SUPERVISORY POLICIES FOR FLEXIBLE MANUFACTURING SYSTEMS

Călin Ciufudean Constantin Filote Adrian Graur
"Ştefan cel Mare" University of Suceava, Faculty of Electrical Engineering
calin@eed.usv.ro
filote@eed.usv.ro
adriang@eed.usv.ro

Abstract – This paper focuses on a method for synthesizing deadlock avoidance controllers in flexible manufacturing systems. The modelling process of a supervisory policy that enforces liveness is realized with Petri nets. Petri nets are a popular modelling paradigm for a wide class of discrete event systems. Our supervisory policy is based on the next two steps in modelling process:

a) A bottom-up approach for the synthesis of a controlled Petri net model for the production flow;
b) A liveness condition for the above mentioned model, under the circumstances of underlying the potential peril path (PPP) in the Petri net model by respecting the concept of minimal resource requirements.

While the deadlock avoidance controller ensures the liveness of the global Petri net model (GPN) of the FMS, an algorithm for estimating the system throughput is introduced. The utility of our approach in alleviating the computational burden of policy synthesis is illustrated via example.

1. Introduction

A flexible manufacturing system (FMS) consists of a number of systems, usually connected in a computer controlledr configuration of various kinds of process actions implemented with material storage facilities, material processing devices, raw material and finite products transportation devices, control units, etc. Various types of jobs are loaded at discrete point of time into the FMS for processing. Each type of job requires a prescribed sequence of technological operations in order to schedule the manufacturing resources. The rational utilisation of limited resources among various competing jobs by operating in an appropriate manner the FMS constitutes the goal of a supervisory policy of a FMS. Many control algorithms in a FMS usually adopt a hierarchical structure due to its complexity: a high level scheduling function to determine the processing sequence among operations of jobs, and a low level real-time control of detailed manufacturing processes [1,2]. Among the many real-time control problems, deadlocks are highly undesirable where a set of jobs are in circular waiting for resources being held by another job in the set while occupying a resource needed by one of the other job in the set [3].

Some deadlock avoidance schemes for controlling a FMS have been proposed lately [4,5]. As the dynamics of a FMS is event-driven, asynchronous and concurrent in nature, many of these schemes adopted Petri net (PN) models as a formalism to describe FMSs and to develop deadlock avoidance mechanisms. Viswanadham et al.[4] considered a generalized stochastic PN (GSPN) model and proposed an on-line, finite-step look-ahead monitoring and deadlock avoidance scheme. By monitoring the system, their controller identifies the current state and examines whether the occurrence of a set of admissible events in the next few steps leads the system to a deadlock state. If so, the controller avoids such an undesirable evolution through controlling controllable events. If not, events for the next step are allowed to take place and the controller repeats the above procedure. Wonham et al.[5] studied the deadlock free supervisory control problem in the context of finite state machine models. They synthesized the supervisory control for discrete event processes by imposing restrictions on the occurrence of controllable events. Two types of deadlock free supervisors were investigated: the total deadlock free supervisor and Σ_0 deadlock free supervisor. The latter prevents the process from reaching a state under which events in a set Σ_0 can no longer occur, while the former has Σ_0 as the set of all admissible events.

In operating a FMS, it is also desirable to make the FMS capable of processing all types of jobs repetitively in addition to keep the FMS live in terms of PN modelling formalism. If a FMS is live, then it is deadlock free but not vice-versa [3]. In this paper, the deadlock avoidance controller (DAC) is driven by a class of PN models which combines the ideas of [1,4] and consists of two ingredients:

a) A bottom-up approach for the synthesis of a controlled Petri net (CPN) model for the production flow;
b) A liveness condition for the above-mentioned model, under the circumstances of underlying the potential peril path (PPP) in the PN model by respecting the concept of minimal resource requirements.

The resultant deadlock avoidance algorithm in conjunction with the mentioned PPP avoidance policy constitutes a DAC as depicted in Fig.1.

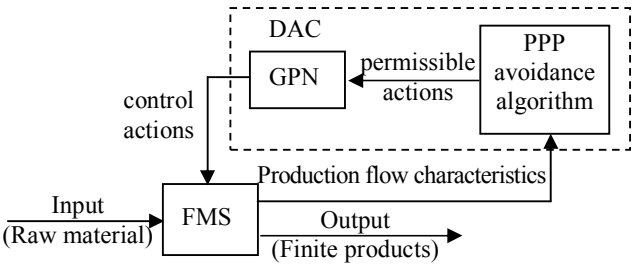

Fig.1 DAC controller for a FMS

The construction of a CPN model starts wits independent PNs to represent the manufacturing processes of individual jobs and the manufacturing activities of individual manufacturing resources. These jobs and resources are merged into a global PN (GPN) according to their common manufacturing activities. The production flow characteristics of the FMS are incorporated into the GPN to form a CPN. A deadlock avoidance controller of the FMS includes a control policy that allows, in accordance with the PPP avoidance algorithm, the occurrence of concurrent production events, by respecting the concept of minimal resource requirements, so that the CPN stays live, and hence deadlock free. As it is well known the deadlock in PNs is related to siphons [6], we use control places to prevent the total marking in the siphons from becoming zero [7]. Finally, note that when the PNs are bounded and the initial marking is fixed, it is possible to transform the problem from the PN framework to finite automata, and so to solve the problem by using finite automata methodologies, such as supervisory control technique. Furthermore, the use of PNs in deadlock prevention may be preferable because deadlock often occurs in systems with concurrencies, which are better modelled by PNs.

2. A Global Petri Net Model

In this section we describe the construction of the GPN given in Fig.1 that consists of resource subnets, job subnets and exogenous control given by an algorithm, in order to describe the PPP avoidance algorithm in the context of FMS.

A. The resource subnets

Let R be the set of resource types in a FMS. We assume that a unit of resource can only be involved in one operation at a time. We model such an activity by a PN, $G = (P,T,F,M_0)$ where P is a finite set of places with cardinality |P| and places represent the state of the resources, T is a finite set of transitions which represent the operations of the respective activity, F is a finite set of transitions arcs, $M_0 : P \rightarrow Z^{|P|}$ is the initial marking of the PN with Z as the set of positive integers. The marking indicates the number of tokens in each place and is a state of the system. The readers may refer to [8] for definitions of the PNs. Since we consider renewable production resources, such as machines, buffers, conveyors, the PN of the k-th activity starts with place $p_k(0) = p(0)$ ($p(0)$ represents the resource idle state), has a transition input arc between $p_k(i-1)$ and $t_k(i)$, $i = 1, \ldots, n$, ($p_k(i-1)$ and $p_k(i)$ represent the state of the resource before and after transition $t_k(i)$ respectively, and n is the number of distinct operations in the k-th activity).

As each transition $t_k(i)$ has one input place and one output place, such a PN is called sure. We define a merging operation, ڤ, as an operation that combines two PNs (PN_1 and PN_2) into a new PN, PN_{new}, by merging each pair of common elements (places, transitions, arcs) between PN_1 and PN_2 into a single element; the remaining distinct parts between PN_1 and PN_2 are kept unchanged and become parts of PN_{new}. We denote $PN_{new} = PN_1$ ڤ PN_2. The number of tokens in a resource subnet corresponds to the capacity of the respective resource; therefore the token flow in the GPN represents the state transition of the resources. We denote the resource subnet of type - r resource as PN_r, where each transition in PN_r maintains exactly one input place and one output place; PN_r therefore is a sure net [7,8].

B. The job subnets

A job subnet is constructed in a similar manner to that of a resource subnet. In constructing a job subnet for one type job, we again use a transition to represent an operation while using a place to represent a job state. There is a source transition for the k-th job $t_k(1)$, that generates processing tokens which model the first operation that releases a type of one job into the production system. We assume that the production process of the job type j consists of a sequence of transitions $t_j(1), t_j(2), \ldots, t_j(k)$ of k activities. The output place for the last transition $t_j(k)$ is a sink place that represents an infinite storage of finished jobs. We denote a job subnet as $PN_j = (P_j, T_j, F_j, m_j)$, where m_j is the number of type j jobs that are being processed. Such a job subnet usually is an acyclic marked graph, since the production process is acyclic, and each place has one input and one output transition; PN therefore is a sure net. Tokens in a job subnet certain that operation holding resources are in process.

C. Global Petri net

By merging resource and job subnets we construct the Global Petri net (GPN) that models the interactions among operations, resources and jobs in the FMS by:

$$GPN = PN_r \text{ ڤ } PN_j \qquad (1)$$

In a FMS there are control points that can be applied to jobs and their operations in order to control the production flow. In terms of PN we add to each transition of the GPN that corresponds to a controlled operation, a control place p_c and a transition input arc between p_c and the transition. The control places model the control conditions in a FMS, respectively incorporate exogenous conditions for enabling the associated transition. A controlled transition (e.g., a transition with an input arc from a control place) may be fired as many times as the number of tokens in the control place. A control policy [5,6] is a mapping that generates a sequence of control actions for the GPN based on its initial marking M_0, which evolves in a set of admissible markings. We exemplify the construction of a GPN (by merging the resource subnets with job subnets): Consider the system depicted in Fig.2., which consists of two machining tools (M_1 and M_2), two robot arms, and two conveyors.

Each machining tool is serviced by a dedicated robot arm, which performs load and unload tasks. One conveyor is used to transport work pieces, a maximum of two at a time. The other conveyor is used to transport empty pallets. There is one pallet available in the system. Each work piece is machined on M_1 or M_2, in this order.

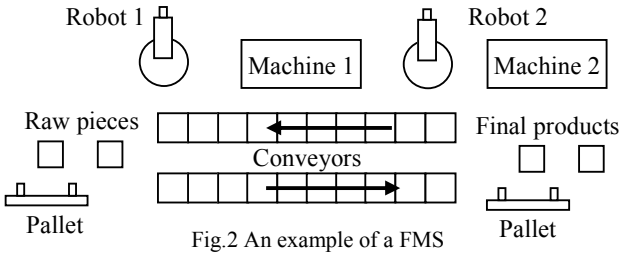

Fig.2 An example of a FMS

Fig.3 shows the two resource subnets which correspond to the jobs of M_1 and M_2. Places p_1, p_2, p_3 model the resources of raw pieces and pallets; the M_1 and M_2 availability, and also final products and empty pallets, respectively. Places p_4 (p_6) and p_5(p_7) model the robot 1 and robot 2 availability (when marked), respectively.

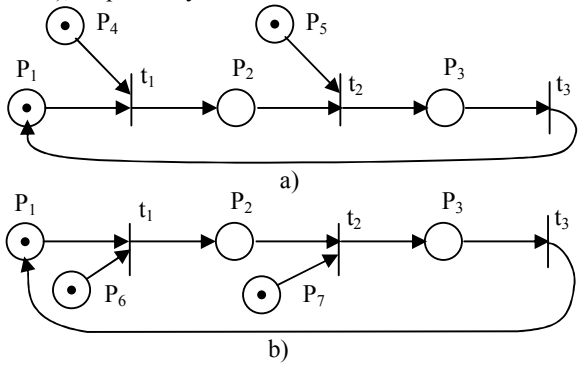

Fig.3 Resource subnets: (a) for M_1; (b) for M_2.

In Fig.4 are depicted the two job subnets of the example given in Fig.2. We notice that the two machines are complementary, i.e., they execute alternatively the same jobs, in order to ensure the necessary throughput of the system.

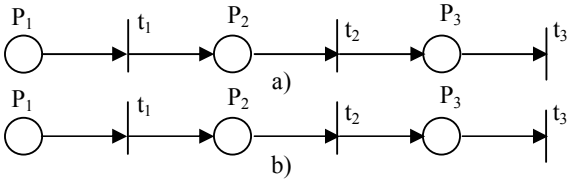

Fig.4 Job subnets: (a) for M_1; (b) for M_2.

Following the construction algorithm of a GPN, as it is described above, we obtain the global Petri net (GPN) model of the FMS, in Fig.5.

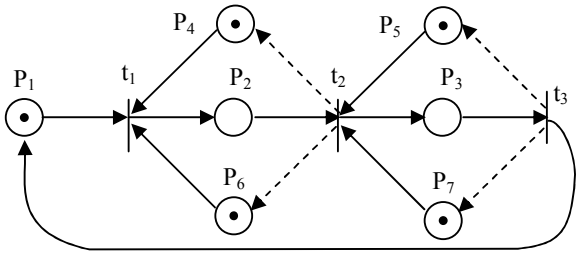

Fig.5 The GPN model for the example given in Fig.2.

We notice that in Fig.5, in order to ensure the liveness of the GPN model, we added some arcs (drawn with dotted lines) which by creating a control mechanism kanban, ensure the verity of the relation [6] (i.e., the PPP avoidance algorithm):

$$\Sigma M(p) \geq 1 \qquad (2)$$

Where M(p) is the marking of the places in the siphon S, and the relation (2) must be verified for all markings M reachable from the initial marking. We apply relation (2) to potential peril paths, PPP, (i.e., kanban structures, which are constructed as shown in Fig.5) which represent, obviously, an extension of syphons (a syphon is a set of places $S \neq \emptyset$ such that $\bullet S \leq S \bullet$, where $\bullet S$ is the input transition of S [7]). In [6] it is proven that if relation (2) is true, then the respective syphon is active, i.e., liveness. It can be easily observed that kanban mechanisms ensure the verity of relation (2), so that PPP such defined are liveness, and they characterize the GPN structure.

For the GPN model of a FMS there is one problem to be solved: evaluating the system performance.

3. Performance evaluation of GPNs

In order to study the performance of a system, the GPN model built in the previous selection is extended to include the notion of time [5,6].

In such extended GPN, an execution time τ is associated with each transition. When a transition initiates its execution, it takes τ units of time to complete its execution. In this paper we extend the performance evaluation given in [7] to GPN characterized by the PPP structures. We mention a theorem given in [7], which states the basis of our algorithm:

Theorem: For a sure Petri net, the minimum cycle time (maximum performance) C is given by:

$$C = \max \{ \frac{T_k}{N_k}, k = 1,2,\ldots,n \} \qquad (3)$$

Where $T_k = \sum_{t_i \in L_k} \tau_i$ = sum of the execution times of the transition in circuit k

$N_k = \sum_{p_i \in L_k} M_i$ = total number of tokens in the places in circuit k

n = number of circuits in the net

L_k = loop (circuit) k

Because a GPN is a sure net, we can apply this theorem to our model of a FMS. A drawback of this approach is that all circuits in the net must be enumerated. In the design of FMS, the required performance is usually given. We give the next procedure for verifying system performance, and we exemplify its extension on a GPN, i.e., for the GPN given in Fig.5, where the firing time τ_i of transitions t_i, i = 1,2,3, are: $\tau_1 = 5$, $\tau_2 = 4$, $\tau_3 = 2$ units of time (u.t.).

A. Algorithm for verifying FMS performance:

1) Express the token loading in an n x n matrix P, where n is the number of places in the GPN model of the FMS. Cell (p_i, p_j), $i \neq j$, in the matrix equal q if there are q tokens in place p_i, and place p_i is connected directly to place p_j by a transition; otherwise (p_i, p_j) equals o. Matrix P of the example system in Fig.5 is shown in Fig.6.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7
p_1	0	1	0	0	0	0	0
p_2	0	0	0	0	0	0	0
p_3	0	0	0	0	0	0	0
p_4	0	1	0	0	0	0	0
p_5	0	0	1	0	0	0	0
p_6	0	1	0	0	0	0	0
p_7	0	0	1	0	0	0	0

Fig.6 Matrix P for GPN given in Fig.5

2) Express transition time in an n x n matrix Q. Cell (p_i, p_j), $i \neq j$, in the matrix equal to τ_i if p_i is an input place of transition i and p_j is one of its output places. Cell (p_i, p_j) contains symbol x if p_i and p_j are not connected. Matrix Q for the example system is given in Fig.7.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7
p_1	x	5	x	x	x	x	x
p_2	x	x	4	5	x	5	x
p_3	2	x	x	x	2	x	2
p_4	x	5	x	x	x	x	x
p_5	x	x	4	x	x	x	x
p_6	x	5	x	x	x	x	x
p_7	x	x	4	x	x	x	x

Fig.7 Matrix Q for GPN given in Fig.5

3) Compute matrix CP-Q (with n-w = ∞, $\forall$ n$\in R_t$) then use Floyd's algorithm to compute the shortest distance between every pair of nodes using matrix CP-Q as the distance matrix. There are three cases:

a) All diagonal cells of matrix CP-Q are positive (i.e., CN_k-T_k > 0), the performance is higher than the given requirement.
b) There are diagonal cells equal to zero and positive, the system performance meets the given requirement.
c) Some diagonal cells of matrix CP-Q are negative; the system performance is lower than the given requirements.

In the example, C = 11, so that CP-Q is given in Fig.8.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7
p_1	∞	6	∞	∞	∞	∞	∞
p_2	∞	∞	- 4	- 5	∞	∞	∞
p_3	- 2	∞	∞	∞	- 2	∞	- 2
p_4	∞	6	∞	∞	∞	∞	∞
p_5	∞	∞	7	∞	∞	∞	∞
p_6	∞	6	∞	∞	∞	∞	∞
p_7	∞	∞	7	∞	∞	∞	∞

Fig.8 Matrix CP-Q for GPN given in Fig.5

After applying Floyd's algorithm to find the shortest distance between every pair of places, as it can easily be seen from Fig.8, the diagonal entries of matrix CP-Q are positive, and the rest of them are zero. This implies that the performance requirement of C = 11 is satisfied. Since cells (p_1, p_1), (p_2, p_2) and (p_3,p_3) are zero's, the bottleneck circuit is $p_1t_1p_2t_2p_3t_3$, as it is proven in [6].

4. Conclusions

In this paper we have discussed a systematic method to build the Petri net model of a controller that keeps a FMS live and we have also discussed an algorithm to evaluate and verify the performance of FMS. We first adapted a bottom up approach to develop a kanban controlled Petri net model, i.e. global Petri net – GPN, of a FMS and we delineate the minimal resource requirements as a necessary and sufficient condition to keep the GPN live.

The performance evaluation of the GPN model was also considered by adopting an algorithm for the GPN model proposed here. In the case of general Petri nets models, the verification of system performance no efficient heuristics methods are known. Further research will focus on this attempt. Colored Petri nets seem to permit a much more flexible approach. Further work will investigate this path.

References

[1] R. Zurawski, M. Ch. Zhon, "Petri nets and industrial applications: A tutorial ", *IEEE Trans. On Ind. Electr.*, Vol. 41, no. 6, pp. 567-583, 1994.

[2] N. Viswanadham, "Composite performance dependability analysis of cellular manufacturing system", *IEEE Trans. On Rob. And Autom.*, Vol.10.

[3] F. S. Hsich, S. Ch. Chang, "Dispatching driven deadlock avoidance controller synthesis for flexible manufacturing systems", *IEEE Trans. On Rob. And Autom.*, Vol. 10, no. 2, pp. 196-209, 1994.

[4] M. D. Jeng, F. Di Cesare, "A review of synthesis techniques for Petri nets, Proc. Of Rensselaer's Second Intern Conference on CIM", *Troy, New York*, May 20-22, pp. 348-355, 1990.

[5] K. P. Valovanis, "On the hierarchical modelling analysis and simulation of FMS with extended Petri nets", *IEEE Trans. On Syst. Man. and Cybernetics*, Vol. 20, no. 1, pp. 94-110, 1990.

[6] C. Ramamoorthy, G. Ho, "Performance evaluation of asynchronous concurrent systems using Petri nets", *IEEE Trans. On Soft Eng.*, Vol. SE-6, no.5, pp. 440-448, 1980.

[7] M. Iordache, J. Moody, P. Antsaklis, "Synthesis of deadlock prevention supervisors using Petri nets", *IEEE Trans. on Rob. And Autom.*, Vol. 18, no. 1, pp. 59-68, 2002.

[8] A. Aybar and A. Iftar, "Overlapping decompositions and expansions of Petri nets", *IEEE Trans. on Rob. and Autom. Contr.*, Vol. 47, no. 3, pp. 511-515, 2002.

[9] E.Mengui, J.L.Boimond, L.Hardouin, J.L.Ferrier, "Just in time control of event graphs: update of reference input, presence of uncontrollable input", IEEE Trans. on Autom. Contr., vol.45, no.9, pp.2155-2159, 2000.