A&QT-R 2002 (THETA 13)
2002 IEEE-TTTC International Conference on Automation, Quality and Testing, Robotics
May 23-25, 2002, Cluj-Napoca, Romania

DISCRETE – EVENT CONTROL OF RAILWAY SYSTEMS

Cãlin CIUFUDEAN **Adrian GRAUR** **George MAHALU**

calin@eed.usv.ro *adriang@eed.usv.ro* *mahalu@eed.usv.ro*
"ªtefan cel Mare" University of Suceava
str. Universitãþii nr.1 RO-5800 Suceava

Abstract. Petri nets are emerging as a important tool to provide an integrated solution for modelling, analysis, simulation and control of transport automated systems. This paper identifies certain criteria to compare LLD's and PN's in designing sequence controllers and responding to the changing control requirements in railway systems.

Keywords: Petri nets, Ladder Logic Diagrams, controllers, railway.

1. INTRODUCTION

A railway transport system consists of several concurrent units such as railway engines, wagon, automated guided vehicles, logic controllers, which function asynchronously to meet the dynamically changing needs of the transport plan. Methods aiming for modelling, analysis, control and simulation of such systems are important. The typical functions of the control are to sequence the operations, monitor the system functioning, and determine the states of different elements in a transport system in respect to real time. Traditionally, ladder logic diagrams (LLD's) are used to capture the sequence of operations executed by the system's control.
Realizing the potential of PN's for control, in France, GRAFCET – a PN – like representation tool is proposed as a standard specification of sequence controllers [4].
The international standard versions are called sequential function carts [2]. More details and advantages of PN's for controlling manufacturing system can be found in [3]. However, the comparison between LLD's and PN's for design of sequence controllers is not encountered in the earlier studies.
The sequence controller in this paper means a class of discrete event controllers without choices in executing the operations.

2. LLD'S AND PN'S AND THEIR COMPARISON CRITERIA

Two of the important factors for comparison of PN and LLD for discrete event control are identified as design complexity and response time as described below. Design complexity is defined as the complexity associated in designing the control logic for a given specification. The number of nodes and links for a given graphical control logic design mainly determines graphical complexity. The important factor that influences graphical complexity and adaptability is the physical appearance (size) of the model, whereas not only the physical appearance but also the method of implementation

influences the response time. Method of implementation constitutes the software and hardware used to control the system using either PN's or LLD's. Graphical complexity and adaptability cannot be quantified, whereas response time can be measured accurately, given a logic design and implementation. The number of nodes and links used in a control logic model gives an idea about the graphical complexity, adaptability, and response time. For PN's the nodes are places and transitions and links are areas; whereas in LLD's, nodes are normally operations, timers, relays, and the links are connections. If more nodes and links are used in a design, it is graphically more complex and thus may need more response time. In a similar manner, control logic is more adaptable if it needs fewer changes in the number of nodes and links compared to another logic to meet a change in the specification. Hence, this study uses the number of nodes and links in LLD and PN as a measure to compare their design complexity and response time.

3. PETRI NET BASED CONTROLLERS

Earlier studies use a variety of places to model timers and counters [4], [5]. This might make the model difficult to understand. In our paper, neither new places nor transitions are introduced for modeling them. Timers are modeled by assigning attributes to transitions and counters by places with initial markings and weights on certain arcs. In [3] new sets of places are introduced for modelling I/O signals, which increases the number of places in the PN model. In our paper, I/O signals are modelled as attributes for places and transitions respectively. Hence, due to the use of attributes, our TPN's have fewer nodes and links compared to PN's in [1], [2], thereby reducing the graphical complexity. Specifically, the input signals are put in a vector (X) that reads the state of the input signals from digital input interface X, associates attributes to every place. $X_i = X(p_i)$ and is an attribute associated with place p_i and represents the input channel number associated with p_i. For example, if p_i models a limit switch, the TPN reads the status of that switch from the digital input interface through the channel number represented by X_i. The initial marking, $m_0(p_i)$ is considered as the first attribute of p_i and X_i is the second one. The contents of any input channel X_i are either 0 or 1. Output signal vector (Y) is intended to send output signals through digital output interface. For example, t_i may model the activity "activate a permissive color to signal A". Writing a specific number on to the digital output interface activates each color of a signal. During the execution of the program, when a transition fires, TPN writes the decimal number corresponding to the output channel to digital output interface. The contents of any output channel are either 0 or 1. There are two events for a transition firing, start firing and end firing. Between these the firing is in progress. The removal of tokens from a transition's input place(s) occurs at start firing. The deposition of tokens to a transition's output place(s) occurs at end firing. While transition firing is in progress, the time to end firing, called the remaining firing time, decreases from firing duration to zero at which its firing is completed. The execution rules of a TPN include enabling and firing rules:

1. A transition $t \in T$ (T is a finite set of transitions) is enabled if $f \forall p \in P$ and $I(p,t) \neq 0$, $m(p) \geq I(p,t)$ and X(p) has content 1, where P is a finite set of places, $I : PXT \rightarrow N$ is an input function that defines the set of directed arcs from P to T, where N={0,1,2,...,}, $m : P \rightarrow N$, is a marking whose i^{th} component represents the number of tokens in the i^{th} place. An initial marking is denoted by m_0.
2. Enabled in a marking m, t fires and results in a new marking m' following the rule:

$m'(p)=m(p)+O(p,t)-I(p,t), \ \forall p \in P$, where, $O:PXT \rightarrow N$, is an output function that defines the set of directed arcs from T to P.

The design procedure for formulating a TPN based controller is briefed in the following five steps:

1. Model the control sequence using PN's to obtain the PN model of the sequence controller.
2. Assign input channels to inputs of the system such as limit switches, sensors, etc. to formulate an input-mapping table.
3. Assign output channels to outputs of the system such as relays, switches, etc. Also, identify timing information for activities to obtain an output-mapping table.
4. In the input-mapping table, assign an input channel number to each place in the PN based controller. The initial state of the system decides the initial marking of TPN. In the PN model some places do not represent the inputs of the system as they represent the intermediate state of the system or logical places to model counters in the sequence. No channel has to be assigned to these places, represented by "-".
5. Using the output mapping table and the action(s) that are modelled by a transition, assign a number to each transition in a PN based controller. The operations and the time delays given in the sequence to be controlled decides firing time function of TPN.

By following the above procedure, a TPN based controller can be formulated, as it is illustrated in Fig.1, for a given sequence.

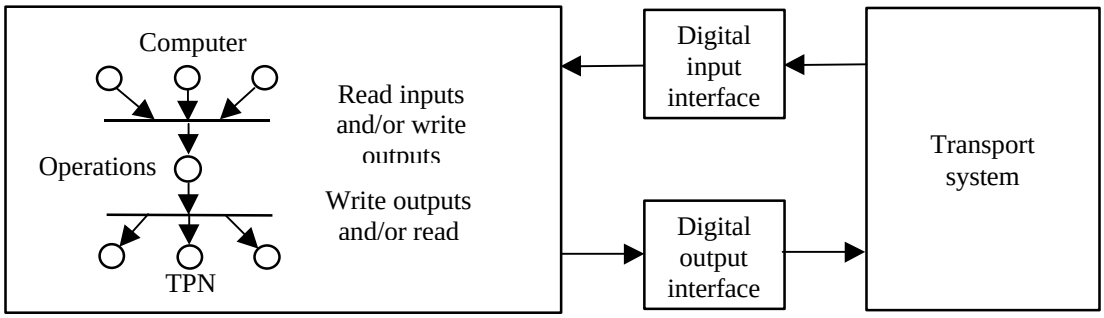

Fig.1 A TPN based controller

4. COMPARISON OF LLD'S AND PN'S FOR A RAILWAY SYSTEM

One effective way to perform the comparison between LLD's and TPN's is through a railway-automated system. The system considered in this paper consists of four automated guided vehicles, agv, (A,B,C and D) utilised for marshalling in order to form a convoy. Each agv has two normally open limit switches (physically these limit switches are isolated railway sections). For example, when agv A contacts limit switch $a_0(a_1)$, it indicates that the agv A is at the and of its return stroke (forward stroke). Three push buttons are provided to start the system (switch SW1), to stop the system normally (switch SW2) and to stop the system immediately in emergency (switch SW3, ES). The system has 11 inputs corresponding to 8 limit switches (two for each agv) and 3 push buttons. The system has 6 outputs corresponding to 4 relays that give the senses of the agv's strokes and two lights that indicate the status of the system (a permissive and a restrictive colour). Tables 1 and 2 show the I/O mapping of the system.

Table 1 (the input mapping table)

Operation	a_1	b_1	c_1	a_0	b_0	c_0	sw 1	ES
X_I	0	1	2	4	5	6	8	10

Table 2 (output mapping table)

Operation	Output channel number	Y_i NA	ND
A	0	1	-1
B	1	2	-2
C	2	4	-4

Where, NA (ND) is the number to be written on digital output interface for a forward (or a return) stroke of an agv.

Sequence 1: A+, B+, {C+,A-}, {B-,C-}

Consider that the system has to be controlled to execute the above sequence where A+ represents that the agv has to do forward stroke and A- return one. {C+, A-} represents two concurrent actions taking place simultaneously: agv C to do a forward stroke and agv A to do a return one. Fig.2 shows the LLD and Fig.3 shows the TPN corresponding to this sequence. Note that in TPN a place has attributes $[n_1,n_2]$ where n_1 is the first attribute representing an n number of tokens and n_2 is the second one mapping an input channel number. A transition has attributes $[n_1',n_2']$ where n_1' is the firing duration and n_2' is to be written on the digital output interface.The LLD shown in Fig.2 has 56 basic elements (23 nodes and 33 links), whereas the TPN shown in Fig.3 has 46 basic elements (19 nodes and 27 links). In order to compare the LLD's and TPN's another sequence with increasing complexity is considered sequence 2: START,5[A+,B+,{C+,A-},{B+,C-}] with Emergency Stop and Counter. In this sequence, there is a need to provide emergency stop and a counter. In this system, both the LLD and TPN are implemented such that when the emergency stop switch, ES is pressed, the whole system, including the active elements, are immediately stopped. Fig.4 shows the LLD and Fig.5 shows the TPN corresponding to this sequence.

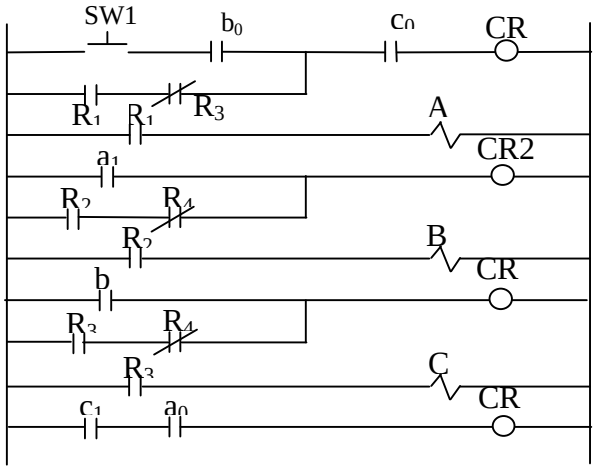

Fig.2 LLD for sequence ST, A+, B+, {C+,A-}, {B-,C-}

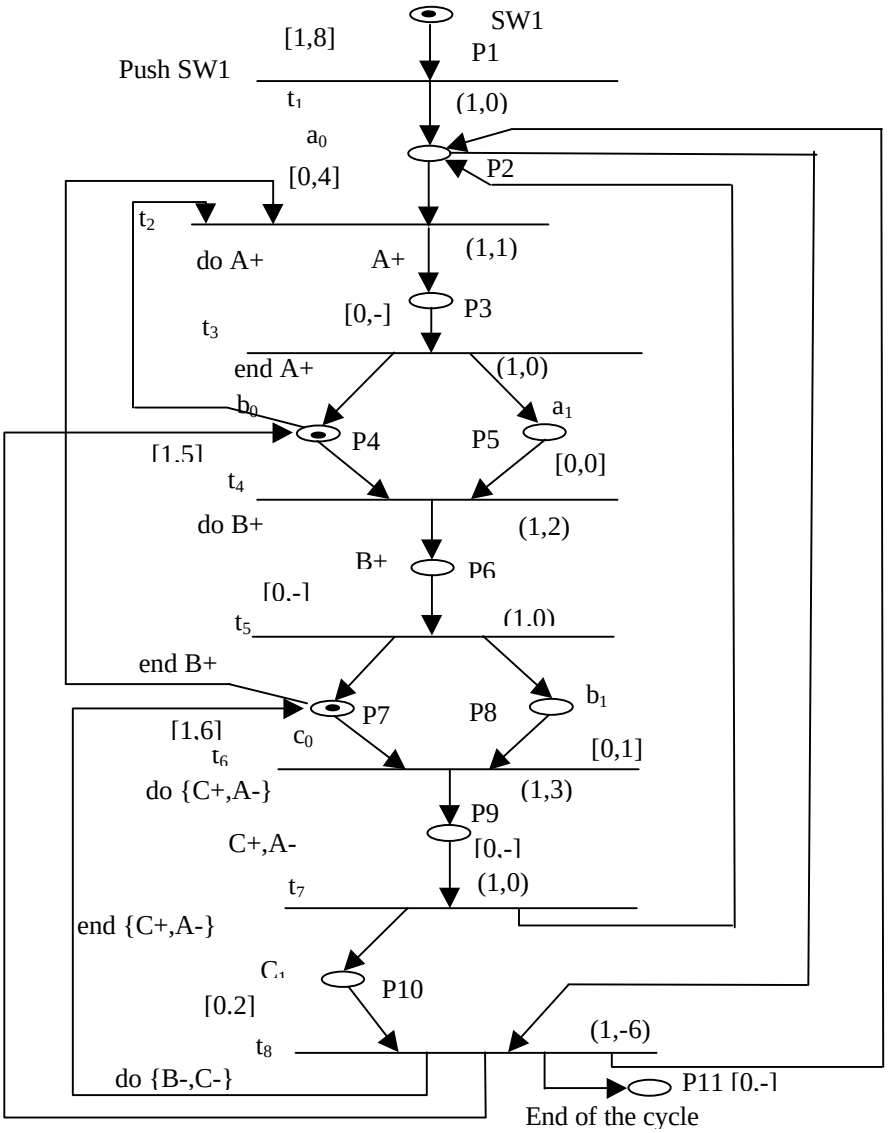

Fig.3 TPN for sequence ST, A+, B+, {C+,A-}, {B-,C-}

The LLD shown in Fig 4 has 86 basic elements (36 nodes and 50 links), whereas the TPN shown in Fig.5 has 59 basic elements (22 nodes and 37 links). The LLD needs more additional basic elements compared to TPN.

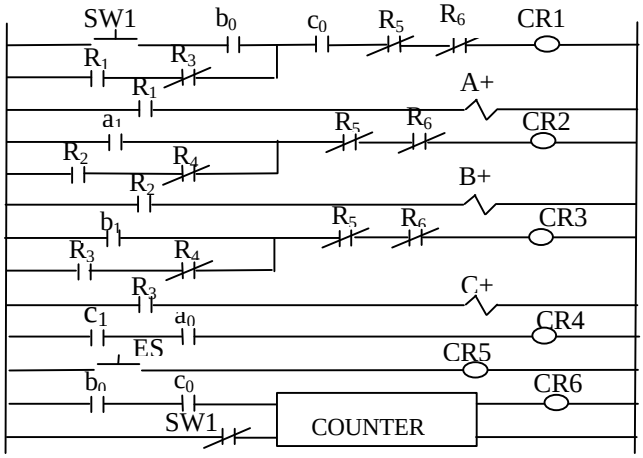

Fig.4 LLD for sequence 2

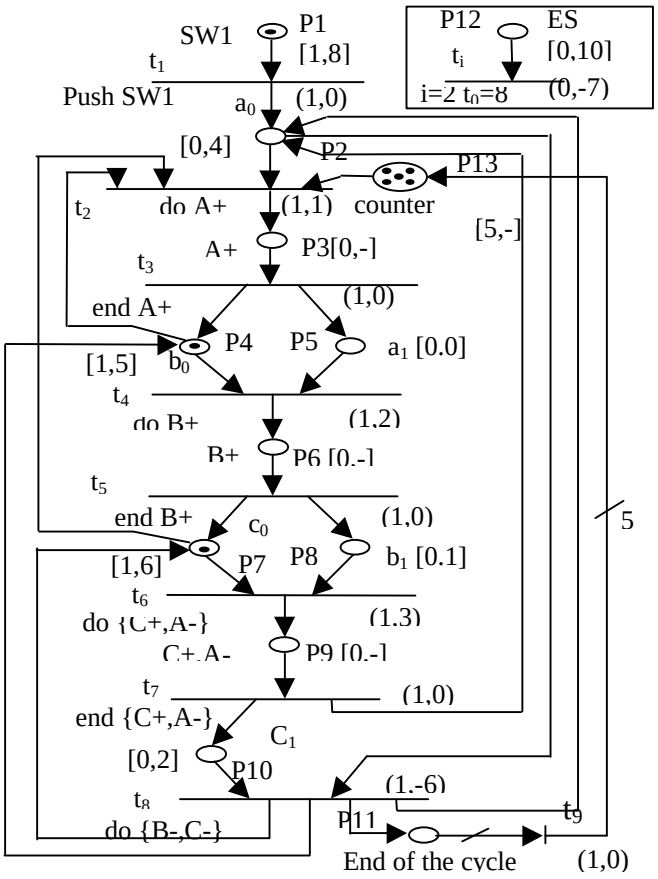

Fig.5 TPN for sequence 2

5. CONCLUSIONS

We notice that as the specification changes, the TPN requires fewer changes compared to the LLD. Using TPN's the control logic can be qualitatively analysed to check properties such as absence of deadlocks and presence of re-initializability in the system. Using LLD's qualitative analysis is not possible until it is stimulated or implemented. Using TPN's, the initial state of the system can be directly represented by its initial marking. The procedure for controlling a system using TPN's is straightforward, simple, and can be applied to control any discrete event system that has digital in/out interface and a computer. Adding more attributes to places, and transitions can extend TPN's in order to control complex transport systems. Designing a discrete event controller with choices to perform a control task can extend the present study.

REFERENCES

[1]M.Silva and R.Valette, [1998], "Petri Net and flexible manufacturing", *Adv. in Petri Nets, Lecture Notes in Computer Science*, Heidelberg: Springer-Verlag, vol.9,pp.374-417.
[2]M.C.Zhon and F.DiCesare, [1998], P.N. Synthesis for DES of FMS. MA:Kluwer.
[3]M.C.Zhon and F.DiCesare, [1992] "Design and implementation of a Petri net supervisor for a FMS", *IFAC J.Automatica*, vol.28, no.6, pp.1199-1208.
[4]R.David and H. Alla,[1992] Petri Nets and Grafcet. Engl. Cliffs, NY: Prentice Hall.
[5]A.Falcione and B.H. Krogh, [1993]"Design recovery for relay ladder logic", *IEEE Trans. on Cont. Syst. Mag.*, vol.24, no.4, pp.90-98.