Sławomir Samolej[*], *Tomasz Szmuc*[**]

TIME EXTENSIONS OF PETRI NETS FOR MODELLING AND VERIFICATION OF HARD REAL-TIME SYSTEMS

The main aim of the paper is a presentation of time extensions of Petri nets appropriate for modelling and analysis of hard real-time systems. It is assumed, that the extensions must provide a model of time flow, an ability to force a transition to fire within a stated timing constraint (the so-called the strong firing rule), and timing constraints represented by intervals. The presented survey includes extensions of classical Place/Transition Petri nets, as well as the ones applied to high-level Petri nets. An expressiveness of each time extension is illustrated using simple hard real-time system. The paper includes also a brief description of analysis and verification methods related to the extensions, and a survey of software tools supporting modelling and analysis of the considered Petri nets.
Keywords: *hard real-time systems, time(d) Petri nets, high-level Petri nets*

ROZSZERZENIA CZASOWE SIECI PETRIEGO DO MODELOWANIA I WERYFIKACJI SYSTEMÓW CZASU RZECZYWISTEGO O TWARDYCH WYMAGANIACH CZASOWYCH
Głównym celem pracy jest prezentacja rozszerzeń czasowych sieci Petriego pod kątem przydatności do modelowania i analizy systemów czasu rzeczywistego o twardych wymaganiach czasowych. Zakłada się, że rozważane rozszerzenia winny spełniać następujące warunki: modelowania upływu czasu, forsowania odpalenia przejścia w określonych warunkach czasowych (tzw. silna reguła odpalania), przedziałowej reprezentacji ograniczeń czasowych. Przegląd dotyczy zarówno klasycznych sieci miejsc i przejść, jak również sieci Petriego wyższego poziomu. Siła ekspresji poszczególnych rozszerzeń jest ilustrowana na wspólnym przykładzie systemu o twardych wymaganiach czasowych. Artykuł zawiera również krótki opis metod analizy i weryfikacji ograniczeń czasowych oraz przegląd systemów wspomagających modelowanie i analizę rozważanych sieci Petriego.
Słowa kluczowe: *systemy czasu rzeczywistego, twarde wymagania czasowe, czasowe sieci Petriego wyższego poziomu*

1. Introduction

Real-time systems constitute an important part of modern computer systems. The main feature of real-time systems is a dependency of their computations not only upon data (events)

[*] Computer and Control Engineering Chair, Rzeszów University of Technology, Rzeszów, e-mail: ssamolej@prz-rzeszow.pl
[**] Institute of Automatics, University of Mining and Metallurgy, Cracow, e-mail: tsz@ia.agh.edu.pl

generated by the environment, but also on time of system response. In general, it is assumed that real-time system has to react to the behaviour of the environment within a stated time or it must cyclically provide computed outputs (signals) to its environment.

Taking into consideration the ability to meet timing constraints, real-time systems can be divided into three classes [31]:

1) hard real-time systems,
2) soft real-time systems,
3) firm real-time systems.

In hard real-time systems, timing requirements have to be strictly fulfilled. It means, that the system has to produce a given output before a stated time interval elapses or, more strictly, the output must be produced exactly at the moment of time. In soft real time systems, it is expected that only an average value of a stated timing constraint should be fulfilled. In firm real-time systems, time requirements are usually a combination of hard and soft timing constraints. The hard constraints have to be fulfilled in these systems in long time intervals, whereas the soft ones are defined for short periods. An exhaustive survey of real-time systems may be found in [31]. The paper concentrates on analysis of applicability of various time extensions (in Petri nets) for modelling and verification hard real-time systems.

Before introducing time extensions of Petri nets, we would like to take note of some features of classical Place/Transition Petri Nets. We will use informal definition of a Petri net to focus on the behaviour of the net, as well as on the classical rules of transitions enabling and firing. A thorough study of classical Petri nets can be found in [21, 22]. Informally, a Petri net is a bipartite graph consisting of "place" nodes and "transition" nodes. The places, drawn as circles or ellipses, are used to represent conditions; the transitions, drawn as bars, are used to represent events. The "marking" (m) of a Petri net is a function from the set of places P to the non-negative integers N, i.e. m: $P \rightarrow N$ that assigns "tokens" to the places of the net. Tokens, drawn in the graphical representations as small dots inside places, specify state of Petri net. A transition to next state is carried out by a flow of tokens along arcs (arrows) connected to the transition chosen for execution. For any transition the places connected by arcs may be classified[1)] as input or output ones. An arc directed from place p_i to transition t_j classifies the place as an input for the transition. The output classifier is prescribed when exists an arc from the transition to the place. Multiple inputs to a transition are indicated by multiple arcs from the input places to the transition. Multiple outputs are represented by multiple arcs correspondingly. Figure 1 provides a graphical representation of a fairly simple Petri net.

Changes of marking of a Petri net are caused by "firings" (executions) of transitions. In a single arc net a transition is "enabled" to fire if and only if at least one token in each input place exists. In the classical definition of Petri nets, time period between enabling firing of a transition is indeterminate. Similarly, indeterminate is the order of firing of two or more currently enabled transitions. The firing of a transition is an instantaneous event during which one token is removed from each of transition's input places and one token is deposited in each of its output places.

1) The classification can divide set of connected places into sets which are not disjoint, i.e. include places which are of input and output sort for a given transition.

If two or more transitions are currently enabled by the presence of a token at the same input place (e.g., transitions t_2 and t_3 in Fig. 1), the firing of any of those transitions removes that token and disables the remaining transitions. These transitions are said to be "in conflict" and the place causing the conflict requires a "decision" to be made between multiple output paths.

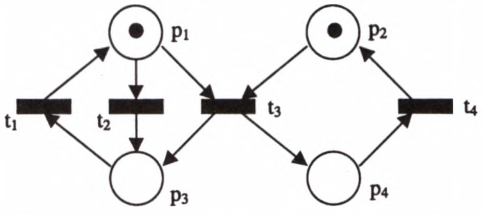


Fig. 1. A simple marked Petri net

Classical Place/Transition Petri Nets are considered as a powerful formalism for modelling of a wide class of systems. The main feature of the nets is an ability to model the flow of information and control in systems, that may be both concurrent and asynchronous. However, there are many systems that can not be efficiently modelled by classical Perti nets. A lack of time model in the net semantics is one limitation of the nets applicability. Hence, the basic Petri net formalism could not be directly used in a real-time system modelling, where an adequate time model is crucial. Therefore, several authors introduced time extensions of classical Petri nets. The first extensions were introduced independently by Merlin [18], and Ramchandani [24]. Both authors proposed to assign of time constraints to transitions, however, they defined different semantics for their nets, which led to distinct interpretations of net behaviour. Coolahan and Roussopoulos [8] proposed a consecutive time extension of Petri net. They assigned an execution time to places of the net. Furthermore, Walter [32] introduced *Arc Time Petri Nets*. The flow of time in these nets was represented by time stamps associated with tokens and arcs labelled by time intervals.

The three main time extensions of Petri nets mentioned above have been developed by several authors [23, 17, 2, 15, 29, 12, 13, 10]. Recent research are concerned with time extensions of high level Petri nets. Ghezzi, *et al.* [11] proposed *Time Environment/Relationship Nets* (*TER Nets*) and *Time Basic Nets* (*TB Nets*). *Coloured Petri Nets* have been modified by van der Aalst [1] and Jensen [16], which led to development of *Interval Timed Coloured Petri Nets* and *Timed Coloured Petri Nets*, respectively. It is purposeful to mention here about an important case of *Statically Timed Petri Nets* introduced by Cerone, et al. [6, 7]. The authors derived a new class of Timed Petri Nets as a consolidation of interval time extensions used in classical Petri nets. A systematic study of time extensions proposed for Petri nets can be found in [29, 26, 3, 4, 7].

The aim of the paper is an evaluation of time extensions of Petri nets as a tool suitable for hard real-time system modelling and verification. It seems that, three basic features should be taken into consideration when Petri net models are selected. Firstly, a model of the net has to provide an ability to measure of time flow. It means, that any sequence of states of the net can be directly mapped (observed) in a defined time domain. Secondly, the semantics of the net must ensure that a transition is forced to fire by timing constraints (the so-called strong timing model). In other words, it is a possibility of time synchronisation modelling of events in observed system. Finally, time constraints have to be also represent by time interval. Time constraints of Petri nets are usually specified by a single time value (duration or delay) or a pair of time values (time, interval). Time extensions of Petri nets with time constraints, modelled only by single values attached to elements of the nets, are not expressive enough for modelling all time mechanisms, that are crucial for hard real-time systems (e.g. *time-out*) [26, 7]. Hence, we shall reject in further considerations all time extensions of those nets, where time is modelled only by a single value. We claim, that only net fulfilling all of the aforesaid features could model a hard real-time system efficiently. Although each time extension of Petri net can model delay of a process, only the nets with strong time semantics and time constraints represented by intervals are able to model a classical time-out or synchronisation by an external event. In order to meet the hard real-time requirements, the strong time semantics changes classical firing rules of Petri nets. Unlike in classical Petri nets, a transition under the strong firing rule has to fire before a stated moment of time if it is enabled. It should be stressed that the strong timing behaviour of the net (transitions are forced to fire within stated time constraints) has a significant influence on the general properties of the net. A study of properties of Petri nets under the strong firing rules can be found in [5, 28, 29, 7].

Recent research concerning modelling complex systems by means of Petri nets led to a remarkable development of high-level Petri nets. *Environment/Relationship Nets* proposed by Ghezzi, *et al.* [11] and *Coloured Petri Nets* proposed by Jensen [16] enable to represent a model of a large system in a concise way. Moreover, time extensions have been proposed for both above mentioned high-level Petri nets. Although a strong timing model has been clearly discussed only for *Time Environment/Relationship* nets and *Time Basic Nets* [11, 9], we have decided to present *Timed Coloured Petri Nets* as well. We will reveal that the firing rule of *Timed Coloured Petri Nets* can be interpreted as the strong firing rule.

Selected time extensions of Petri nets that may be used for modelling and verification of hard real-time systems will be described in the following sections. An example of hard real-time system is introduced to examine these extensions using common criterion. The system will be modelled in every section by means of the discussed time extension.

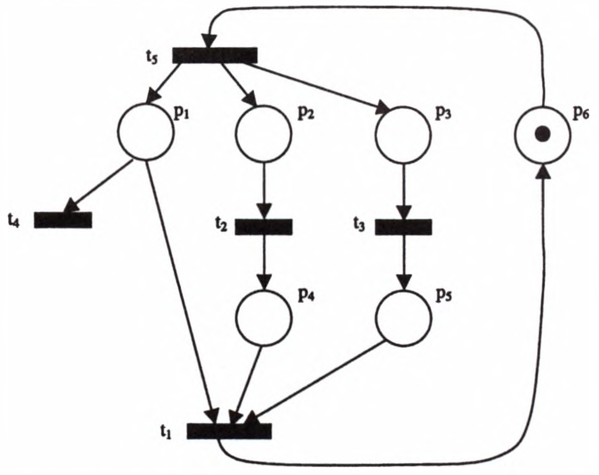


Fig. 2. Place/Transition Petri net model of a simple embedded system

Model of the system (simple embedded controller) specified using classical Petri net is shown in Figure 2.

The model includes two independent threads of computations synchronised by a global counter. Firing of transition t_5 denotes beginning of computation cycle of the controller. Firing of transitions t_2 and t_3 may be denoted as an execution of computation stages. Transitions t_1 and t_4 are in conflict. Firing of t_1 can be interpreted as a successful accomplishment of controller timing cycle, whereas firing of transition t_4 can be interpreted as a fault-stop in the system. Taking into consideration the classical rules of enabling and firing of transitions, one can not decide which transition t_4 or t_1 will fire in a currently executed "cycle" of the controller.

Therefore, additional rules of firing should be defined in order to model the behaviour more precisely. The additional rules should provide an ability to measure and control of time flow in the modelled system. In the following sections, we shall describe the selected time extensions of Petri nets appropriate for modelling of all time mechanisms characteristic for hard real-time systems. The proposed model of the system will be executed under a time-out strategy to conserve a stated cycle time. Section 2 describes *Time Petri Nets*, for the first time proposed by Merlin and Farber [18]. This net is considered as the earliest strong time model of Petri nets and it is still widely used (and modified) for modelling of systems with strong time requirements. Time constraints in Merlin's nets are attached to transitions and are represented by a pair of time values, defining time interval. Any such interval specifies time period where the transition is enabled for firing. Section 3 presents *Arc Time Petri Nets* developed by Hanisch [12, 13], where timing constraints are defined for the incoming arcs of transitions. A pair of values attached to each incoming arc of a transition defines the permeability of the arc. Section 4 includes presentation of *Statically Timed Petri Nets* (STPNs) proposed by Cerone, *et al.* [7]. STPN is a generalisation of basic time model developed for Place/Transition nets. Time constraints of STPN are time intervals that may be attached to most of elements of the nets (places, transition's incoming arcs and transitions). Section 5 describes time extensions proposed for high-level Petri nets. Time *Environment/Relationship Nets* [11], *Time Basic Nets* [11, 9] and *Timed Coloured Petri Nets* [16] are considered and analysed here. A summary and suggestions for future researd conclude the paper in Section 6.

2. Time Petri nets

Time Petri nets have been developed during research on recoverability of computer systems [18]. Merlin, who introduced Time Petri Nets, proposed two basic time extensions of classical Petri nets. The first extension consists in an assignment of two time values to every transition. These values constituted time period when the corresponding transition had to fire, if it was enabled. In the second extension, a transition was forced to fire before a stated time interval elapsed or was disabled by the firing of another transition. Merlin's model of Petri net was developed by several authors [17, 2] and became one of the most popular Petri net formalisms used for evaluation and modelling of real-time systems.

Informally, the behaviour of a Time Petri Net (according to Berthomieu, et al. [2]) can be represented as follows. Each transition has associated two values of time a and b, with $a \leq b$.

Assuming that any transition t_i is continuously enabled after it has been enabled, the time values are interpreted in the following way:

- a ($0 \leq a$), is minimal time that must elapse, starting from the time at which transition t_i is enabled, until this transition can fire;
- b ($0 \leq b$), denotes maximal time during which transition t_i can be enabled without being fired.

Times a and b, for transition t_i, are related to the moment at witch transition t_i is enabled. Assuming that transition t_i has been enabled at time τ, then t_i, even if it is continuously enabled, cannot fire before time $\tau + a$ and must fire before or at time $\tau + b$, unless it is disabled before by the firing of another transition. Note that the Time Petri net may be regarded as equivalent to a standard Petri net if all transitions have associated time constraints as $(0, \infty)$.

Formally, a Time Petri Net (*TPN*) is a tuple (P, T, B, F, M_0, SIM) where:

- P is a finite non-empty set of places p_i;
- T is a finite non-empty set of transitions t_i;
- B is the backward incidence function B: $P \times T \rightarrow N$, where N is the set of non-negative integers;
- F is the forward incidence function F: $P \times T \rightarrow N$;
- M_0 is the initial marking function M_0: PN; (P, T, B, F, and M_0 together define a Petri net);
- SIM is a mapping called static interval SIM: $T \rightarrow Q^* \times (Q^* \cup \infty)$, where Q^* is a set of positive rational numbers.

A general form for state S of a *TPN* can be defined as a pair $S = (M, I)$ consisting of:

- marking M;
- firing interval set I which is a vector of possible firing times.

Basic properties of Time Petri nets enable to model a wide range of real-time systems. Figure 3 provides Time Petri net model of simple embedded controller discussed in section 1. In the initial state of the net only place p_6 is marked and contains one token. As transition t_5 is only transition enabled, I has one entry equal to (0,0), which means that t_5 must fire immediately. Time constraints attached to transitions t_2 and t_3 define the intervals within which individual threads of computations have to be completed. We assume that the maximum cycle of computations can not exceed 100 time units (e.g. 100 μs). Consequently, when transition t_1 do not fire before 100 time units after the beginning of current controller's cycle, firing of transition t_4 (101 time units after the beginning of current controller's cycle) disables the ability to fire for transition t_1. The controller reaches the fault-stop state and it does not continue its computations. When transition t_1 fires before 100 time units after beginning of controller's cycle, firing of transitions t_6 and t_7 ensure 100 time units cycle of controller's computations. Due to the ability to estimate the time of the events in the system and, moreover, the strong timing rule of firing used in Time Petri nets, we have obtained a clear and suggestive model of a hard-real time system. Yet to obtain a satisfactory behaviour of system's model, the basic net, proposed in section 1, has to be extended by several additiond places and transitions.

The extension results from specific properties of Time Petri Nets. Each of time extensions that will be presented in the paper imposes modifications of basic system's model proposed in Section 1.

Time Petri Nets provide fairly simple and expressive model of time flow. The transitions of the net may fire only if they are enabled by appropriate marking and fulfil stated timing constraints. The behaviour of the transition may be considered locally. It has a kind of local virtual clock associated. The clock starts counting at the moment when the transition becomes enabled by the marking. The initial state of the clock is a global simulation time at which the net reached the marking. The state of the clock with reference to timing constraints associated to the transition decides when the transition may or must fire. The basic definition of Time Petri Nets does not provide the ability to construct hierarchical nets. Consequently, the nets do not have enough expressiveness for modelling of large systems.

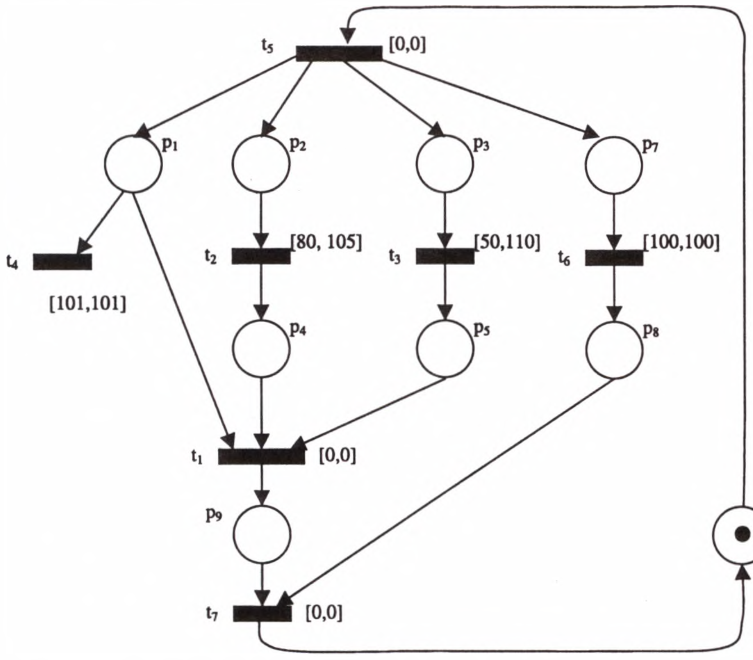


Fig. 3. Time Petri Net model of a simple embedded system

Merlin's model of time extension for Petri nets, as one of the earliest and popular, is widely used in most of software Petri nets simulators. The thorough study concerning software tools for Petri nets analysis and simulation can be found in [30]. Two selected software tools that support Time Petri Nets are ARP (LCMI Lab., Federal University of Santa Catarina, Brazil) and INA (*Integrated Net Analyser* – Institut für Informatik der Humboldt-Universität, Berlin, Germany) [25]. Most of software tools enable to editing and simulation of Time Petri Net model, some of them offer selected methods for analysis of net properties.

An analysis of Time Petri Net is usually complicated. An enabled transition may fire in any moment within a time interval associated with it. Different moments of the firing may lead to different states of the net. In general, the amount of states after the firing of a transition may be infinite. The most common way of verification of Time Petri Net model is simulation. Formal method of the analysis of the nets has been proposed in [2]. The behaviour of the net has been described by state classes. A class is a group of all possible states that may appear after firing of transition t_i. The state classes are nodes in the extended reachability graph used for formal analysis of the nets.

As it was mentioned in [7], a Time Petri Net model reveals some shortcomings, too. For example, the semantics of the net that defines initialisation of a time measure after the enabling of a transition may cause the modelling of some time critical system cumbersome. It could happen, when we would be interested in a time interval within which a token is situated in a place, but the transition that it can enable to fire is not enabled.

3. *Arc Time Petri Nets*

Arc Time Petri Nets have been introduced by Walter [32]. The basic models of the nets were changed and extended by Hanisch [12, 13], who proposed the extension of the nets by adding time stamps and the strong firing rule. The nets with timing constraints attached to the arcs were also used and interpreted in [6, 26, 7, 10]. In this section we shall use the name Arc Time Petri Nets introduced by Starke [29] for the nets with time constrains associated with arcs. However, it should be mentioned, that the nets with time constraints attached to arcs have been also called *Timed Place/Transition nets* [32], *Place/Transition nets* with *Timed Arcs* [12], *Timed Link Petri Nets* [6], *Timestamp nets* [13] and *Timed-Arc Petri Nets* [10]. Arc Time Petri Nets developed by Hanisch are the nets where intervals are assigned to the incoming arcs of transitions. An interval assigned to an arc (p, t) describes the permeability of the arc relative to the time stamp of the token on p. A time stamp of each token denotes the time when the token was put on the place. A transition is enabled if all its input places carry enough tokens according to the token weight of the arc, and if all the arcs directing to the transition are permeable. The calculation of permeability of the arc starts at the moment when a token appears in its respective place. An enabled transition must fire (the strong firing rule) within a time interval defined by timing constraints, or is not longer enabled, due to the firing of other transitions. However, a marked transition with two or more incoming places may not fire under a marking, if only its incoming arcs are not permeable simultaneously.

Formally, an Arc Time Petri Net (Time Stamp Net) is a tuple $N = (P, T, F, I)$, where (P, T, F) is a Petri net, and I: $(P \times T) \cap F \rightarrow I_R^+$ assigns a non-negative time interval to each incoming arc of a transition, describing the permeability of the arc. Given $I(p, t) = [r; l]$, $I_r(p, t) = r$ denotes the beginning and $I_l(p, t) = l$ the end of the permeability of the arc $(p, t) \in (P \times T) \cap F$, relative to the time when the incoming place is marked. The marking m of an Arc Time Petri net $N = (P, T, F, I)$ is a function m: $P \rightarrow ((R_0^+) \cup \{0\})$. If $m(p) = 0$, then $p \in P$ is not marked. If a place $p \in P$ is marked and $m(p) = \langle ts \rangle$, the attribute ts is a time stamp of the token, indicating at which time the token was put on p.

Figure 4 provides an Arc Time Petri Net model of the simple embedded controller introduced in section 1.

The time constraints attached to the incoming arcs of transitions and the firing rule of transitions enable to reduce a number of places and transitions of the net in comparison with the model proposed in section 2. The timing constraints associated with the incoming arcs of transitions t_2 and t_3 model the time intervals within which each thread of controller's computations must be completed. Thanks to ability to assign timing constraints to incoming arcs of the transitions we have obtained a more natural model of time-out mechanisms concentrated around place p_1 of the net. The time interval associated with the arc from place p_1 to transition t_1 ensures synchronisation of the threads of computations as well as a constant cycle of the modelled controller. The time interval connected to the incoming arc of transition t_4 provides the fault-stop of the system if the computations of the threads exceed the stated 100 time units cycle of the controller.

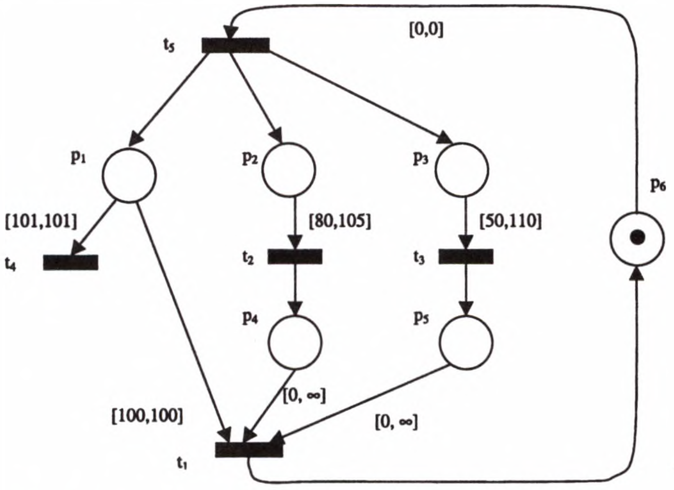


Fig. 4. An Arc Time Petri Net model of a simple embedded system

Arc Time Petri Nets provide a consistent and continuous model of time thanks to the initialisation of time measurement at the moment of token's arrival at a place. The state of a net is described by marking and a set of local clocks denoting the ages of markings of places. Timing constraints, defined as time intervals attached to the arcs of the nets, offer a significant potential in a wide range of hard real-time systems modelling [6, 26, 13]. It should be noticed that Arc Time Petri Nets enable to model essential real-time mechanisms in a fairly natural way (in our example we have obtained a simple model of time-out mechanism). Similarly to Time Petri Nets, Arc Time Petri Nets do not offer in their basic definition the ability to construct the hierarchical models of the nets. So, their main applications are rather medium-scale models of systems.

Several modern software tools support the modelling of time extensions of Petri nets with timing constraints attached to arcs. First software tool that has been used for Arc Time Petri Nets modelling was ATNA – *Arc Time Net Analyser* [12]. Arc Time Petri Net model proposed by Hanisch also may be constructed and verified by INA (*Integrated Net Analyser*– Institut für Informatik der Humboldt-Universität, Berlin, Germany) [25].

Formal analysis of Arc Time Petri Nets encounters similar restrictions to the analysis of other time Petri net extensions with timing constraints represented by intervals. In a general case, even for bounded Arc Time Petri Nets, the number of clock positions can be infinite. Software tools usually offer only the ability to model a "token game" under a stated firing rule for the nets. The formal analysis method for Arc Time Petri Nets under the earliest and maximum firing rule ([28]) has been proposed by Hanisch [12]. The author introduces the state graph for Arc Time Petri Net, which is an equivalent to the reachability graph of classical Place/Transition net. The state graph includes the set of all states of Arc Time Petri Net that are reachable from the initial state. The graph may be used to the performance analysis and optimisation of the net.

The possibility of so called timewise stuck of Arc Time Petri Nets (an inability of firing of a transition because its incoming arcs have not been permeable simultaneously) may be used to verification of time parameters of modelled system, as well [13].

4. *Statically Timed Petri Nets*

Statically Timed Petri Nets (STPN) have been proposed by Cerone, *et al.* [6, 7]. It seems that, the nets may be treated as a generalisation of time extensions developed for classical Petri nets with timing constraints defined as static intervals. The main feature of STPN is that time constraints are intervals statically associated with places, transitions and incoming links (arcs). Furthermore, the authors provide strong time semantics for STPN (in fact the authors consider both strong and weak timing semantics of STPN). Each token in the nets has an age (time stamp), that is either the time elapsed since the token was generated by a transition firing, or time 0, if its presence in the place is due to the initial marking. As in STPN time constraints may be associated with places, incoming links/arcs or transitions, the fireability of transition t may depend not only on the timing constraint that is possibly associated with t itself, but also on those that are possibly associated with its incoming places or its incoming links/arcs.

Statically Timed Petri Net can be defined formally as follows:

Let $\sum$ be a finite alphabet, and λ be an empty sequence on $\sum$. A Statically Timed Petri Net (STPN) is a tuple $N = (S_N, T_N, L_N, l_N, \delta_N, \Delta_N, M_N)$, where:

- S_N is a finite set of places, T_N is a finite set of transitions, with $S_N \cap T_N = \emptyset$, and $L_N = I_N \cup O_N$ is a finite set of links (arcs), where $I_N \subseteq S_N \times T_N$ is a set of incoming links (arcs), $O_N \subseteq T_N \times S_N$ is a set of outgoing links;
- l_N: $T_N \rightarrow \sum \cup \{\lambda\}$ is labelling function;
- the functions:
  δ_N: $S_N \cup T_N \cup L_N \rightarrow R_+$,
  Δ_N: $S_N \cup T_N \cup L_N \rightarrow R_+ \cup \{\infty\}$,
  called the lower and upper timing function, respectively, are such that:
  (a) for each $x \in S_N \cup T_N \cup L_N$, $\delta_N(x) \leq \Delta_N(x)$:
  (b) for each $t \in T_N$, if there exists no $s \in S_N$ such that $(s, t) \in I_N$, then:
  $\delta_N(t) = 0$ and $\Delta_N(t) = \infty$,
- M_N: $S_N \rightarrow N$ is the initial marking.

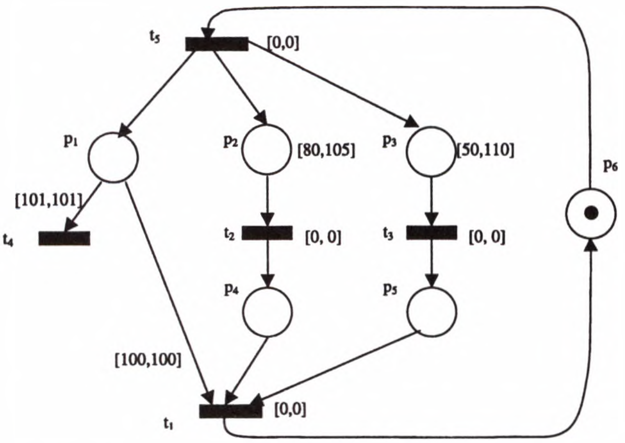


Fig. 5. A Statically Timed Petri Net model of a simple embedded system

The state of STPN may change because of two different reasons:
1) increase in the time,
2) firing of transitions.

The Statically Timed Petri Net presented in Figure 5 is a model of the simple embedded system discussed in the paper.

Although the discussed model of an embedded system may be represented by a drastically reduced STPN, we decided to preserve the "classical" shape of the net to focus on the flexibility of accessing time constraints to elements of the net. The durations of computations are represented by timing constraints attached to places p_2 and p_3, whereas the time synchronisation of the threads of computations and the time-out are represented identicaly as in Arc Time Petri Nets. The firing of transitions may be controlled by timing constraints similar to those used in Time Petri nets. It should be noticed that all the timing constraints affect the behaviour of STPN simultaneously. Moreover, each timing constraint influences a separate set or tuple of tokens. For example, the timing constraint associated with transition t_1 in Figure 5 is concerned with a tuple of tokens from the places p_1, p_4, and p_5, whereas the timing constraints attached to the links determine the behaviour of individual tokens situated in the respective places.

Note that the possibility of associating timing constraints with any element of the net offers significant flexibility in modelling a wide range of systems. Most timing mechanisms used in hard real-time systems may be modelled by means of STPN in a fairly natural way. However, the combination of a set of timing constraints may lead to complicated methods of the net analysis. So far, only the results of the research concerning expressivity of STPN have been presented [7]. Additionally, the authors who proposed STPN, did not point to any software tools, that may be used for developing and verification of STPN models. Simultaneously, STPN do not provide the ability to construct hierarchical models of systems. Hence, their practical application is bounded to modelling and simulating medium-scale models of real-time systems.

5. Time extensions of high-level Petri nets

Over the past few years, real-time computer systems were being increasingly used in time critical applications. Simultaneously, real-time systems, even hard real-time systems, became more complex. Hence, recent research concerning real-time systems modelling by means of extended Petri nets have been focused not only on the development of adequate time extensions, but also on enhancement of functional capabilities of the nets. The most representative functional enhancements of Petri nets are Environment/Relationship Nets (ER Nets) [11, 9] and Coloured Petri Nets (CP-nets) [16]. Both of the aforesaid enhancements have time extensions that can be used in a simulation of a time flow in a modelled system. The functional extension of the nets lies in the development of the idea of classical tokens, transitions and arcs. The token in extended Petri nets (or high-level Petri nets) may be represented by a record, mathematical expression or a function. Moreover, the firing of the transition may be executed as a calculation of stated mathematical formulas (ER nets), or as an interpretation of programming language formulas (CP-nets). Additionally, CP-nets provide the arc expressions that may be attached to the arcs of the net, as well as an ability to construct a hierarchical model of the net (Hierarchical Coloured Petri nets). Generally, high-level Petri nets enable to model complex systems in a concise way without losing minuteness of detail. Besides, high-level Petri nets provide new capabilities of representing time in a modelled system. The following subsections present the time extensions of ER nets (Time ER Nets and Time Basic Nets) [11, 9] and Timed Coloured Petri Nets [1, 16], respectively.

5.1. *Time Environment/Relationship Nets, Time Basic Nets*

Time Environment/Relationship Nets (TER Nets) proposed by Ghezzi, *et al.* [11] are time extensions of Environment/Relationship Nets (ER Nets). Before we introduce TER Nets and Time Basic Nets (TB Nets), we will briefly present the most important properties of ER Nets [11].

ER Nets are high-level Petri nets where tokens are functions associating values to variables, called environments. Additionally, each transition has an action associated. The action describes which input tokens can participate in a firing and which possible tokens are being produced during the firing.

Formally, a ER Net [11] is a net where:

- Tokens are environments on ID and V, i.e., possibly partial, functions: $ID \rightarrow V$, where ID is a set of identifiers and V a set of values. Let $ENV = V^{ID}$ be the set of all environments.
- Each transition t is associated with an action. The action is a relationship $\alpha(t) \subseteq ENV^{k(t)} \times$ $\times\ ENV^{h(t)}$. Here $k(t)$ and $h(t)$ denote cardinalities of the preset and postset of the transition t, respectively (we consider only arcs with weight 1). It is intended that $\alpha(t)$ refers to each input and output place of transition t. The projection of $\alpha(t)$ on $ENVk(t)$ is denoted by $\pi(t)$ and is called the predicate of transition t.
- A marking m is an assignment of multi-sets of environments to places.
- A transition t is enabled in a marking m iff, for every input place p_i of t, there exists at least one token env_i such that $< env_1, ..., env_{k(t)} > \in \pi(t)$. $< env_1, ..., env_{k(t)} >$ is called an enabling tuple for transition t.

- A firing is a triple $x = < enab, t, prod >$, such that $< enab, prod > \in \alpha(t)$. The *enab* is called the input tuple, while *prod* is called the output tuple.
- The occurrence of a firing $< enab, t, prod >$ in a marking m consists of producing a new marking m' for the net. Marking m' is obtained from the marking m by removing the enabling tuple *enab* from the input places of transition t and storing the tuple *prod* in the output places of transition t.

In Time ER Nets, it is assumed that each environment contains a variable, called chronos, whose value is of numerical type, representing the time stamp of the token. The actions associated with the transitions are responsible for producing time stamps for the tokens that are inserted in the output places, based on the values of the environments of the chosen input enabling tuple. The basic idea is that a time stamp represents the time when the token was produced. Both weak and strong timing models have been developed for TER Nets. Our further considerations will be concerned with the behaviour of TER Nets under the strong timing semantics (However, it should be noted that Ghezzi, *at al.* [11] provide an interesting interpretation of the weak time model of the nets.). In the strong time model of TER Nets it is assumed that if transition is enabled and remains enabled for all the possible time values at which it can fire, then it must fire. Moreover, a global timer controls the firing of the transitions, so each subsequently fired transition produces tokens with increased time stamp. Generally, in Strong TER Nets (STER Nets) the enabled transitions must fire when they are enabled and all firing sequences must be time ordered.

Time Basic Nets (TP Nets) are a particular case of TER Nets where each token (an environment) is associated only with a time stamp (a chronos variable), representing the time at witch the token has been created by a firing. Each transition is associated with a time-function, which describes the relation between the time stamps of the tokens removed by the firing and the time stamps of the tokens produced by the firing.

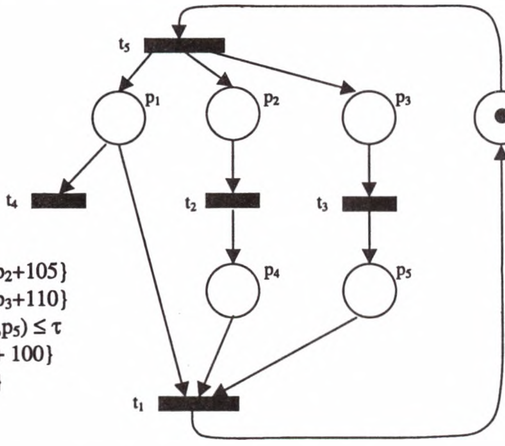
A formal definition of a TB Net according to Felder, *et al.* [9] can be formulated as follows.

A TB Net is a 6-tuple $(P, T, \Theta, F, tf, m_0)$, where:

- P, T, and F are the sets of places, transitions, and arcs of a net, respectively;
- Θ is a numerical set, whose elements are the timestamps that can be associated with the tokens; a timestamp of the token represents the time at which it was created;
- tf is a function that associates the function tf_t (called the time function) with each transition t; let *en* denote a tuple of tokens, one for each place in the preset of transition t; function tf_t associates with each tuple *en* a set of values θ ($\theta \in \Theta$), such that each value in θ is not less than the maximum of the time stamps associated with the token belonging to tuple *en*. $\theta = tft(en)$ represents the set of possible times at which transition t can fire, if enabled by tuple *en*; when transition t fires, the firing time of t under tuple *en* is arbitrarily chosen from the set of values θ; the chosen firing time is a value of the time stamps of all the tokens produced;
- m_0, the initial marking, is a function associating a (finite) multi-set of tokens with each place; in general, we use function m to denote a generic marking of nets, i.e., $m(p)$ denotes the multi-set of tokens associated with place p by marking m.

Like in TER Nets, TB Nets can be analysed under strong or weak time semantics. Our further consideration will be concerned with strong time semantics, which means that a transition must fire if it reaches its maximum firing time unless it is disabled earlier by firing of the other transition.

Figure 6 provides a TB net under the strong time semantic which models the behaviour of the simple embedded controller discussed in the paper. Timing constraints, presented in TB Nets in the form of mathematical formula, define possible time intervals (or time sets) within which the transitions must fire. It should be noticed that the formula uses time stamps of the tokens, as well as a set of mathematical functions associated to the transitions of the net, to determine the time when a transition can or finally must fire. For example, the computations one of the threads must last between 80 time units and 105 time units after the arriving of the token in place p_2. Although timing constraints are attached only to the transitions of TB Nets, the ability to formulate a fairly advanced mathematical formula representing the time flow in the system enable to reduce the Time Petri Net model of the discussed system without losing its functionality (Compare Time Petri Nets – section 2). The firing of transition t_1 depends on the time stamps of the tokens that reside in places p_1, p_4 and p_5. The conjunction of two mathematical formulas included in time function tf_{t1} ensures the time synchronisation of the modelled system.



$tf_{t5}(p_6) = \{\tau \mid \tau = p_6\}$
$tf_{t2}(p_2) = \{\tau \mid p_2+80 \leq \tau \leq p_2+105\}$
$tf_{t3}(p_3) = \{\tau \mid p_3+50 \leq \tau \leq p_3+110\}$
$tf_{t1}(p_1,p_4,p_5) = \{\tau \mid \max(p_4,p_5) \leq \tau$
$\text{and } \tau = p_1 + 100\}$
$tf_{t4}(p_3) = \{\tau \mid \tau = p_1 + 101\}$

Fig. 6. TB Petri Net model of a simple embedded system

The ability to formulate time constraints by means of mathematical formulas is a very important feature of TB Nets. The time constraints influence on the behaviour of the local part of the net. The crucial factor is the timestamp of a token that denotes the moment at which the token was created. However, strong time semantics for TB Nets assumes global time synchronisation for the firings of transitions. According to [11, 9] the semantics of TB Nets enables successfully model time extensions of Petri nets with timing constraints associated with transitions or places. Basic Time ER Net and TB Net models did not provide to construct hierarchical models, however latest research led to development of Hierarchical Timed ER Nets [33].

Apart from the simulation, the formal verification and analysis of Time ER Nets and TB Nets may relay on the construction of the Time Reachability Tree (TRT) [9]. A TRT is a (generally infinite) tree that describes all possible reachable markings. The procedure that builds it is based on symbolic execution. A symbolic state is defined by symbolic values for token timestamps and by predicate specifying a constraint on the symbolic values. The symbolic state stands for all actual states where the timestamps satisfy the constraint. The Time Reachability Tree enables to derive the firing set of possible values for a timestamp in a form of set of inequalities and algebraic formulas.

Hierarchical Time ER Nets may be modelled and analysed by CABERNET [33] (Politecnico di Milano, Italy) software tool. The modelling and analysis of TB nets is supported by MERLOT [34] (Politecnico di Milano, Italy) programme. The aforesaid software tools enable to model Time ER and TB nets and implement the reachability analysis algorithm based on the Time Reachability Tree.

Recent research concerning Time ER Nets and TB Nets that led to development of the software tools as well as the hierarchical nets allows the application of the nets in the modelling and verification of a wide range of hard real-time systems.

5.2. Timed Coloured Petri Nets

Timed Coloured Petri nets are time extensions of Coloured Petri Nets. A thorough study concerning Coloured Petri Nets, as well as Timed Coloured Petri Nets, can be found in [16]. Before we introduce Timed Coloured Petri Nets, we will informally present basic concepts of Coloured Petri Nets.

Coloured Petri Nets (CP-Nets) are high-level nets where each token is equipped with an attached data value – the token colour. The data value may be freely complex (e.g. integer number or record of data). For each place, a colour set is defined which characterises acceptable colours of the token in the place. All declarations concerning the behaviour of the net and colours of tokens are written in the CPN ML language. The distribution of tokens in the places is called marking. The initial marking determines the initial state of the net and is specified by initialisation expressions. The marking of each place is a multi-set over the colour set attached to the place. Moreover, arc expressions (the expressions that may be attached to arcs) decide on the token flow in the net. Arc expressions may denote the number and the colour set of flowing tokens, as well as may be functions that manipulate tokens and their colours. Before the occurrence of a transition, the variables defined in the net are bounded to colours of corresponding types, which is called a binding. A pair (t, b) where t is a transition and b a binding for t is called a binding element. For each binding, it can be checked if the transition is enabled to fire in a current marking. An enabled transition may occur. The transitions of the net may have guards attached to them. The guards are boolean expressions attached to a transition that may define additional constraints that must be fulfilled before the transition is enabled. For the effective modelling Coloured Petri Net enable to distribute parts of the net across multiple subnets. The ability to define the subnets enables to construct a large Hierarchical CP-Net by composing a number of smaller nets. Hierarchical CP-Nets offer two mechanisms for interconnecting CP-Net structure on different layers: substitution transitions and fusion places. A substitution transition is a transition that stands for a whole subnet of net structure. A fusion place is a place that has been equated with one or more other places, so that the fused places act as a single place with a single marking. Each hierarchical CP-Net can be translated into behaviourally equivalent non-hierarchical CP-Net, and vice versa. Thanks to the ability to handle additional information or data in a Coloured Petri Net structure and thanks to introduction of the hierarchy, the nets manage to model complex systems in a consistent way.

Two main time extensions of Coloured Petri Nets have been proposed. Van der Aalst introduced Interval Timed Coloured Petri Nets [1]. Each transition in an Interval Timed Coloured Petri Net may contain a statically associated time interval, which was interpreted as a possible firing delay of the transition. Firing of a transition incremented a time stamp of the token by a value from the interval associated with the transition. An enabled transition might occur at time x if all the tokens to be consumed had a time stamp not later than time x.

The second time extension of Coloured Petri Nets (called Timed Coloured Petri Nets) has been proposed by Jensen [16]. Jensen introduced a global clock whose values represent the model time. Tokens may carry a time value, also denoted as a time stamp. The time stamp represents the earliest model time at which the token can be used. Hence, to occur, a transition must be both colour enabled and ready. It means that the transition must fulfil the usual enabling rule and all the time stamps of the removed tokens must be less than or equal to the current model time. When a token is created, the time stamp is specified by an expression. This means that it is possible to specify all kinds of delays (e.g. constant, interval, or probability distribution). Moreover, the delay may depend upon the binding of the transition that creates the token. In the following part of the subsection we will concentrate on Jensen's model of Timed Coloured Petri Nets, which seems to be more general than the model proposed by van der Aalst.

Although for none of time extensions of Coloured Petri Nets clear strong or weak time semantics has been proposed, we would like to take notice of the execution of a Timed Coloured Petri Net published in [16, 19, 20]. The system remains at a given model time, as long as there are colour enabled binding elements that are ready for execution. When no more binding elements can be executed at the current model time, the system increments the clock to the next model time at which binding elements can be executed. Each marking exists in a closed interval of the model time (which may be a point, i.e., a single moment). The occurrence of a binding element is instantaneous. Hence, a Timed Coloured Petri Net may be interpreted as the net that is executed under the earliest and maximum occurrence (firing) rule (The rule seems to be similar to the one proposed by Starke in [28]). The aforesaid occurrence rule, in our opinion, enables to construct time mechanisms that can successfully model hard real-time systems.

According to [16] Timed Coloured Petri Nets can be defined formally as follows.

A Timed Coloured Petri Net is a tuple TCPN = $(\Sigma, P, T, A, N, C, G, E, I, R, r_0)$ satisfying the requirements below:

- Σ is a finite set of non-empty types, called colour sets;
- P is a finite set of places;
- T is a finite set of transitions;
- A is a finite set of arcs such that:
  $P \cap T = P \cap A = T \cap A = \varnothing$;
- N is a node function; it is defined from A into $P \times T \cup T \times P$;
- C is a colour function; it is defined from P into Σ;
- G is a guard function; it is defined from T into expressions such that:
  $\forall t \in T$: [Type$(G) = \mathcal{B} \wedge$ Type(Var$(G(t))) \subseteq \Sigma$];

- E is an arc expression function; it is defined from A into expressions such that: $\forall a \in A$: [Type($E(a)$) = $C(p(a))_{MS} \wedge$ Type(Var($E(a)$)) $\subseteq \Sigma$], where $p(a)$ is the place of $N(a)$ and $C(p(a))$ is timed or untimed multi-set over $C(p(a))$;
- I is an initialisation function; it is defined from P into closed expressions such that: $\forall p \in P$: [Type($I(p)$) = $C(p)_{MS}$], where $C(p)$ is timed or untimed multi-set over $C(p)$;
- R is a set of time values, also called time stamps; it is a subset of $\mathcal{R}$closed under + and containing 0;
- r_0 is an element of R, called the start time.

The behaviour of the net is specified by a pair (M, r), where M is a marking (a timed multi-set over the set of all token elements) and r is a time value.

Figure 7 provides a Hierarchical Timed Coloured Petri Net model of the embedded controller discussed in the paper. Tokens task1 and task2 denote two separate control tasks that are executed simultaneously in the system. Tokens tim1, tim2 and tim3 denote three elements of the timer subsystem that all ensure the 100 or 101 cycle period of the modelled system or cause the firing of transition "Process time-out fault-stop" which places token timeout_err in place "Time-out error". Tokens get time stamps via expressions called delay expressions. A delay expression has the form: @+ *expression* where "@+" appears literally, and *expression* is an arithmetic expression. A delay expression may be associated with a transition (e.g. "Process tasks") or can be appended to an arc inscription (e.g. the output arc of transition "Process timer1"). The occurrence of transition "Process tasks" may be interpreted as an execution of computations by the tasks. The time period associated with the execution of the task1 and task2 is determined by the result of the function ran't1_delay() or ran't2_delay(), respectively. Probabilistic distribution of time stamps is used to realistically determine possible delays. The control task and the timer task co-operation are simulated by the occurrence of transition "Synchronise". The occurrence of transition "Process time-out timers 2, 3" removes unused tokens tim2 and tim3 in case of a successfully accomplished cycle of the controller.

The current marking of the net presented in Figure 7 enables the occurrence of a binding elements: b_1 = <"Synchronise", (1`tim1, 1`task1, 1`task2) >, $b_2$ = <"Process time-out fault-stop", (1`tim1, 1`tin[illegible]> or $b_3$ = <" Process time-out timers 2, 3", (1`tim2, 1`tim3)>. The time stamp associated with the token in place "Timer 1 executed" is equal to the model time (200 time units). When we assume the earliest and maximum rule of occurrence of the net, transition "Synchronise" must occur at the current state of the system. Simultaneously, the time stamps of tokens tim2 and tim3 (201 and 202 time units) are higher than the current time model what prevents from the occurrence of transition "Process time-out fault-stop". Transition "Process time-out fault-stop" occurs only if the time stamp of token task1 or task2 is higher than the time stamp of token tim1, what means that the execution of at least one of the control tasks exceeded a stated controller's cycle of computations.

The Timed Coloured Petri Net model of the embedded controller differs from the models previously presented in the paper. On one hand the part of the net that models the computations of the tasks became reduced, on the other hand the part of the net that models the timer had to be extended. The reduction results from an ability to define the colours of the tokens. The extension results from the time model of the nets.

```
color PROCESS = with task1 | task2 | tim1 |tim2 | tim3 timed;
color TIMEOUT_ERR = with timeout_err;
var proc: PROCESS;
val Tmint1=80; val Tmaxt1=105;
val Tmint2=50; val Tmaxt2=110;
color t1_delay = int with Tmin1..Tmax1 declare ran;
```

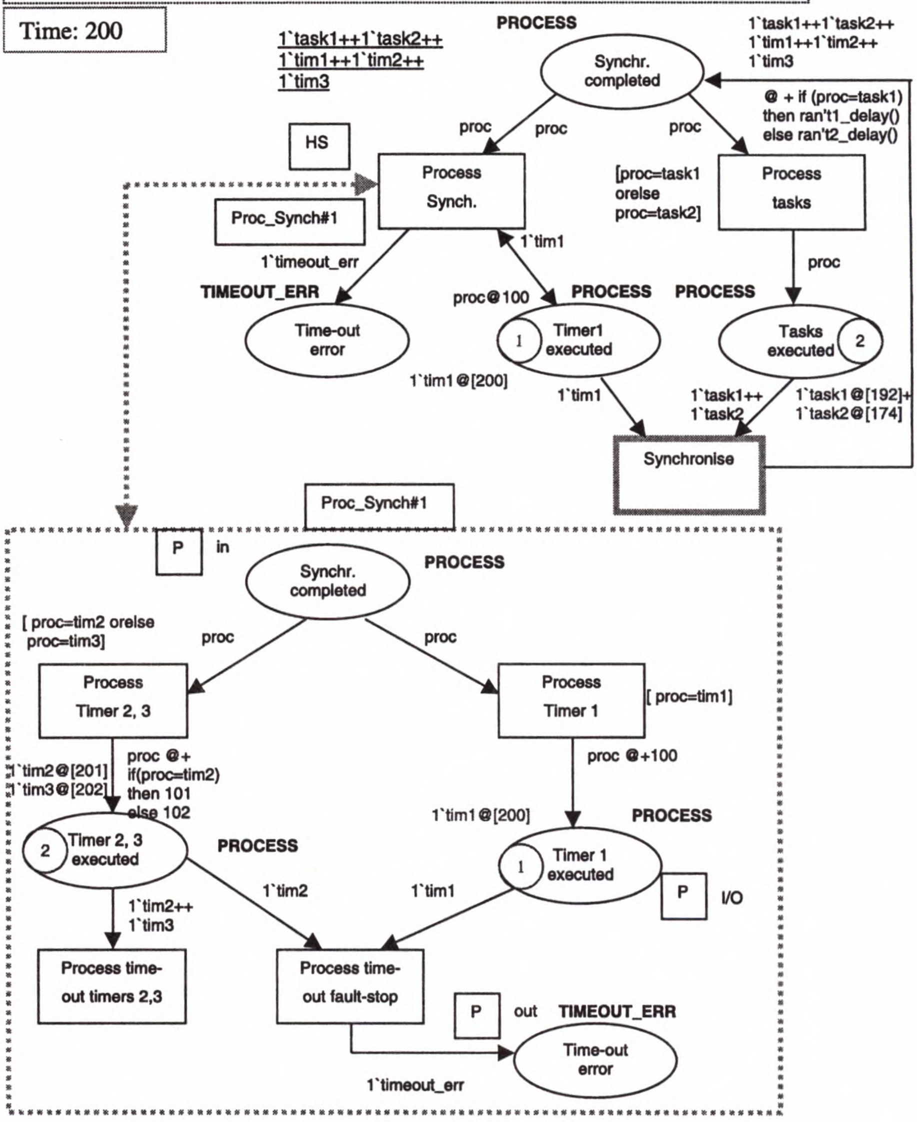


Fig. 7. Timed Coloured Petri Net model of a simple embedded system[1)]

[1)] The net was edited using CPN/Desing tool

The time stamps of the tokens are naturally interpreted as the duration of the computation, whereas in the previously presented nets (sections 2, 3, 4, 5.1) the timing constraints were interpreted as the enabling duration (the time interval within which the transition must occur or fire) [3, 4]. However, the hierarchical approach to representation of the system permits hiding the extended time synchronisation model into one substitution transition ("Process synch.") (Fig.7). As a result we have obtained a fairly simple high-level model of the system supplemented by low-level sub-net, which implements a time-out fault-stop mechanisms.

Timed Coloured Petri Nets seem to be one of the most flexible and general time extensions of high-level Petri nets. The nets combine the ability to represent large systems in a concise way (the possibility to construct hierarchical model of a system) with a simple and flexible model of the time.

Timestamps associated with the tokens influence on the local behaviour of the net with respect on the state of global clock. The time model of Timed CP-Nets assumes global synchronisation of occurrences. Systematically developed and easy to obtain "CPN/Design" [19, 20] and "CPN Tools" programs enable to provide tools for analysis and simulation of Coloured Petri Net models of developed systems.

The main method of verification of Timed CP-Net models is the construction of timed occurrence graph [16]. Each node of the graph contains a time value with timed marking and each arc denotes occurring binding element. For a non-cyclic CP-Net, a timed occurrence graph will often be much smaller than the corresponding untimed occurrence graph, because the time constraints limit the possible orders in which binding elements may occur. However, if we use non-deterministic time delays, with a large number of possible time values, we may get a larger occurrence graph. For a cyclic system a timed occurrence graph usually become infinite. The way of solving the explosion of states in the graph is a construction of subgraphs that enable to analyse parts of modelled system.

6. Conclusions

In this paper we have discussed the abilities to specify and verify hard real-time systems by means of time extensions of Petri nets. All presented time extensions of the nets meet requirements fixed in the first section of the paper: the so-called strong firing semantics and the ability to represent time as time intervals within which events must occur. Consequently, we omitted several published time extensions of the nets as Timed Petri Nets or Stochastic Petri Nets that, in our opinion, can not successively be used for hard real-time system specification. (However, some of them were successively used in the hard real-time system performance evaluation, e.g. [8].)

The strong firing semantics and hard timing constraints that influence on the firing of transitions in the nets cause that the nets must be considered under the global mechanism of time synchronisation. The local firing or occurrence rules of transitions are always globally synchronised in the time domain. Hence, we can always obtain strict state of modelled system with respect to the executing or fulfilled tasks as well as the time.

The introduction of time extensions of Petri nets led to a significant development of real-time system modelling methodologies. However, the greater experssivity of the nets result in difficulties in formal verification of systems modelled. The enabling time of transition characterised by a time interval may cause the infinite amount of states of the net after the transition firing. Consequently, the new methods of analysis had to be proposed for the nets. Most of them were pointed out in the paper. In general, the analysis methods of time extensions of Petri nets use the idea of extended reachability graph. In comparison with classical Petri net reachability graph, the extended one usually includes the information about the time domain of the net and sets of possible markings after the firing of the transition. Additionally, for some extensions of the nets, place or transition invariant methods of analysis were proposed [2, 16].

For almost all presented in the paper time extensions of Petri Nets (except from Statically Timed Petri Nets) adequate software tools were developed. The tools manage to simulate behaviour of the modelled net as well as provide a possibility to formally analyse and verify the net by means of selected analysis methods. Even if the net cannot be successively analysed by means of the formal method, the visualised and registered by software tool execution of the net may bring significant information about the properties of modelled system or object.

At present, time extensions of Petri Nets are used as a specification semi-language for real-time system developing. The Petri Net models enable the simulation and the partial formal analysis of developed systems. As the modern real-time systems become seriously complicated, only the net semantics that provide the possibility to construct the hierarchical nets seems to be applicable enough. Hence, Time Hierarchical ER/TB Nets and Timed Hierarchical Coloured Petri Nets seem to be the most adequate time extensions for modern real-time system modelling. However, it should be mentioned that the most popular hierarchical time extension of the nets are Timed Coloured Petri Nets. Additionally, high-level Petri nets provide a new approach to time modelling.

Table 1

Basic properties of time extensions of Petri nets for hard real-time systems specification and verification

Property	Time Petri Nets	Arc Time Petri Nets	Statically Timed Petri Nets	Time ER/TB Nets	Timed Coloured Petri Nets
Firing rule	Strong	Strong	Strong/Weak	Strong/Weak	Strong, earliest, maximum
Timing constraint type	Static interval	Static interval	Static interval	Mathematical formula	Single constant/ stochastic value
Timing constraint attachment	Transition	Transition's incoming arc	Transition, place, incoming arc	Transition	Transition, transition's outgoing arc
Timing constraint meaning	Enabling duration	Enabling duration	Enabling duration	Enabling duration	Firing duration
Time synchronisation	Global	Global	Global	Global	Global
Formal analysis method	Extended reachability graph	State graph	–	Time Reachability Tree	Timed occurrence graph
Software tools for simulation and analysis	INA, ARP	ATNA, INA	–	CABERNET/ MERLOT	CPN/Design, CPN Tools
Ability to construct hierarchical models	Yes	No	No	No	Yes

The ability to use mathematical or programming language formulas to model the time in the nets enables to construct more complex and flexible models of a wide range of real-time systems.

A breakdown of the basic properties of time extensions of Petri nets presented in the paper is included in Table 1.

We can conclude that the modern time extensions of Petri nets with the strong firing semantics may be designed to model and verify hard real-time systems. Apart from time extensions of classical Place/Transitions Nets, time/timed high-level Petri nets seem to provide a very promising methodology of time modelling. However, it must be noted that time extensions lead to significant complexity in mathematical analysis of the net behaviour. Hence, in our opinion, future research concerning time extensions of Petri nets will concentrate on the following problems. Firstly, an adequate methodology of hard real-time systems modelling by means of time extensions of Petri nets should be proposed. The methodology might provide the rules of hard real-time system specification as well as the net models of adequate time mechanisms crucial for a system development. Secondly, formal approach to the complex analysis and verification of Petri net models of the systems should be developed. The approach might enable to formally prove selected properties of modelled systems.

References

[1] van der Aalst W.M.P.: *Interval Timed Coloured Petri Nets and their Analysis.* Application and Theory of Petri Nets 1993, Proc. of the 14th International Petri Net Conference, Chicago 1993, Lecture Notes in Computer Science, vol. 691, Springer-Verlag 1993, 452–427

[2] Berhtomieu B., Diaz M.: *Modeling and Verification of Time Dependent Systems Using Time Petri Nets.* IEEE Transactions on Software Engineering, vol. 17, No. 3, March 1991, 259–273

[3] Bowden F.D.J.: *Modelling Time in Petri Nets.* Proc. of the Second Australia-Japan Workshop on Stochastic Models in Engineering, Technology and Management, Gold Coast, Australia, July 1996

[4] Bowden F.D.J.: *A Brief Survey and Synthesis of the Roles of Time in Petri Nets.* Mathematical and Computer Modelling, 1999

[5] Burkhard H.D.: *Ordered Firing in Petri Nets.* Journal of Information Proc. and Cybernetics EIK, 17, 1981, 71–86

[6] Cerone A.: *A Net-Based Approach for Specifying Real-Time Systems.* Dipartamento di Informatica, Universita di Pisa 1993 (Ph.D., Thesis, TD-16/93)

[7] Cerone A., Maggiolo-Shettini A.: *Time-Based Expressivity of Time Petri Nets for System Specification.* Theoretical Computer Science, vol. 216, 1999, 1–53

[8] Coolahan J.E. Jr., Roussopoulos N.: *Timing Requirements for Time-Driven Systems Using Augmented Petri Nets.* IEEE Transactions on Software Engineering, vol. SE-9, No. 5, September 1983, 603–616

[9] Felder M., Ghezzi C., Pezze M.: *Formal Specyfication and Timing Analysis of High-Integrity Real-Time Systems.* Real Time Computing, NATO ASI Series, vol. 127, 1992, 187–211

[10] de Frutos-Escrig D., Ruiz V.V., Alonso O.M.: *Decidability of Properties of Timed-Arc Petri Nets.* Application and Theory of Petri Nets 2000, 21st International Conference, ICATPN 2000, Aarhus, Denmark, June 26–30, 2000, 187–206

[11] Ghezzi C., Mandrioli D., Morasca S., Pezze M.: *A Unified High-Level Petri Net Formalism for Time-Critical Systems.* IEEE Transactions on Software Engineering, vol. 17, No. 2, February 1991, 160–172

[12] Hanisch H.M.: *Analysis of Place/Transition Nets with Timed Arcs and its Application to Batch process Control.* Application and Theory of Petri Nets 1993, Proc. of the 14th International Petri Net Conference, Chicago 1993, Lecture Notes in Computer Science, vol. 691, Springer--Verlag 1993, 282–299

[13] Hanisch H.M., Thieme J., Lautenbach K., Simon C.: *Timestamp nets in technical applications.* Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC'98), San Diego, CA, 11–14 October 1998, 119–124

[14] Halang W.A., Sacha K.M.: *Real-Time Systems.* London, World Scientific Publishing Co. 1992

[15] Holliday M.A., Vernon M.K.: *A Generalized Timed Petri Net Model for Performance Analysis.* IEEE Transactions on Software Engineering, vol. SE–13, No. 12, December 1987, 1297–1310

[16] Jensen K.: *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. vol. 1–3.* Berlin, Springer-Verlag 1995/96

[17] Leveson N.G., Stolzy J.L.: *Safety Analysis Using Petri Nets.* IEEE Transactions on Software Engineering, vol. SE-13, No. 3, March 1987, 386–397

[18] Merlin P.M., Farber D.J.: *Recoverability of Communication Protocols – Implications of a Theoretical Study.* IEEE Transactions on Communications, September 1976, 1036–1043

[19] Meta Software Corporation: *Design/CPN Tutorial for X-Windows.* Meta Software 1993

[20] Meta Software Corporation: *Design/CPN Reference Manual for X-Windows.* Meta Software 1993

[21] Murata T.: *Petri Nets: Properties, Analysis and Applications.* Proc. of the IEEE, vol. 77, No. 4, 1989, 541–580

[22] Peterson J.L.: *Petri Net Theory and The Modelling of Systems.* Englewood Cliffs, Prentice--Hall, 1981

[23] Ramamoorthy C.V., Ho G.S.: *Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets.* IEEE Transactions on Software Engineering, vol. SE-6, No. 5, September 1980, 440–449

[24] Ramchandani C.: *Analisys of Asynchronous Concurrent Systems by Petri Nets.* Project MAC, MAC-TR 120, MIT 1974 (Ph.D. Thesis)

[25] Roch S., Starke H.P.: *INA Integrated Net Analyzer, Version 2.2, Manual.* Berlin, Humboldt--Universität zu Berlin, Institut für Informatik, Lehrstuhl für Automaten und Systemtheorie 1999

[26] Sacha K.: *Projektowanie oprogramowania systemów wbudowanych.* Warszawa, Politechnika Warszawska 1996 Prace Naukowe Elektronika, z. 115,

[27] Sifakis J.: *Performance Evaluation of Systems Using Nets.* Net Theory and Applications, Proc. of the Advanced Course on General Net Theory of Processes and Systems, Hamburg, October 1979, Lecture Notes in Computer Science, vol. 84, Springer 1980, 307–319

[28] Starke P.H.: *Some Properties of Timed Nets under the Earliest Firing Rule.* Lecture Notes in Computer Science, vol. 424, Advances in Petri Nets 1989, Germany, Springer-Verlag 1990, 418–432

[29] Starke P.H.: *A Memo on Time Constraints in Petri Nets.* Informatik-Bericht, Nr 46

[30] Störrle H.: *An Ealuation of High-End Tools for Petri-Nets.* Ludwig-Maximilians-Universität München, Institut für Informatik, June 1998

[31] Szmuc T.: *Zaawansowane metody tworzenia oprogramowania systemów czasu rzeczywistego.* Kraków, Cracow Centre for Advanced Training in Information Engineering, vol. 15, 1998

[32] Walter B.: *Timed Petri Nets for Modelling and Analysis Protocols with Real-Time Characteristics.* Proc. 3rd IFIP Workshop on Protocol Specification, Testing, and Verification, North--Holland, Amsterdam 1983, 149–159

[33] http://www.daimi.au.dk/PetriNets/classification/tools/cabernet.html

[34] http://www.daimi.au.dk/PetriNets/classification/tools/merlot.html