Integrating Implementation Properties in Analysis of Petri Nets Handling Exceptions

Hélène Leroux, Karen Godary-Dejean, David Andreu

12th IFAC/IEEE Workshop on Discrete Event Systems
Cachan, France. May 14-16, 2014

Integrating implementation properties in analysis of Petri nets handling exceptions

Helene LEROUX * **Karen GODARY-DEJEAN** **
David ANDREU *

* *DEMAR INRIA LIRMM, Montpellier II University, Montpellier, FRANCE(leroux,andreu@lirmm.fr).*
** *LIRMM, Montpellier II University, Montpellier, FRANCE(godary@lirmm.fr).*

Abstract: To design and implement complex digital systems, designers need to have an efficient methodology. In this goal, HILECOP has been developed to transform automatically Petri nets in a VHDL code. To ease design and increase the reactivity of exception handling, the mechanism of macroplace has been added to the formalism of Petri nets. This article describes an automatic model transformation for the analysis step. It integrates implementation properties to enhance reliability.

Keywords: Formal verification, Exception, Petri-nets, Implementation, Discrete-event systems

1. CONTEXT

To design and implement complex digital systems, designers must have an efficient and reliable design process. Hence a methodology, called HILECOP (High level hardware component programming) has been developed [Souquet et al. (2008)]. This component-based approach allows designers to easily handle complex digital architecture. The components and their compositions are described thanks to Interpreted Time Petri Nets (ITPN) [Leroux et al. (2013)]. It allows to benefit from their intuitive graphical representation, but also formal and structural analyses capabilities of Time PN. Once the components and their interactions have been defined, the initial model is automatically transformed (model-to-text transformations) resulting into two different models: the implementation model (IM) written in VHDL and the analysis model (AM) written in PNML. Since HILECOP does not contain analysis facilities, the AM is used in existing analysis tools to validate and optimize the implementation. The IM will be implemented on FPGA devices.

Several other methodologies to translate PN models in VHDL have been developed [Silva et al. (2010)][Tkacz and Adamski (2012)]. The advantage of the HILECOP one is that it handles generalized T-time PN whereas other methodologies deal with binary non-time PN. Furthermore, as far as we know, the question of the correspondence between the analyzed model and the implementation has not been treated in these methodologies.

The HILECOP methodology has been successfully used in industrial applications, especially in the field of implantable active medical device [Andreu et al. (2009)]. Yet designers encountered some issues for handling exceptions in a reactive and efficient way [Leroux et al. (2013)]. Hence the PN formalism is enhanced with a new mechanism to deal with exceptions: the macroplace (MP). The concept of MP or exception handling for PN have already been studied [Holvoet and Verbaeten (1995)][de Oliveira et al. (2002)] but not in an automatized methodology. Moreover existing MP do not satisfy all of our constraints: it must among other things preserve the conformity and efficiency of the implementation but also the analyzability of the model with existing analysis tools.

The analysis is notably used to guarantee the behavior. To have confident validation results, the behavior of the AM must include every possible implementation behavior. Now the analysis complexity of the AM has to be contained to face the classical combinatorial explosion problem, a fortiori dealing with industrial size models. The goal of this article is then to introduce efficient model transformations from the initially designed model to both the AM and the IM. First, the formalism of ITPN with macroplaces is defined. Second, its implementation is described. Then the generation of the analysis model is explained. Last, obtained results are presented.

2. ITPN WITH MACROPLACE

2.1 Definition

The formalism used to describe a HILECOP-component behavior is generalized Interpreted T-time Petri Nets with test and inhibitor arcs and with macroplaces (cf Fig. 1). Broadly speaking, the model is composed of a main PN which is an ITPN (described in Leroux et al. (2013)) and macroplaces (MP). A MP is an entity containing an ITPN called refinement and is represented by a double ellipse. The main PN and MP are linked by specific arcs: entry and exit arcs represented by dashed arrows. A situation is associated to these arcs to describe the interaction between a transition and a MP (cf Fig. 1). The situation (*) can be associated to an exit arc to describe an exception arc. An exception transition is a transition targeted by an exception arc as t_{exc} in Fig.1.

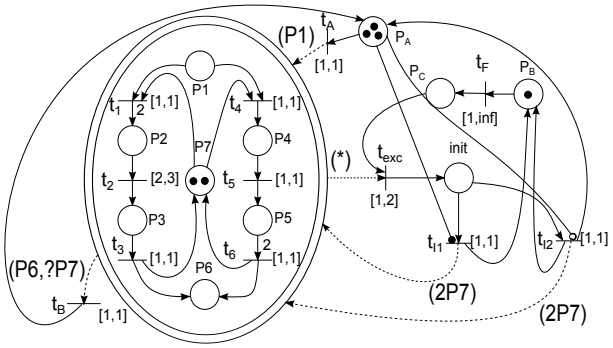

Fig. 1. Example of ITPN with macroplace

Our notations and some definitions are inspired from [Traonouez et al. (2009)] and [Berthomieu et al. (2007)]. Let I^+ be the set of non empty real intervals with non negative rational endpoints. For $i \in I^+$, $\downarrow i$ is its left end-point and $\uparrow i$ is its right end-point or ∞ if i is unbounded. The interpretation of a ITPN consists of conditions, functions and actions. Let $\mathcal{C}$, $\mathcal{F}$, $\mathcal{A}$ be respectively the set of conditions, functions and actions described in VHDL. They interact with inputs and outputs of the system or manipulate internal variables. ϵ means that there is no condition nor function linked to a transition nor action linked to a place.

An ITPN with MP is a tuple $< P, T, M, Pre, Pre_t, Pre_i, Post, Entry, Exit, m_0, I_s, C, F, A >$, in which :

- P, T are respectively the set of places and transitions of the main PN, M is the set of MP.
- m_0 is the initial marking.
- $mp \in M$ is an ITPN defined with $< P^{mp}, T^{mp}, Pre^{mp}, Pre_t^{mp}, Pre_i^{mp}, Post^{mp}, m_0^{mp}, I_s^{mp}, C^{mp}, F^{mp}, A^{mp} >$
- $P_{all} = P \cup \bigcup_{mp \in M} P^{mp}$, $T_{all} = T \cup \bigcup_{mp \in M} T^{mp}$.
- $Pre, Pre_t, Pre_i, Post : T \to P \to \mathbb{N}$ are respectively the precondition function, the test function, the inhibition function and the postcondition function.
- $\forall mp \in M, Pre^{mp}, Pre_t^{mp}, Pre_i^{mp}, Post^{mp} : T \to P^{mp} \to \mathbb{N}$ are respectively the precondition function, the test function, the inhibition function and the postcondition function of the refinement of the MP.
- $Entry = (Pre_M, Pre_{Mt}, Pre_{Mi})$ and $Exit = (Post_M, Post_{exc})$ are the description of entry and exit situations.
- $Pre_M, Pre_{Mt}, Pre_{Mi}, Post_M : T \to \bigcup_{mp \in M} P^{mp} \to \mathbb{N}$ are the entry precondition function, the entry test function, the entry inhibition function and the entry postcondition function, respectively.
- $Post_{exc} : T \to M \to \mathbb{B}$ is the exception function. It is equal to 1 when there is an exception arc between a MP and a transition, 0 otherwise.
- $I_s : T_{all} \to I^+ \cup \varnothing$ is the static interval function.
- $C : T_{all} \to \mathcal{C} \cup \epsilon \to \mathbb{B} \cup \epsilon$ is the condition function.
- $F : T_{all} \to \mathcal{F} \cup \epsilon$ is the impulsive action function.
- $A : P_{all} \to \mathcal{A} \cup \epsilon$ is the continuous action function.

Moreover, a place or a transition cannot be in the main PN and in a refinement ie : $\forall mp \in M, P \cap P^{mp} = \emptyset$ and $T \cap T^{mp} = \emptyset$; a place or a transition cannot be in two different refinements, ie: $\forall (mp, mp') \in M^2, P^{mp} \cap P^{mp'} = \emptyset$ and $T^{mp} \cap T^{mp'} = \emptyset$. Also, if there is an exception arc between a MP and a transition there cannot be an other exit arc between them, ie : $\exists (mp, t) \in (M, T) \backslash Post_{exc}(t)(mp) = 1 \Rightarrow \forall p \in P^{mp}, Post_M(t)(p) = 0$.

An entry situation is written such as $(\{n_i P_i\})$ where $n_i \in \mathbb{N}^+$ and $P_i \in P^{mp}$. An exit situation is written either such as: $(\{X n_i P_i\})$ where $n_i \in \mathbb{Z} \backslash \{0\}$, $P_i \in P^{mp}$ and $X \in \{\epsilon, ?\}$ or as $(*)$ to describe an exception. ? allows to describe test and inhibitor arcs.

Let $M_{exc}(t)$ be the set of MP linked to the transition t by an exception arc: $\forall mp \in M, mp \in M_{exc}(t) \Leftrightarrow Post_{exc}(t)(mp) = 1$. For $f, g : P \to \mathbb{N}, f \geq g$ means $(\forall p \in P)(f(p) \geq g(p))$ and $f\{+| - | = | \neq\}g$ maps $f(p)\{+| - | = | \neq\}g(p)$ with every p. The function $m : P_{all} \to \mathbb{N}$ is the marking. The marking of a MP mp is defined by the function $m^{mp} = m_{|P^{mp}}$. A transition $t \in T^{mp}$ is sensibilized by m iff $(m \geq Pre^{mp}(t) + Pre_t^{mp}(t)) \wedge (m < Pre_i^{mp}(t))$. A transition $t \in T$ is sensibilized by m iff $(m \geq Pre(t) + Pre_t(t) + Pre_M(t) + Pre_{Mt}(t)) \wedge (m < Pre_i(t) + Pre_{Mi}(t)) \wedge \forall mp \in M_{exc}(t), m^{mp} \neq 0$ where 0 is the null function. In these cases, we note $t \in sens(m)$. Let m' be the marking of the ITPN after the firing of t from the marking m. The set of transitions newly sensibilized is noted $\uparrow sens(m, t)$. A transition $k \in T$ is newly sensibilized by the firing of the transition t from the marking m iff :

$$\begin{cases} \text{if} & t \in T^{mp}, k \in sens(m') \\ & \wedge\, [(k = t) \vee k \notin sens(m - Pre^{mp}(t))] \\ \text{if} & t \in T, k \in sens(m') \\ & \wedge\, [(k = t) \vee k \notin sens(m - Pre(t) - Pre_M(t))] \end{cases}$$

A MP mp is active iff $m^{mp} \neq 0$ where 0 is the null function. A state of an ITPN is $s = (m, I)$ where m is the marking and I is a function called the time-interval function. Function $I : T_{all} \to I^+ \cup \varnothing$ associates a time-interval with every transition sensitized by m. A transition $t \in T_{all}$ is fireable from (m, I) if : $t \in sens(m) \wedge (C(t) = 1 \vee C(t) = \epsilon) \wedge 0 \in I(t)$. It is denoted *fireable(m)*.

2.2 Semantics

Because of interpretation, the strong semantic used for time PN cannot be used. Indeed, it cannot be guaranteed that, for a transition t, we have $C(t) = 1$ and $0 \in I(t)$ at the same time. Weak semantics is not used either as it does not allow to describe the emergency of some events. So we defined in [Leroux et al. (2013)] a new semantics for ITPN allowing to block a transition t if $C(t) = 0$ during all the time slot. It is unblocked when t becomes newly sensibilized. We adapted this semantics to ITPN with MP.

The main difference between the semantics of ITPN and the one of ITPN with MP is the firing of an exception transition. When an exception transition is fired, the marking of the MP linked to it is cleared, as well as the time counter of every transition of these MP and every transition having these MP in input. Let $T_{iexc}(t)$ be the set of transitions defined by : $\forall k \in T, k \in T_{iexc}(t) \Leftrightarrow \exists p \in \bigcup_{mp \in M_{exc}(t)} P^{mp} \backslash Pre_{Mi}(k)(p) = 1$. It is the transitions of the main PN linked to a cleared MP by only an inhibitor arc with a weight equal to one.

The semantics of an ITPN with MP is the timed transition system $< S, s_0, \rightsquigarrow >$ where:

•S is the set of states (m, I) of the ITPN with MP.
•$s_0 = (m_0, I_0)$ is the initial state, where m_0 is the initial marking and I_0 is the static interval function I_s restricted to the transitions sensitized by m_0.
• $\rightsquigarrow \subseteq S \times (T_{all} \cup \mathbb{R}^+) \times S$ is the state transition relation, defined as follows ($(s, a, s') \in \rightsquigarrow$ is written $s \overset{a}{\rightsquigarrow} s'$):

1. Discrete transitions: $(m, I) \overset{t}{\rightsquigarrow} (m', I')$ iff :

- $t \in T^{mp} \wedge t \in fireable(m)$
- $m' = m - Pre^{mp}(t) + Post^{mp}(t)$
- $(\forall k \in T_{all} \backslash k \in sens(m'))$, if $k \in\uparrow sens(m, t)$, $I'(k) = I_s(k)$, otherwise $I'(k) = I(k)$.

Or

- $t \in T \wedge t \in fireable(m)$
- $\forall p \in (P \cup \bigcup_{mp \in M \backslash M_{exc}(t)} P^{mp})$, $m'(p) = (m - Pre(t) - Pre_M(t) + Post(t) + Post_M(t))(p)$ and $\forall p \in (\bigcup_{mp \in M_{exc}(t)} P^{mp})$, $m'(p) = 0$
- $\forall k \in (T \backslash T_{iexc}(t) \cup \bigcup_{mp \in M \backslash M_{exc}(t)} T^{mp}) \backslash k \in sens(m')$, if $k \in\uparrow sens(m, t)$, $I'(k) = I_s(k)$, otherwise $I'(k) = I(k)$ and $\forall k \in (T_{iexc}(t) \cup \bigcup_{mp \in M_{exc}(t)} T^{mp})$, if $k \in sens(m')$, $I'(k) = I_s(k)$.

2. Continuous transitions: $(m, I) \overset{\theta}{\rightsquigarrow} (m, I')$ iff $\theta \in \mathbb{R}^+$ and:

- $(\forall t \in T_{all})\ I(t) \neq \varnothing \wedge t \in sens(m) \Rightarrow \theta \leq\uparrow I(t)$
- $(\forall t \in T_{all})\ I(t) \neq \varnothing \wedge t \in sens(m) \Rightarrow\downarrow I'(t) =\downarrow I(t) - \theta \wedge\uparrow I'(t) =\uparrow I(t) - \theta$
- $(\forall t \in T_{all})\ I(t) = \varnothing \Rightarrow I'(t) = I(t)$

3. Blocking transitions: $(m, I) \overset{t}{\rightsquigarrow} (m, I')$ iff $t \in T_{all}$ and:

- $t \in sens(m) \wedge (C(t) = 0) \wedge (\uparrow I(t) = 0) \Rightarrow I'(t) = \varnothing$
- $(\forall k \in T_{all} \backslash t)\ I'(k) = I(k)$

3. IMPLEMENTATION

3.1 Implementation of an ITPN

By definition, a PN is an asynchronous model. Yet an ITPN can contain functions and, on a FPGA, there is no easy way to know when the execution of a function is finished (i.e. when output signals are stable) as a VHDL code is executed in a combinatorial way. When a transition is fired, the execution of its associated function must be finished before the firing of the next transition. Hence it is very intricate to implement ITPN automatically on a FPGA asynchronously. So we choose to implement it synchronously (cf Fig.2) with the constraint that every function takes less than one clock period to execute. The marking of each place is updated on the raising edge (①) and the decision to fire or not is taken on the falling edge for each transition (③). If a transition is fired, associated functions are launched on the following raising edge (①). If a place is marked, associated actions are executed on the following falling edge (③).

To ensure a deterministic behavior, if a transition is fireable, it is fired immediately even if its time slot allows it to be fired later. Also, every concurrent transitions would be fired together, which is not in keeping with the asynchronism of the formalism of PN. The designer must ensure thanks to the conditions, that transitions in a structural conflict can never be in an effective conflict.

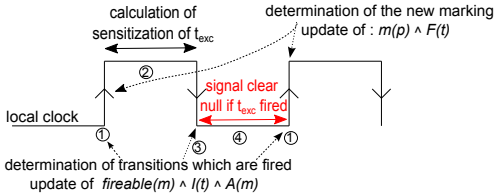

Fig. 2. Synchronous execution of an ITPN

3.2 Implementation of the MP

An efficient way to implement a MP must be defined to guarantee the reactivity and the reliability of this mechanism, but also to keep the benefit of the compactness of the expression of the MP. To implement the MP, entry and exit arcs are easily transformed in classical arcs as they have the same behavior. Therefore there are two issues to consider : how to translate the fact that an exception transition is sensitized by a macroplace and how to handle the firing of an exception transition.

Sensitization of an exception transition An exception arc sensitizes its output transition if the MP is active, i.e. if and only if at least one of the places of the refinement of the MP is marked. Hence for each macroplace, a process is defined to determine whether the markings of the refinement places are nul or not. This process is running asynchronously as soon as the marking of places is modified (cf Fig. 2).

Firing of an exception transition Firing an exception transition means clearing the marking of all internal places of the given MP but also the order to fire and the time counter of internal and exit transitions. It also resets the actions linked to the internal places. Functions do not have to be taken care of as the command to trigger functions is given at the following raising edge of the clock (①) and depends on the decision to fire a transition which have already been cleared if necessary (during ④). Even if the implementation of the PN is synchronous, it has been chosen to asynchronously deal with the effect of the firing of an exception transition. For example, the marking is immediately updated (④) without waiting for the next rising edge of the clock (①). It allows to naturally prioritize the effect of the exception over the ones of the normal transitions.

To deal with an exception, a signal *clear* is associated to each MP. The firing of an exception transition immediately puts this signal down to 0 (④). The orders to clear the marking, time counters and actions signals are handled combinatorially. As the implementation of the MP is synchronous, it is possible that an entry transition and an exception transition are simultaneously fired. In this case, the MP is deactivated but the firing order of the entry transition is not cancelled.

4. GENERATION OF THE ANALYSIS MODEL

To be able to guarantee the behavior of a model in every case, it is necessary to use formal analysis and not solely

simulations. In our case, model-checking is used therefore we need to be able to obtain the reachability graph (RG) of our model written as an ITPN with MP. The interpretation cannot be analyzed so it must be removed to obtain the analysis model (AM). Doing so leads to authorize some behaviors in the AM that were not in the implementation. Moreover, to rely on analysis results, it must be guaranteed that all possible behaviors in the implementation are also considered on the AM. Yet, it is better to reduce the unrealistic behaviors to prevent false analysis results. Additionally the AM must be optimized for getting the RG, hence it is written to reduce as much as possible the risk of combinatorial explosion (notably caused by interleaving transitions).

The initial model is transformed in a Prioritized T-Time Petri Nets (PrTPN) as this formalism is supported by existing analysis tools. The definition of PrTPN is given in Berthomieu et al. (2007). Two main issues must be considered to ensure that all behaviors of the implementation will be in the AM: the withdraw of the interpretation and the asynchronism of the exception. The main point of the model transformation of an ITPN into a PrTPN will be presented and then the principle to flatten a model with macroplaces will be given.

4.1 Transforming an ITPN into a PrTPN

The PrTPN obtained after the transformation must consider not only the interpretation but also the synchronous implementation of the ITPN. One major point in the model transformation is to determine the time slot associated to each transition of the PrTPN. Let I_s be the static interval function of the ITPN and I'_s the one of the PrTPN. To take into account the synchronism of the implementation, for every non-time transition t of the ITPN we have $I'_s(t) = [1, 1]$ in the PrTPN. Moreover, for timed transition, we have : $\downarrow I_S(t) \geq 1 \Rightarrow \downarrow I'_S(t) = \downarrow I_S(t)$ and $\downarrow I_S(t) < 1 \Rightarrow \downarrow I'_S(t) = 1$. For the upper limit of time transition, two points must be considered: a fireable transition is fired as soon as possible and a transition which has a condition can be blocked (cf §2.2). Hence if $C(t) = \epsilon, \uparrow I'_S(t) = \downarrow I_S(t)$ else $\uparrow I'_S(t) = \uparrow I_S(t)$.

The possible blocking of a transition is handled if needed through automatically added places and transitions. The principle (cf Fig. 3) is to add for each transition t_i two types of transitions: one to block t_i, the other to unblock it. The blocking transition is sensibilized if and only if t_i is but classical arcs are replaced by test ones. The time slot of this transition is defined such as $I'_S(blocking\_ti) = [\uparrow I_S(ti), \uparrow I_S(ti)]$. To unblock t_i, a transition is added for every input arcs of t_i to verify if t_i is still sensibilized. In the particular case when $Pre(t_i) = 0$, another place must be added to represent the firing of t_i. This place is immediatly cleared thanks to a sink transition.

Another issue is that the ITPN is implemented synchronously (except for the exception) whereas the AM is asynchronous. Hence two transitions that will be fired simultaneously in the implementation will be fired sequentially in the AM. The behavior of an autonomous synchronized PN is included in the one of the corresponding unsynchronized PN[David and Alla (2008)].

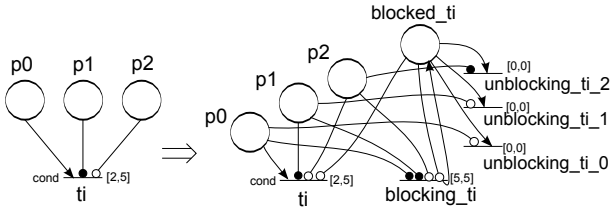

Fig. 3. Transformation for potentially blocked transition

4.2 Flattening a model with a macroplace for analysis

The MP is not supported by existing analysis tools. Therefore it is necessary to flatten the model in a behaviorally equivalent PrTPN. This transformation should be automated in order to prevent any human error in the transformation process. To flatten a model containing macroplaces, three main issues must be solved: how to translate entry and classical exit arcs, how to translate exception arcs (i.e. how to modelize the activity of a macroplace and the purge of the places) and how to handle the simultaneous firing of entry/exit transitions.

4.3 Translation of entry arcs and classical exit arcs

Let us consider an entry arc associated to a transition t. For each place P in the set of places in its associated entry situation, we have: $Pre(t)(P) = n_i$. Now if a classical exit arc is associated to t, for each place P in the set of places in its associated exit situation:

- if $X_i = \epsilon$, $Pre(t)(P) = n_i$.
- if $X_i =?$ and $n_i > 0$, $Pre_t(t)(P) = n_i$.
- if $X_i =?$ and $n_i < 0$, $Pre_i(t)(P) = n_i$.

4.4 Translation of exception arcs

Translation of the MP activity The principle consists of using a place called *active MP* to explicitly represent the MP activity. The solution of handling the marking of this place by checking if all the refinement places are marked or not at each time step, is not considered since it increases the analysis complexity. Indeed, it leads to use concurrent transitions. The idea is then to handle the MP activity by directly using the transitions producing activity (cf Fig. 4). An intermediate place *activation asked* is added to ensure the boundedness of *active MP*. As we define the time slot of intermediate transitions leading to the marking of the *active MP* place equal to [0,0], it is marked at the same instant when the MP becomes active. So the AM behavior is equivalent to the implementation.

To know when the *active MP* must be unmarked, the simpliest solution is to add an inhibitor arc between every place of the macroplace refinement and a specific transition *deactivation*. The time slot of this transition is defined equal to [0,0] in order to unmark the place *active MP* as soon as the macroplace is effectively deactivated.

Clearing the macroplace refinement The firing of an exception transition must lead to the immediate clearing of the macroplace refinement. As far as implementation is concerned, the clear is done on every place and transition simultaneously and asynchronously in less than 1 time unit (§3.2). Hence the idea, for the AM, is to empty the places

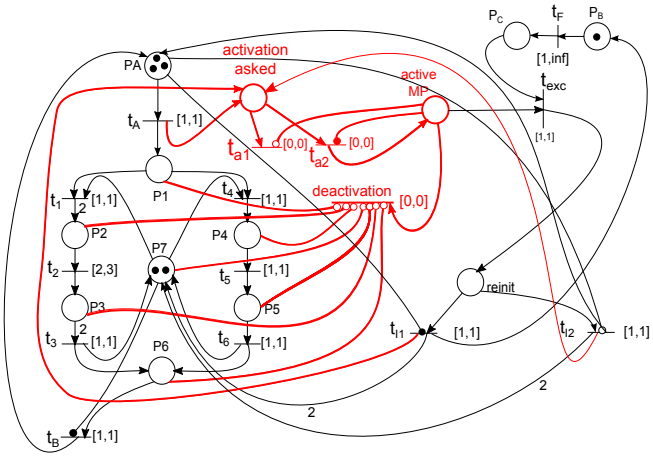

Fig. 4. Modeling the activity of the MP

through added transitions only used for analysis with a time slot equal to [0,0]. This guarantees that all the tokens will be withdrawn before the next time step, and hence the equivalence with the implementation.

Two methods can be used to empty the refinement places: in parallel or sequentially. Although the parallel method looks closer to the implementation and requires less added places and transitions, the two methods are equivalent in the temporal behavior point of view, as every firing of added transitions is made in null time. But in the analysis point of view the sequential method is better. Indeed the parallel method will create interleavings between all the added transitions leading to a more complex analysis. An example of the sequential method is given in figure 5.

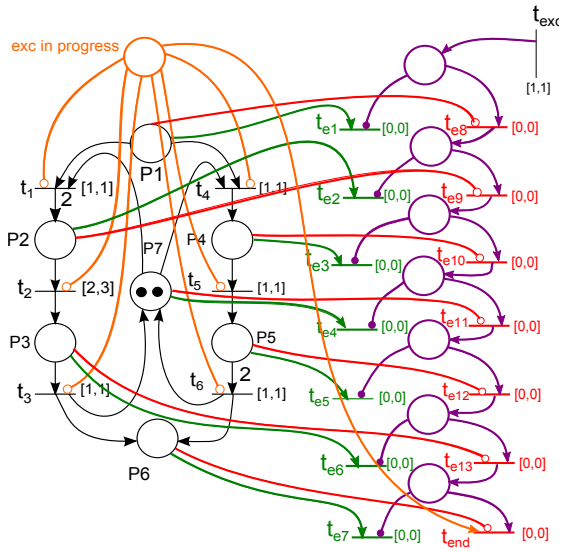

Fig. 5. Clearing the macroplace refinement

Yet the internal transitions t_1 to t_6 of the MP could also be interlaced with the added transitions t_{e1} to t_{e13} and t_{end}. To prevent this, and to make sure that no evolution occurs in the MP during the clearing, a place *exc in progress* is introduced (cf Fig. 5). An inhibitor arc is added between this place and every internal transitions. This solution limits the behavior of the analysis model to the one of the implemented model and prevents useless combinatorial explosion in the building of the reachability graph. The second goal of the place *exc in progress* is to reinitialize the counter of internal transitions even when the transitions have only inhibitor arcs in input.

4.5 Taking care of simultaneous firing for the MP

As the implementation of the PN is synchronous, more than one entry/exit transition can be fired simultaneously. The rules for the activity of the MP in the implementation has been defined (cf §3.2), hence this must be taken into account in the AM. If an entry transition and a classical exit one are simultaneously fired, the MP should stay active. Hence the entry transitions have an higher priority than the *deactivation* transition. If an entry transition and an exception transition are simultaneously firable, the MP must be cleared before and then the tokens given by the entry transition must be added. To guarantee this behavior, the transitions t_{exc}, t_{ei} and t_{end} have a higher priority than every entry transition. Moreover, the *active MP* place is emptied when the transition t_{end} is fired. Hence, in case of simultaneous firing of an entry and an exception transition, there is a deactivation then a reactivation like in the implementation. To be conform with the asynchronous implementation of the effect of an exception, t_{exc} has also a higher priority than every transition of the refinement and all transitions (entry and exit transitions) linked to the MP.

Figure 6 gives the whole analysis model for the example given in figure 1. This figure is given only to show that the AM is less compact than the initial one and quite complex. Yet it is obtained automatically so there is no risk of human errors. Moreover all have been made to reduce the complexity of the analysis, which is the main criterion to consider from an analysis point of view.

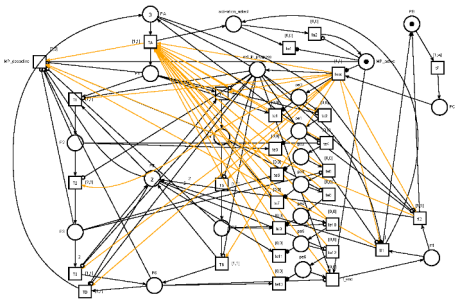

Fig. 6. Analysis model of the designed model given Fig 1

5. RESULTS

To ensure that our solution is adequate, two criteria must be checked: the behavior of the IM must be equivalent or at least included in the one of the AM, and the analysis complexity of the AM must remain acceptable.

5.1 Validating the behavior

The behavior of the implementation of the case study used in this article (cf Fig. 1) has been simulated using the software Libero since we implement it on a Microsemi FPGA. The analysis of the AM behavior has been done

using TINA [Berthomieu et al. (2004)]. In this article, we illustrate the validation principle with only one property: when t_{exc} is fired, the marking of the MP must be cleared in less than one clock period. The result of the VHDL simulation is given in figure 7. It shows the evolution of the clock, the signal *texc_fired* which shows the exception transition firing and the signal *marking_mp* giving the marking of the MP. This last one is created for observation and is equal to one if the MP is marked and zero if not.

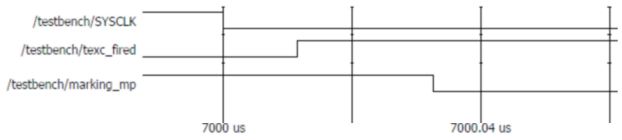

Fig. 7. VHDL simulation of an exception

To validate the clearing of the marking, we used this LTL formula : $[](t_{exc} \Rightarrow <> (P1 = 0 \wedge P2 = 0 \wedge P3 = 0 \wedge P4 = 0 \wedge P5 = 0 \wedge P6 = 0 \wedge P7 = 0))$. It guarantees that if t_{exc} is fired, the marking of the MP will be eventually cleared but not that it is cleared in less than one time unit. To do so, one solution is to use TLTL formulae instead of LTL one, but TINA does not support TLTL. Therefore an observer has been added to the model to be able to verify this quantitative property. Hence we are able to guarantee that, for this property, the implementation and the analysis model have the same behavior. The same has been done for the other properties.

5.2 Size of the resulting reachability graph

To prove the analyzability of models using MP, the size of the RG of models designed with and without MP has been compared. The same 'normal behavior' is considered and cleared in case of an exception either with a macroplace or thanks to an ITPN. There are multiple ways to design this clearing with an ITPN. Here 2 examples will be considered: to empty all places in the same time (parallel) or one after each other (sequential). In both cases, tokens are withdrawn one by one. According to the results given in the table 1, the size of the RG is in the same range for the sequential method and the MP one but the model with MP is far more reactive. The parallel method is more reactive than the sequential one but less than with a MP. Yet the size of the RG in the parallel case is significantly increased because of interleaving transitions. Hence our model transformation benefits from size and reactivity advantages of macroplaces without loosing the possibility of using formal analysis results.

Table 1. Complexity results

model	places	trans	states	trans(RG)	reactivity
parallel	18	26	36 429	120 658	4 c.p.
sequential	19	26	3 744	8 859	15 c.p.
MP	21	19	4 275	8 078	1 c.p.

6. CONCLUSION

In the context of complex digital systems design, the methodology HILECOP uses ITPN with macroplaces. A model designed thanks to ITPN with MP cannot be analyzed because of interpretation, and macroplaces are not handled by existing analysis tools. Hence a model transformation was presented in this article allowing to use existing analysis tools. This transformation guarantees that every possible behavior of the implementation is described in the analysis, and leads to reliable analysis results. Moreover the analysis complexity of a model with MP is in the same size range as for a model without MP. Hence designers benefit from a mechanism allowing to ease design, have more reactivity to exceptions and reduce the FPGA implementation size without losing the benefits of formal analysis. Moreover, this method can be used in industrial size cases. As a perspective, the model transformation between ITPN and PrTPN should be formally proven. Also, better results on analysis complexity could be obtained if the macroplace was directly integrated within an analysis tools instead of flattening it.

REFERENCES

Andreu, D., Guiraud, D., and Souquet, G. (2009). A distributed architecture for activating the peripheral nervous system. *Journal of Neural Engineering*, 6(2), 026001.

Berthomieu, B., Ribet, P.O., and Vernadat, F. (2004). The tool tina c construction of abstract state spaces for petri nets and time petri nets. *International Journal of Production Research*, 42(14), 2741–2756.

Berthomieu, B., Peres, F., and Vernadat, F. (2007). Model checking bounded prioritized time petri nets. In *Automated Technology for Verification and Analysis*, 523–532. Springer.

David, R. and Alla, H. (2008). Discrete, continuous and hybrid petri nets. *Control Systems, IEEE*, 28(3), 81 –84.

de Oliveira, W., Marranghello, N., and Damiani, F. (2002). Exception handling with petri net for digital systems. In *Proceedings of the 15th symposium on Integrated circuits and systems design*, 229–235. IEEE Computer Society.

Holvoet, T. and Verbaeten, P. (1995). Petri charts: an alternative technique for hierarchical net construction. In *Proceedings of the 1995 IEEE Conference on Systems, Man and Cybernetics (IEEE-SMC95*, 22–25. IEEE Press.

Leroux, H., Godary-Dejean, K., and Andreu, D. (2013). Complex digital system design: A methodology and its application to medical implants. In *Proc. of the 18th International Workshop on Formal Methods for Industrial Critical Systems*, 94–107. Madrid, Spain.

Silva, C., Quintans, C., Colmenar, A., Castro, M., and Mandado, E. (2010). A method based on Petri nets and a matrix model to implement reconfigurable logic controllers. *IEEE Transactions on Industrial Electronics*, 57(10), 3544–3556.

Souquet, G., Andreu, D., and Guiraud, D. (2008). Petri nets based methodology for communicating neuroprosthesis design and prototyping. In *ISABEL'08*, 5. Aalborg, Danemark.

Tkacz, J. and Adamski, M. (2012). Logic design of structured configurable controllers. In *Proc. of the IEEE 3rd International Conference on Networked Embedded Systems for Every Application*, NESEA, 1–6. Liverpool, United Kingdom.

Traonouez, L.M., Lime, D., Roux, O.H., et al. (2009). Parametric model-checking of stopwatch petri nets. *J. UCS*, 15(17), 3273–3304.