COMPUTER AIDED DESIGN OF SUPERVISORY CONTROLLER FOR DISCRETE EVENT SYSTEMS

A.H. Jones and Y.M. Li

Intelligent Control Group, School of Aeronautical & mechanical Engineering,
University of Salford, Manchester M5 4WT, UK
E-mail: y.m.li@pgr.salford.ac.uk a.h.jones@salford.ac.uk

Abstract: This paper presents a formal method of designing compiled supervisors for Discrete Event systems (DES). The method is based on the concept of supervisory control theory and the enforcement of a control policy by modifying the Petri net model in a compiled form. In this paper the concept of Boolean logic reduction is used to simplify the control policy. This process re-defines the control policy as steering places which then facilitates the implementation of the control policy in the form of inhibitor and enabling arcs. The method provides a very efficient technique for the development of maximally permissive compiled supervisors for Petri nets based DES. *Copyright © 2000 IFAC*

Keywords: Computer aided design, Petri nets, Supervisory Controller, Boolean Logic reduction, Discrete Event Systems.

1. INTRODUCTION

In the supervisory control theory, developed by Ramadge and Wonham (Ramadge, et al., 1987a, 1987b), a unifying framework for the control of Discrete Event Systems is provided. The supervisory control theory provides researchers and application engineers a foundation to fully understand all the issues involved in the design of DES.

One of the principal control problem in DES involves defining a set of forbidden states, and designing a supervisor which ensures that none of these forbidden states are entered, whilst at the same time ensuring the remain behavior of the system is maximally permissive within a required behavior region. The supervisor is the agent that specifies which events are to be enabled and disabled. One approach is to enforce the model of the system to remain within the set of required behavior by only enabling transitions which keep the system inside the required behavior.

The key concepts in the design of supervisory controllers for DES are as following:

- There are two types of events that may occur in the DES, namely controllable events, that may be controlled by control actions, and uncontrolled events, that may not be controlled by control actions.
- Given a model of a DES and a specified required behavior of the controlled system, the objective is to synthesize a supervisor and a supervisory control policy to realize the specified controlled behavior.
- The controlled behavior of the DES must be nonblocking, i.e., it must not contradict the specifications given.
- The controlled behavior of the DES must be maximally permissive within the specifications given.

In general, a design and implementation method for DES controller must provide tools for the system specification, modeling, analysis and implementation. In this paper, Petri nets are used as the basic modeling tool because they have a greater modeling power than other techniques (Giua,1996). Petri nets offer a structured model of DES that can be exploited in developing more efficient algorithms for controller synthesis.

One approach to the design of a compiled supervisor is to formulate the control policy such that it is added to the original Petri net to form the supervisor. Such controllers have been defined by Yamalidou et al

(Yamalidou, et al., 1996) using place invariant. However, such an approach adds places to the original Petri net and is not always maximally permissive. The design method presented in this paper also makes use of the original Petri net to form the controller, however, in this case only enabling arcs and inhibiting arcs are added to original Petri net to form the supervisor. Moreover, the supervisor can be guaranteed to be maximally permissive in all cases.

In this paper, a new method is presented for the synthesis of supervisory controllers modeled by Petri nets. The key features of the new technique are the **E**nabling an**D** **I**nhibiting of **C**ontrollable **T**ransitions to effect supervisory control, The word *edict* is defined as a set of instructions which have to be obeyed. This statement has close parallels to the supervisory controller thus as it enforces a control policy which only permits the required behavior to occur.

2. THE MODELING TOOL OF DES

2.1 Petri Nets basics

Petri net may be identified as a particular kind of bipartite directed graph populated by three types of objects(Peterson, 1981). These objects are **places**, **transitions** and **directed arcs** connecting places to transitions and transitions to places. Directed arcs (arrows) connect the places and the transitions, with some arcs directed from the places to the transitions and vice versa. It is necessary for an arc to have a node, i.e., a place or a transition, at each of its end.

A **marking** M is an assignment of to the places of a Petri net. A **token** is a primitive concept for Petri nets (like places and transitions). Tokens are assigned to, and can be thought to reside in, the places of a Petri net. The number and position of tokens may change during the execution of a Petri net. The tokens are used to define the execution of a Petri net. Tokens reside in places and wtravel along arcs and their flow through the net is controlled by transitions. They are represented by dots. The marking M(p) of a Petri net is a mapping of each place to a non-negative integer.

2.2 Extended Petri nets

The modeling power of ordinary Petri nets has been found too limited and there has been an ongoing program to extend the modeling power and several extensions have been made to the ordinary Petri net framework. Indeed, in order to design supervisory controllers, the ability to block transitions is very important. Within this category inhibitor arcs and enabling arcs have been defined (David, et al.,1994. Jones, et al., 1996). They offer very powerful features in this regard (as shown in Fig. 1 and Fig. 2).

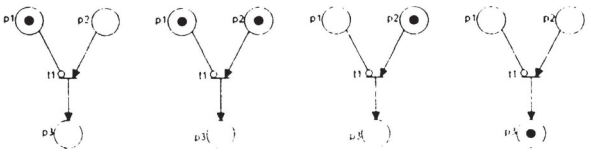

Fig. 1 Inhibitor arc Petri net. (a) Not enabled. (b) Not enabled. (c) Enabled(before firing). (d) Enabled(after firing).

The modeling power of Petri nets can be increased by adding the '*zero testing*' ability, i.e., the ability to test whether a place has no token. This is achieved by introducing an *inhibitor arc*. As shown in Fig.1, the inhibitor arc connects an input place to a transition and is represented by an arc whose end is marked by a small circle. The presence of an inhibitor arc means that the transition is only enabled if the input place does not have any tokens. Firing of a transition does not change the marking of a place, which is connected to the transition with an inhibitor arc. The Petri net is not enabled in Fig.1.(a), because $M(p_1) = 1$ and $M(p_2) = 0$, similarly in Fig.1.(b), transition t_1 is not enabled , because $M(p_1) = 1$, although $M(p_2) = 1$. However , the Petri net in Fig.1.(c) is enabled, because $M(p_1) = 0$ and $M(p_2) = 1$. When transition t_1 fires, it removes one token from place p_2 and deposits one token into place p_3 (shown in Fig.1.(d)). Note that after the firing of transition t_1 , the marking of the place p_1 remains the same.

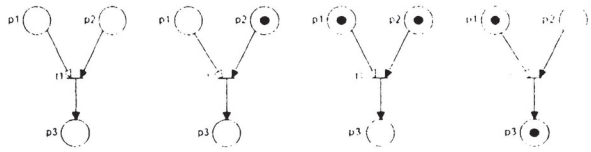

Fig. 2 Enabling arc Petri net. (a) Not enabled. (b) Not enabled. (c) Enabled(before firing. (d) Enabled (after firing)

The modeling power of Petri net can also be increased by adding the '*one testing*' ability, i.e., the ability to test whether a place has a token. This is achieved by introducing an *enabling arc*. As shown in Fig.2, the enabling arc connects an input place to a transition and is represented by an arc whose end is marked by an empty arrow. The presence of an enabling arc means that the transition is only enabled if the input place has a token. The firing does not change the marking in the enabling arc connected place. The Petri net is not enabled in Fig.2.(a), because $M(p_1) = 0$ and $M(p_2) = 0$, similarly in Fig.2.(b), transition t_1 is not enabled , because $M(p_1) = 0$, although $M(p_2) = 1$. However , the Petri net in Fig.2.(c) is enabled, because $M(p_1) = 1$ and $M(p_2) = 1$. When transition t_1 fires, it removes one token from place p_2 and deposits one token into place p_3 (shown in Fig.2.(d)). Note that after the firing of

transition t_l , the marking of the place p_l remains the same.

The following rules are used to govern the flow of tokens:

- Enabling Rule:

A transition t_j is said to be *enabled* if each place in $Pre(p_i,t_j)$ contains at least the number of tokens equal to the weight of the directed arc connecting p_i to t_j .

i.e., $\forall P_i \in Pre(p_i, t_j) \quad \exists M(P_i) \geq Pre(p_i, t_j)$

An enable transition t_j may or may not fire depending on the additional interpretation.

- Firing Rule:

A transition t_j is *fireable* if its associated interpretation is satisfied. A fireable transition never executes until the enabling rule is satisfied.

3. SUPERVISORY CONTROL

3.1 Previous work

The supervisory control theory was introduced just more than 10 years ago by Ramadge and Wonham, it is based on finite state machines (FSM) and formal language concepts. In their work, FSM were used to model the DES and specifications. Although FSM provide a general framework for establishing fundamental properties of DES control problems, there are some disadvantages in using FSM. The principal disadvantage is that a meaningful graphical representation is difficult to achieve using FSM.

Petri-nets-based solutions, can offer a more compact description and can be represented graphically in an easily understood format. The models can be used for analysis of their properties, performance evaluation and systematic construction of the discrete event supervisors. Because of these advantages, Petri nets have gained in popularity as an alternative framework for the design of supervisory controllers for DES (Krogh, et al., 1991, Giua, et al., 1991, Sreenivas, et al., 1992).

In the forbidden state problem, the control specifications are expressed as forbidden conditions. Forbidden conditions are a compact way of defining classes of undesirable markings which should be avoided (Holloway, et al., 1996).

Li and Wonham proposed the *linear integer programming* approach (Li, et al., 1993, 1996) and Krogh and Holloway proposed *path-based* approach(Krogh, et al.,1991), both of these two approaches represent the control strategy that must be computed off-line and on-line. On the other hand, a *place invariant* approach by Moody et al.[4] and *extended place invariant* approach by Karimzadgan and Jones (Karimzadgan, et al., 1997) capture the desired control requirements in a Petri net form. All these methods provide practical means of theoretical designed supervisory control system for DES of medium to large size.

It is evident that since 1987 there has been considerable success in deploying a theoretical basis for the design of controllers in DES. However, to-date there is no general methodology for design maximally permissive compiled supervisory Petri net.

3.2 EDICT supervisory controller design method

The design method presented in this paper makes use of the original Petri net to form the controller as Moody et al did using place invariant. However, in this case only enabling arcs and inhibitor arcs are added to original Petri net to form the supervisor and the supervisor can be guaranteed to be maximally permissive in all cases.

The method to design a supervisory controller for a discrete event system such that the system does not enter any forbidden markings and is maximally permissive is to take the initial marking of the discrete event system and generate the next set of possible markings from the model by firing either controllable or uncontrollable transitions one at a time (sequential solution). If a bad marking is found during this process, it is blocked from further expansion. This process generates the blocked reachability graph (BRG) of the discrete event system. Once this has been done the blocked reachability graph can be scanned for forbidden markings. Each forbidden marking can then be traced back through the reachability graph until a controllable transition is found. This controllable transition can then be used to block the forbidden marking and as such formulates one part of the control policy. The complete control policy can be obtained by back tracking all of the forbidden markings to controllable transitions and then be refined by considering each controllable transition in turn and collecting together all of the markings which require the transition to be blocked and all of the markings which require the transition to be enabled.

By collecting together two sets namely the set to be blocked and the set to be enabled it is possible to simplify the control policy for each transition by using Boolean logic reduction. Boolean logic reduction techniques sub divide the problem into three distinct types namely blocking, enabling and don't care. One of the most powerful Boolean logic reduction techniques is **Espresso** which was developed by Brayton et al

(Brayton, et al., 1984). Moreover, Espresso can deal with multi-valued logic which thus encompass multi-token situations. The Espresso technique provides a near optimal minimal logic solution to large scale Boolean logic reduction problem. Once these logical expressions have been obtained they represent minimal logic solution of the control policy for each controllable transition.

The control policy is in the form of a bit pattern which will contain **1**'s, **0**'s or _ . A '1' from a place implies its presence enables the transition, a '0' from a place implies its absence enables the transition and a '_' represents a don't care in the sense that place has no influence on the transition. The places which contain a '1' or '0' form a sub set of steering places which individually influence each controllable transition, the control policy is a unique set of bit patterns at the appropriate steering places which will enable a particular controllable transition.

To enable a controllable transition from one of the steering places with a logic value **1**, i.e., a token is present at the steering place, an enabling arc is used; To enable a controllable transition from one of the steering places with a logic value **0**, i.e., a token is not present at the steering place, an inhibiting arc is used.

The advantage of implementing the control policy using enabling and inhibiting arcs from the steering places is that this results in the controllable transition being enabled from the steering places in accordance with the control policy, and it ensures that the tokens at the steering places are unaffected by the control policy when the transition is fired.

This features a very important as it ensures that the addition of the control policy in the form of enabling and inhibiting arcs ensures that the control policy is implemented in a manner such that the maximally permissive behavior is enabled and as a consequence the forbidden states are blocked.

The EDICT design method thus involves the following five steps:

1. Designing the uncontrolled model of the system using Petri nets;
2. Generate the blocked reachability graph of the unsupervised system;
3. Define the control policy for each controllable transition;
4. Apply the Espresso minimization technique to each controllable transition within the control policy to obtain the steering places.
5. Generate the supervisory controller by adding enabling and inhibiting arcs from the steering places to the controllable transitions of the Petri net model.

4. ILLUSTRATIVE EXAMPLE

To illustrate the application of the method, we consider a mutual exclusion example previously studied by Giua.

Step 1. Design the uncontrolled model of the system using Petri nets

Petri net model of Fig. 3 is for the example of partially controllable net. It is consist of 7 places and 6 transitions. The initial marking is:

$$M_1 = (2\ 2\ 0\ 0\ 0\ 1\ 0)^T$$

It is assumed that controllable transitions are to be t_1, t_2 and t_5, and the uncontrollable transitions are to be t_3, t_4 and t_6. The forbidden state is: $M(p_5) + M(p_7) <= 1$, that means both places p_5 and p_7 can not contain token(s) simultaneously. Using Yamalidou's place invariant method, when the transitions of the model are fully controllable, it is maximally permissive, but it is not maximally permissive if the transitions of the model are partially controllable.

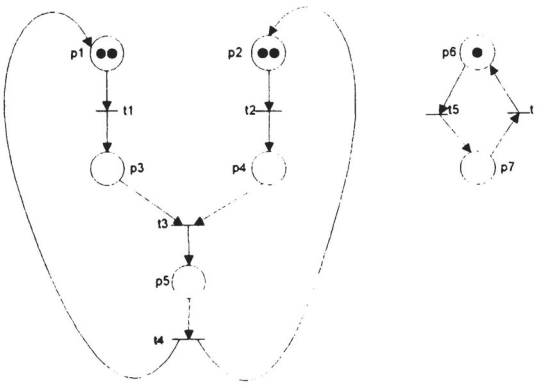

Fig. 3 PN model of Partially controllable system

Step 2. Generate the blocked reachability graph of the model

The Blocked Reachability Graph of the Petri net model, in which each node represents a marking reachable from the initial marking M1 through the sequential transitions and each arc represents the firing of transition, is generated in Fig. 4. There are twenty four markings in the BRG. Each node in the graph stands for a unique state of the net, the seven digit number in each node shows the token content of place 1 to 7 respectively. The directed arcs show the possible transition of state in the net.

The BRG has been developed by checking if the current marking is a forbidden state. If the marking is a forbidden state it is blocked and no further expansion from that marking takes place.

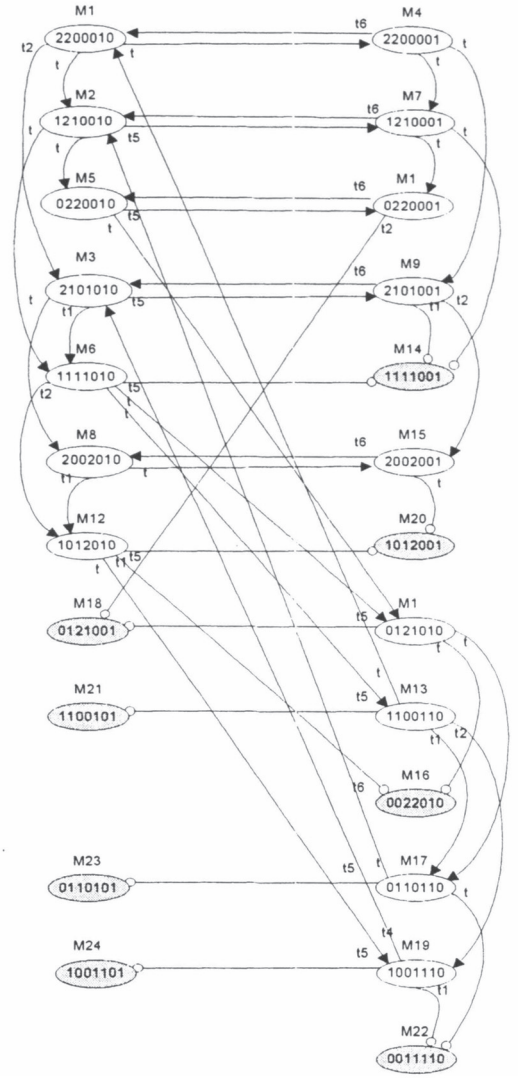

Fig. 4 The Blocked Reachability Graph of the Model

In the partially controllable Petri net not only the actual forbidden states should be avoided but also all other states from which a forbidden state can be uncontrollably reached must be avoided. These states are called Co-forbidden states.

Step 3. Search the BRG to define the control policy for each controllable transition

For each controllable transition, the markings are configured in the form of either good-to-good (arrow arcs of Fig.4) or good-to-bad (circle arcs of Fig.4). Good-to-good represents markings of required behavior and good-to-bad represents markings from the required behavior to a forbidden state. The control policy is defined by good-to-good markings being enabled and good-to-bad markings being blocked as shown in the Table 1.

Step 4. Apply the Espresso minimization technique to each controllable transition within the control policy

For each transition, each of the good-to-good and good-to-bad markings are passed to the Espresso algorithm. In this case the controllable transition t_1 , t_2 and t_5 result in any enabling or blocking behavior.

Table 1. The control policy of the problem

Markings	Trans.	Blo.	Ena.	Input Data of Espresso
M_1=(2200010)	t_1,t_2,t_5	-,-,-	√,√,√	2200010 -> 1
M_2 =(1210010)	t_1,t_2,t_5	-,-,-	√,√,√	1210010 -> 1
M_3 =(2101010)	t_1,t_2,t_5	-,-,-	√,√,√	2101010 -> 1
M_4 =(2200001)	t_1,t_2	-,-	√,√	2200001 -> 1
M_5 =(0220010)	t_2,t_5	-,-	√,√	0220010 -> 1
M_6 =(1111010)	t_1,t_2	-,-,	√,√,	1111010 -> 1
M_6 =(1111010)	t_5	√	-	1111010 -> 0
M_7 =(1210001)	t_1	-	√	1210001 -> 1
M_7 =(1210001)	t_2	√	-	1210001 -> 0
M_8 =(2002010)	t_1,t_5	-,-	√,√	2002010 -> 1
M_9=(2101001)	t_1	√	-	2101001 -> 0
M_9 =(2101001)	t_2	-	√	2101001 -> 1
M_{10}=(0121010)	t_2,t_5	√,√	-,-	0121010 -> 0
M_{11}=(0220001)	t_2	√	-	0220001 -> 0
M_{12}=(1012010)	t_1,t_5	√,√	-,-	1012010 -> 0
M_{13}=(1100110)	t_1,t_2	-,-	√,√	1100110 -> 1
M_{13}=(1100110)	t_5	√	-	1100110 -> 0
M_{15}=(2002001)	t_1	√	-	2002001 -> 0
M_{17}=(0110110)	t_2,t_5	√,√	-,-	0110110 -> 0
M_{19}=(1001110)	t_1,t_5	√,√	-,-	1001110 -> 0

In applying the Espresso logic minimization algorithm it is evident from the markings that places p_1 , p_2 , p_3 and p_4 will have to be represented as multi-valued logic states. As all four places have up to two tokens in the control policy. Where two-level multi-valued logic is used. Thus, in this case Brayton's definition of multi-valued logic is:

0 0 1 ≡ 0 token
0 1 0 ≡ 1 token
1 0 0 ≡ 2 tokens
1 1 1 ≡ don't care

Thus, on applying the Espresso algorithm to transition t_1 , the following set of steering places are obtained.

p_1	p_2	p_3	p_4	p_5	p_6	p_7	
111	111	111	001	–	–	–	enable t_1
111	010	111	111	–	–	0	
111	111	001	111	0	–	0	

For t_2, the following set of steering places are obtained.

$$\left.\begin{array}{lllllll} 111 & 111 & 111 & 001 & 0 & - & 0 \\ 111 & 111 & 001 & 111 & - & - & - \\ 010 & 111 & 111 & 111 & - & - & 0 \end{array}\right\} \text{ enable } t_2$$

For t_5, the following set of steering places are obtained.

$$\left.\begin{array}{lllllll} 111 & 111 & 111 & 001 & 0 & - & - \\ 111 & 111 & 001 & 111 & 0 & - & - \end{array}\right\} \text{ enable } t_5$$

Step 5. Generate the supervisory controller by adding enabling and inhibiting arcs to the Petri nets model.

The Espresso method searches a set of steering places for each controllable transition. In this case there are 3 controllable transitions t_1, t_2 and t_5. The control policy is represented by adding two extra controllable transitions for both t_1 and t_2. and one extra transition for t_5. Using the steering places results from the Espresso algorithm, the influence of each steering place is added to the original Petri net by using either an enabling or a inhibiting arc to the appropriate controllable transition. Fig.5 shows the supervisory controlled Petri net. This represents the first maximally permissive compiled supervisory solution to this problem.

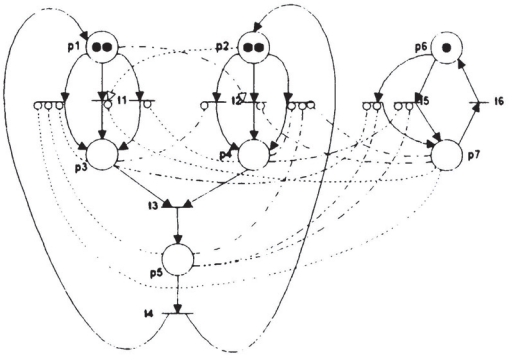

Fig. 5 The controlled Petri net model (supervisor) of Fig. 3

5. CONCLUSION

This paper presents a new method for the design of supervisory controllers of Discrete event system. The design method involves using Espresso Boolean logic reduction algorithm to re-define the control policy as set of steering places which enable controllable transitions and implementing the steering place control policy in the form of inhibitor and enabling arcs to the appropriate controllable transition. The resulting method known as EDICT can thus provide maximally permissive compiled supervisory solution to the forbidden state problem.

REFERENCES

Ramadge, P.J. and W.M. Wonham (1987a). *Supervisory Control of Discrete Event Proces*, SIAM Journal on Control and Optimization, Vol.25, No.1,pp.206~230, Jan.

Ramadge, P.J. and W.M. Wonham (1987b). *Modular Feedback Logic for Discrete Event Systems*, SIAM Journal on Control and Optimization, Vol.25, No.5, pp.1202~1218, Sept.

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, June.

Yamalidou, E., J.O. Moody and M.D. Lemmon (1996). *Feedback Control of Petri Net Based on Place Invariants*, Automatica 32(1), pp.15-28.

Peterson, J.L (1981). *Petri Net Theory and The Modeling of Systems*, Prentice Hall, Englewood Cliffs, N.J.

David, R. and H.Alla (1994). *Petri Nets for Modeling of Dynamic Systems, A Survey*, Automatica 30(2). pp.175~202.

Jones, A.H. and M.Uzam (1996). *Design of Knowledge Based Sequential Control System*, Proc. of the World Automation Congress (WAC'99), Montpelier, France pp.343~348.

Krogh, B.H. and L.E.Holloway (1991). *Synthesis of Feedback Control Logic for Discrete Event Systems*, Automatica, 27(4), pp.641~645.

Guia, A. and DiCesare (1991). *Supervisory Design Using Petri Nets*, Proc. of the 30th Conf. on Decision and Control, pp.92~97, Brighton.

Sreenivas, R.S. and B.H.Krogh (1992). *On Petri Net Model of Infinite State Supervisors*, IEEE Trans. on Automatic Control, Vol. 37, No.2, pp.247-277.

Holloway, L.E., X.Guan and L.Zhang (1996). *A Generalization of State Avoidance Policies for Controlled Petri Nets*, IEEE Trans. on Automatic Control, Vol. 41, No.6, pp.804-816.

Li, Y. and W.M.Wonham (1993). *Control of Vector Discrete Event Systems, I - The Base Model*, IEEE Trans. on Automatic Control, Vol. 38, No.8, pp.1214-1227.

Li, Y. and W.M.Wonham (1994). *Control of Vector Discrete Event Systems, II - Controller Synthesis*, IEEE Trans. on Automatic Control, Vol. 39, No.3, pp.512-531.

Karimzadgan, D. and A.H.Jones (1997). *Design of Discrete Event Control System for IEC1131-3 Compatible PLCs*. IFAC Workshop on Manufacturing Systems: MIM'97,Vienna, Feb 3-5, pp.105-110.

Brayton R.K., G.D. Hachtel, C.T. McMullen and A.L. Sangiovani-Vincentelli (1984). *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers.