Design and Implementation of Decentralized Supervisory Control for Manufacturing System Automation

Hyoung Il Son

Design and Implementation of Decentralized Supervisory Control for Manufacturing System Automation

Journal:	*International Journal of Computer Integrated Manufacturing*
Manuscript ID:	TCIM-2010-IJCIM-0120.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	30-Dec-2010
Complete List of Authors:	Son, Hyoung Il; Max Planck Institute for Biological Cybernetics, Dept. of Human Perception, Cognition and Action
Keywords:	AUTOMATED MANUFACTURING SYSTEMS, MANUFACTURING CONTROL SYSTEMS
Keywords (user):	Decentralized control, discrete event systems

Design and Implementation of Decentralized Supervisory Control for Manufacturing System Automation

Hyoung Il Son

Department of Human Perception, Cognition and Action, Max Planck Institute for Biological Cybernetics, Spemannstraße 38, 72076 Tübingen, Germany

Tel: +49-7071-601-218, Fax: +49-7071-601-616, e-mail: hyoungil.son@tuebingen.mpg.de

Abstract

Supervisory control theory, which was first proposed by Ramadge and Wonahm, is a well-suited control theory for the control of complex systems such as semiconductor manufacturing systems, automobile manufacturing systems, and chemical processes because these are better modeled by discrete event models than by differential or difference equation models at higher levels of abstraction. Moreover, decentralized supervisory control is an efficient method for large complex systems according to the divide-and-conquer principle. This paper presents a solution and a design procedure of supervisory control problem (SCP) for the case of decentralized control. We apply the proposed design procedure to an experimental miniature computer integrated manufacturing (CIM) system. This paper presents the design of fourteen modular supervisors and one high-level supervisor to control the experimental miniature CIM system. These supervisors are controllable, nonblocking, and nonconflicting. After the verification of the supervisors by simulation, the collision avoidance supervisors for AGV system have been implemented to demonstrate their efficacy.

Keywords: Decentralized control, discrete event systems, manufacturing automation, supervisory control theory.

1. Introduction

As any manufacturing system becomes larger and more complex, more systematic and rigorous methods are needed for the modeling and control of such large complex systems. Supervisory control theory (Ramadge and Wonham 1987, Wonham and Ramagde 1987) , which was proposed by Ramadge and Wonham and based on discrete event system (DES) methods, is recognized as one of the promising techniques for the design and control of large complex systems such as semiconductor manufacturing systems, chemical processes, HVAC (heating, ventilation and air conditioning), and power plants. Recently, the supervisory control theory has received much focus in many applications such as robotics (Ricker *et al.* 1996, Chung and Lee 2005), traffic control (Giua and Seatzu 2001), logistics (Jafari *et al.* 2002), failure diagnosis (Son and Lee 2007), and manufacturing systems (Golmakani *et al.* 2006) because it can satisfy control specifications of a plant to be controlled systemically by permitting eligible events in the plant maximally. Also it has been proved that the supervisory control theory is very efficient for the control of highly complex systems (Cassandra and Lafortune 1999, Ramadge and Wonham 1989) which are modeled as Petri nets (Basile *et al.* 2004, Dai *et al.* 2009) or automata (Lee and Lee 2002, Ramires-Serrano and Benhabib 2003).

A general problem of the design and control of target systems based on the supervisory control theory (Wonham 1998) is named as supervisory control problem (SCP). The SCP is, generally, used to find a supervisory controller, i.e. centralized

supervisor, which satisfies the legal language (behavior specification) of a system (Wonham 1998). However, as the system becomes larger and more complex, the computational complexity of the SCP increases exponentially due to the increase of eligible events. The divide and conquer principle is very useful to solve this problem, because the computational complexity can be decreased exponentially if the SCP is solved by dividing the system into several sub-systems (Rudie and Wonham 1992). Supervisory controllers designed, based on this approach, are called as modular supervisory controllers horizontally (Wonham and Ramadge 1988) and high-level supervisory controllers hierarchically (Leduc *et al.* 2006). Finally, a decentralized supervisory control system is defined as a supervisory control system which consists of the modular supervisors and the high-level supervisors (Yoo and Lafortune 2002).

A hierarchical supervisory control is presented in Tittus and Lennartson (2002) as a Petri net-based approach and in Leduc *et al.* (2005, 2009) as an automata-based approach. They proved that a proposed hierarchical supervisor is by far less complex than a non-hierarchical one theoretically. Yoo and Lafortune (2002) presented a generalized form of the conventional decentralized control architecture for discrete event systems. They proposed a concept of fusion operation using both the union and the intersection of enabled events. Their method is extended to allow the making of conditional decisions also, "enable if nobody disables" and "disable if nobody enables", in addition to unconditional decisions, "enable" and "disable" in Yoo and Lafortune (2004). They, however, did not present a design procedure with a practical example for the easy use of the presented theory even if their method is rigorous. Feng *et al.* (2009) proposed a similar method for a decentralized nonblocking supervisory control. They briefly outlined the proposed theory with a practical example. The other approach, so called supervisor localization, is proposed by Kai and Wonham (2010) for the distributed control architecture of large scale discrete event systems. They analyze tradeoffs between the decentralized and distributed control architecture. A practical implementation method is not presented in Feng *et al.* (2009) and Cai and Wonham (2010).

Queiroz and Cury presented an implementation method of modular supervisory controller using a programmable logic controller (PLC) (Queiroz and Cury 2002). They explained their method with a simple manufacturing cell example. They, however, showed only simulation results using the proposed method. Supervisor implementation using the PLC is also presented by Ramires-Serrano *et al.* (2002) and by Petin *et al.* (2007).

Petri net is, usually, more efficient as a modeling and analysis method for the deadlock avoidance (Lerrarini *et al.* 1999) and performance evaluation (Tsinarakis *et al.* 2005) of a manufacturing system. We, therefore, use automata to model the manufacturing system for the supervisory control in this paper.

In this study, a concept of sub-plant is proposed to reduce the computational complexity for controllability in the SCP and then, a generalized solution of the SCP for the modular supervisors is proposed. A solution of the SCP for a high-level plant with respect to a high-level behavior specification is also developed using the proposed concept of the sub-plant and Wonham *et al.*'s method. The developed solutions are proved theoretically.

Modular and high-level supervisors are designed, implemented, and verified using an experimental miniature computer integrated manufacturing (CIM) system using the proposed decentralized supervisory control scheme. The experimental miniature CIM system consists of three industrial robots, two automated guided vehicles (AGVs), two numerical controlled (NC) machines, several conveyor belts, and sensors. A plant of the miniature CIM system is modeled as the deterministic automaton. Operation rules of the miniature CIM system is defined as the behavior specifications (legal languages) and supervisors are then designed with respect to these specifications. And the designed supervisors are then transformed into the clocked Moore synchronous state machine (Wakerley 1990) for the implementation. We, finally, verify a supervisor for a collision avoidance of AGVs via an experiment which is a critical problem for the material transfer in production lines (Singh *et al.* 2010).

This paper is organized into six chapters. In the chapter following this introduction, a background of the supervisory control theory is presented. In the third chapter, the design methodologies of the decentralized supervisory control and its theoretical proofs are presented. An application to the experimental miniature CIM system of the proposed control and an implementation of the designed controller are presented in the fourth and fifth chapters, respectively. Finally, the main contributions of this paper are summarized in the last chapter.

2. Background

2.1 System Modeling

Discrete event system (DES) is modeled as the automaton $G = \{Q, \Sigma, \delta, q_0, Q_m\}$ where Q is the state set, Σ is the event set, $\delta : Q \times \Sigma^* \to Q$ is the state transition function, q_0 is the initial state, Q_m is the marked state set which is a subset of Q. In δ, Σ^* is the set of null event and string (sequence) expressed as ε and $\sigma_1\sigma_2\sigma_3\ldots\sigma_k, k \geq 1$, respectively. In particular, the event set Σ is divided into two disjoint sets, i.e., the controllable event set Σ_c and the uncontrollable event set Σ_{uc}. And Σ is also partitioned into the observable event set Σ_o and the unobservable event set Σ_{uo}. The language, which is generated by G, is defined as in (1)

$$L(G) = \{s \mid s \in \Sigma^*, \delta(q_0, s)!\} \tag{1}$$

where $\delta(q_0, s)!$ means that a next state is defined after the occurrence of the string s in the state q_0. The prefix closure of $L(G)$ is defined as

$$\overline{L(G)} = \{t \in \Sigma^* \mid t \leq s \ \textit{for some}\ s \in L(G)\} \tag{2}$$

And the marked language of G is defined as follows.

$$L_m(G) = \{s \mid \delta(q_0, s)! \in Q_m, L_m(G) \subseteq K(G)\} \tag{3}$$

If G satisfies $\overline{L_m(G)} = L(G)$ then $L(G)$ is nonblocking. The nonblockingness is then the necessary condition to design a proper supervisor in the supervisory control theory [14]. And if the prefix closures of two languages are disjoint, these languages are nonconflicting as defined in (4).

$$\overline{L(G)} = \overline{L(G_1)} \cup \overline{L(G_2)}, \ null = \overline{L(G_1)} \cap \overline{L(G_2)} \Rightarrow L(G_1) \wedge L(G_2) \tag{4}$$

Finally, the projection map P is defined as $P(\sigma) = \varepsilon$ and $P(s\sigma) = P(s)$ for $\sigma \in \Sigma_{uo}, s \in L(G)$ [14].

2.2 Supervisory Control

Supervisor is also defined as the automaton $S = \{X, \Sigma, \xi, x_0, X_m\}$ where X, Σ, $\xi : X \times \Sigma^* \to X$, x_0, and X_m are the state set, the event set, the state transition function, the initial state, and the marked state, respectively. Let the plant to be controlled be defined as G and then the behavior of plant G under the supervision of S is represented as (5).

$$S/G=\{X\times Q,\Sigma,\xi\times\delta,(x_0,q_0),X_m\times Q_m\} \tag{5}$$

The controllability and the observability of $L(S)$ with respect to $L(G)$ are defined in Definition 1 and 2 respectively.

Definition 1: For $S\subseteq G$, S is *controllable* with respect to (G,Σ_{uc}) if the following is satisfied.

$$(\forall s,\sigma)\ s\in\overline{L(S)},\sigma\in\Sigma_{uc},s\sigma\in L(G)\Rightarrow s\sigma\in\overline{L(S)} \tag{6}$$

The physical meaning of controllability is that an arbitrary string s, which is permissible by the supervisor S and an uncontrollable event σ, is eligible in the plant G, if the string $s\sigma$ is eligible in G and if S also permits $s\sigma$, then S is controllable with respect to G.

Definition 2: For $S\subseteq G$, S is *observable* with respect to (G,P,Σ_{uo}) if the following is satisfied.

$$\forall s,s',\sigma\in\overline{L(S)},P(s)=P(s'),\sigma\in\Sigma_{uo},s\sigma\in\overline{L(S)},s'\sigma\in L(G)\Rightarrow s'\sigma\in\overline{L(S)} \tag{7}$$

Observability means that if $s\sigma$ is permissible by S and $s'\sigma$ is eligible in G, then S also have to permit $s'\sigma$ where two strings s and s' are recognized as the same string by the projection map P and are also permissible by the supervisor S as well as σ is an unobservable event.

Supervisory control problem (SCP) is defined in Definition 3 based on Definition 1.

Definition 3: For a given K and G, where $K\subseteq G$, find a supremal language S which satisfies $L(S/G)=K$ and $L(S/G)=\overline{L_m(S/G)}$ and is controllable with respect to (G,Σ_{uc}).

If K is, therefore, defined as the legal language for the plant G to be controlled, then the SCP is to find a supervisor which satisfies $L(S/G)=K$ and is nonblocking and controllable with respect to G. In addition, the supervisor which satisfies the constraints and is controllable, need not be unique. Among these supervisors, a supremal controllable sublanguage of G with respect to K is the unique solution of the SCP. Therefore, the supervisor S which satisfies Definition 3 can permit the language occurred in the plant G maximally. A supervisory control system is illustrated in Fig. 1.

3. Decentralized Supervisory Control

3.1 Design of Modular Supervisor

Let us consider two fundamental issues, the computational complexity and the implementation simplicity in this section. Firstly, a method to reduce the computational complexity is presented. Solving the SCP with respect to all the plants takes a tremendous computational complexity. Therefore, the computational complexity can be decreased exponentially if the SCP is solved with respect to several sub-plants. This approach is presented in Theorem 1.

Theorem 1: For a given plant G which can be expressed as $G=G_1\times G_2\times\ldots\times G_n$, let us define a sub-plant $G_{sub,i}\in G$ for the legal language $K_i, i=1,\ldots,m$. If S_i is a solution of the SCP with respect to $(K_i,G_{sub,i})$ and is nonconflicting with the $G_{sub,j}, i\neq j$, then S_i is the solution of the SCP with respect to (K_i,G).

Proof: Based on the SCP, we have to prove that S_i is controllable with respect to (K_i, G), is nonblocking and is the maximally permissible language. Firstly, let us consider the controllability of S_i. If the event sets of G and $G_{sub,i}$ are defined as Σ and $\Sigma_{sub,i}$, then new event set $\Sigma^c_{sub,i} = \Sigma - \Sigma_{sub,i}$ can be defined. Here, every uncontrollable event, which is an element of $(\Sigma^c_{sub,i})_{uc} \subseteq \Sigma^c_{sub,i}$, is permitted by S_i because $\Sigma^c_{sub,i}$ is the event set consisting of self-loops in S_i. Therefore, S_i is controllable with respect to G. And S_i is nonblocking because it is nonconflicting with respect to $G_{sub,j}, i \neq j$. Finally, K_i is the maximally permissible language because every event in $\Sigma^c_{sub,i}$ is permitted by S_i. ■

Secondly, let us consider finding an equivalent supervisor which is less complex to implement because it has less states and less state transitions even if it generates same language with the original solution of the SCP. Generally, the supervisor S satisfies

$$L(S/G) = L(S) \tag{8}$$

However, the supervisor S becomes much complex because it has much more states than those in the legal language K with respect to the events generated in the plant G. Practically, this complexity creates a problem in the implementation of the supervisor. Therefore, it will be relatively easier to implement the simpler supervisor S' which satisfies (9)

$$L(S'/G) = L(S) \tag{9}$$

This means that the supervisor S' which is simpler than S can be designed by satisfying (9) while the language of plant behavior under the supervision of S' is same with the one under the supervision of S. If the legal language K is defined and then the maximum number of state in S' is the same with that of K while the maximum number of states in S is the same with that of $K \wedge G$. We summarized this issue in Theorem 2.

Theorem 2: If the supervisor S' satisfies the following conditions, S' is an *optimal (or minimally restrictive) proper supervisor* with respect to the plant G.

1) The supervisor S' is controllable with respect to the plant G.

2) $\overline{L_m(S')} = L(S')$

3) $\overline{L_m(G) \wedge L_m(S')} = L(G) \wedge L(S')$

4) If S is the supremal controllable sublanguage with respect to K, $L(S'/G) = L(S)$ has to be satisfied.

Proof: Condition 1) means that the designed supervisor has to satisfy the controllability with respect to the plant and the second condition represents the supervisor has to be nonblocking. And the nonconflictness of the supervisor with the plant is represented in Condition 3). In other words, the third condition means that the supervisor has to be nonblocking with respect to the plant. Therefore, if S' satisfies Conditions 1), 2), and 3) S' is a proper supervisor. Condition 4) represents the behavior of the plant under the supervision of S', which has to generate the maximally controllable sublanguage. Finally, S' becomes the optimal supervisor. ■

The modular supervisor is defined in Definition 4 based on Theorems 1 and 2.

Definition 4: For the legal languages $K_j, j=1,2,\ldots,n$, let us design the supervisors $S_i, i=1,2,\ldots,m$ which satisfy Theorems 1 and 2. And if S_i satisfies the nonconflictness condition $\overline{S}=\overline{S_1}\wedge\overline{S_2}\wedge\ldots\wedge\overline{S_m}$, then S_i is defined as the *modular supervisor*.

Finally, the solution of the modular SCP is presented in Theorem 3 using Theorems 1 and 2. The computational complexity of the algorithm for the controllability, the nonblocking, the nonconflictness tests which is presented in Theorem 3 is same with the one proposed by Ramadge and Wonham (1987, 1989).

Theorem 3: For a given plant G and legal languages $K_i, i=1,2,\ldots,m$, modular supervisors S_i or S_i' are the solutions of the SCP using the following procedure.

Modular SCP solution procedure:

Step 0: Define the automaton G of the plant to be controlled and the automation K_i of the legal languages.

Step 1: Design the sub-plants $G_{sub,i}$.

Step 2: Check the controllability of K_i with respect to $G_{sub,i}$ using the controllable events in K_i. If K_i is controllable, go to Step 5, otherwise go to next step.

Step 3: Reconstruct K_i by considering the events which do not satisfy the controllability in the controllable events in K_i.

Step 4: Go to Step 2. If K_i cannot be reconstructed while satisfying the controllability, then go to Step 7.

Step 5: Check the nonblockingness of K_i. Delete the state if there exists a state which makes K_i as blocking and then go to Step 2.

Step 6: Check the nonconflictness of K_i with respect to $G_{sub,i}$. If K_i is nonconflicting then $S_i'=K_i$. Otherwise, reconstruct K_i by checking the string which makes K_i as conflicting and then go to Step 2.

Step 7: Find the supremal controllable sublanguage S_i of K_i with respect to $G_{sub,i}$. S_i is the solution of the modular SCP with respect to $(K_i, G_{sub,i})$.

Step 8: If $L(S_i)=L(S_i'/G_{sub,i})$ then S_i' is the solution of the modular SCP with respect to $(K_i, G_{sub,i})$.

Proof: The proof is omitted because it is straightforward from the proofs of Theorems 1 and 2. ■

3.2 Design of High-level Supervisor

Let us represent the plant G as the low-level plant $G_{lo}=\{Q_{lo},\Sigma_{lo},\delta_{lo},q_{lo,0},Q_{lo,m}\}$ and define the high-level plant G_{hi} which satisfies $L(G_{hi})=\Theta\{L(G_{lo})\}$ with the information map Θ. The information map is defined as $\Theta: L(G_{lo})\rightarrow T^*$ where $T=\{\tau_0,\tau_1,\tau_3,\ldots\}$ is the set of events which have the physical meaning in the high-level plant among the low-level events. The information map is an arbitrary projection map. The high-level plant G_{hi} is also represented as the automation $G_{hi}=\{Q_{hi},\Sigma_{hi},\delta_{hi},q_{hi,0},Q_{hi,m}\}$ similar to G_{lo}. Therefore, the high-level supervisor can be designed if we solve the SCP with respect to the high-level plant G_{hi} and the high-level legal language K_{hi}.

The information map Θ is defined by mapping the high-level event τ as the state output of the states of G_{lo}. The state in G_{lo} which has the state output about Θ is defined as the vocal state. And then G_{hi} can be constructed from Θ and G_{lo}. Before constructing G_{hi}, G_{lo} has to be reconstructed using the following two conditions to make G_{hi} maintain the control structure of G_{lo}.

Condition 1 for the high-level plant: OCC (Output Control Consistency)

$$\Theta^{-1}\{L(G_{hi})\}^{\uparrow} = L(G_{lo}) \tag{10}$$

Condition 2 for the high-level plant: SOCC (Strictly Output Control Consistency)

$$\Theta\left[\Theta^{-1}\{L(G_{hi})\}^{\uparrow}\right] = L(G_{hi}) \tag{11}$$

Condition 1 means that in a certain low-level state when there is a state transition by the low-level event which has the state output and also there is a state transition by the low-level event which, however, has no state output, this low-level state has to be divided with respect to two different state transitions. And Condition 2 represents that the state outputs have to be redefined according to whether the low-level event which makes the state transition with respect to the reconstructed low-level states by Condition 1 is controllable or not. Both Condition 1 and 2 are defined as the hierarchical consistency.

The design procedure of the high-level supervisor is presented in Theorem 4 using Theorem 3 and OCC and SOCC condition.

Theorem 4: For a given low-level plant G_{lo}, an information map Θ, and high-level legal languages $K_{hi,i}, i = 1,2,\ldots,m$, high-level supervisors $S_{hi,i}$ or $S'_{hi,i}$ are solutions of the SCP using the following procedure.

<*High-level SCP solution procedure*>

Step 0: Define the automaton G_{lo} of the low-level plant to be controlled and the automation $K_{hi,i}$ of the high-level legal languages. And also define the information map Θ.

Step 1: Design the sub-plants $(G_{lo})_{sub,i}$.

Step 2: Construct $(G_{lo})^{vocal}_{sub,i}$ of $(G_{lo})_{sub,i}$ using Θ.

Step 3: Construct $\{(G_{lo})^{vocal}_{sub,i}\}_{OCC}$ of $(G_{lo})^{vocal}_{sub,i}$ using (10).

Step 4: Construct $\{(G_{lo})^{vocal}_{sub,i}\}_{SOCC}$ of $\{(G_{lo})^{vocal}_{sub,i}\}_{OCC}$ using (11).

Step 5: Construct $G_{hi} = \Theta[\{(G_{lo})^{vocal}_{sub,i}\}_{SOCC}]$ and define G_{hi} as the high-level plant.

Step 6: Run from Step 2 to Step 8 of Theorem 3 with respect to $(K_{hi,i}, G_{hi})$.

Proof: The proof is omitted because it is straightforward from OCC and SOCC condition and the proofs of Theorems 1 and 2. ■

Finally, the architecture of decentralized supervisory control is illustrated in Fig. 2.

4. Application: Miniature CIM System

4.1 Layout

In this paper, an experimental miniature computer integrated manufacturing (CIM) system is experimented to verify the proposed decentralized supervisory control. The miniature CIM system consists of two NC machines, three industrial robots, two AGVs, several conveyor belts, detection sensors, and so on. This system is designed to have two types of production lines, the cumulative and non-cumulative way, under the assumption of the manufacturing of two products. The layout of the miniature CIM system is shown in Fig. 3.

4.2 Plant Model

The automaton of the plant G can be designed by the synchronous product (Wonham 1998) of all the automata of each component, after modeling the components of the plant such as the NC machine, the industrial robot, etc. as automata. In this paper, the number of the state and the event is minimized in the component model. This minimization is done by the projection of the events which are unnecessary to observe and unobservable by a supervisor and do not effect the behavior of a legal language towards the null event ε. For example, a velocity change of AGV is not modeled in the automaton of AGV because the velocity is controlled not by the supervisor but by a local controller of the AGV. The designed automata are projected to generate same language regardless of the states because this paper applies the supervisory control theory as the event-based approach. The number of states can also be minimized by this state projection. However, if the designed automata are changed to the nondeterministic ones after the state projection, the automata are transformed into the deterministic ones using the subset construction (Giua and Seatzu 2001).

Every component in the miniature CIM system (two AGVs, three robots, two NC machines, five conveyor belts, seventeen detection sensors, restraint pin and solenoid) are modeled as an automaton with two states. The designed plant models are shown in Figs. 4, 5, 6, 7, 8, 9, and 10. Every event defined in the miniature CIM system is listed in Table I. The automaton of the plant G is constructed by (12) using the designed automata of all components. This paper used the open software for the supervisory control theory, *TCT* (Wonham 1998), for the design and calculation of the automata. The state number of G is 4,294,967,296, which is constructed using *SYNC* function of *TCT* as shown in (12).

$$G = SYNC\left(AGV_i, ROBOT_j, NCMachine_k, ConvBelt_m, SENSOR_n, ResPin, Sol\right) \tag{11}$$

4.3 Modular Supervisor

In this section, the modular supervisors are presented for the decentralized supervisory control of the experimental miniature CIM system using Theorem 3. The specifications for the modular supervisors are shown in the following.

Specifications:

1) Buffer size of the conveyor belts 2, 4, and 6 are two work-pieces.
2) The robot 1 picks up the work-piece from the conveyor belt 1 and moves it into the NC machine 1. After the completion of machining in the NC machine 1, the robot 1 picks up the work-piece and moves it onto the conveyor belt 2.
3) The robot 2 picks up the work-piece from the conveyor belt 3 and moves it into the NC machine 2. After the completion of machining in the NC machine 2, the robot 2 picks up the work-piece and moves it onto the conveyor belt. 4
4) The robot 3 picks up the work-pieces from the conveyor belts 5 and 6 and moves those into the AGV-1 and AGV-2 separately.
5) The solenoid separates the work-pieces onto the conveyor belts 4 and 6.
6) Two AGVs unload two types of work-pieces to the specific places separately; AGV-1 and AGV-2 unload work-pieces 1 and 2 at S16 and S14, respectively.
7) AGVs travel only in counterclockwise direction and have to avoid the collision.

The modular supervisors are designed which satisfy the nonblockingness and the nonconflictness with respect to the specifications. The number of designed supervisors are eight for the specifications 1) ~ 5) and six for the specifications 6) and 7).

4.3.1 Modular supervisors for production line

Firstly, the legal languages are designed for the specifications 1) ~ 5). The eight modular supervisors are designed with respect to the designed legal languages using Theorem 3. These supervisors are shown in Figs. 11, 12, and 13.

Let us explain how the supervisor controls the plant using the example of the buffer size supervisor for the conveyor belt 2 as shown in Fig. 11(a). The control data of this supervisor are enabling all events at the initial state and the state 1 and disabling the event mv_Conv1 at the state 2. This means that the buffer size supervisor for the conveyor belt 2 will not permit the occurrence of the event mv_Conv1 after the occurrence of the string $\varepsilon \cdot mv\_Conv1 \cdot \varepsilon \cdot WP\_at1 \cdot \varepsilon \cdot mv\_Conv1 \cdot \varepsilon \cdot WP\_at1 \cdot \varepsilon$.

4.3.2 Modular supervisors for AGV

The seven modular supervisors for the supervisory control of AGVs are designed with respect to the specifications 6) and 7) using Theorem 3. The modular supervisor for the specification 6) is shown in Fig. 14. The legal languages for the specification 7) are designed as six legal languages by dividing the AGV lane into six sections as shown in Fig. 3 and then six supervisors are designed with respect to each legal language. The AGVs always travel under the supervision of these seven modular supervisors.

The AGV collision avoidance supervisor for the section 1 is shown in Fig. 15. For other sections, the collision avoidance supervisors can be easily designed by changing only transition events according to the sensor signals of each section. The AGV collision avoidance supervisor for the section 1 has the control data which disables the event mv_AGV2 and mv_AGV1 at the state 2 and 4, respectively. This means that the event mv_AGV2 and mv_AGV1 will be disabled after the occurrence of the string $\varepsilon \cdot (mv\_AGV1 + mv\_AGV2) \cdot \varepsilon \cdot AGV1\_at12 \cdot \varepsilon \cdot (mv\_AGV1 + mv\_AGV2) \cdot \varepsilon \cdot AGV2\_at17 \cdot mv\_AGV1 \cdot \varepsilon$ and the string $\varepsilon \cdot (mv\_AGV1 + mv\_AGV2) \cdot \varepsilon \cdot AGV2\_at12 \cdot \varepsilon \cdot (mv\_AGV1 + mv\_AGV2) \cdot \varepsilon \cdot AGV1\_at17 \cdot \varepsilon \cdot mv\_AGV1 \cdot \varepsilon$, respectively.

4.4 High-level Supervisor

The high-level specification of the miniature CIM system is shown in the following.

High-level specification:

1) The total buffer size of the conveyor belts 2 ~ 4 is three workpieces.

The designed high-level supervisor for the high-level specification is shown in Fig. 16. All high-level events which are not illustrated in Fig. 16, form the selfloop events at all states.

The design procedure of the high-level supervisor, as shown in Fig. 16, is specifically represented using Theorem 4 in the following. All automaton constructed during the design procedure are represented as the number of the states and the transitions because the states are too many to illustrate.

Design procedure for the high-level supervisor:

Step 0: The low-level plant G_{lo} is constructed as $SENSOR_1 \times SENSOR_2 \times SENSOR_3 \times SENSOR_4 \times SENSOR_5 \times SENSOR_6 \times SENSOR_7 \times ConvBelt_1 \times ConvBelt_2 \times ConvBelt_3 \times ConvBelt_4$. The number of states and transitions in G_{lo} are 1,024 and 13,312 respectively. The high-level legal language $K^0_{hi,1}$ is defined as the automation shown in Fig. 16 except the self-loop at every states. The designed $K^0_{hi,1}$ has 4 states and 9 transitions. Finally, the information map Θ is defined in Table II. The controllability of the high-level events is same with the low-level event defined in Table I.

Step 1: $(G_{lo})_{sub,1}$ is designed as the synchronous product of the buffer size supervisor for the conveyor belt 2 and the buffer size supervisor for the conveyor belt 4 which are shown in Fig. 11 (a) and (b) respectively. The designed $(G_{lo})_{sub,1}$ has 9 states and 102 transitions.

Step 2: $(G_{lo})^{vocal}_{sub,1}$ is constructed from $(G_{lo})_{sub,1}$. The designed $(G_{lo})^{vocal}_{sub,1}$ has 27 states and 309 transitions. The part of this construction is illustrated in Fig. 17. The event WP_at1 makes the transition from the state 0 to the state 11 and the state output becomes τ_1 in $(G_{lo})^{vocal}_{sub,1}$ as shown in Fig. 17 (b). And the state 0 becomes the state 9 and 10 after the occurrence of the events WP_at7 and mv_Conv1, respectively and the state output becomes τ_2 and τ_3 at the state 9 and 10, respectively. The state output at other states is τ_o.

Step 3: $\{(G_{lo})^{vocal}_{sub,1}\}_{OCC}$ is constructed from $(G_{lo})^{vocal}_{sub,1}$. $\{(G_{lo})^{vocal}_{sub,1}\}_{OCC}$ has 48 states and 548 transitions. This procedure is explained using Fig. 18 as follows. The state output of the case, when the event WP_at7 has occurred without the occurrence of the event mv_Conv3 at the state 0, has to be defined differently with the case when the event WP_at7 has occurred after the occurrence of the event mv_Conv3 at the state 0. Because mv_Conv3 is the controllable event, the former case cannot disable the occurrence of WP_at7 while the latter case can disable WP_at7 by disabling mv_Conv3. Therefore, in the latter case, the state output has to be defined as the controllable event. The information map, which has to be added into the information map Θ defined in Table II, is defined in Table III to solve this problem. In $\{(G_{lo})^{vocal}_{sub,1}\}_{OCC}$ shown in Fig. 18 (b), which is redesigned using the additional information map, the state 0 goes to the state 9 and the state output becomes τ_2 after the occurrence of WP_at7. And the next state becomes the state 27 after the occurrence of mv_Conv3 and if WP_at7 has occurred again, the state output becomes τ_5 but not τ_2.

Step 4: The designed $\{(G_{lo})^{vocal}_{sub,1}\}_{SOCC}$ has 39 states and 499 transitions. Because the new events τ_4 and τ_5 which are defined in Table III are not eligible in the plant, those events have to be redefined as τ_2 in this step, which are eligible high-level events. This means that the information map makes the state output as τ_2 if WP_at7 has occurred regardless of the previous string and new state outputs, i.e. new high-level events, have to be defined for the low-level event occurred after WP_at7. Let us make the partition for the low-level events as (13) before defining the new information map Θ.

$$\Pi\{(\Sigma_{lo})_{sub,1}\} = \begin{cases} \Sigma_1 = \{WP\_at1, WP\_at7, mv\_Conv1\} \\ \Sigma_2 = \{mv\_Conv3\} \\ \Sigma_3 = (\Sigma_{lo})_{sub,1} - \Sigma_1 - \Sigma_2 \end{cases} \tag{13}$$

The physical meaning of the partition $\Pi\{(\Sigma_{lo})_{sub,1}\}$ is as follows. The high-level events, which are defined in Table II, are partitioned into Σ_1. The controllable events and the uncontrollable events in the other low-level events are partitioned into Σ_2 and Σ_3, respectively. The final information map Θ is defined in Table IV according to this partition. New high-level events $\tau_6 \sim \tau_{17}$ only represent whether the controllable event, which occurred in the low-level plant, has transferred into the high-level plant. And those events are partitioned according to the state of $\{(G_{lo})^{vocal}_{sub,1}\}_{OCC}$ after the occurrence of the low-level string. The state output is τ_2 if WP_at7 has occurred at every state as shown in Fig. 19 and it becomes τ_7 if mv_Conv3 has occurred.

Step 5: The designed high-level plant $G_{hi} = \Theta[\{(G_{lo})^{vocal}_{sub,1}\}_{SOCC}]$ has 14 states and 78 transitions.

Step 6-0: The control data of $K^0_{hi,1}$ with respect to G_{hi} disables the uncontrollable event $\tau_{uc,1} \in T_{uc,1}$ which is not defined at each state. This means that $K^0_{hi,1}$ does not satisfy the controllability because it disables τ_2 and τ_1 at the state 0

and 3, respectively. Therefore, let us redesign the high-level legal language as $K^1_{hi,1}$ by adding the self-loop of these events at all states.

Step 6-1: The control data of $K^1_{hi,1}$ satisfies the controllability because it disables the controllable event τ_3 at the state 3.

Step 7: $K^1_{hi,1}$ is nonblocking as shown in Fig. 19.

Step 8: $K^1_{hi,1}$ is nonconflicting with G_{hi} because $K^1_{hi,1}$ satisfies $\overline{L_m(G_{hi}) \wedge L_m(K^1_{hi,1})} = L(G_{hi}) \wedge L(K^1_{hi,1})$ therefore, the high-level supervisor is designed as $S^{'}_{hi,1} = K^1_{hi,1}$. $G_{hi} \wedge K^1_{hi,1}$ has 52 states and 287 transitions and $S^{'}_{hi,1}$ has 4 states and 59 transitions.

Step 9: The automaton of the supremal controllable sublanguage of $K^1_{hi,1}$ has 52 states and 287 transitions with respect to G_{hi}. This automaton is the solution of the SCP, $S_{hi,1}$.

Step 10: Finally, $S^{'}_{hi,1}$ is the solution of the SCP with respect to $(K^1_{hi,1}, G_{hi})$ which has less states and transitions than $S_{hi,1}$ because it satisfies $L(S^{'}_{hi,1} \times G_{hi}) = L(S_{hi,1})$.

The designed high-level supervisor $S^{'}_{hi,1}$ makes the state transition only for the high-level events τ_1 and τ_2. And it disables mv_Conv1 at the states 3 while it enables mv_Conv1 at the states 0, 1, and 2. Therefore, only high-level events defined in Table II have meaning.

5. Implementation

5.1 CMSSM Transform

In this paper, the designed modular supervisors are transformed into the clocked Moore synchronous state machine (CMSSM) for implementation purposes. The CMSSM is a machine which has specific outputs for the current state, the input, and the clock (Wakerley 1990). The supervisor, which is transformed into the CMSSM, can be implemented as the programmable logic controller (PLC) or the digital circuit (Brandin 1994). The CMSSM of the AGV collision avoidance supervisor for section 1 is shown in Fig. 20. In Fig. 20, $D0 \sim D2$ represents the state of the CMSSM and mv_AGV1 and mv_AGV2 are the outputs of each state. And the state output is operated as the edge trigger for the current input.

There is an issue which has to be considered when the supervisor is transformed into the CMSSM. The event can be recognized into more than one event if the event occurrence time is longer than the CMSSM clock. An example of this problem is as follows. If WP_at1 has occurred at the initial state then the state will be state 1 and it will go to the state 2 after the additional occurrence of WP_at1 in the buffer size supervisor for the conveyor belt 2 shown in Fig. 11 (a). However, if the event occurrence time of WP_at1 is longer than one clock of the CMSSM as shown in Fig. 21, the CMSSM will recognize this event as several occurrences of WP_at1. As a result, the initial state will go to state 2 even only one WP_at1 has occurred. Therefore, the event which can make state transition continuously has to be differentiated when the supervisor is transformed into the CMSSM. In the case of this example, the CMSSM has to be designed by differencing WP_at1 occurring at the initial state with WP_at1 occurring at the state 1. In the CMSSM of the buffer size supervisor, the latter case of WP_at1 is redefined as $WP\_at1^{'}$ as shown in Fig. 22. Also, this means that the additional sensor for the new event $WP\_at1^{'}$ is needed for the implementation.

The logic is designed for the inputs and the outputs of the CMSSM (Wakerley 1990). In the case of the CMSSM shown in Fig. 22, the sensor signals WP_at1, $WP\_at1^{'}$, WP_at3, and $WP\_at3^{'}$ are the inputs and the control signal for the

conveyor belt 1 mv_Conv1 is the output. Finally, the logic for the CMSSM of the buffer size supervisor in the conveyor belt 2 is shown in (14), (15), (16), (17), and (18).

$$D2_{new} = (D1 \wedge D0 \wedge AGV1\_at17 \wedge \sim AGV2\_at13) \vee (D2 \wedge \sim AGV2\_at13) \tag{14}$$

$$\begin{aligned} D1_{new} = {} & (\sim D1 \wedge D0 \wedge AGV2\_at17 \wedge \sim AGV1\_at13) \vee (D1 \wedge \sim D0 \wedge \sim AGV1\_at13) \\ & \cdots \wedge (\sim D2 \wedge \sim D1 \wedge \sim D0 \wedge AGV2\_at12) \vee (D1 \wedge D0 \wedge \sim AGV1\_at17 \wedge \sim AGV2\_at13) \end{aligned} \tag{15}$$

$$\begin{aligned} D0_{new} = {} & \{(\sim D2 \wedge \sim D1 \wedge \sim D0) \wedge (AGV1\_at14 \vee \sim AGV2\_at12)\} \\ & \cdots \vee (\sim D1 \wedge D0 \wedge \sim AGV2\_at17 \wedge \sim AGV1\_at13) \vee (D1 \wedge D0 \wedge \sim AGV1\_at17 \wedge \sim AGV2\_at13) \end{aligned} \tag{16}$$

$$mv\_AGV1 = \sim D2 \tag{17}$$

$$mv\_AGV2 = \sim D1 \vee D0 \tag{18}$$

5.2 Simulation

The designed CMSSMs are verified using the circuit design and analysis software PSpice. In simulation, the outputs are tested with the arbitrary input to the CMSSM. All designed supervisors are simulated and the simulation result of the AGV collision avoidance supervisor for the section 1 is shown in Fig. 23.

In Fig. 23, $AGV1\_at12$, $AGV2\_at12$, $AGV1\_at13$, $AGV2\_at13$, $AGV1\_at17$, and $AGV2\_at17$ are the sensor signals which are used as the input and the output signals are $AGV1$ and $AGV2$ which are the control signals of the AGVs. And $D0$, $D1$, and $D2$ are the states of the CMSSM. Let us analyze the simulation result shown in Fig. 23. In the beginning, all states are 0. The state does not change even after the occurrence of $AGV1\_at17$ because $AGV1\_at17$ is the selfloop event at the initial state. And then $D0$ becomes 1 due to the occurrence of $AGV1\_at12$. At the same time, if $AGV2\_at17$ has occurred, $D1$ becomes 1 while $D0$ becomes 0. Therefore, the state of the CMSSM becomes 2 and $AGV2$ is disabled. If $AGV1\_at13$ has occurred, the state goes back to 0 and $AGV2$ will be enabled again. This means that if a certain AGV enters the section 1 and also if the other AGV enters the section 1 before the previous AGV leaves the section, the supervisor will disable the latter AGV until the previous supervisor leaves the section 1. We can see the same control action when the AGV2 enters the section 1 at first, i.e. when $AGV2\_at12$ has occurred at the state 0 in Fig. 23.

5.3 Implementation

The AGV collision avoidance supervisors for all sections are implemented and experimented as shown in Fig. 24. These supervisors control the AGVs as follows. The sensors located in the AGV lane will detect the AGV 1 and 2 and then these signals will be transmitted to the collision avoidance supervisors. Each supervisor will output the control signal to the motor driver of the AGVs using the embedded logical circuits with the transmitted sensor signal. The implemented AGV collision avoidance system is shown in Fig. 25. The implanted collision avoidance supervisors are operated in an exactly similar manner as that of the simulation result.

6. Conclusion

In this paper, the decentralized supervisory control scheme is presented for large complex systems which are modeled as discrete event systems. The proposed decentralized control scheme is divided into the modular supervisory control and the high-level supervisory control. The generalized solution for the modular supervisory control problem is presented with the concept of the sub-plant to reduce the computational complexity and it is also proved theoretically. The modular supervisors, designed using the proposed solution, are the maximum permissible and controllable sublanguage of the given legal languages with respect to the plant to be controlled. For the high-level supervisory control problem, the generalized solution

is also presented and then proved, which guarantees the hierarchical consistency. The high-level supervisors are also the maximum permissible and controllable sublanguage of the given high-level legal languages with respect to the high-level plant designed using the proposed Theorem.

The proposed decentralized control scheme is applied for the control of the experimental miniature CIM system. The miniature CIM system is modeled as 31 automata. The first eight and next six modular supervisors are designed using the proposed modular SCP solution procedure with respect to the legal languages for the production line and the AGV control respectively. In addition, one high-level supervisor, which has 52 states and 287 transitions, is designed using the high-level SCP solution procedure proposed in Theorem 4 to control the buffer size of all conveyor belts.

The designed decentralized supervisors are transformed into the CMSSM in order to apply and verify the proposed control scheme for real-world problems. The control logic is designed based on the transformed CMSSM and this logic is implemented and embedded in the digital circuits. Finally, the AGV collision avoidance system is constructed to verify the performance of the proposed control scheme. The implemented supervisors accurately perform their functions which satisfy the control specifications.

References

Basile, F., Chiacchio, P., Vittorini, V., and Mazzocca, N., 2004. Modeling and logic controller specification of flexible manufacturing systems using behavioral traces and Petri net building blocks. *Journal of Intelligent Manufacturing*, 15 (3), 351-371.

Brandin, B.A., 1994. The real-time supervisory control of an experimental manufacturing cell. *Systems Control Group Report No. 9404*. Department of Electrical and Computer Engineering, University of Toronto.

Cassandra, C.G. and Lafortune, S., 1989. *Introduction to discrete event systems*, Kluwer Academic Publishers.

Cai, K. and Wonham, W.M., 2010. Supervisor localization for large discrete-event systems. *International Journal of Advanced Manufacturing Technology*, 50 (9-12), 1189-1202.

Chung, S.Y. and Lee, D.Y., 2005, An augmented Petri net for modelling and control of assembly tasks with uncertainties. *International Journal of Computer Integrated Manufacturing*, 18 (2-3), 170-178.

Dai, X., Li, J., and Meng, Z., 2009, Hierarchical Petri net modeling of reconfigurable manufacturing systems with improved net rewriting systems. *International Journal of Computer Integrated Manufacturing*, 22 (2), 158-177.

Feng, L. Cai, K, and Wonham, W.M., 2009. A sturctual approach to the non-blocking supervisory control of discrete-event systems, *International Jouranl of Manufacturing Technology*, 41 (11-12), 1152-1168.

Ferrarini, L., Piroddi, L., and Allegri, S., 1999. A comparative performance analysis of deadlock avoidance control algorithms for FMS. *Journal of Intelligent Manufacturing*, 10 (6), 569-585.

Giua, A. and Seatzu, C., 2001. Supervisor control of railway networks with Petri nets. *Proceedings of the IEEE Conference on Decision and Control*, 5004-5009.

Golmakani, H.R., Mills, J.K. and Benhabib, B., 2006, On-line scheduling and control of flexible manufacturing cells using automata theory. *International Journal of Computer Integrated Manufacturing*, 19 (2), 178-193.

Jafari, M.A., Darabi, H., Boucher, T.O., and Amini, A., 2002. A distributed discrete event dynamic mode for supply chain of business enterprises. *Proceedings of the Workshop on Discrete Event Systems*, 279-285.

Leduc, R.J., Brandin, B.A., Lawford, M., and Wonham, W.M., 2005. Hierarchical interface-based supervisory control-part I: serial case. *IEEE Transactions on Automatic Control*, 50 (9), 1322-1335.

Leduc, R.J., Lawford, M., and Dai, P., 2006. Hierarchical Interface-based supervisory control of a flexible manufacturing system. *IEEE Transactions on Control Systems Technology*, 14 (4), 654-668.

Leduc, R.J., Dai, P., and Song, R., 2009. Synthesis method for hierarchical interface-based supervisory control. *IEEE Transactions on Automatic Control*, 54 (7), 1548-1560.

Lee, J-K., and Lee, T-.E., 2002, Automata-based supervisory control logic design for a multi-robot assembly cell. *International Journal of Computer Integrated Manufacturing*, 15 (4), 319-334.

Petin, J.-F., Gouyon, D., and Morel, G., 2007. Supervisory synthesis for product-driven automation and its application to a flexible assembly cell. *Control Engineering Practice*, 15 (5), 595-614.

Queiroz, M.H.D. and Cury, J.E.R., 2002. Synthesis and implementation of local modular supervisory control for a manufacturing cell. *Proceedings of the International Workshop on Discrete Event Systems*, 377-382.

Ramadge, P.J. and Wonham, W.M., 1987. Supervisory control of a class of discrete event process. *SIAM Journal of Control and Optimization*, 25 (1), 206-230.

Ramadge, R.J. and Wonham, W.M., 1989. The control of discrete event systems. *Proceedings of IEEE*, 77 (1), 81-98.

Ramirez-Serrano, A., Zhu, S.C., Chan, S.K.H., Chan, S.S.W., Ficocelli, M., and Benhabib, B., 2002. A hybrid PC/PLC architecture for manufacturing-system control-theory and implementation. *Journal of Intelligent Manufacturing*, 13 (4), 261-281.

Ramires-Serrano, A. and Benhabib, B., 2003, Supervisory control of reconfigurable flexible-manufacturing workcells - temporary addition of resources. *International Journal of Computer Integrated Manufacturing*, 16 (2), 93-111.

Ricker, S., Sarkar, N., and Rudie, K., 1996. A discrete-event system approach to modeling dexterous manipulation. *Robotica*, 14 (5), 515-526.

Rudie, K. and Wonham, W.M., 1992. Think globally, act locally: decentralized supervisory control. *IEEE Transactions on Automatic Control*, 37 (11), 1692-1708.

Singh, N., Sarngadharan, P.V., and Pal, P.K., 2010. AGV scheduling for automated material distribution: a case study. *Journal of Intelligent Manufacturing*, DOI: 10.1007/s10845-009-0283-9.

Son, H.I. and Lee, S., 2007. Failure diagnosis and recovery based on DES framework. *Journal of Intelligent Manufacturing*, 18 (2), 249-260.

Tittus, M. and Lennartson, B., 2002. Hierarchical supervisory control for batch process. *IEEE Transactions on Control System Technology*, 7 (5), 542-554.

Tsinarakis, G. J., Tsourveloudis, N.C., and Valavanis, K.P., 2005. Modular Petri net based modeling, analysis, synthesis and performance evaluation of random topology dedicated production systems. *Journal of Intelligent Manufacturing*, 16 (1), 67-92.

Wonham, W.M. and Ramadge, P.J., 1987. On the supremal controllable sublanguage of a given language. *SIAM Journal of Control and Optimization*, 25 (3), 637-659.

Wonham, W.M. and Ramadge, P.J., 1988. Modular supervisory control of discrete event systems. *Mathematics of Control Signals and Systems*, 1 (1), 13-30.

Wonham, W.M., 1998. *Notes on control of discrete event systems*, Department of Electrical and Computer Engineering, University of Toronto.

Wakerley, J., 1990. *Digital Design Principles*, Prentice-Hall, Inc.

Yoo, T.-S. and Lafortune, S., 2002. A general architecrue for decentralized supervisory control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 12 (3), 335-377.

Yoo, T.-S. and Lafortune, S., 2004. Decentralized supervisory control with conditional decisions: supervisor existence. *IEEE Transactions on Automatic Control*, 49 (11), 1886-1904.

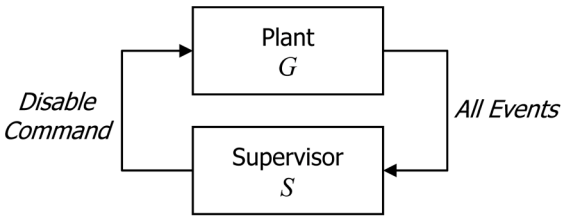


Fig. 1. Concept of supervisory control system.

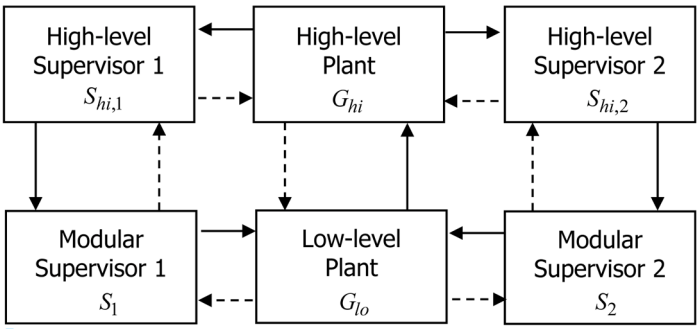


Fig. 2. Structure of decentralized supervisory control system.

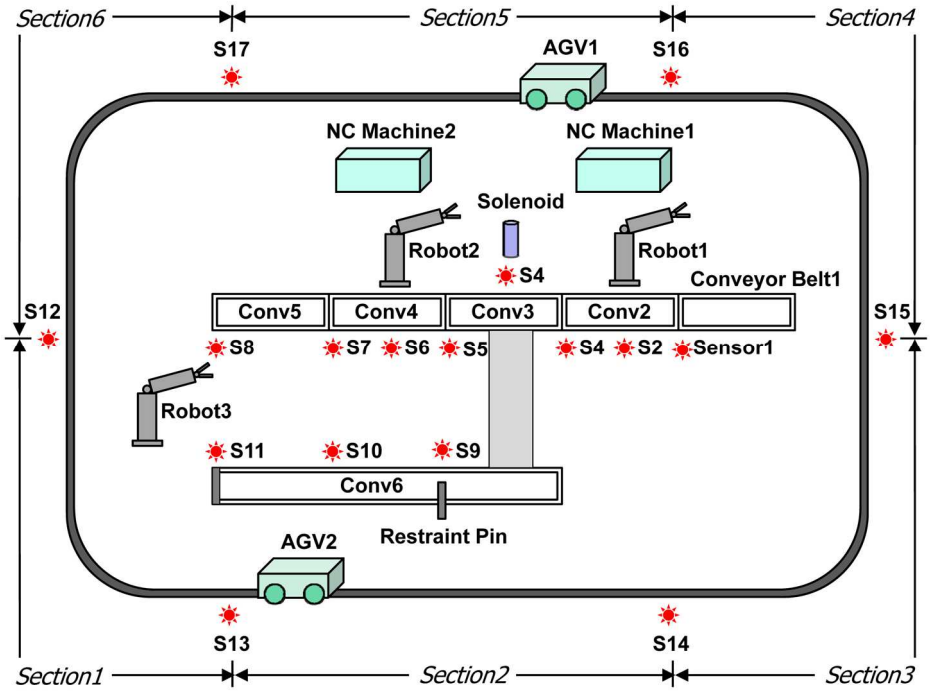


Fig. 3. Experimental miniature computer integrated manufacturing system.

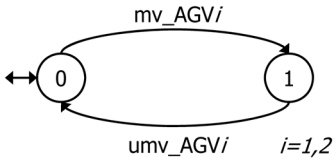
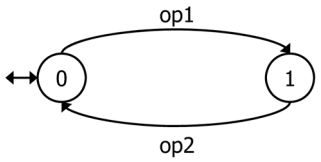
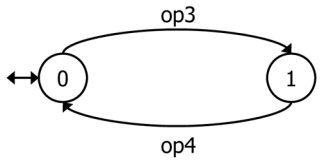


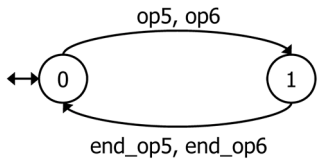
Fig. 4. Automation model of AGVs, AGV_i.



(a)



(b)



(c)

Fig. 5. Automation model of robots, $ROBOT_i$. (a) Robot 1, (b) Robot 2, (c) Robot 3.

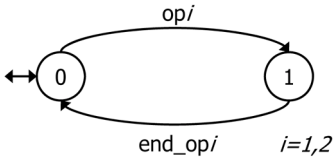


Fig. 6. Automation model of NC machines, $NCMachine_i$.

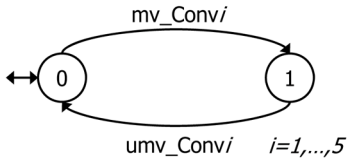
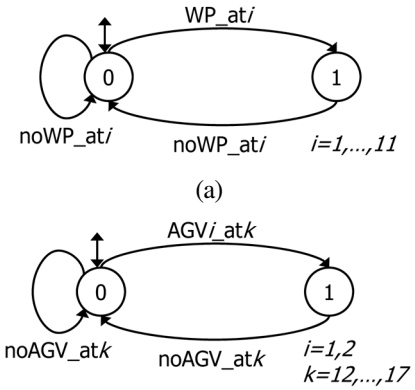


Fig. 7. Automation model of conveyor belts, $ConvBelt_i$.



(a)

(b)

Fig. 8. Automation model of sensors, $SENSOR_i$. (a) Sensor for conveyor belt, (b) Sensor for AGV.

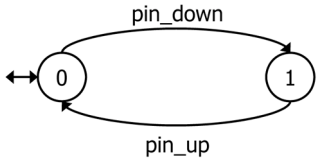


Fig. 9. Automation model of restraint pin, *ResPin* .

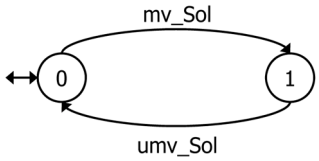
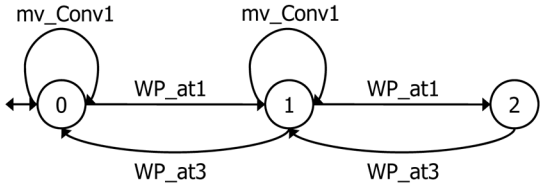
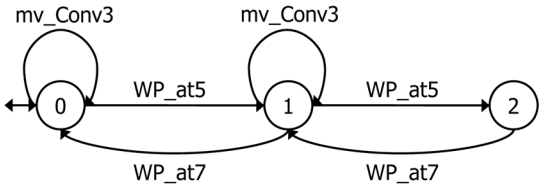


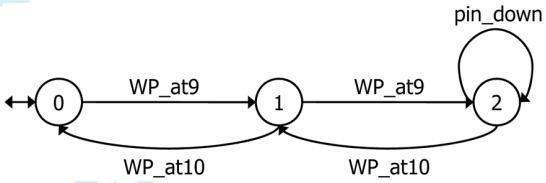
Fig. 10. Automation model of solenoid, *Sol* .



(a)



(b)



(c)

Fig. 11. Buffer size supervisor. (a) Buffer size supervisor for conveyor belt 2. (b) Buffer size supervisor for conveyor belt 4. (c) Buffer size supervisor for conveyor belt 6.

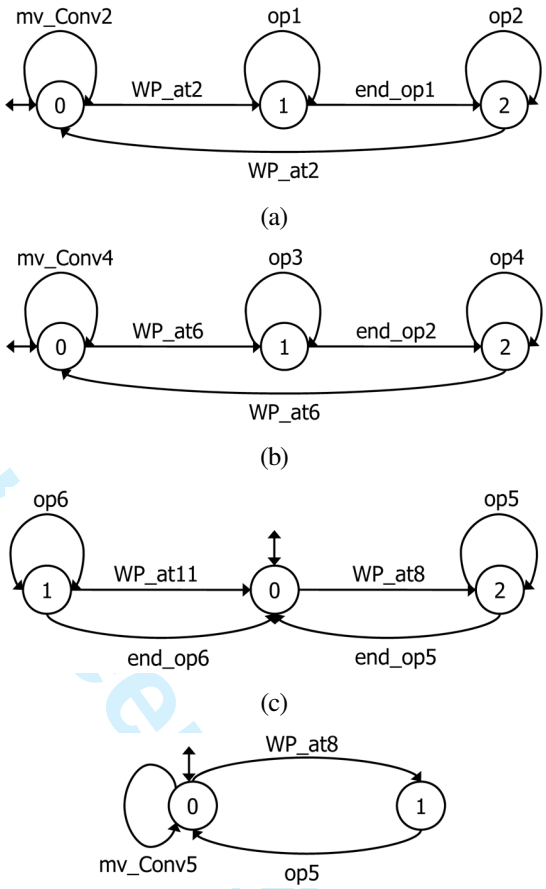


Fig. 12. Routing supervisor. (a) Routing supervisor for robot 1 and NC machine 1, (b) Routing supervisor for robot 2 and NC machine 2, (c) Routing supervisor for robot 3 and production line 1 and 2 (d) Routing supervisor for robot 3 and conveyor belt 5.

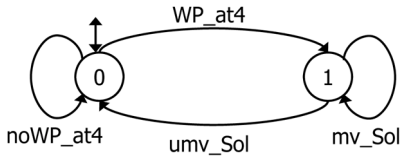


Fig. 13.Workpiece selection supervisor.

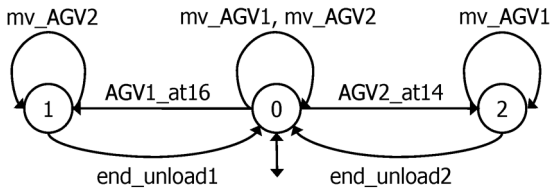


Fig. 14. AGV unloading supervisor.

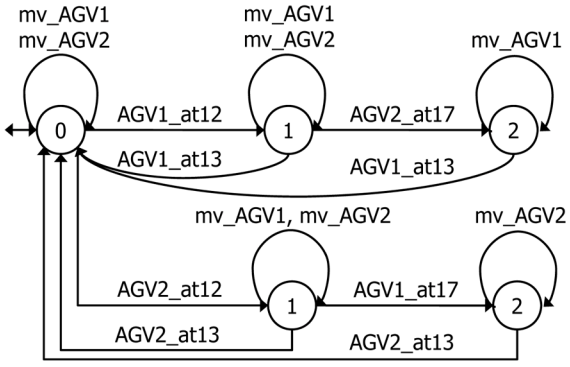


Fig. 15. AGV collision avoidance supervisor for section 1.

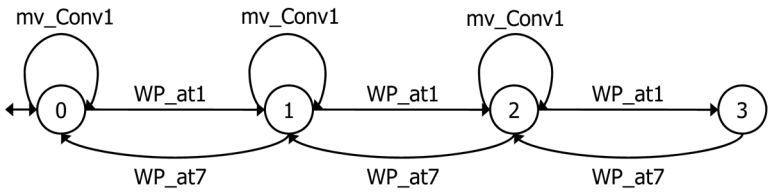
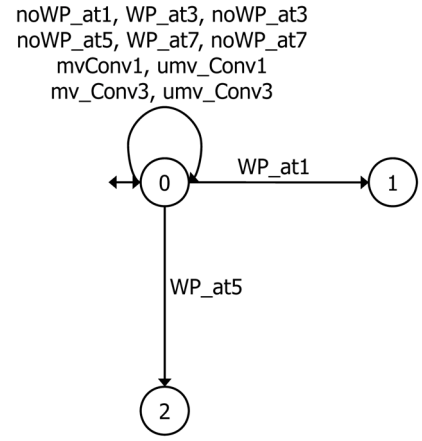
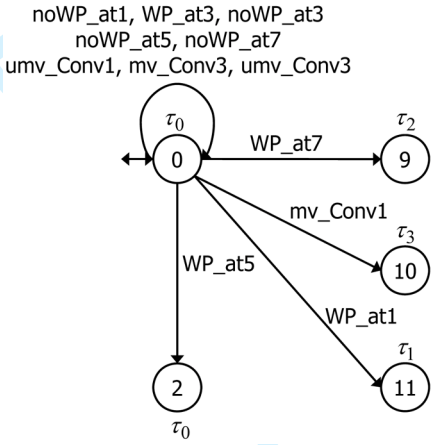


Fig. 16. High-level supervisor.

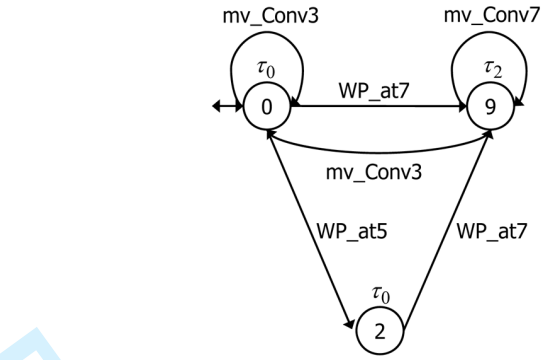


(a)

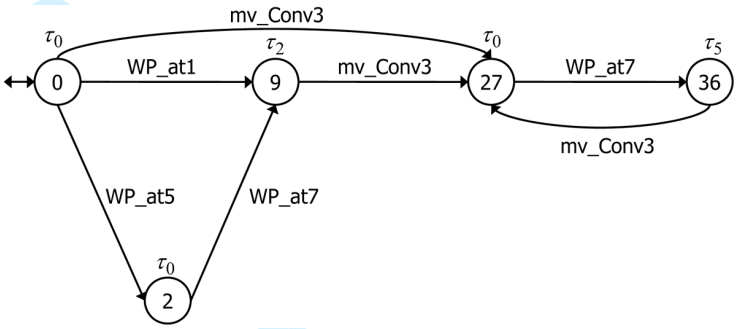


(b)

Fig. 17. (a) Part of $(G_{lo})_{sub,1}$. (b) $(G_{lo})^{vocal}_{sub,1}$ for $(G_{lo})_{sub,1}$ shown in Fig. 17 (a).



(a)



(b)

Fig. 18. (a) Part of $(G_{lo})^{vocal}_{sub,1}$. (b) $\{(G_{lo})^{vocal}_{sub,1}\}_{OCC}$ for $(G_{lo})^{vocal}_{sub,1}$ shown in Fig. 18 (a).

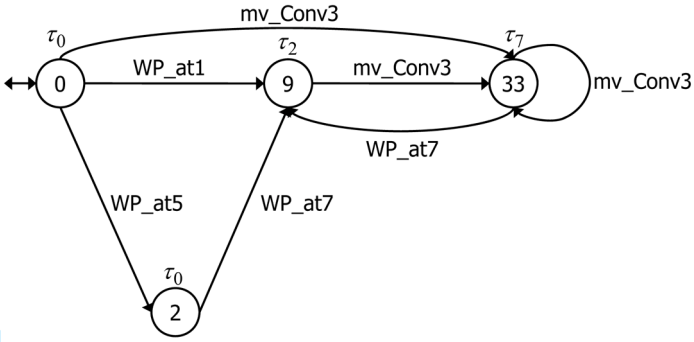


Fig. 19. $\{(G_{lo})^{vocal}_{sub,1}\}_{SOCC}$ for $\{(G_{lo})^{vocal}_{sub,1}\}_{OCC}$ shown in Fig. 18.

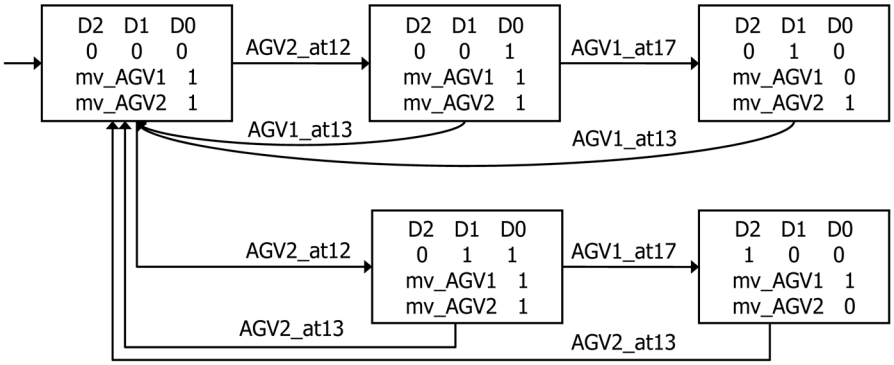


Fig. 20. CMSSM of AGV collision avoidance supervisor for section 1.

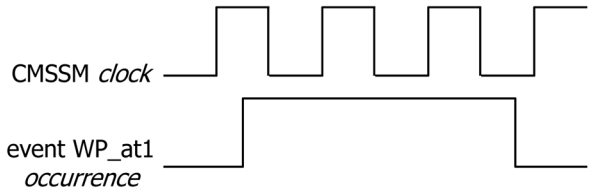


Fig. 21. Event occurrence signal.

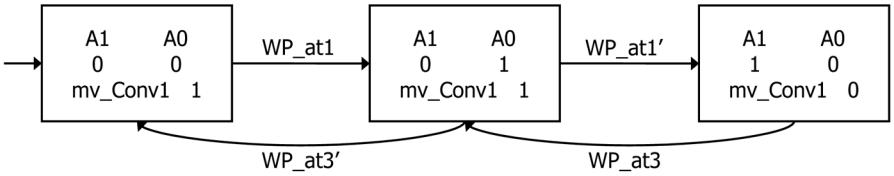


Fig. 22. CMSSM of buffer size supervisor for conveyor belt 2.

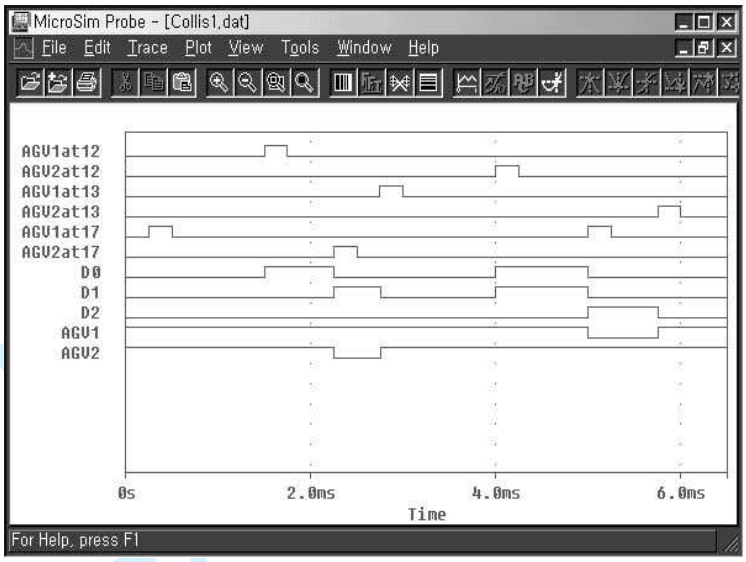


Fig. 23. Simulation result for AGV collision avoidance supervisor for section 1.

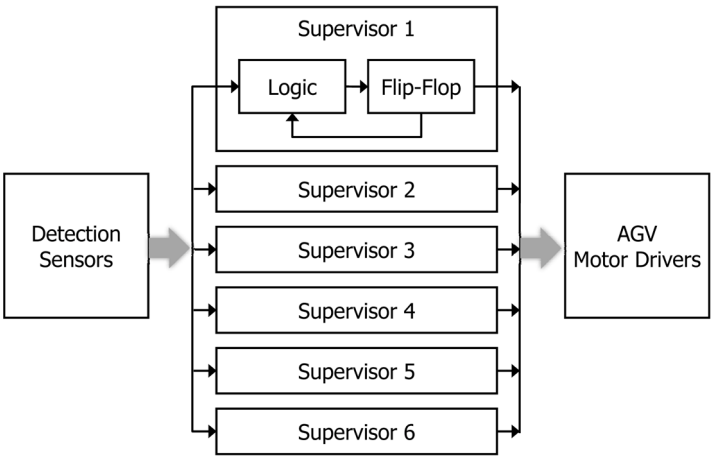


Fig. 24. Block diagram of AGV collision avoidance system.

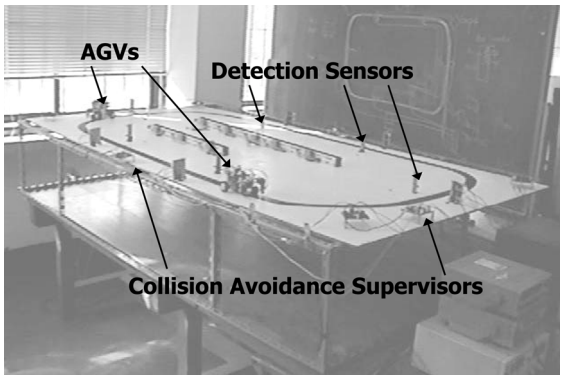


Fig. 25. Experimental AGV collision avoidance system.

Table I Event list.

Plant	Event	Controllability
AGVs	$mv\_AGV\,i$	*Controllable*
	$umv\_AGV\,i$	*Uncontrollable*
Robots	$op1$	*Controllable*
	$op2$	*Controllable*
	$op3$	*Controllable*
	$op4$	*Controllable*
	$op5$	*Controllable*
	$op6$	*Controllable*
	end_op5	*Uncontrollable*
	end_op6	*Uncontrollable*
NC Machines	$op\,i$	*Controllable*
	$end\_op\,i$	*Uncontrollable*
Conveyor Belts	$mv\_Conv\,i$	*Controllable*
	$umv\_Conv\,i$	*Uncontrollable*
Sensors	$WP\_at\,i$	*Uncontrollable*
	$noWP\_at\,i$	*Uncontrollable*
	$AGV\,i\_at\,k$	*Uncontrollable*
	$noAGV\,i\_at\,k$	*Uncontrollable*
Restraint Pin	pin_down	*Controllable*
	pin_up	*Controllable*
Solenoid	mv_Sol	*Controllable*
	umv_Sol	*Uncontrollable*

Table II. Information map Θ .

Low-level event	$\tau = \Theta(\sigma)$	High-level event
*WP_at*1	τ_1	*WP_at1*
*noWP_at*1	τ_0	*null*
*WP_at*2	τ_0	*null*
*noWP_at*2	τ_0	*null*
*WP_at*3	τ_0	*null*
*noWP_at*3	τ_0	*null*
*WP_at*5	τ_0	*null*
*noWP_at*5	τ_0	*null*
*WP_at*6	τ_0	*null*
*noWP_at*6	τ_0	*null*
*WP_at*7	τ_2	*WP_at*7
*noWP_at*7	τ_0	*null*
*mv_Conv*1	τ_3	*mv_Conv*1
*umv_Conv*1	τ_0	*null*
*mv_Conv*2	τ_0	*null*
*umv_Conv*2	τ_0	*null*
*mv_Conv*3	τ_0	*null*
*umv_Conv*3	τ_0	*null*
*mv_Conv*4	τ_0	*null*
*umv_Conv*4	τ_0	*null*

Table III. Additional information map to satisfy the condition for OCC.

Low-level sequence	$\tau = \Theta(\sigma)$	High-level event
$mv\_Conv3 \cdot WP\_at1$	τ_4	Controllable WP_at1
$mv\_Conv3 \cdot WP\_at7$	τ_5	Controllable WP_at7

Table IV. Additional information map to satisfy the condition for SOCC.

Low-level sequence	$\tau = \Theta(\sigma)$	High-level event
$\sigma_1\sigma_2$	$\tau_7, \tau_9, \tau_{11}, \tau_{13}, \tau_{15}, \tau_{17}$	Controllable event occur in G_{lo}
$\sigma_1\sigma_2\sigma_3$	$\tau_6, \tau_8, \tau_{10}, \tau_{12}, \tau_{14}, \tau_{16}$	Uncontrollable event occur in G_{lo}

Response to the Reviewers' Comments

This response regards the manuscript TCIM-2010-IJCIM-0120 "Design and Implementation of Decentralized Supervisory Control for Manufacturing System Automation" submitted to the International Journal of Computer Integrated Manufacturing. The publication decision was 'Major Revision'.

The author is grateful for the reviewers' constructive comments regarding the first manuscript. I have done my best to answer all the questions raised by the reviewers, the Associate Editor, and the Editor-in-Chief, and have revised the manuscript as per their advice. I have quoted below all the reviewers' comments in order, and put my answers after each comment.

Editor-in-Chief's Comment

Manuscript ID TCIM-2010-IJCIM-0120 entitled "Design and Implementation of Decentralized Supervisory Control for Manufacturing System Automation" which you submitted to the International Journal of Computer Integrated Manufacturing, has been reviewed. The comments of the reviewer(s) are included at the bottom of this letter.

The reviews are in general favourable and suggest that, subject to certain revisions, your paper could be suitable for publication. Could you please consider these suggestions and I do hope that you will wish to revise and re-submit.

To revise your manuscript, log into http://mc.manuscriptcentral.com/tcim and enter your Author Center, where you will find your manuscript title listed under "Manuscripts with Decisions." Under "Actions," click on "Create a Revision." Your manuscript number has been appended to denote a revision.

You will be unable to make your revisions on the originally submitted version of the manuscript. Instead, revise your manuscript and upload and submit it through your Author Centre. Please also highlight the changes to your manuscript within the document by using the track changes mode in MS Word or by using bold or coloured text.

When submitting your revised manuscript, it will help reassessment if you will please indicate how you have responded to the comments made by the reviewer(s) in the space provided. You can use this space to document any changes you make to the original manuscript. In order to expedite the processing of the revised manuscript, please be as specific as possible in your response to the reviewer(s). Should you disagree with any detail of the review, your counter-argument will be helpful and welcome.

Response: I have carefully reviewed the comments of the reviewers and have made significant changes in the manuscript. The author thanks the reviewers for providing such constructive comments which have definitely improved the quality of this paper. I have also prepared a response sheet wherein all the questions of the reviewers have been addressed. And revised parts in the manuscript are written in blue to mark up. I sincerely hope that my revised manuscript would be a worthy candidate for publication to your esteemed Journal.

Associate Editor's Comment

1. Please revise your paper and respond to the referees comments in a separate .doc file and submit this with the revised manuscript.

Response: I have carefully revised the manuscript as per the reviewer's comments. The objective and contribution of this manuscript are summarized as follows. A detailed response for each comment from the reviewers is also addressed in the following.

1. Objective of the manuscript

The first objective of this manuscript is to present a systematic and rigorous design procedure of a decentralized supervisory controller for manufacturing system automation. The proposed design procedure has to be proved theoretically. Secondly, the manuscript aims to provide a practical method for the implementation of a designed decentralized supervisory controller. And the proposed implementation method also has to be verified via both simulations and experiments.

2. Contribution of the manuscript

This manuscript proposed a systematic design procedure for a decentralized supervisory controller of a manufacturing system. The proposed design procedure is rigorous and systematic because it is designed based on the discrete event system (DES) methods and also, its correctness is proved theoretically. In detail, the manuscript proved controllability and nonblockingness which are conditions for all the supervisors, nonconflictness which is a condition for a modular supervisor and hierarchical consistency which is a condition for a high-level supervisor of the proposed design procedure theoretically.

The manuscript presented an implementation method of the decentralized supervisory controller from a practical viewpoint. We used the clocked Moore synchronous state machines (CMSSMs) which is very common and efficient for digital circuit design and also easy to use for the implementation purposes. The proposed implementation method is verified via both simulations and experiments.

2. Please have your paper proof read by a native English Speaker or a person more familiar with the English language.

Response: I have revised the entire paper and have rectified many grammatical mistakes. The manuscript is also proof read by a native English speaker.

3. Please update your references to IJCIM format as they should be alphabetical not numbered. Use the authors names in the text ie (Newman, 2007) Also please check the journal website http://www.informaworld.com/smpp/title~content=t713804665~db=all

Response: I have revised and updated the references according to the IJCIM format.

4. Please check for IJCIM appropriate references as well. You currently have only a few references from IJCIM or none.
http://www.informaworld.com/smpp/title~content=t713804665~db=all

Response: I have carefully gone through the publications of IJCIM and I have included some relevant references. This comment indeed helped me to refer some of the important references which, further proves the importance and contribution of my work in light of the existing literature. The following references were added in the manuscript.

[1] Chung, S.Y. and Lee, D.Y., 2005, An augmented Petri net for modelling and control of assembly tasks with uncertainties. *International Journal of Computer Integrated Manufacturing*, 18 (2-3), 170-178.

[2] Ramires-Serrano, A. and Benhabib, B., 2003, Supervisory control of reconfigurable flexible-manufacturing workcells - temporary addition of resources. *International Journal of Computer Integrated Manufacturing*, 16 (2), 93-111

[3] Golmakani, H.R., Mills, J.K. and Benhabib, B., 2006, On-line scheduling and control of flexible manufacturing cells using automata theory. *International Journal of Computer Integrated Manufacturing*, 19 (2), 178-193.

[4] Lee, J-K., and Lee, T-.E., 2002, Automata-based supervisory control logic design for a multi-robot assembly cell. *International Journal of Computer Integrated Manufacturing*, 15 (4), 319-334.

[5] Dai, X., Li, J., and Meng, Z., 2009, Hierarchical Petri net modeling of reconfigurable manufacturing systems with improved net rewriting systems. *International Journal of Computer Integrated Manufacturing*, 22 (2), 158-177.

5. Please resubmit your paper in 3 (THREE) .doc word files formatted in SINGLE Column representing the paper Text, Figures and Tables.

Response: I resubmitted the revised paper accordingly.

Reviewer 1

1. Collision avoidance of AGV via an experiment. Question, is this good enough to prove the proposed method to be effective?

Response: The proposed method is for a decentralized (modular and hierarchical) supervisory control. Although I verified the proposed method via modeling and simulation of a miniature CIM system, only the collision avoidance of AGV is experimentally implemented as a modular supervisory control system as the reviewer noticed. The other parts of the simulated decentralized supervisory controller (especially, high-level supervisor), however, are under implementation and they have shown similar performance to that of simulation results until now.

2. Every component of the study case is modeled with two states. This seems too simple, especially for NC machine.

Response: I modeled each manufacturing component as automaton with the minimum number of states because other states can be redundant in a framework of the supervisory control of discrete event systems. More explanation about the plant model is presented in the 'Plant Model' section. Basically, I followed the modeling procedure of manufacturing components presented in [1] and [2]. Additional states can also be added if those represent a fundamental behavior of components for example; a failure state can be added for NC machine. A failure of machine, however, is not considered in this research. Modeling with a failure is considered in the author's other research [3].

[1] Ramadge, R.J. and Wonham, W.M., 1989. The control of discrete event systems. *Proceedings of IEEE*, 77 (1), 81-98.

[2] Feng, L. Cai, K, and Wonham, W.M., 2009. A sturctual approach to the non-blocking supervisory control of discrete-event systems, *International Jouranl of Manufacturing Technology*, 41 (11-12), 1152-1168.

[3] Son, H.I. and Lee, S., 2007. Failure diagnosis and recovery based on DES framework. *Journal of Intelligent Manufacturing*, 18 (2), 249-260.

3. Will the hierarchical state machine (HSM) be helpful to solve SCP? If Petri Net method is mentioned, it is necessary to investigate FSM/HSM etc as literature survey.

Response: I have added more literature survey about the hierarchical supervisory control as shown below. The high-level supervisor proposed in this manuscript is by far less complex than the normal one because an information map Θ filters the meaningless states in a high-level plant. Generally, it is known that a hierarchical supervisory control is much less complex than a non-hierarchical supervisory control as also proved in the following references.

[1] Tittus, M. and Lennartson, B., 2002. Hierarchical supervisory control for batch process. *IEEE Transactions on Control System Technology*, 7 (5), 542-554.

[2] Leduc, R.J., Brandin, B.A., Lawford, M., and Wonham, W.M., 2005. Hierarchical interface-based supervisory control-part I: serial case. *IEEE Transactions on Automatic Control*, 50 (9), 1322-1335.

[3] Leduc, R.J., Dai, P., and Song, R., 2009. Synthesis method for hierarchical interface-based supervisory control. *IEEE Transactions on Automatic Control*, 54 (7), 1548-1560.

Reviewer 2

1. 4) Does the paper appropriately compare the performance of proposed methodologies with those found in the published literature?

: No. There is good background review in the introduction, but the paper would benefit by more detail on the value of the proposed algorithm vs. other algorithms. Aren't some of the steps computationally expensive? In particular, in theorem 3, it would be good to give some discussion of the complexity of some of the steps (controllability test, nonconflicting test, nonblocking test, etc.). Since the algorithm iterates and does some of these test repeatedly, is this computationally expensive?

Response: I absolutely agree with the reviewer. The author cannot argue that the proposed algorithm is better than the previous ones from the viewpoint of computational complexity because there is an iterative procedure in Theorem 3 as well as Theorem 4. However, I would like to add the following justification. Computational complexity of controllability test, nonblocking test, and nonconflicting test is same with the one proposed by Ramadge and Wonham (1987, 1989) because Theorems 1 and 2 are proposed based on those studies. I, however, tried to reduce the computational complexity using a decentralized supervisory control which can reduce the computational complexity from $O(mn^2)$, an exponential function, to $O(mn)+...+O(mn)$, a proportional function. The iterative procedure can also increase the computational expense, however; this procedure is for the reconstruction of legal languages when they do not satisfy the controllability, nonblocking, and nonconflicting conditions. Therefore, the iterative procedure, fundamentally, does not increase the computational complexity of those tests. The author is also working on presenting a systematic method for the reconstruction of legal languages when they do not pass the controllability, nonblocking, and nonconflicting tests to make the proposed algorithm more practical.

2. 7) Is adequate credit given to other contributors in the field and are references sufficiently complete? (Please indicate any significant omissions.)

: Sufficient references, but would like more discussion of the complexity of certain steps in the algorithm.

Response: I have explained the computational complexity of the proposed algorithm in the 'Decentralized Supervisory Control' chapter. And please refer the first response for reviewer 2.

3. 9) Is the paper clearly, concisely, accurately and logically written? Are there any errors? Could it benefit from condensing or expansion? (Please give details.)

: Generally yes.

**** Fig 14 arrows are not clear -- it appears that once leaving state 0 can never return?**

Response: There is a mistake in the naming of states in the original figure. I have corrected the figure and revised it to make arrows more clear.

4. 9) ** Theorem 4 step 2 is not clear what is being done.

Response: The purpose of the step 2 in Theorem 4 is to construct a new automaton $(G_{lo})^{vocal}_{sub,i}$ which has vocal states defined in an information map Θ from a low-level plant G_{lo}. The step 2 has been revised to make it more clear.

4. 9) ** Example of Fig 3: would help if better explanation of AGV operation. Other than the load station at S12, do the AGVs do anything else? Are they restricted to only clockwise or counterclockwise travel, or both?

Response: I have added more explanation about the AGV operation in the 'Modular Supervisor' and 'Modular supervisors for AGV' sections. The AGVs can travel only in counterclockwise direction. A role of the AGVs is that the AGV-1 and AGV-2 load the work-pieces 1 and 2, respectively at the S12 and AGV-1 and AGV-2 unload their work-pieces at the S16 and the S14, separately. Except the loading and unloading operations, the AGVs always travel under the supervision of the AGV unloading supervisor and the AGV collision avoidance supervisors presented in Figs. 14 and 15, respectively.