PETRI NETS IN ASIC DESIGN

Marian Adamski*

This paper shows how Petri nets can be used for the design of a complex digital system. Many well known methods for the system description such as flowchart (ASM-charts) are not suitable for the description of parallel processes. Petri nets provide a mechanism which is well suited for representing such important phenomena in digital systems as parallelism and hierarchism.

1. Introduction

The synthesis is considered as a formal transformation of a rule-based specification (Adamski, 1990 and Adamski, 1990b) which is extracted from one-level or hierarchical, structured Petri net into an implementation (parallel controller specification). It is directly transferable to FPLDs (Adamski, 1991, Bolton, 1990 and Stewart et al,1991).

2. Direct Implementation of Petri Net

A Petri net is represented by a graph containing two types of nodes: places and transitions. Graphically we use circles for places and bars for transitions. The relationships from places to transitions and from transitions to places are represented by directed arcs.

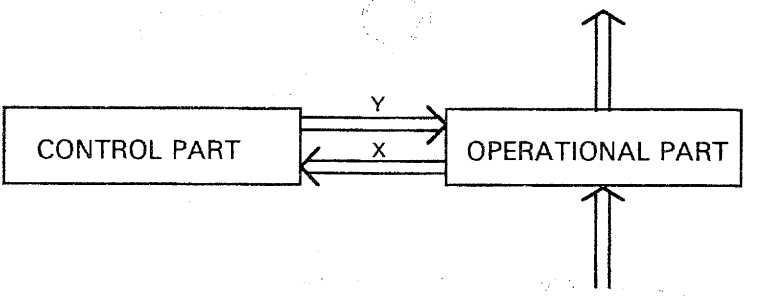

Fig. 1. Parallel state machine.

* Department of Computer Engineering and Electronics, Higher College of Engineering, ul. Podgórna 50, 65-246 Zielona Góra, Poland

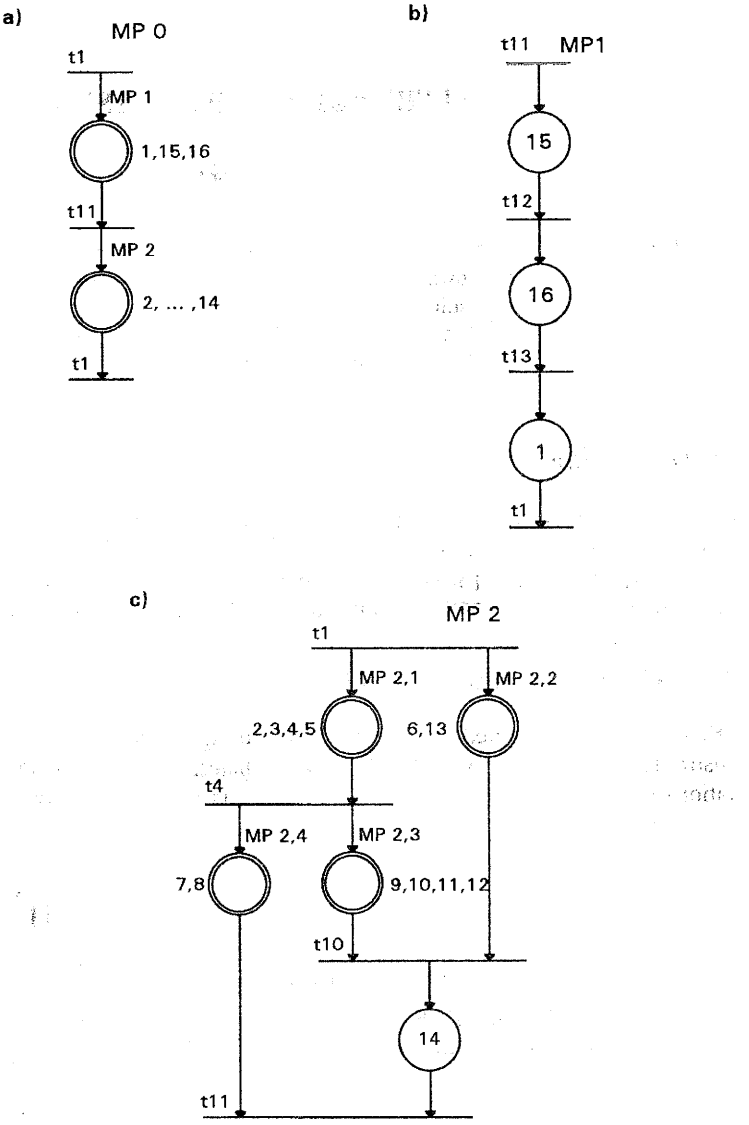

Fig. 2. Interpreted Petri net describing the behaviour of parallel controller.

A parallel state machine (Fig. 1) is a generalisation of an ordinary sequential state machine which can simultaneously be in several local states. Each global state is a collection of local states. The local state will be set to hold at least in one of the possible global states. In generating the Petri net model, each local state in the digital system specification is mapped into a distinct Petri net place. The location of the tokens in the places indicates the particular global state. Each transition determines local state

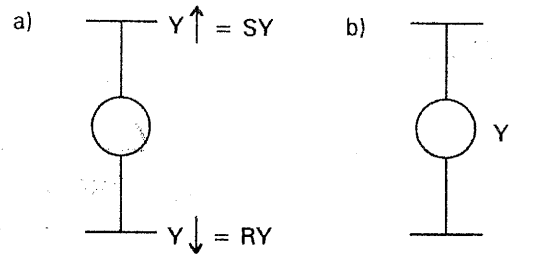

Fig. 3. Two representations of output signal Y : a) registered output, b) direct output.

changes. There is a unique set 't of local states (input places of transition t) which cease to hold before transition t states. has occurred and unique set of t' of local states (output places of transition t) which begin to hold after transition has occurred.

An interpreted, labelled Petri net (Fig. 2) is obtained from an ordinary net by adding a set X of input symbols (external conditions), a set Y of output symbols (actions in operational part of digital system) and labelling functions for places and transitions (Fig. 2).

The description of the Petri net including one which is presented in the papers (Adamski, 1987, Adamski, 1990, Adamski, 1990b and Adamski, 1991) is very easy to obtain by hand. The notation SY and RY represents the rising edge and the falling edge of the output signal Y respectively (Fig. 3, 4).

The rule -based symbolic Petri net description in the form of Gentzen logic sequents (Tabl. 1), (extended production rules) which is easily obtained by inspection of the net is recommended as an intermediate form of representation of the designed parallel controller. It is transformed (Tabl. 2, 3) into logic expressions accepted by standard PLD software (Adamski, 1991).

The most important features of the presented structural method of the interpreted Petri net realization are as follows:

1) direct interpretation of each transition in the form of a separate fragment (logical product term) of a combination part of a controller circuit;
2) direct interpretation of each place reflected as the state (product term of some flip-flops signals) of a limited part of the controller memory register.

Places (local internal states) encoding is done according to the following rules:

1) codes of simultaneously marked places (concurrent places) are non-orthogonal,
2) codes of non-simultaneously marked places (non-concurrent places) are orthogonal.

After the local states (places) are binary encoded (Table 2) the behaviour of a controller is represented by logical sequents (Table 3), which specify the binary values of the state register variables and output variables for each combination of binary input values.

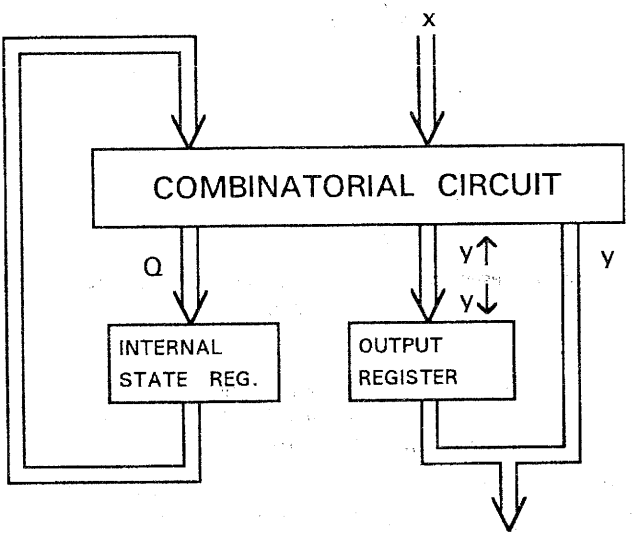

Fig. 4. Direct one-level parallel controller implementation.

The binary sequents are transformed to the excitation sequents by means of simple transformation (Adamski, 1987, Adamski, 1990). The next state variables Qi' whose values are different from Qi are replaced by Ji if Qi=0, Qi'=1 or by Ki, if Qi=1, Qi'=0 and omitted if Qi=Qi' (JK flip-flops). For the sake of simplicity the symbols of flip-flops which do not change their states are rejected from the right sides of the sequents, in advance. The derivation of D flip-flop excitation functions requires the transformation Di=/Qi*Ji + Qi*/Ki of J, K functions into D function.

3. Petri Net Hierarchical Behavioural Model of Digital Systems

The strategy that leads to a representation of digital processes as a collection of simple sequential processes may involve a complicated communication structure, which is very often extremely difficult to understand and to specify correctly. More complex processes may be represented by means of a collection of relatively simple Petri subnets, including state machine subnets as a special case of Petri nets. In a hierarchical specification a system (Fig. 5) consisting of a finite set of subnets is considered, in which all the subnets of the composite net, except one basic net , are well-formed blocks (Adamski, 1990 and Tal and Yuditskii, 1982).

The set of subnets is partially ordered (Fig. 6). The relations between the subnets are graphically represented by the relation tree of the system of subnets. Such a system of subnets is called a composite Petri net with hierarchical structure. The subnet on the higher level of hierarchy contains special places, called doubles, which represent particular subnets at the lower level of hierarchy belonging to the considered subnet.

Table 1. Sequent description of Petri net.

```
PLACE P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13,P14,P15,P16
INPUT X0,X1,X2,X3,X4,X5,X6,X7,X8,X9
REG_OUT Y1,Y5,Y6,Y8
COMB_OUT Y2,Y3,Y4,Y7,Y9
MARKING P1
TRANSITION
T1 : P1*X0          |- P2*P3*P6*SY1;
T2 : P2*X1          |- P4*RY1;
T3 : P3*X3          |- P5;
T4 : P4*P5          |- P8*P9*P10;
T5 : P8*X5          |- P7;
T6 : P7*/X5         |- P8;
T7 : P9*/X2         |- P11;
T8 : P10*/X4        |- P12;
T9 : P6*X7          |- P13;
T10 : P11*P12*P13   |- P14*SY5;
T11 : P8*P14*/X6    |- P15*RY5*SY8;
T12 : P15*X8        |- P16*RY8*SY6;
T13 : P16*/X9       |- P1*RY6;

P3      |- Y2;
P6      |- Y9;
P7      |- Y7;
P9      |- Y3;
P10     |- Y4.
```

Table 2. Codes of places.

```
P16     =Q1*/Q2*/Q3*/Q4
P15     =Q1*/Q2*/Q3*Q4
P1      =Q1*/Q2*Q3*/Q4
P13     =Q2*Q3*/Q7
P6      =Q2*Q3*Q7
P12     =Q1*Q2*Q3*/Q6
P10     =Q1*Q2*Q3*Q6
P11     =Q1*Q2*Q3*/Q5
P9      =Q1*Q2*Q3*Q5
P8      =Q1*Q2*/Q4
P7      =Q1*Q2*Q4
P5      =/Q1*/Q4
P3      =/Q1*Q4
P4      =/Q1*/Q5
P2      =/Q1*Q5
P14     =Q1*Q2*/Q3
```

Table 3. Sequents with encoded places.

```
INPUT X0,X1,X2,X3,X4,X5,X6,X7,X8,X9
REG_OUT Y1,Y5,Y6,Y8
COMB_OUT Y2,Y3,Y4,Y7,Y9
IN_OUT  Q1,Q2,Q3,Q4,Q5,Q6,Q7
TRANSITION
T1 : Q1*/Q2*Q3*/Q4*X0       |- /Q1*Q5*Q4*Q2*Q7*SY1;
T2 : /Q1*Q5*X1              |- /Q5*RY1;
T3 : /Q1*Q4*X3              |- /Q4;
T4 : /Q1*/Q5*/Q4            |- Q1*Q2*Q3*Q5*Q6;
T5 : Q1*Q2*/Q4*X5           |- Q4;
T6 : Q1*Q2*Q4*/X5           |- /Q4;
T7 : Q1*Q2*Q3*Q5*/X2        |- /Q5;
T8 : Q1*Q2*Q3*Q6*/X4        |- /Q6;
T9 : Q2*Q3*Q7*X7 |- /Q7;
T10 : Q1*Q2*Q3*/Q5*/Q6*/Q7|- /Q3*SY5;
T11 : Q1*Q2*/Q4*/Q3*/X6     |- /Q2*Q4*RY5*SY8;
T12 : Q1*/Q2*/Q3*Q4*X8      |- /Q4*RY8*SY6;
T13 : Q1*/Q2*/Q3*/Q4*/X9    |- Q3*RY6;

/Q1*Q4          |- Y2;
Q2*Q3*Q7        |- Y9;
Q1*Q2*Q4        |- Y7;
Q1*Q2*Q3*Q5     |- Y3;
Q1*Q2*Q3*Q6     |- Y4.
```

Each double corresponds to a compound operation, which is itself a discrete subprocess described by the double block.

The composite Petri net will be extended to coloured Petri nets (Banaszak, 1991). Colours (attributes) will be used to distinguish particular processes and to keep control using place invariants during the composition and decomposition of the net (Adamski, 1990). The canonical structure of the hierarchical implemation, especially useful when FPGA are used, is presented in Figure 7. It may by modified according to the best decomposition of the digital system, including control part as will as an operational part (Fig. 1) and taking into account the implementation constrains.

4. Conclusions

A specification of a digital system may be translated through automated processes into an implementation. The syntactic and semantic compatibility of Petri net descriptions and rule based descriptions is as close as possible. Formal logic transformations map algorithms into specific structures or Boolean equations. The sequent logic language input makes itpossible to integrate the design system with existing or developing formal

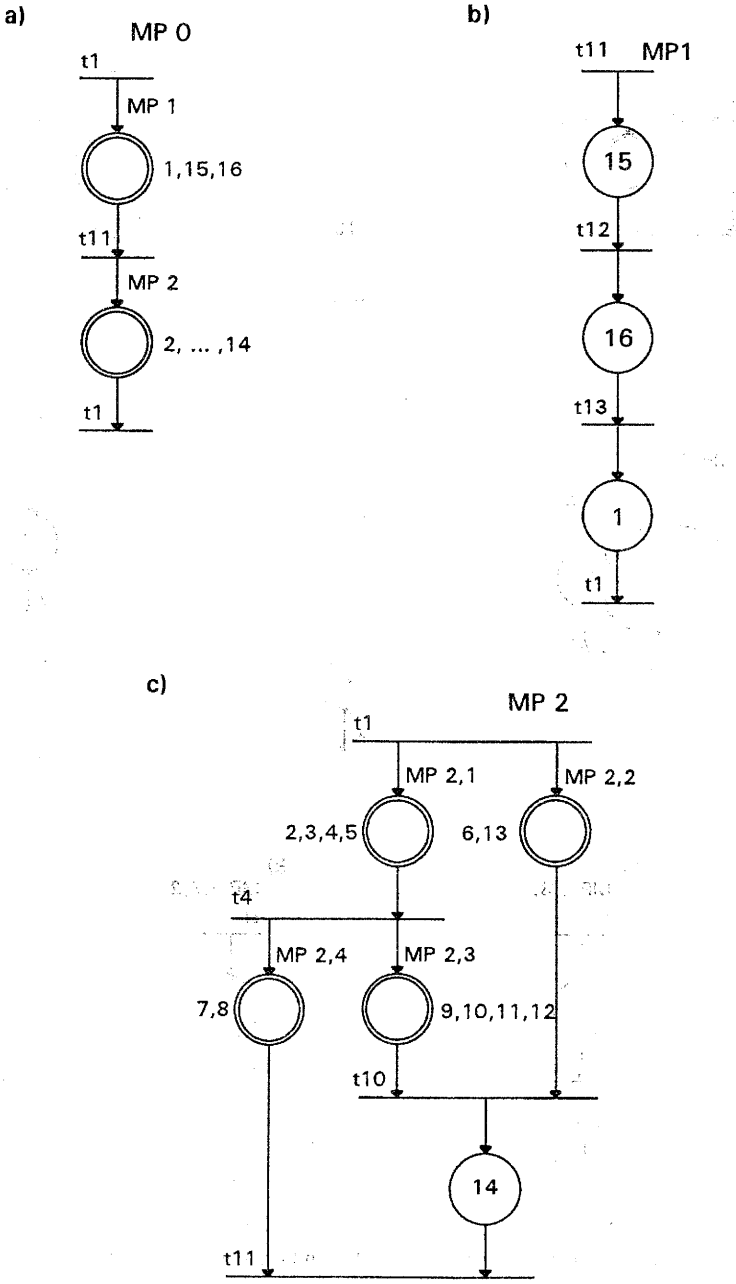

Fig. 5. Composite Petri net with hierarchical structure (part 1).

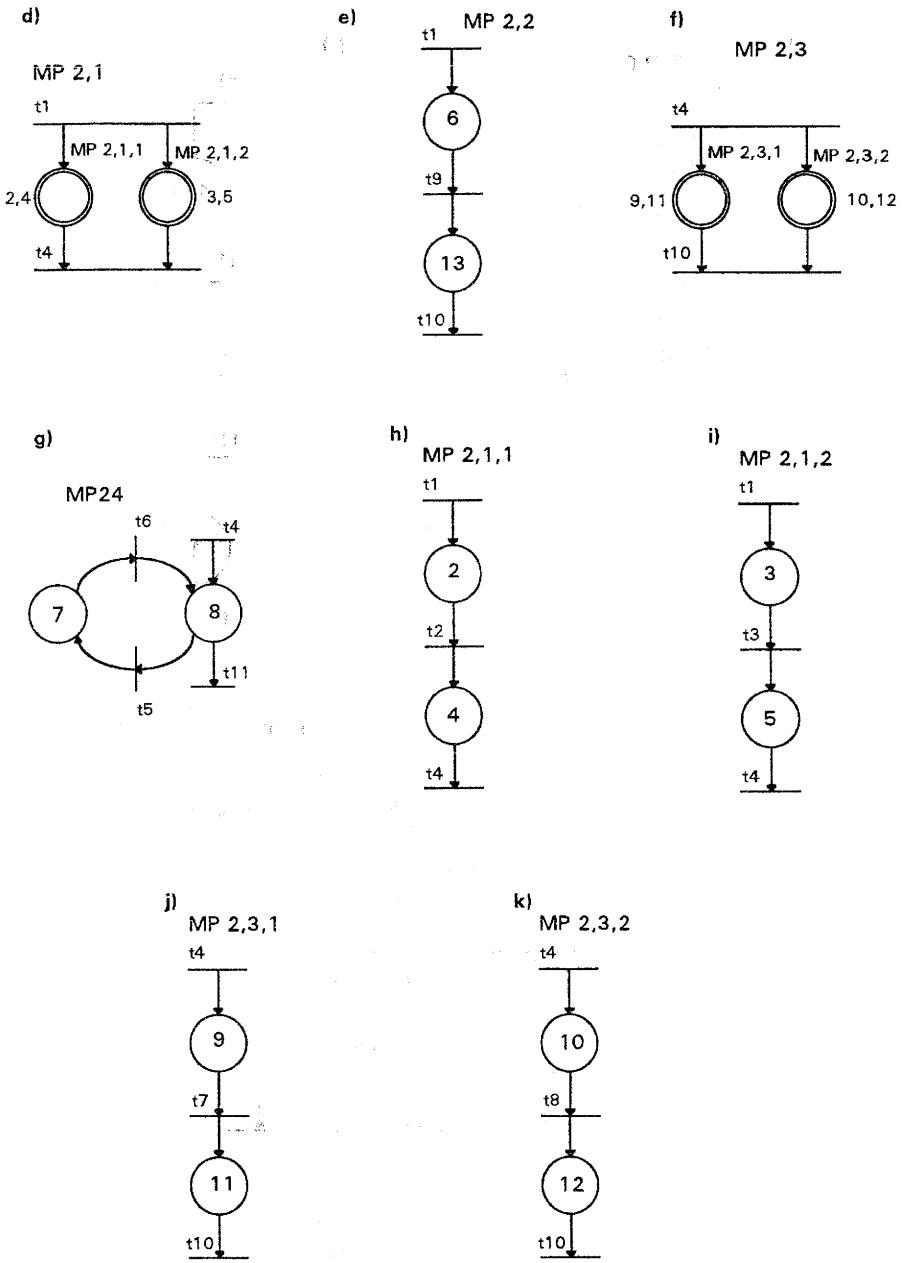

Fig. 5. Composite Petri net with hierarchical structure (part 2).

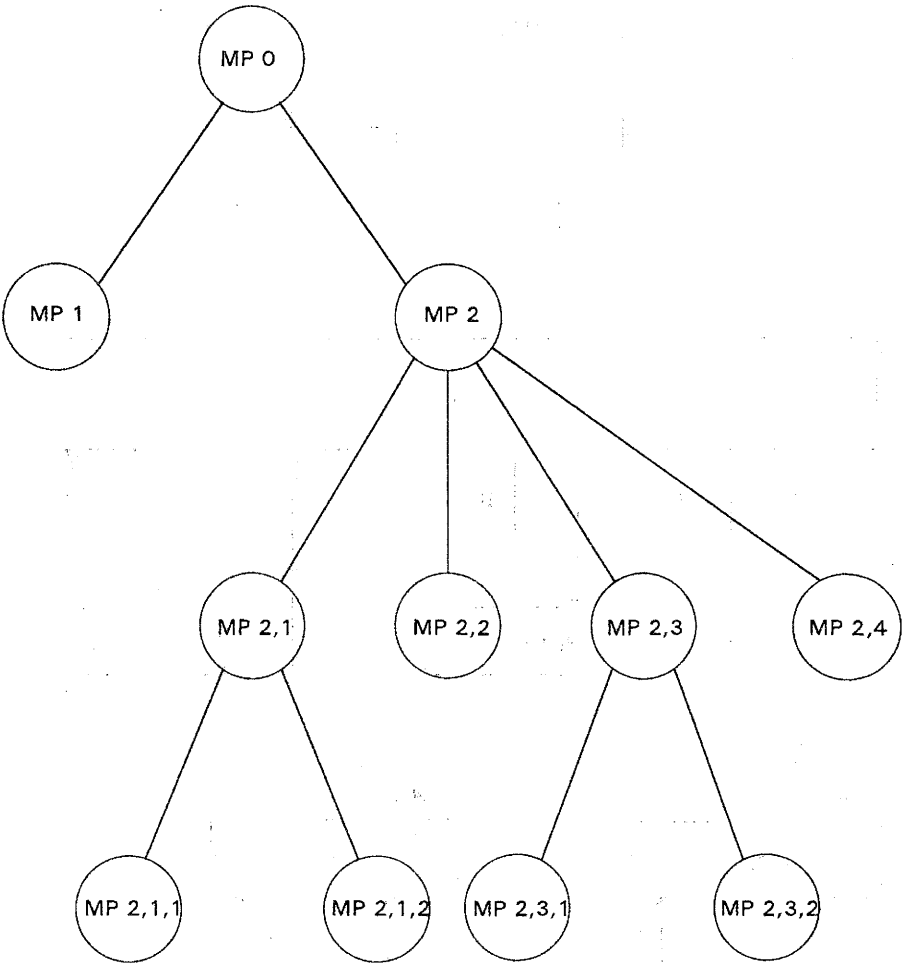

Fig. 6. Relation tree of the system of subnets.

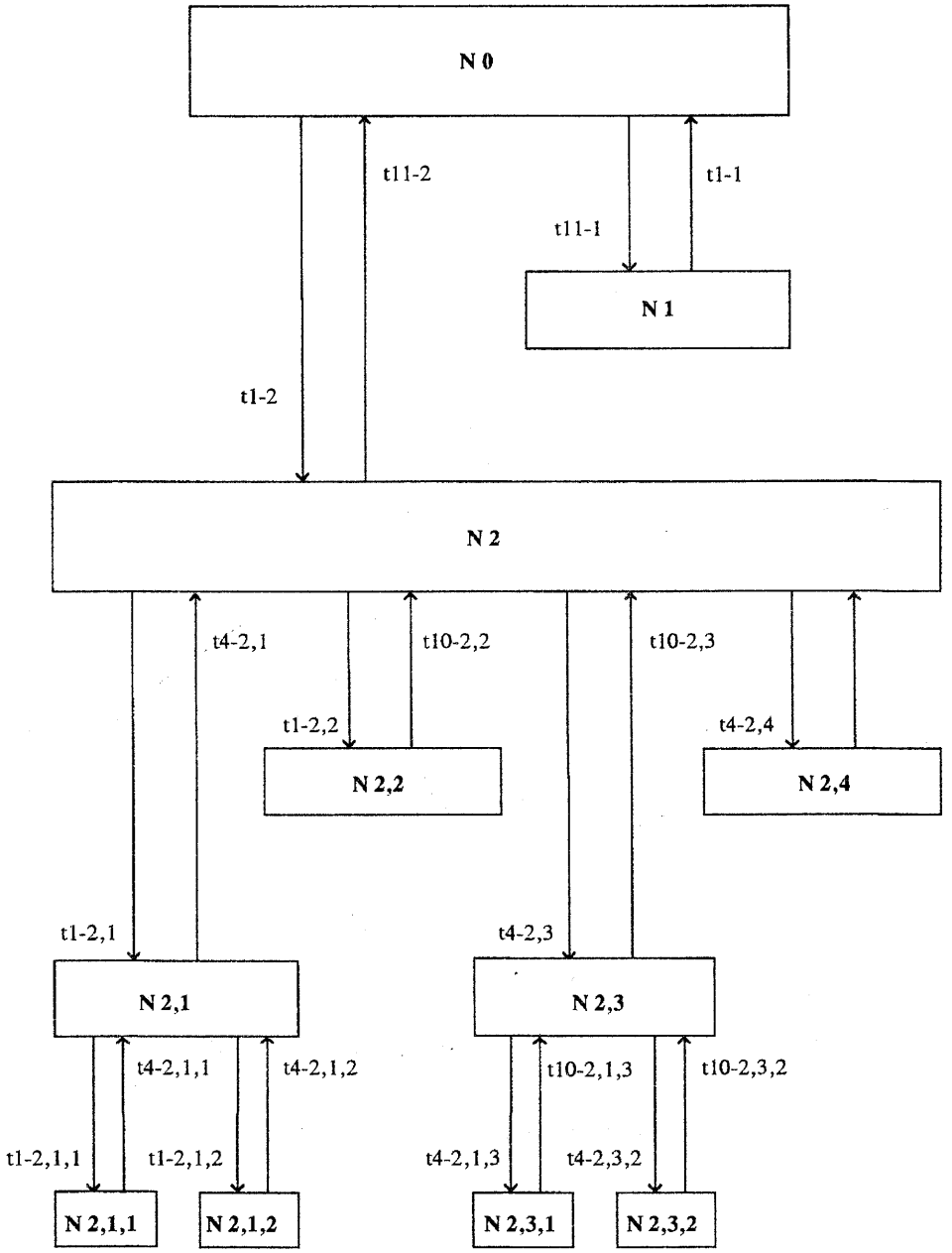

Fig. 7. Hierarchical implementation of Petri net.

logic based design systems. For the simple designs it is possible to apply proposed technique manually, as an alternative to the well known state machine charts (ASM-charts) technique.

References

Adamski M. (1987): *Direct implementation of Petri net specification.*- Proc. of the 7th Int. Conf. on Control Systems and Computer Science, Bucharest, pp.74-85.

Adamski M. (1991): *Parallel controller implementation using standard PLD software.*- FPGAs, (Eds. W.Moore, W. Luk), Edited Papers from the Int. Workshop on Field Programmable Logic and Applications, pp.296-304.

Adamski M. (1990): *Digital Systems Design by Means of Rigorous and Structural Method.*- Zielona Góra Higher College of Engineering Press, Zielona Góra (in Polish).

Adamski M. (1990b): *Digital systems design by formal transformation of specification.*- 35 IWK, Int. Conf. Proc., Technical University of Ilmenau, Germany, HEFT3, pp.62-65.

Amoroun A. and Bolton M. (1989): *Synthesis of controllers from Petri net descriptions and application of Ella.*- Proc. of the IFIP Workshop on Applied Formal Methods for Correct VLSI Design, North Holland, pp.57-74.

Banaszak Z. (1991): *Modelling and Control of FMS. Petri Nets Approach.*- Wrocław Technical University Press.

Bolton M. (1990): *Digital Systems Design With Programmable Logic.*- Addison-Wesley.

Pardey J. and Bolton M. (1991): *Logic synthesis of synchronous parallel controllers.*- Proc. of the IEEE Int. Conf. on Computer Design, pp.454-457.

Stewart J., Dagless E., Milford D. and Miles O. (1991): *A Petri net-based framestore.*- FPGAs, (Eds. W. Moore, W. Luk), Edited Papers from the International Workshop on Field Programmable Logic and Applications, pp.332-342.

Tal A.A. and Yuditskii S.A. (1982): *Hierarchy and parallelism in Petri nets 1, Composite Petri Net.*- Automation and Remote Control, v.43, No.7, pp.936-943.

Tal A.A. and Yuditskii S.A. (1982b): *Hierarchy and parallelism in Petri nets 2, automation Petri nets with parallelism.*- Automation and Remote Control, v.43, No.7, pp.1167-1171.