IP Core For Timed Petri Nets Execution

Orlando Micolini*, Julian Nonino†, Carlos Renzo Pisetta‡
Laboratorio de Arquitectura de Computadoras
Facultad de Ciencias Exactas, Fisicas y Naturales
Universidad Nacional de Córdoba
* Email: omicolini@compuar.com
† Email: noninojulian@gmail.com
‡ Email: renzopisetta@gmail.com

***Abstract*—In this article, we present a Timed Petri Nets processor wich can be can be directly programmed using vectors and matrices of Petri Nets formalism. This processor can leverage the power of Petri Nets for modeling real-time systems and formally verify their properties, which prevent programming errors.**

The Petri Nets processor was developed as an IP-core to be inserted in a multi-core system. Therefore, we can model the system requirements with Petri Nets, formally verifying all its properties and using the IP-core to implement the system is possible to ensure that all properties will be met.

***Index Terms*—Multi-Core, Petri Net, Processor.**

I. Introduction

The evolution of the processors is consequence of the greater integration and composition of different types of functionalities integrated into a single processor. Today, the availability of transistors has made possible to integrate several processor cores on a single chip, which has resulted in the development of Multi-Core technology [1].

Diminishing returns of Instruction Level Parallelism (ILP) and the cost of the increase of frequency, mainly due to power limitations (suggests that a 1% increase in clock speed results in a power increase of 3%) [2], leads to the use of multicore processors to improve performance. This increase deficiency results in lower run times, lower consumption, lower energy density, lower latency and higher bandwidth inter-core communications.

Furthermore, the heterogeneous multi-core systems have the advantage of employing specialized cores, each of them designed for specific tasks. That is, optimized for a particular need. These processors have the ability to use the available hardware resources when they are specifically required by the software [3].

In order to increase performance, these systems make use of multi-threading and/or multi-tasking allowing take advantage of the multi-cores. However, it takes more effort to design applications because they must provide solution to the problems of concurrent systems. Thus, the parallel programming is essential to improve the performance of the software in these systems.

At the Computer Architectures Laboratory of the FCEFyN-UNC a Petri processor has been developed to directly execute ordinary Petri Nets. In this article, we present a new Petri Nets Processor capable to execute Timed Petri Nets and to be programmed directly with the vectors and matrixes that define the system and its state.

II. Objectives

A. Main Objective

The main objective of this work is to design and implement a Petri Nets Processor capable to execute the Timed Petri Nets semantics and to be programmed directly from the modelś state equations.

B. Secondary Objectives

The secondary objectives are:

- Briefly describe Timed Petri Nets in order to implement a processor capable of execute them.
- Keep executing ordinary Petri Nets with time parameters on two processor clock cycles.
- Implement the Timed Petri Nets processor as an IP-core.

III. Petri Nets Considering Time

In the Ordinary Petri Nets formalism, the time is not consider and this results in indeterminism regarding the time. It is not specified when a sensibilized transition will be fired or even if it will be fired. Neither can be said which transition from a group of transitions in conflict will be fired [4].

There are three different interpretations about how the time should be consider. All of them have focus on reduce the indeterminism regarding time in Petri Nets.

A) Stochastic Petri Net
Introduces a stochastic estimation on the instant of firing of a transition.

B) Timed Petri Net
Introduces a time condition which specifies the duration of the transition.

C) Time Petri Net
Introduce temporary dimensions between which the transition should be fired.

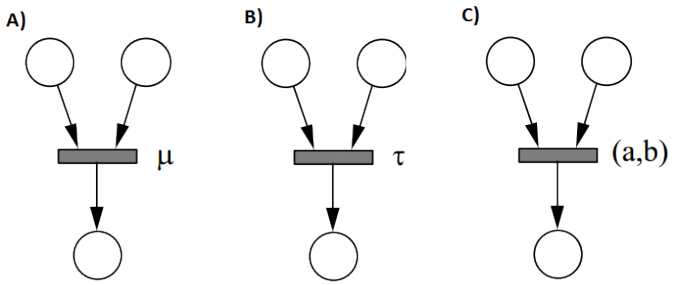

Figure 1. Different ways to intruduce time in Petri Nets

The temporal parameters associated with transitions can be interpreted in these three different ways[1]:

1) Generalised Stochastic Petri Nets (GSPNs) [2] have two different types of transitions: immediate transitions and timed transitions. When a transition t is sensitized, its firing could be:
   a) with a duration equal to zero if the transition t is immediate.
   b) after lapse of a random time. This random time is expressed by an exponential distribution. The A net from the figure 1 graphically represents an stochastic timed transition where its probability to be fired is represented by μ.
2) Timed Petri Nets have two different types of transitions: immediate transitions and timed transitions. When a transition t is sensitized, its firing could be:
   a) with a duration equal to zero if the transition t is immediate.
   b) with immediate removal of tokens from $\bullet t$ set but placing the tokens in the $t\bullet$ that only after time τ has elapsed. Meanwhile, the transition can not be sensitized. The B net from the figure 1 graphically represents a timed transition with a delay equal to τ.
3) Timed Petri Nets have two different types of transitions: immediate transitions and time transitions. When a transition t is sensitized, its firing could be:
   a) with a duration equal to zero if the transition t is immediate.
   b) if it is a time transition, at the time it is sensitized, a timer stars. The transition can only be fired when the timer value is between the limits of the interval [a, b]. Otherwise, the transition can not be fired. Once the firing was performed, the timer is restarted. The C net from the figure 1 graphically represents a time transition with an associated interval equal to [a, b].

[1] $\bullet t$ is the set of places that are inputs to a transition, mathematically defined as: $\bullet t = \{p \in P : (p,t) \in F\}$.
$t\bullet$ is the set of positions that are outputs of a transition, mathematically defined as: $t\bullet = \{p \in P : (t,p) \in F\}$.
F is the set of arcs, input and output to the transitions-

Should be noted that all the firings are performed in two steps.

1) The removal of the tokens from the $\bullet t$ set. This is an atomica action and the amount of tokens removed from each place is equal to the weight of the arcs joining each place $p \in \bullet t$ with the transition t.
2) The atomic action of placing in each place of $t\bullet$ set the amount of tokens indicated by the weight of the arcs joining each place $p \in t\bullet$ with the transition t..

IV. Timed Petri Nets

In this nets, each timed transition has an associated a parameter τ which represents the duration of the transition. In order to standardize the mathematical definition we will call *immediate* to those transitions where τ is zero.

A. Mathematical Definition

A *Marked Timed Petri Net* [4], is mathematically defined as a 8-tuple as follows:

$$PN = \{P, T, I^-, I^+, H, C, m_0, \Gamma\}$$

Where the terms $\{P, T, I^-, I^+, H, C, m_0\}$ represent a marked Petri Net with inhibitors arms and bounded places. Γ is a vector composed of the values of duration τ associated to each transition.

The meaning of each term of the tuple is:

- **P** is a non-empty finite set of places.
- **T** is a non-empty finite set of transitions.
- I^+ and I^- are the possitive and negative incidence matrices.

$$P \times T \to \mathbb{Z}$$

- **H** is the inhibitors arcs matrix. $P \times T \to \{0, 1\}$
- m_0 is the net initial marking. $P \to \mathbb{N}$
- C is a vector containing the values that represent the maximum amount of tokens that each place of the net can hold. $C \to \mathbb{N}$
- Γ is the set of static intervals associated with each transition. $T \to \mathbb{Q}^+ \times (\mathbb{Q}^+ \cup \infty)$ For each transition t the associated value $timer_t$ is:

$$\Gamma(t) = (tau_t \text{ where } t \in T \text{ and } \tau \to \mathbb{Q}^+$$

$timer_t$ represents the time elapsed since the firing of the transition start and its value its value is zero at any other moment. τ_t is the duration of the transition. For that reasons, the following conditions must be met:

$$0 \leq timer_t \leq \tau_t$$
$$0 \leq \tau_t \leq \infty$$

B. States of a Timed Petri Net

In these Petri Nets, the net state is defined by the marking vector ($m = i$) and an *timer* vector that indicates the time stamp of each trasition. Therefore the net state is:

$$S = (m_i, \tau)$$

C. Sensitization of the Transitions and Firing Rules

When we refer to transitions we need to establish the diference between an enabled or sensitized transition, a not enabled or sensitized transition and the firing of a transition.

In a Marked Petri Net whose current mark is m_k we say that a transition t_j is enabled or sensitized if and only if $timer_{tj} = 0$ and the amount of tokens in all places p_i belonging to the set $\bullet t_j$ is at least equal to the weight of the arc that connects them with the transition t_j ($w(p_i, t_j)$). Mathematically:

$$\forall p_i \in \bullet t_j : m(p_i) \geq w(p_i, t_j) \wedge timer_{tj} = 0$$

In summary, every place connected to the transition t_j have at least the number of tokens indicated by the weight of the arc and there is no firing in progress for that transition.

Sensitized transitions can be fired and every time the firing of a transition is completed it generates a new marking for the Petri Net. This means that the net changes its state.

To determine the new state or mark of the net after firing a transition t_j, you should the *state equation* $\delta(m_k, t_j)$.

$$\delta(m_k, t_j) \begin{cases} m_{k+1}(p_i) = m_k(p_i) - W_{ij} & \forall p_i \in \bullet t_j \\ m_{k+1}(p_i) = m_k(p_i) + W_{ij} & \forall p_i \in t_j \bullet \wedge timer_{tj} = \tau_t j \\ m_{k+1}(p_i) = m_k(p_i) & \text{in the rest of the cases} \end{cases}$$

$timer_{tj}$ is incremented in every clock cycle after the firing of the transition started.

D. Interpretation of the firing of transitions in the system

The figure 2 represents a reactive systems that responds to events which come from the environment, it interacs with the environment. Those events are directed to the Time Petri Nets processor.

The responsibility of the processor is arrange events according to system constraints. These constraints are modeled by the TImed Petri Net which is used to program the processor. On the other hand, multicore system threads also generate events (to request resources, to synchronize) that are directed to the processor.

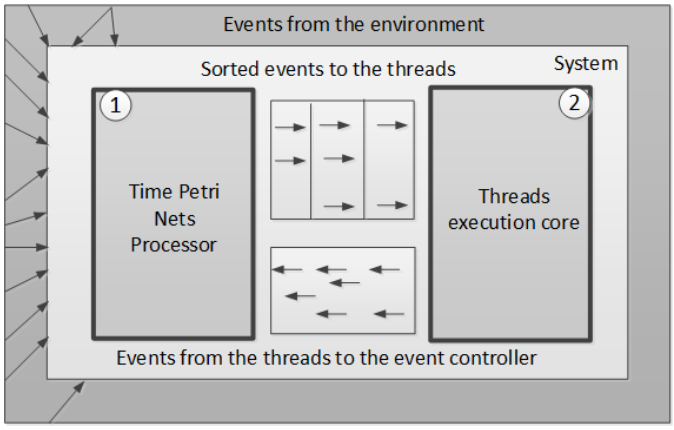

Figure 2. Reactive systems

Module 1 from the Figure 2 receive unsorted events from the environment and from the system itself. After the sorting of the events the Timed Petri Nets processor transmits the result to the threads execution cores (module 2) of the system and the propers actions are taken.

No se entiende el significado

Si en nuestro sistema se asocian el cumplimiento de las condiciones del programa a las que las transiciones estén sensibilizadas, la resolución de un disparo representaran el cumplimiento de dichas restricciones y si asociamos la solicitud de verificación de las condiciones a la solicitud de un disparo, la resolución de un disparo comunica que las condiciones se han cumplido.

The conditions to be met for the firing of a transition from Timed Petri Processor are:

1) The transition must comply with sensitization conditions named in Section IV-C.
2) The shot must be explicitly communicated to the processes or implicitly recorded in the module of systolic firings.
3) Since it is possible that multiple transitions simultaneously satisfy the conditions described in paragraphs 1 and 2, the Timed Petri Nets processor will execute first the firing with higher priority.

Figure 3 show us how the Timed Petri Nets processor is conected in a multicore system.

In case that the firing of the transition can not be resolve, it is queued in the input queue, as shown in Figure 3, until the conditions of the system allow its resolution. The solution of the firing is notified to threads through the system bus, using the output queue. The threads of the system will execute the proper actions as indicated by the firings that have been resolved, since the resolution of the firings depends on the Time Petri Nets processor state, which itself represents the state of the system.

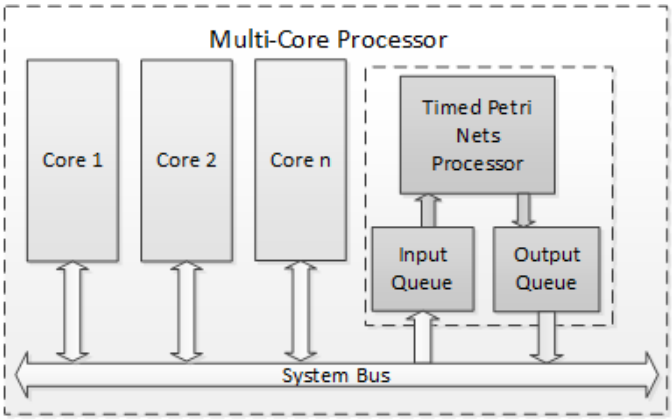

Figure 3. Multicore system with Timed Petri Nets processor

V. Timed Petri Nets Processor Architecture and Operation

The processor executes the state equation solving only a firing of a transition at a time, this way can solve all cases

of firings, the simple ones (single firings) and the multiple firings, performing as a single-firing sequence, as a result, the hardware is simpler.

The resolution of firings is requested by the threads running on the cores through the system bus, as emerging requests that system is running. These firing requests are received by the Timed Petri processor and stored in the input queue. This queue is FIFO according to each transition, the output of this queue is a binary word of size equal to the number of transitions. This word has *ones* in the positions corresponding to transitions with firings requested. The order of the bit in the word equals the number of transition over which the firing is requested. The bits that correspond to the transitions which has no firing request are *zero*.

The output queue has a similar structure, but its function is to communicate to the threads those firings that have been resolved.

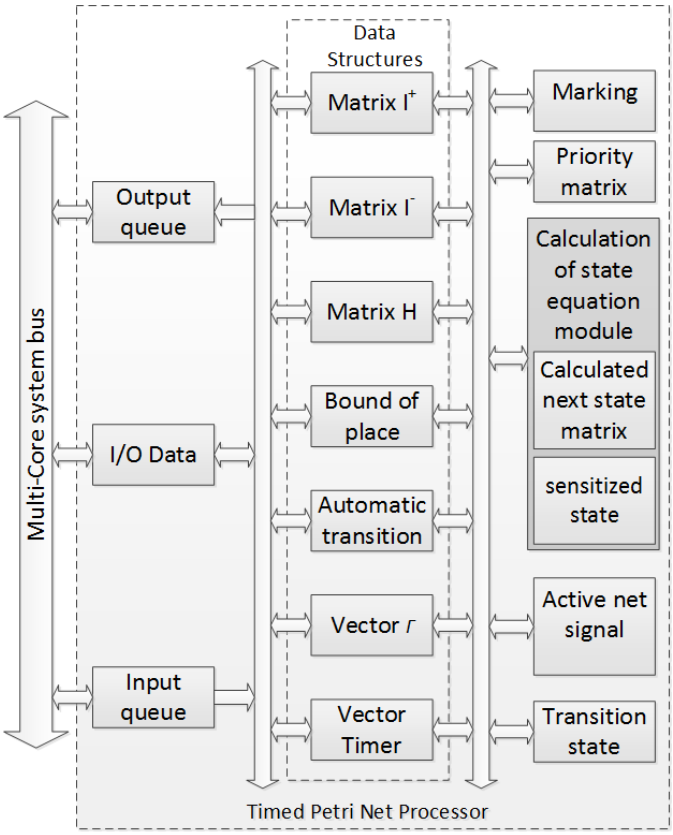

Figure 4. Timed Petri Nets Processor

The data I/O module manages the access of the cores to the matrices and vectors that program the system. The module manages the access of the cores to the matrices and vectors that program the system.

The matrices and vectors described in the equation of state are the system program. This allows us to program the processor directly from the Timed Petri Net.

Here we have added the inhibitor arcs matrix and the vector indicating the maximum number of tokens in the places. This terms are not present in the state equitions shown in this work but you can consult the work [5] to obtain more information about those elements.

The module in charge of solving the state equation of the Petri has the following responsabilities: La responsabilidad del Módulo de Cálculo de la Ecuación de estado es la siguiente:

1) Calculating the new state that would result from each transition firing only once, thereby generating a number of vectors calculated states equal to the number of transitions. Then, these vectors are stored. This is performed by subtracting the current state parallel to each column of I^+ and storing all resulting vectors, which will be evaluated to determine if the new state that each transition would produce is valid. This operation is performed whenever you change the timed Petri Nets Processor status (current marking vector).
2) Determine which transition is sensitized. To do this, take all vectors calculated in step 1 and verify that there is no place to have a negative marking and neither exceeding the limit[2]of tokens it can hold.
3) From the group of sensitized transitions determined at step 2 and its firing has been requested we select the one with higher priority. This transition will be used to determine the new state of the net. This update of the marking vector will be perform by replacing of the current vector with the one calculated in step 1 corresponding to the selected transition. At that moment, in Timer Module, starts the timer corresponding to the transition fired.
4) Compare each component of the Γ vector with the one in $Timer$ vector and verify that it meets the following condition:

$$\Gamma_t \leq Timer_t$$

The fulfillment of this condition means that the transition *t* has reached the delay time required. Then, the transition of higher priority than meets the above condition is chosen to update the marking vector. This means, add to the current marking vector the column of the matrix I^+ corresponding to the transition *t*. At the same time the position *t* of the $Timer$ vector is set to zero ($Timer_t = 0$).
5) Execute the steps 1, 2, 3 and 4 as a continuous cycle.

The system also has a unit that detects when no transition is sensitized and the $Timer$ vector is zero. When this happens, the system generates an interruption notifying that the system has finished its execution or is deadlocked. This feature is very useful to verify the operation of the design and implementation of the system.

VI. Conclusion

The conclusion goes here.

[2]It is noted that this is a weak bound, since the marks in the squares are incremented in step 4 and the limit is checked in step 2. This simplification facilitates hardware implementation.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, 2007.

[2] M. A. Marsan, G. Balbo, and G. Conte, "A class of generalised stochastic petri nets for the analysis of multiprocessor systems," *ACM Trans, On Computer Systems*, 1984/5.

[3] *Embedded Multiprocessors: Scheduling and Synchronization (Signal Processing and Communications)*. CRC Press, 2009.

[4] F. J. García Izquierdo, "Modelado e implementación de sistemas de tiempo real mediante redes de petri con tiempo," Ph.D. dissertation, Universidad de Zaragoza, 1999.

[5] M. M. G. N. A. A. M. A. Micolini, Orlando; Pereyra, "Procesador de petri para la sincronización de sistemas multi-core homogéneos," *Congreso Argentino de Sistemas Embebidos*, 2012.