Implementation of Petri Nets Based Controller using SFC

A. Dideban*, M. Kiani, H. Alla*****

** Electrical and Computer Faculty, Semnan University, Semnan, Iran*
(*Tel: +982313354123; e-mail: adideban@semnan.ac.ir*)
*** Electrical and Computer Faculty, Semnan University, Semnan, Iran*
(*e-mail:mokiani@ Semnan.ac.ir*)
**** GIPSA lab, 38402 St Martin d'Heres Cedex, France*
(*e-mail :Hassane.alla@inpg.fr*)

Abstract: This paper presents a method for implementing a controller synthesized by PN-based SCT theory. Despite the great acceptance of SCT in controller design, there are problems with its practical implementation. Sequential Function Chart, as an international standard programming language, is used for implementing PN-based controllers. One problem in the implementation step is the occurrence of simultaneous events in mutually exclusive transitions. In SFC standards, it is possible to define a constant priority (e.g. left to right). A method for developing dynamic priority is presented in this research. If two users have requests for a common resource simultaneously, priority goes to the one that did not use the resource in the earlier turn. Existence of non-safe places in PN model is another problem in the implementation process. In this paper an alternative for this problem is also presented. Therefore, if the controller model becomes non-safe, it could implement using SFC. Finally the idea is demonstrated using an example.

Keywords: Supervisory Controller, Sequential Function Chart (SFC), Controller Implementation, Dynamic Priority

1. INTRODUCTION

Supervisory control theory (SCT), first presented by Ramadge and Wonham (1989), is a general theory for controlling Discrete Event Systems (DES) given a specification describing its allowed and desired behavior. The resulting controller, the supervisor, restricts the behavior of a plant as much as possible that the given specifications are fulfilled. This theory is based on languages and automata. However, the great number of states representing the behaviour of the system, and the lack of structure in the model, limits the possibility of developing an effective algorithm for the analysis and synthesis of real systems. For solving this problem, Petri-net-based approaches to supervisory control design have also been studied by many authors including: Giua and Dicesare (1991), Giua (1992, 1996), Yamalidou et al. (1996), Holloway and Krogh (1990), Kumar and Holloway (1996), Moody and Antsaklis (2000), Uzam and Jones (2002).

In PN-based controller synthesizes using SCT framework, the first step is the modelling of the plant and the specifications and then in the next step, synchronized composition between two models gets the controlled model. Generally, due to uncontrollable and unobservable transitions, it is necessary to change this synchronized model. There are many approaches for resolving this problem (Guia et al. 1992, Dideban and Alla 2005, 2008. The final model is composed of the supervisor and the uncontrolled model of the plant. The next step is implementing this controller.

Implementation of the controller requires an appropriate method for developing a PLC program corresponding with the automaton that represents the theoretical supervisor. Therefore implementing the controller is a matter of developing an appropriate PLC program (M. Cantarelli(2006)).

Programmable logic controller (PLC) is a specific application computer and has greatly been used in new automation systems. The ISO/IEC61131 (2001) standard is defined for PLC. Third part of this standard defines programming languages. Sequential Function Chart (SFC) is one of these languages, which is a graphical and high-level language. This language is inspired from PNs and seems to be the ideal choice for implementing controllers designed by PNs. Ladder Diagram (LD) is another standard language for PLC, which is greatly used by programmers. Conversion of Petri net model into LD have been addressed by Peng and Zhou (2004), Boucher et al. (1989), Lee G.B. and Lee J.S (1995), Jackman et al.(1995), Uzam et al.(1996, 1998), Zhou and Twiss (1998), Chirn and McFarlane (2000). However for complex systems this approach is not efficient and has some difficulties.

Some Researches have been accomplished for converting PNs to SFC: Music and Matko (1998, 1999), Music et al. (2000, 2005), Hellgren et al.(2001, 2005), Ferrarini and Piroddi (2003) and Zhou et al. (1992). In Music and Matko (1998, 1999), Music et al. (2000), Hellgren (2001), simultaneous events in mutually exclusive (SEME) transitions, is discussed and this problem is solved by

creating constant priority. SEME is usually used for resource allocation. With assigning constant priority to users, in simultaneous requesting of multiple users for a common resource, resource always is allocated to the user that has a higher priority. In some cases, this method is constraining and is not efficient. In Hellgren et al. (2005) execution modes of SFC are discussed and for solving SEME, IT/IA (Immediate Transit/ Immediate Action) mode were proposed but generally DT/DA (Deferred Transit/Deferred Action) mode is considered in PLC. In this paper, a method for dynamic priority assignment is presented. In this method, in each turn that two users have requests for a common resource simultaneously, priority goes to the user that did not use the resource in earlier turn. In other words, the priority changes dynamically between the users. The dynamic priority presented here is applicable in all of the three modes of execution IT/IA, IT/DA, and DT/DA, because transition conditions are rewrote in compositional form. Dynamic priority is not needed for all processes. So, simply transition conditions can reform to developing the constant priority.

In previous studies, it is supposed that PNs model is safe and therefore direct conversing of place to step is possible (Hellegren et al. 2005). Another problem in PN-based controller implementation is non-safe places. It is possible that the model of the process and the specification are created by a safe model but when resolving the problem of forbidden states, sometimes the control places may become non safe. In PN to LD conversion approach, a counter for each non-safe place can be used that contains the number of tokens. But in accomplished researches for implementation using SFC, safe PN modelling was used.

If the conversion of none-safe PN to SFC becomes possible, a better use of PNs potential for controller synthesizes is realized and the control system becomes more efficient. In this paper an alternative for this conversion is proposed as well.

This paper is organized as following: In section 2, the preliminary definition about PNs and SFC is given. In Section 3, the SCT briefly is introduced. Controller implementation, dynamic priority and non-safe place conversion is covered in Section 4. In section 5, an example is presented for describing the idea of the paper. Finally, the paper is concluded in section 6.

2. PRELIMINARY DEFINITION

2.1. Petri net

PN is a powerful graphical and mathematical tool for modeling and analyzing DESs. In this paper, it is supposed that the reader is familiar with PN (David and Alla, 2005).

Definition1: A Petri Net is the 5-tuple set, given by:

PN = $<P, T, F, W, M_0>$

Where $\boldsymbol{P = \{p_1, p_2,\ldots, p_3\}}$ is a non-empty and finite set of places and $\boldsymbol{T = \{t_1, t_2,\ldots, t_3\}}$ is a nonempty and finite set of transitions. It is assumed that $\boldsymbol{P \cup T \neq \phi}$ and $\boldsymbol{P \cap T \neq \phi}$. $\boldsymbol{F}$ is the incident relation function which represents the set of directed arcs connecting places to transitions and vice versa. $\boldsymbol{W: F \rightarrow Z^+}$ is the weight function which assigns an integer number as weight to each arc. $\boldsymbol{M_0}$ is the initial marking of PN. $\boldsymbol{(P \times T)}$ is called $\boldsymbol{pre(P_i \times T_j)}$ and represent the arc connected between place (i) and transition (j) which Place(i) is the input place of Transition(j). $\boldsymbol{(T \times P)}$ is called $\boldsymbol{pre(T_i \times P_j)}$ and represents the arc connecting Transition(i) to Place(j); Place(j) is the output place of Transition(i).

2.2. *Sequential Function Chart*

SFC is one of the standard programming languages for PLC (ISO/IEC (2001)). This language is similar to GRAFCET (David and Alla, 1995) and was inspired from PN. In this language, a process is divided into separate parts, which execute sequentially to execution of the whole process. By dividing the process into multiple parts, its management becomes easier.

Elements of an SFC are:

Step (S): Initial step, Simple step, Macro step

Transition (T): Simple, Alternative Branching (OR), Divergence, Convergence (AND), Compositional

Action (A): Action name, Action qualifier

These elements are indicted in figure 1.

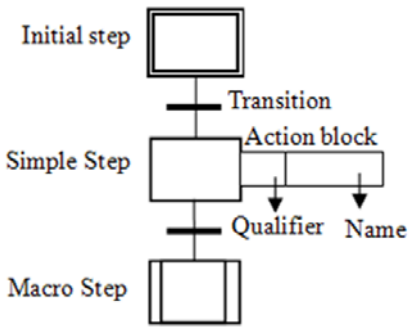

Figure 1. SFC elements.

A is a set of actions which dedicates zero, one or more actions to each Step. Each $a \in A$ has an action name and an action qualifier. A step may have no action. The set of actions belongs to $s \in S$ denoted by $A_C(s)$. S_0 is the initial step. When running the SFC, S_0 becomes active. For each SFC there is just one initial Step.

In SFC, if input steps or the steps before a transition become active, this transition is enabled. If a transition has been enabled and its conditions are true, by the occurrence of the event associated with this transition, it fires. By firing a transition, all the input steps would become inactive and all the output steps would become active. By activation of a step, its actions would be executed.

A transition in SFC might be one of the transitions depicted in Figure 2. Figure 2a represents a simple transition. Figure 2b represents an alternative branching or “OR” transition. Figure 2c represents a divergence transition. In this type of

transition, firing of T_4 activates both the S_7 and S_8 steps. Figure 2d represents a convergence transition. Both S_9 and S_{10} step must be activated in order to enable T_5. Figure 2e represents compositional transition which is made up of divergence and convergence transitions.

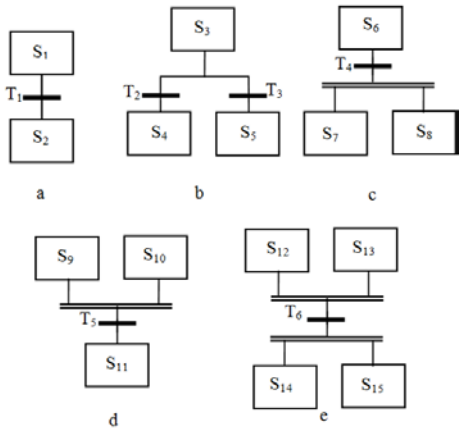

Figure 2. Transition types**:** (a) Simple, (b) alternative branching, (c) divergence transition, (d) convergence transition and (e) compositional transition.

3. CONTROLLER SYNTHESIS

Supervisory Control Theory as a general theory for the controller synthesizes for discrete event systems was introduced with the aim of restricting the behaviour of the system in desired framework in a maximally permissive manner. This theory is based on automata and due to the weakness of automata in modelling of complicated systems, this approach is not effective. PN models have received attention as alternative models for investigating the discrete event control theory (Cassandras and Lafortune, 2008) and many studies have been done for Petri Net based Supervisory Control (PNBSC). In PNBSC, the first step is modelling the uncontrolled plant with PN. Then supervisor controller is designed based on the desired specification. Synchronized composition between supervisor and uncontrolled model gives the closed loop controller.

The next step after synthesizing final controller is implementing the designed controller. Nowadays PLC has a great use in automation systems and seems to be a good choice for being used as the control agent. So the synthesized controller should convert to appropriate language for PLC. IEC1131-3 standard defines some languages for PLC programming, which can be used for this purpose. LD is one of the commonly used programming languages and some studies have been accomplished for converting PN to LD. For complex systems, this approach is not efficient. A big drawback of LD programming is its weak structure and therefore it cannot represent a dynamic system as good as PNs. It causes difficulty in changing, maintaining, and documenting of the program.

The structure of SFC language is similar to PN modelling and is a good choice for implementation. In the next section, controller implementation by SFC will be discussed.

4. CONTROLLER IMPLEMENTATION

As mentioned earlier, some studies are carried out for converting PN into SFC. One of the differences between PN and SFC is related to mutually exclusive transitions. In PN, the order of transitions firing caused by simultaneous events is not important but in SFC, firing order of transitions must be defined for simultaneous event. According to the standard, one transition has priority over the next transition.

In Hellgren et al. (2001, 2002) and Music and Matko (1999), the authors studied SEME transitions and as a result, method for creating constant priority was proposed.

In the following section, converting mutual exclusion in PN to SFC by dynamic priority assignment for transitions is discussed.

4.1. *Mutual exclusion*

One of the modelling capabilities of PN is mutual exclusion (ME) that is usually used for resource allocation (resource sharing) or for making constraint in PN. Figure 3a depicts mutual exclusion. In the case that P_1 has one token; both transitions t_1 and t_2 are enabled. If both conditions of these transitions become true (events allocated to those occurs), only one of them can be fired and after firing, the second transition is not enabled anymore. It is not important that which one of the transitions is fired. In PN, the behaviour of DES can be studied theoretically and both possibilities can be considered in the analysis. Moreover, it is not possible to have simultaneous occurring of events since they are independent. However, when a SFC is used in practical application, this causes a problem. Figure 3b represents a SFC equivalent for Figure 3a.

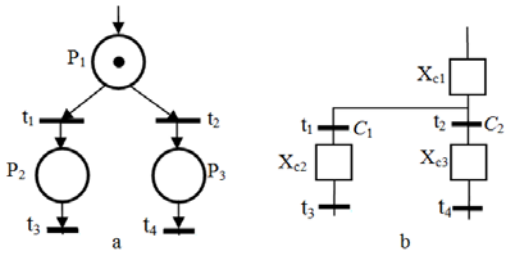

Figure 3. (a) Mutual exclusion, (b) SFC equivalent.

In SFC model, if the conditions of both transitions t_1 and t_2 become true (C_1 and C_2) and X_{c1} has been activated, just one of t_1 or t_2 must fire and it should be defined which one. According to the standard of SFC it is possible to define a constant priority. Assume that t_1 has the priority over t_2. This constant priority is not proper in some situations. For example suppose a system with one manipulator for transferring work-piece in two manufacturing line. After modelling this system, one ME appears in the model for assigning the manipulator to one of the manufacturing lines. If the number of simultaneous requesting for manipulator was high, only one of the manufacturing lines takes the manipulator (the prioritized line). This strategy is not a good and applicable method and in many applications it is better that manipulator is shared equally among the lines. If

dynamic priority is assigned to these lines, in each turn of simultaneous requests, manipulator allocates to one of the lines (one left one right). So this problem can be solved with dynamic priority.

4.2. *Dynamic priority*

As mentioned earlier, in SFC standard constant priority is considered for ME, and so in requesting of multiple processes at the same time, resource is always assigned to the prioritized process. In this section a method to creating dynamic priority is presented to change the priority between the users dynamically. This mechanism must obey the following conditions (two users and one resource):

- If just one user (process) has a request for taking the resource, it can take it.

- If two users have requests for taking the resource at the same time, the one which did not use the resource in the previous simultaneous request, this turn can take the resource.

For creating dynamic priority for two mutual exclusive transitions, the previous state that SEME occurred is needed (which transition fires). It must be defined that resource is allocated to which user, and based on this allocation, the proper priority can be determined. Two steps and two transitions are used for creating an auxiliary SFC that saves the previous state. Figure 4a shows the SFC equivalent for ME that models two users with a shared resource. Transitions t_1 and t_2 are related to assigning the resource to the first and the second users respectively. The resource is released by firing of t_3 and t_4. The auxiliary SFC is depicted in figure 4b.

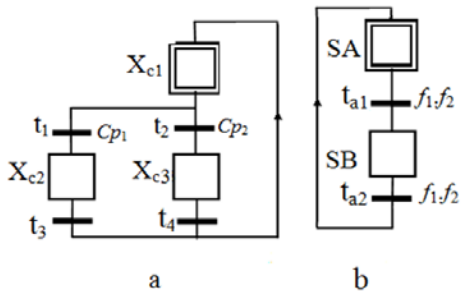

Figure 4. Dynamic priority assignment: (a) Mutual exclusion and (b) auxiliary SFC

In the initial state, step *SA* is enabled. If SEME occurs, it results in the firing of ta1, activation of *SB* and deactivation of *SA*. Using this auxiliary SFC, the transition conditions for obtaining dynamic priority is calculated. In table 1 all the possible states for two ME transitions, considering the previous state are represented. In the first and second columns of table 1, f_i = 1 shows that transition *i* is firable (i.e. transition condition is satisfied and all its input steps are active). In the third column *S* =0 means that SA is active and *S* =1 means that SB is also active. Cp_1 and Cp_2 are new conditions for t_1 and t_2. If both of f_1 and f_2 become true, the transition related to the one that in the previous SEME was not fired; this turn is fired. Simplifying of Cp_1 and Cp_2 columns with Karnaugh map yields final condition for two transitions as relations (1) and (2) for the transitions t_1 and t_2 respectively. Note that it is considered that if *S* equals to one, means that in the previous simultaneous request, the first user with transition t_1 and transition condition Cp_1 is used of the resource. So in the next simultaneous requesting, priority goes to the second user (Cp_2=1).

$$Cp_1 = f_1 f'_2 + f_1 S \quad (1)$$

$$Cp_2 = f_2 f'_1 + f_2 S' \quad (2)$$

Table 1. State table for calculating of conditions

f_1	f_2	S	Cp_1	Cp_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

f_i=1 means transition (i) is fireable

C_i=1 means condition of t_i become true

S : State of auxiliary SFC

That f_i' shows that t_i is not fireable and S is true if state SB is active and S' is true if state SA is active. Using Cp_1 and Cp_2 as new transition conditions for mutually exclusive transitions, dynamic priority is accomplished which solves the problem of SEME. If any application needs constant priority, transition conditions become as relation (3) and (4). t_1 has priority over t_2.

$$Cp_1 = f_1 \quad (3)$$

$$Cp_2 = f_2 f'_1 \quad (4)$$

4.3. *Dynamic priority assignment for three transitions*

In this section, dynamic priority is developed for three mutually exclusive transitions. For three transitions, six priority states are possible. These states are given by equations (5) to (10). For example, in equation (5), t_1 has priority over t_2 and t_3, and t_2 has priority over t_3. For creating these six states of priority, six steps and some transitions as auxiliary SFC are needed. This auxiliary SFC is depicted in Figure 5. In this figure, Xp_1 is initially active and for this step, priority state 1 (ps_1) is considered. So if Xp_1 is active, t_1 has priority over t_2 and t_2; has priority over t_3. If transition t_1 becomes fireable (f_1=1) simultaneously with one or both of other transitions, transition t_1 must be fired and then the state of priority goes to state 2. If only transitions t_2 and t_3 become

fireable simultaneously, according to the priority state 1, transition t_2 must be fired and the priority state goes to state 3. In fact, after firing of a transition that has priority, priority of this transition changes to the lowest level. Based on the state of auxiliary SFC and transition states, an appropriate transition would be fired and the priority changes. Transition condition is calculated using f_i and auxiliary SFC as shown in equations (11) to (13). In these conditions, S_i is as equation (14). Cp_1, Cp_2 and Cp_3 are conditions for t_1, t_2 and t_3, respectively.

$$ps_1 = t_1 > t_2 > t_3 \quad (5)$$

$$ps_2 = t_2 > t_3 > t_1 \quad (6)$$

$$ps_3 = t_1 > t_3 > t_2 \quad (7)$$

$$ps_4 = t_3 > t_1 > t_2 \quad (8)$$

$$ps_5 = t_2 > t_1 > t_3 \quad (9)$$

$$ps_6 = t_3 > t_2 > t_1 \quad (10)$$

$$Cp_1 = f_1 f'_2 f'_3 + f_1 f_2 S_1 + f_1 f_3 S_1 + f_1 f_2 S_3 + f_1 f_3 S_3 + f_1 f_2 f'_3 S_4 + f_1 f'_2 f_3 S_5 \quad (11)$$

$$Cp_2 = f'_1 f_2 f'_3 + f'_1 f_2 f_3 S_1 + f_1 f_2 S_2 + f_2 f_3 S_2 + f_1 f_2 S_5 + f_2 f_3 S_5 + f_1 f_2 f'_3 S_6 \quad (12)$$

$$Cp_3 = f'_1 f'_2 f_3 + f_1 f'_2 f_3 S_2 + f'_1 f_2 f_3 S_3 + f_1 f_3 S_4 + f_2 f_3 S_4 + f_1 f_3 S_6 + f_2 f_3 S_6 \quad (13)$$

$$S_i = Xp_i \,.\, X_{C1} \quad (14)$$

Obviously, by increasing the number of transitions, complexity of the transition conditions and the size of the auxiliary SFC are increased. In fact, for 'n' transition, there are 'n!' priority states. But in practice, usually, only two transitions are mutually exclusive and, the probability of simultaneous occurrence of more than two events is low. In the next section, semi-dynamic priority is developed for more than two ME transition that needs smaller auxiliary SFC and transition conditions becomes simpler.

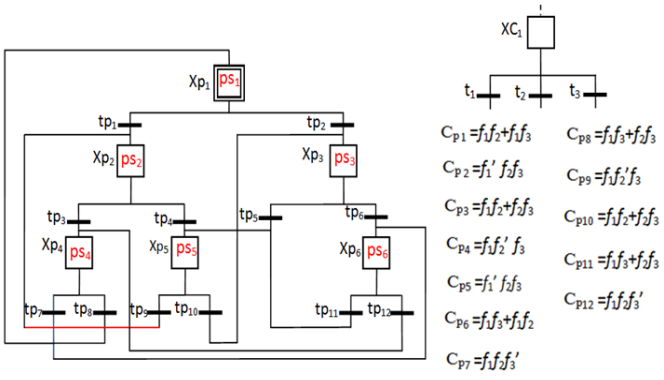

Figure 5. Auxiliary SFC for creating dynamic priority in three ME transitions.

4.4 *Semi-dynamic priority*

For dynamic priority realization in more than two mutually exclusive transitions, conditions become more complex and extend exponentially by increasing the number of transitions. In addition, for creating dynamic priority between n transitions, $n!$ steps and $(n\text{-}1)(n!)$ transitions as auxiliary SFC are necessary. Therefore, realization of dynamic priority for a great number of transitions is not efficient. Instead of creating dynamic priority, it is possible to create semi-dynamic priority that is a composition of the dynamic and constant priorities. Semi-dynamic priority can be realized in several forms.

4.4.1. *Dynamic priority between two transitions* (*DPTT*)

In this method dynamic priority is created for one pair of transitions. If this pair becomes fireable simultaneously among all of transitions, then dynamic priority defines which transition can be fired. Conditions of these two transitions other than the satisfying dynamic priority between this pair, also must satisfy constant priority (say left to right) of the whole transitions in mutual exclusion.

Figure 6a represents a safe place with four output transitions and Figure 6b represents its SFC equivalent. Figure 6c represents SFC equivalent with dynamic priority between second and third transitions. New conditions for transitions are computed as equations (15), (16), (17), and (18). For computing these conditions, a table with all possible states is made and by using this table and some simplifications, Cp_1 to Cp_4 are defined as transition conditions. In this SFC, t_1 has the highest and t_4 has the lowest priority. t_2 and t_3 have constant priority over t_4, and dynamic priority between themselves. For example if t_1, t_2, and t_4 become fireable simultaneously, t_1 is authorized to be fired. If t_2, t_3, and t_4 become fireable simultaneously, t_4 is ignored; and dynamic priority defines whether t_2 or t_3 should be fired.

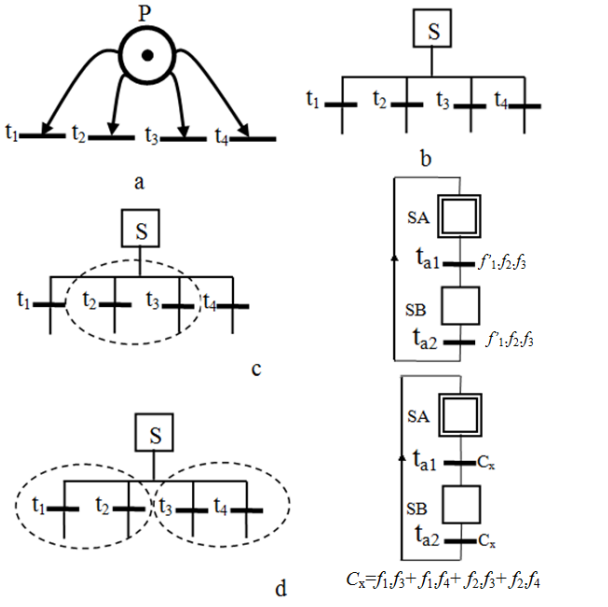

Figure 6. Semi-dynamic priority: (a) place with four output transition and (b) its SFC equivalent, (c) dynamic priority in two transitions (DPTT) and (d) dynamic priority in group of transitions (DPGT).

$$Cp_1 = f_1 \quad (15)$$
$$Cp_2 = f'_1 f_2 f'_3 + f'_1 f_2 S \quad (16)$$
$$Cp_3 = f'_1 f'_2 f_3 + f'_1 f_3 S' \quad (17)$$
$$Cp_4 = f'_1 f'_2 f'_3 f_4 \quad (18)$$

4.4.2. *Dynamic priority between groups of transitions* (*DPGT*)

In DPGT method, all transitions are divided into two groups. If at least one transition of each group becomes fireable, dynamic priority defines which group has priority. Within each group, constant priority is assigned to transitions.

As indicated in Figure 6a and, b, four transitions can be divided into two groups with t_1 and, t_2 in the first group and, t_3 and, t_4 in the second group. In the first group, t_1 has a constant priority over t_2 and in the second group t_3 has a constant priority over t_4. For creating DPGT, a table for all possible states is made and then appropriate conditions for four transitions are calculated. The new conditions are shown by equations (19) to (22). Figure 6d depicts DPGT. If one or two transitions of the first group and one or two transitions of the second group become fireable, the priority goes to the group that in the previous turn did not take the priority. In equations (19) to (22), '*S*' defines this dynamic priority. If '*S*' equals to zero, priority is assigned to the first group and if '*S*' equals to one, the second group has priority. '*S*' changes each turn and therefore, transitions from different groups become fireable simultaneously. For example assume that t_1, t_2, and t_4 become fireable simultaneously and 'S' is zero. As a result, priority is assigned to the first group and t_1 can be fired because it has a constant priority over t_2.

$$Cp_1 = f_1 f'_3 f'_4 + f_1 S' \quad (19)$$
$$Cp_2 = f'_1 f_2 f'_3 f'_4 + f'_1 f_2 S' \quad (20)$$
$$Cp_3 = f'_1 f'_2 f_3 + f_3 S \quad (21)$$
$$Cp_4 = f'_1 f'_2 f'_3 f_4 + f'_3 f_4 S \quad (22)$$

4.4.3. *Compositional semi-dynamic priority*

Both DPTT and DPGT can be used in a compositional form. In this method all transitions can be divided into two groups and each group can contain dynamic priority between a pair of transitions. However, by using this method, transition conditions become complex and this method is only applicable for special cases.

4.5. *Non-safe place conversion*

In this section, an alternative for converting non-safe places (i.e. place that may contain more than one token) to SFC equivalent is proposed. It is started with a simple place and then is continued by places with more than one input and one output transition.

4.5.1. *Simple place*

In the first step it is assumed that a non-safe place has only one input and one output transition. Figure 7a shows such a place that is denoted by P_1. Figure 7b shows SFC equivalent for this place that is constructed by three steps X_{c1}, X_{c2}, X_{c3} and one counter *C*. The value of the counter represents (simulates) the number of tokens in relevant place, P_1. Step X_{c4}, that here is assumed to be safe place is relevant to P_2. Initially, there are two tokens in place P_1 and therefore, the initial value of counter C must be set to two. In PN, firing of t_1 adds one token to P_1 and firing of t_2 removes one token from it. In the SFC equivalent, firing of t_1 forces the counter to increment by one. This is done by an action in X_{c1} (C=C+1). Firing of t_2 causes counter decrements by one. This is done by an action in X_{c3} (C=C-1). As long as C is greater than zero, the output transition must remain enabled. For this reason intermediate step (X_{c2}) has a loop and after firing the output transition and execution of decrement action in auxiliary step, Xc_3, this step becomes active again. This step always remains active even though counter has been equal to zero. In Figure 7b intermediate step is depicted as the initial step because it must be active initially, if initial marking of the relevant place has been non-zero. Conditions for t_1 and t_2 are as Cn_1 and Cn_2 (23, 24). C_i =1 shows that the condition of t_i is satisfied and "e" represents the true condition. Therefore, a transition with event "e" is fired immediately after activating the precedent steps. Because the intermediate step is always active and obviously if the counter becomes zero, the output transition can not be fired. Condition $C \geq 1$ in output transition condition does not allow the transition to be fired as long as counter is equal to zero. Note that two auxiliary steps X_{c1} and X_{c2} cause an scan cycle delay between firing of input transition and reactivating X_{c2} as well as firing of output transition and activating X_{c4}. The delay between firing of the output transition and activation of X_{c4} is equal to one scan cycle time in PLC and in most cases can be ignored.

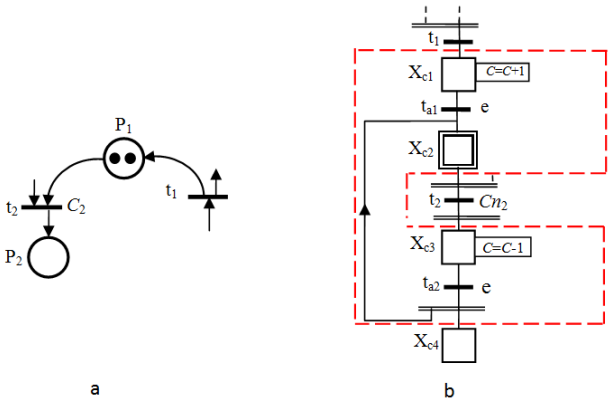

Figure 7. (a) simple non-safe place, (b) it's SFC equivalent, (c) reformed SFC.

$$Cn_1 = C_1 \quad (23)$$
$$Cn_2 = C_2 (C \geq 1) \quad (24)$$

4.5.2. *Conversion with non-simultaneous event*s

Usually, a non-safe place may have more than one input and output transition (Figure 8a). This section covers converting such places to SFC. In the first step, it is assumed that events cannot occur simultaneously. In Figure 8a simultaneous

firing of input transitions t_1 and t_2 and also output transitions t_3 and t_4 is not possible. Figure 8b shows the proposed SFC equivalence for Figure 8a. Transition conditions for output transitions are computed by using equations (25) and (26). For example, suppose that initial value of the counter is two (initial marking of P is two). If the events relevant to t_3 or t_4 occur, because C is equal to two, relevant transition conditions (Cn_3 or Cn_4) become true and transition can be fired. If the counter is equal to zero, besides that X_{c2} is active, output transitions cannot be fired until firing of input transitions causes the counter to become greater than zero. If C=1, because it is assumed that simultaneous events cannot occur, the first output transition becoming fireable can be fired and then the counter becomes zero.

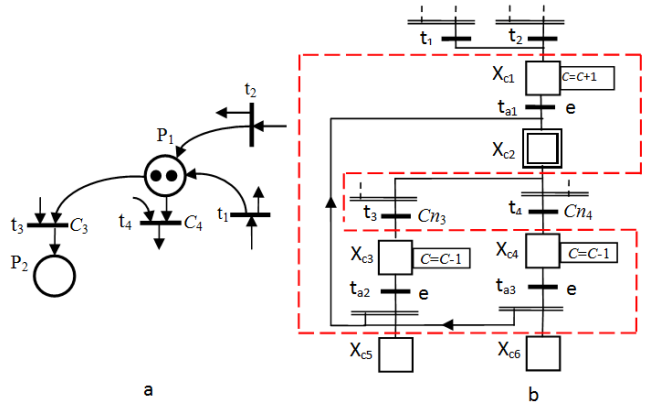

Figure 8. (a) non-safe place with more than one in-out transitions, (b) it's SFC equivalent.

$$Cn_3 = C_3(C \geq 1) \quad (25)$$

$$Cn_4 = C_4(C \geq 1) \quad (26)$$

4.5.3. *Conversion with simultaneous event: constant priority*

Simultaneous events are not possible in SCT while in real systems the occurrence of simultaneous events is possible and a controller connected to a physical system may observe simultaneous input changes. In this section a method for converting a non-safe place to SFC with the possibility of simultaneous firing is proposed. Figure 9a shows conversion of Figure 8a to its SFC equivalent. Input transitions can be fired simultaneously. So two input steps are needed here. For output transitions, a constant priority is assigned which t_3 has priority over t_4. Transition conditions for t_3 and t_4 are given by equations (27) and, (28). If each of t_3 or t_4 becomes fireable, and the counter value equals to 1, the constant priority defines which transition should be fired but if the counter value has been greater than one, both transitions can be fired.

$$Cn_3 = f_3(C \geq 1) \quad (27)$$

$$Cn_4 = f'_3 f_4(C = 1) + f_4(C > 1) \quad (28)$$

Note that in SFC, for mutual exclusion transitions, just one of them can be fired simultaneously. In a situation that the value of the counter is greater than one, if two transitions become fireable simultaneously, only one of them can be fired (according to SFC standard). But in PN, simultaneous firing is possible. If ME is relevant to a resource, it cannot assign to both of the users (relevant place is safe). But if in another situation, simultaneous firing of both transitions is admissible, one transition can be added to the output transition. If the value of the counter is greater than one the condition of this transition becomes true and both transitions become fireable simultaneously. By firing this transition, both output steps and the intermediate step must become active.

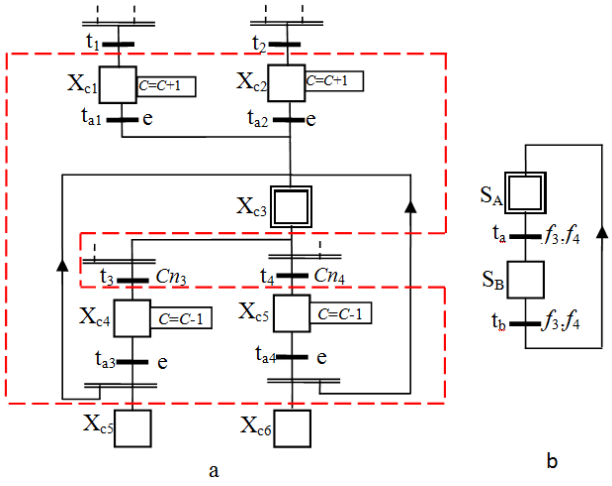

Figure 9. SFC equivalent for non-safe place and non-simultaneous event.

4.5.4. *Conversion with simultaneous events: dynamic priority*

As mentioned earlier, constant priority method is not proper for all situations and dynamic priority method can be used instead that causes equal utilization of the resource in simultaneous requests. In this section, output transition conditions for the non-safe place depicted in Figure 8a is defined in order to create dynamic priority.

Appropriate transition conditions for t_3 and t_4 are calculated according to equations (29) and (30). If counter equals to zero, none of the output transitions can be fired and if the counter value is greater than one, both output transitions can be fired simultaneously. But in the case that the counter is equal to one and both output transitions become fireable, dynamic priority defines which one to fire.

$$Cn_3 = f_3(C > 1) + f_3(C = 1).S' + f_3.f'_4(C = 1) \quad (29)$$

$$Cn_4 = f_4(C > 1) + f_4(C = 1).S + f'_3.f_4(C = 1) \quad (30)$$

4.5.5. *Conversion of two complementary non-safe place*

In some PN models, two complementary non-safe places appear in the model, usually for bounding the non-safe place (Figure 10a). For simplicity in conversion and for avoiding confusion, the SFC equivalent for such places is presented in this section. For these two places, two counters are needed. Transition t_1 is the input of the first place and output of the second place and transition t_2 is the output of the first place and input of the second place. Based on the presented method, the SFC equivalent is shown in Figure 10b.

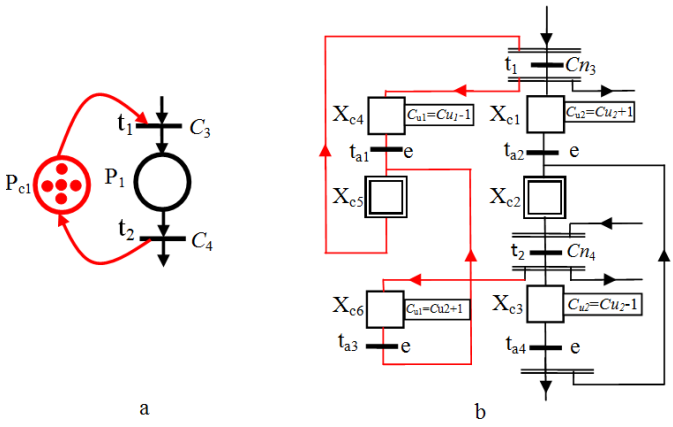

Figure 10. SFC equivalent for two complementary non-safe places.

5. EXAMPLE

In this section, an example of a manufacturing station is considered and controller synthesis for this system with PN-based SCT and implementing it using SFC is discussed. PN-based controller can possess non-safe places and therefore, the model is not safe but is bounded. In the first step, a model for the process is created and then by adding a supervisor, the model is completed as a controlled model. Then this controller is implemented using SFC.

5.1. *Process description and controller synthesizes*

The layout of the system is depicted in Figure 11. In fact, this system is one part of a bigger manufacturing system. It is composed of two machines named as M_1 and M_2, an intermediate buffer, B, and a manipulator robot named R. First of all, the raw material is transferred to M_1 by R and then machining operations are performed on it and the work-piece lies in the output of M_1. Then, the manipulator, R, transfers it to the intermediate buffer. In order to produce the final product, work-pieces are transferred from the intermediate buffer to the input of M_2 and after processing by M_2, the final product exits from the system. Note that it is supposed that input to M_1, is always available.

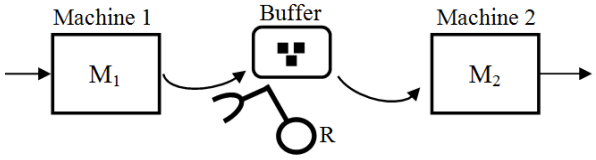

Figure 11. An example of a manufacturing system.

This system is modelled using PN. The uncontrolled model is depicted in Figure 12a.

For achieving the controlled model, supervisor controller is designed and added to the model. In the modelling procedure, it is assumed that the capacity of the intermediate buffer is equal to five and the capacities of the machines are one. Therefore, in each time, a maximum of five work-pieces can exist in the buffer and machines have the ability to process one work-piece at a time. These three specifications can be written as non-equality constraints according to equations (31) to (33). Supervisor controller is designed based on these specifications. Using the method described by Yamalidou et al.(1996), the control places are calculated and synthesized with the uncontrolled model that yields the final controller. Control places are depicted with red colour in Figure 11. The controlled model or the final controller is depicted in Figure 12b. The next step is implementing this controller and this is done with converting the final controller to its SFC equivalent.

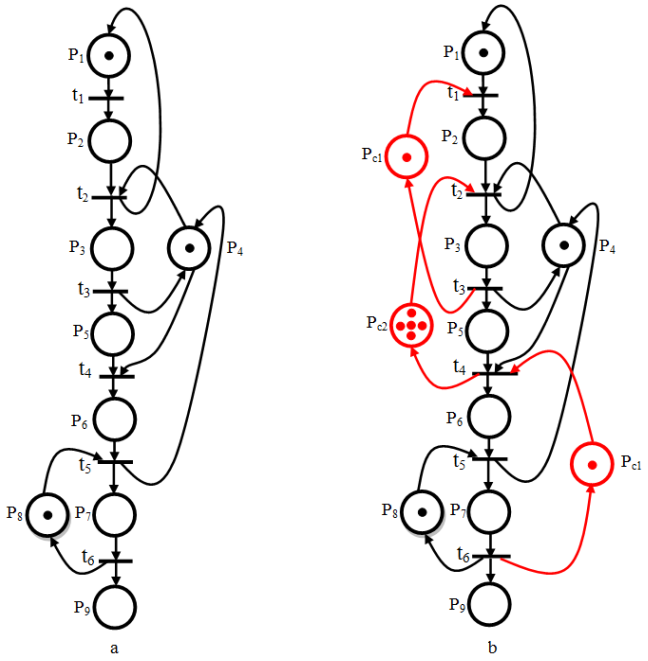

Figure 12. (a) Uncontrolled PN model, (b) controlled PN model for the manufacturing system depicted in figure 11.

For achieving the controlled model, supervisor controller is designed and added to the model. In the modelling procedure, it is assumed that the capacity of the intermediate buffer is equal to five and the capacities of the machines are one. Therefore, in each time, a maximum of five work-pieces can exist in the buffer and machines have the ability to process one work-piece at a time. These three specifications can be written as non-equality constraints according to equations (31) to (33). Supervisor controller is designed based on these specifications. Using the method described by Yamalidou et al.(1996), the control places are calculated and synthesized with the uncontrolled model that yields the final controller. Control places are depicted with red colour in Figure 11. The controlled model or the final controller is depicted in Figure 12b. The next step is implementing this controller and this is done with converting the final controller to its SFC equivalent.

$$m(p_2)+ m(p_3)<=1 \quad (31)$$

$$m(p_3)+ m(p_5)<=5 \quad (32)$$

$$m(p_6)+ m(p_7)<=1 \quad (33)$$

p_1: M_1 is ready.

p_2: M_1 is busy.

p_3: work-piece is transfer with robot to the buffer (robot is busy)

p_4: robot is ready.

p_5: intermediate buffer.

p_6: work-piece is transfer with robot to the M_2 (robot is busy)

p_7: M_2 is busy.

p_8: M_2 is ready.

p_9: output of the system.

5.2. *Controller implementation*

As mentioned earlier, SFC is an ideal choice in controller implementation due to its similarities with PN. In this section, the designed controller for the system that discussed in the earlier section is implemented.

In the final controller, a mutual exclusion is appeared in the model. One transition is for transferring requests from M1 to the intermediate buffer, and another is for transferring requests from intermediate buffer to the M_2. In implementing these ME transitions, the dynamic priority discussed in sub section 4.2 is used.

Also, the controller has two non-safe places; one for modelling the 5-capacity intermediate buffer and the other as a control place for bounding the capacity of this place to five. The method that was discussed in sub section 4.5 is used for implementing these two places to SFC equivalent. In the first step, each place is converted to a step in SFC. This conversion is depicted in Figure 13. Steps with dashed line are relevant to the non-safe places in PN controller.

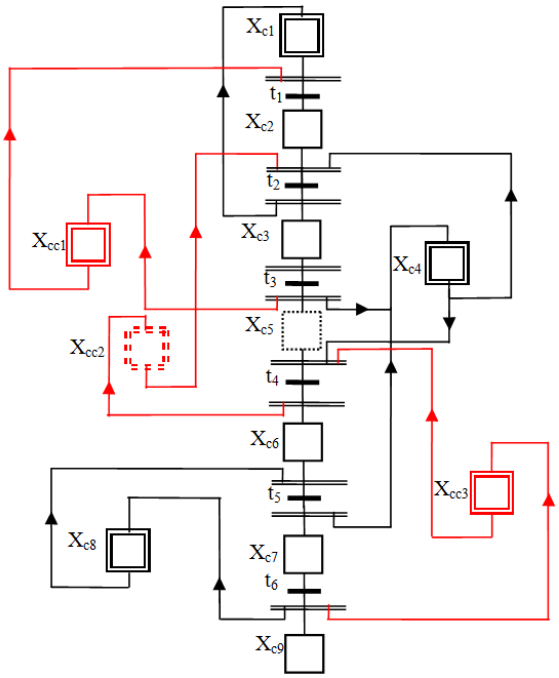

Figure 13. (a) Initial SFC equivalent for the uncontrolled PN model, (b) initial SFC equivalent for the controlled PN model for the manufacturing station.

In the next step, two non-safe places are converted to the SFC equivalent (Figure 14). Also the dynamic priority is assigned to t_2 and t_4 transitions. Equations (34), (35) are related to mutually exclusive transitions, t_2 and t_4, for creating dynamic priority between them.

Steps that are depicted with double line body are equivalent in places that have non-zero initial marking. In SFC, these steps must be activated initially. In the SFC standard, an SFC has only one initial step. By using this initial step, all these steps can be activate.

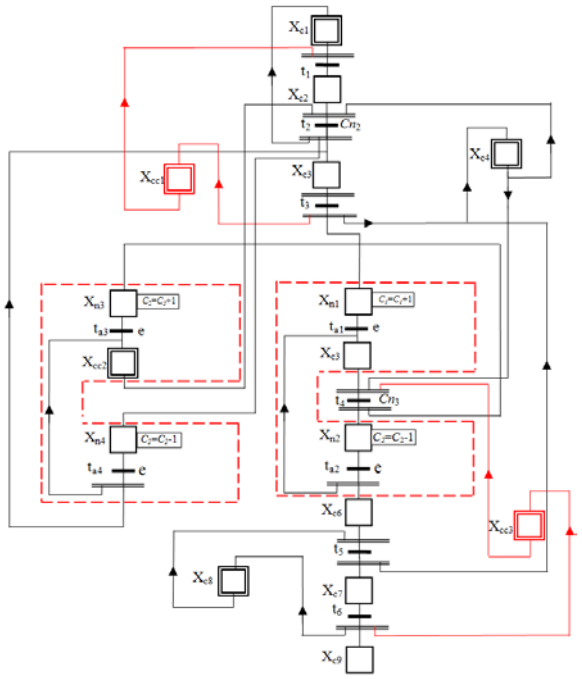

Figure 14. Converting non-safe places in the model to SFC equivalent.

$$Cp_1 = f_2 f'_4 + f_2 S \quad (34)$$

$$Cp_2 = f'_2 f_4 + f_4 S' \quad (35)$$

6. CONCLUSION

Supervisory control theory is a general framework for controller synthesis and PN is a powerful tool for designing a controller in supervisory frameworks. Despite the great acceptance of SCT in controller designing, there are problems with practical implementation. One method for implementation is the conversion of PN controller to SFC standard language. One of the biggest problems in this way is the occurrence of simultaneous events in mutually exclusive transitions or briefly SEME transitions. In previous studies, this problem was solved by assigning constant priority to transitions but in some cases this is not a good method. In this paper, a dynamic priority for two and three transitions and semi-dynamic priority for more than two transitions is proposed. Dynamic priority reforms the resource usage of device. In other words, in the simultaneous requesting of a common resource, resource is shared equally between users.

Another problem in controller implementation is the conversion of non-safe places. In this paper a method to

simulate the non-safe place in SFC is proposed as well. As a result, by using this method, some places of the controller like control places can be non-safe and therefore, the controller can become more efficient. The solution of the discussed problems could redound the usage of SCT in practice.

REFRENCES

Boucher, T.O., Jafari, M.A., and Meredith, G. A. (1989), 'Petri net control of an automated manufacturing cell', *In Proc. 11th Annual Conference on Computers and Industrial Engineering*, pp. 459–463.

Cassandras, C.G., Lafortune, S. (2008), *Introduction to Discrete event systems.* Second edition, Springer Science.

Chirn, J.L, McFarlane, D.C. (2000), 'Petri nets based design of Ladder logic diagrams', *Control 2000, Cambridge*, UK, September.

David, R. (1995), 'Grafcet: A Powerful Tool for Specification of Logic Controllers', *IEEE transactions on control systems technology*, VOL. 3, NO. 3, 253-268.

David, R., Alla, H. (2005), Discrete, Continuous and Hybrid Petri Nets. Springer.

Dideban A., Alla H., (2005), "From Forbidden State to Linear Constraints for the Optimal Supervisory Control", *Control Engineering and applied Informatics* (*CEAI*), 7(3), 48-55.

Dideban, A., Alla, H., (2008), Reduction of Constraints for Controller Synthesis based on Safe Petri Nets. *Automatica*,44(7): 1697-1706.

Ferrarini, L., Piroddi L. (2003), 'Modular design and implementation of a logic control system for a batch processs', *Computers and Chemical Engineering*, Vol. 27, pp. 983-/996.

Giua, A., Dicesare F. (1991), 'Supervisory Design Using Petri Nets', *Proc. of the 30th Conf. on Decision and Control*, pp. 92 - 97, Brighton-England.

Giua, A., Dicesare F., Silva M. (1992), 'Generalized Mutual Exclusion Constraints on Nets with Uncontrollable Transitions', *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics (Chicago, USA)*, pp. 974-799.

Giua, A. (1992), 'Petri nets as discrete event models for Supervisory control', *Doctoral thesis*, Rensselaer Polytechnic Institute, (Troy, New York).

Giua, A. (1996), 'Petri Net Techniques for Supervisory Control of Discrete Event Systems', *Proc. of 1st Int. Workshop on Manufacturing and Petri Nets, Osaka - JAPAN*, pp. 1 – 21.

Hellgren, A., Fabian M., and Lennartson B. (2001), 'Modular Implementation of Discrete Event Systems as Sequential Function Charts applied to an assembly cell', *In Proceedings of the 2001 IEEE Conference on control applications, Mexico City, Mexico.*

Hellgren, A., Fabian, M., and Lennartson B. (2002), 'On the execution of Sequential Function Charts', *Control Engineering Practice*, 13, 1283–1293.

Holloway, L.E., Krogh B.H. (1990), 'Synthesis of Feedback Control Logic for a Class of Controlled Petri Nets', *IEEE Trans. on Aut. Cont.*, vol. 35, no. 5, pp. 514-523.

ISO/IEC. (2001), International standard IEC 61131-3 (2nded). Programmable logic controllers-Part3. ISO/IEC (final draft).

Jackman, J., Linn, R., and Hyde, D. (1995), 'Petri net modeling of relay ladder logic', *Journal of Design & Manufacturing*, Vol. 5, pp. 143–151.

Kumar, R., Holloway, L.E. (1996), 'Supervisory control of deterministic Petri nets with regular specification languages', *IEEE Trans. Automatic Control*, 41(2):245-249.

Lee G.B., Lee J.S. (2000), 'The state equation of Petri net for the LD program', *In Proc. IEEE Int. Conf. Systems, Man, & Cybernetics*, pp. 3051–3056.

Cantarelli M. (2006), 'Control system design using Supervisory Control Theory: from theory to implementation'Universita degli Studi di Cagliari. Master thesis.

Moody, J.O., Antsaklis, P. (2000), 'Petri net supervisors for DES with uncontrollable and unobservable transition', IEEE *Trans. Automatic Control*, 45(3):462-476.

Music, G., Matko, D. (1998), 'Petri net based supervisory control of flexible batch plants', *Preprints of the 8th IFAC/IFORS/IMACS/IFIP Symposium on Large Scale Systems, Rio Patras, Greece*, Vol. 2, pp. 989-994.

Music, G., Matko, D. (1999), 'Petri Net Based Control of a Modular Production System', *Proceedings of the IEEE International Symposium on Industrial Electronics - ISIE '99, Bled, Slovenia*, July 12 - 16, vol. 3, pp. 1383-1388.

Music, G., Matko, D., and Zupancic, B. (2000), 'Modeling, Synthesis, and Simulation of Supervisory Process Control Systems', *Mathematical and Computer Modeling of Dynamical Systems*, Vol. 6 No. 2, pp. 169-189.

Music, G., Gradisar, D., and Matko D. (2005), 'IEC 61131-3 Compliant Control Code Generation from Discrete Event Models', *Proceedings of the 13th Mediterranean Conference on Control and Automation Limassol, Cyprus*, 346-351.

Peng, S., Zhou, M. (2004), 'Ladder Diagram Petri-Net-based discrete event control design methods', IEEE *Transactions on systems, man, and cybernetics, part c: applications and reviews*, Vol. 34, No. 4, November.

Ramadge, P.J., Wonham, W. (1989), 'The Control of Discrete Event Systems', *Proceedings of the IEEE, Special issue on Dynamics of Discrete Event Systems*, Vol. 77, No. 1:81-98.

Uzam, M., Jones, A., and Ajlouni, N. (1996), 'Conversion of Petri nets controllers for manufacturing systems into ladder

logic diagrams', *IEEE Symposium on Emerging Technology and Factory Automation, ETFA.* Vol. 2, pp. 649-655.

Uzam, M., Jones, A. (1998), 'Discrete event control system design using automation Petri nets and their ladder diagram implementation', *International Journal of Advanced Manufacturing Technology*, Vol. 14, No. 10, pp. 716–728.

Uzam, M., Jones, A.H. (2002), 'A new Petri-Net-based synthesis technique for supervisory control of discrete event systems', *Turk J Elec Engin*, VOL.10, NO.1 pp.85-99.

Yamalidou, K., Moody, J., Lemmon, M. and Antsaklis, P. (1996), 'Feedback control of Petri Nets based on place invariants', *Automatica*, 32(1):15-28.

Zhou, M., Twiss, E. (1998), 'Design of Industrial Automated Systems via Relay Ladder Logic Programming and Petri Nets', *IEEE transactions on Systems, Man, and Cybernetics - part C: application and reviews*, vol. 28, NO. 1, February.

Zhou, M.C., DiCesare, F., and Rudolph D. L. (1992), 'Design and Implementation of a Petri Net Based Supervisor for a Flexible Manufacturing System', *Automatica*, No. 28, pp. 1199-1208.