Modeling by Petri Nets

H. Kubátová

One specific model of a digital system in different types of Petri nets is presented. The formal definitions of the basic (black-and-white) Petri net, a place/transition net (P/T net), an arc-constant coloured Petri net (ac-CPN) and a coloured Petri net (CPN) are presented and explained on the basis of this example. Real models of dining philosophers, a producer-consumer system and railway tracks are described.

Keywords: Petri nets, formal models, hardware, digital design, Field Programmable Gate Array (FPGA), PNML, VHDL, Finite State Machine (FSM)

1 Introduction and motivation

Petri nets (PN) are a well established mechanism for system modeling. They are a mathematically defined formal model, and can be subjected to a large variety of systems. PN based models have been widely used due to their ease of understanding, declarative, logic based and modular modeling principles, and finally because they can be represented graphically. Since Petri nets began to be exploited in the 1960s, many different types of models have been introduced and used. The most popular models are presented in this paper by their definitions and by specific models. Their main advantages are shown and the differences between them are mentioned.

Petri net based models have been used in our research on digital design methodology: the design of a processor or control system architecture with special properties, e.g. fault-tolerant or fault-secure, hardware-software co-design, computer networks architecture, etc. This has led to the development of PN models in some Petri net design tools (Design/CPN, [1], JARP, [2], CPN Tools, [3]), and analysis and simulation of the model using these tools. After this high-level design has been developed and validated it becomes possible, through automatic translation to a VHDL description, to employ an FPGA implementation that will enable a custom device to be rapidly prototyped and tested (ASIC implementation is also possible). An FPGA version of a digital circuit is likely to be slower than the equivalent ASIC version, due to the regularly structured FPGA wiring channels compared to the ASIC custom logic. However, the easier custom design changes, the possibility of easy FPGA reconfiguration, and relatively easy manipulation make FPGAs very good final implementation bases for experiments.

Most models used in the hardware design process are equivalent to the Finite State Machine (FSM), [4], [5], [6], [7]. It is said that the resulting hardware must be deterministic, but we have found real models that are not equivalent to an FSM and their real behavior was tested on the final FPGA design kit platform [8]. Therefore we have concentrated on those models with really concurrent actions, with various types of dependencies (mutual exclusion, parallel, scheduled), and have studied their hardware implementation.

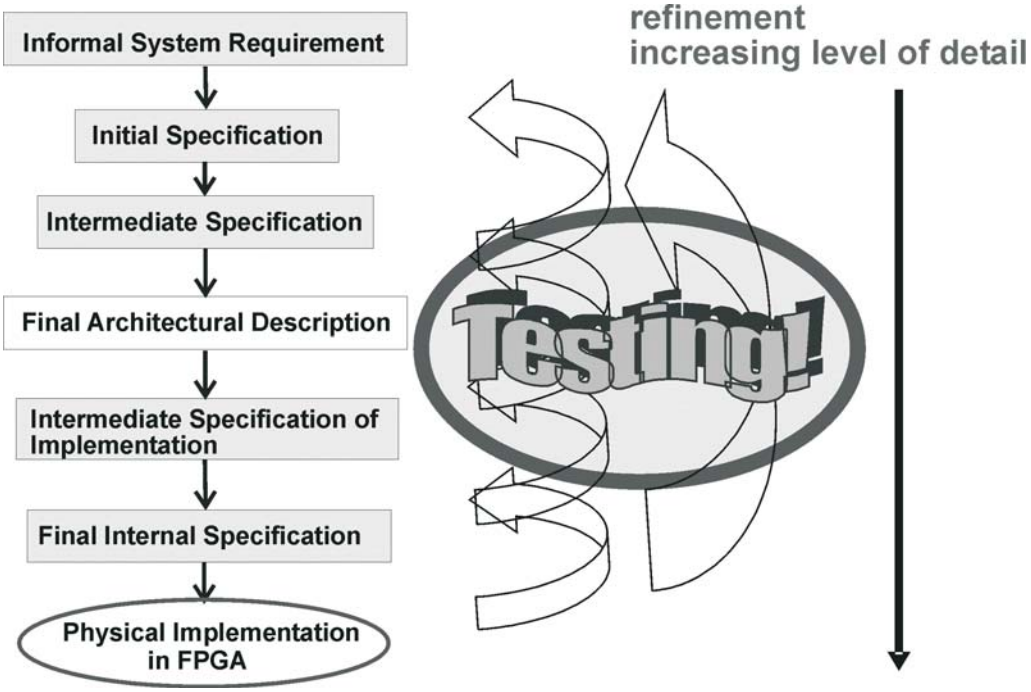

Fig. 1: Design methodology block diagram with dark parts corresponding to possible use of Petri nets

Petri nets are a good platform and tool in the "multiple-level" design process, see Fig. 1. They can serve as a specification language on all levels of specifications, and as a formal verification tool throughout these specification and architectural description levels. The first problem to be solved during the design process is the construction of a good model, which will enable the specification and further handling and verification of the different levels of this design. Therefore this paper presents such model constructions on the basis of a number of simple examples.

2 Petri net definitions and examples

Petri nets can be introduced in many ways, according to their numerous features and various applications. This text will focus on basic principles and modeling of actions. In this section, a formal definition of place/transition nets and coloured Petri nets is given. They have been presented in many books and publications, the definitions presented here being taken from [9]. Many attempts have been made to define the principles of basic types of Petri nets. The way chosen here involves a brief introduction to the basic principles and to the hierarchical construction of the most complicated and widely used Petri net based models used in professional software tools.

The essential features of Petri nets are the principles of duality, locality, concurrency, graphical and algebraic representation. These notions will be presented on a simple model of a handshake used by printers communicating with a control unit that transmits data according to the handshake scheme. The control unit uses the control signal STROBE to signal "data valid" to the target units – printers, receivers. The printers signal "data is printing" to the control unit by ACK signals. After the falling edge of a STROBE signal, all printers must react by the falling edges of ACK signals to obtain the next portion of data (e.g., a byte). Our Petri net will model cooperation between only two printers A, and B, with one control unit C, see Fig. 2.

Following essential conditions and actions have been identified:

- List of conditions:
  - p1: control unit C has a byte prepared for printing
  - p2: control unit C is waiting for signals ACK
  - p3: control unit C is sending a byte and a STROBE signal to printer A
  - p4: printer A is ready to print
  - p5: printer A is printing a byte
  - p6: printer A sends ACK signal
  - p7: control unit C sends STROBE = 0 to A
  - p8: control unit C is sending a byte and a STROBE signal to printer B
  - p9: printer B is ready to print
  - p10: printer B is printing a byte
  - p11: printer A sends ACK signal
  - p12: control unit C sends STROBE = 0 to B
- List of actions:
  - t1: control unit C sends STROBE = 1
  - t2: control unit C sends STROBE = 0
  - t3: printer A sends ACK = 1
  - t4: printer A sends ACK = 0
  - t5: printer B sends ACK = 1
  - t1: printer B sends ACK = 0

Separating or identifying passive elements (such as conditions) from active elements (such as actions) is a very important step in the design of systems. This duality is strongly supported by Petri nets. Whether an object is seen as active or passive may depend on the context or the point of view of the system. But it is always necessary to construct a correct Petri net model according to Definitions 1 – 5. Basically, the edges must connect only places with transitions, or vice-versa (Petri nets are a bipartite graph). The following principles belong to the essential features of Petri nets that express locality and concurrency:

- The principle of *duality* for Petri nets: there are two disjoint sets of elements: *P-elements* (*places*) and *T-elements* (*transitions*). Entities of the real world, interpreted as passive elements, are represented by *P*-elements (conditions, places, resources, waiting pools, channels etc.) Entities of the real world, interpreted as active elements, are represented by *T*-elements (events, transitions, actions, executions of statements, transmissions of messages etc.).

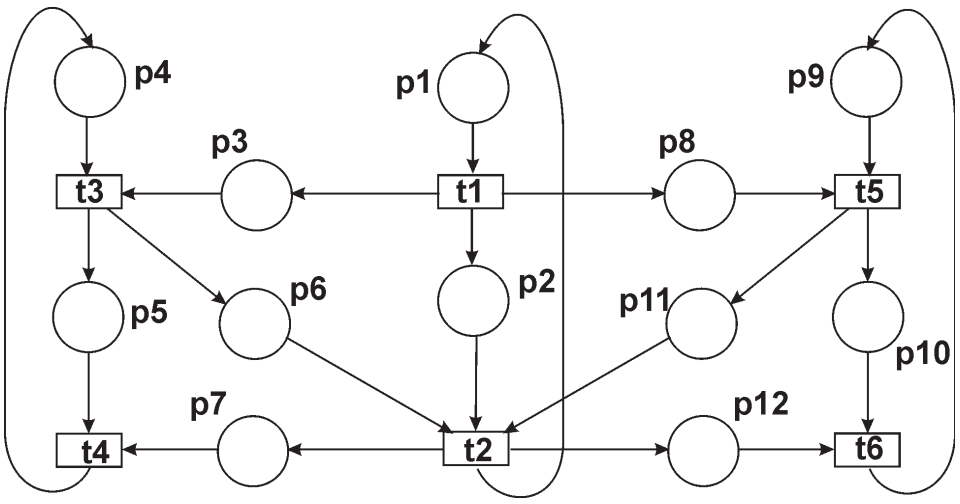

Fig. 2: The Petri net model of two printers working in parallel

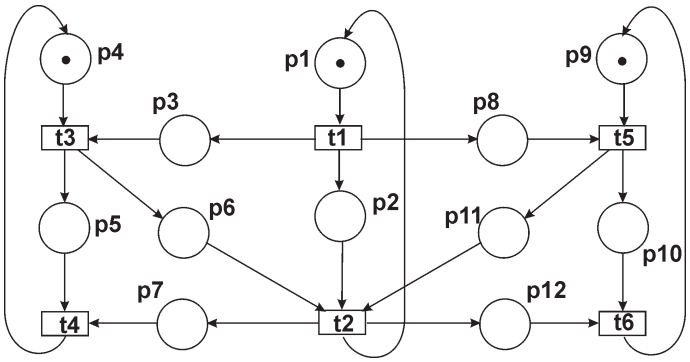

Fig. 3.: Initial state of the Petri net from Fig. 2

- The principle of *locality* for Petri nets: the behavior of a transition depends exclusively on its locality, which is defined as the totality of its input and output objects (pre- and post- conditions, input and output places, …) together with the element itself.
- The principle of *concurrency* for Petri nets: transitions having a disjoint locality occur independently (concurrently).
- The principle of *graphical representation* for Petri nets: *P*-elements are represented by rounded graphical elements (circles, ellipses, …), *T*-elements are represented by edged graphical symbols (rectangles, bars, …). Arcs connect each *T*-element with its locality, which is a set of *P*-elements. Additionally, there may be inscriptions such as names, tokens, expressions, guards.
- The principle of *algebraic representation* for Petri nets: For each graphical representation there is an algebraic representation containing equivalent information. It contains the set of places, transitions and arcs, and additional information such as inscriptions.

In contrast to concurrency, there is the notion of conflict. Some transitions can fire independently (e.g. t4 and t6 in Fig. 2, only tokens must be inside the input places), but there can be Petri nets that model mutual exclusion, see Fig. 3. Concurrent transitions behave independently and should not have any impact on each other. Sometimes this can depend on the state of the net – these transitions can behave independently. Situations that show the sophisticated interaction of concurrency and conflicts are called confu-

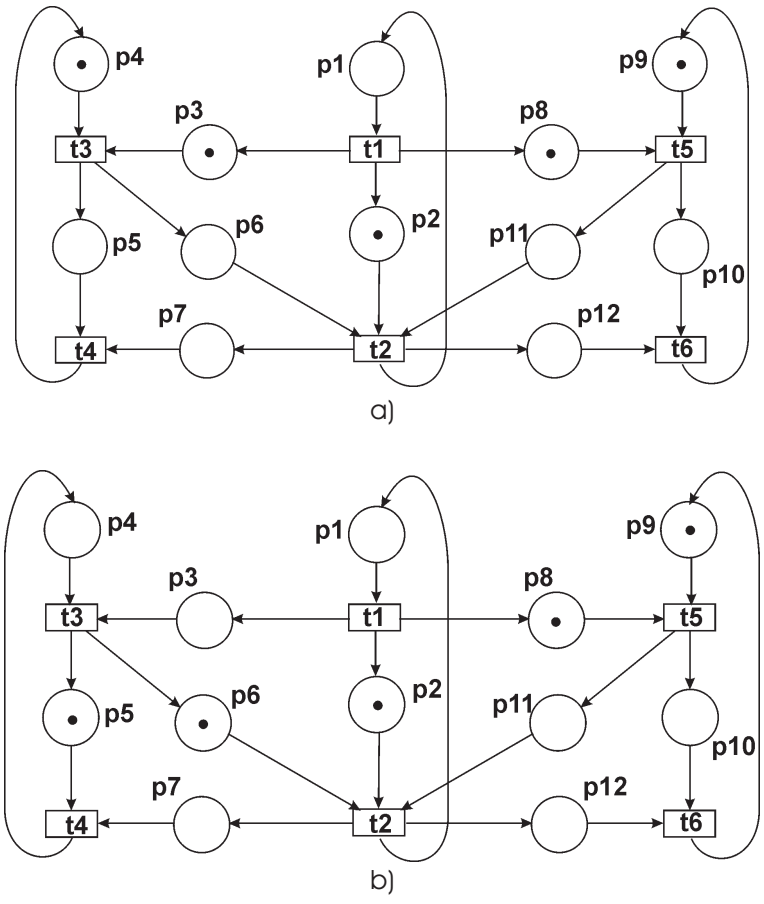

Fig. 4: Concurrency of t3 and t5 transitions a) after t1 firing both t3 and t5 are enabled, b) after t3 firing t5 still remains enabled

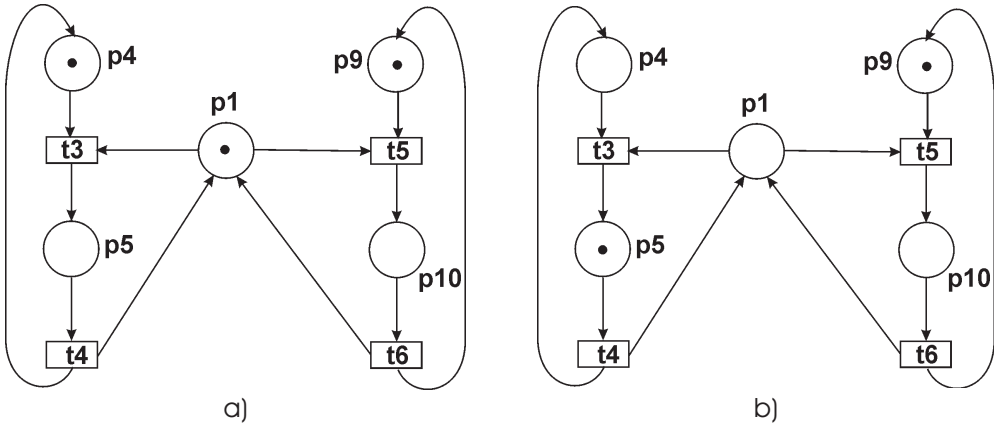

Fig. 5: Mutual exclusion of places p5 and p10, transition t3 and t5 in conflict, a) initial state where t3 and t5 are both enabled, b) the state after t3 firing, where t5 is not enabled

sions [9], [10]. Building hierarchies by abstraction or refinement is an important technique in system design. PN supports such approaches by abstraction techniques that are inherently compatible with the structure of the model, [9].

Definition 1: *A net is a triple N = (P, T, F) where*

- *P is a set of places*
- *T is a set of transitions, disjoint from P, and*
- *F is a flow relation* $F \subseteq (P \times T) \cup (T \times P)$ *for the set of arcs.*

If P and T are finite, the net is said to be finite.

The state of the net is represented by tokens in places. The tokens distributions in places are called markings. The holding of a condition (which is represented by a place) is represented by a token in the corresponding place. In our example, in the initial state control system C is prepared to send data (a token in place p1), printers A and B are ready to print (token s in places p4 and p9), see Fig. 3. A state change or marking change can be performed by firing a transition. A transition "may occur" or "is activated" or "is enabled" or "can fire" if all its input places are marked by a token. Transition firing (the occurrence of a transition) means that all tokens are removed from the input places and are added to the output places. The transitions can fire concurrently (simultaneously – independently, e.g. t3 and t5 in Fig. 4, or in conflict, see Fig. 5).

The arc in the sense of Definition 1 can be only simple – only one token can be transmitted (removed or added) from or to places by transition firings. Place/transition nets are nets in the sense of Definition 1, together with a definition of arc weights. This can be seen as an abstraction obtained from more powerful coloured Petri nets by removing the individuality of the tokens, see below. The example derived from the Petri net from Fig. 2 is shown in Fig. 6. Here more (two) printers are expressed only by two tokens in one place p4. The condition "all printers are ready" expressed by two tokens in place p4 and fulfilled by multiply edge from place p4 to transition t3.

Definition 2: *A place/transition net (P/T net) is defined as a tuple* $N^{PT} = < P, T, \mathbf{Pre}, \mathbf{Post} >$ *where*

- *P is a finite set (the set of places of* N^{PT}*),*
- *T is a finite set (the set of transitions of* N^{PT}*), disjoint from P, and*

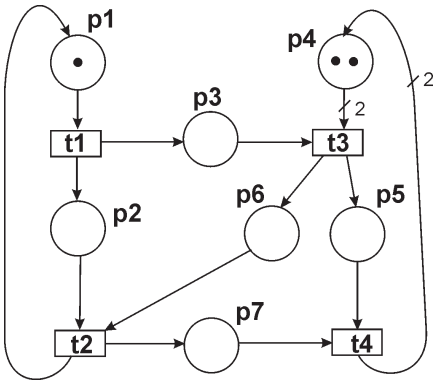

Fig. 6: Place/transition net

- **Pre**, **Post** $\in \mathbb{N}^{|P| \times |T|}$ *are matrices (the backward and forward incidence matrices of* N^{PT}*).* $\mathbf{C} = \mathbf{Pre} - \mathbf{Post}$ *is called the incidence matrix of* N^{PT}*.*

The set of these arcs is
$F := \{(p,t) \in P \times T \mid \mathbf{Pre}[p,t] > 0\} \cup \{(t,p) \in T \times P \mid \mathbf{Post}[p,t] > 0\}$.
This interpretation leads to the alternative definition, which is closer to the graphical representation.

Definition 3: *A place/transition net (P/T net) is defined as a tuple* $N^{PT} = < P, T, F, W >$*, where*

- *(P, T, F) is a net (see Definition 5.1) with finite sets P and T, and*
- $W : F \to \mathbf{N} \setminus \{0\}$ *is a function (weight function).*

N^{PT} together with an initial marking (m_0) is called a P/T net system $S = < N^{PT}, m_0 >$ *or* $S = < P, T, F, W, m_0 >$. For a net system $S = < N^{PT}, m_0 >$ the set
$RS(S) := \{m \mid \exists\, w \in T^*, m_0 \to^w m\}$, where T^* is the sequence of transitions and $w \in T^*$ and $t \in T$, is the reachability set.
$FS(S) := \{w \in T^* \mid \exists\, m, m_0 \to^w m\}$ is a set of occurrence-transition sequences (or a firing-sequence set) of S. It is sometimes convenient to define the set $Occ(S)$ of occurrence sequences to be the set of all sequences of the form

$$\mathbf{m}_0, t_0, \mathbf{m}_1, t_1, \mathbf{m}_2, t_2, \ldots, t_{n-1}, \mathbf{m}_n \qquad (n \geq 1)$$

such that $\mathbf{m}_i \to^{ti} \mathbf{m}_{i+1}$ for $i \in \{0, \ldots, n-1\}$.

The tokens in Figs. 2–5 are not distinguished from each other. The tokens representing printers A and B are distinguished by their places p4 and p9. A more compact and more natural way is to represent them in one place p4&p9 by individual tokens A and B. Distinguishable tokens are said to be coloured. Colours can be thought of as data types. For each place p, colour set $cd(p)$ is defined. In our case $cd(p4 \,\&\, p9) = \{A, B\}$. For a coloured net we have to specify the colours, and for all places and transitions, particular colour sets (color domains). Since arc inscriptions may contain different elements or multiple copies of an element, multisets (bags) are used, '*bg*'. A bag over a non-empty set A is a function $bg: A \to \boldsymbol{N}$, sometimes denoted as a formal sum $\sum_{a \in A} bg(a)'a$. Extending set operations *sum* and *difference* to Bag(A) are defined [9].

Definition 4. *An arc-constant coloured Petri net (ac-CPN) is defined as a tuple* N^{aC}=<P, T, **Pre**, **Post**, C, cd> *where*

- P *is a finite set (the set of places of* N^{aC}*),*
- T *is a finite set (the set of transitions of* N^{aC}*), disjoint from* P*,*
- C *is the set of colour classes,*
- $cd: P \to C$ *is the colour domain mapping, and*
- $\mathbf{Pre}, \mathbf{Post} \in \mathrm{B}^{|P| \times |T|}$ *are matrices (the backward and forward incidence matrices of* N^{aC}*) such that* $\mathbf{Pre}[p,t] \in \mathrm{Bag}(cd(p))$ *and* $\mathbf{Post}[p,t] \in \mathrm{Bag}(cd(p))$ *for each* $(p,t) \in P \times T$. $\mathbf{C} = \mathbf{Post} - \mathbf{Pre}$ *is called incidence matrix of* N^{aC}.

In this definition $\boldsymbol{B}$ is taken as the set Bag(A), where A is the union of all colour sets from C. The difference operator in $\mathbf{C} = \mathbf{Post} - \mathbf{Pre}$ is a formal one here, i.e. the difference is not computed as a value. A marking is a vector $\boldsymbol{m}$ such that $\boldsymbol{m}[p] \in \mathrm{Bag}(cd(p))$ for each $p \in P$. The reachability set, firing sequence, net system and occurrence have the same meaning as for P/T nets.

The example for constructing Coloured Petri nets (CPN) is discussed in several following examples and figures derived from our original model of parallel printers. Arc-constant CPN in Fig. 7 is simply derived from the initial example, with the same meaning of all places and transitions. Places p4 and p9 (and p5 and p10) originally used for distinguishing two printers are connected ("folded") to one place here named p4&p9 (p5&p10). For a transition t, it is necessary to indicate which of the individual tokens should be removed (with respect to its input places). This is done by the inscriptions on the corresponding arcs in Fig. 7. Transition t3 can fire if there is an object A in place p4&p9 (and an indistinguishable token in the place p3). When it fires, token A is removed from place p4&p9 and is added to place p5&p10, and an (indistinguishable) token is added to p6. Places p4&p9 and p5&p10 have the colour domain printers = {A, B} denoting printer A and printer B. The control process is modeled by token s (STROBE). Colour domains are represented by lower case italics near the place symbols in Fig. 7. Places p3, p6, p7, p8, p11 and p12 are assumed to hold an indistinguishable token and therefore have the colour domain token = {•}, which is assumed to hold by default. The net from Fig. 2 (ordinary or black-and-white PN) and the net from Fig. 7 (coloured PN) contain the same information and have similar behavior. Only two places are "safe". This CPN is called arc-constant, since the inscriptions on the arcs are constants and not variables.

The next step will be to simplify the graph structure of ac-CPN. We will represent the messages "STROBE signal sent to printer A" (*stA*) and "STROBE signal sent to printer B" (*stB*), ACK signal sent from printer A (*ackA*) and ACK signal sent from printer B (*ackB*). We can connect places p3 and p8, p6 and p11, p7 and p12, in Fig. 8 they are named by the first name of the connected places. The behaviour of the net is the same. As a new feature of this net, transition t2 has to remove both signals *ackA* and *ackB* from place p6. The expression $ackA + ackB$ denotes the set $\{ackA, ackB\}$. Transi-

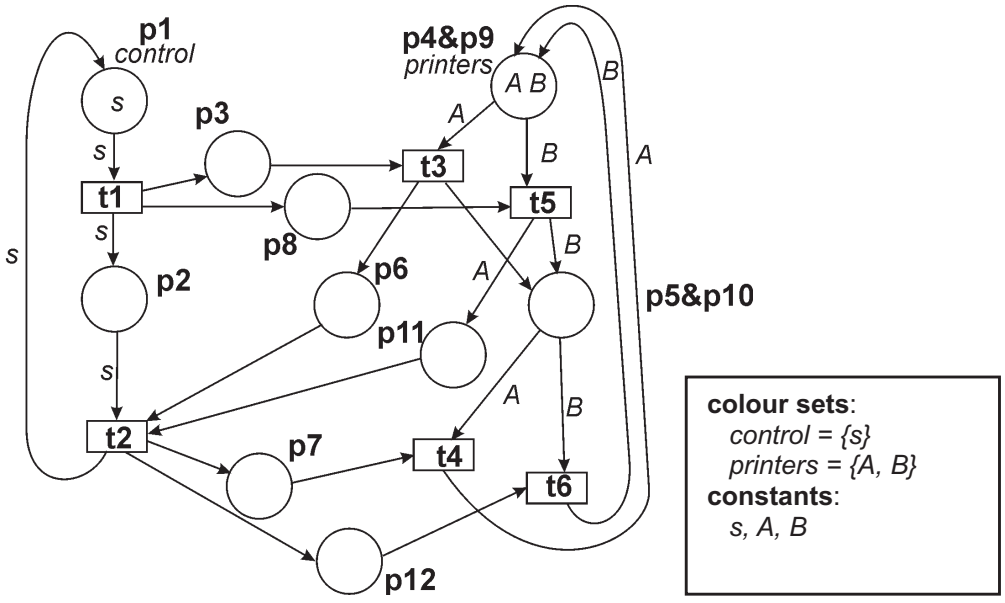

Fig. 7: Arc-constant CPN

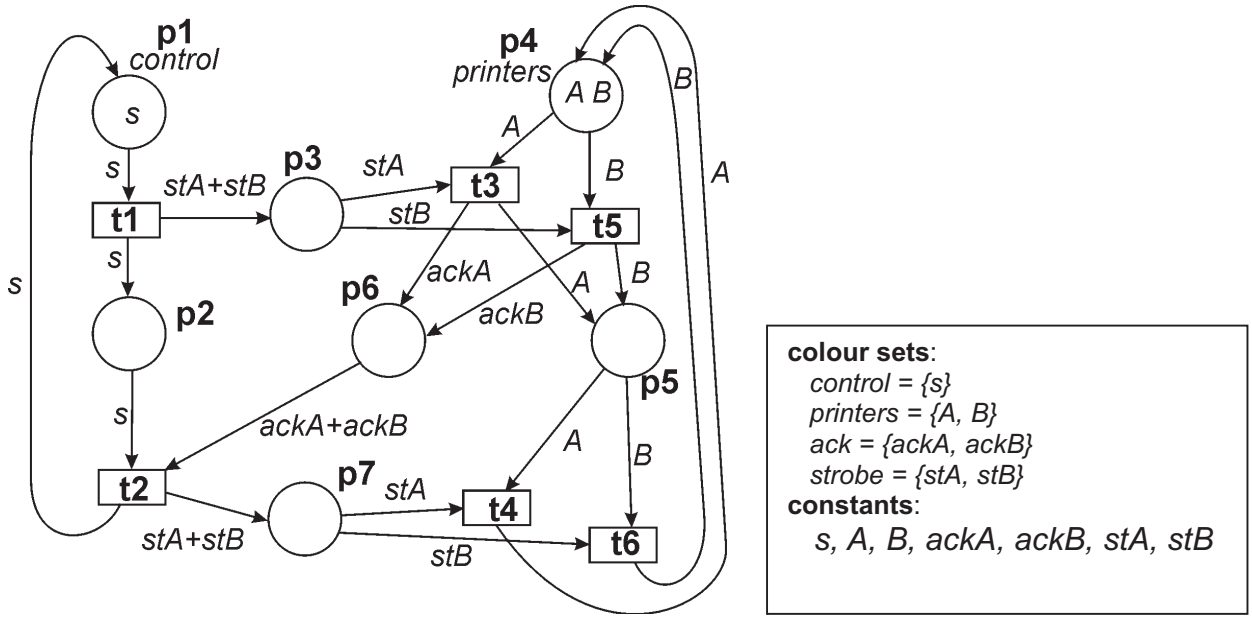

Fig. 8: Arc-constant CPN without three places

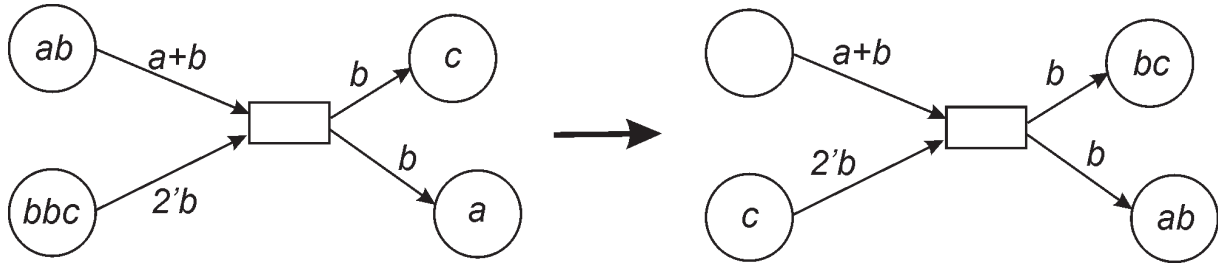

Fig. 9: Firing rule for ac-CPN

tion t3 is enabled if both *ackA* and *ackB* are in place p4 and by t3 firing both tokens are removed. Therefore in the general case, bags (multisets) will be used instead of sets. The transition firing rule for arc-constant CPN can be expressed as: all input places must contain at least as many individual tokens as specified by the corresponding arcs. The transition firing means that these tokens are removed and added to the output places as indicated by arc inscriptions.

The firing rule for ac-CPN is sketched in Fig. 9.

In a coloured Petri net the incidence matrices cannot be defined over $\boldsymbol{B} = \mathrm{Bag}(A)$ as for arc-constant CPNs. The different modes or bindings of a transition have to be represented. These are called colours, and are denoted by $cd(t)$. Therefore the colour domain mapping cd is extended from P to $P \bigcup T$. In the entries of the incidence matrices for each transition colour, a multiset has to be specified. This is formalized by a mapping from $cd(t)$ into the bags of colour sets over $cd(p)$ for each $(p, t) \in P \times T$.

Our example expressed by CPN is shown in Fig. 11. The number of places and transitions corresponds to the P/T net in Fig. 6, but the expression power is greater. For each transition a finite set of variables is defined which is strictly local to this transition. These variables have types or colour domains which are usually the colours of the places connected to the transition. In Fig. 11 the set of variables of transition t3 is $\{x, y\}$. The types of x and y are $\mathrm{dom}(x) = printers$ and $\mathrm{dom}(y) = ack$, respectively. An assignment of values to variables is called a binding. Not all possible bindings can be allowed for a correctly behaving net. The appropriate restriction is defined by a predicate at the transition, which is called a guard. Now the occurrence (firing) rule is as follows, see Fig. 10, where all places have the colour set $cd(p) = \mathrm{objects} = \{a, b, c\}$, and the colour domain of all variables is also objects:

1. Select a binding such that the guard holds (associate with each variable a value of its colour), Fig. 10b.
2. Temporarily replace variables by associated constants, Fig. 10c.
3. Apply the firing rule from ac-CPN from Fig. 9 as shown in Fig. 10d (remove all appropriate tokens from input and add to output places according the arc inscriptions).

The firing rule should be understood as a single step from Fig. 10a to d. If the binding $x = a$, $y = b$, $z = c$ is selected, then

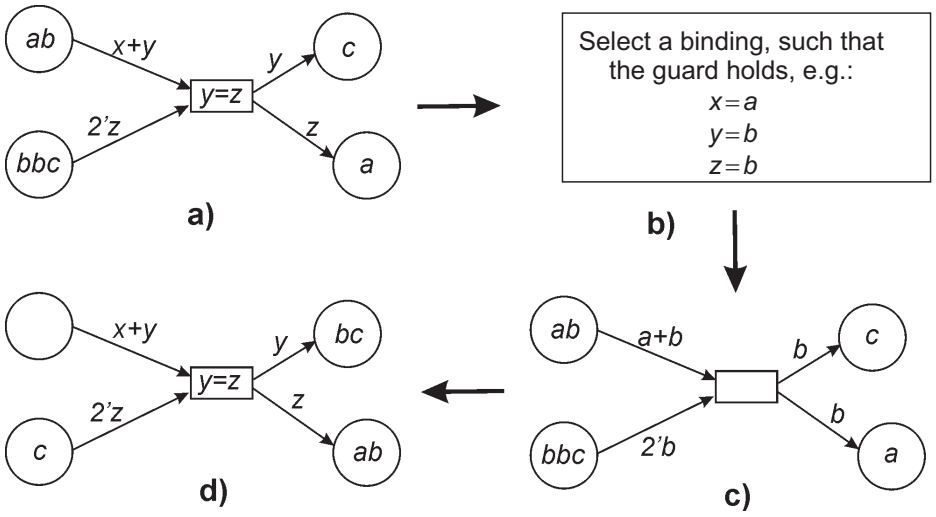

Fig. 10: Firing rule for CPN

the transition is not enabled in this binding, since the guard is not satisfied. The selection of a binding is local to a transition.

Definition 5. *A coloured Petri net (CPN) is defined by a tuple* N^{CPN}=<*P*, *T*, **Pre**, **Post**, *C*, *cd*> *where*

- *P is a finite set (the set of places of* N^{CPN}*),*
- *T is a finite set (the set of transitions of* N^{CPN}*), disjoint from P,*
- *C is the set of colour classes,*
- *cd:* $P \cup T \rightarrow C$ *is the colour domain mapping, and*
- **Pre**, **Post** $\in \boldsymbol{B}^{|P|\times|T|}$ *are matrices (the backward and forward* incidence matrices *of* N^{CPN}*) such that*
  Pre *[p, t]* : *cd(t)* $\rightarrow$ Bag*(cd(p)) and*
  Post *[p, t]* : *cd(t)* $\rightarrow$ Bag*(cd(p)) are mappings for each pair (p, t)* $\in P\times T$.

B *can be taken as the set of mappings of the form* $f: cd(t) \rightarrow \mathrm{Bag}(cd(p))$.
C = **Post** – **Pre** *is called incidence matrix.*

The mapping **Pre**$[p,t]:cd(t) \rightarrow \mathrm{Bag}(cd(p))$ defines for each transition the colour (occurrence mode) $\beta \in cd(t)$ of t a bag **Pre**$[p,t](\beta) \in \mathrm{Bag}(cd(p))$ denoting the token bag to be removed from p when t occurs (fires) in colour β. In a similar way, Post$[p,t](\beta)$ specifies the bag to be added to p when t occurs (fires) in colour β. The overall effect of the action performed on the transition firing is given by a tuple corresponding to the arcs connected with t.

The colours of the transition can be seen as particular subsets of tuples $cd(t) \rightarrow \mathrm{Bag}(cd(p1)) \times \ldots \times \mathrm{Bag}(cd(p|P|))$, i.e., vectors having an entry for each place. But this can be an arbitrary set as well. Effective representations of this set are

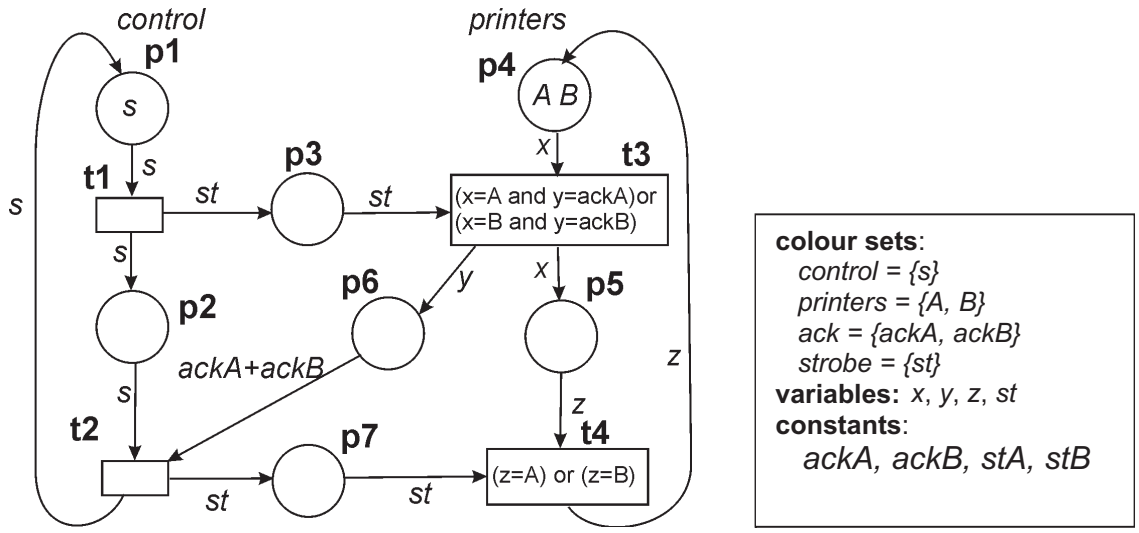

Fig. 11: CPN model of printers

necessary. The mappings **Pre**$[p,t]$ and **Post**$[p,t]$ can be denoted by vectors, projections, functions and terms with functions and variables.

3 Experiments with hardware implementation

We performed several experiments with direct implementation of the Petri nets model in hardware (FPGA). The results were presented in [14], [15] and [16].These models are briefly described here. They were constructed in software tools (Design/CPN or JARP editor) and from these tools their unified description in PNML language [11], [12], was directly transformed into the FPGA bitstream.

We have modeled 5 philosophers, who are dining together, Fig. 12. The philosophers each have two forks next to them, both of which they need in order to eat. As there are only five forks it is not possible for all 5 philosophers to be eating at the same time. The Petri net shown here models a philosopher who takes both forks simultaneously, thus preventing the situation where some philosophers may have only one fork but are not able to pick up the second fork as their neighbors have already done so. The token in the fork place (places P1, P2, …, P5) means that this fork is free. The token in the eat place (places P6, P7, …, P10) means that this philosopher is eating.

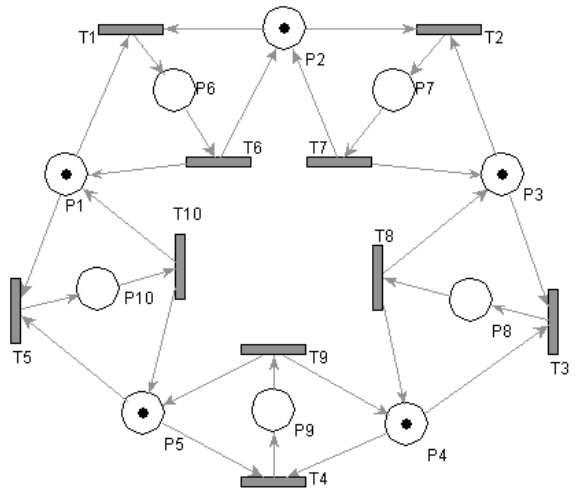

Fig. 12: The dining philosophers PN model

We also performed experiments with a "producer-consumer" system, Fig. 13. Our FPGA implementation used 59 CLB blocks, 47 flip-flops with maximum working frequency 24.4 MHz. The maximum input capacity parameter for places (the size of the counter) was set to the value 3. The average buffer occupation during 120 cycles (transition firings) was 1.43, [13], [14].

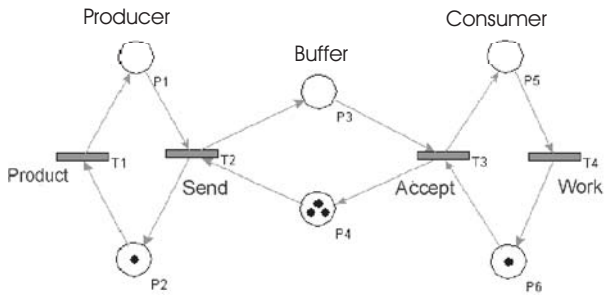

Fig. 13: Producer – consumer model

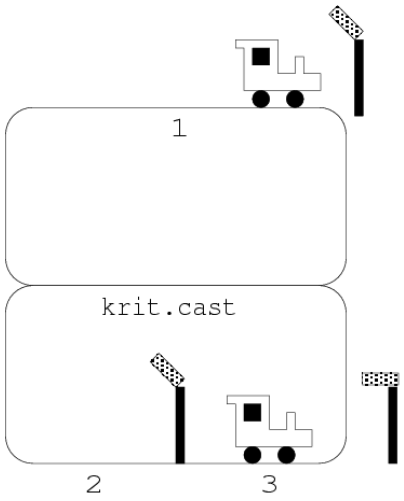

Fig. 14: Railway semaphore model

Our real application experiment modeled a railway with one common critical part – a rail, see Fig. 14. The PN model, Fig. 15, has the initial marking where tokens are in places "1T" and "3T" (two trains are on rails 1 and 3, respectively), "4F" (a critical rail is free) and "2F" (rail 2 is free). This model has eight places, two places T (train) and F (free) for each rail: a token in the first place means that the train is in this rail (T-places), and the second means that this rail is free (F-places). This was described and simulated in the Design/CPN system and then it was implemented in the real FPGA design kit (ProMoX, [15]).

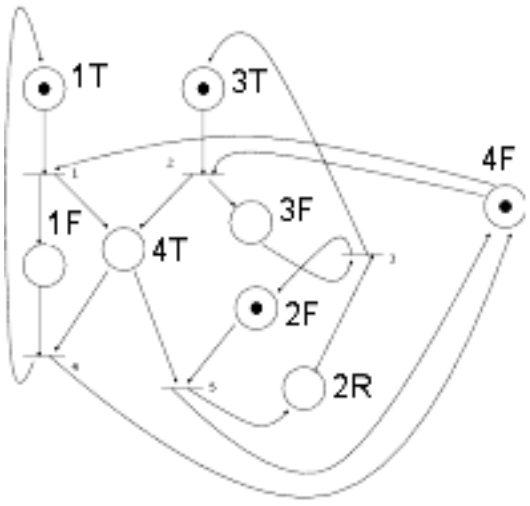

Fig. 15: The PN model of 4 rails

Conclusions

This paper deals with the practical use of Petri nets and modeling by Petri nets. Different levels and types, practical and concrete styles of modeling are presented on the basis of a simple and clear example. The practical results obtained for

specific FPGA implementations have been published and can be found in [14], [15], [16]. The specific Petri net models are shown here. The example presented here in which parallel printers are served by a controlling process, was chosen due its practical presentation and practical iterative construction during the teaching process at the Department of Computer Science and Engineering (DSCE) of the Czech Technical University in Prague.

Future work will involve optimizing the direct implementation of Petri nets with respect to space, time, power and reliability.

Acknowledgment

This research was in part supported by a grant 102/04/0737 of the Czech Grant Agency (GAČR) and by the MSM 212300014 research program.

References

[1] http://www.daimi.au.dk/designCPN/

[2] Home Page, 2002: http://jarp.sourceforge.net/

[3] http://wiki.daimi.au.dk/cpntools/cpntools.wiki

[4] Adamski, M.: "A Rigorous Design Methodology for Reprogrammable Logic Controllers." Proc. DESDes '01 Zielona Gora, Poland, 2001, p. 53–60.

[5] Erhard, W., Reinsch, A. Schober, T.: "Modeling and Verification of Sequential Control Path Using Petri Nets." Proc. DES Des '01 Zielona Gora, Poland, 2001, p. 41–46.

[6] Gomes, L., Barros, J-P.: "Using Hierarchical Structuring Mechanism with Petri Nets for PLD Based System Design." Proc. DESDes '01 Zielona Gora, Poland, 2001, p. 47–52.

[7] Uzam, M., Avci, M. Kürsat, M.: "Digital Hardware Implementation of Petri Nets Based Specification: Direct Translation from Safe Automation Petri Nets to Circuit Elements." Proc. DESDes '01, Zielona Gora, Poland, 2001, p. 25–33.

[8] Projects (2003): http://service.felk.cvut.cz/courses/36APZ/archives/2002-2003/W/prj/36APZ124/

[9] Girault, C., Valk, R.: *Petri Nets for Systems Engineering*. Berlin, Heidelberg: Springer-Verlag, 2003, 607 p.

[10] Češka, M.: *Petriho sítě*. Brno: Akademické nakladatelství CERM, 1994, 95 p. (in Czech)

[11] Humboldt-Universität Berlin: http://www.informatik.hu-berlin.de/top/pnml

[12] Kindler, E., ed.: "Definition, Implementation and Application of a Standard Interchange Format for Petri Nets." Proceedings of the Workshop of the satellite event of the 25th International Conf. on Application and Theory of Petri Nets, Bologna, Italy, June 2004, 85 p.

[13] Koblížek, M.: "Hardware Implementation of Petri Nets." Diploma thesis, CTU, Prague, 2002, 51 p. (in Czech).

[14] Kubátová, H.: "Direct Implementation of Petri Net Based model in FPGA." Proceedings of the International Workshop on Discrete-Event System Design – DESDes '04. Zielona Gora: University of Zielona Gora, 2004, p. 31-36.

[15] Kubátová, H.: *Petri Net Models in Hardware*. ECMS 2003, Technical University, Liberec, 2003, p. 158–162.

[16] Kubátová, H.: "Direct Hardware Implementation of Petri Net based Models." Proceedings of the Work in Progress Session of EUROMICRO 2003, Linz: J. Kepler University – FAW. 2003, p. 56–57.

Ing. Hana Kubátová, CSc.
e-mail: kubatova@fel.cvut.cz

Department of Computer Science and Engineering

Czech Technical University in Prague
Faculty of Electrical Engineering
Karlovo nám. 13
121 35 Prague, Czech Republic

