Localidad estructural, criterio de división para la ejecución de redes de Petri no autónomas en IP-Core

Dr. Ing. Orlando Micolini[1], Geol. Marcelo Cebollada y Verdaguer[1], Ing. Ventre, Luis Orlando[1].

[1]Laboratorio de Arquitectura de Computadoras - FCEFyN - Universidad Nacional de Córdoba
omicolini@compuar.com, {mcebollada, lventre}@gmail.com

Abstract. — Este trabajo propone un nuevo concepto, localidad estructural, para dividir y representar redes de Petri no autónomas con el fin de reducir de forma significativa los recursos de hardware de los IP-Core que las ejecutan. Con la consecuente ventaja de lograr asi abordar problemas de mayor tamaño.

Las redes así expresadas dan origen a un algoritmo de ejecución que preserva el modelo original y facilita el paralelismo.

En este trabajo se expone un caso de aplicación donde se manifiestan las ventajas de la localidad estructural aplicada a una red de Petri con diferentes semánticas temporales y tipos de brazos, obteniendo una disminución de recursos en la FPGA que implementa el IP-Core.

Abstract. — This paper proposes a new concept, structural locality, to divide and represent no autonomous Petri nets, with the aim of significantly reduce the hardware resources of the IP-cores that run them. With the consequent advantage of addressing in this way larger problems.

The Petri nets represented by this way raise an algorithm of execution which preserves the original model and facilitates the parallelism.

Finally in this paper we expose a real case of application where shows the advantages of the structural locality applied to a Petri net with different temporal semantics and types of arms, achieving an important decrease in resources of the FPGA that implements the IP-Core.

Keywords: Procesador de Petri, Redes de Petri jerárquicas, localidad estructural, IP-Core.

1 Introducción

Para mejorar el rendimiento de los sistemas informáticos y la capacidad de cómputo, las aplicaciones actuales implementan hilos [1] que se ejecutan en sistemas multicore. Estos hilos cooperan y se ejecutan concurrentemente. Las aplicaciones diseñadas de esta manera generan una complejidad adicional con respecto a los programas secuenciales. Esta complejidad se manifiesta en el diseño, detección de errores, testing, validación y mantenimiento [2]. Además es necesario introducir mecanismos de control, como los semáforos, que generan penalización en los tiempos de ejecución.

Por todo esto es importante resolver el sistema formalmente, lo que puede realizarse haciendo un modelo que facilite la programación.

Investigaciones recientes mostraron que los modelos obtenidos con redes de Petri (RdP) pueden facilitar la implementación de sistemas de manera directa, utilizando procesadores o IP-Core que ejecutan RdP [3-6].

Los principales problemas de estas soluciones son: el tamaño de las matrices que representan los modelos, las distintas semánticas temporales, tipos de brazos y tipos de eventos en las transiciones. Los que limitan el tamaño de los problemas que se pueden abordar, debido a los recursos de hardware disponibles.

Los procesadores de RdP (PP) han sido implementados como IP-Cores, en una FPGAs Spartan-6, como se muestra en [7, 8]. Debido a las limitaciones de hardware existentes, se pueden sintetizar con facilidad PP capaces de contener hasta 50 plazas y 50 transiciones. Para resolver sistemas de mayor envergadura se requiere modelos de RdP con mayores números de componentes o elementos, como: plazas, transiciones, tipos de brazos y distintas semánticas temporales. Hay que notar que los recursos demandados a la FPGA, para implementar los IP-Core, aumentan en proporción al producto de la cantidad de plazas por la cantidad de transiciones y por la cantidad de distintos tipos de brazos.

Por cada tipo de brazo existe una matriz de dimensiones Plazas por Transición y además por cada transición existe una cola de eventos de entrada y una de salida. También según sea el tipo de semántica temporal pueden existir registros de tiempo y un contador, ambos de 32 bit.

En el presente trabajo se amplía el uso de los mecanismos de división de las RdP, introducido en [8], haciendo uso del concepto de localidad estructural, para obtener RdP jerárquicas (HPN) que aprovechan los recurso de manera más efectiva, y mantienen todas las propiedades de la ecuación de estado de las RdP no autónomas.

2 Redes de Petri jerárquicas

En las HPN [9], cada subred que compone el sistema posee un estado particular en un momento dado, esto da lugar a que haya distintas transiciones sensibilizadas por cada subred del sistema en un instante determinado. Estas transiciones pueden disparse arbitrariamente si no hay conflicto entre ellas. En caso de conflicto se emplea un esquema de prioridades para mantener el determinismo del sistema.

Existen dos tipos de transiciones en las subredes, las transiciones internas y las transiciones de borde [9]. Para que una transición de borde pueda ser disparada es necesario que todas las transiciones de borde que representan a esa única transición en el sistema original estén sensibilizadas, esto conserva la semántica original de las transiciones en las RdP.

Las transiciones de borde de cada subred, son transiciones que han sido divididas y pertenecen a distintas subredes. Como consecuencia de la división una transición queda distribuida y pertenece a subredes distintas.

$$[T_i] \xrightarrow{Ri^{|Ti|x|Tb|}} [Transiciones\ de\ Borde\ de\ la\ subred\ [b]]$$

Donde T_i son las transiciones de la subred i.

Todas las transiciones de borde que no tienen transición interna en la subred i deben estar permanentemente sensibilizada en la transición de borde de la subred i, para esto se realiza una máscara de la subred i. La máscara tiene un uno por cada transición de borde que no está en la subred.

$$sensibilizados\_de\_borde\_de\_la\_subred[i] = Transiciones\ de\ Borde\ de\ la\ subred\ [i]\ \ OR\ mascara\_subred[i]$$

$$sensibilizados\_distribuidos = \bigwedge_{j=0}^{r-1} sensibilizados\_de\_borde\_de\_la\_subred[i]$$

Donde r es la cantidad de subredes en que se ha dividido el sistema.

La Figura 1 muestra las transiciones divididas, distribuidas y de borde de una RdP que modela un productor consumidor.

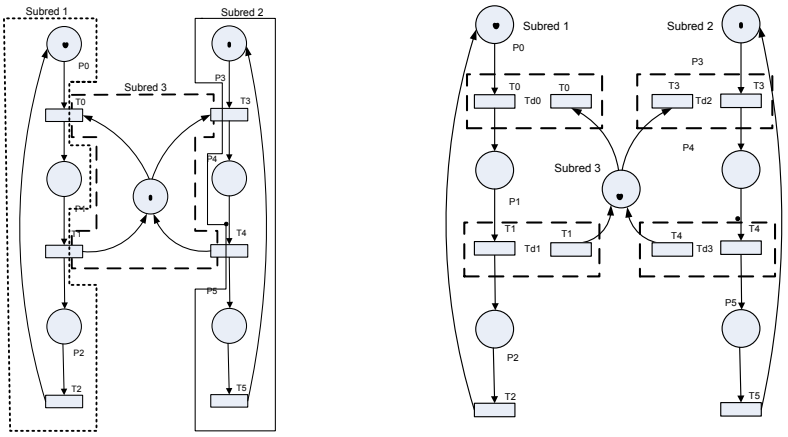

Figura 1. Red de productor consumidor dividida en tres subredes

Las transiciones distribuidas, en la figura son: Td0, Td1, Td2 y Td3. Mientras que las transiciones internas de la subred 1 son T0, T1 y T2, para la subred 2 son T3, T4 y T5 y para la subred 3 son T0, T1, T3 y T4.

El algoritmo de disparo de HPN [9] mantiene la ejecución de la RdP original, y es implementado en hardware mediante un circuito combinacional, que ejecutan la lógica de las ecuaciones del apartado anterior. Esto ha sido realizado con compuertas simples que permiten mantener los beneficios temporales alcanzados por el PP [8].

Interpretación del rango de la matriz de incidencia

De la semántica de disparo de una RdP podemos interpretar a la matriz de incidencia considerando a las columnas (transiciones) como la evaluación conjuntiva de las

restricciones que imponen las filas (plazas). Es decir que en una matriz de incidencia de dimensión mxn se evalúan n combinaciones de m variables lógicas, como se expresa en la siguiente ecuación:

$$if_{h=0}^{n-1}(\bigwedge_{k=0}^{m-1} p_i \geq w_{i,h})$$

Donde $w_{i,h}$ son los elementos de la matriz I [3] que resulta de la resta entre las matrices I_i^+e I_i^-, y representan el peso y la dirección de los arcos de la RdP. Donde p_i es la marca en la plaza i.

3 División de RdP

La investigación bibliográfica del análisis de división de RdP ha sido realizada en trabajo [6].

Si consideramos que las RdP están constituidas por plazas, transiciones y brazos, a los cuales llamamos elementos. Donde, estos elementos participan de la matriz de incidencia, de la siguiente manera: las transiciones como la cantidad de columnas, las plazas como la cantidad de filas y los brazos son los valores que relaciona una plaza con una transición, según un peso y una dirección.

En las RdP vemos que estos elementos se agrupan según subconjuntos, caracterizados por estar fuertemente relacionados entre sí. Es decir, que hay brazos relacionando plazas con transiciones y viceversa. A su vez, estos subconjuntos se relacionan con otros subconjuntos de una manera débil, es decir que hay pocos brazos que relaciona a ambos sub conjuntos. Por esto vemos a la matriz de incidencia, como un caso de "matriz rala" [10] [11] [12].

Como podemos ver en la bibliografía, los algoritmos para resolver estas matrices ralas no son aplicables a implementaciones del PP con FPGA. Los algoritmos evaluados usan técnicas de compactación y/o punteros para direccionar los elementos, no usan operaciones lógicas simples; todo esto demanda recursos y ciclos de máquina, lo que se traduce en excesiva sobrecarga.

Con el fin de obtener una reducción de recursos se propone dividir la red en subredes. Para explicar esto, ahora suponemos que una red con M plazas y N transiciones es posible dividirla en dos subredes, donde cada una tienen la mitad de plazas (M/2) y transiciones (N/2), lo que resulta en:

- Para el sistema sin dividir $\boldsymbol{M} * \boldsymbol{N}$ elementos en la matriz.
- Para el sistema dividido $\boldsymbol{2}((\boldsymbol{M}/\boldsymbol{2}) * (\boldsymbol{N}/\boldsymbol{2})) = (\boldsymbol{1}/\boldsymbol{2})\boldsymbol{M} * \boldsymbol{N}$, más una sobrecarga que especifica las relaciones entre las subredes.

Hacemos notar que la cantidad total de plazas y transiciones del sistema se mantiene, por lo que en principio es posible mantener su representación, siempre y cuando seamos capaces de determinar cómo relacionar las subredes. Además, la sobrecarga que especifica las relaciones entre las subredes tiene que ser menor a $(1/2)M * N$. Esta diferencia se traducirá en la reducción de recursos. Para este caso, en que la red

ha sido dividida en dos subredes, la ganancia máxima es de dos (sin considerar la sobrecarga).

Podemos considerar que si se divide la red en J partes, la ganancia máxima teórica sería función de J, menos el costo de interconexión. La determinación de la ganancia máxima teórica sin tener en cuenta la localidad estructural ha sido tratada en [9].

Teniendo en cuenta lo dicho, proponemos reducir aún más la cantidad de recursos en el HPP, donde se realizara la división logrando que cada PP que conforma al HPP tenga sólo los recursos necesarios para ejecutar la subred.

3.1 Consideraciones para División RdP

Para realizar la disminución de recursos que nos hemos propuesto, dividiendo una RdP de la forma y el modo anteriormente propuesto, es necesario contestar los siguientes interrogantes:

- ¿Cuánto y cuál es la ganancia de dividir las RdP? Esto ha sido contestado, sin considerar la localidad estructural, en el trabajo [9].
- ¿Las RdP resultante son más simples? Las subredes son más simples, dado que es posible realizar la división de las transiciones en cualquier número de partes, lo que nos permite elegir subredes simples.
- ¿Existe un criterio de división de la RdP original con el que se obtenga una mejora respecto a una división en subredes equilibradas en cuanto a la cantidad de elementos? Esta es la respuesta que buscamos en este trabajo.

La RdP dividida tiene que expresar y preservar el comportamiento de la RdPnA. Esto tiene dos aspectos: la regla de disparo de las transiciones y la relación de la red con los eventos.

El primer aspecto ha sido mostrado en[9] . En cuanto al segundo, es menester que se mantengan las relaciones entre las transiciones y los eventos que disparan las transiciones. Además las comunicaciones que generan los disparos de las transiciones deben poder ser informadas. Es decir que la división no introduzca ambigüedad en los eventos generados por el disparo de una transición, esto ha sido considerado y resuelto en [6].

Finalmente el esquema de prioridades tiene que ser posible de implementar y facilitar la ejecución en paralelo. Para esto, las transiciones en conflicto en cada subred, son resueltas por un esquema de prioridad local. Mientras, las transiciones de borde tienen más prioridad que las locales de cada subred y además un esquema de prioridades propio.

La Figura 1, muestra una RdP dividida por las transiciones. Las transiciones divididas se encontrarán presentes en cada una de las subredes resultantes, la comunicación se realiza por las transiciones de borde, que son los límites de las subredes. Podemos entonces decir que existe una relación entre todas las partes de cada transición cortada, que originalmente constituía una única transición. Estas partes están comunicadas y se evalúan y disparan de forma simultánea.

Hay que notar que el hecho de dividir las redes de esta forma, genera una jerarquía explícita de redes, con un concepto más amplio, que llamamos ¨Redes de Petri Jerár-

quicas Divididas por Transiciones¨. De esta división resulta una reducción de recursos con respecto a la red original.

Por esto consideramos al sistema resultante como una HPN, lo que facilita la interpretación y la implementación en distintos dispositivos intercomunicados como se realiza en [13].

3.2 Consideraciones para obtener Redes Jerárquicas Divididas por Transiciones

Lo primero que observamos, es que sólo existe una red. No hay una red global y subredes, sino que todas las partes de la red o subredes tienen la misma jerarquía y se relacionan entre ellas mediante transiciones de borde. No obstante a cada parte de la red obtenida por la división la llamaremos subred.

1. Las transiciones distribuidas están constituidas por el conjunto de todas las transiciones de borde, que son el resultado de la división de una misma transición de la red total.
2. Para obtener el estado global del sistema consideramos todas las plazas en conjunto, es decir que hay que tener el marcado de cada subred.
3. Cuando se ejecuta una transición distribuida, se deben ejecutar todas las transiciones de borde de todas las subredes a las que pertenece, con el fin de garantizar la semántica del disparo. Esto es, cuando todas las transiciones distribuidas que pertenecen a una misma transición están sensibilizadas, esta transición está en condiciones de dispararse.
4. Con el fin de que un disparo de una transición distribuida no genere conflicto en una subred (con otras transiciones en conflicto), solo se permite la ejecución de una sola transición distribuida por ciclo de disparo.
5. La red se divide cortando las transiciones, de modo que las transiciones elegidas como límite entre dos o más subredes quedan partidas, constituyendo una transición de borde para cada una de las subredes. Estas transiciones de borde, en cada subred estarán relacionadas, constituyendo una transición distribuida.
6. Como cada subred mantiene su funcionamiento por separado, se aumenta el nivel de paralelismo en la ejecución del sistema, pudiendo darse como máximo que todas las subredes ejecuten un disparo (en transiciones que no son de borde) en el mismo ciclo.
7. La matriz binaria de relación de transiciones distribuidas, indica como las transiciones distribuidas se reagrupan para constituir la transición original en cada subred.
8. Para evitar excesiva comunicación entre las subredes, ante un conflicto, las transiciones distribuidas tienen mayor prioridad de disparo que cualquier transición interna de la subred.
9. Si es necesario que una transición interna sea de mayor prioridad que una transición de borde en conflicto, se la puede definir como una transición distribuida que solo está relacionada con la subred a la que pertenece.

Ahora introducimos el concepto de localidad estructural para guiar la división de la red, esto es: "*dividir las redes por las transiciones agrupando elementos en los subconjuntos con características comunes en cada subred*". Estos elementos son: arcos con peso uno, arcos con pesos mayor a uno, transiciones con tiempo, transiciones temporizadas y otros tipos de arcos.

4 Un ejemplo: División de una red con N lectores y un escritor en tres subredes

En la Figura 1. Red de productor consumidor dividida en tres subredes, se muestra la RdP que modela a N lectores y un escritor, que ha sido dividida en tres subredes.

Originalmente se parte de una RdP de 7 plazas (m) y 6 transiciones (n) más la matriz de prioridades (nxn), para esta RdP el PP requiere 372 bits.

$$PN = (mxn)8 + (nxn) = 372$$

Dividiendo ésta RdP, como lo muestra la Figura 1. Red de productor consumidor dividida en tres subredes , en tres sub redes los recursos necesarios son 290 bits. Lo que muestra en la Tabla 1.

Tabla 1: Recursos RdP con N lectores y un escritor, dividida en tres subredes

	Plazas	Transiciones	Matriz de P.	Recursos
Red0 y Red1	3 c/u	3 c/u	3x3	81 c/u = 162
Red2	1	4	4x4	48
Matriz de relación	Son 8 las transiciones de borde, con cuatro sub redes 4x4x3			48
Matriz de P. de T. de borde.	4*4			16
Total de recursos de la red divida- Reducción de recursos 79%				290/79%

Por lo que la división tiene una ganancia de recursos es del 21%.

Si además observamos que las subredes uno y dos solo necesitan brazos con peso uno y menos uno en la matriz, por lo que se requiere dos bit, mientras que en la subred tres se requiere de brazos con peso N, para lo cual se usan 8 bit. Los recursos necesarios son 55,9%, esta ganancia adicional es resultado de aplicar la localidad estructural

5 Caso de Aplicación

A continuación aplicamos el concepto de localidad estructural a un caso de modelado con RdP para el análisis de desempeño de aplicaciones de búsqueda WEB.

El caso, se ha tomado de [14] (A Modeling Technique for the Performance Analysis of Web Searching Applications). En este trabajo se investiga el comportamiento de un sistema cliente /servidor, haciendo uso de un modelo no-Markoviano realizado

con RdPnA para obtener los índices de rendimiento. También se hizo uso de un entorno experimental, con el fin de obtener mediciones reales que son utilizadas para validar el modelo analítico.

5.1 Modelo de RdP para el cálculo de tiempo de respuesta del sistema Cliente Servidor

La Figura 2 muestra la RdP, presentada por el autor, que ha sido dividida en 5 subredes. La división se ha realizado con el criterio de localidad estructura, agrupando las transiciones del mismo tipo y además respetando los bloques de procesos de la red original.

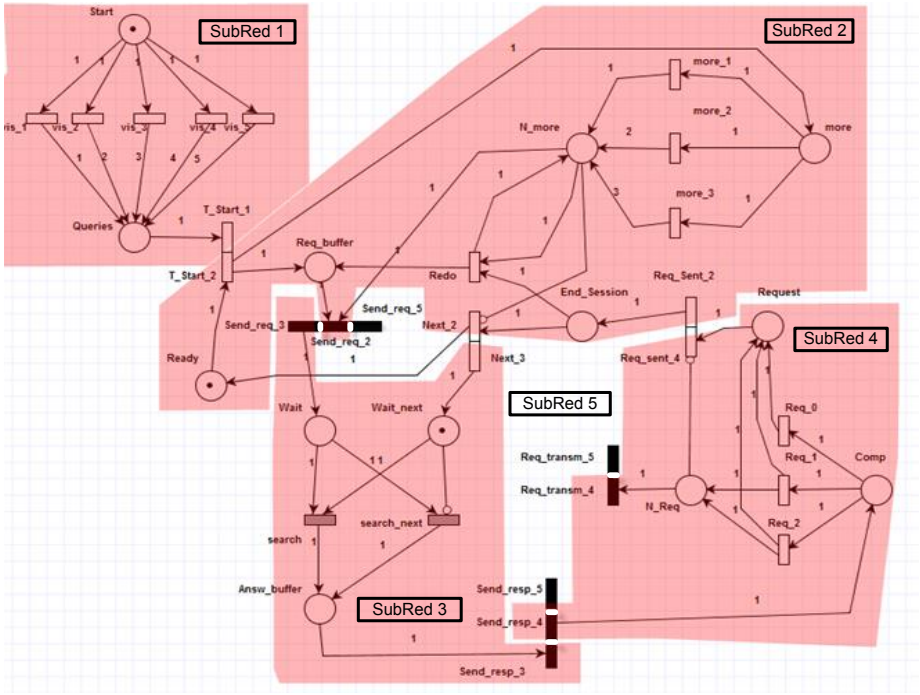

Figura 2: Modelo con RdP dividida para el análisis de búsqueda WEB

En la Figura 2 podemos ver tres tipos de transiciones: transiciones inmediatas representadas por un rectángulo vacío, transiciones temporizadas representadas por un rectángulo gris y las con retardo que son representadas por un rectángulo negro.

Hay que hacer notar que el PP con tiempo [5, 15] original no es capaz de soportar los dos tipos de transiciones con tiempo simultáneamente, mientras que el HPP soporta los dos tipos de transiciones con tiempo en distintas subredes. En concordancia con el concepto de localidad estructural, desarrollado en este trabajo, se ha dividido la red agrupando los distintos tipos de transiciones temporales en una misma subred,

Los recursos empleados para implementar el procesador de Petri temporal (PPT) sin dividir son: 3220.

Plazas= 13, transiciones= 20, comparadores de tiempo 2 x 20 y contadores de tiempo= 1 x 20. Cada comparador y contador de tiempo requiere de 32 bit.

Cantidad de bit para soportar el peso máximo de los brazos 4 bit (incluyendo el signo).

Para simplificar el cálculo de recursos se ha tomado como unidad un FF por bit, por lo que se requiere de 3220 bit para implementar el PP sin dividir, mientras que para implementar el PP jerárquico (HPNP) se requiere de 1384 bit, por lo que se obtiene una ganancia de 232,6%.

Tabla 2: Recursos para la RdP de la figura 2 sin dividir y dividida

	PPT sin dividir	Recursos empleados para implementar las 5 subredes en el HPP				
		Transiciones de la red divididas 6				
		Subred 1	Subred 2	Subred 3	Subred 4	Subred 5
Plazas	13	2	5	3	3	0
Transiciones	20	11	10	8	9	6
PxT (4bit de peso)	13x20x4	2x11x4	5x10x4	3x8x1	3x9x1	1x8x1
Arcos inhibidores	13x20x1	no	5x10x1	3x8x1	3x9x1	no
Transiciones divididas en la subred	NC	1	3	3	2	6
Transiciones no divididas	NC	5	4	2	3	0
Contadores de Tiempo, 32bit	20x32	0	0	8x32	0	6x32
Comparadores de Tiempo 32 bit	2x20x32	0	0	8x32x2	0	0
Recursos necesarios	3220	88	250	792	54	200
subtotales	3220		1384			
Ganancia	232,6%					

6 Conclusión y trabajos futuros

Como conclusión del presente trabajo podemos destacar que: la aplicación del concepto de localidad estructural y división de la red impactan directamente en la reducción de los recursos de hardware empleado para instanciar el IP-Core. Si bien no es posible hacer una generalización, para el caso de análisis que no es el óptimo se ha obtenido un ahorro del 57% en recursos. Esto con respecto a un procesador que instancie todas las posibilidades para todas las transiciones.

La división en subredes, atendiendo a que cada subred sea de una semántica temporal común, permite instanciar RdP heterogéneas haciendo uso de menos recursos de hardware puesto que cada subred tiene los recursos estrictamente necesarios.

Podemos inferir que mientras más heterogéneas sea la semántica de las transiciones, más rectangulares sean las matrices de subredes y menor la cantidad de transiciones divididas mayor es el beneficio.

Según los resultados obtenidos, en las pruebas realizadas usando el IP Core HPNP para sistemas concurrentes y paralelos con tiempo, la mejora en tiempo de respuesta con respecto al uso de semáforos, se traduce en mejoras entre el 40% y el 60%. Esto son los mismos resultados obtenido en los PP.

Finalmente mencionamos, que el uso de la localidad estructural mejora la versatilidad de las RdP para modelar sistemas concurrentes y paralelos, lo que hace que el sistema resultante sea flexible ante cambios en la implementación.

Como trabajo futuro se está desarrollando un algoritmo para la división automática de la red.

REFERENCIAS

1. R. A. B. David R. Martinez, M. Michael Vai, *High Performance Embedded Computing Handbook A Systems Perspective*. Massachusetts Institute of Technology, Lincoln Laboratory, Lexington, Massachusetts, U.S.A.: CRC Press, 2008.
2. M. Domeika, *Software Development for Embedded Multi-core Systems*. 30 Corporate Drive, Suite 400, Burlington, MA 01803, USA Linacre House, Jordan Hill, Oxford OX2 8DP, UK, 2008.
3. M. Diaz, *Petri Nets Fundamental Models, Verification and Applications*. NJ USA: John Wiley & Sons, Inc, 2009.
4. M. Pereyra, M. A. N. Gallia, and O. Micolini, "Heterogeneous Multi-Core System, synchronized by a Petri Processor on FPGA," in *IEEE LATIN AMERICA TRANSACTIONS*, 2013, pp. 218-223.
5. J. N. y. C. R. P. Ornaldo Micolini, "IP Core Para Redes de Petri con Tiempo," *CASIC 2013,* pp. 1097-110, 2013.
6. O. Micolini, "ARQUITECTURA ASIMÉTRICA MULTI CORE CON PROCESADOR DE PETRI," Doctor, Informatica, UNLaP, La Plata, Argentina, 2015.
7. O. Micolini, J. Nonino, and C. R. Pisetta, "IP Core Para Redes de Petri con Tiempo," in *CASIC 2013*, 2013, pp. 1097-110.
8. N. G. M. Pereyra, M. Alasia and O. Micolini, "Heterogeneous Multi-Core System, synchronized by a Petri Processor on FPGA," *IEEE LATIN AMERICA TRANSACTIONS,* vol. 11, pp. 218-223, 2013.
9. O. Micolini, E. Arlettaz, S. H. B. Baudino, and M. Cebollada, "Reducción de recursos para implementar procesadores de redes de Petri," in *en Jaiio 2014*, 2014.
10. S. Pissanetzky, *Sparse Matrix Technology*. Bariloche, Argentina: Centro Atómico Bariloche, 1984.
11. S. Kestury, J. D. Davisz, and E. S. Chungz, "Towards a Universal FPGA Matrix-Vector Multiplication Architecture," *Field-Programmable Custom Computing Machines (FCCM), IEEE 20th Annual International Symposium on,* 2012.
12. G. H. Golub and C. F. V. Loan, *Matrix Computations* Johns Hopkins, 2012.
13. R. Pais, Barros, J.P. ; Gomes, L., "From Petri net models to C implementation of digital controllers " *Emerging Technologies and Factory Automation. ETFA 2005. 10th IEEE Conference on,* 2010.
14. M. Scarpa, A. Puliafito, M. Villari, and A. Zaia, "A modeling technique for the performance analysis of Web searching applications," *Knowledge and Data Engineering, IEEE Transactions on,* vol. 16, pp. 1339-1356, 2004.
15. M. Orlando, J. Nonino, and C. R. Pisetta, "IP Core for Timed Petri Nets," in *CASECongreso Argentino de Sistemas Embebidos*, 2013, pp. 3-8.