Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 121-132
http://publicaciones.uci.cu

Tipo de artículo: Artículo original
Temática: Impacto de las TIC en la sociedad
Recibido: 06/03/2020 | Aceptado: 29/06/2020 | Publicado: 1/08/2020

Procesador generador de combinaciones para Red de Procesadores Distribuidos

Combinations Processor Generator for Network of Evolutionary Processors

José A. Castaño[1*], Yamila Mateu[1], Valery Moreno[2], Alejandro Cabrera[2].

[1] Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños km 2 ½, Reparto Torrens, Boyeros, La Habana. CP 19370. joseantonio@uci.cu , ymateu@uci.cu

[2] Facultad de Ingeniería Automática y Biomédica, Universidad Tecnológica de La Habana "Jose A. Echeverría", CUJAE. Calle 114 No. 11901 e/ Ciclovía y Rotonda, Marianao, La Habana, Cuba. CP 19390. valery.moreno@gmail.com , alex@automatica.cujae.edu.cu

[*] Autor para correspondencia: joseantonio@uci.cu

Resumen

En este trabajo se implementa un Procesador Generador de Combinaciones para una Red de Procesadores Evolutivos (NEP). El objetivo principal es implementar en Hardware un Procesador Generador de Combinaciones y una Red de Procesadores Evolutivos para solucionar el problema de los Tres Colores. El proyecto fue desarrollado utilizando la herramienta ISE Design Suite 14.7 de Xilinx, utilizando como lenguaje de descripción de hardware VHDL. Para la simulación se utilizó la herramienta ISim 14.7, también de Xilinx. La implementación en hardware fue a través de un FPGA Spartan-6 LX45 de Xilinx, contenido en un kit de desarrollo Atlys Board de Digilent. El resultado del trabajo demuestra la viabilidad de la implementación en hardware de algoritmos paralelos para solucionar problemas NP – Completos, utilizando pocos recursos de la FPGA, de forma fiable y rápida. Se muestran imágenes de la simulación realizada y tablas que avalan los resultados.

Palabras clave: Procesador, NEP, FPGA, Algoritmos Paralelos, Problemas NP Completos.

Abstract

In this paper a Processor Generator of Combinations for an Evolutionary Processor Network (NEP) is implemented. The main objective is to implement in hardware a processor that generates combinations of data associated with colors, to solve the Problem of the Three Colors through a NEP, also implemented in hardware. The project was

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 121-132
http://publicaciones.uci.cu

developed using the Xilinx ISE Design Suite 14.7 tool, using VHDL hardware description language. For the simulation, the tool ISim 14.7, also of Xilinx, was used. The hardware implementation was via an Xilinx Spartan-6 LX45 FPGA, contained in an Atlys Board development kit from Digilent. The result of the work demonstrates the feasibility of hardware implementation of parallel algorithms to solve NP - Complete problems, using few resources of the FPGA, in a reliable and fast way. Images of the simulation performed and tables that support the results are shown.

Keywords: *Processor, NEP, FPGA, Parallel Algorithms, NP Complete Problems.*

Introducción

La naturaleza, ha sido fuente indiscutible de inspiración para el ser humano, con el fin de resolver los problemas con los que se ha enfrentado a lo largo de la historia. La búsqueda de métodos o algoritmos para la resolución de problemas complejos, donde la información se presenta de forma masiva e imprecisa ha conllevado al hombre a imitar como éstos son resueltos por la naturaleza. [Martín, 2001]

Se han obtenido al respecto múltiples resultados, siendo ejemplos conocidos de la aplicación del conocimiento obtenido de la biología las Redes Neuronales Artificiales y los Algoritmos Genéticos, o de forma más general, la computación evolutiva. Todos estos modelos inspirados en la naturaleza se han agrupado bajo la denominación de Computación Natural. [Díaz, 2009] Dentro de este ámbito, se encuentran actualmente otros dos modelos computacionales muy importantes: La Computación con ADN (DNA Computing) [Păun, 1998] y la Computación con Membranas (Membrane Computing) [Păun, 2000].

En el caso de la Computación con Membranas, también conocidos como P – Sistemas (Transition P Systems) se procesan multiconjuntos de objetos, de forma sincronizada en los compartimentos delimitados por una estructura de membranas celulares, las cuales determinan una estructura jerarquizada y pueden ser disueltas, divididas, creadas y su permeabilidad puede ser alterada. El resultado de una computación satisfactoria (el sistema alcanza un estado de equilibrio) es el número de objetos (componentes químicos) presentes al finalizar la computación en la membrana de salida que es determinada al comienzo de la computación. Estos sistemas constituyen una clase de modelos de computación masivamente paralela, distribuidos y no deterministas que simulan la forma en que las células procesan los compuestos químicos, la energía y la información. La importancia de estos nuevos sistemas de computación, radica en aprovechar al máximo el paralelismo inherente de los procesos biológicos en los que se inspiran. [Martínez, 2008]

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 121-132
http://publicaciones.uci.cu

Basados en este modelo han surgido múltiples variantes: membranas activas [Păun, 1999] [Pan, 2004], Tissue P Systems [Martín, 2003], Symport/antipor [Ionescu, 2002] y Redes de Procesadores Evolutivos (Network of Evolutionary Processor - NEP) [Castellanos, 2001] [Castaño, 2015].

Las NEP, son modelos de optimización y búsqueda de soluciones, son no deterministas y masivamente paralelos. En este modelo, las células son descritas por un conjunto de palabras, evolucionando producto de mutaciones, las cuales son representadas como operaciones sobre estas palabras, asemejándose a la manera en que este proceso es llevado a cabo en las cadenas de ADN. [Păun, 1998]

Al terminar el proceso de mutación, solo las células con cadenas correctas sobrevivirán, es decir, las que serán soluciones a los problemas. La principal fortaleza de estos modelos radica en la simultaneidad en la ejecución del proceso, para lo cual es necesaria una arquitectura en paralelo y distribuida que consiste de una serie de procesadores simples, los cuales se ubican en los nodos de un grafo completo, capaces de manipular datos asociados con el nodo correspondiente. Cada nodo opera sobre los datos locales con la posibilidad de realizar algún tipo de mutación puntual (inserción, borrado o sustitución de un símbolo), similar a la capacidad de las células para dividirse. Estos nodos actúan como un filtro, cuyo funcionamiento está definido por alguna regla, conocidas como Reglas de Evolución. Los datos locales son entonces enviados a través la red y los que sean capaces de pasar el proceso de filtrado podrán ser comunicados, hasta terminar el proceso. [Castaño, 2015]

Para la realización de este trabajo, se parte del interés de implementar en hardware algoritmos complejos, para demostrar su factibilidad. Como punto de partida se tiene la implementación en Java de una NEP para solucionar el Problema de los Tres Colores, clasificado como NP – Completo, el cual consiste en colorear nodos de un grafo de forma que los vértices y aristas adyacentes no posean el mismo color. [Díaz, 2009] [Castaño, 2015]

Este trabajo tiene como objetivo principal, implementar en Hardware un Procesador Generador de Combinaciones y una Red de Procesadores Evolutivos para solucionar el problema de los Tres Colores.

Como parte del trabajo se implementa en Hardware también, una memoria donde se almacenarán los resultados de la aplicación de este modelo computacional.

Materiales, métodos y metodología computacional.

Para solucionar el Problema de los Tres Colores a través de la implementación de una NEP, se tiene que el grafo del problema es el siguiente:

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 121-132
http://publicaciones.uci.cu

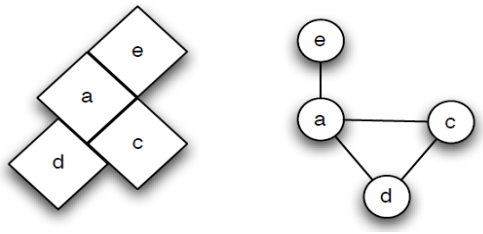

Figura 1. Grafo del problema a resolver.

El grafo se define como: $G = (\{1,2, \ldots, n\}, \{e_1, e_2, \ldots, e_m\})$ con $e_t = \{k_t, l_t\}$, $1 \leq k_t \leq l_t \leq n$, $1 \leq t \leq m$. Se considera el alfabeto $U = V \cup V' \cup T \cup A$, donde $V = \{b, r, g\}$, $T = \{a_1, a_2, \ldots, a_n\}$ y $A = \{A_1, A_2, \ldots, A_n\}$.

Se construyen los siguientes procesadores de un NEP:

- Un procesador generador, encargado de generar todas las posibles combinaciones de colores, soluciones o no al problema y enviarlas a los siguientes procesadores. Se define como:

  $N_0 = \{\{ a_1, a_2, \ldots, a_n \}, \{a_i \rightarrow bA_i, a_i \rightarrow rA_i, a_i \rightarrow gA_i \mid 1 \leq i \leq n\}, \quad 0, \{A_i \mid 1 \leq i \leq n \}\}$

- Cuatro procesadores de filtrado para cada arista del grafo $e_t = \{k_t, l_t\}$ (donde i = $\{k_t, l_t\}$):
  - $Ne_t^1 = \{0, \{gA_i \rightarrow g'a_i, rA_i \rightarrow r'a_i\}, \{ A_i\}, \{ g', r'\}\}$
  - $Ne_t^2 = \{0, \{gA_i \rightarrow g'a_i, bA_i \rightarrow b'a_i\}, \{ A_i\}, \{ g', b\}\}$
  - $Ne_t^3 = \{0, \{bA_i \rightarrow b'a_i, rA_i \rightarrow r'a_i\}, \{ A_i\}, \{ b', r'\}\}$
  - $Ne_t^4 = \{0, \{ r'a_i \rightarrow rA_i, g'a_i \rightarrow gA_i, b'a_i \rightarrow bA_i \}, \{ a_i\}, \{ A_i\}\}$

Es decir, el procesador N_0 genera todas las cadenas que codifican posibles combinaciones de color para cada par de nodos y posteriormente los procesadores Ne_t^1, Ne_t^2, Ne_t^3, Ne_t^4 filtran dichas cadenas en la arista e_t. La repetición de dicho proceso de filtrado para el resto de las aristas provoca la obtención de la solución al problema. Por tanto, en cada paso evolutivo (filtro) se obtienen combinaciones de colores posibles soluciones al problema, aunque no sean soluciones definitivas y al terminar todo el proceso evolutivo (paso por todos los filtros) es que se obtiene la solución final. [Castaño, 2015] [Díaz, 2008]

La arquitectura de la NEP que se define para el Problema de los Tres Colores, a partir del grafo G, es la siguiente:

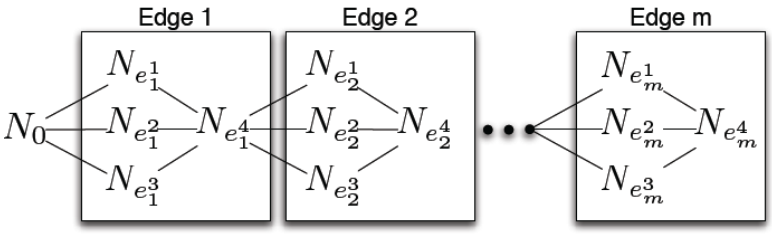

Figura 2. Arquitectura de la NEP.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 121-132
http://publicaciones.uci.cu

Para la implementación del Procesador Generador de Combinaciones para la Red de Procesadores Evolutivos para solucionar el problema antes mencionado, se deben diseñar tres bloques: Procesador N_0, Bloque de Filtrado y una Memoria donde se almacenarán las combinaciones soluciones al problema. El diagrama en bloques se muestra a continuación.

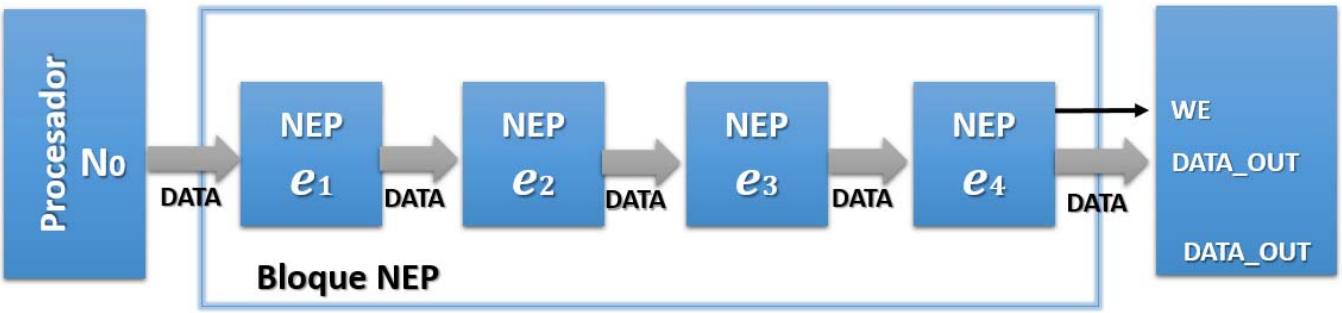

Figura 3. Diagrama en Bloques del Sistema.

El dispositivo en el cual se implantará físicamente el sistema es un FPGA Spartan-6 LX45 de Xilinx [Xilinx, 2011], empotrado en un kit de desarrollo Atlys Board de Digilent [Digilent, 2013]. El proyecto fue desarrollado utilizando la herramienta ISE Design Suite 14.7 de Xilinx [Xilinx, 2013], utilizando como lenguaje de descripción de hardware al VHDL. Se realizó la descripción comportamental de cada bloque y un diseño jerárquico basado en componentes [Ashenden, 1995]. Igualmente de Xilinx, se utilizó la herramienta ISim 14.7 para la simulación.

Procesador.

El Bloque Procesador es el encargado de generar las combinaciones de colores que serán analizadas por el Bloque NEP, para determinar si son soluciones o no al problema. Para conformar el dato asociado a las combinaciones de colores, se establece el siguiente formato:

<color><NODO A> <color><NODO C> <color><NODO D> <color><NODO E>

Por tanto, una combinación de colores es, por ejemplo: bAgCrDbE. En este caso el nodo A está coloreado en azul (blue), el nodo C en verde (green), el D en rojo (red) y el E en azul (blue). Nótese que se toma para el caso de los colores la primera letra del mismo en inglés.

Para trabajar con estos datos en formato binario, se codificaron los colores y los nodos con 2 bits cada uno. Esto implica que las cadenas que codifican combinaciones de colores tengan 16 bits, 4 bits por cada formato <color><NODO>.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 121-132
http://publicaciones.uci.cu

Los nodos fueron codificados de la siguiente manera: 00 - Nodo A, 01 - Nodo C, 10 - Nodo D y 11 - Nodo E. Por su parte, los colores se codificaron de la siguiente forma: 00 – Rojo, 01 – Verde, 10 - Azul y 11 - Sin color. La combinación 11 representa el caso en que un nodo no está coloreado. En la siguiente tabla, se representan todos los valores de combinaciones <color><NODO> posibles.

Colores Nodo A	**Nodo A**	**Colores Nodo C**	**Nodo C**	**Colores Nodo D**	**Nodo D**	**Colores Nodo E**	**Nodo E**
00	00	00	01	00	10	00	11
01		01		01		01	
10		10		10		10	
11		11		11		11	
Nibble 1		Nibble 2		Nibble 3		Nibble 4	

Tabla 1. Combinaciones posibles de colores y nodos.

De la tabla se deduce, que como los valores de los Nodos no varían en el formato <color><NODO>, se tiene que cada Nibble tendrá sólo 4 combinaciones posibles, siendo los bits asociados a los colores los únicos que cambian. Por tanto: 4 Nibble x 2 bit asociados a colores = 8 bit asociados a los colores en toda la palabra de datos, lo cual implica $2^8 = 256$ combinaciones posibles de colores.

Para generar estas 256 combinaciones se utiliza un contador binario de 8 bit, sincronizado con el reloj del sistema (FPGA). Cada byte generado se desglosa por cada 2 bit, asumiendo lo siguiente:

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Color Nodo A		Color Nodo C		Color Nodo D		Color Nodo E	

Tabla 2. Asignaciones de Bits por Nodos, del Byte generado por el Contador.

Estos valores se insertan en la palabra de datos mediante el siguiente código, aprovechando las facilidades del Lenguaje VHDL [Ashenden, 1995]:

```
-- Asignación fija de los datos binarios de los nodos.
        nodo_in(13 downto 12)<= "00";     -- Nodo A.
        nodo_in(9 downto 8)<= "01";       -- Nodo C.
        nodo_in(5 downto 4)<= "10";       -- Nodo D.
        nodo_in(1 downto 0)<= "11";       -- Nodo E.
  -- Asignación de los datos binarios de los colores.
        nodo_in(15 downto 14) <= generador(7 downto 6);   -- Color Nodo A.
        nodo_in(11 downto 10) <= generador(5 downto 4);   -- Color Nodo C.
```

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 121-132
http://publicaciones.uci.cu

```
        nodo_in(7 downto 6) <= generador(3 downto 2);      -- Color Nodo D.
        nodo_in(3 downto 2) <= generador(1 downto 0);      -- Color Nodo E.
    -- Asignación de los datos binarios en formato de 16 bit, con las combinaciones de los Nodos y los colores.
        combinacion <= nodo_in;
```

De esta forma se obtienen las 256 combinaciones posibles del formato: <color><NODO A> <color><NODO C> <color><NODO D> <color><NODO E>. Estas palabras, pueden ser soluciones o no al Problema de los Tres Colores. En el Bloque NEP se realiza el proceso de filtrado y ahí se determinan cuáles son las combinaciones soluciones al problema.

Para la programación del Contador en Lenguaje VHDL se utiliza la directiva Process. Esta ejecuta internamente sus sentencias de forma secuencial, debido a existencia del reloj síncrono. Sin embargo, el Process insertado dentro del sistema desarrollado funciona como una entidad de ejecución paralela.

Bloque NEP.

Para el proceso de filtrado se diseña el Bloque NEP, el cual cuenta con 4 bloques internos. Cada uno de estos bloques está en función de una arista (e_t) del grafo del problema. En estos, es donde se implementan las reglas de evolución y se nombran NEP e1 (Ne_t^1), NEP e2 (Ne_t^2), NEP e3 (Ne_t^3) y NEP e4 (Ne_t^4). Internamente cada bloque de filtrado está formado por 4 procesadores evolutivos. Es decir, las combinaciones de colores deben ser analizadas por 16 procesadores evolutivos para determinar si son solución o no al problema.

La arquitectura de cada bloque de procesamiento (filtrado), implementados en VHDL a partir de la descripción de su estructura, se muestra a continuación.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 121-132
http://publicaciones.uci.cu

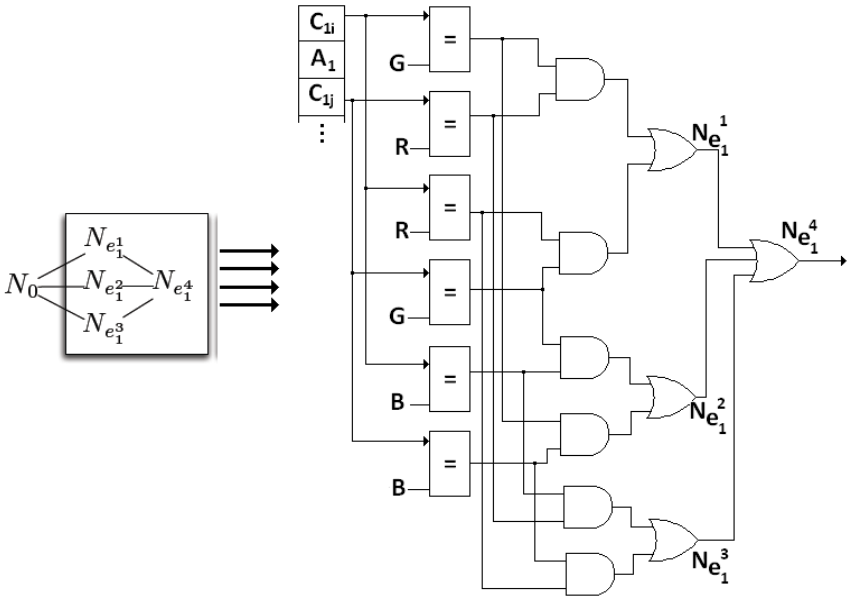

Figura 4. Arquitectura de los bloques de filtrado.

En la figura anterior se puede apreciar la implementación de las reglas de evolución, donde A_1 es la arista en análisis (e_t) y C_{1i} y C_{1j} son los colores de los nodos en los extremos de la arista.

El bloque de filtrado NEP e_1 (Ne_{t1}), es el encargado de recibir el dato procedente del Procesador N_0. En este se valida la pertinencia de los datos asociados a los colores y los nodos de toda la combinación de colores. Si los datos anteriores son válidos y se cumplen las reglas de evolución para la arista et1, el dato pasa al bloque siguiente en su formato original. De lo contrario, lo que se transfiere es el valor "0000" hexadecimal, lo cual indica que la cadena recibida no es solución al problema. Lo anterior significa, que la célula no sobrevivió al proceso de mutación.

Los bloques NEP e_2 (Ne_{t2}) y NEP e_3 (Ne_{t3}) trabajan de igual forma, actuando sobre sus aristas e_{t2} y e_{t3} respectivamente. En ambos casos se recibe la combinación de colores procedente del bloque anterior, si se cumplen las reglas de evolución, el dato se transmite al siguiente. De lo contrario, también se transmite el dato "0000" hexadecimal.

Al final, el bloque NEP e_4 (Ne_{t4}) recibe la cadena procedente de NEP e_3 (Ne_{t3}) para filtrar la arista e_{t4}. Si se cumplen las reglas de evolución, entonces la combinación de colores recibida es solución al problema, por ser este el último paso evolutivo. Se procede entonces a enviar el dato para su almacenamiento en la memoria RAM_OUT y se habilita la escritura (WE) en dicha memoria. Si el dato no cumpliera con las reglas, no se guarda nada en RAM_OUT.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 121-132
http://publicaciones.uci.cu

En este proyecto, el procesamiento de las combinaciones de colores entre los bloques de filtrado se realiza de forma serie. Esto se debe a que cada bloque solo puede comprobar el cumplimiento de las reglas de evolución de su arista (et) cuando le llega el dato desde el bloque anterior. Sin embargo, el procesamiento interno de cada bloque de filtrado es en paralelo, pues la salida toma valores según la concurrencia con que cambie la entrada. [Castaño, 2015]

Bloque RAM_OUT.

Este bloque corresponde a la memoria donde se almacenan las combinaciones de colores generadas por el procesador N_0 que cumplen con todas las reglas de evolución, es decir, las soluciones al problema. Esta memoria se implementa a partir de la descripción de su comportamiento en VHDL y tiene un tamaño de 8 Kb. También se ubica dentro del FPGA en uno de los bloques predefinidos para memorias de este tipo.

Resultados y discusión

Luego del diseño del sistema, se procede con la síntesis e implementación, utilizando el ISE 14.7. Una vez implementado se realiza la simulación, donde se comprueba el correcto funcionamiento del Procesador Generador, los Bloques de Filtrado y la Memoria.

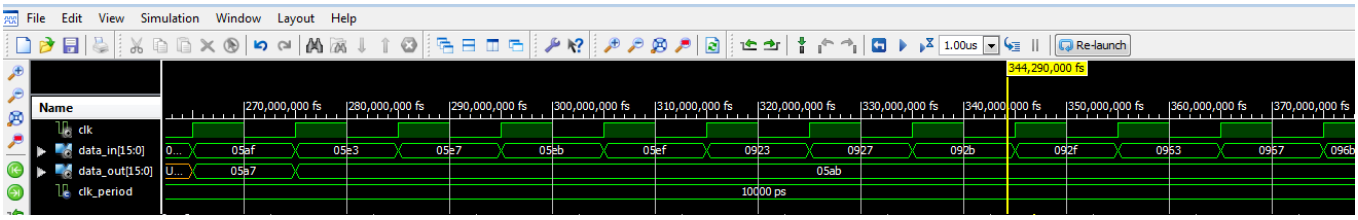

Figura 5. Simulación del sistema.

En la figura 5 se muestra una vista de la simulación del sistema. Nótense las señales:

- clk: reloj del sistema.
- data_in: señal generada por el Procesador, con las combinaciones de colores.
- data_out: datos resultantes del filtrado guardados en la memoria, es decir, las soluciones al problema de Los Tres Colores.

Las soluciones al Problema de los Tres Colores, a partir del grafo *G,* se muestran en la Tabla 3. Estos valores son los contenidos en la Memoria RAM_OUT al terminar el proceso de filtrado, es decir, los resultantes luego de la aplicación de las reglas de evolución. Se muestran los datos en hexadecimal, para facilitar la comprensión de los datos.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 121-132
http://publicaciones.uci.cu

No.	Combinación	Dato Binario	Dato Hexadecimal
1	rAbCgDbE	0000010110100111	05A7
2	gAbCrDbE	0100100100101011	492B
3	gAbCrDrE	1000010100100011	8523
4	rAgCbDgE	1000000101100011	8163
5	bAgCrDgE	0100000110100011	41A3
6	rAgCbDbE	1000000101100111	8167
7	rAbCgDgE	0100000110101011	41AB
8	bAgCrDrE	0000100101100111	0967
9	bArCgDrE	0000100101101011	096B
10	gArCbDrE	0100100100100011	4923
11	bArCgDgE	1000010100100111	8527
12	gArCbDbE	0000010110101011	05AB

Tabla 3. Soluciones al Problema de los Tres Colores para el Grafo *G*.

Como resultado de la implementación, se comprobó una baja utilización de los componentes internos del FPGA. En la figura 6 se muestra una tabla obtenida del ISE, con un resumen de la utilización de recursos internos del dispositivo empleado.

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	16	4800	0%
Number of Slice LUTs	32	2400	1%
Number of fully used LUT-FF pairs	16	32	50%
Number of bonded IOBs	33	132	25%
Number of Block RAM/FIFO	1	12	8%
Number of BUFG/BUFGCTRLs	1	16	6%

Figura 6. Resumen de la utilización de la FPGA.

Los resultados demuestran que el algoritmo implementado en hardware para solucionar el Problema de los Tres Colores, arrojó las debidas soluciones para el Grafo *G* y utilizando muy pocos recursos del FPGA, lo cual demuestra la factibilidad de la solución propuesta.

Conclusiones

El trabajo presentado, cumplió con su objetivo principal: implementar en hardware un Procesador Generador de combinaciones de colores para una Red de Procesadores Evolutivos.

A través de la NEP, se solucionó el Problema de Los Tres Colores, logrando también su implementación en hardware. Se implementó además, una memoria donde se almacenaron los datos resultados de la solución al problema analizado. El FPGA empleado permitió la implementación de un algoritmo que para el procesamiento de los datos de forma paralela, garantizando rapidez en la ejecución del algoritmo, con una muy baja utilización de recursos internos.
Todo lo anterior demuestra la viabilidad de la implementación de las NEP en dispositivos de hardware y su utilización como algoritmo para la solución de problemas complejos.

Referencias

1. [Castaño, 2015] José A. Castaño, Valery Moreno, Alejandro Cabrera, Abraham Gutiérrez y Víctor Martínez. "Red de Procesadores Evolutivos para solucionar el Problema de los Tres Colores. Implementación en Hardware". Revista Cubana de Ciencias informáticas, año 2015, Vol. 9, Núm. 4.
2. [Păun, 2000]. Gheorghe Păun. "Computing with Membranes". Journal of Computer and System Science, año 2000, Vol. 61. Páginas 108 – 143.
3. [Păun, 1999]. Gheorghe Păun. "P Systems with Active Membranes: Attacking NP Complete Problems". Research Report Series, mayo de 1999, CDMTCS – 102.
4. [PAN, 2004]. Linqiang PAN y Tseren-Onolt ISHDORJ "P Systems with Active Membranes and Separation Rules". Journal of Universal Computer Science, año 2004, Vol. 10. Páginas 325 – 341.
5. [Martín, 2003]. Carlos Martín-Vide, Gheorghe Păun, Juan Pazos y Alfonso Rodríguez Patón. "Tissue P systems". Theoretical Computer Science, año 2003, Vol. 296. Páginas 295 – 326.
6. [Ionescu, 2002] Mihai Ionescu, Carlos Martín-Vide, Gheorghe Păun. "P Systems with Symport/Antiport Rules: The Traces of Objects". A Journal of Mathematical Research on Formal and Natural Languages, año 2002, Vol. 5. Páginas 65-79. ISSN: 1386-7393
7. [Castellanos, 2001] J. Castellanos, C. Martín-Vide, V. Mitrana, and J. Sempere. "Solving np-complete problems with networks of evolutionary processors". Lecture Notes in Computer Science, año 2001. Páginas 621- 628.
8. [Ashenden, 1995] Ashenden, Peter J. "The Designer's Guide to VHDL". Morgan Kaufmann Publishers, San Francisco, año 1995, ISBN 1-55860-270-4.

9. [Martín, 2001] Bonifacio Martín del Brío y Alfredo Sanz Molina. "Redes Neuronales y Sistemas Difusos". Editorial Alfaomega, Saragoza, año 2001. Páginas 8 - 10.

10. [Păun,1998] Păun, Gh., Rozenberg, G., & Salomaa, A. "DNA Computing: New Computing Paradigms", Springer Verlag, Berlin, año 1998.

11. [Díaz, 2009] Díaz Martínez, Miguel Ángel. "Modelo de Computación Conexionista Inspirado en las Redes de Procesadores Evolutivos y su Aprendizaje". Tesis Doctoral, Universidad Politécnica de Madrid, España, año 2008.

12. [Martínez, 2008] Martínez Hernando, Víctor. (2008). "Desarrollo de Sistemas Físicos para Implantar Modelos de Computación con Membranas". Tesis Doctoral, Universidad Politécnica de Madrid, España, año 2008.

13. [Xilinx, 2011] Xilinx, Product Specification, DataSheets DS160 (v2.0), año 2011. [Consultado el: 21 de octubre de 2017]. Disponible en https://www.xilinx.com/support/documentation/data_sheets/ds160.pdf

14. [Digilent, 2013] Digilent. Refference Manuall, Revision: August 5, año 2013. [Consultado el: 21 de octubre de 2017]. Disponible en https://reference.digilentinc.com/atlys/atlys/refmanual

15. [Xilinx, 2013] Xilinx, ISE Design Suite 14: Release Notes, Installation, and Licensing UG631 (v14.7), año 2013. [Consultado el: 21 de octubre de 2017]. Disponible en https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/irn.pdf