IP cores para la Ejecución de Redes de Petri Temporales en FPGA

Julián Nonino, *Member, IEEE,* Carlos Renzo Pisetta, *Member, IEEE,* and Orlando Micolini, *Member, IEEE*

***Resumen*—En este trabajo, se presenta un análisis de las Redes de Petri con Tiempo. De esta manera, se puede aprovechar el poder de las Redes de Petri para modelar sistemas de tiempo real, verificando formalmente todas sus propiedades.**

Posteriormente, se presenta el desarrollo de un IP cores, capaz de ejecutar Redes de Petri con Tiempo. De esta manera, es posible realizar la implementación del sistema utilizando este IP core asegurando que todas las propiedades del modelo se verifican en el sistema real.

***Index Terms*—Redes de Petri Temporales, IP core, FPGA.**

I. Introducción

DESDE hace tiempo, los sistemas de computación son multiprogramados, esto significa que en un momento dado existen múltiples procesos cooperando por un fin común y/o compitiendo por recursos limitados. En un sistema monoprocesador, se produce una intercalación de instrucciones de diversos procesos concurrentes aparentando una ejecución paralela. Esto, tiene el problema de la sincronización para el uso de recursos, en la mayoría de los casos un proceso no podrá compartir un recurso mientras lo usa, por lo tanto, los demás deben bloquearse y esperar hasta que el recurso sea liberado. Además, si los procesos comparten datos, se debe asegurar que solo uno de ellos lo modifique en un mismo instante. Por ello, para que el sistema funcione correctamente y se conserve la integridad de los datos se deben utilizar diversos mecanismos de sincronización y de exclusión mutua. Sumado a esto, en la actualidad, la mayoría de los sistemas incluyen múltiples procesadores, por ende, los problemas anteriormente mencionados se multiplican ya que la ejecución paralela es real e intercalada.

Los problemas generados por la ejecución concurrente son: la necesidad de exclusión mutua, condiciones de sincronización, interbloqueos e inanición. Para detectar estos problemas se debe modelar el sistema. Una de las herramientas para modelar procesos concurrentes son las máquinas de estado. El problema de esta herramienta, es la distancia que existe entre el modelo y la implementación, lo que en consecuencia, nos dificulta asegurar que la implementación cumpla con las propiedades validadas y verificadas en el modelo.

En el Laboratorio de Arquitecturas de Computadoras desde hace tiempo se trabaja con otra herramienta para modelar sistemas concurrentes. Ésta, es una herramienta gráfica con una base matemática formal y se conoce como Redes de Petri. Las Redes de Petri, logran modelar los diferentes estados de los sistemas reactivos y las posibles transiciones entre ellos. Este modelo gráfico puede ser traducido a una ecuación de estado que definirá el estado siguiente del sistema según el estado actual y de las transiciones que desean dispararse en forma algebraica. Pero el hecho más importante, es que estas redes no solo sirven para la simulación sino, que además, pueden ser ejecutadas, con lo cual, la distancia entre el modelo y la implementación no existe. Modelando con Redes de Petri, el funcionamiento y las propiedades del sistema pueden ser asegurados aún antes de la implementación. Hay que destacar la importancia del estudio de las Redes de Petri, ya que en los últimos diez años se han realizado 10294 publicaciones con referato.

I-A. Objetivos

I-A1. Objetivo principal: El objetivo principal de este trabajo es diseñar e implementar un procesador de Redes de Petri que sea capaz de procesar Redes de Petri Temporales para la semántica **Redes de Petri con Tiempo**. Manteniendo la condición de programación directa entre el modelo y la implementación.

I-A2. Objetivos secundarios: Los objetivos secundarios de este trabajo son:

- Analizar las Redes de Petri Temporales con el fin de evaluar su implementación por hardware.
- Rediseñar el procesador de Redes de Petri con el objetivo de posibilitar la inserción de parámetros temporales.
- Implementar el procesador de Redes de Petri como un IP core.

II. Redes de Petri Temporales

En los modelos de Redes de Petri descriptos hasta el momento, el tiempo no estaba considerado.

En el formalismo de Redes de Petri básico, o autónomo, la abstracción del entorno en el que la red evoluciona, incluyendo el tiempo como parte de este entorno, es total. Por lo que existe cierto indeterminismo en cuanto al tiempo: no se especifica cuándo se disparará una transición que está sensibilizada (ni si se disparará realmente), tampoco cuál de entre un grupo de transiciones en conflicto será la disparada [1].

Las distintas interpretaciones con tiempo de las Redes de Petri han tratado de reducir el indeterminismo de distintas maneras. Entre estas interpretaciones están:

A. Redes de Petri Estocásticas (Stochastic Petri Net)
Se introduce una estimación estocástica respecto del instante de disparo de una transición.

B. Redes de Petri Temporizadas (Timed Petri Net)
Introduce una condición de tiempo que establece la duración de la transición.

C. Redes de Petri con Tiempo (Time Petri Net)
Introducen cotas temporales entre las cuales la transición puede o debe ser disparada.

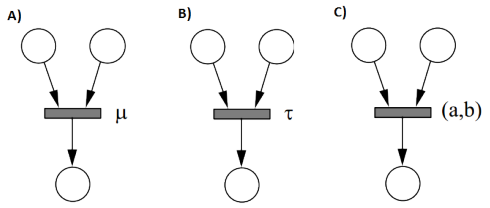


Figura 1. Interpretaciones de Redes de Petri Temporales

Existen dos maneras de interpretar el parámetro temporal asociado a una transición:

- Cuando el parámetro temporal determina el tiempo que ha de transcurrir desde que una transición queda sensibilizada hasta que se dispara, procediéndose entonces a la retirada y colocación de marcas de forma atómica, se habla de ***tiempo de sensibilización (enabling time)***. Está relacionada con las ***Redes de Petri con Tiempo*** (Red C de la figura 1).
- El parámetro temporal puede determinar también el tiempo que debe transcurrir entre la retirada (instantánea) de marcas de los lugares de entrada, y la colocación (instantánea) de marcas en los lugares de salida; en este caso se habla de tiempo de ***disparo (firing time)***. Esto es, el disparo de la transición tiene tres fases (retirada de marcas de entrada, disparo, colocación de marcas de salida) y no es atómico, sino que tienen una duración. Por ello esta interpretación es también conocida como semántica de duración.
  Están asociadas con las ***Redes de Petri Temporizadas*** (Red B de la figura 1).

III. Redes de Petri con Tiempo

En estas redes, cada transición tiene asociado un intervalo $[a, b]$ que abarca todas las posibilidades de duración de la actividad que la transición modela.

III-A. Definición matemática

Una *Red de Petri Marcada con Tiempo*, se define matemáticamente como una 7-tupla de la siguiente manera:

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

Dónde,

- **P** es un conjunto finito y no vacío de plazas.
- **T** es un conjunto finito y no vacío de transiciones.
- I^+ e I^- son las matrices de incidencia positiva y negativa respectivamente.
  $$P \times T \to \mathbb{Z}$$
- H es la matriz de arcos inhibidores. $P \times T \to \{0, 1\}$
- m_0 es el marcado inicial de la red. $P \to \mathbb{N}$
- IS es el conjunto de intervalos estáticos asociados a cada transición. $T \to \mathbb{Q}^+ \times (\mathbb{Q}^+ \cup \infty)$

La función ***IS*** asocia a cada transición un par de valores que representan los límites temporales máximo y mínimo entre los cuales la transición podrá ser disparada. De manera tal que $IS(t) = [min, max]$. Donde t es una transición perteneciente a T. Como la función IS representa un intervalo temporal se deben cumplir las siguientes condiciones:

- $0 \leq min \langle \text{ínf}$
- $0 \leq max \leq \text{ínf}$
- $min \leq max$ si $max \neq \infty$
- $min \langle max$ si $max = \infty$

Al valor min se le suele llamar **Earliest Firing Time EFT (Instante de disparo más temprano)**. Y, al valor max se le llama **Latest Firing Time LFT (Instante de disparo más tardío)**.

Existen dos tipos de intervalos destacables:

- *Intervalo puntual*
  $$[\alpha, \alpha]$$
  En este caso, el tiempo de sensibilización es fijo, luego de que trascurra el tiempo α la transición debe dispararse. Un disparo inmediato es representado por $\alpha = 0$ y se comporta como en las Redes de Petri vistas anteriormente.
- *Intervalo sin restricción temporal*
  $$[0, \infty]$$
  La transición no tiene restricciones temporales para dispararse, se disparará en algún momento después de sensibilizarse.

III-B. Estados en una Red de Petri con Tiempo

En estas Redes de Petri, el estado de la red queda definido por el vector de marcado m de la misma y por un vector $Intervalo$ que lleva la marca de tiempo de cada transición. Por lo tanto el estado de una Red de Petri con Tiempo queda definido por:

$$E = (m_0, Intervalo)$$

Ahora, al disparar una transición t en un instante w_j produce un cambio desde el estado $E = (m, Intervalo)$ a un nuevo estado $E' = (m', Intervalo')$. El nuevo marcado m' se determina con la ecuación de estado vista anteriormente. Por otro lado, la actualización del intervalo para cada transición k sigue las siguientes reglas:

- $Intervalo'(k) = \infty$ si la transición k no está sensibilizada.
- $Intervalo'(k) = Intervalo(k) + 1$ si $k \neq t$, en el marcado m esta sensibilizada y sigue estándolo en el marcado m'.
- $Intervalo'(k) = 0$ si $k = t$ o k comienza a estar sensibilizada en el marcado m'.

III-C. Regla de disparo de transiciones

El intervalo de tiempo definido (min, max) indica el tiempo *mínimo* que debe transcurrir luego del momento en el que se sensibiliza la transición y el tiempo *máximo* para que pueda ser disparada, luego de esto, la transición no podrá ser disparada.

Suponiendo que la transición t_i" comienza a estar sensibilizada en el instante w_i, que continúa sensibilizada y que su intervalo asociado es (min_i, max_i), el disparo de la transición

se producirá no antes del instante $w_i + min_i$, y no mas tarde del instante $w_i + max_i$. El intervalo de tiempos de disparo validos para ti será, por tanto $[w_i + min_i, w_i + max_i]$.

III-D. Determinación de disparos posibles en Redes de Petri con Tiempo

Para que un disparo sea posible en una Red de Petri, debe cumplir las condiciones dadas por que todas las transiciones de las cuales toma tokens tengan la cantidad necesaria de tokens, que las plazas a las cuales esta conectada con arcos inhibidores no tengan tokens y que las plazas en las cuales depositan tokens no superen los limites impuestos por las cotas en las plazas. En una ***Red de Petri con Tiempo***, además de las condiciones anteriores, se debe cumplir que la marca de tiempo asociada a la transición sea mayor o igual al límite de tiempo inferior (*EFT*) y menor o igual al límite de tiempo superior *LFT*. Se debe recordar la semántica temporal utilizada, el vector de tiempo asociado a una transición, lleva cuenta del tiempo desde el instante en el que la transición se sensibilizó.

Durante el proceso de carga y en el instante en el cual este termina, las marcas de tiempo de todas las transiciones valen cero. Luego, a cada ciclo de reloj, si la transición esta sensibilizada la marca de tiempo se incrementa la cantidad de unidades que indica el *vector de incrementos de marcas de tiempo*. La marca de tiempo vuelve a cero en dos situaciones, si la transición es disparada o si deja de estar sensibilizada.

Una ***transición sensibilizada***, si no deja de estarlo, incrementará su *marca de tiempo* hasta que alcance el valor indicado en el *vector EFT*. A partir de dicho instante, la transición se convierte en un ***disparo posible***. El incremento en la marca de tiempo continua hasta que se solicita el disparo de la transición o deja de estar sensibilizada. Si la marca de tiempo supera el valor indicado en el *vector LFT*, y no ha sido disparada, dejará de ser un disparo posible y ya no podrá dispararse.

IV. IP CORES

La arquitectura del procesador de Redes de Petri ya existente se expande para soportar las nuevas sintaxis,se agregan estructuras de datos necesarias.

- Vector ***Earlier Firing Time (EFT)***.
- Vector de ***marcas temporales***.
- Vector ***Latest Firing Time (LFT)***.
- Vector de ***escala de incrementos de tiempo***.

Los cuatro vectores tienen como cantidad de elementos el número de transiciones. Los tres primeros tienen un tamaño de elementos parametrizable pero por defecto tienen 48 bits. El vector de escala de incrementos de tiempo, por defecto toma el un tamaño de elementos de 5 bits.

IV-A. Algoritmo de Ejecución con Redes de Petri con Tiempo

Basados en la teoria descripta se creo un algoritmo de ejecución de disparos en una red de petri que se describe a continuación es sintetizable en hardware y requiere únicamente 2 ciclos de reloj para ejecutar todos los pasos y a la vez permita un diseño parametrizable en cuanto al tamaño y cantidad de elementos que soporte.

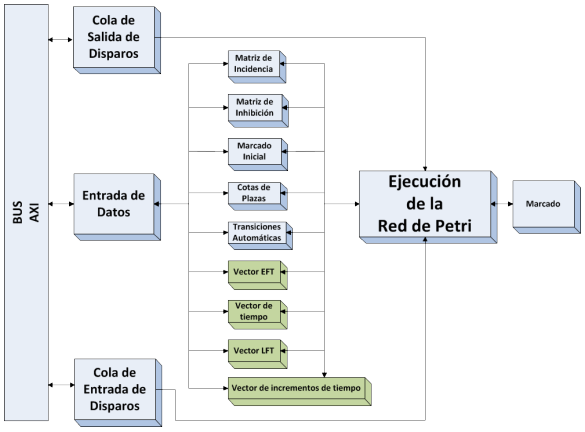


Figura 2. Arquitectura del Procesador de Redes de Petri con Tiempo

1. Espera de disparo en Cola de entrada de disparos.
2. Llegado el disparo se calcula un vector binario de longitud cantidad de transiciones con un único 1 en el lugar correspondiente al número de disparo, en función del número de transición que contenga. Este vector se utiliza para incrementar los contadores de disparos. Son considerados disparos en espera.
3. Se calcula todos los posibles resultados para todos los disparos, hayan sido pedidos o no para confeccionar una matriz resultado donde cada columna Ci representa el nuevo marcado si la transición Ti se disparara.
4. Se crea una matriz de signos auxiliar con los signos correspondientes a cada elemento de la matriz resultado.
5. Se crea una matriz de inhibición auxiliar en función del marcado actual y la Matriz de Inhibición determinando las plazas con arcos inhibidores que tienen tokens.
6. Se crea una matriz de cotas en función de la matriz resultados y las cotas de las plazas determinando cual fue superada para cada plaza por cada posible resultado
7. Se crea un vector en el cual cada elemento corresponde a una columna de la matriz de signos y determina si esa transición es o no posible en función si algún elemento de su resultado es negativo.
8. Se crea un vector en el cual cada elemento corresponde a una columna de la matriz de inhibición y determina las transiciones que no pueden ser disparada debido a arcos inhibidores.
9. Se crea un vector en el cual cada elemento corresponde a una columna de la matriz de cotas el cual determina que transiciones no pueden ser disparadas debido a que provocarían superar las cotas de las plazas.
10. En función de los vectores creados en los puntos 7, 8 y 9 se crea un vector que determina las transiciones sensibilizadas.
11. Se genera un vector en el cual cada elemento corresponde a la transición de igual índice y se verifica que vector de tiempo correspondiente se encuentre entre el EFT y LFT para determinar si la transición se puede ejecutar en función del tiempo

12. Se genera un vector en función de lo obtenido en los puntos 10 y 11 determinando que disparos son posibles.
13. Para determinar las transiciones a disparar se unen los disparos pendientes y las transiciones automáticas. Luego en función de los disparos posibles y que la cola de disparos de salida no esté llena se actualiza el marcado actual a partir del resultado obtenido en el punto 3 considerando la mayor prioridad al índice de mayor valor.
14. Se incrementa contador de cola de salida correspondiente a transición ejecutada.
15. En cada clock de reloj se actualiza el vector de tiempo (contador saturado) de cada transición sensibilizada incrementándolo según el vector de incremento de tiempo correspondiente a partir de que la transición es sensibilizada.

V. Crecimiento del IP Core

Se evaluó el crecimiento del procesador en función de los parámetros que posee. Se generaron procesadores de 8x8, 16x16, 32x32, 48x48 y 64x64 con capacidad de 7 bits por plaza y elementos temporales de 48 bits. Se graficaron los valores que se pueden observar en la Figura 3.

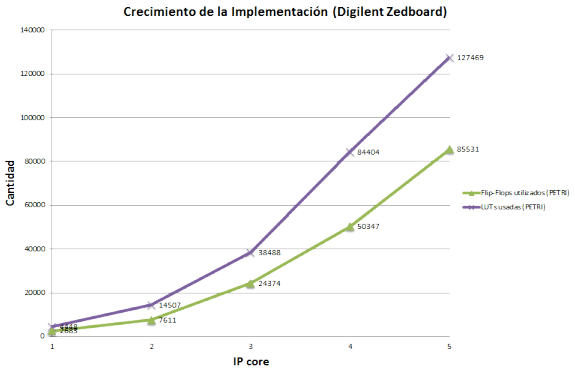


Figura 3. Diferentes implementaciones del procesador de Redes de Petri

Es posible ver que el crecimiento del IP Core no es algo para despreciar, puesto que se incrementa rápidamente a medida que aumentamos el tamaño de elementos soportados. Se observa también que si bien la pendiente aumenta, éstos incrementos son cada vez menores, tendiendo a linealizar para los IP Cores más grandes.

Tambien podemos observar los elementos disponibles que tenemos en una determinada FPGA, para este caso una Spartan6 de Xilinx. Vemos que desarrollar una implementacion de 32x32 es inalcansable, en este caso es necesario pasar a una con mas recursos.

VI. Análisis de Rendimiento

Una vez terminado el IP Core para comprobar su correcto funcionamiento y ver si éste realmente tiene una mejora en los tiempos de sincronización, se realizaron mediciones para distinto número de iteraciones, tanto para el Procesador de Petri como para Semáforos. La elección de este segundo método de sincronización se basa en que son el mecanismo más ligero para realziar éstas tareas. A partir de estas mediciones se realizó la gráfica de la Figura 4

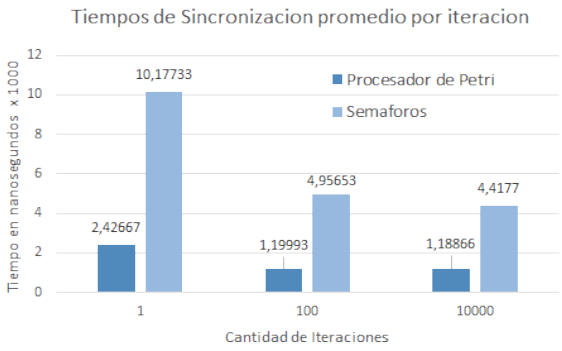


Figura 4. Tiempos de sincronización procesador de Redes de Petri vs. Semáforos

Se puede observar que, para todos los casos, el procesador de Petri es aproximadamente cuatro veces más rapido que los Semáforos. Esta medicion se realizo con tiempos iniciales nulos, de manera que el rendimiento es el mismo obtenido en el procesador de Redes de Petri sin semantica temporal.

En la Figura **??** se puede observar por que se analiza el rendimiento sin tener en cuenta el tiempo. Esto se debe a el tiempo EFT de una transición es parte del modelo, es decir, el mismo que esperara la implementacion con semáforos; a la vez el procesador necesita unicamente un semi-ciclo de reloj desde que el contador alcanza el valor hasta que se ve representada a la salida y esta demora es despreciable en relación con el tiempo que tiene un ΔT de un ciclo de reloj.

VII. Conclusion

The conclusion goes here.

Acknowledgment

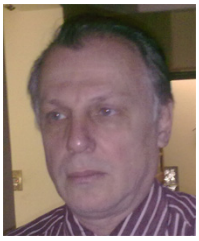
The authors would like to thank...

Referencias

[1] 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.

Julián Nonino (M’2011) nació en Rio Cuarto, Córdoba, Argentina el día 16 de octubre de 1988. En el año 2012 obtuvo el título de grado Ingeniero en Computación de la Facultad de Ciencias Exactas, Físicas y Naturales de la Universidad Nacional de Córdoba. Actualmente se desempeña como Ingeniero de Software en Motorola Mobility of Argentina S.A. En el ambito universitario, precisamente en la Facultad de Ciencias Exactas, Físicas y Naturales de la Universidad Nacional de Córdoba, colabora en las cátedras Ingeniería de Software y Gestión de la Calidad de Software. También, en el Laboratorio de Arquitectura de Computadoras de la misma universidad, realiza actividades relacionadas con el diseño e implementación de software y hardware optimizado para sistemas de computación en paralelo basados en Redes de Petri.

Carlos Renzo Pisetta (M'2011) nació en Córdoba, Argentina el día 07 de junio de 1989. En el año 2012 obtuvo el título de grado Ingeniero en Computación de la Facultad de Ciencias Exactas, Físicas y Naturales de la Universidad Nacional de Córdoba. Actualmente se desempeña como investigador en el Laboratorio de Arquitectura de Computadoras en la UNC(FCEFyN) abocado al diseño e implementación de software y hardware optimizado para sistemas de computación en paralelo basados en Redes de Petri.



Orlando Micolini Grado en Ingeniero Electricista Electrónico, año 1981, de la UNC (FCEFyN) Argentina. Postgrado Especialista en Telecomunicaciones, año 2002, de la UNC (FCEFyN) Argentina. Magíster en Ciencias de la Ingeniería, año 2002, de la UNC (FCEFyN) Argentina. Director de SCS S.R.L, desde 1991 a 2008. Director de la Carrera de Ingeniería en Computación en la UNC (FCEFyN) Argentina (2004-actualidad). Titular de la asignatura Arquitectura de Computadoras en la UNC (FCEFyN) Argentina (1996-actualidad). Actualmente está trabajando, realizando el doctorado de Ciencias Informática en la Universidad Nacional de la Plata, en sistemas multi-core heterogéneos con Redes de Petri.