IP cores para la Ejecución de Redes de Petri Temporizadas

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

***Resumen*—En este trabajo, se presenta un análisis de Redes de Petri Temporizadas. 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 Temporizadas. 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 de tiempo **Redes de Petri Temporizadas**. 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.

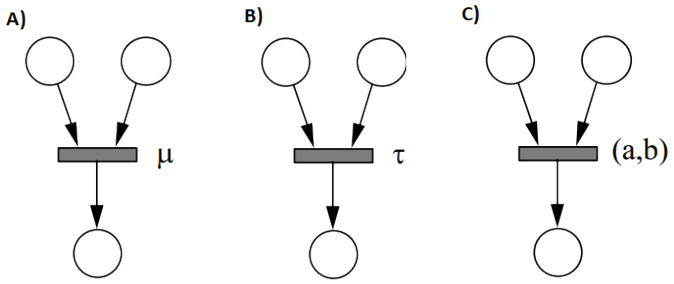


Figura 1. Interpretaciones de Redes de Petri Temporales

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 Temporizadas

En estas redes, cada transición tiene asociado un parámetro $[a]$ que establece la duración de la transición, es decir la duración del intervalo entre que se extraen los tokens de las plazas de entrada y se colocan los tokens en las plazas de salida. En ésta sección se decribirán las ***Redes de Petri Temporizadas*** como la red *B* de la figura 1).

III-A. Definición matemática

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

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

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.
$$P \to \mathbb{N}$$
- es el conjunto de valores de duración asociados a cada transición.
$$T \to \mathbb{Q}^+$$

El conjunto de valores ***Timer*** asocia un valor racional *duración* a cada transición. De manera tal que $Timer(t) = duracion$. Donde t es una transición perteneciente a T.

Dado que duración es una referencia de tiempo, se debe cumplir que $0 \leq duracion \langle \infty$.

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 disparos y estados en una Red de Petri Temporizada

En una *Red de Petri Temporizada*, el disparo de una transición implica tres etapas.

1. Al momento que se sensibiliza la transición, se extrae la cantidad de tokens de las plazas de entrada indicada por el arco que los une a dicha transición. Además, se marca la transición como no sensibilizada.
$$m_{i+1} = m_i + I^- \times d$$
Se genera un nuevo estado provisorio formado a partir la extracción de los tokens que sensibilizaban la transición que se esta disparando. También, en ese instante se comienza la cuenta de tiempo.
2. Se espera el tiempo indicado por el valor de duración de la transición. Con la transición marcada como desensibilizada, se espera hasta que la cuenta de tiempo, que comenzó al extraerse los tokens, llegue al valor indicado por $Timer(d)$.
3. Transcurrido el tiempo indicado por el valor de duración asociado a la transición se colocan los tokens en las plazas de salida según como indican los arcos que unen a las transiciones con dichas plazas.
$$m_{i+2} = m_{i+1} + I^+ \times d$$

III-D. Determinación de disparos posibles

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 Temporizada, al cumplirse las condiciones antes mencionadas, se dice que la transición esta ***sensibilizada***, entonces, se ejecuta la ecuación de estado de la Red de Petri utilizando la matriz I^-. De esta manera, se remueven todos los tokens de las plazas de entrada.

$$m_{i+1} = m_i + I^+ \times \delta \tag{1}$$

Dónde δ es el vector de disparo que se desea ejecutar. A partir de dicho momento, la marca de tiempo asociada a la transición comienza a incrementarse según su vector de incrementos de tiempo. Cuando esta marca de tiempo alcanza el valor indicado por el ***vector duración***. Cuando esto ocurre, el disparo ya esta habilitado para ser ejecutado completamente. Esto se hace ejecutando la ecuación de estado con la matriz de incidencia positiva, esto implica poner los tokens en todas las plazas de salida.

$$m_{i+1} = m_i + I^- \times \delta \tag{2}$$

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 se sensibilizó y logo ejecutar la primera etapa del disparo, 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 solo cuando la transición completa ambas fases de su disparo.

IV. IP CORES

La arquitectura de este IP core se mantiene en su mayor parte igual a la de las versiones anteriores. Con respecto a la versión que procesa *Redes de Petri con Tiempo*, se quitaron los vectores *EFT* y *LFT*, en su lugar se agregó el ***vector duración***. Se mantuvo el ***vector de incrementos de tiempo*** y el ***vector de tiempo de sensibilización***.

IV-A. Estructuras de datos para Redes de Petri Temporizadas

Como se muestra en la figura 2 las estructuras de datos necesarias para la resolución de Redes de Petri Temporizadas son tres:

- Vector ***duración***.
- Vector de ***marcas temporales***.
- Vector de ***escala de incrementos de tiempo***.

Los cuatro vectores tienen como cantidad de elementos el número de transiciones. Los dos primeros son tiene un tamaño de elementos parametrizable pero por defecto tienen 32 bits. El vector de escala de incrementos de tiempo, por defecto toma el un tamaño de elementos de 5 bits.

IV-B. Algoritmo de Ejecución con Redes de Petri Temporizada

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.

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. 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 únicamente retirando tokens de las plazas de entrada en función de la matriz I-.
12. Se habilita contador de transición temporizada.
13. 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 disparada.
14. Cuando el vector de tiempo supera la duración de la transición se completa el disparo actualizando el marcado únicamente agregando los tokens en las plazas

de salida en función de la matriz I+.

15. Se incrementa contador de cola de salida correspondiente a transición ejecutada.

V. Crecimiento del IP Core

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

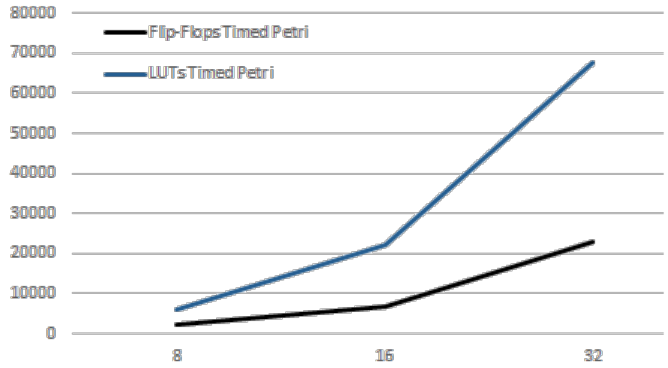


Figura 3. Diferentes implementaciones del procesador de Redes de Petri

Es observa que el crecimiento del IP Core no es algo para despreciar, puesto que se incrementa rápidamente a medida que aumentamos la cantidad 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.

VI. Timed vs Time

Debido a la posibilidad de implementar Timed Petri Nets utilizando plazas y transiciones intermedias utilizando un procesador de Time Petries Nets, se compararon las curvas de crecimiento, las cuales se muestran en la la Figura4

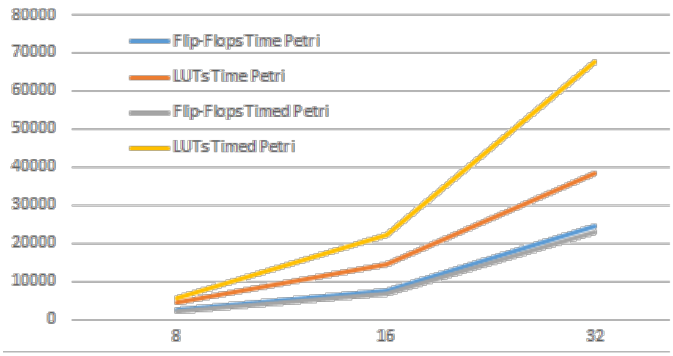


Figura 4. Comparación del crecimiento del IP Core

Se Puede observar que el incremento de Flip-Flops se mantienen parejos, pero en el caso de las LUTs el crecimiento exponencial posee mayor pendiente en el Timed ip core, necesitando un 90 % mas que el Time ip core.

VII. Análisis de Rendimiento

Para comprobar correcto funcionamiento del IP Core y analisar los tiempos de sincronización, se realizaron mediciones para distinto número de iteraciones. Luego se compararo el Procesador de Petri con una implementación utilizando

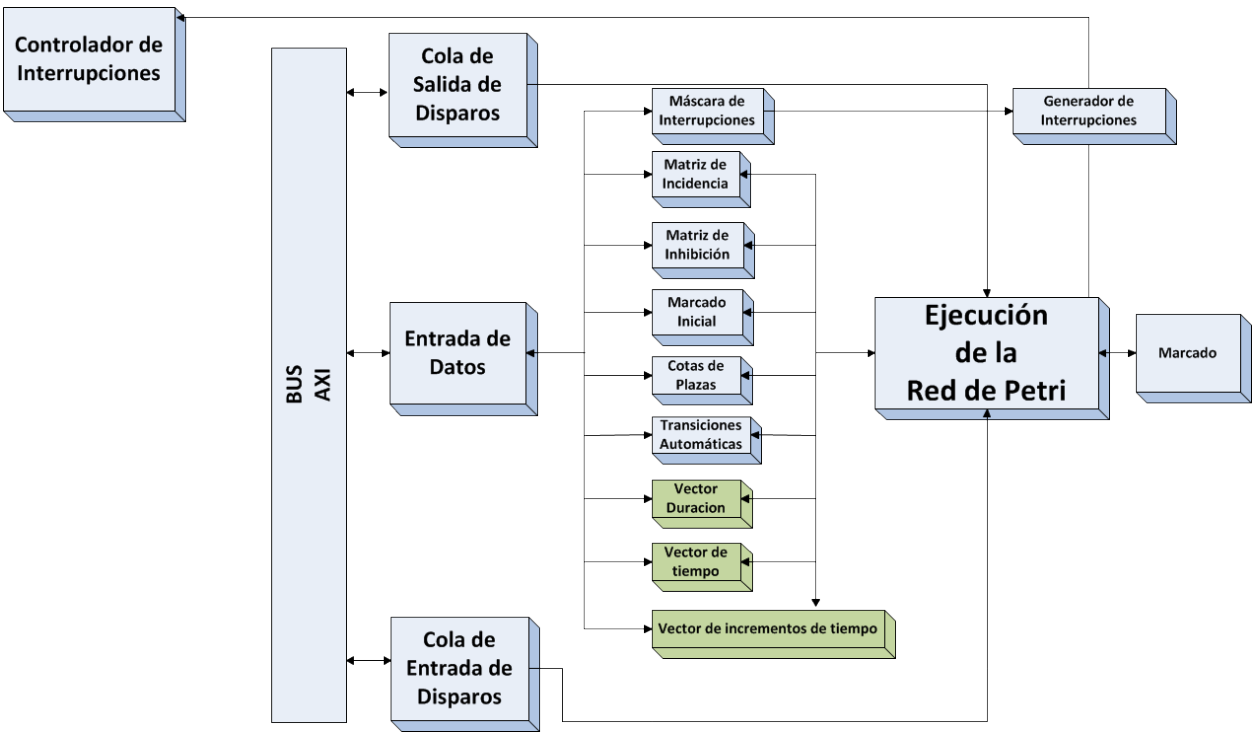


Figura 2. Arquitectura del procesador de Redes de Petri Temporizadas

semáforos, ambos resolviendo un mismo problema. La elección de este segundo método de sincronización se basa en que son el mecanismo más ligero para realizar éstas tareas. A partir de estas mediciones se realizó la gráfica de la Figura 5

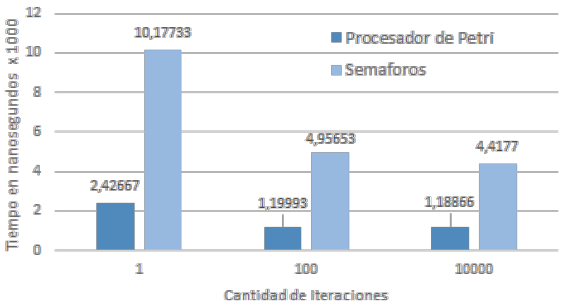


Figura 5. Tiempos de sincronización procesador de Redes de Petri vs. Semáforos

Se puede observar que, para todos los casos, el procesador de Petri es aproximadamente cuatro veces más rapido que los Semáforos.

Esta medicion se realizó con tiempos τ nulos, de manera que el rendimiento es el mismo obtenido en el procesador de Redes de Petri sin semántica temporal. Esto es valido ya que el tiempo de una transición es parte del modelo ,es decir, es el mismo para el procesador de petri como para la implementeción con semáforos.

Ademas como se observa en la Figura 6, el procesador necesita únicamente un semi-ciclo de reloj, desde que el contador alcanza el valor τ hasta que el disparo se coloca en la cola de salida. La demora introducida es despreciable en relación con el tiempo que tiene un ΔT de un ciclo de reloj.

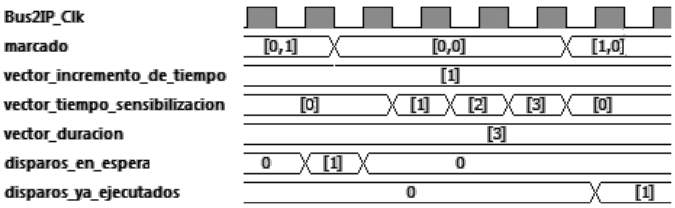


Figura 6. Tiempos de ejecución2

Teniendo en cuenta la latencia despreciable y tomando el tiempo como parte del modelo es posible analizar el rendimiento sin tener en cuenta el tiempo.

VIII. Conclusion

The conclusion goes here.

Acknowledgment

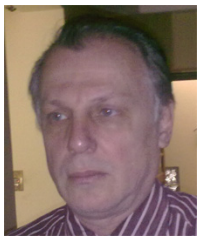
The authors would like to thank...

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.