International Journal of Production Research
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tprs20

Design of a rule-based flexible manufacturing system controller using modified IDEF0 methodology

S. G. Chen , M.Z. Wu & R.K. Li
Published online: 15 Nov 2010.

To cite this article: S. G. Chen , M.Z. Wu & R.K. Li (1997) Design of a rule-based flexible manufacturing system controller using modified IDEF0 methodology, International Journal of Production Research, 35:10, 2793-2820, DOI: 10.1080/002075497194453

To link to this article: http://dx.doi.org/10.1080/002075497194453

INT. J. PROD. RES., 1997, VOL. 35, NO. 10, 2793–2820

Design of a rule-based flexible manufacturing system controller using modified IDEF0 methodology

S. G. CHEN†, M. Z. WU‡ and R. K. LI§*

This article proposes a new method to design a rule-based FMS controller. This approach adopts modified IDEF0 (MI) diagrams as a graphical representation of the production rules. Initially, the material flows or functional requirements for the FMS are specified by synthesizing the MI diagram primitives. The control flows for the FMS are then created by a number of transformation rules. The manufacturing policy, e.g. deadlock avoidance policy, is also specified and attached. The MI diagrams are therefore transformed to the final MI (FMI) diagrams which can be directly transformed to the production rules. Thus, a concise rule-based FMS controller is developed. The fact that the production rules are created by systematic transformation eliminates any redundant, contradictory, or unnecessary rules. Two approaches can verify the consistency of the designed FMS controller. The controlled Petri net approach can be adopted for verifying a small system, while a simulation approach is preferable for verifying a large system. By iterating the process, a feasible rule-based FMS controller can be systematically developed. This method not only provides graphical representations to construct well-organized production rules, but also includes a systematic transformation of the control flows for an FMS controller.

1. Introduction

Developing an FMS controller is a complicated process. Such an undertaking stipulates that various requirements be satisfied: (1) connectivity – the controller should have the ability to connect to lower-level and upper-level devices; (2) configurability – the controller must be highly configurable in software; (3) software portability – the controller system designer should allow application software to be reusable in the future, regardless of hardware changes; (4) optimizibility – the controller should have the ability to achieve a better planning, scheduling, and control solution for a specific application; and (5) intelligibility – the controller should have the ability to diagnose the unpredicted errors occurring in the cell and recovery from the errors (Xiang and O'Brien 1995). Many practitioners, users and researchers have proposed several methods regarding these topics. These contributions include defining the controller (Franks *et al.* 1990), proposing the controller's design requirements (Bauer *et al.* 1991), defining the controller's architecture (Jones and McLean 1986), and modelling the controller's behaviour (Zhou *et al.* 1992). Notably, on the implementation aspects, Jafari and Boucher (1994) presented a high-level specification

Received February 1996.

† Graduate Student, Institute of Industrial Engineering, National Chiao Tung University, Taipei, Taiwan, R.O.C.

‡ Director of Technical Department, AsiaTEK Inc., Taipei, Taiwan, R.O.C.

§ Professor, Institute of Industrial Engineering, National Chiao Tung University, 1001, Ta-Shieh Rd., Hsin Chu, Taiwan, R.O.C.

* To whom correspondence should be addressed.

model to implement the ladder logic approach for the FMS controller. They applied the IDEF0 methodology to specify the logic of activities. Next, the interpreted Petri nets (PNs) (David and Alla 1992) were transformed from these IDEF0 diagrams. After analysing the PNs, the ladder diagrams were transformed from them. Their efforts made the ladder diagrams more tractable. However, the software portability, optimizibility, and intelligibility for the ladder logic approach are still difficult to pursue. Murata *et al.* (1986) proposed the Petri net-based approach to design the controller. Petri nets (PNs) have been applied to modelling, specification, verification, analysis, performance evaluation, control, and simulation of automated manufacturing systems (Cecil *et al.* 1992). These applications are typically initiated by properly designing PN models for manufacturing systems. Next, the PN's properties are verified. A specific hardware for the PN-based controller should be adopted. The designed PNs can be directly executed on the specific hardware. In view of the potential costly analysis to verify the model's validity for the system, Zhou *et al.* (1992) proposed using a hybrid methodology to synthesize PN models for a manufacturing system. They developed some PN primitives for top-down and bottom-up syntheses of the PN model for the system. Consequently, the costly analysis to the synthesized PNs could be omitted. However, the PN formalisms were often disrupted by the cluttered flow diagrams when modelling complex manufacturing systems and were dissatisfied by the heavy consumption of computational resources when performing simulations (Cecil *et al.* 1992). The software portability, optimizibility and intelligibility for this approach are also difficult to fulfil.

Recently, developing an expert system to control an FMS has received increasing attention (Sauve and Collinot 1987, Wu and Wysk 1988, Teng and Black 1989). Others have indicated that this approach has high potential to satisfy the various requirements to design an FMS controller (Kusiak 1990, Meyer 1990). An expert system normally consists of a rule-based knowledge, an inference engine and a user interface, while the rule-based knowledge is the main part of an FMS controller. The production rules (Valette 1987, Kusiak 1990) are normally involved in designing a rule-base knowledge. Several advantages of using this approach are as follows. First, a rule-based controller has more flexibility and extendibility than the controller designed by other approaches. Second, adding other knowledge bases, e.g. fault diagnosis and troubleshooting, to the controller is relatively easy. Third, modifying and maintaining a rule-based controller can be achieved without breaking the entire system. However, inefficiency of program execution may be encountered when a large set of production rules exists, which may contain too many inconsistent, redundant, contradictory or unnecessary rules. Therefore, how to design a rule-based controller consistently and concisely is particularly challenging.

Hong (1993), Liang and Hong (1994) proposed a knowledge acquisition process called Hierarchy Transformation Method (HTM) or IDEF0/CPN/G2 approach, consisting of a series of transformations from IDEF0 diagrams to the coloured PNs and to the knowledge base of G2 (Gensym 1994) expert system. This approach attempted to systematically construct a rule-based system. However, its application was limited to the repetitive manufacturing system. In their study, constructing the IDEF0 diagrams was based mainly on heuristics and could not avoid redundancy. Moreover, the transformation rules involved in the process were also too weak to be followed, thereby limiting its use.

This study proposes a new method to design a rule-based FMS controller. This approach adopts modified IDEF0 (MI) diagrams as a graphical representation of the

production rules. Initially, the material flows or functional requirements for the FMS are specified by synthesizing the MI diagram primitives. The control flows for the FMS are then created by a number of transformation rules. The manufacturing policy, e.g. deadlock avoidance policy, is also specified and attached. The MI diagrams are therefore transformed to the Final MI (FMI) diagrams which can be directly transformed to the production rules. Thus, a concise rule-based FMS controller is developed. The fact that the production rules are created by systematical transformation eliminates any redundant, contradictory or unnecessary rules. Two approaches can verify the consistency of the designed FMS controller. The C′PN approach (Holloway and Krogh 1990) can be adopted for verifying a small system; while a simulation approach is preferable for verifying a large system. By iterating the process, a feasible rule-based FMS controller can be systematically developed.

This method not only provides graphical representations to construct well-organized production rules, but also includes a systematical transformation of the control flows for an FMS controller.

2. The design procedure for an FMS controller

Figure 1 illustrates an FMS controller's design procedure, as expressed by the IDEF0 diagram. Initially, synthesizing the corresponding MI diagram primitives in relation to the system's requirements allows for the material flows and resource utilization of an FMS to be specified. Section 3 provides details of the MI diagrams for each FMS primitive. These diagrams construct the system specification for the FMS. The basic control flows for each of the FMS components is created by applying the transformation rules. Section 4 outlines the transformation rules. However, avoiding a deadlock and enhancing the system performance require additional controls to manipulate the material flows in the system. These additional controls are

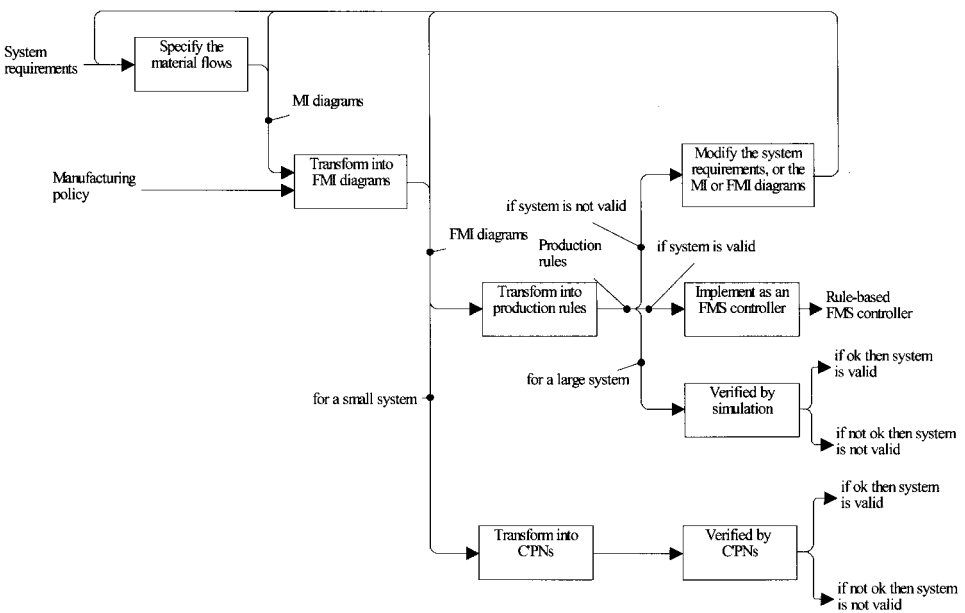


Figure 1. The proposed scheme of system development.

considered as manufacturing policies (or strategies). A different policy would cause a different system behaviour and performance. Thus, selecting the optimal policy depends on the performance index used in the system. Moreover, control policy design also depends on the system configuration employed. The same policy has a different design for each different system configuration.

The MI diagrams are then transformed into the FMI diagrams. These FMI diagrams provide road maps of control logic for the system and are important documents in the life cycle of the system's development. The control flows in the FMI diagrams can be transformed into the production rules and the C′PNs. The production rules can be executed on an expert system. Thus, an FMS controller is developed. The controller's properties can be verified by considering two approaches. For a small system, they can be verified by the properties of C′PN. For a large system, although the corresponding C′PN can be treated by transformation, a simulation model is preferred. This is owing to the fact that analysing a large C′PN is costly and may be infeasible. Meanwhile, a complete testing of a simulation model can still produce high quality implementation (Goodenough and Gerhart 1975). The modification information is then fed back to the specification or the design stages to correct the system's improper design. The design process is therefore iterated until a satisfied design is achieved.

3. The design primitives for the FMS components

IDEF0 is well known for its ease to use and to follow when specifying a system's functionality, thereby making itself a general purpose method for activities modelling. Each arrow and activity box has no specific formats or constraints for application. In the context of an FMS, however, explicit definition of arrows and boxes can clarify the specification of an FMS operation. In this article, some modifications for the IDEF0 definitions are made and an example of an MI diagram is shown in Fig. 2. The material flows, e.g. parts, pallets, and fixtures, are depicted as the bold arrows. The control flows for an activity are depicted as the hairline arrows. Three types of control flows are employed: controllable expressions, controlled expressions and external expressions. A controllable expression specifies a state equation of a variable which can be altered by the function block's activity. A controlled expression specifies a state equation of a variable which is altered by the function block's activity. An external expression which is a tunnelled-tail arrow specifies a state

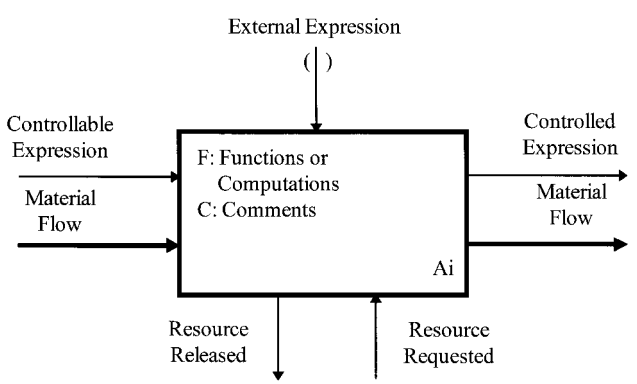


Figure 2. The basic configuration of an MI diagram.

equation of variables which constrain the function block's activity. Normally, the variables for the controllable and controlled expressions are internal system state variables and have one-to-one correspondences. They describe the states of resources used in the system. The variables for the external expressions are variables which describe the states of external events, e.g. the state of a sensor, or manufacturing policy constraints. The resources requested or released by a function block's activity are depicted as the hairline arrows. The activity describing a function block has two formats: one is a string for a decomposed function block; the other is described as

F: for computations and functions,
C: for comment

and is for a primitive function block.

In an FMS, two sets of entities are identified. One is the set of active entities, which include flexible machines, robots, AGVs, conveyors, and AS/RSs. An active entity can be controlled by a controller and performs specific tasks. The other is the set of passive entities, which include raw materials, pallets, fixtures, and finished products. A passive entity is handled by the active entities and cannot be directly controlled by a controller. Therefore, the active entities construct the resources employed in the system, while the passive entities describe the system's material flows. An FMS generally consists of flexible machines and transportation vehicles, e.g. AGVs and robots. They can be classified into three categories: the shared resources, the non-shared resources and the buffers. Each type of resource is specified in the following by the MI diagrams as the design primitives for an FMS controller.

3.1. *The non-shared resources*

In a control software, each controlled entity must be identified by the controller. Therefore, each flexible machine must be identified by a controller regardless of whether a flexible machine's multiplicity is more than one. A non-shared resource is dedicated to the production of some material or operation and is not shared by the other operations. Figure 3 (a) illustrates the MI diagram for such a resource. Assume that several tasks for part 1 are handled by the non-shared resource. Part 1 requests the non-shared resource, performs a series of tasks, and finally releases the non-shared resource. A non-shared resource can be any activity entity in an FMS if it is not shared by the other operations.

3.2. *The shared resources*

A shared resource is shared by several operations in any precedence. For instance, an AGV can be shared by several production lines for transferring workpieces among the workstations. Therefore, the AGV is a shared resource. Figure 3 (b) depicts the MI diagram for such a resource. Assume that several tasks are assigned to parts employing the same resource. Each line of tasks is similar to those handled by the non-shared resource. Deadlocking (Coffman *et al.* 1971) may occur when the resource is shared by several tasks. Section 4 discusses those conditions.

3.3. *The buffers*

Buffers are common in most manufacturing systems. Storage areas, stocks or even conveyors are buffers. Zhou and DiCesare (1990) discussed the PN modelling of

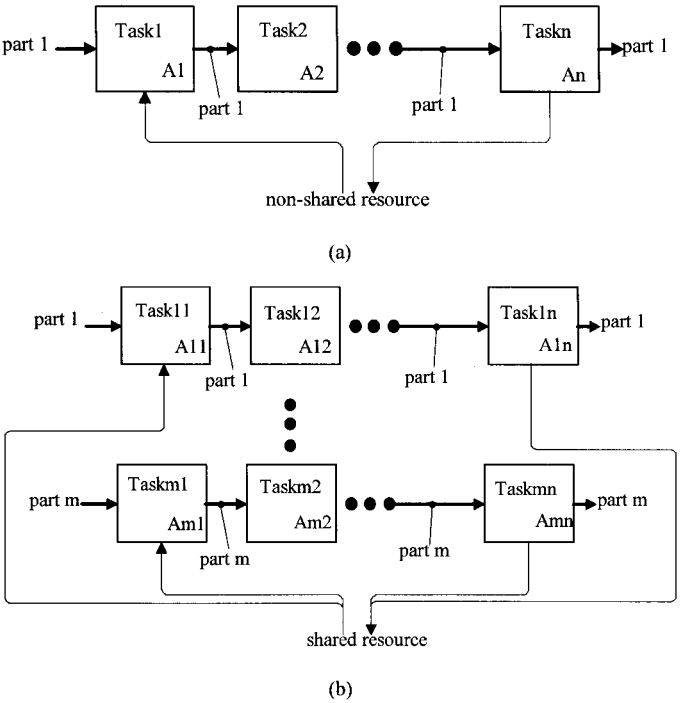


Figure 3. (a) The MI diagram for a non-shared facility. (b) The MI diagram for a shared facility.

buffers including simple, safe and generalized ones. Here, the buffer specifications are explored by MI diagrams. Buffers may either have control elements or not; however, they all have capacity constraints. If having control elements, the buffers can be manipulated by the controller. That is, such a buffer would be accompanied with a resource entity. If not having control elements, they are the constraints to the controller. The buffers are classified here according to the nature of current devices: the simple buffer, the ordered buffer and the generalized buffer. Each can be shared or non-shared by different operations. Only the shared buffer for each category is discussed here since a non-shared buffer is a special case of a shared buffer.

3.3.1. *A simple buffer*

A simple buffer is simply a buffer for temporary storage where each element in the buffer has no precedent relations between each other. For instance, a storage area or an AS/RS is a simple buffer. Figure 4 (a) depicts the MI diagram for such a buffer. Two parts, part 1 and part 2, with four tasks share a simple buffer 'buffer 1' which has three capacities. The diagram closely resembles those for a shared resource except that the buffer has a capacity denotation.

3.3.2. *An ordered buffer*

An ordered buffer is one in which the elements have precedent relations. A conveyor is normally an ordered buffer in a manufacturing system, except that the conveyor is a flexible conveyor where several workstations share this conveyor, and

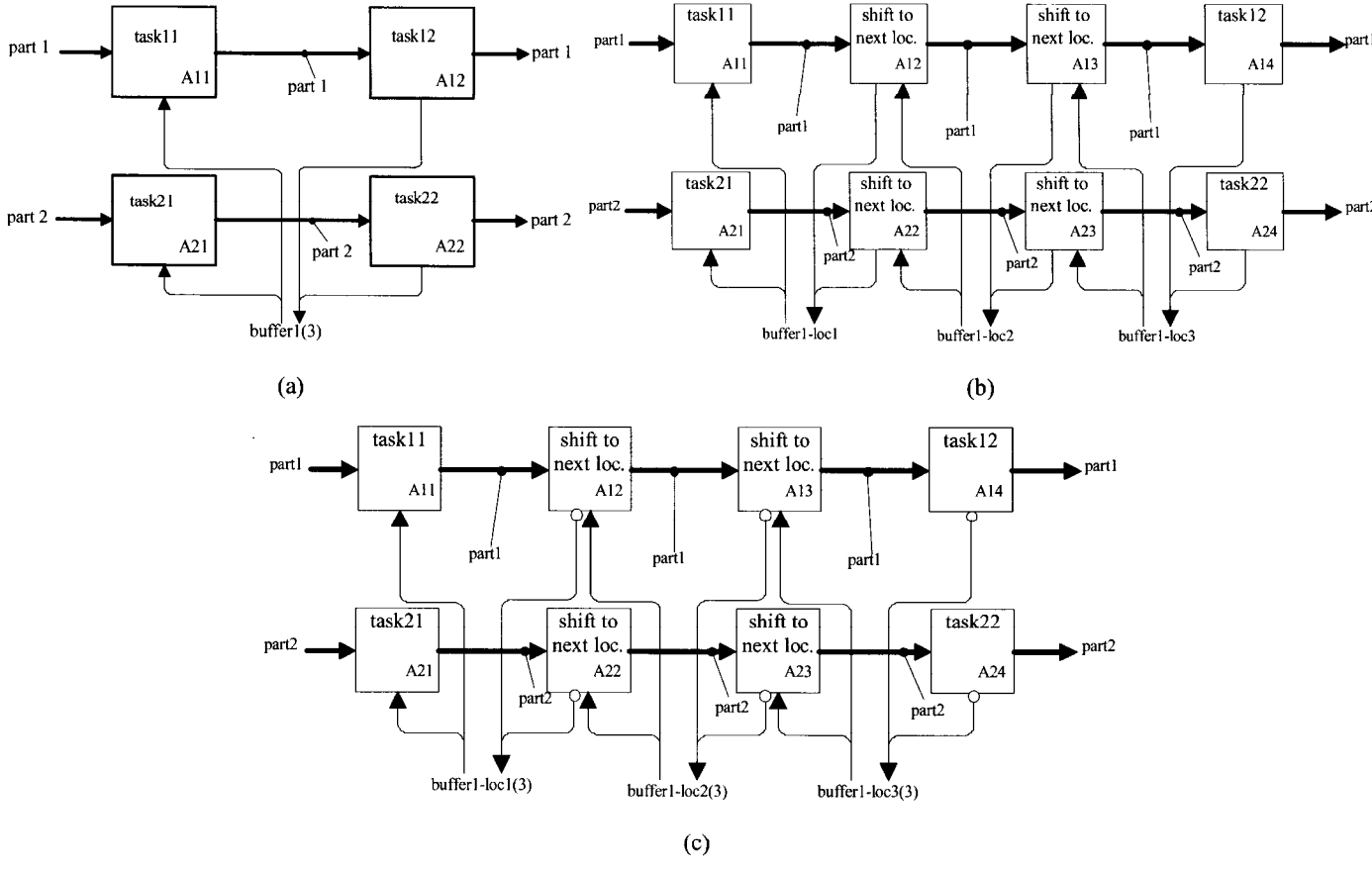


Figure 4. (a) The MI diagram for specifying a shared simple buffer1 with capacity three. (b) The MI diagram for specifying an ordered buffer1 with capacity three which is shared by two parts. (c) The MI diagram for specifying a generalized buffer1 with three locations where each location has three capacities.

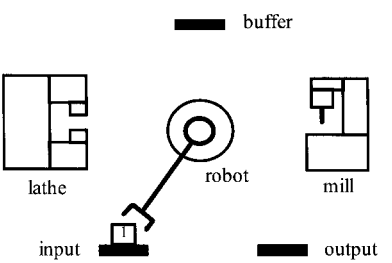
the parts can be avoided or passed from some of the workstations (Dupont-Gatelmand 1982). Such flexible conveyors are considered as simple buffers since no precedent relations can be derived from the parts upon the conveyors. Figure 4 (b) depicts the MI diagram for such an ordered buffer. Two parts, part 1 and part 2, transport on a normal conveyor alternately. This conveyor is an ordered buffer with three capacities. Each capacity is denoted as a location in this buffer. If a location is occupied by one part, the next part must wait until the location is released.

3.3.3. *A generalized buffer*

A generalized buffer is a combination of a simple buffer and an ordered buffer. For instance, a tow-line conveyor which drags several carts of small parts is a generalized buffer in manufacturing systems. The carts on the conveyor have precedent relations; however, the parts in the cart do not have any. Figure 4 (c) depicts the MI diagram for such a buffer. An ordered buffer 1 with three locations (where each location has three capacities) serves the manufacturing of two parts, part 1 and part 2. The circles in the diagram denote the inhibition of activities. When the capacities of the location of buffer 1 are not all requested, the cart is not allowed to shift to the next location.

3.4. *An illustrative example of a robotic FMC*

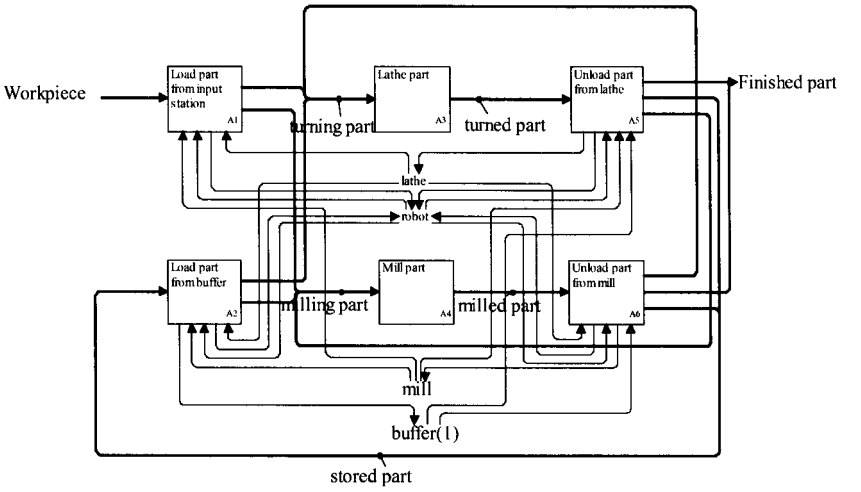
This section applies the above design primitives for the FMS components to illustrate the synthesizing process for the functional specification of the robotic FMC example. Figure 5 provides its layout. The robot serves two machines, a lathe and a mill, with one loading station, one unloading station and a simple buffer with one capacity. Four types of products are to be manufactured as depicted in Fig. 5. Each type has a different production route. The functional specification for such a manufacturing system can be readily available by synthesizing the design primitives of shared resources and buffer. Figure 6 displays the MI diagrams for the FMC. The A0 page is an overview of the functional specification. The part is



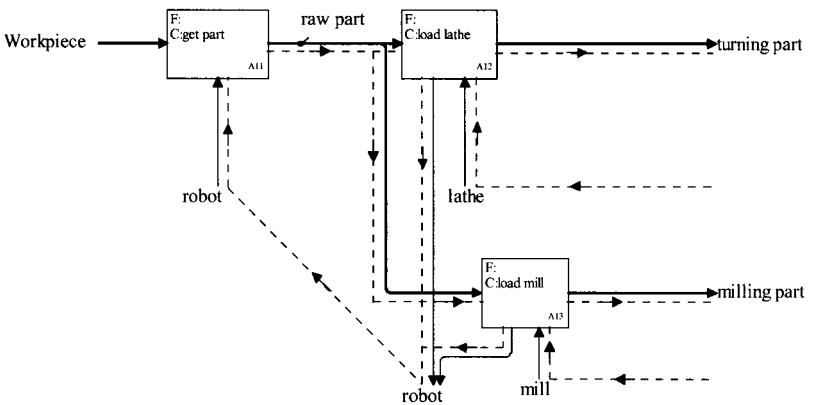
Products	Routes
1	I-L-O
2	I-M-O
3	I-L-(B)-M-O
4	I-M-(B)-L-O

I : Input loading
O: Output unloading
L : Lathe operation
M: Mill operation
B : Buffer storage (option)

Figure 5. The layout of a robotic FMC and its products.



(a) The A0 page



(b) The A1 page

Figure 6. The MI diagrams for the FMC: (a) The A0 page. (b) The A1 page.

initially loaded from the input station and then machined by a mill or lathe depending on its type. Next, it may be finished, moved on to the next operation, or stored in a buffer waiting for the machine to be ready. The A1 page describes that the robot is shared by two flows of tasks: for turning and for milling. With respect to an FMS, a branch of material flow refers to selecting the manufacturing routes. In this case, a part loaded from the input station woud be machined exclusively by a lathe or mill.

4. The transformation rules

Three kinds of transformations are proposed here, i.e. for the FMIs, for the production rules and for the C′PNs. They are detailed in the following.

4.1. *The transformation for the FMIs*

An FMS controller's control flows can be created by transforming the synthesized MI diagrams. Five rules are employed to perform such transformations.

- *Rule* 1. Creating control flows for the resource is based on the resource utilization cycles which are formed by connecting the requesting resource arrows, the material flows and the releasing resource arrows.

For instance, the dashed cycles in Fig. 6 (b) which denote A11 and A12 or A11 and A13 form the resource utilization cycles for the robot. A12 forms a part of the resource utilization cycle for the lathe. A13 forms a part of the resource utilization cycle for the mill. Each cycle is replaced by a state-transition cycle which describes the state transition of the variable representing the machine status. That is, the machine-status variable controls the machine's transitions (the operations of the machine). Figure 7 (a) denotes that the dashed cycles, A11 and A12 or A11 and

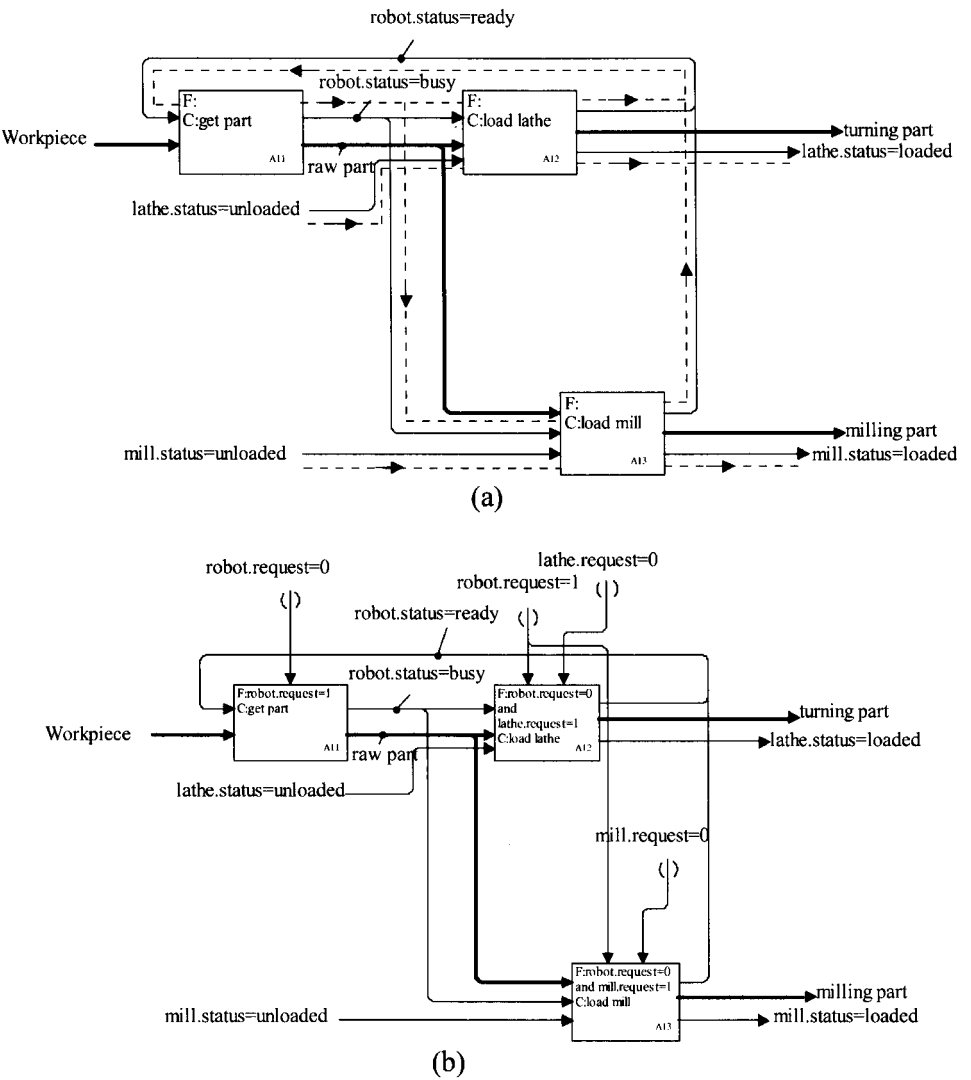
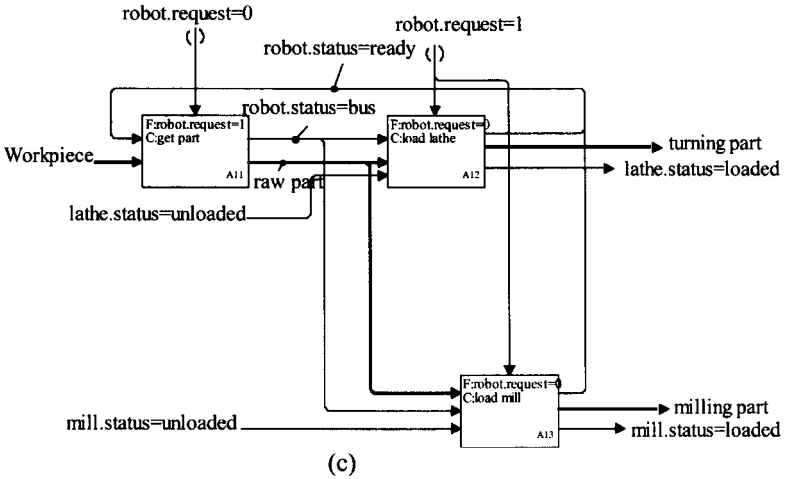
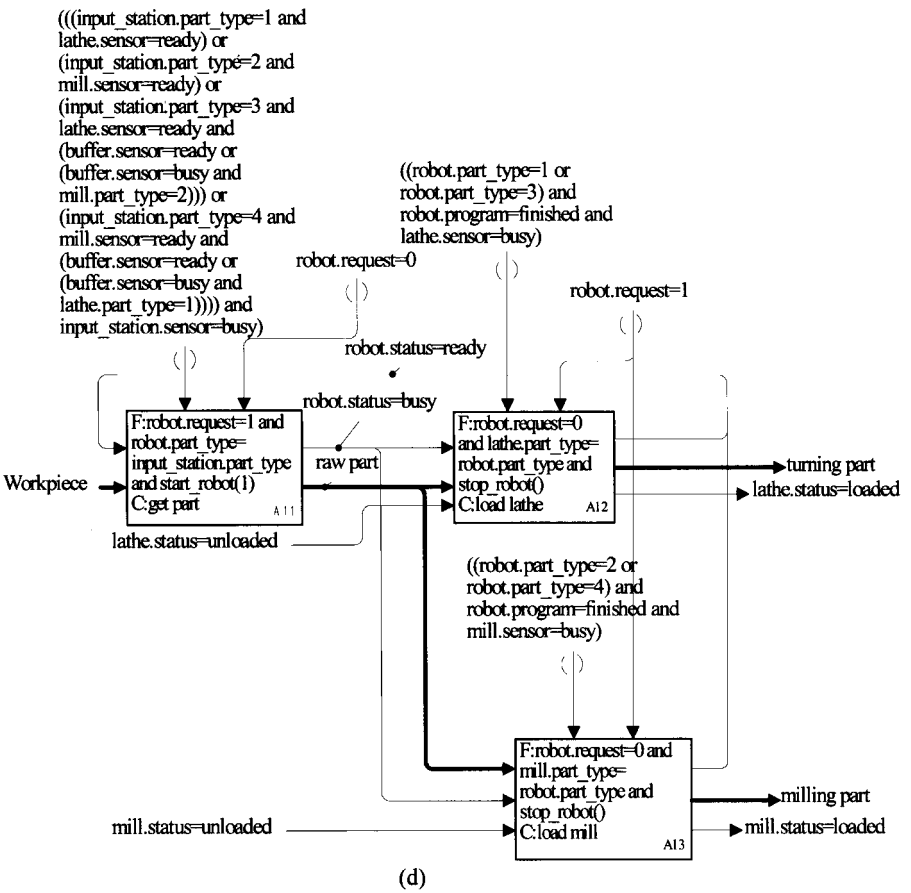


Figure 7. The transformation of control information for Fig. 6 (b): each applying (a) rule 1, (b) rule 2.



(c)



(d)

Figure 7 (*continued*). The transformation of control information for Fig. 6 (b): each applying (c) rule 3, (d) rule 4, and rule 5, respectively.

A13, form the state transition cycles for the robot by replacing the original resource utilization cycles. These can also be applied to the state-transition cycles for the lathe and mill, respectively. Similar diagrams can be found in the subpages of A2, A5 and A6. The states for the robot are ready and busy states. The states for the lathe or mill are unloaded, loaded, started, and stopped states.

- *Rule* 2. The control for the sharing of resources is based on flags. Each resource utilization cycle of the same resource with all different activity boxes is assigned a unique requesting number for the same flag.

Thus, a resource is employed by a resource utilization cycle, and it can be released only after the flag is reset to zero. In the case of a buffer, however, this scheme is only applied for the ordered buffers where each location of the buffer is viewed as a resource. For the simple buffer, different flags are required for each resource utilization cycle, where each flag records the number of capacities of the buffer requested. For the generalized buffer, different flags are also required for each resource utilization cycle; however, they are simultaneously reset to zero when the capacities for the location of the buffer are exhausted. For instance, when applying rule 2, Fig. 7 (a) evolves into Fig. 7 (b). The state-transition cycles A11 and A12 or A11 and A13 employ the flag 'robot.request' with the number 1, since both cycles use the same A11 activity box, the requesting number for the flag is the same. The flag 'robot.request' is shared with the other subpages of A2, A5 and A6 where the numbers 2, 3 and 4 are assigned, respectively. When the flag is 0, the A11 can be enabled and the flat is set to 1. When the flag is 1, the A12 or A13 can be enabled and the flag is set to 0. Moreover, A12 and A13 are mutually exclusive during operation.

- *Rule* 3. The redundancy flags for the resource sharing control can be removed when the corresponding utilization cycles are mutually exclusive.

This is owing to the fact that the mutually exclusive cycles never occur simultaneously, making the control flags no longer necessary. For instance, in Fig. 7 (c), A12 and A13 are mutually exclusive, the flags 'lathe.request' and 'mill.request' are removed.

- *Rule* 4. The manufacturing policies can be listed in a table which describes the conditions in which each activity box is to be enabled.

These conditions include the acknowledgment of sensors, the interlock of machines or robots, the type of parts and the status of the entire systems. According to this table, these conditions are assigned to each corresponding activity box to handle the deadlock problems. For instance, Table 1 lists the deadlock avoidance policies for the A1 page. Figure 7 (d) denotes the conditions attached to A11, A12, and A13. Assume that each device has a sensor to indicate the existence of a part on it. The activity boxes can be enabled when the deadlock avoidance policies hold. The policies can be confirmed by the corresponding C′PN or simply by simulation.

- *Rule* 5. The animation procedures can be initiated at each activity box if it is desired to create an animation corresponding to the actual manufacturing system.

Activity box	Part type	The deadlock avoidance policy to enable the activity box
A11	1	lathe.sensor= ready and input_station.sensor= busy
	2	mill-sensor= ready and input_station.sensor= busy
	3	lathe.sensor= ready and (buffer.sensor= ready or (buffer.sensor= busy and mill-part_type= 2)) and input_station.sensor= busy
	4	mill-sensor= ready and (buffer.sensor= ready or (buffer.sensor= busy and lathe.part_type= 1) and input_station.sensor= busy
A12	1	lathe.sensor= busy and robot.program= finished
	3	
A13	2	mill.sensor= busy and robot.program= finished
	4	

Table 1. The deadlock avoidance policies for the A1 page.

For instance, the animation procedure of start movement for the robot with route 1 is described in A11 (Fig. 7 (d)), the start_robot(1). The procedure of stop movement for the robot is described in A12 or A13, the stop_robot().

By the five rules above, the complete control flows for the system can be created according to the functional specifications. This finding suggests the possibility of systematically constructing the control flows of a manufacturing system controller.

4.2. *The transformation for the production rules*

The FMI diagrams created from the above transformation are important documents for maintaining the underlying system. These diagrams can be directly transformed to the production rules as a knowledge base for an expert system. Only one rule performs such a transformation, i.e.

- Each activity box forms a rule, where its external and controllable expressions construct the rule's antecedent, its primitive computations or functions and the controlled expressions construct the rule's conclusions.

By this rule, Figure 7 (d) can be transformed to three production rules as described in the following.

(1) **if** the status of robot is ready and the request of robot= 0 and (((the part_type of input_station= 1 and the sensor of lathe is ready) or (the part_type of input_station= 2 and the sensor of mill is ready) or (the part_type of input_station= 3 and the sensor of lathe is ready and (the sensor of buffer is ready or (the sensor of buffer is busy and the part_type of mill= 2))) or (the part_type of input_station= 4 and the sensor of mill is ready and (the sensor of buffer is ready or (the sensor of buffer is busy and the part_type of lathe= 1)))) and the sensor of input_station is busy)
then conclude that the part_type of robot= the part_type of input_station and conclude that the request of robot= 1 and start start_robot(1) and conclude that the status of robot is busy.

(2) **if** the status of robot is busy and the status of lathe is unloaded and ((the part_type of robot= 1 or the part_type of robot= 3) and the sensor of lathe is busy and the program of robot is finished) and the request of robot= 1
then conclude that the part_type of lathe= the part_type of robot and start stop_robot() and conclude that the status of robot is ready and conclude that the status of lathe is loaded.

(3) **if** the status of robot is busy and the status of mill is unloaded and ((the part_type of robot= 2 or the part_type of robot= 4) and the sensor of mill is busy and the program of robot is finished) and the request of robot= 1
then conclude that the part_type of mill= the part_type of robot and start stop_robot() and conclude that the status of robot is ready and conclude that the status of mill is loaded.

The above production rules are coded in G2 (Gensym 1994) syntax. G2 is a real-time expert system for monitoring and control in a manufacturing system. From the FMI diagrams, the forward reasoning scheme of the three production rules can be easily inspected. That is, rule A11 would chain rule A12 or A13; while rule A12 or A13 would chain rule A11. They chain each other in turn.

4.3. *The transformation for the controlled Petri nets*

The FMI diagrams can also be transformed to the C′PNs. The C′PN is used to verify the dynamic properties, e.g. liveness and boundedness, of the designed controller. A C′PN is defined as a six-tupe $\Phi = \{P, T, I, B, D, m_0\}$ (Holloway and Krogh 1990), where P is the finite set of state places, T is the finite set of transitions, $I = (P \times T) \cup (T \times P)$ is a set of directed arcs connecting state places and transitions, B is the finite set of control places, $D = (B \times T)$ is the set of directed arcs associating control places and transitions. $m_0 : P \rightarrow N^p$ is the initial marking of the system, N is the set of positive integers. The set of places which are inputs to a transition $t \in T$ is denoted by ${}^{(p)}t$. The set of places which are outputs to a transition $t \in T$ is denoted by $t^{(p)}$. A transition $t \in T$ is said to be enabled under a marking m(p) if for all $p \in {}^{(p)}t$, $m(p) \geq 1$.

Three rules are used to transform an FMI diagram to the corresponding C′PN.

- *Rule* 1. Each activity box in an FMI diagram is a transition in the C′PN.
- *Rule* 2. Each simple arrow in an FMI diagram is replaced by a state place with input and output arcs, or by a control place with an output arc in the C′PN. A simple arrow in an FMI diagram is an arrow such that no branch or join occurs in this arrow.
- *Rule* 3. Each arrow with a branch or joint in an FMI diagram is replaced by a common state place with an input arc and multiple output arcs or multiple input arcs and an output arc.

For instance, by the rules described above, the C′PNs for the FMC example can be illustrated in Fig. 8, where (a) ~ (f) are the C′PNs corresponding to the A1 ~ A6 FMI diagrams, respectively. The shaded places are the control places where enabling conditions are described as their labels. The places with the same number are indicated as the same places. There are 18 transitions, 12 state places, and 19 control places for the FMC example. The markings of the 19 control places depend on the markings of the 12 state places and the four types of parts. Thus, there are $C_4^{12} * 4 = 1980$ possible markings to verify since the multiplicity for the robot,

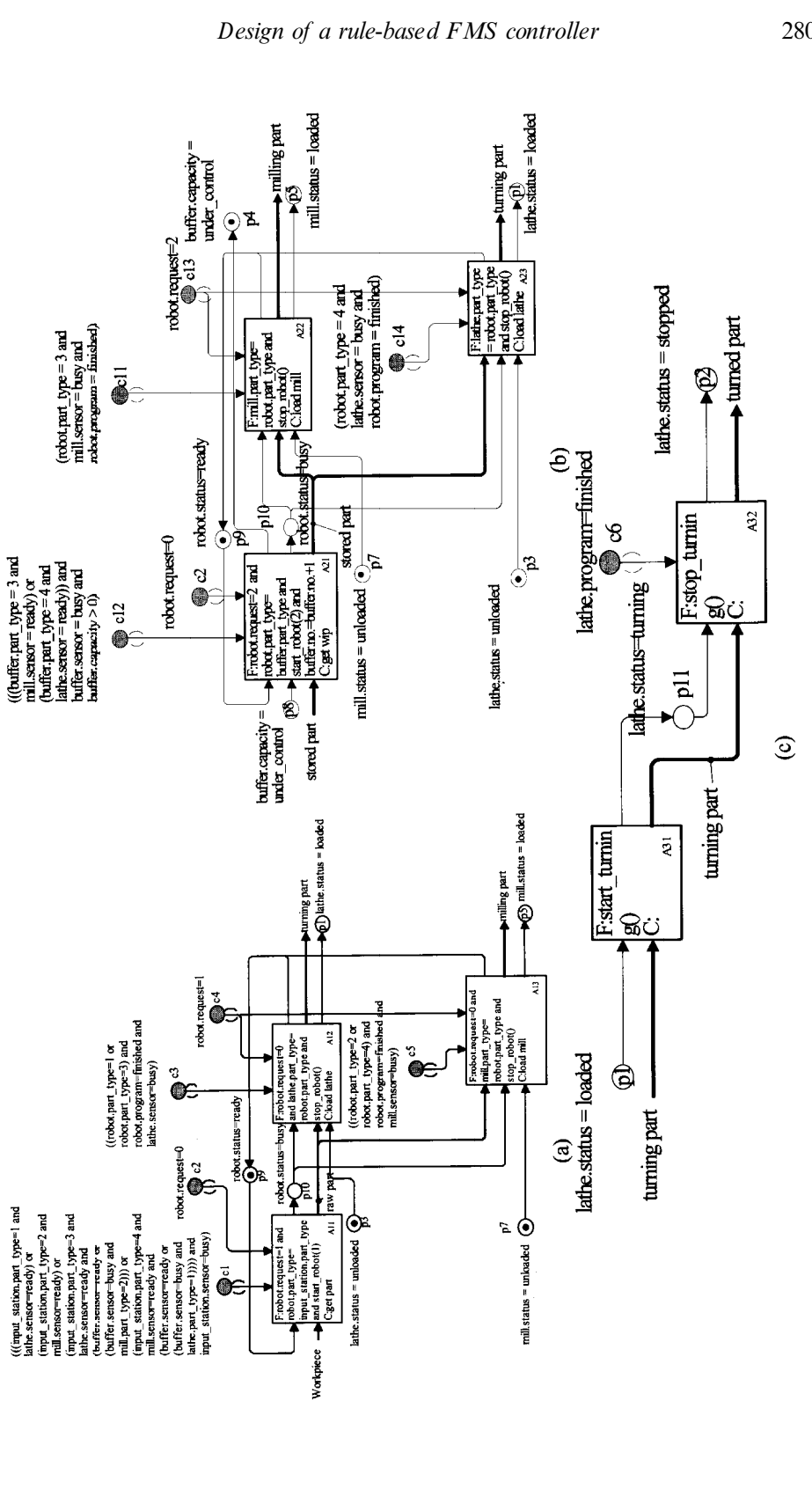


Figure 8. The transformed C-PN for the FMC example, where (a) for A1, (b) for A2, (c) for A3.

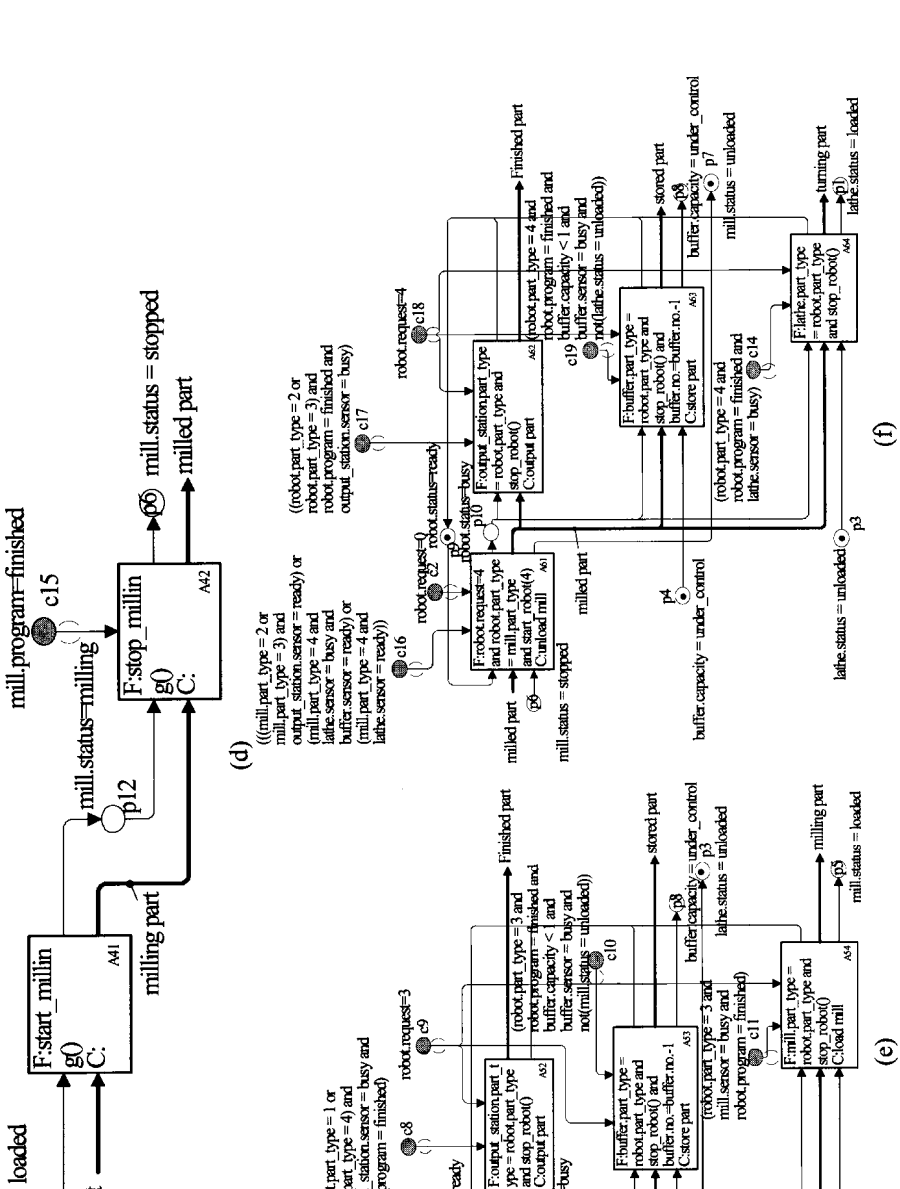


Figure 8 (*continued*). The transformed C-PN for the FMC example, where (d) for A4, (e) for A5, and (f) for A6 pages.

lathe, mill and buffer are all ones. An exhaustive search for these markings is nearly impossible. The P-invariants property of the C′PN can be employed to inspect the dynamic properties of the underlying systems. Four P-invariants are available, i.e.,

$$\begin{aligned} &p1 + p2 + p3 + p11 = 1, && \text{(for the lathe)}\\ &p5 + p6 + p7 + p12 = 1, && \text{(for the mill)}\\ &p4 + p8 = 1, && \text{(for the buffer)}\\ &p9 + p10 = 1. && \text{(for the robot)} \end{aligned}$$

Proof of safeness: The possible tokens for the 12 state places are all ones, this can be inspected by the four P-invariants above. The possible tokens for the 19 control places are also ones, by definition. Therefore, this C′PN is safe.

Proof of liveness: We examine that at least one transition is enabled in all of the reachable markings. This can be inspected by the following cases.

Case 1: $p1 + p2 + p3 + p11 + p5 + p6 + p7 + p12 + p4 + p8 = 3$. Assume that $p9 = 1$. The enabled transition is A11 if no part is in the system, or is A21 if a part is in the buffer and the lathe or mill is idle, or is A51 if a part is finished in the lathe and the output station, mill or buffer is idle, or is A61 if a part is finished in the mill and the output station, lathe or buffer is idle. Assume that $p10 = 1$. The enabled transition is A12 (or A23, A64) if a part is loaded on the lathe, or is A13 (or A22, A54) if a part is loaded on the mill, or is A52 (or A62) if a part is loaded on the output station, or is A53 (or A63) if a part is loaded on the buffer.

Case 2: $p1 + p2 + p3 + p11 + p5 + p6 + p7 + p12 + p9 + p10 = 3$. Assume that $p4 = 1$. The enabled transition is A53 (or A63) if a part is loaded on the buffer. Assume that $p8 = 1$. The enabled transition is A21 if a part is in the buffer and the lathe or mill is idle.

Case 3: $p1 + p2 + p3 + p11 + p4 + p8 + p9 + p10 = 3$. Assume that $p5 = 1$. A41 is enabled. Assume that $p12 = 1$. A42 is enabled if the milling program is finished. Assume that $p6 = 1$. A61 is enabled if a part is finished in the mill and the output station, lathe or buffer is idle. Assume that $p7 = 1$. A13 (or A22, A54) is enabled if a part is loaded on the mill.

Case 4: $p5 + p6 + p7 + p12 + p4 + p8 + p9 + p10 = 3$. Assume that $p1 = 1$. A31 is enabled. Assume that $p11 = 1$. A32 is enabled if the turning program is finished. Assume that $p2 = 1$. A51 is enabled if a part is finished in the lathe and the output station, mill or buffer is idle. Assume that $p3 = 1$. A12 (or A23, A64) is enabled if a part is loaded on the lathe.

The controller for the FMC cannot be deadlocked.

Verifying the FMS controller design can be costly if the transformed C′PN is too large. Moreover, designing the deadlock avoidance policy would be too complex for the C′PN to handle. An efficient way of verification is by simulation.

5. The testing guidelines

Simulation is a testing process. When the analytical model of a system is infeasible for verification, using a simulation model to approximate the verification results is a promising alternative. A simulation model for a controller is a model that, by providing the simulated input data and system reacting signals, the underlying

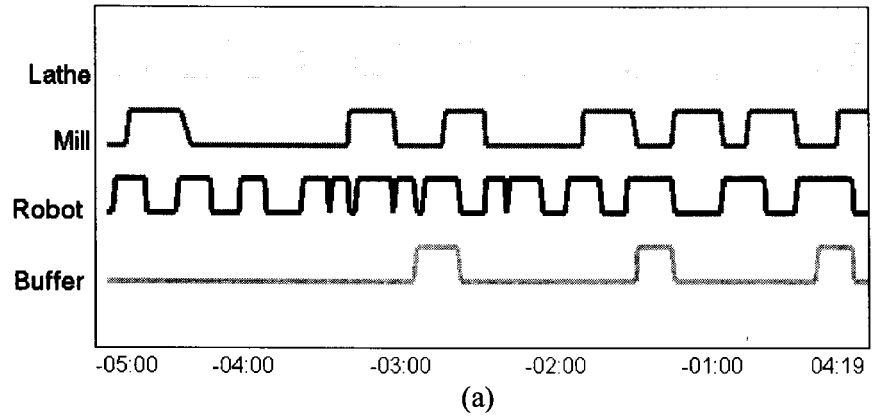
system's behaviour can be inspected or predicted. Goodenough and Gerhard (1975) presented a method for test data selection. They emphasized the importance of the program testing even though the theoretical proof for this program is available. Normally, good data selection would enhance the efficiency of program testing. However, this may be a non-termination process to pursue. Five guidelines are available when performing such a simulation test.

(1) Generating the test cases from the system specification. Meeting the system specification is the priority concern. Therefore, the testing cases must be derived from the system specification. For instance, 64 product mixes are available to test the FMC controller since a maximum of three products can be simultaneously loaded in the system.
(2) Using animation to visualize the simulation results if possible. The system's slightest abnormal behaviour can often be detected by visual inspection. Animation is also an effective approach to illustrate the complex simulation results.
(3) Reducing the time scale to speed up the simulation process. A proportional reduction of time scale is normally valid for the simulation process. This can accelerate a time consuming process and improve the simulation performance.
(4) Constructing the simulation calendar if the simulation process is extremely large. Deriving a systematically constructed calendar for a large simulation process is crucial. The completeness of a simulation process can be guaranteed if all these schedules in the calendar are followed.
(5) Comparing the simulation performance if alternative approaches are available. Different approaches can be compared in the simulation process, thereby providing an effective approach to obtain the optimum design of a manufacturing system.

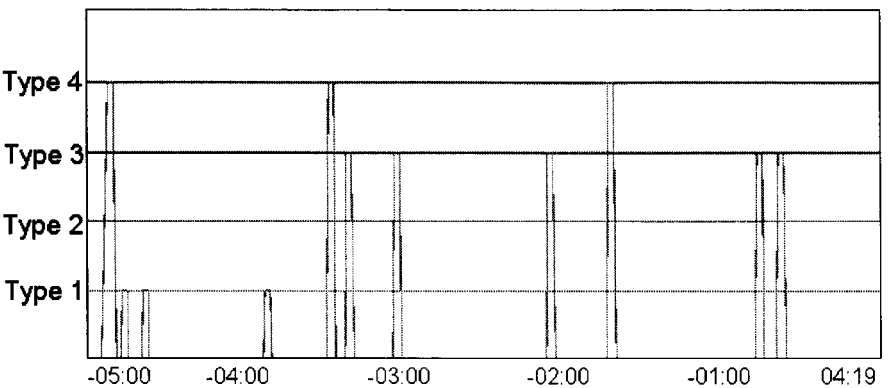
For instance, the testing data based on the specification of the FMC example can be generated as 64 (4^3), cases of a part mix. Thus, an optimal (shortest) schedule to test the FMC controller can be obtained as a string:

1424222132111341144212433241311224441412313344312143423432232333 14

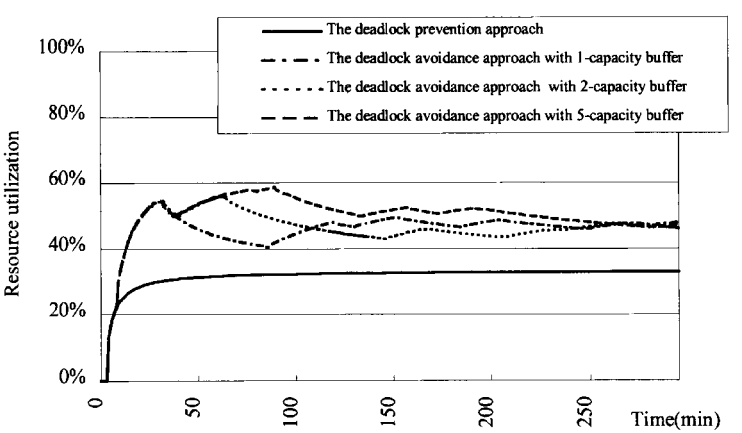
where each number represents the type of parts. Only 66 data entries are necessary to test the entire 64 cases. The worst schedule needs 192 data entries, the concatenation of 64 cases each having three data entries. The FMC example was implemented on a Pentium PC running Windows NT. The simulation process was created by a G2 expert system (Gensym 1994). Figure 9 illustrates the resources timing chart, the loading sequence of the jobs and the performance comparison of alternative approaches. Four alternative designs were compared: (1) a deadlock prevention approach where only one part is allowed to enter the system in a fixed period; (2) a deadlock avoidance approach with a one-capacity buffer; (3) a deadlock avoidance approach with a two-capacity buffer; (4) a deadlock avoidance approach with a five-capacity buffer. Those results indicate that the performance improves significantly when the deadlock avoidance policy is employed instead of using the deadlock prevention policy. However, the number of buffer capacity only slightly affects the performance.



(a)



(b)



(c)

Figure 9. The statistics for the FMC simulation: (a) the resources timing chart, (b) the loading sequence of the jobs, (c) the performance comparison of alternative approaches.

6. The FMS example

This section provides a complete FMS example. Zhou and DiCesare (1993) introduced this FMS, as developed at Rensselaer Polytechnic Institute, USA. The following demonstrates that the method proposed here is feasible for such an FMS.

The system takes two types of raw stock, machines them into desired shapes, and then assembles these two finished parts into a product. Assume that two product types are to be manufactured. Each has two different parts (block and peg). The type of parts are numbered one to four. Figure 10 illustrates the FMS layout. The major components of the system are one CNC mill and drill machine, one CNC lathe, a Microbot robot to load and unload the materials between the lathe and conveyor 3 (C3), and between the mill and C2, and AS/RS with 19 usable pallet-storage bins for buffering the raw materials and intermediate parts, four two-way conveyors with sensors (sensor1 and sensor2) at each end, a Gantry robot for transferring the materials between the four conveyors, and a Scorbot robot to assemble the parts. Therefore, the main routes for the block and peg are

$$\mathrm{B} \rightarrow \mathrm{C4} \xrightarrow{\rightarrow \mathrm{C1}(\rightarrow \mathrm{AS/RS} \rightarrow \mathrm{C1}) \rightarrow} \mathrm{C2} \rightarrow \mathrm{M} \rightarrow \mathrm{C2} \xrightarrow{\rightarrow \mathrm{C1}(\rightarrow \mathrm{AS/RS} \rightarrow \mathrm{C1}) \rightarrow} \mathrm{C4} \rightarrow \mathrm{A} \rightarrow \mathrm{F}$$

and

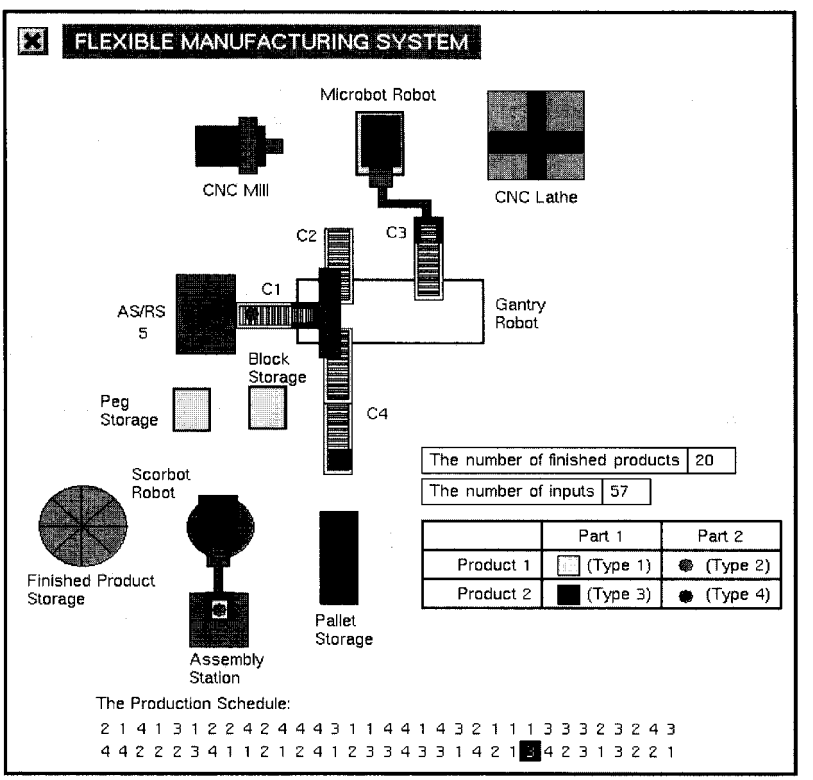


Figure 10. The FMS layout.

$$\mathrm{P} \rightarrow \mathrm{C4} \xrightarrow{\rightarrow \mathrm{C1}(\rightarrow \mathrm{AS/RS} \rightarrow \mathrm{C1}) \rightarrow} \mathrm{C3} \rightarrow \mathrm{L} \rightarrow \mathrm{C3} \xrightarrow{\rightarrow \mathrm{C1}(\rightarrow \mathrm{AS/RS} \rightarrow \mathrm{C1}) \rightarrow} \mathrm{C4} \rightarrow \mathrm{A} \rightarrow \mathrm{F}$$

where B is the block storage, P is the peg storage, Cn is the conveyor, M is the CNC mill, L is the CNC lathe, A is the assembly station, and F is the final product carousel.

The material flow in the FMS is stated as in the following (Zhou and DiCesare 1993).

(1) The Scorbot robot moves an empty pallet from the gravity fed storage to C4.
(2) The Scorbot robot takes a raw block from the block storage and places it in the empty pallet on C4.
(3) C4 moves the loaded pallet to the Gantry robot.
(4) the Gantry robot moves the pallet from C4 to C2 (C3).
(5) C2 (C3) moves the pallet to the CNC machine.
(6) The Microbot robot takes the raw block from the pallet and loads it into the CNC machine.
(7) The CNC machine fixes the raw block and machines the part.
(8) The Microbot robot unloads the finished part from the CNC machine to the pallet on C2 (C3).
(9) C2 (C3) moves the pallet to the Gantry robot.
(10) The Gantry robot moves the pallet from C2 (C3) to C4.
(11) C4 moves the pallet with the finished part to the Scorbot robot.
(12) The Scorbot robot takes the finished part and places it in the assembly cell. If two relative parts are present, it assembles the final product.
(13) The Scorbot robot moves the finished product to the output carousel.
(14) The Scorbot robot moves the empty pallet from C4 to the pallet storage.

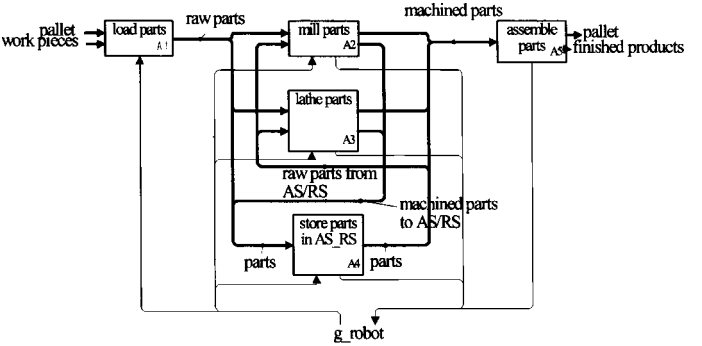
The system can process concurrent works simultaneously.

The functional specification for the FMS is next constructed. Figure 11 (a) illustrates the MI diagram for the functional specification of the FMS. The material flow's flexibility is easily inspected. When the CNC machines are busy, the raw parts can be stored in the AS/RS. While the assembly station is busy, the machined parts can also be stored in the AS/RS for buffering. When the AS/RS is full, the C1 can hold an additional one part for buffering. Figure 11 (b) presents the FMI diagram for the FMS. The control information is created by the transformation rules discussed in the previous section. Figure 11 (c) to (g) highlights the details of A1 to A5 pages. The deadlock avoidance policies are described as in the external expressions of each activity box. Table 2 only illustratively lists the transformed rules for the A1 page of the FMS controller.

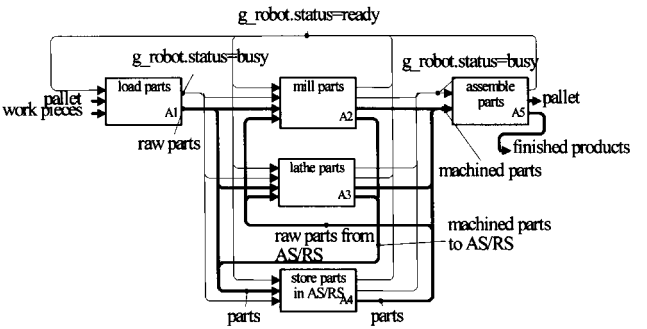
To test the controller, total test cases are $4^{22}/4^{19} = 64$, since the AS_RS is a simple buffer. An optimal (shortest) test schedule can be available as indicated in the following,

2141312242444311441432111333232434422234112124123343314213423132 21,

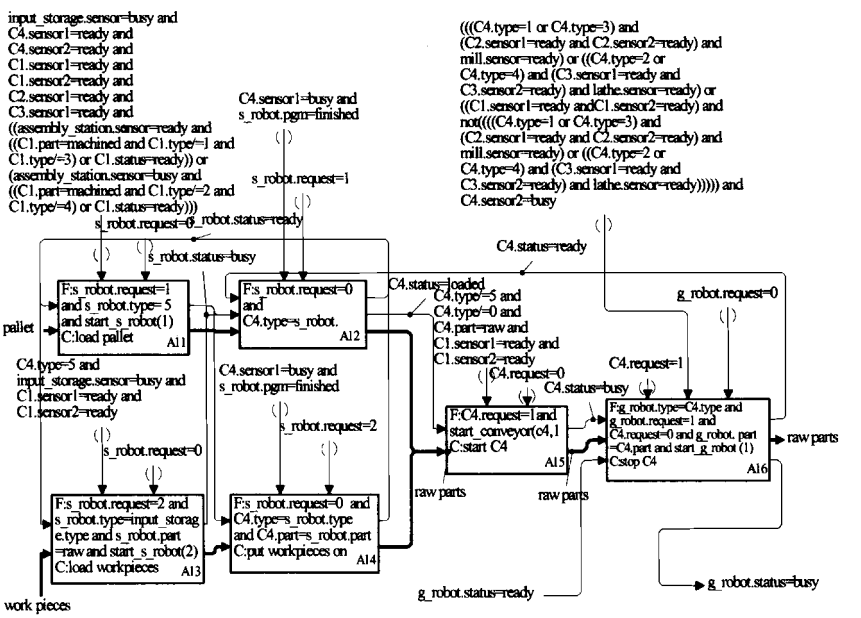
where each number denotes the part type of each product. Only 66 cases are required to test the controller. Figure 12 illustrates the inventories of input, output and AS/RS, and the machine utilization for the Scorbot, Gantry and Microbot robot of this



(a)

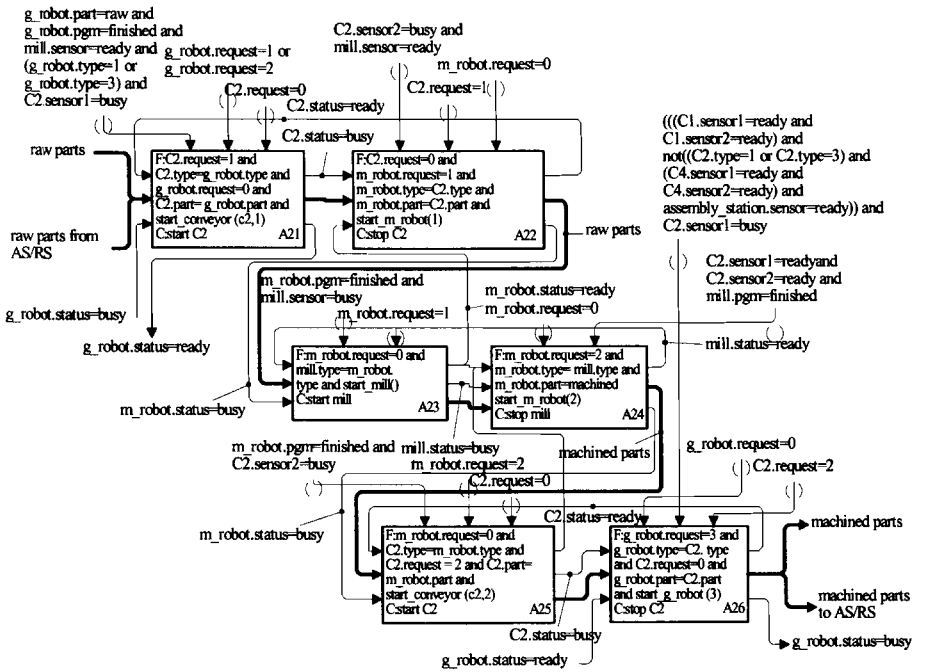


(b)

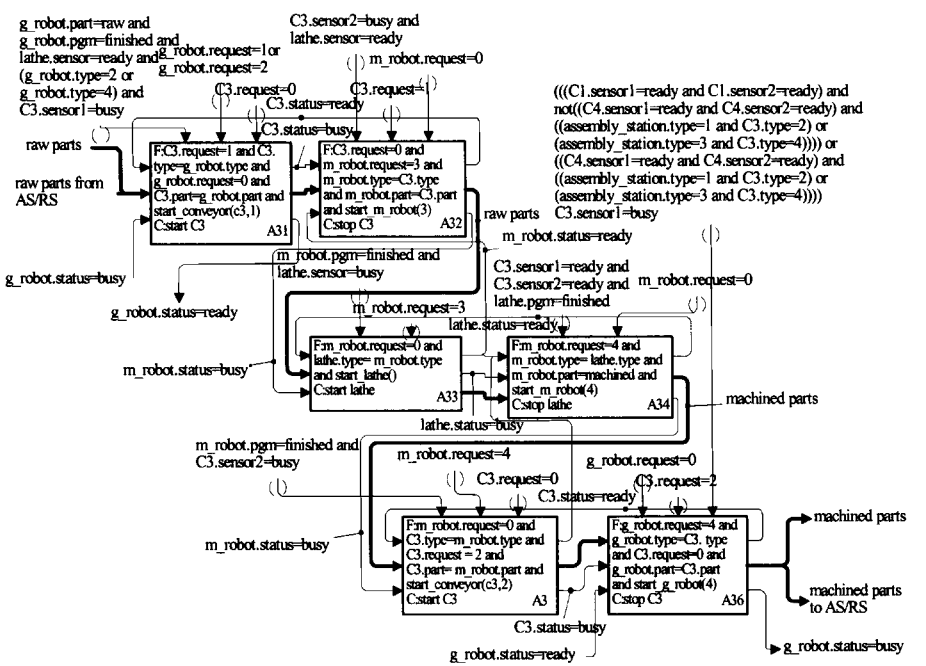


(c)

Figure 11. The MI diagram for the FMS, (a) the functional M1 diagram, (b) the augmented MI diagrams, (c), (d), (e), (f), and (g) are the details of the A1 to A5 pages.

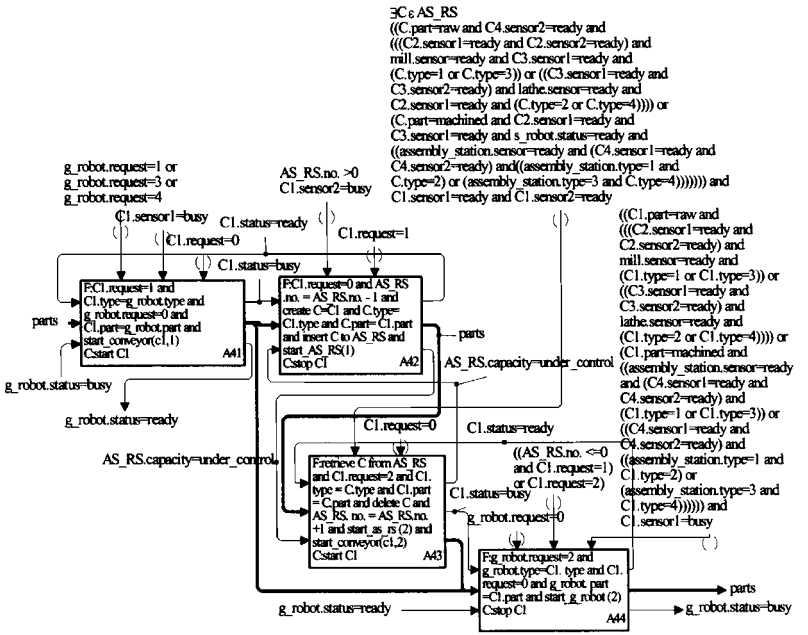


(d)

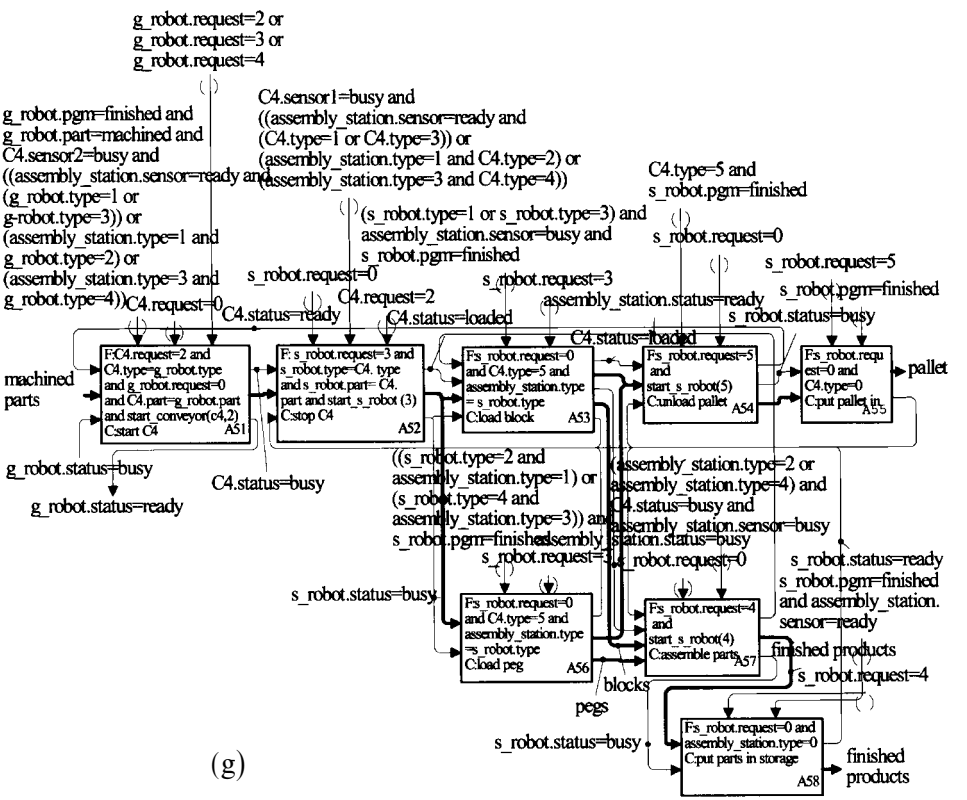


(e)

Figure 11 (*continued*).



(f)



(g)

Figure 11 (*continued*).

Activity box	Rules
A11	if the sensor of input_storage is busy and the sensor1 of c4 is ready and the sensor2 of c4 is ready and the sensor1 of c1 is ready and the sensor2 of c1 is ready and the sensor1 of c2 is ready and the sensor1 of c3 is ready and ((the sensor of assembly_station is ready and ((the part of c1 is machined and the type of c1/= 1 and the type of c1/= 3) or the status of c1 is ready)) or (the sensor of assembly_station is busy and ((the part c1 is machined and the type of c1/= 2 and the type of c1/= 4) or the status of c1 is ready))) and the request of s_robot= 0 and the status of s_robot is ready then conclude that the request of s_robot= 1 and conclude that the type of s_robot= 5 and start start_s_robot(1) and conclude that the status of s_robot is busy
A12	if the sensor1 of c4 is busy and the pgm of s_robot is finished and the request of s_robot= 1 and the status of s_robot is busy and the status of c4 is ready then conclude that the request of s_robot= 0 and conclude that the type of c4= the type of s_robot and conclude that the status of s_robot is ready and conclude that the status of c4 is loaded
A13	if the type of c4= 5 and the sensor of input_storage is busy and the sensor1 of c1 is ready and the sensor2 of c1 is ready and the request of s_robot= 0 and the status of s_robot is ready then concude that the request of s_robot= 2 and conclude that the type of s_robot= the type of input_storage and conclude that the part of s_robot is raw and start start_s_robot(2) and conclude that the status of s_robot is busy.
A14	if the sensor1 of c4 is busy and the pgm of s_robot is finished and the request of s_robot= 2 and the status of s_robot is busy then conclude that the request of s_robot= 0 and conclude that the type of c4= the type of s_robot and conclude that the part of c4= the part of s_robot and conclude that the status of s_robot is ready
A15	if the type of c4/= 5 and the type of c4/= 0 an the part of c4 is raw and the sensor1 of c4 is busy and the sensor2 of c4 is ready and the request of c4= 0 and the status of c4 is loaded then conclude that the request of c4= 1 and conclude that the status of c4 is busy and start start_conveyor(c4, 1)
A16	if (((the type of c4= 1 or the type of c4= 3) and (the sensor1 of c2 is ready and the sensor2 of c2 is ready) and the sensor of mill is ready) or((the type of c4= 2 or the type of c4= 4) and (the sensor1 of c3 is ready and the sensor2 of c3 is ready) and the sensor of lathe is ready) or ((the sensor1 of c1 is ready and the sensor2 of c1 is ready) and not ((((the type of c4= 1 or the type of c4= 3) and (the sensor1 of c2 is ready and the sensor2 of c2 is ready) and the sensor of mill is ready) or ((the type of c4= 2 or the type of c4= 4) and (the sensor1 of c3 is ready and the sensor2 of c3 is ready) and the sensor of lathe is ready))))) and the sensor2 of c4 is busy and the request of c4= 1 and the status of c4 is busy and the request of g_robot= 0 and the status of g_robot is ready then conclude that the type of g_robot= the type of c4 and conclude that the request of g_robot= 1 and conclude that the request of c4= 0 and conclude that the part of g_robot= the part of c4 and conclude that the status of c4 is ready and conclude that the status of g_robot is busy and start start_g_robot(1).

Table 2. The transformed rules for the A1 page of the FMS controller.

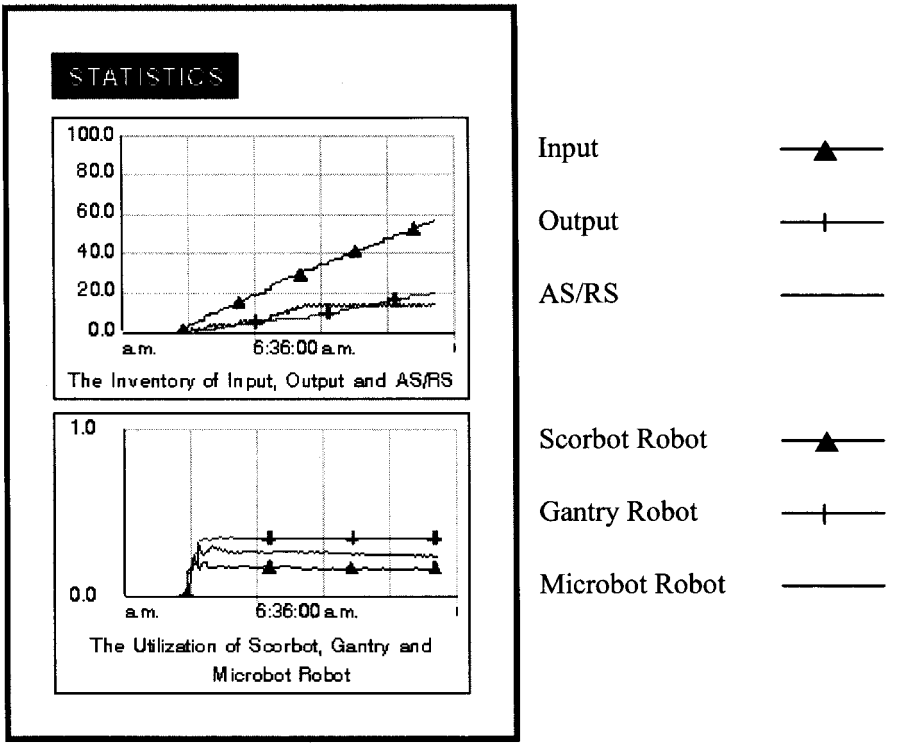


Figure 12. The inventory and machine utilization chart.

simulation testing. The steady state for the FMS results in an inventory of AS/RS as 14 parts. The Gantry robot gains a maximal machine utilization. Those statistics reveal that the FMS controller is very stable.

7. Conclusion

In summary, this work has presented a systematical method of designing a rule-based FMS controller. The functional specification of the FMS controller is firstly synthesized by the individual MI diagram primitives. The control flows can be created by transforming the resource utilization cycles. The manufacturing policies are also specified and attached. The FMI diagrams can be created by the transformation and attachment. The FMI diagrams can be transformed to the production rules which can be executed on an expert system. The rule-based FMS controller is therefore constructed.

To verify this controller, two approaches are proposed. For a small system, the C′PN model is suggested. This model can be created by transforming the FMI diagrams. Properties of the C′PN model, e.g., the safeness and liveness, can then be inspected by the P-invariants method, or the reachability tree. For a large system, however, simulation testing is recommended instead since the analytical C′PN model may not be feasible. Some general guidelines for the simulation process are proposed. Two examples are employed to illustrate the method provided here. The first example is the common FMC with one robot, two machines and one buffer. The other is the famous FMS example introduced by Zhou and DiCesare (1993), where

one AS/RS, two machines, three robots, four conveyors and an assembly cell are employed. Both examples demonstrate the usefulness and rapid prototyping capability of the proposed method.

The fault diagnosis problem for the approach may be an area of future research. Results in this study can hopefully contribute toward the design of an intelligent FMS controller.

References

Bauer, A., Bowden, R., Browne, J., Duggan, J., and Lyons, G., 1991, *Shop Floor Control Systems – From Design to Implementation* (London: Chapman & Hall).

Cecil, J. A., Srihari, K., and Emerson, C. R., 1992, A review of Petri-net applications in manufacturing. *International Journal of Advanced Manufacturing Technology*, **7**, 168–177.

Coffman, Jr, E. G., Elphick, M., and Shoshani, A., 1971, System deadlocks. *Computer Surveys*, **3** (2), 67–78.

David, R., and Alla, H., 1992, *Petri Nets & Grafcet – Tools for Modeling Discrete Event Systems* (Englewood Cliffs, NJ: Prentice Hall).

Dupond-Gatelmand, C., 1982, A survey of flexible manufacturing systems. *Journal of Manufacturing Systems*, **1** (1), 1–16.

Franks, I., Loftus, M., and Wood, N. T. A., 1990, Discrete cell control. *International Journal of Production Research*, **28** (9), 1623–1633.

Gensym, 1994, *G2 Reference Manual, Version 4.0* (USA: Gensym Corporation).

Goodenough, J. B., and Gerhart, S. L., 1975, Towards a theory of test data selection. *IEEE Transactions on Software Engineering*, **SE-1** (2), 156–173.

Holloway, L. E., and Krough, B. H., 1990 Synthesis of feedback control logic for a class of controlled Petri nets. *IEEE Transactions on Automatic Control*, **35** (5), 514–523.

Hong, H. M., 1993, IDEF/CPN/G2 approach to the implementation of real-time shop floor control system. MS thesis, National Chiao Tung University, Taiwan.

Jafari, M. A., and Boucher, T. O., 1994, A rule-based system for generating a ladder logic control program from a high-level systems model. *Journal of Intelligent Manufacturing*, **5**, 103–120.

Jones, A. T., and McLean, C. R., 1986, A proposed hierarchical control model for automated manufacturing systems. *Journal of Manufacturing Systems*, **5** (1), 15–25.

Kusiak, A., 1990, *Intelligent Manufacturing Systems* (Englewood Cliffs, NJ: Prentice Hall).

Liang, G. R., and Hong, H. M., 1994, Hierarchy transformation method to manufacturing system specification, design, verification, and implementation. *Computer-Integrated Manufacturing Systems*, **7** (3), 191–205.

Meyer, W., 1990, *Expert Systems In Factory Management – Knowledge-Based CIM* (London: Ellis Horwood).

Murata, T., Komoda, N., Matsumoto, K., and Haruna, K., 1986, A Petri net-based controller for flexible and maintainable sequence control and its applications in factory automation. *IEEE Transactions on Industrial Electronics*, **IE-33** (1), 1–8.

Sauve, B., and Collinot, A., 1987, An expert system for scheduling in a flexible manufacturing system. *Robotics and Computer-Integrated Manufacturing*, **3** (2), 229–233.

Teng, S., and Black, J. T., 1989, An expert system for manufacturing cell control. *Computers in Industrial Engineering*, **17** (1–4), 18–23.

Valette, R., 1987, Nets in production systems. In *Advances in Petri Nets: Petri Nets Applications and Relationships to Other Models of Concurrency, Lecture Notes in Computer Science*, **255** (New York: Springer), pp. 191–217.

Wu, S. Y. D., and Wysk, R. A., 1988, Multi-pass expert control system – a control/scheduling structure for flexible manufacturing cells. *Journal of Manufacturing Systems*, **7** (2), 107–120.

Xiang, D., and O'Brien, C., 1995, Cell control research – current status and development trends. *International Journal of Production Research*, **33** (8), 2325–2352.

ZHOU, M., and DICESARE, F., 1990, Modeling buffers in automated manufacturing systems using Petri nets. *Proceedings of Rensselaer's Second International Conference on Computer Integrated Manufacturing*, pp. 265–272.

ZHOU, M., DICESARE, F., and DESROCHERS, A. A., 1992, A hybrid methodology for synthesis of Petri net models for manufacturing systems. *IEEE Transactions on Robotics and Automation*, **8** (3), 350–361.

ZHOU, M., and DICESARE, F., 1993, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems* (Boston, MA: Kluwer Academic Publishers).