A PETRI NET-BASED DEADLOCK CONTROL POLICY FOR FLEXIBLE ASSEMBLY SYSTEMS

Naiqi Wu, **MengChu Zhou***, and **Elzbieta Roszkowska**[#]

Dept. of ME, Guangdong Univ. of Tech., Guangzhou 510090, P. R. China
**Dept. of ECE, New Jersey Institute of Technology, Newark, NJ 07102-1982, USA*
[#]Institute of Engineering Cybernetics, Wroclaw Univ. of Tech., ul. Janiszew skiego 11/17, 50-372 Wroclaw, Poland

Abstract**:** This paper presents a Petri net-based method for deadlock control in flexible assembly systems (FAS). Instead of a process-oriented modeling method, a resource-oriented Petri net modeling method is used. A new control policy is thus formulated and proved to outperform the existing one in the literature. The deadlock problem in FAS is often more difficult since deadlock can arise from improper base component flow and part flow, as well as ill-synchronized assembly operations.

Keywords: Petri net, assembly system, deadlock avoidance, and modeling and analysis

1. INTRODUCTION

Deadlock resolution in automated manufacturing systems (AMS) has been widely studied, for example (Zhou and DiCesare, 1991; Ezpeleta, et al., 1995; Banaszak and Krogh, 1990; Lawley, 1999; Fanti and Zhou, 2004). Recently, the deadlock problem in another AMS class characterized by non-sequential resource requirement became a hot topic (Fanti, et al., 1997; Hsieh, 2004; and Roszkowska, 2004). Flexible assembly system (FAS) is such a system. In FAS, parts need to be mounted on base components of the products. Often, bases and pallets are presented and delivered separately with parts. Both bases and parts take spaces in the assembly process. They cannot be finished without each other. Fanti, et al. (1997) and Hsieh (2004) studied the deadlock avoidance problem in FAS by using digraphs and Petri net, respectively. An assembly process is realizable if and only if there exists a feasible execution sequence (Roszkowska and Wojcik, 1993). By using Petri nets, they studied FAS with fork/join material flow, and proved that the realizability problem is NP-complete and it is NP-hard to find the maximum permissive deadlock avoidance policy. A control policy is proposed based on buffer space reservation (Roszkowska, 2004). This paper studies the same problem. Resource-oriented Petri net (ROPN) (Wu and Zhou, 2001; 2004a) is used to model it. An algorithm to calculate the realizable resource requirements and deadlock avoidance control policy are presented.

2. A FLEXIBLE ASSEMBLY SYSTEM

An FAS in Fig. 2.1 is adopted from (Roszkowska, 2004). It has two robots $r_{1\text{-}2}$ and three workstations $w_{1\text{-}3}$. There are an input buffer b_0 and output buffers b_1 for w_1, b_2 for w_2, and b_3 for w_3. Between them, b_4 can be accessed by w_1 and w_2, and b_5 by w_2 and w_3. The robots handle the transport of materials from/to system input/output and between buffers. r_1 delivers trays with parts and unpalletized subassemblies, while r_2 handles pallets with base. In the assembly process, trays with parts and subassemblies are held in b_4 or b_5, and pallets with base are put onto $b_{0\text{-}3}$. When a workstation performs an operation, it may take the parts in a tray at b_4 or b_5 and mounts them onto the base in $b_{0\text{-}3}$. Notice that workstations themselves cannot hold any component.

A and B-products are assembled concurrently with their assembly processes in Fig. 2.2, where “tray” denotes trays with parts, and “base” denotes base components. Take A-part as an example. After r_1 takes a tray with parts to be mounted onto the base into b_4, r_2 transports a base into b_0, then w_1 mounts the parts in b_4 onto the base in b_0, and the finished one is moved into b_1, while the tray with other parts remains in b_4. Then r_2 can deliver the base into b_2 and

w_2 performs an operation on it, and after that it remains in b_2. It can then be delivered into b_3 by r_2. If, at the same time, the tray in b_4 is delivered into b_5 by r_1, and other parts from the central storage are released into the system and put into b_5 by r_1, then w_3 is ready to perform its assembly. After that, the finished product remains in b_3 and the tray in b_5, respectively. They are ready to exit the system and the buffer spaces are then released. When w_3 does its assembly, it takes two parts from b_5, one is delivered to b_5 from b_4, and the other from the central storage.

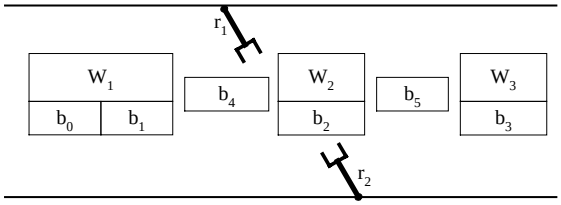

Fig. 2.1. An FAS (Roszkowaska, 2004)

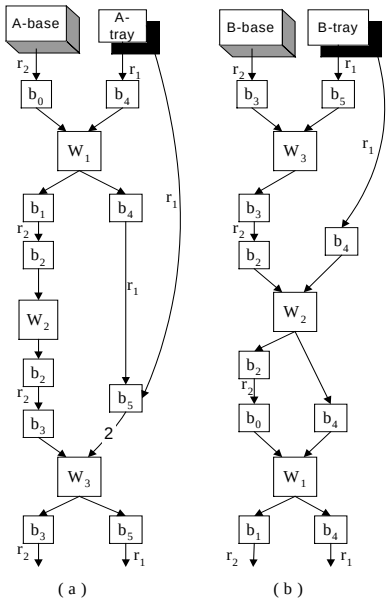

Fig. 2.2. The assembly processes of two products

3. MODELING BY ROPN

To avoid deadlock in FAS, the mechanism of dynamic resource allocation should be modeled. The FAS is modeled in (Roszkowska, 2004) by a process-oriented PN, where an operation place is introduced for each operation. This paper will model it by ROPN. It models each resource by only a single place and uses no operation places. All assembly routes are modeled via token flows.

Finite Capacity PN: Petri nets are powerful in modeling the behavior of resource allocation. Because the resources are limited in FAS, a finite capacity Petri net is an ideal choice to model them. The concept of PN presented here is based on (Zhou and Venkatesh, 1998). A finite capacity PN is a directed graph PN = (P, T, I, O, M, K) where 1) $P = \{p_1, p_2, ..., p_m\}$ is a finite set of places; 2) $T = \{t_1, t_2, ..., t_n\}$ is a finite set of transitions, $P \cup T \neq \varnothing$, $P \cap T = \varnothing$; 3) $I : P \times T \rightarrow N$ is an input function where $N=\{0, 1,2, \ldots\}$; 4) $O: P \times T \rightarrow N$ is an output function; 5) $M: P \rightarrow N$ is a marking representing the numbers of tokens in places with M_0 denoting the initial marking; and 6) $K: P \rightarrow \{1, 2, 3, \ldots\}$ is a capacity function where K(p) represents the maximal number of tokens that place p can hold at a time.

The *preset* of transition t is the set of all input places to t, i.e., $^\bullet t=\{p \in P: I(p, t) > 0\}$. The *postset* of t is the set of all output places from t, i.e., $t^\bullet=\{p \in P: O(p, t) > 0\}$. Similarly, p's *preset* $^\bullet p=\{t \in T: O(p, t) > 0\}$ and *postset* $p^\bullet =\{t \in T: I(p, t) > 0\}$.

Definition 3.1: A transition $t \in T$ in a finite capacity PN is enabled if for all $p \in P$,

$$M(p) \geq I(p, t) \quad (1)$$

and $$K(p) \geq M(p) - I(p, t) + O(p, t) \quad (2)$$

If a transition is enabled, it can fire. Firing an enabled transition t in marking M yields

$$M'(p) = M(p) - I(p, t) + O(p, t) \quad (3)$$

Definition 3.1 means that t is enabled and can fire if all the places in $^\bullet t$ have enough tokens and all the places in $t^\bullet$ have enough free spaces. Thereafter, when condition (1) is met, t is said to be process-enabled. When condition (2) is met, t is resource-enabled. Thus, t is enabled only if it is both process and resource-enabled. A sequence of firings results in a sequence of markings. M_i is said to be reachable from M_0 if there exists a sequence of firings that transforms M_0 to M_i. The set of all markings reachable from M_0 is denoted by $R(M_0)$. A place p is said to be full if M(p) = K(P). A transition in a PN is live if it can fire at least once in some firing sequence for every marking M reachable from M_0. A PN is live if every transition is live. The liveness of a PN assures that all events or activities in the model can happen. It implies the deadlock-free operation of an FAS if its PN model is live.

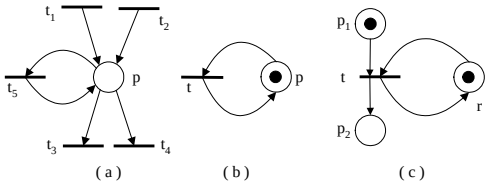

Fig. 3.1. PN models for resources

Models for Resources: Buffers are different from workstations and robots. The former is called H-resource and the latter G-resource. An H-resource is modeled by an H-place as shown in Fig. 3.1(a). A token in it represents a part occupying a space in the buffer, no matter whether it is just sitting there or being processed. Its self-loop transition, i.e., t_5, represents assembly. A G-resource is modeled by a G-place in Fig. 3.1(b). Unlike an H-place, a token in it represents that the G-resource is available. Firing t in Fig. 3.1(b) implies that the resource performs an operation. After its firing, the token comes back and the resource becomes available again.

PN Models for Individual Products: The first basic operation is material delivery by robots. It can be

modeled by a G-resource model in Fig. 3.1 (c). Firing t implies a part moved from p_1 to p_2 by r.

There are four situations for the operations performed by a workstation as modeled and explained in Fig. 3.2. Place w models a G-resource and others model H-resources. These PN models are called primitives. Consider Fig. 3.2(d). It indicates that when the assembly is completed the base and part still occupy the buffer spaces. However, if this operation is the last one for a product, then the completed product can be moved away and will no longer occupy buffer spaces. Thus, this model becomes Fig. 3.3 (b).

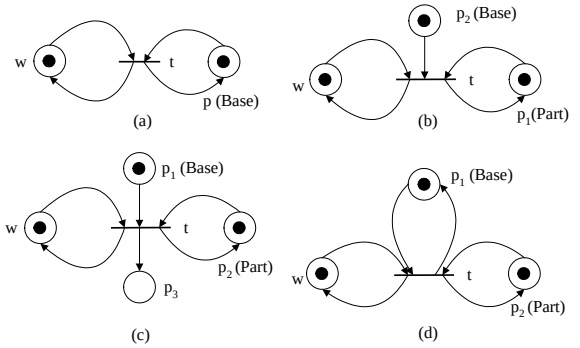

Fig. 3.2. PN primitives: (a) w processes a base in p; (b) w assembles all the parts in p_1 to base in p_2; (c) w assembles some parts in p_2 to base in p_1 and moves it to p_3; and (d) w assembles some parts in p_2 to base in p_1, after that they remain in p_1 and p_2.

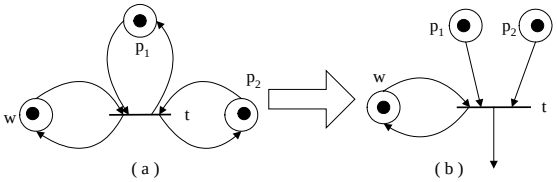

Fig. 3.3. PNs for regular and last assembly operations

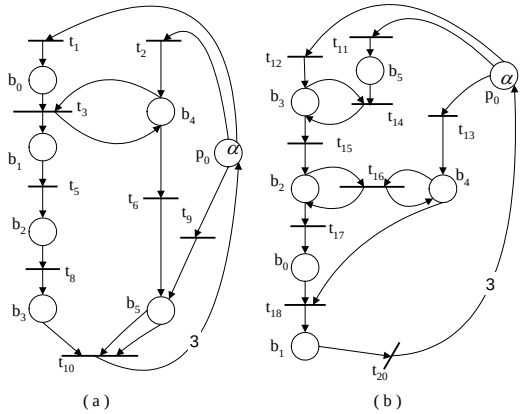

Fig. 3.4. Subnets for assembling products A and B

By using Fig. 3.1-3, we present ROPN for individual products. For easy understanding, we use b_i, i = 0, 1, …, 5, to name the places corresponding to buffers, $w_{1\text{-}3}$ for three workstations, and $r_{1\text{-}2}$ for two robots. Place p_0 represents the central storage for FAS, which hosts all the raw and final pieces. It is shown in (Wu, 1997) that by modeling the G-resources in this way deadlock resulting from processes performed by G-resources can be eliminated. Thus, in the sense of deadlock avoidance, these G-places and their associated arcs can be removed from the model. In this way, the PNs for assembling products A and B are obtained as shown in Figs. 3.4 (a) and (b) named A and B-subnets for short.

ROPN for the Whole System: We can obtain the ROPN for the whole system by merging the PNs for the individual products (Wu and Zhou, 2001). The union of the two subnets in Fig. 3.4 leads to Fig. 3.5. We use solid, dashed, and bold-solid lines to denote the individual part flows for products A and B, and shared one, respectively.

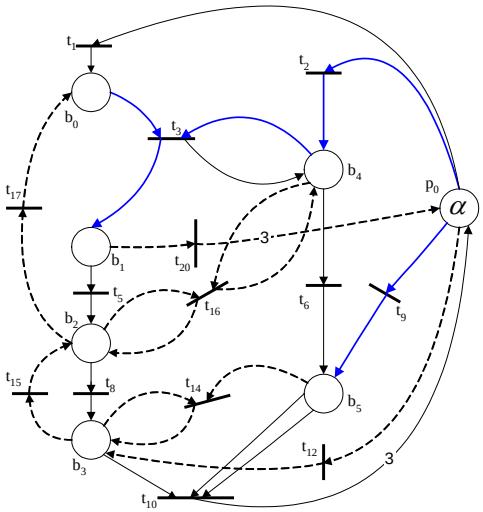

Fig. 3.5. The ROPN for the whole system

The ROPN obtained is compact, but cannot describe the material flow exactly. For example, when a token is in b_2, one may not know which of t_8, t_{16}, or t_{17} it enables. This problem can be solved by introducing colors into ROPN resulting in colored ROPN (CROPN). When we obtain the ROPN by merging the subnets we allow only one transition between two places and in the same direction, so some transitions come from only one subnet, others are from multiple subnets. If $t \in T$ is from a single subnet, we call t a nonshared transition. If it is from $k \geq 2$ subnets, we call it k-shared. For example, t_1 and t_{20} are nonshared, but $t_{2\text{-}3}$ are 2-shared.

Definition 3.1: $C(t_i) = \{c_i\}$ is the only color for a nonshared transition $t_i \in T$, and $C(t_i) = \{c_{iA}, c_{iB}, \ldots, c_{iK}\}$ for a multiple-shared transition.

A multiple-shared transition represents the assembly of multi-products. The assembly of different products is distinguished by the transition's colors.

Definition 3.2: If $t_i \in p_i^\bullet$, the tokens in place p_i enabling t_i have the color f_i associated with color c_i.
In Fig. 3.5, t_1, t_5, t_6, t_8, t_{10}, t_{14}, t_{15}, t_{17}, and t_{20} are single-colored, but t_2, t_3, and t_9 have two colors named as c_{2A} and c_{2B}, c_{3A} and c_{3B}, and c_{9A} and c_{9B}, respectively. For example, if there is a token in b_1 representing an A-base with color f_{AB2}, then t_5 with color c_5 can fire, but if it represents a B-base with color f_{BB4}, then t_{20} with color c_{20} can fire. Transition t_3 has two colors, when there is a token in b_0 representing an A-base with color f_{AB1} and an A-part

in b_4 with color f_{AP1}, t_3 can fire with color c_{3A}. Note that all the tokens in p_0 should have colors. For simplicity, we just put α in p_0 to indicate α tokens.

The color of a token may change. However, such change occurs only when an assembly transition fires. In Fig. 3.5, in p_0, color f_{AB1} is used for a token representing an A-base enabling t_1, f_{AP1} for an A-part enabling t_2, f_{AP3} for an A-part enabling t_9, f_{BB1} for a B-base enabling t_{12}, f_{BP1} for a B-part enabling t_9, and f_{BP2} for a B-part enabling t_2. When a token with color f_{AB1} enters b_0 by firing t_1, and a token with color f_{AP1} comes to b_4 by firing t_2 with color c_{2A}, these two tokens then enable t_3 with color c_{3A}. After firing t_3, their colors change into f_{AB2} and f_{AP2}, respectively.

4. REALIZABLE RESOURCE REQUIREMENT

Given a set of product types to be processed in FAS concurrently, the set of buffers with their capacity, the FAS layout, and product routes, can each operation be executed at least once? This problem is called realizability problem that is NP-complete (Roszkowska, 2004). This implies that to find a minimal realizable resource requirement (RRR) is NP-complete. However, RRR is needed as a necessary condition for deadlock control. This paper intends to find a simple algorithm for RRR. To complete the assembly of a product, its base must first be released into the system. Nevertheless, RRR can be reduced if a part to be mounted onto a base is released into the system just before FAS needs it.

Definition 4.1: Assume that in CROPN of FAS, $\exists t$, $\ni {}^\bullet t = \{p_0\}$, $p_e \in t^\bullet$, and $t_a \in p_e^\bullet$, where t_a is an assembly transition with color c. To fire t_a, it requires tokens in p_i, i = 1, 2, …, k, k ≥ 1, with color a_i, and tokens with color h_e that move into p_e by firing t. If in marking M, $\exists$ i, $\ni M(p_i)(a_i) < I(p_i, t_a)(a_i, c)$ and t fires, then firing t is premature where $M(p_i)(a_i)$ is the number of tokens of color a_i in p_i and $I(p_i, t_a)(a_i, c)$ represents the number of arcs from p_i to t_a with respect to token color a_i and transition color c.

For example, to perform the final assembly of A, there should be an A-base in b_3, an A-part from b_4, and another A-part from p_0. If the former two items are not ready at b_3 and b_5, firing t_9 to release A-parts into b_5 is of no use for the assembly process but occupies buffer spaces. Thus, such premature firing of t_9 should be avoided. Assume that there are β types of products to be assembled, product i needs d_i parts to be mounted onto its base. Thus, if FAS is required to assembly one product for each type, there are totally $\alpha = \beta + \sum_{i=1}^{\beta} d_i$ parts including β bases.

Algorithm 4.1: Find RRR for assembling one product of each type concurrently.

Step 1: Initialization: let K(p) = ∞ for all $p \in P = \{p_0, p_1, p_2, \ldots, p_m\}$ in the CROPN, put α tokens with their colors in p_0 and set the marking of the CROPN $M = M_0$ with all tokens in p_0, and let R[] = 0 be an m-vector with integer values;

Step 2: Find $T_E \subseteq T$ such that $\forall t \in T_E$ is enabled in the current marking M. If $T_E = \varnothing$ then stop, otherwise go to the next step;

Step 3: In T_E find the set of transitions T_{EF}, such that $\forall t \in T_{EF}$ whose firing is premature, and set $T_E = T_E - T_{EF}$, do the following:

1) Form a firing sequence f in T_E, and the order can be set arbitrarily;
2) Calculate M' = M[>f, and let M = M';
3) For i = 1 to m do
   If $M(p_i) > R[i]$, $R[i] = M(p_i)$;
4) Go to Step 2.

When algorithm 4.1 ends, vector R[] returns RRR.

Example 1: Find RRR in the CROPN shown in Fig. 3.6 via Algorithm 4.1. We obtain RRR for b_0, b_1, b_2, b_3, b_4, and b_5 are 1, 1, 2, 1, 1, and 2, respectively.

Base on Algorithm 4.1, we know that for the set of products to be processed in FAS all the operations can be executed at least once. Thus, according to the definition of realizability in (Roszkowska, 2004), the assembly process is realizable.

Property 4.1: For the given set of product types, the assembly process is realizable with the buffer capacity obtained by Algorithm 4.1.

Assume that there are m places (buffers) and n transitions in CROPN of FAS. Then Algorithm 4.1's computational complexity is as follows.

Property 4.2: The computational complexity to find the realizable buffer capacity requirements by Algorithm 4.1 is $O(\alpha n(n + m))$.

The solution found by Algorithm 4.1 is shown to be feasible for the execution of a single product for each type of product. However, we still do not know if it is feasible when there are multiple products for each type. By Algorithm 4.1, we can find $R_i[]$ by releasing one product of a type i into the CROPN for product type i. Based on $R_i[]$, $i \in \{1, 2, \ldots, \beta\}$, RRR for deadlock avoidance with multiple products in FAS can be calculated as follows.

Definition 4.2: Place p is said to be an assembly place if $\exists t \in p^\bullet$ such that t is an assembly transition. Further, if t is for the assembly of product type i, it is said that i is a product type assembled in p.

For an assembly place p, let AS(p) denote the set of product types that are assembled in p, NAS(p) denote the set of product types that use p in the assembly process, but are not assembled in p. Clearly, we have $AS(p) \cap NAS(p) = \varnothing$. If $AS(p) = \varnothing$, p is said to be a non-assembly place. Let $Z[k] = \sum_i R_i[k]$, $i \in AS(p_k)$ and $W[k] = \max_j(R_j[k])$, $j \in NAS(p_k)$, then R[k] = Z[k] + W[k] is the minimal resource requirement for deadlock control with multiple products in the system, where k is the kth place.

Example 2: Find RRR for deadlock control in the CROPN shown in Fig. 3.5. From the CROPNs shown in Fig. 3.4 (a) and (b), we obtain $R_A[] = (1, 1, 1, 1, 1,$

2) and $R_B[] = (1, 1, 1, 1, 1, 1)$. Based on Definition 4.2 we obtain R[] = (2, 1, 2, 2, 2, 3) that is smaller than (2, 1, 2, 2, 3, 3) obtained in (Roszkowska, 2004).

5. DEADLOCK AVOIDANCE POLICY

We assume that the buffer capacity in the system is greater than or equal to R[]. Deadlock due to base component and part flows takes place in circuits. In Fig. 3.5, if circuits $C_1 = \{b_0, t_3, b_1, t_5, b_2, t_{17}, b_0\}$ and $C_2 = \{b_2, t_8, b_3, t_{15}, b_2\}$ are full, deadlock occurs. Consider a subnet shown in Fig. 5.1 where $K(p_i)=1$, i=1-4. Assume that t_3 is an assembly transition for A-products. It requires an A-base and an A-part in p_3 and p_4, respectively. t_4 is an assembly transition for B-products, it requires a B-base and a B-part in p_3 and p_4, respectively. If at some time, p_3 has A-base and p_4 B-parts, although there may be B-base in p_1 and A-parts in p_2, assembly cannot be done and a deadlock occurs. Hence, to avoid deadlock in FAS we should avoid deadlock resulting from base component flow, part flow and assembly.

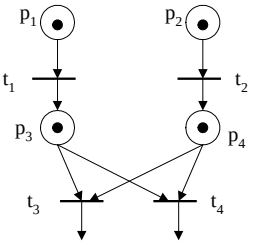

Fig. 5.1. PN subnet for assembly operations

The buffers belonging to a workstation can be seen as a special set of buffers denoted by B_i for workstation i. It may contain only one buffer. B_i is called a buffer group. We also notice that there is some $t \in p_0^\bullet$ and $t^\bullet$ is an input place of an assembly transition such that firing t releases a part into the system. Such a set of transitions is denoted by T_{tray}. For example, in the CROPN shown in Fig. 3.8, t_2 and t_9 are such transitions.

WZ-Policy: A transition $t \in T$ in CROPN for FAS in marking M is enabled and its firing changes M to M'. Then t can fire only if all the following conditions are met:

1) Assume that there are k groups of buffers $B_{1\text{-}k}$ for the base. Then in M' there are at least k-1 free buffer spaces such that at most one group B_i is full;
2) Assume that there are k buffers $b_{1\text{-}k}$ for the parts to be mounted onto the base. Then in M' there are at least k-1 free buffer spaces such that at most one buffer is full;
3) Assume $t \notin T_{tray}$, and its firing moves V_h tokens of type-h product into p_i. Let $M(p_i)(k)$ denote the number of tokens representing type k product in p_i in M. Further assume $t_1 \in T_{tray}$, and its firing moves U_h tokens of type-h product into p_i. Then
   a) $K(p_i) - W[i] - \sum_{k \in AS(j), k \neq h} \max(M(p_i)(k), R_k[i]) - M(p_i)(h) - U_h \geq V_h$, if $h \in AS(i)$;
   b) $K(p_i) - Z[i] - M(p_i)(h) \geq V_h$, if $h \in NAS(i)$;
4) Assume $t \in T_{tray}$, its firing moves U_h tokens of type-h product with color C_1 into p_i, and these tokens together with V_h tokens of type-h product in p_j and Y_h tokens with color C_2 in p_i enable assembly transition t_a. Then $M(p_j)(h) \geq V_h$, $M(p_i)(C_2) \geq Y_h$, and $M(p_i)(C_1) = 0$.

Conditions 1-2 ensure that the base and part flows will not be deadlocked, respectively. Condition 3 ensures that the state shown in Fig. 5.1 can never occur. Condition 4 avoids any premature firing so that the buffer spaces are effectively used. Conditions 3 and 4 make no restriction on a non-assembly place.

Theorem 5.1: The CROPN for an FAS is deadlock-free if WZ-policy is applied.

Proof: See (Wu and Zhou, 2004b).

Although WZ-policy is a group of sufficient conditions, it is not too conservative. For we require only n-1 free space for the base component and part transportation, respectively, with n groups of buffers. In general, in FAS each buffer has multiple spaces. Hence the restriction imposed is not too much. We impose less restriction on the number of parts in FAS, rather restrict the ratio of parts among the types.

Theorem 5.2: WZ-policy is less conservative than R-policy (Roszkowska, 2004).

Proof: See (Wu and Zhou, 2004b).

Theorem 5.3: The complexity for implementing WZ-policy is $O(|P|\beta)$, where P is the place set in CROPN and β is the number of product types.

6. ILLUSTRATIVE EXAMPLE

Example 3: Consider the FAS shown in Fig. 2.1. The capacity of $b_{0\text{-}5}$ is 2, 4, 4, 3, 6, and 5, respectively. The CROPN is shown in Fig. 3.5. By using R-policy (Roszkowska, 2004), a marking is reached such that 1, 1, 3, 1 A-bases are in b_0, b_1, b_2, and b_3, respectively. 3, 3 A-parts are in b_4, b_5, respectively. 1, 1, 1 B-bases are in b_0, b_1, and b_3, respectively. 1, 1 B-parts are in b_4, b_5, respectively. In this marking t_{12} is forbidden by R-policy. This implies that no more jobs can be released into the system.

Consider the system under WZ-policy. From Section 4, $R_A[] = (1, 1, 1, 1, 1, 2)$ and $R_B[] = (1, 1, 1, 1, 1, 2)$. Places b_0 and $b_{2\text{-}5}$ are assembly places. In this marking, it is easy to verify that Condition 3) is satisfied for both products A and B. There are three groups of places for the base flow $B_1 = \{b_0, b_1\}$, $B_2 = \{b_2\}$, and $B_3 = \{b_3\}$, Condition 1 is satisfied at this marking. The free spaces in b_4 and b_5 are 1 and 1, or Condition 2 is also satisfied. This implies that this marking is a legal marking for WZ-policy and can be reached. However, there are more free spaces in the buffers than what is required by WZ-policy, so more parts can be released into the system if other conditions are satisfied. In fact, in this marking, t_{12} is

enabled and firing it does not violate WZ-policy, so it can fire.

In fact, we have $\sum_{i=0}^{3} K(b_i) = 13$. By Condition 1, only two free spaces are needed. This implies that at most 11 products can be released into the system for assembly. Assume that initially $M_0(p_0) > 0$ and for any i, $M_0(b_i) = 0$, the evolution from M_0 is presented in Table 6.1 where the number in the brackets after a pace represents the capacity of the place. In the table, ① is used for a token of A-base with color f_{AB1} and ② for a token of A-base with color f_{AB2}; ③ for A-part with color f_{AP1}, ④ for A-part with color f_{AP2}, and ⑤ for A-part with color f_{AP3}; ❶ for B-base with color f_{BB1}, ❷ for B-base with color f_{BB2}, ❸ for B-base with color f_{BB3}, and ❹ for B-base with color f_{BB4}; ❺ for B-part with color f_{BP1}, ❻ for B-part with color f_{BP2}, and ❼ for B-part with color f_{BP3}. It is shown that totally, 11 products are released into the system for assembly. R-policy allows only 9 products to be released into it. This example shows Theorem 5.2's correctness.

Table 6.1. The token evolution from M_0

Marking	Type	$b_0(2)$	$b_1(4)$	$b_2(4)$	$b_3(3)$	$b_4(6)$	$b_5(5)$
M_1	A	①				③	
	B				❶		❺
M_2	A	①	②			③④	
	B				❷❶		❺
M_3	A	①	②	②		③④	④
	B			❷	❷❶	❻	❺
M_4	A	①	②	②②		③④	④④
	B			❸❷	❷❶	❼❻	❺
M_5	A	①	②②	②	②	④④	④④④⑤
	B	❸		❸❷	❷❶	❼❼❻	❺
M_6	A	①	②②	②②	②	④④③	④④④⑤
	B	❸	❹	❸	❷❷	❼❼	

7. CONCLUSIONS

In FAS, base components are transported with pallets, and parts to be mounted onto them are transported with no pallets but via trays. When an assembly operation is performed by using some parts in a tray, the tray still occupies a buffer space. In this way, an assembly/disassembly material flow is formed. In such systems, deadlock can occur in the base component flow, part flow and assembly. Thus, it is a great challenge to avoid deadlock in FAS. This paper proposes to use resource-oriented Petri nets to capture the discrete event dynamics of FAS concisely and presents a deadlock control policy. The policy is computationally efficient and better than the existing one. It may be used to FAS in (Hsieh, 2004).

REFERENCES

Z. A. Banaszak and B. H. Krogh (1990). Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows, *IEEE Trans. on Robotics and Aut.*, 6(6), pp. 724-734.

J. Ezpeleta, J. M. Colom, and J. Martinez (1995). A Petri net based deadlock prevention policy for flexible manufacturing systems, *IEEE Trans. on Robotics and Aut.*, vol. 11, no. 2, 171-184, 1995.

M. P. Fanti, B. Maione, and B. Turchiano (1997), Event control for deadlock avoidance in assembly systems, in *Proc. IEEE Conf. Systems, Man, & Cybernetics*, 3756-3761.

M. P. Fanti and M. Zhou (2004). Deadlock control methods in automated manufacturing systems, *IEEE Trans. on Systems, Man, & Cybernetics, Part A*, vol. 34, no. 1, 5-22.

F.-S. Hsieh (2004). Fault-tolerant deadlock avoidance algorithm for assembly processes, *IEEE Trans. on Systems, Man, & Cybernetics, Part A*, vol. 34, no. 1, 65-79.

M. Lawley (1999). Deadlock avoidance for production systems with flexible routing, *IEEE Trans. on Robotics and Aut.*, 15(3), 497-509.

E. Roszkowska (2004) Supervisory control for deadlock avoidance in compound processes, *IEEE Trans. on Systems, Man, & Cybernetics, Part A*, vol. 34, no. 1, 52-64.

E. Roszkowska and R. Wojcik (1993). Problems of process flow feasibility in FAS, in *CIM in Process and Manufacturing Industries*, Oxford, UK: Pergamon, 115-120.

N. Q. Wu (1997). Avoiding deadlocks in automated manufacturing systems with shared material handling system, in *Proc. of 1997 IEEE Int. Conf. on Robotics and Aut.*, pp. 2427-2433.

Wu, N. Q. and M. C. Zhou (2001). Avoiding deadlock and reducing starvation and blocking in automated manufacturing systems, *IEEE Trans. on Robotics and Aut.*, vol. 17, no.5, 657-668.

N. Q. Wu and M. C. Zhou (2004a). Modeling and deadlock control of automated guided vehicle systems, *IEEE Trans. on Mechatronics*, 9(1), 50-57.

N. Q. Wu and M. C. Zhou (2004b). Resource Oriented Petri Nets for Deadlock Avoidance in Flexible Assembly Systems, Technical Report #2004-45, ECE, New Jersey Inst. of Technology.

M. Zhou and F. DiCesare (1991). Parallel and sequential mutual exclusions for Petri net modeling of manufacturing systems with shared resources, *IEEE Trans. on Robotics and Aut.*, vol. 7, no. 4, 515-527.

M. C. Zhou and K. Venkatesh (1998). *Modeling, Simulation and Control of Flexible Manufacturing Systems: A Petri Net Approach*, World Scientific, Singapore.