacks of R-TNCEs on investigating REMSs, and represents the proposed extended R-TNCESs.

3.1. R-TNCEs

Definition 1 (see [30]). A R-TNCES is a structure $RN = (\mathcal{B}, \mathcal{R})$, where $\mathcal{B}$ is a behavior module and $\mathcal{R}$ is a control module.

The behavior module $\mathcal{B}$ is a union of n superposed TNCESs. For any $i \in \{0, \ldots, n-1\}$, the TNCES Γ_i is denoted by $\Gamma_i = (N_{\Gamma_i}, z_{0i})$ with $N_{\Gamma_i} = (P_i, T_i, F_i, CN_i, EN_i, em_i, DC_i)$. Then $\mathcal{B}$ can be represented as

$$\mathcal{B} = \left(\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{CN}, \mathcal{EN}, \mathcal{EM}, DC, \mathcal{Z}_0\right). \quad (3)$$

$\mathcal{P} = P_0 \cup P_1 \cup \cdots \cup P_{n-1}$ (resp., $\mathcal{T} = T_0 \cup T_1 \cup \cdots \cup T_{n-1}$) is a superset of places (resp., transitions). $\mathcal{F} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$ is a superset of flow arcs. $\mathcal{CN} \subseteq (\mathcal{P} \times \mathcal{T})$ (resp., $\mathcal{EN} \subseteq (\mathcal{T} \times \mathcal{T})$) is a superset of condition (resp., event) signals. $\mathcal{DC} = DC_0 \cup DC_1 \cup \cdots \cup DC_{n-1}$ is a set of time intervals to input flow arcs. $\mathcal{EM}: \mathcal{T} \to \{\mathbb{M}, \mathbb{A}\}$ maps an event processing mode (AND or OR) for each transition. Let $\mathcal{Z}_0 = (\mathcal{M}_0, \mathcal{D}_0)$, where $\mathcal{M}_0 : \mathcal{P} \to \{0, 1\}$ is the initial marking. $\mathcal{D}_0 : \mathcal{P} \to \{0\}$ is the initial clock position. $\Omega = \{N_{\Gamma_0}, N_{\Gamma_1}, \ldots, N_{\Gamma_{n-1}}\}$ is a set of all TNCES structures that can be represented by RN.

The control module $\mathcal{R}$ is a set of reconfiguration functions. A reconfiguration function r is a structure $r = (\text{Cond}, s, x)$. $\text{Cond} \to \{\text{true}, \text{false}\}$ is the precondition of r. $s : \Omega \to \Omega$ is the structure modification instruction. $x : R(N_{\Gamma i}, z_{0i}) \to \mathcal{Z}_{0j}$ is the state correlation function, where $\mathcal{Z}_{0j}$ is a set of feasible initial states of Γ_j. ${}^{\star}r = \Gamma_i = (N_{\Gamma i}, z_{0i})$ (resp., $r^{\star} = \Gamma_j = (N_{\Gamma j}, z_{0j})$) denotes the TNCES before (resp., after) the implementation of r.

Definition 2 (see [30]). A state $\mathcal{Z}$ of R-TNCES RN is a pair $[N_\Gamma, z]$, where $N_\Gamma \in \Omega$ identifies the activated TNCES with $N_\Gamma = (P, T, F, CN, EN, em, DC)$, and $z = (m, d)$ is a state of N_Γ with $m : P \to \{0, 1\}$ and $d : P \to \{0, 1, 2, \ldots\}$.

In a R-TNCES, each TNCES in the behavior module models a configuration. For a R-TNCES RN, only one of the TNCESs of the behavior module $\mathcal{B}$ is activated at the beginning until a reconfiguration function is implemented. Other TNCESs with net structures defined in Ω can be activated only after implementing reconfiguration functions. At any time, only one of the TNCESs with net structures defined in Ω is activated.

If a reconfiguration function $r = (\text{Cond}, s, x)$ meets its precondition, that is, Cond = True, it is enabled. A reconfiguration function can fire if it is enabled, that is, to implement it. The evolution of a R-TNCES depends on what events (reconfiguration functions or transitions) take place. Let Γ_i be the activated TNCES with $\Gamma_i = (N_{\Gamma i}, z_{0i})$, where $N_{\Gamma i} = (P_i, T_i, F_i, CN_i, EN_i, em_i, DC_i)$. If a maximal step $u \in T_i$ fires, Γ_i evolves from its one inner state to another. However, if a reconfiguration function r fires, then Γ_i is transformed into Γ_j by changing its net structure and updating its state, where ${}^{\star}r = \Gamma_i$, $r^{\star} = \Gamma_j$, and $\Gamma_j = (N_{\Gamma j}, z_{0j})$.

3.2. Drawbacks of R-TNCESs. The TNCES models for the four behavior modes of AAS are denoted by $\Gamma_1 = (N_{\Gamma_1}, z1_0)$, $\Gamma_2 = (N_{\Gamma_2}, z2_0)$, $\Gamma_3 = (N_{\Gamma_3}, z3_0)$, and $\Gamma_4 = (N_{\Gamma_4}, z4_0)$, respectively. The set of all possible reconfiguration events of AAS is marked by $\mathcal{R}$=$\{r_{1,s}, r_{2,s}, r_{3,s}, r_{4,s}, r_{1,s}^{-1}, r_{2,s}^{-1}, r_{3,s}^{-1}, r_{4,s}^{-1}, r_{1,2}, r_{1,3}, r_{1,4}, r_{2,4}, r_{3,4}\}$. The reconfiguration event $r_{i,s}$ indicates a local reconfiguration that transforms robot Rbi into its energy-efficient mode and $r_{i,s}^{-1}$ is the reverse of $r_{i,s}$, that is, to transform robot Rbi from its energy-efficient mode into its working mode. The implementation of the events $r_{i,s}$ and $r_{i,s}^{-1}$ does not change the current behavior mode but can switch robot Rbi between its working mode and energy-efficient mode according to its busy/idle status and waiting time. Finally, $r_{i,j}$ $(i \neq j)$ denotes a global reconfiguration event that transforms AAS from the configuration Mode i into Mode j.

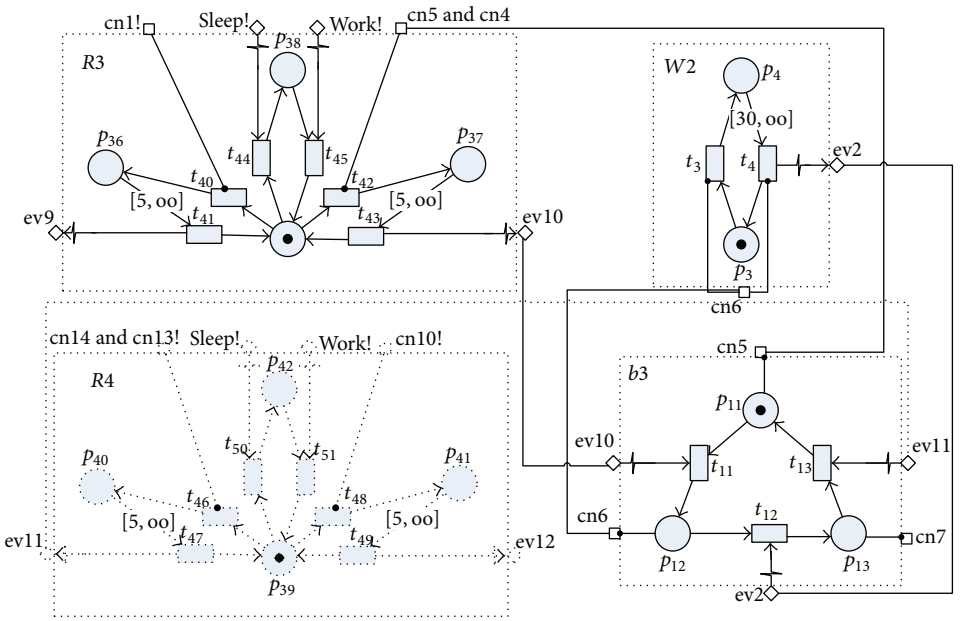

Figure 3: TNCES-based model of $R3$, $R4$, and $W2$ in Mode 1.

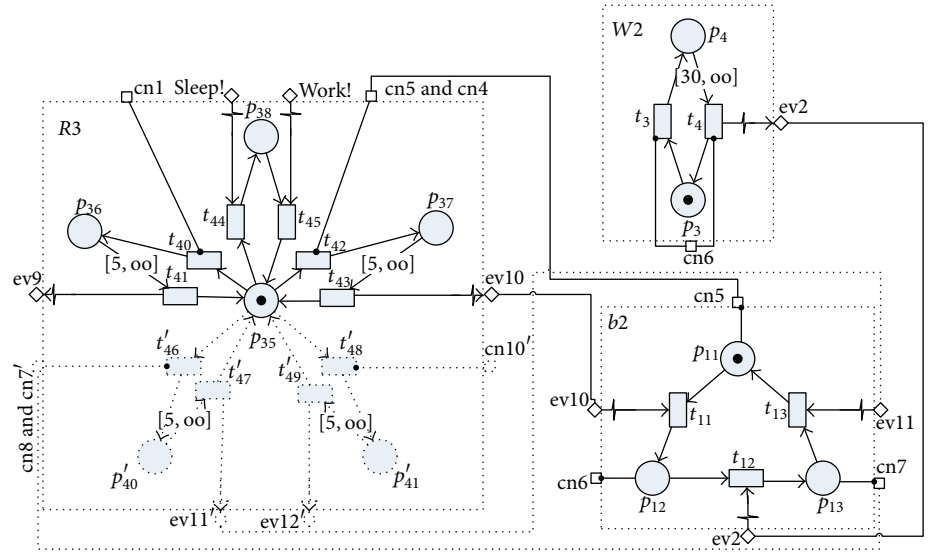

Figure 4: TNCES-based model of $R3$, $R4$, and $W2$ in Mode 4.

The firing of a reconfiguration function of a R-TNCES changes the system configuration. As a consequence, if reconfiguration functions are applied to model local reconfiguration events for switching components between their working modes and energy-efficient modes directly, the number of system configurations should be enlarged. For example, configuration Mode 4 should be considered as four different configurations: (1) Both Rb1 and Rb3 are in their working modes, (2) Rb1 is in working mode and Rb3 is in energy-efficient mode, (3) Rb3 is in working mode and Rb1 is in energy-efficient mode, and (4) both Rb1 and Rb3 are in their energy-efficient modes. These four configurations are with the same structure. However, they should be verified separately. Obviously, this increases the verification cost and burdens the whole design process.

Generally, transitions in a R-TNCES model normal events of a reconfigurable discrete event control system, whereas reconfiguration functions are used to model system reconfiguration events. However, the concurrence of reconfiguration functions and transitions is not allowed in R-TNCESs, which is in fact inconsistent with requirements of REMSs. To make it clearer, let us take the modules Rb3, Rb4, and $W2$ as an example. Their TNCES-based models in Mode 1 and Mode 4 are shown in Figures 3 and 4, respectively. The differences between them are marked by dotted lines.

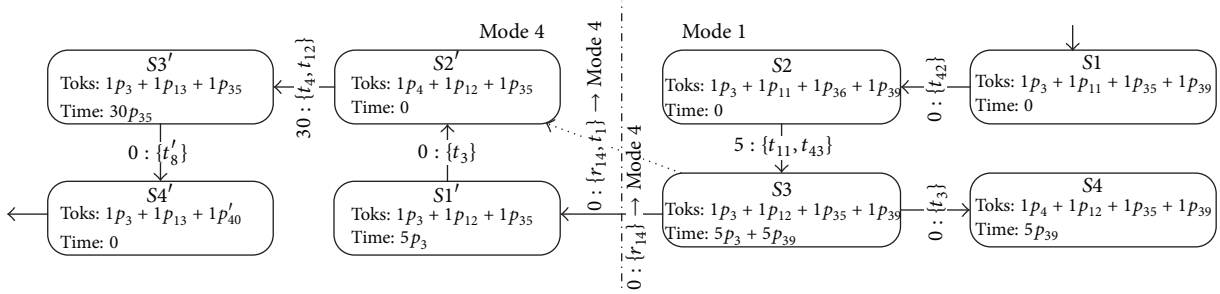

Figure 5: State transition graph of Example 3.

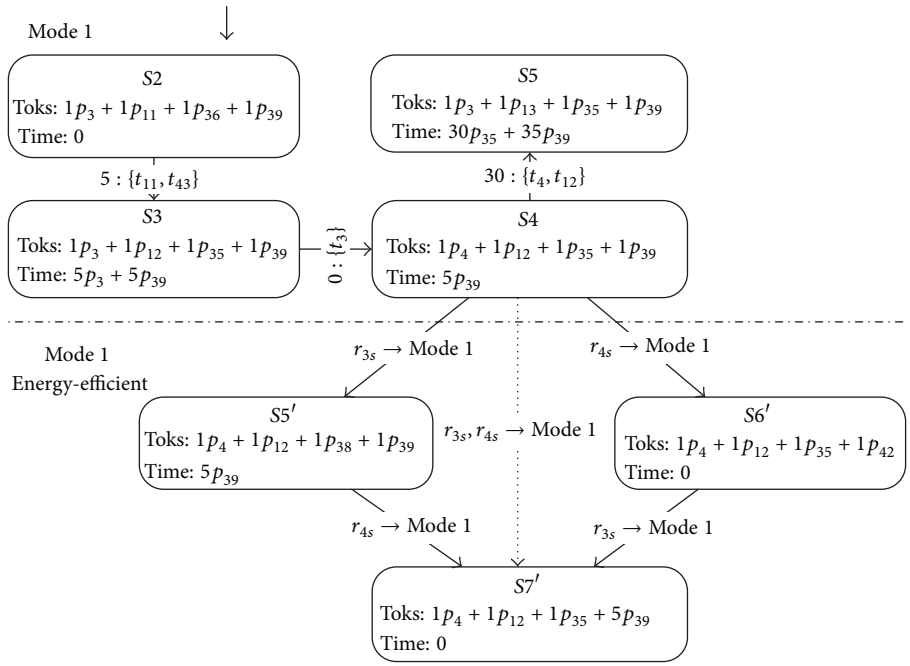

Figure 6: State transition graph of Example 4.

Example 3. Suppose that a reconfiguration function $r_{1,4}$ gets enabled at state $S3$ when AAS is in Mode 1. The physical meaning of $S3$ is that (1) Rb3 just finishes transporting S_A to $b2$ and (2) $W2$ is ready to process S_A. Assume that at this time a fault is detected in Rb4. Rb4 should be removed. Meanwhile, Rb3 must update itself soon in order to cover Rb4's task. According to the design requirements for AAS, $W2$ should go on working "naturally" at this time, that is, the enabled transition t_3 can fire at this state. However, the concurrence of reconfiguration functions and transitions is not allowed in R-TNCESs. Therefore, at state $S3$, only $r_{1,4}$ fires alone and AAS turns to the state $S1'$. Afterwards, t_3 fires, which leads to the next state $S2'$. However, if $r_{1,4}$ and t_3 fire together, AAS turns to the state $S2'$ directly without generating $S1'$. The state transition graph of this case is shown in Figure 5.

Example 4. Assume that two reconfiguration functions $r_{3,s}$ and $r_{4,s}$ get enabled simultaneously at state $S4$. The physical meaning of $S4$ is that (1) $W2$ just starts its work and (2) both Rb3 and Rb4 are idle. The firing of $r_{3,s}$ and $r_{4,s}$ only changes the states inside their modules but neither alters the system structure nor enables/disables any other transitions outside. That is to say, the firing of $r_{3,s}$ and $r_{4,s}$ does not change the current system configuration. According to the design requirements for AAS, both Rb3 and Rb4 can reconfigure themselves into energy-efficient modes freely when they are idle for more than two time units. However, the concurrence of multiple reconfiguration functions is not allowed in R-TNCESs. Therefore, at state $S4$, only $r_{4,s}$ or $r_{3,s}$ fires alone. After that, the remaining one fires since it is still enabled. However, if $r_{3,s}$ and $r_{4,s}$ fire together, AAS turns to the state $S7'$ directly. The state transition graph of this case is shown in Figure 6.

In conclusion, the original R-TNCESs are not sufficient to model a REMS. The reason can be explained from the following three aspects.

(i) Reconfigurations at the component level only change component behavior modes between their working modes and energy-efficient modes rather than

changing system configurations. If this kind of reconfigurations is modeled by reconfiguration functions directly, the number of system configurations should be enlarged, which increases the verification cost and burdens the whole design process.

(ii) The concurrence of reconfiguration functions and transitions is not allowed in R-TNCESs. However, from the above examples, the concurrence of reconfiguration events and normal events is a common phenomenon in a REMS.

(iii) Since the local reconfigurations for energy-efficient operations cannot be properly described, their corresponding dynamics and reasonable analysis cannot be performed.

To this end, this paper extends R-TNCESs to achieve two aims. First, all possible events including concurrent events that may occur in REMSs can be properly described. Second, the concurrence of reconfiguration functions and transitions should be controlled to ensure the system correctness.

3.3. Extended R-TNCESs. An extended R-TNCES has the same structure as the original R-TNCES. It is composed of a behavior module and a control module, denoted by $\text{eRN} = \{\mathscr{B}, \mathscr{R}\}$. The definition of system states is not changed, as shown in Definition 2 in Section 3.1. However, in the extended R-TNCES, reconfiguration functions are newly assigned with action ranges and concurrent decision functions. In addition, the firing rules of transitions and reconfiguration functions are updated such that they are conditionally allowed to fire concurrently.

3.3.1. Modified Reconfiguration Functions. In order to model the two types of reconfiguration events in a REMS directly, a concept, namely, *action range,* is developed for each reconfiguration function of a R-TNCES. In addition, a *concurrent decision function* is also assigned to a reconfiguration function to constrain concurrent transitions that may lead to undesired states such as deadlocks and overflow during a reconfiguration. For the sake of brevity, a reconfiguration function indicates a modified reconfiguration function in what follows.

Definition 5. A reconfiguration function r of an extended R-TNCES eRN is a structure $r = (\text{Cond}, s, x, \Lambda, \Pi)$. $\text{Cond} \to \{\text{true, false}\}$ is the precondition of r. $s: \Omega \to \Omega$ is the structure modification instruction. $x : R(N_{\Gamma i}, z_{0i}) \to \mathscr{Z}_{0j}$ is the state correlation function, where $\mathscr{Z}_{0j}$ is a set of feasible initial states of Γ_j. ${}^{\star}r = \Gamma_i = (N_{\Gamma i}, z_{0i})$ (resp., $r^{\star} = \Gamma_j = (N_{\Gamma j}, z_{0j})$) denotes the TNCES before (resp., after) r fires. $\Lambda \in (N_{\Gamma i} \cup N_{\Gamma j})$ denotes the action range of r. $\Pi(r, \mathscr{Z}) \to T$ is a concurrent decision function deciding a set of forbidden transitions that cannot fire together with r at state $\mathscr{Z}$.

The reconfiguration functions of extended R-TNCESs are divided into two types: major and minor reconfiguration functions. For a reconfiguration function $r = (\text{Cond}, s, x, \Lambda, \Pi)$ with ${}^{\star}r = \Gamma_i = (N_{\Gamma i}, z_{0i})$ and $r^{\star} = \Gamma_j = (N_{\Gamma j}, z_{0j})$, it is a major reconfiguration function if and only if $N_{\Gamma i} \neq N_{\Gamma j}$. Otherwise, it is a minor reconfiguration function. Let $\mathscr{R}_{\text{ma}}$ and $\mathscr{R}_{\text{mi}}$ denote the sets of major and minor reconfiguration functions of eRN, respectively. Then we have $\mathscr{R} = \mathscr{R}_{\text{ma}} \cup \mathscr{R}_{\text{mi}}$ and $\mathscr{R}_{\text{ma}} \cap \mathscr{R}_{\text{mi}} = \emptyset$.

The implementation (firing) of a major reconfiguration function changes the structure of the current activated TNCES, whereas the implementation (firing) of a minor reconfiguration function only adjusts partial states of the activated TNCES within its action range.

Similar to Petri nets, the "conflict" concept is proposed for two enabled reconfiguration functions. We have the following two cases.

(1) For two reconfiguration functions within the same type, that is, both being minor or major reconfiguration functions, if their action ranges have intersections, they are conflicting.

(2) For a minor reconfiguration function and a major reconfiguration function, if the action range of the minor reconfiguration function is not completely covered by that of the major reconfiguration function, they are conflicting.

If two reconfiguration functions are conflicting, they cannot be implemented simultaneously. The symbol $r_1 \parallel r_2$ denotes that reconfiguration functions r_1 and r_2 are not conflicting.

Similar to the definition of *steps* in TNCES, a r-step in an extended R-TNCES is a maximal set of reconfiguration functions that can fire simultaneously at a particular state. A r-step should satisfy the following two conditions.

(1) For any two reconfiguration functions r_i and r_j ($r_i \neq r_j$) in a r-step γ, r_i and r_j are not conflicting; that is, $r_i \parallel r_j$.

(2) There does not exist any other maximal set of reconfiguration functions γ' such that $\gamma \subset \gamma'$.

Accordingly, two r-steps γ_1 and γ_2 are conflicting, if $\exists r_1 \in \gamma_1$, $\exists r_2 \in \gamma_2$, $r_1 \neq r_2$, r_1 and r_2 are conflicting.

3.3.2. Dynamics of Extended R-TNCESs. Suppose that, at state $\mathscr{Z} = \lceil N_\Gamma, z \rfloor$, multiple reconfiguration functions get enabled, to be denoted by

$$R^* = \gamma_1 \cup \gamma_2 \cup \cdots \cup \gamma_g, \tag{4}$$

where γ_i ($i \in [1, g]$) is a maximal r-step at $\mathscr{Z}$ and, for all i, $j \in [1, g]$, $i \neq j$, γ_i and γ_j are conflicting. At the same state $\mathscr{Z}$, the set of all enabled transitions is denoted by

$$T^* = u_1 \cup u_2 \cup \cdots \cup u_k, \tag{5}$$

where u_i ($i \in [1, k]$) is a maximal step and, for all i, $j \in [1, k]$, $i \neq j$, u_i and u_j are conflicting. For more information on how these steps are computed, please see [34, 35].

As a consequence, different compositions of r-steps and steps can occur simultaneously at this state. Given an enabled reconfiguration function r, we use $\mathscr{D} \cdot T$ (resp., $\mathscr{D} \cdot P$) to denote the set of deleted transitions (resp., deleted places) and $\mathscr{A} \cdot T$ (resp., $\mathscr{A} \cdot P$) to denote the set of added transitions

(resp., added places) by firing it, where ${}^{\star}r = \Gamma_i = (N_{\Gamma i}, z_{0_i})$, $r^{\star} = \Gamma_j = (N_{\Gamma j}, z_{0_j})$, $N_{\Gamma i} = (P_i, T_i, F_i, CN_i, EN_i, em_i, DC_i)$, and $N_{\Gamma j} = (P_j, T_j, F_j, CN_j, EN_j, em_j, DC_j)$. We have the following two cases.

(1) For a transition $t \in T_i$, if it is enabled simultaneously with a minor reconfiguration function $r = (\text{Cond}, s, x, \Lambda, \Pi)$ at state $\mathscr{Z} = \lceil N_\Gamma, z \rfloor$ and $t \notin \Lambda$, then t can fire simultaneously with r; that is, $t \notin \Pi(r, \mathscr{Z})$.

(2) For a transition $t \in T_i$, if it is enabled simultaneously with a major reconfiguration function $r = (\text{Cond}, s, x, \Lambda, \Pi)$ at state $\mathscr{Z} = \lceil N_\Gamma, z \rfloor$, then we have the following two subcases.

(A) A spontaneous transition t is forbidden to be concurrent with r at $\mathscr{Z}$, if it meets one of the following conditions.

(i) If it is deleted by r, that is, $t \in \mathscr{D} \cdot T$, it is forbidden by r; that is, $t \in \Pi(r, \mathscr{Z})$.

(ii) If $t \notin \mathscr{D} \cdot T$ and all its elements are not changed by firing r, then it is allowed to fire simultaneously with r. Formally, if ${}^{\bullet}t_i = {}^{\bullet}t_j$, $t_i^{\bullet} = t_j^{\bullet}$, ${}^{-}t_i = {}^{-}t_j$, ${}^{\sim}t_i = {}^{\sim}t_j = \emptyset$, and $em(t)_i = em(t)_j$, we have $t \notin \Pi(r, \mathscr{Z})$.

(iii) If $t \notin \mathscr{D} \cdot T$, some of its elements are modified by r, which include its preset, postset, source places, and firing mode, and we have the following two cases.

(a) The preset, source places, and firing mode of t decide whether t is enabled after the firing of r. Therefore, if its preset, source places, or firing mode is changed by r, it can fire simultaneously with r. Formally, if ${}^{\bullet}t_i \neq {}^{\bullet}t_j$, ${}^{-}t_i \neq {}^{-}t_j$, or $em(t)_i \neq em(t)_j$, then $t \notin \Pi(r, \mathscr{Z})$.

(b) The postset of t does not change its enabling condition but influences the structure of the net. Therefore, it is forbidden by r. Formally, if ${t^{\bullet}}_i \neq {t^{\bullet}}_j$, we have $t \in \Pi(r, \mathscr{Z})$.

(B) A forced transition t is forbidden to be concurrent with r at $\mathscr{Z}$, if it further meets one of the following conditions.

(i) Its firing mode is $\mathbb{M}$ and all of its forcing transitions are forbidden to be concurrent with r; that is, if $em_i(t) = em_j(t) = \mathbb{M}$ and, for all $t' \in {}^{\sim}t, t' \notin \Pi(r, \mathscr{Z})$, then $t \in \Pi(r, \mathscr{Z})$.

(ii) Its firing mode is $\mathbb{N}$ and at least one of its forcing transitions is forbidden by r; that is, if $em_i(t) = em_j(t) = \mathbb{N}$ and $\exists t' \in {}^{\sim}t$, $t' \notin \Pi(r, \mathscr{Z})$, then $t \in \Pi(r, \mathscr{Z})$.

Since an extended R-TNCES allows the concurrence of multiple reconfiguration functions and transitions, the reachability graph of an extended R-TNCES is defined as follows.

Definition 6. The reachability graph of an extended R-TNCES eRN is a combination of several labeled directed graphs whose nodes are the states of eRN and whose arcs are of three kinds: steps, r-steps, and combinations of a step and a r-step.

(i) The arc from state $\lceil N_{\Gamma i}, z_i \rfloor$ to state $\lceil N_{\Gamma i}, {z'}_i \rfloor$ is denoted by a step u represented by $\lceil N_{\Gamma i}, z_i \rfloor [u\rangle \lceil N_{\Gamma i}, {z'}_i \rfloor$, where ${z'}_i \in R(N_{\Gamma i}, z_i)$.

(ii) The arc from state $\lceil N_{\Gamma i}, z_i \rfloor$ to state $\lceil N_{\Gamma j}, z_j \rfloor$ is labeled with a r-step γ represented by $\lceil N_{\Gamma i}, z_i \rfloor [\gamma\rangle \lceil N_{\Gamma j}, z_j \rfloor$, if γ contains major reconfiguration functions with $N_{\Gamma i} \neq N_{\Gamma j}$. Otherwise, we have $N_{\Gamma i} = N_{\Gamma j}$ and $z_j \notin R(N_{\Gamma i}, z_i)$.

(iii) The arc from $\lceil N_{\Gamma i}, z_i \rfloor$ to state $\lceil N_{\Gamma j}, z_j \rfloor$ is labeled with a step and a r-step $\{\mathscr{R}, u\}$ represented by $\lceil N_{\Gamma i}, z_i \rfloor [\gamma, u\rangle \lceil N_{\Gamma j}, z_j \rfloor$, if γ contains major reconfiguration functions with $N_{\Gamma i} \neq N_{\Gamma j}$. Otherwise, we have $N_{\Gamma i} = N_{\Gamma j}$.

Obviously, the graphical representation of an extended R-TNCES model is the same as that of a R-TNCES model. However, system dynamics get enriched along with the changes of reconfiguration functions. If we use an extended R-TNCES to model AAS, the graphical TNCES models shown in Figures 3 and 4 are still correct. However, their reachability graphs get enriched during same reconfiguration.

Example 7. A fragment of the reachability graph of the extended R-TNCES-based model of the example composed of Rb3, Rb4, and $W2$ is shown in Figure 7. AAS starts running in *Mode* 1. When it arrives at state $S3$, two minor reconfiguration functions $r_{3,s}$ and $r_{4,s}$ get enabled and fire simultaneously to reconfigure robots Rb3 and Rb4 into their energy-efficient modes. After 28 time units, they reconfigure back to working modes. Assume that $R4$ is detected to have a fault at state $S10$, the major reconfiguration function $r_{1,4}$ gets enabled. In the meantime, $t3$ gets enabled simultaneously with $r_{1,4}$. Therefore, $t3$ fires simultaneously with $r_{1,4}$, which leads to the transformation of AAS into *Mode* 4.

4. Verification of REMSs Based on Extended R-TNCESs

In order to perform correct formal verification of AAS, an extended R-TNCES-based model should be built for it. The extended R-TNCES based model of AAS is marked by $\text{eRN}_{\text{AAS}} = \{\mathscr{B}, \mathscr{R}\}$, $\mathscr{B} = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4$, $\mathscr{R} = \mathscr{R}_{\text{ma}} \cup \mathscr{R}_{\text{mi}}$, $\mathscr{R}_{\text{ma}} = \{r_{1,2}, r_{1,3}, r_{1,4}, r_{2,4}, r_{3,4}\}$, and $\mathscr{R}_{\text{mi}} = \{r_{1,s}, r_{2,s}, r_{3,s}, r_{4,s}, r_{1,s}^{-1}, r_{2,s}^{-1}, r_{3,s}^{-1}, r_{4,s}^{-1}\}$. We have $\Omega = \{N_{\Gamma_1}, N_{\Gamma_2}, N_{\Gamma_3}, N_{\Gamma_4}\}$. The four major reconfiguration functions are conflicting with each other. The minor reconfiguration functions $r_{i,s}$ and $r_{i,s}^{-1}$ ($i \in [1, 4]$) are conflicting but others are not. The behavior module of eRN_{AAS} is shown in Figure 8, where elements drawn by dotted lines are possibly modified during the implementation of a major reconfiguration function. In order to apply automatic model checking to an extended R-TNCES, a TNCES-based nested state machine is developed to implement its control module.

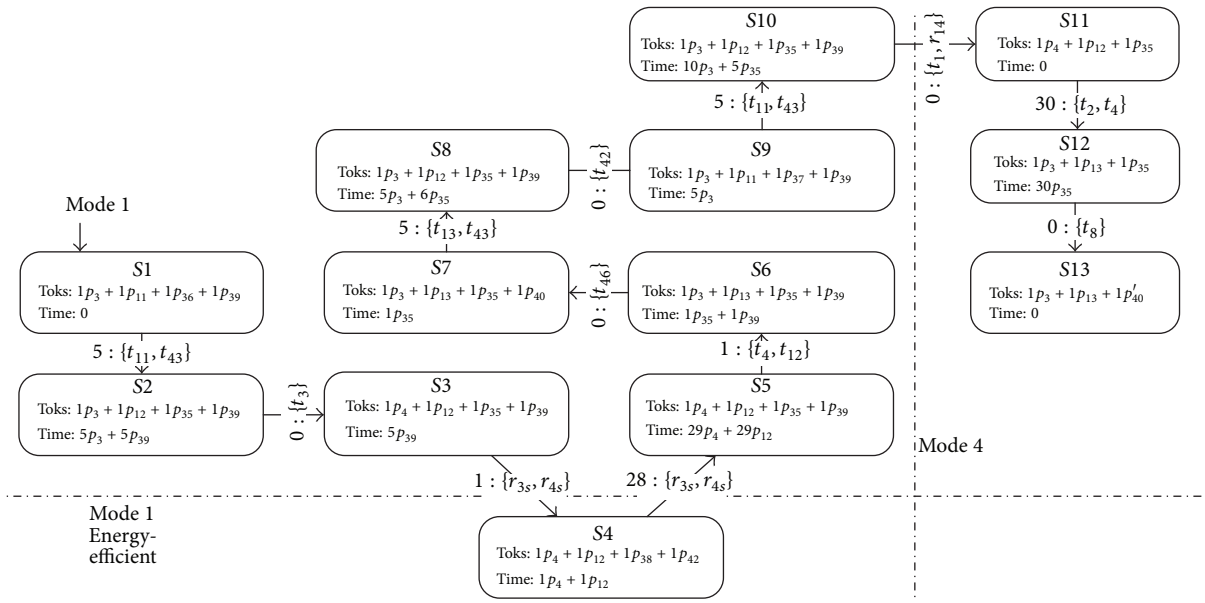

FIGURE 7: A fragment of the reachability graph of Example 7.

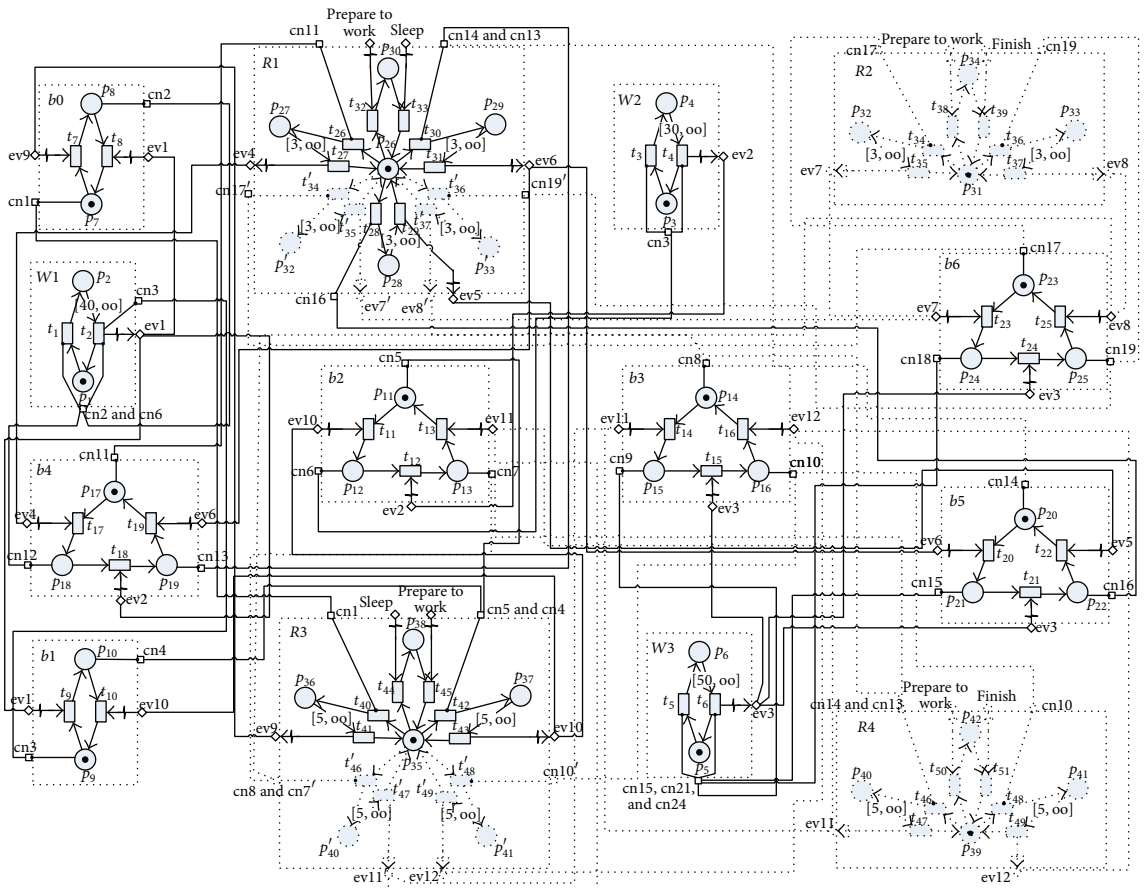

FIGURE 8: Behavior module of eRN_{AAS}.

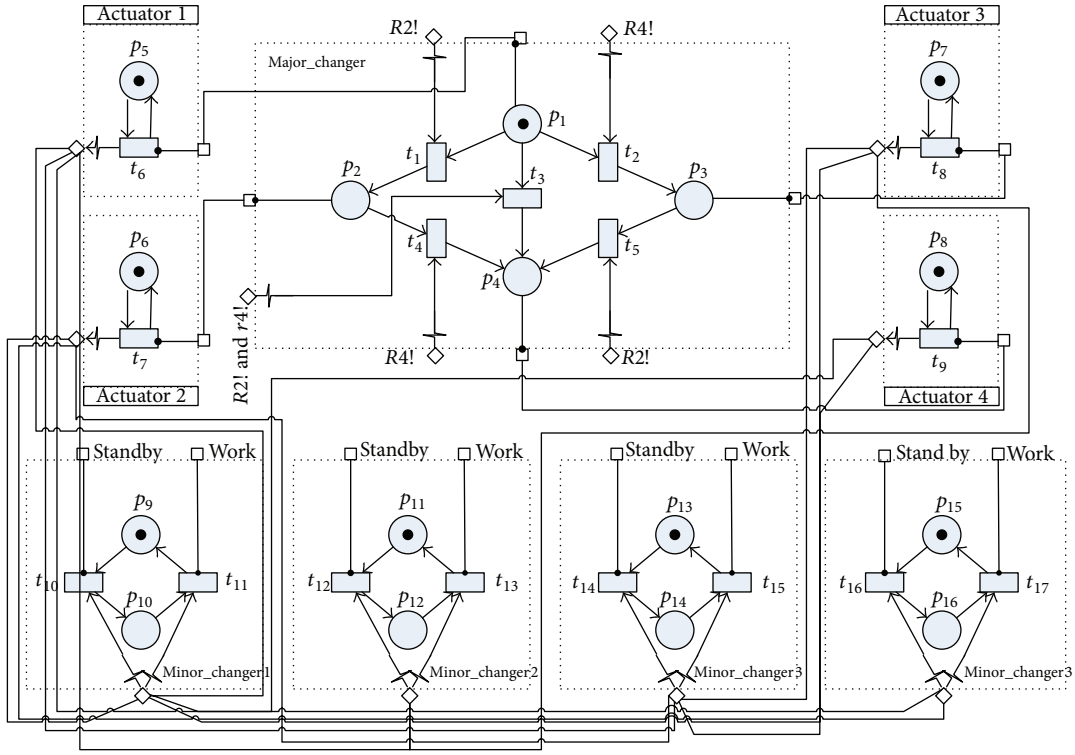

Figure 9: Control module of eRN_{AAS}.

4.1. Implementation of Extended R-TNCESs. First of all, major reconfiguration functions are grouped according to their action ranges. A set of state machines specified by TNCESs, which are called Major_changers, is defined. Each state machine corresponds to a group of major reconfiguration functions that share the same action range. In a particular Major_changer, each transition corresponds to a major reconfiguration function. The transitions in a state machine cannot fire simultaneously, which means that these modeled major reconfiguration functions by one state machine are conflicting with each other. Firing a transition st in a Major_changer implies that a major reconfiguration function is implemented. A Major_changer is formalized as follows:

$$\text{Major\_changer} = (P, T, F, em, z_0), \tag{6}$$

where, for any $t \in T$, $|{}^\bullet t| = |t^\bullet| = 1$, $\sum M_0(P) = 1$, which means that only one place in P owns a token at the initial state, and $em : T \to \{\boxtimes\}$. The precondition Cond can be modeled by input event/condition signals from external to transitions in a Major_changer.

In addition, an actuator denoted by Actuator is defined for each place sp in all Major_changers, which is marked by Actuator = Act(sp). Each actuator is composed of a place ap and a transition at only, where ${}^\bullet ap = ap^\bullet = \{at\}$, ${}^\bullet at = at^\bullet = \{ap\}$, and $M(ap) = 1$. When the place sp in a Major_changer receives a token, the actuator Actuator = Act(sp) is activated. An Actuator is formalized as follows:

$$\text{Actuator} = (P, T, F, em, z_0), \tag{7}$$

where $|P| = |T| = 1$, ${}^\bullet at = at^\bullet = \{ap\}$, ${}^\bullet ap = ap^\bullet = \{at\}$, $m_0(P) = 1$, and $em : T \to \{\boxtimes\}$.

Similar to major reconfiguration functions, minor reconfiguration functions are grouped according to their action ranges. A set of state machines specified by TNCESs, which are called Minor_changers, is defined. Each state machine corresponds to a group of minor reconfiguration functions. If the action ranges of two minor reconfiguration functions are the same, they are modeled by transitions in a Minor_changer. If the action range of a group of minor reconfiguration functions, to be modelled by a Minor_changer, is completely covered by that of a group of major reconfiguration functions, to be modeled by a Major_changer, then this Minor_changer is activated while this Major_changer is activated.

A Minor_changer is formalized as follows:

$$\text{Minor\_changer} = (P, T, F, em, z_0), \tag{8}$$

where, for any $t \in T$, $|{}^\bullet t| = |t^\bullet| = 1$, $\sum M_0(P) = 1$, which means that only one place in P owns a token at the initial state, and $em : T \to \{\boxtimes\}$. The precondition Cond can be modeled by input event/condition signals from external to transitions in a Minor_changer.

Example 8. Figure 9 depicts the TNCES-based control module of eRN_{AAS}. It has only one Major_changer, since the four major reconfiguration functions share the same action range. It has four Minor_changers, since the four robots have four distinguished action ranges. Places p_1, p_2, p_3, and p_4 in Major_changer correspond to Mode 1, Mode 2, Mode 3, and

Table 1: Time of robots on their energy-efficient modes.

Configuration	Mode 1				Mode 2			Mode 4	
System uptime	6690				4127			1525	
Robot	Rb1	Rb2	Rb3	Rb4	Rb1	Rb3	Rb4	Rb1	Rb3
Time on energy-efficient mode	3233	4455	3818	2643	1004	2458	2428	523	435

Mode 4, respectively. When t_3 fires, the major reconfiguration function $r_{1,4}$ is implemented. Robots Rb3 and Rb1 are applied in every mode of AAS. Therefore, minor reconfiguration functions that transform them between energy-efficient modes and working modes are activated in every system behavior mode. Moreover, it is possible for them to fire simultaneously with other major reconfiguration functions.

4.2. Formal Verification of AAS. Since the time when a major reconfiguration function can get enabled and fire cannot be predicted, this paper applies an instruction insertion method to simulate AAS. In addition, eRN_{AAS} evolves according to fired maximal steps and r-steps. Assume that AAS should finish 100 subassemblies. It starts with Mode 1. At time t_1 when it finishes the 60th subassembly, it reconfigures into Mode 2 due to the fault detection of Rb2. Then, it goes on working in Mode 2. At time t_2 when the 91st subassembly is being processed, it transforms into Mode 4 according to the fault detection of Rb4. During the whole process, minor reconfigurations, that is, transforming robots between their working modes and energy-efficient modes, are applied.

SESA is applied to compute the reachability graph of this whole process. A minimal path regarding time consumption from the initial state to the objective state is computed in each mode. In Mode 1, it generates 23044 states, taking 6990 time units to finish assembly of the first 60 subassemblies in the minimal path. In Mode 2, it generates 85259 states, costing 4127 time units to finish assembling the next 30 subassemblies in the minimal path. Finally, in Mode 4, it generates 195007 states, taking 1525 time units to finish assembling the last 10 subassemblies in the minimal path. Note that two states can be considered to be same if and only if they have the same token numbers and time status.

Since each TNCES-based model of the behavior modes of AAS is a well-designed control system, they are proved to be qualified according to SESA, where eCTL based functional properties and TCTL based temporal properties are checked. In addition, the following eCTL formula is applied to the control module of eRN_{AAS}:

$$Z_0 = EX\left\langle t4ANDt12\right\rangle X\left\langle p12 = 1\right\rangle. \tag{9}$$

This formula is proved to be false by SESA. Transition t_{12} corresponds to minor reconfiguration function $r_{2,s}$. Therefore, it can fire only when AAS is in Mode 1 or Mode 2. The following formula is proved to be true:

$$Z_0 = EX\left\langle t2ANDt10\right\rangle X\left\langle p10 = 1\right\rangle. \tag{10}$$

It means that when robot Rb4 breaks down, two reconfiguration functions $r_{1,3}$ and $r_{1,s}$ are possible to fire simultaneously.

The triggering conditions of minor reconfiguration functions can be computed previously. There are several possible state/event paths showing system behavior from the initial state to the objective state, at which 100 subassemblies are finished. We select a minimal path regarding time for each TNCES-based model of the three configurations, to be denoted by $\text{Path} = \mathscr{Z}_1, \mathscr{Z}_2, \ldots, \mathscr{Z}_n$, where energy-efficient operations are not included. That is to say, all robots should stay in their working modes in this case although they should wait for a period of time before the next task comes. After that, based on the states on this path, the time when a minor reconfiguration function gets enabled and fires can be computed. For example, if an activated robot starts to wait at a particular state $\mathscr{Z}_i$, at which the system time is τ_1, a search is performed along this minimal path at τ_1. If it is found that at $\mathscr{Z}_j$ the robot works again, at which the system time is τ_2, then the time delay $\Delta\tau = \tau_2 - \tau_1$ between these two states is obtained. The round local reconfigurations for switching a robot between its working mode and energy-efficient mode take two time units. Therefore, if the time delay is larger than two, that is, $\Delta\tau > 2$, a local reconfiguration can be applied to this robot. The system time for reconfiguring this robot from its working mode to its energy-efficient mode is τ_1. The system time for reconfiguring this robot from its energy-efficient mode to its working mode is $\tau_2 - 1$.

The time of robots on their energy-efficient modes in minimal paths is computed during the assembly of 100 subassemblies. They are shown in Table 1 together with the whole system uptime in each mode. Take Mode 1 as an example. Assume that Rb1 consumes one energy unit per time unit in its working time but only consumes 30% energy unit per time unit in its energy-efficient mode. In *Mode* 1, if there is no minor reconfiguration applied to Rb1 for saving energy, it will consume 6990 energy units. However, it only consumes $6990 - 3233 + 30\% \times 3233 = 4726.9$ energy units in Mode 1 if minor reconfigurations are applied when it is idle. In the same way, the energy saved by the robots during this simulation is shown in Table 2, where the third row shows the energy consumption of each robot if no minor reconfigurations are applied, the fourth row shows the energy consumption of each robot when minor reconfigurations are applied, and the last row shows the saved energy of each robot during this process.

5. Conclusion

A reconfigurable and energy-efficient manufacturing system (REMS) is a typical reconfigurable discrete event control system. It allows two kinds of dynamic system reconfigurations: local and global reconfigurations. The former ones are

Table 2: Energy consumption of robots.

Configuration	Mode 1				Mode 2			Mode 4	
Robot	Rb1	Rb2	Rb3	Rb4	Rb1	Rb3	Rb4	Rb1	Rb3
Energy 1	6690	6690	6690	6690	4127	4127	4127	1525	1525
Energy 2	4726.9	3871.5	4303.4	5139.9	3424.2	2406.4	2427.4	1158.9	1220.5
Saved energy	1963.1	2818.5	2386.6	1550.1	702.8	1720.6	1699.6	366.1	304.5

applied to save energy for components, whereas the latter ones are applied to change system configurations according to changed inner/outer execution environments. Meanwhile, normal events should be conditionally allowed to occur simultaneously with these system reconfigurations, such that the system can reconfigure smoothly and safely. In order to easily model conditioned concurrence of reconfiguration events and normal events and represent all interesting system behavior, this paper extends the reconfigurable timed net condition event systems (R-TNCESs) formalism. Original reconfiguration functions are newly assigned with action ranges and concurrent decision functions. Accordingly, the dynamics of R-TNCES is updated. After that, a TNCES-based implementation method for the proposed extended R-TNCES is developed such that automatic model checking can be applied. The verified properties include functional, temporal, and energy properties that are specified by Computation Tree Logic (CTL), extended Computation Tree Logic (eCTL), or Timed Computation Tree Logic (TCTL). An automatic assembly system is used to illustrate the whole work.

In the future, the authors will focus on reasonably optimal reconfigurable control systems that can save more energy and the applications of the proposed method to the crude-oil operation enterprises with huge energy consumption [48].

Conflict of Interests

The authors declare that there is no conflict of interests for this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant no. 61374068 and the Science and Technology Development Fund, MSAR, under Grant nos. 065/2013/A2 and 066/2013/A2.

References

[1] Y. Koren, U. Heisel, F. Jovane et al., "Reconfigurable manufacturing systems," *CIRP Annals—Manufacturing Technology*, vol. 48, no. 2, pp. 527–540, 1999.

[2] M. Khalgui, O. Mosbahi, J. F. Zhang, Z. W. Li, and A. Gharbi, "Feasible dynamic reconfigurations of petri nets: application to a production systems," in *Proceedings of the 6th International Conference on Software and Database Technologies (ICSOFT '11)*, pp. 105–110, Sevilla, Spain, July 2011.

[3] T. Parisini and S. Sacone, "Fault diagnosis and controller reconfiguration: an hybrid approach," in *Proceedings of the IEEE International Symposium on Intelligent Control (ISIC '98)*, pp. 163–168, September 1998.

[4] P. Leitão, J. Alves, J. M. Mendes, and A. W. Colombo, "Energy aware knowledge extraction from petri nets supporting decision-making in service-oriented automation," in *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE '10)*, pp. 3521–3526, Bari, Italy, July 2010.

[5] S. Karnouskos, A. W. Colombo, J. L. M. Lastra, and C. Popescu, "Towards the energy efficient future factory," in *Proceedings of the 7th IEEE International Conference on Industrial Informatics (INDIN '09)*, pp. 367–371, Cardiff, Wales, June 2009.

[6] K. Bunse, M. Vodicka, P. Schönsleben, M. Brülhart, and F. O. Ernst, "Integrating energy efficiency performance in production management—gap analysis between industrial needs and scientific literature," *Journal of Cleaner Production*, vol. 19, no. 6-7, pp. 667–679, 2011.

[7] S. Mechs, S. Lamparter, and J. P. Müller, "On evaluation of alternative switching strategies for energy-efficient operation of modular factory automation systems," in *Proceedings of the IEEE 17th International Conference on Emerging Technologies and Factory Automation (ETFA '12)*, pp. 1–8, IEEE, Kraków, Poland, September 2012.

[8] S. Mechs, J. P. Muller, S. Lamparter, and J. Peschke, "Networked priced timed automata for energy-efficient factory automation," in *Proceedings of the American Control Conference (ACC '12)*, pp. 5310–5317, Montreal, Canada, June 2012.

[9] Z. M. Bi and L. Wang, "Optimization of machining processes from the perspective of energy consumption: a case study," *Journal of Manufacturing Systems*, vol. 31, no. 4, pp. 420–428, 2012.

[10] Y. Oda, Y. Kawamura, and M. Fujishima, "Energy consumption reduction by machining process improvement," *Procedia CIRP*, vol. 4, pp. 120–124, 2012.

[11] A. Cannata, S. Karnouskos, and M. Taisch, "Energy efficiency driven process analysis and optimization in discrete manufacturing," in *Proceedings of the 35th Annual Conference of the IEEE Industrial Electronics Society(IECON '09)*, pp. 4449–4454, Porto, Portugal, November 2009.

[12] G. Mouzon and M. B. Yildirim, "A framework to minimise total energy consumption and total tardiness on a single machine," *International Journal of Sustainable Engineering*, vol. 1, no. 2, pp. 105–116, 2008.

[13] D. Shorin and A. Zimmermann, "Model-based development of energy-efficient automation systems," in *Proceedings of the 17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '11)*, Chicago, Ill, USA, April 2011.

[14] C.-W. Park, K.-S. Kwon, W.-B. Kim et al., "Energy consumption reduction technology in manufacturing—a selective review of policies, standards, and research," *International Journal of Precision Engineering and Manufacturing*, vol. 10, no. 5, pp. 151–173, 2009.

[15] P. Stoffels, W. M. Boussahel, M. Vielhaber, and G. Frey, "Energy engineering in the virtual factory," in *Proceedings of the IEEE 18th International Conference on Emerging Technologies and Factory Automation (ETFA '13)*, pp. 1–6, Cagliari, Italy, September 2013.

[16] N. Wu, M. Zhou, and Z. Li, "Resource-oriented Petri net for deadlock avoidance in flexible assembly systems," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 38, no. 1, pp. 56–69, 2008.

[17] Z. Li and M. Zhou, "Two-stage method for synthesizing liveness-enforcing supervisors for flexible manufacturing systems using Petri nets," *IEEE Transactions on Industrial Informatics*, vol. 2, no. 4, pp. 313–325, 2006.

[18] Z. W. Li, H. S. Hu, and A. R. Wang, "Design of liveness-enforcing supervisors for flexible manufacturing systems using Petri nets," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 37, no. 4, pp. 517–526, 2007.

[19] Z. Li and M. Zhou, "Control of elementary and dependent siphons in Petri nets and their application," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 38, no. 1, pp. 133–148, 2008.

[20] Z. Li, M. Zhou, and N. Wu, "A survey and comparison of Petri net-based deadlock prevention policies for flexible manufacturing systems," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 38, no. 2, pp. 173–188, 2008.

[21] Z. Li, N. Wu, and M. Zhou, "Deadlock control of automated manufacturing systems based on petri nets—a literature review," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 42, no. 4, pp. 437–462, 2012.

[22] Z. Y. Ma, Z. W. Li, and A. Giua, "Design of optimal Petri net controllers for disjunctive generalized mutual exclusion constraints," *IEEE Transactions on Automatic Control*, 2015.

[23] J. H. Ye, Z. W. Li, and A. Giua, "Decentralized supervision of Petri nets with a coordinator," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 6, pp. 955–966, 2015.

[24] Z. W. Li, M. C. Zhou, and M. D. Jeng, "A maximally permissive deadlock prevention policy for FMS based on petri net siphon control and the theory of regions," *IEEE Transactions on Automation Science and Engineering*, vol. 5, no. 1, pp. 182–188, 2008.

[25] Z. W. Li, G. Y. Liu, H.-M. Hanisch, and M. C. Zhou, "Deadlock prevention based on structure reuse of petri net supervisors for flexible manufacturing systems," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 42, no. 1, pp. 178–191, 2012.

[26] Y. F. Chen and Z . W. Li, "Design of a maximally permissive liveness-enforcing supervisor with a compressed supervisory structure for flexible manufacturing systems," *Automatica*, vol. 47, no. 5, pp. 1028–1034, 2011.

[27] Y. F. Chen, Z. W. Li, M. Khalgui, and O. Mosbahi, "Design of a maximally permissive liveness- enforcing Petri net supervisor for flexible manufacturing systems," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 2, pp. 374–393, 2011.

[28] Y. F. Chen, Z. W. Li, K. Barkaoui, and M. Uzam, "New Petri net structure and its application to optimal supervisory control: Interval inhibitor arcs," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 10, pp. 1384–1400, 2014.

[29] X. Wang, I. Khemaissia, M. Khalgui, Z. Li, O. Mosbahi, and M. Zhou, "Dynamic low-power reconfiguration of real-time systems with periodic and probabilistic tasks," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 258–271, 2015.

[30] J. Zhang, M. Khalgui, Z. Li, O. Mosbahi, and A. M. Al-Ahmari, "R-TNCES: a novel formalism for reconfigurable discrete event control systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems and Humans*, vol. 43, no. 4, pp. 757–772, 2013.

[31] J. F. Zhang, M. Khalgui, Z. W. Li, G. Frey, O. Mosbahi, and H. B. Salah, "Reconfigurable coordination of distributed discrete event control systems," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 1, pp. 323–330, 2015.

[32] C. Gerber, S. Preuße, and H.-M. Hanisch, "A complete framework for controller verification in manufacturing," in *Proceedings of the 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA '10)*, pp. 1–9, September 2010.

[33] C. Gerber, *Implementation and Verification of Distributed Control Systems*, Logos, Berlin, Germany, 2011.

[34] H.-M. Hanisch, J. Thieme, A. Lueder, and O. Wienhold, "Modeling of PLC behavior by means of timed net condition/event systems," in *Proceedings of the IEEE 6th International Conference on Emerging Technologies and Factory Automation (ETFA '97)*, pp. 391–396, IEEE, Los Angeles, Calif, USA, September 1997.

[35] M. Rausch and H. M. Hanisch, "Net condition/event systems with multiple condition outputs," in *Proceedings of the 1995 INRIA/IEEE Symposium on Emerging Technologies and Factory Automation*, pp. 592–600, Paris, France, October 1995.

[36] T. Murata, "Petri nets: properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[37] Y. F. Chen, Z. W. Li, and M. C. Zhou, "Optimal supervisory control of flexible manufacturing systems by Petri nets: a set classification approach," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 549–563, 2014.

[38] Z. W. Li and M. C. Zhou, "Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems"," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans.*, vol. 34, no. 1, pp. 38–51, 2004.

[39] Z. W. Li and M. C. Zhou, "Clarifications on the definitions of elementary siphons in Petri nets," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 36, no. 6, pp. 1227–1229, 2006.

[40] Z. Li and M. Zhao, "On controllability of dependent siphons for deadlock prevention in generalized Petri nets," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 38, no. 2, pp. 369–384, 2008.

[41] P. H. Starke and S. Roch, "Analysing signal-net systems," Tech. Rep., Informatik Berichte, Humboldt-University, Berlin, Germany, 2002.

[42] E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*, MIT Press, 1999.

[43] E. A. Emerson, A. K. Mok, A. P. Sistla, and J. Srinivasan, "Quantitative temporal reasoning," in *Computer-Aided Verification*, vol. 531 of *Lecture Notes in Computer Science*, pp. 136–145, Springer, Berlin, Germany, 1991.

[44] S. Roch, "Extended computation tree logic," in *Proceedings of the Informatik-Bericht Workshop on Concurrency, Specification and Programming*, pp. 225–234, 2000.

[45] S. Preuße, D. Missal, C. Gerber, M. Hirsch, and H. M. Hanisch, "On the use of model-based IEC 61499 controller design," *International Journal of Discrete Event Control Systems*, vol. 1, no. 1, pp. 115–128, 2011.

[46] S. Preuse, H.-C. Lapp, and H.-M. Hanisch, "Closed-loop system modeling, validation, and verification," in *Proceedings of the IEEE 17th International Conference on Emerging Technologies and Factory Automation (ETFA '12)*, pp. 1–8, IEEE, Kraków, Poland, September 2012.

[47] S. Preußse and H.-M. Hanisch, "Verifying functional and non-functional properties of manufacturing control systems," in *Proceedings of the 3rd International Workshop on Dependable Control of Discrete Systems (DCDS '11)*, pp. 41–46, Saarbrücken, Germany, June 2011.

[48] N. Q. Wu, M. C. Zhou, and Z. W. Li, "Short-term scheduling of crude-oil operations: petri net-based control-theoretic approach," *IEEE Robotics & Automation Magazine*, 2015.