Petri net approach to solve a flexible manufacturing system problems

M.A. Shouman [a] and Ibrahim M. Buseif [b]
[a] *Faculty of Computers and Informatics, Zagazig University, Zagazig, Egypt*
Profshouman@hotmail.com
[b] *Faculty of Eng., Industrial Eng. Dept., University of Garyounis, Benghazi, Libya*

In this work, Petri nets are used to analyse Flexible Manufacturing Systems (FMS) to derive a theorem for checking the buffer overflows in an FMS. The second objective is to schedule the earliest starting times for a sequence of operations in an FMS by the timed Petri net. The FMS may involve many part types, machines, pallets and so on. The third objective is to detect the deadlock phenomenon by a corrected algorithm of the P - invariant theory for Petri nets. This method will enable the manager to know whether there exists any deadlock in an FMS. The Petri net is basically a graph structure. By adding the activity duration times and resource allocation conditions, the dynamic behaviour of a system at any one time can be studied.

هذه الورقة البحثية تهدف إلى وضع الحلول المناسبة لتكدس المواد بالقرب من الالات مع جدولة تسلسل عمليات التصنيع واكتشاف الإختناقات ووضع الحل المناسب لها ٠تعتبر شبكة بيترى من الأدوات المفضلة والأكثر استخداما فى تصميم أنظمة التصنيع المرن وأكثر فعالية فى تحليل المشاكل مثل الإختناقات، الجدولة، الإنتاجية... الخ. بإضافة مصادر المواد الخام أو المعدات إلى شبكة بيترى تتحول شبكة بيترى الإعتيادية الثابتة إلى شبكة بيترى الديناميكية٠ ومن خلال ذلك نستطيع إيجاد الإختناقات أو تكدس المواد وذلك بمعرفة خواص شبكة بيترى. أما جدولة عمليات التصنيع فتتم بإضافة زمن كل عملية إلى شبكة بيترى وتسمى شبكة بيترى الزمنية. تم عمل برنامج كمبيوتر لشبكة بيترى لإيجاد وأكتشاف تكدس المواد للإختناقات ووضع الحل لها من خلال خواص شبكة بيترى وماركت البيانية.

Keywords: Automated manufacturing systems, Petri nets, Automatic control, Structured analysis, Computer control

1. Introduction and background

In the present competitive market, greater flexibility is required for the automated manufacturing systems in order to respond to the needs of different product types of market. But this greater flexibility implies a greater complexity in manufacturing processes. In order to run Flexible Manufacturing Systems (FMS) smoothly and efficiently, mathematical or computerized tools may be used.

In this paper the study of Petri nets which is gaining more popularity in the analysis of the FMS performance such as scheduling, throughput, boundness, and deadlocks. Petri nets are suitable tool for analyzing these problems. It was originally developed in computer science field where many interactions and dynamic behaviours in computer architecture can be determined by Petri nets.

The Petri net is basically a graph structure. By adding the activity duration times and resource allocation conditions, the dynamic behaviour of a system can be obtained at any time. The emphasis of this work is to solve some problems in FMS such as checking buffer overflows, determining the earliest starting times of the machines, identifying deadlocks.

A Work In Process (WIP) of FMS analysis methods based on (Time Petri Net) unfolding is proposed by Kun [1]. Unfolding of PN is a partial order based method for the verification of concurrent system without the state space explosion. The aim of this work is to formulate the general cyclic state-scheduling problem to minimize WIP to satisfy economical constraints. The method is based on unfolding of the original net into the equivalent acyclic description.

Chung et al. [2] presented a Genetic Algorithm (GA) for embedded adaptive scheduling over a Timed Petri Net (TPN) model provides a new method for a FMS. The chromosome

representation of the search nodes is constructed directly from TPN model of FMS.A TPN based schedule builder receives a chromosome and an initial marking as input, and then produces a near-optimal schedule.

Petri net based method for dead lock-free scheduling of flexible manufacturing systems is presented by Ziong and Zhou [3]. The deadlock states are explicitly defined in Petri net framework, so no more equations are used to describe deadlock avoidance constraints to derive deadlock-free schedules. Hybrid heuristic search algorithm, which combines heuristic best-first search and near-optimal backtracking search, is proposed to generate an optimal or near-optimal deadlock-free schedule. Interesting results are reported through comparison of the Petri net method and other methods such as mathematical programming and dispatching rules.

A Petri net model, called Colored Resource-Oriented Petri Net (CROPN) is developed by Wu [4]. The concurrent resource contention and the important characteristics of the production processes necessary for deadlock control are well represented by this model based on the developed model, necessary and sufficient conditions and an efficient control law are presented for deadlock-free operation in FMSs. This control law is a policy of dynamic resource allocation. It determines when a resource can be allocated to which job to avoid deadlock. This control law allows as many active parts as possible to be in the system, while deadlock is totally avoided. This control law is easy to implement and can be embedded into the real-time scheduler.

A heuristic search method using Petri net structures for FMS scheduling is presented by Jeng and Chen [5]. To minimize make-span, an FMS scheduling problem is formulated as finding a firing sequence reveals the minimal cost. The reachability graph is partially generated and searched. The search process is guided by a heuristic function based on firing count vectors of the state equation, predicting the total cost. The proposed algorithm exploits the concurrency information to reduce the searched state space.

A general approach to modelling and analysis of Flexible Manufacturing Systems (FMSs) is proposed by Basile et al. [6]. It puts together a trace-based formal specification method and a compositional Petri Nets (PN) approach with predefined building blocks. The structure of this system leads to a unified framework whose goals are to cope with the complexity of the FMSs behaviours and constraints, and provide a practical engineering means to translate behaviours into PN libraries. The P-invariants of the resulting model are used to obtain a reduced net of the system which points out the resources availability rather than the behaviours of the system components.

Zhang et al. [7] introduced a Generic Petri Net (GPN) model and proposes an approach for the development of control software for FMSs. The principle of this approach is based on checking the control parts of FMSs with the help of temporal relationships between physical operation, and the specification of the FMS controller with GPN. The strategy of GPM modelling is then incorporated with more general strategies in artificial intelligence. A case study for testing FMS controller software is provided to show effectiveness and cost saving over development of conventional methods in which only ordinary Petri net and procedural language is used.

A software tool for modelling and simulation of flexible manufacturing systems is proposed by. Patak and Struhar [8]. A variety of robots, conveyors and working machines can be used in building the FMSs control programs for the whole system are written through Petri net models. Transformation from Petri net into control program in automated programming language is derived from real language for robot control. Simulation is done in 3D environment with full real-time animation.

A Colored Timed Petri Net (CTPN) is used to model the process behaviour of an FMS by Kuo and huang [9]. CTPN-based SPC (statistical process control), fault diagnosis, and failure model and effect analysis are modelled and analysed individually. The proposed CTPN-based simulator can be acted as a real-time FMS monitor and controller through the G2 standard interface.

An optimal deadlock prevention policy for flexible manufacturing systems (FMSs) is proposed by Uzam [10]. The proposed optimal

deadlock prevention policy is based on the use of reachability graph analysis of a Petri Net Model (PNM) of a given FMS and the synthesis of a set of new Net elements, namely places with initial marking and related arcs, to be added to the PNM, using the theory of regions, which is a formal synthesis technique to derive Petri nets from automaton-based models. The policy proposed is optimal in the sense that it allows the maximal use of resources in the system according to the production requirements.

Chen, and Chen [11] introduced an object-oriented approach to modelling of FMS dynamic tool allocation and control under a non-hierarchical shop floor control scheme using modelling method. The complete FMS model is partitioned into individual classes (machines, magazines, tool transport system, SGVs, tool storage, etc.) thereby significantly reducing the complexity of the model to a tractable size .The proposed method can provide the designer of a tool management system with a high-level and structured representation of tool sharing control. It also provides an effective method for prototyping and evaluating performance of object-oriented shop floor control software.

Fu-Shiung [12] presented a framework to model and control FMS based on fusion of perti net and multi-agent system theory. The main results include: (1) a multi-agent model and a collaboration process to form commitment graphs in FMS based on contract net protocol, (2) a procedure to convert commitment graph to Collaborative Perti Net (CPN), and (3) feasible conditions and collaborative algorithms to award contracts in FMS based on CPNs.

2. Petri nets

A Petri Net (PN) is formally defined as a four-tuple C= (P, T, I, O) where P is a finite set of places p, T is a finite set of transitions t, I is a mapping from transitions to a bag of places such that I (t) defines the input places of transition t, and O are mapping from transitions to a bag of places such that O (t) defines the output places of t. A PN can also be represented by a bipartite directed graph with two types of nodes: circles for places and bars for transitions. Directed arcs connected the circles and bars. Note that a bag is a multiple set occurrences of an element that are allowed. Let $B(p,t)$ and $F(p,t)$ be, respectively, the number of occurrences of places P in the input and output bags of transition t. Then B, F and $A = F - B$, respectively, define the backward, forward and incidence matrices of PN. These matrices define the PN topology. The dynamics of PN are defined by marking μ_0 of the PN; μ is a state vector with $\mu(p)$ being the number of tokens in place p. The PN dynamics are controlled by the execution of it. A PN executes by firing its transitions. A transition fires by removing tokens from its input places and depositing tokens at its output places. A transition may fire if it is enabled. A transition t is enabled in marking μ if $\mu \geq B.f_t$ where f_t =(0,0,...,1,0,...,0) with 1 corresponding to transition t. If μ' is a new marking after firing transition t, then $\mu' = \mu + A.f_t$ defines the PN dynamics. For a sequence σ of n transitions, the dynamics equation becomes $\mu_n = \mu_0 + A.f_t$ where $f = \sum_{t \in \tau}$, τ is a set of n transitions and μ_0 is the initial marking; f is called the firing sequence vector. Each marking defines a state. Firing a transition may result in a new state. All the possible states define the PN state space. From an analytical perspective, it is quite important to determine all the *reachable states*. It is also important to determine whether or not PN is *live or deadlock free*, *bounded* (number of tokens in any place is finite in any marking), *conservative* (the weighted number of tokens in any marking is fixed and finite) and *consistent* (there is a firing vector with all positive elements). A live and consistent PN is cyclic, which is typical property of manufacturing systems. One may also be interested in other features of a PN as a controller, such as *recoverability* and *fairness*. Some of these properties can be mathematically analyzed through P- and T- invariants of PN [13-16].

3. Buffer overflow detection and determination of the earliest schedule time

3.1. Buffer overflow analysis

Definition - 1: *A* Petri net C = (P, T, I, O) with

initial marking μ is strictly conservative if for all $\mu` \in R(C, \mu)$,

$$\sum \mu` (p_i) = \sum \mu (p_i),\ p_i \in P \qquad p_i \in P.$$

If a marked Petri Net model is conservative then the sum of all tokens will remain a constant in all-reachable markings. Such a PN model will represent a system with a constant number of resources or jobs, for example, a closed manufacturing system. If it is required to prove the conservative property of a Petri Net model, then a nonzero weight vector N should be defined such that the weighted sums over all the reachable markings are equal. In a real system, the meaning of the conservative property is that the system is operating normally and there are no resources, such as machines, robots, and AGV's, which are inoperative. In an FMS, if the buffer overflow situation is considered, then the machine (s) before an overflow buffer will be forced down and the marked Petri Net will not possess the conservative property [14,15]. The mathematical derivation of the process is shown as follows:

$$\mu_0 * N = \mu * N \quad (N \text{ is a nonzero vector}), \qquad (1)$$

where μ and μ_0 are 1 by m matrices, and N is an m by 1 matrix.

$$\mu \in R [\mu_0].$$

So there must exist a sequence of transitions to be fired.

where,

$$\mu = \mu_0 + Y * C. \qquad (2)$$

Combined (1) and (2) we get:

$$\mu_0 * N = (\mu_0 + Y * C) * N$$
$$= \mu_0 * N + (Y * C) * N,$$

and

$$(\mu_0 * N) - (\mu_0 * N) = (Y * C) * N.$$

By the association of matrix multiplication: $(\mu_0 - \mu_0) * N = (Y * C) * N$, and for a zero – matrix: $0 * N = Y * (C * N)$, then:

$$0 = Y * (C * N). \qquad (3)$$

This is true for every Y vector. The result can be as follows: "A Petri net model is conservative $\Leftrightarrow$ There exists a nonzero positive vector N such that $C * N = 0$".

The example in fig. 1 is used to verify that the above theorem is a sufficient condition but not a necessary condition. The example is an FMC, which has one machine and two part types.

This machine can only serve one part type at a time. Transitions and places are changed slightly. Let p_1 and p_3 be the input buffer for part A and part B respectively, and p_2 be the machine. Let t_1 and t_2 represent starting times of machining part A and part B separately, t_3 and t_5 denote the finishing times of machining A and B, and t_4 is the return operation of this machine. p_4 and p_5 are the operational processes of parts A and B individually.

$$C = \begin{vmatrix} -1 & 0 & 1 & 0 & 0 \\ -1 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 1 & 0 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 & -1 \end{vmatrix}$$

$$-n_1 + n_3 = 0$$
$$-n_1 - n_2 + n_4 = 0$$
$$-n_2 + n_5 = 0$$
$$n_1 - n_3 - n_4 = 0$$
$$n_2 - n_4 - n_5 = 0$$

The first column (-1, -1, 0, 1, 0) of the incidence matrix indicates that p_4 (the machining operation of part A) must wait until p_1 (input part for part A) and p_2 (the machine) are both available. The third column (1, 0, 0, -1, 0) indicates that p_1 can only send one more part A to the machine until it finishes processing a part p_4. Other columns in the incidence matrix are obtained similarly.

The system is not conservative from the following calculations. Then $n_1 = n_4 = n_5 = -n_3$ and $n_2 = 0$. So the PNM is not conservative and buffer overflows may occur. Some machines will be forced down due to buffer overflows [16-18].

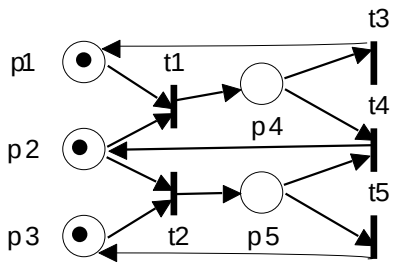

Fig. 1. PNM for a system of one robot and two part types.

3.2. TPN and earliest schedule for FMS operations

In this section, Timed Petri Net (TPN) is used to apply a certain methodology for finding the earliest starting times for a sequence of transitions corresponding to the real operations of a system. The earliest schedule for a sequence of transitions can be used to decide the earliest possible starting times for each machine when there are no machine breakdowns, part jams or other delays.

3.2.1. Timed petri net

By adding the ' time quantity 'to the ordinary (pure) Petri net for every transition, a new PN called TPN is obtained. The TPN is a powerful tool for modelling systems with dynamic behaviours. In such extended nets, an execution time r is associated with each transition is fired, it takes r units of time to complete its execution [16, 18].

Definition 2: A timed Petri net is a five – tuple structure $C = (P, T, I, O, R)$, P, T, O, and I are defined before, and R is a function of the following type: R: $t \rightarrow R$, where R is a positive real number and t is a transition.

The main concept of the methodology is to use a stack and its recursive calls [16,17, 19].

1. First, the overall Petri Net Module (PNM) is traced and finds all time point sets of transitions, which are possibly fired.
2. List time point sets of transitions and denote them by A (1), A (2),..., $A(N)$ in a table.
3. Use the recursive calls and stack to partition the friable sequence of transitions. For example, if $\delta = (((((\ldots (t_1\ (t_2)\ t_3)\ t_4) \ldots t_{n-1})\ t_n)$, put them into a stacks as follows. S_1 is a pointer to the top position of the stack and S_n indicates position n in the stack.
4. Let the initial condition $a_1 = 0$ for transition t_1.
5. Determine a_2 if the following two conditions are satisfied: (a) $X \in A$ (a_1) and X is active at a_1, and (b) $\mu(p)$ at the end of $x \geq I$ (p, t_2). The meaning of condition (b) is that the number of tokens, which are required to start $t2$, must be larger or equal to the present token number in the places.
6. Pop $S_2 = (t_1 t_2)\ t_3$ to determine a_3.
7. Continue to calculate S_3, S_4,..., S_{n-1} until all the (a_1, a_2,..., a_n) are found.

3.2.1. An example

Fig. 2 is a PNM of an FMS. The system contains one lathe, one milling machine and one drilling machine. The system produces two part types. Both of t_1 (3) and t_3 (3) represent the operational times of part type A at p_2 (lathe) and p_5 (drilling machine) separately. The t_2 (2) and t_4 (3) represent the operational times of part B at p_2 (lathe) and p_5 (drilling machine). Transition t_6 (1) is the transfer time of a part. P_7 denotes the milling machine. The friable sequence of transitions (operations) are $\delta = t_1 t_7 t_2 t_5 t_2 t_7 t_1 t_4 t_3 t_6$. The above method is used to find the earliest schedule for this sequence.

Step 1:

We trace the entire PNM by moving the friable tokens to classify the event sets. We use three three tuples ($a1$: B, a_2: E, t) where a_1 represents the time when transition t is fired, and a_2 represents the finishing time of t.

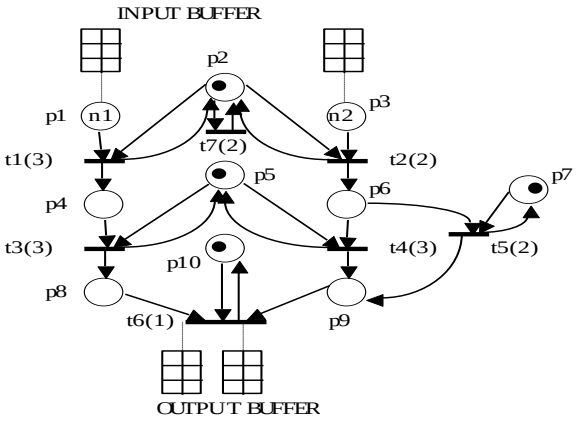

Fig. 2. The FMS example.

Step 2:
All the possible friable and ending times for all transitions are showing in the following table:

All the possible friable and ending times for all transitions

0: B	t_1, t_2, t_7	2: E	t_2, t_7
2: B	t_4, t_5	3: E	t_1
5: B	t_3	4: E	t_5
8: B	t_6	5: E	t_4
		8: E	t_3
		9: E	t_6

Step 3:
$(t_1)\ t_7$, $(\ t_1 t_7\ )t_2$, …. , $(\ t_1 t_7 \ldots t_4\ )t_3$, and $(\ t_1 t_7 \ldots t_3\ )t_6$.
Step 4:
$a_1 = 0$ for all t_1.
Step 5:
$X \in A$ (0) and $\mu(P)$ at the finishing time of $X \geq I$ (P, t_7), and we get $X = 0$.
Step 6:
Determine a_3 from the schedule $(a1, a_2) = (0, 0)$ and $a_3 = 0$ for transition t_2.
Step 7:
The above steps are continued to obtain the following schedule diagram.
The overall scheduling starting times are shown in fig. 3, whereas t_{iS} are different from those in fig. 2. Here, the t_{iS} represent the earliest starting times for the operations from the start of FMS at time 0.

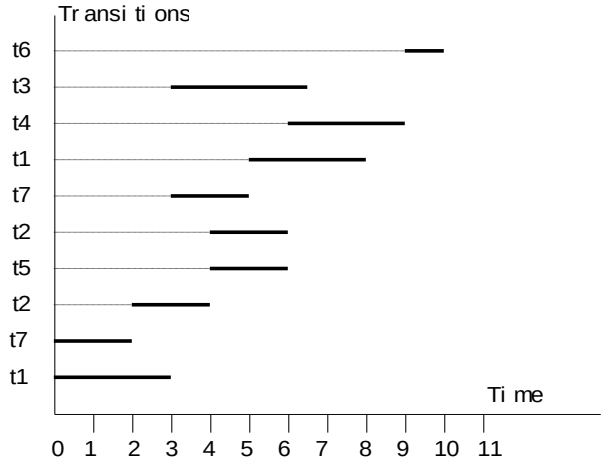

Fig. 3 The starting times of sequence of transitions sequence.

4. Deadlock detection in the PNM

4.1. Deadlock detection by using the p-invariant

This method includes the following steps:
1. Before obtaining the final PN model of a given system, one can construct the union of simpler PN models that represent the various functional subsystems into which the given system can be decomposed. This divide-and-conquer approach to obtaining the PN of a given system can help us in computing P-invariants of a PN models. If a given system is very large, the corresponding PNM is also large. The computation of P-invariant will become very complex. So the overall system should be partitioned to compute the P-invariant of subsystems and then combine them to obtain the overall P-invariant of the entire system.
2. The method can help to compute the P-invariant of the union of two PNs when P-invariant of individual nets are known and the nets satisfy the theorem conditions.
3. Basically the method uses the incidence matrix representation C of a PN to compute P-invariant characterized by U through the equation $U * C = 0$. Again from the equation;

$\mu = \mu_0 + C*Y$, multiplying by U, we obtain:

$U*\mu = U*\mu o + U*(C*Y)$,

where,

$U*C = 0$.

Then,

$$U*\mu = U*\mu_0 \ . \qquad (4)$$

4. Step 3 is used to get some equations and then choose some states to verify if these states are deadlocked. Then one can say if the original system is deadlocked or not.

By applying the same example (fig. 1) to detect the deadlock of a PN model by using P-invariant. First the incidence matrix are evaluated by using this equation $C = C^+ - C^-$, where C is the incidence matrix and C^+ is output matrix and C^- is an input matrix.

$$C = \begin{vmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{vmatrix} - \begin{vmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{vmatrix}$$

$$= \begin{vmatrix} -1 & 0 & 1 & 0 & 0 \\ -1 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 1 & 0 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 & -1 \end{vmatrix}$$

The relation $U * C = 0$, $u_i \geq 0$ can rewritten as:

$-u_1 - u_3 = 0$

$- u_1 - u_2 + u_4 = 0$

$- u_2 + u_5 = 0$

$+ u_1 - u_3 - u_4 = 0$

$u_2 - u_4 - u_5 = 0$

From these equations, it can be concluded that:

$u_1 = u_4 = u_5 = -u_3$, and $u_2 = 0$.

Some values of $u_i < 0$, then this means that the PN model is not deadlock-free.

The implementation of the above method in computers has some difficulties. The first one is how to solve the equation $U * C = 0$. Because there are no exact solutions of the equation, we must assume some unknown integers to represent these variables. Using a computer it is difficult to implement the process. The second problem is in step 4, where it is difficult to choose a special marking of deadlock property. The computer is not intelligent enough to know the event.

The completeness and consistency examination are closed linked to the reachability problem. In order to illustrate this fact let us introduce the concept of a Marked Graph (MG) which is a Petri net $C = (P, T, I, O, \mu_0)$ [17- 19].

4.2. Marked graph

A marked graph is a PN in which each place is an input for exactly one transition and an output for exactly one transition. Alternatively, each place exactly considered as one input and one output [13, 20].

A marked graph is a PN $C = (P, T, I, O)$ such that for each $p_i \in P, /I(p_i)/ = /\{t_j / p_i O(t_j)\}/ = 1$ *and* $/O(p_i)/ = /\{t_j / p_i I(t_j)\}/ = 1$.

Marked graphs can model concurrence and synchronisation but cannot model conflict or data-dependent decisions. The properties, which have been investigated for marked graphs, have been Liveness, safeness, and reachability.

In the investigation of these properties, the major structural parts of a marked graph of interest are its cycles. A cycle in a marked graph is a sequence of transitions $t_{j1}t_{j2}...t_{jk}$ such that for each t_{jr} and t_{jr+1} in the sequence there is a place p_{ir} *the* $p_{ir} \in O(t_{jr})$ and $p_{ir} \in I(t_{jr+1})$ and $t_{j1} = t_{jk}$.

A cycle is such a closed path from a transition back to that same transition. The importance of cycles for marked graphs derives from a number of theorems that are coverd in the paper [16].

4.3. Computer implementation of the corrected algorithm

The mean idea of the use of P-invariants to detect the system is free of deadlocks or not, and also to determine the other properties of PN (Safe, Reachable and Conservative). This algorithm is represented in fig. 4 [16, 18].

In order to illustrate the application of the approach proposed, consider the example provided in fig. 1. An event graph (like PN in general) said to be strongly connected if there is a directed path joining any node A to any node B of the graph.

The event graph presented in fig. 1 strongly connected. In this aspect, an elementary circuit in a strongly connected event as a directed path goes from one node, i.e. a place or transition, back to the same node, which any other node is not repeated. For instance, fig. 1 exposes four elementary circuits, namely:

$\gamma_1 = (p1, t_2, p_4, t_3)$,

$\gamma_2 = (p2, t_2, p_4, t_4)$,

$\gamma_3 = (p2, t_1, p_5, t_4)$, and

$\gamma_4 = (p3, t_1, p_5, t_5)$.

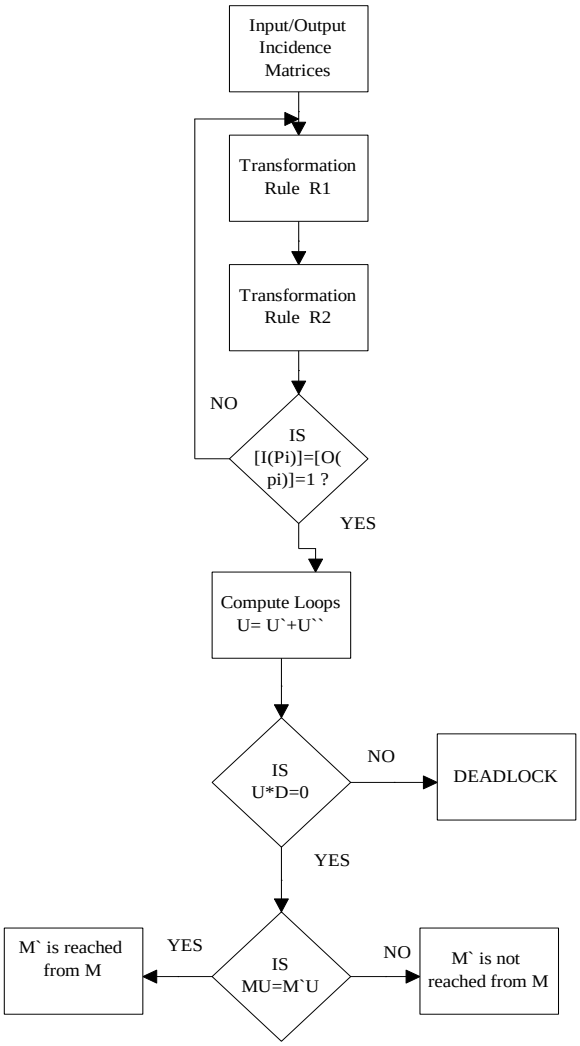

Fig. 4. Flow chart of computing P-invariants

If the number of tokens in a marking remains fixed for all markings in the reachability set, the Petri net is then said to be strictly conservative. An immediate consequence of conservativeness is boundedness of the Petri net. The boundedness property implies that the number of tokens in any place does not increase beyond a limit. This in turn guarantees a finite reachability set for the Petri net. In some manufacturing applications, the tokens in the Petri net model could represent moving entities, such as workpieces or parts, in the system. Here, the strict conservativeness guarantees that the total number of parts in the system remains fixed at all times. Through, the conservativeness property can be established through the reachability set or graph, a more efficient approach would be through the P-invariants of a Petri net. If there exist a U with all positive elements the Petri net is then said to be conservative. If U = (1,...,1) then the Petri net is strictly conservative.

The total number of tokens in γ_i (i=1,2,...,4) is then:

$n_1(\gamma_1)$ = (number of tokens in p_1) + (number of tokens in p_4) = 1 + 0 = 1

$n_2(\gamma_2) = \mu(p_2) + \mu(p_4) = 1 + 0 = 1$,

$n_3(\gamma_3) = \mu(p_2) + \mu(p_5) = 1 + 0 = 1$, and

$n_4(\gamma_4) = \mu(p_3) + \mu(p_5) = 1 + 0 = 1$.

Therefore, the relevant set of P-invariants contains the following elements;

$U1$= (10010), $U2$ = (01010), $U3$ = (01001), $U4$ = (00101), $U5$ = (12122)

4.4. Applying the equations

U^*C = (12122) * $C \neq 0$, this means that the system is not Live (Deadlock is not free). The system is not conservative from the previous equation. So the PNM is not conservative and the buffer overflows may occur.

$U^T * \mu$ = (12122) ^{T} (11100) = 4, and $U^T * \mu_0$ = (12122) ^{T} (00110) = 3, this means that the marking μ cannot be reached from μ_0, i.e. that $\mu \not\subset R(C, \mu_0)$. But, in each cycle there is only one token, it means that the system is Safe.

Let consider the example provided in the previous example of fig. 1 by using the software analysis.

1. The result of applying two rules R_1 and R_2 were as shown in fig. 5:

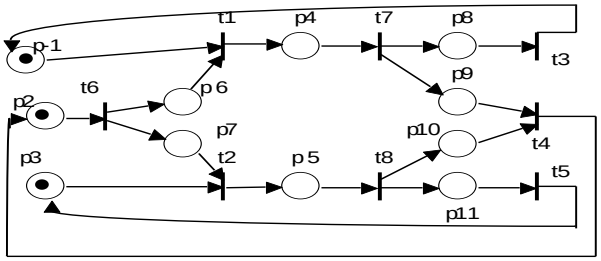

Fig. 5. New Petri net model after applying R_1 and R_2 rules.

2. Computes loops of the new PN model by using the computer software.

Loops:

$p1^*$; $p4$; $p8$; (1: $p1$),

$p2^*$; $p6$; $p4$; $p9$; (1: $p2$),

$p2^*$; $p7$; $p5$; $p10$; (1: $p2$), and

$p3^*$; $p5$; $p11$; (1: $p3$).

P invariants (U):

(1,2,1,2,2,1,1,1,1,1,1)

U by incidence matrix = (0,0,0,0,0,0,0,0), then the system is live (deadlock-free)

U by tokens:

$U * \mu_0 = U * \mu = 4$, then μ is reached from μ_0.

Each loop has one token and a marked graph PN in which each place is an input for exactly one transition and an output for exactly one transition, this means that the system is safeness.

5. Conclusions and remarks

In this work, Petri nets are used to study three problems: the check of buffer overflows, the determination of earliest starting times, and deadlock detection. The first two problems can be done manually. A theorm is proposed to find the overflow conditions and a methodology to find the earliest starting times for a given sequence of operations has been suested. This methodology is based on the timed Petri net, which is developed by using the concept of recursion. This method is better than the other methods in its dynamic behaviour. The last problem can be solved by the proposed computer programs, which are based on the Petri net concept.

The advantages of the Petri net include its ease of understanding and its readability. The dynamic behaviour and parallelism are the two major aspects for PN outperform other tools. So Petri net is must suitable for use in the real time control. If the Petri net can be embedded in a controller then it can make fast decisions and correct the system states as needed.

There are some suggestions and future research areas for Petri nets. For the deadlock problem, deadlock prevention and recovery can be a future research.

References

[1] L.J. Kun, "Working in Process (WIP) Analysis of the Flexible Manufacturing System using Time Petri Net Unfolding." In: Proc. IEEE Int. Conf .on System, Man, and Cybernetics (SMC'98), 11-14 October 1998, San Diego, CA, pp. 136-141 (1998).

[2] Y.Y. Chung, and L.C. Fu, and M.W. Lin, "Petri net based Modelling and GA based Scheduling for a Flexible Manufacturing System," In: Proc.37th IEEE Conf. on Decision and Control, 18-20 Tampa, FL, Vol.4, pp. 4346-4347 (1998).

[3] H.H. Ziong, and M. Zhou, "Deadlock-free Scheduling of Flexible Manufacturing System Based on Perti Nets." In: International Journal of Intelligent Control Systems, Vol. 3 (3), pp. 277-295 (1999).

[4] N. Wu, "Necessary and Sufficient Conditions for Deadlock–Free Operation in Flexible Manufacturing Systems Using a Colored Perti Net Model," In: IEEE Trans. on Systems, Man, and Cybernetics: Part C: Applications and Reviews, Vol. 29 (2), pp. a92-204 (1999).

[5] M.D. Jeng, and S.C. Chen, "Heuristic Search Based on Perti Net Structures for FMS Scheduling," In: IEEE Trans. On Industry Applications, Vol.1, pp. 196-202 (1999).

[6] F. Basile, P. Chiacchio, V. Vittorini, and N. Mazzocca, "Specification and Modelling of Flexible Manufacturing Systems Using Behaviours and Petri Nets Building Blocks," In: Proc. Int. Symp. on Software Engineering for Parallel and Distributed Systems, pp. 17-18 (1999), Los Angeles, CA, pp. 110-123 (1999).

[7] Zhang, W.J., Li, Q., Bi, Z, M., and Zha, X.F, "A Generic Petri Net Model for Flexible Manufacturing Systems and its Use for FMS Control Software Testing,"

In: International Journal of Production Research, Vol. 38 (5), pp. 1109-1131 (2000).

[8] R. Patak, and M. Struhar," Modelling and Simulation of Flexible Manufacturing Systems Controlled be Perti Nets." In: proc. Ifac Conf. on Control Systems Design (CSD'2000), 18-20 June 2000, Bratislava, Slovak Republic, pp. 384-389 (2000).

[9] M. Uzam, "An Optimal Deadlock Prevention Policy for Flexible Manufacturing Systems Using Petri Net Models with Resources and The Theory of Region," In: International Journal of Advanced Manufacturing Technology 19 (3), pp. 192-208 (2002).

[10] F.F. Chen, and J. Chen, "Performance Modelling and Evaluation of Dynamic Tool Allocation in Flexible Manufacturing Systems Using Coloured Petri Nets: An Object-Oriented Approach," In: the International Journal of Advanced Manufacturing Technology, Vol. 21 (2), pp. 98-109 (2003).

[11] Hsieh, Fu-Shiung, "Model and Control Holonic Manufacturing Systems Based on Fusion of Contract Nets and Petri Nets," in: Automatica, Vol. 40, pp. 51-57 (2004).

[12] C.H. Kuo, and H.P. Huang, "Failure Modelling and Process Monitoring for Flexible Manufacturing Systems Using Colored Timed Petri Nets," In: IEEE Trans. On Robotics and Automation, Vol. 16 (3), pp. 301-312 (2000).

[13] Z. Banaszak Modelling and Control of FMS: A Petri Net Approach, Wroclaw Technical University Press, Wroclaw (1991).

[14] Z. Banaszak K. Jedrzejek "Rule-Based Knowledge Verification using Petri Nets," The Third Turkish Symposium on AI and Networks (1994).

[15] Banerjee, Al Maliki, "A Structured Approach to FMS Modelling," Int. J. Computer Integrated Manufacturing, Vol.1 (2) (1995).

[16] M.I. Buseif, an Approach to FMS Design Using SADT and PN tools, PhD. Thesis Warsaw University of Technology, Faculty of Production Engineering, Warsaw-Poland (1997).

[17] Chih-Ming Liu and Feng-Cheng Wu, "Using Petri nets to Solve FMS Problems," Int. J. Computer Integrated Manufacturing, Vol. 6 (3), pp. 175-185 (1993).

[18] F.G. Dicesare J.M. Harhalakis, M. Proth, F.B. Silva, and Vernadat, Practice of Petri Nets in Manufacturing, Chapman and Hall (1993).

[19] K. Santarek M. and Ibrahim Buseif, "A Structured Approach to FMS Modelling and Design," First International Conference on Mechanical Engineering Advanced Technology for Industrial Production, Assuit University Egypt (1994).

[20] T.H. Boucher M.A. Jafari, "Design of a Factory Floor Sequence Controller from High Level System Specification," Journal of Manufacturing Systems, Vol. 11 (6) (1990).

Received April 23, 2005
Accepted June 6, 2005