Citación: Rodríguez, C. D. y Melgarejo, M. A.(2015). Arquitectura FPGA para simulación de aprovisionamiento de alimentos en colonias de hormigas artificiales. En: Ingeniería, Vol. 20, No. 2, pp. 245–260

En línea DOI: http://dx.doi.org/10.14483/udistrital.jour.reving.2015.1.a05

Artículo de Investigación

Arquitectura FPGA para simulación de aprovisionamiento de alimentos en colonias de hormigas artificiales

A FPGA architecture for foraging behavior in simulation and colonies

Cristian David Rodríguez Rodríguez
Universidad Distrital
Francisco José de Caldas
Facultad de Ingeniería
cridrodriguezr@correo.udistrital.edu.co

Miguel Alberto Melgarejo Rey
Universidad Distrital
Francisco José de Caldas
Facultad de Ingeniería
mmelgarejo@ieee.org

Resumen

Se presentan algunos resultados en relación con el diseño y la implementación de una arquitectura que soporta una plataforma experimental para simular el proceso de alimentación de las colonias de hormigas. Los algoritmos Ant-system y Ant-Cycle modelan el comportamiento de las hormigas. La plataforma permite cambiar parámetros como la cantidad y la velocidad de las hormigas, la cantidad y ubicación de los alimentos y el radio y la frecuencia de difusión de la feromona de las hormigas. Dichos parámetros se visualizan a través de una interfaz VGA. La implementación de hardware se lleva a cabo sobre tecnología FPGA de Xilinx©. La teoría detrás de este diseño considera que los comportamientos complejos pueden surgir de sistemas con una estructura simple. Este trabajo se enfrenta a la pregunta por la complejidad global emergente de un sistema cuya complejidad estructural es mínima o inexistente.

Palabras claves: colonia, FPGA, hormigas, procesador, sistemas biológicos

Abstract

This paper presents some results regarding the desing and implementation of an architecture that supports an experimental platform for simulating the foraging process of ant colonies. Both the Ant-System and the Ant-Cycle algorithms model the behavior of ants. The platform allows to change parameters like the quantity and speed of ants, the amount and location of food and the ratio and difussion frequency of ant pheromone. These parameters are visualized through a VGA interface. The hardware implementation is carried out over FPGA Xilinx© technology. Theory behind this design considers that complex behaviors can emerge from systems with simple structure. This work confronts the question about global complexity emerging from a system whose structural complexity is minimal or inexistent.

Key words: ants, biological systems, colony, FPGA, processor

Recibido: 15-05-2015
Modificado: 28-07-2015
Aceptado: 24-08-2015

1. Introducción

Las colonias de hormigas son sistemas sociales inteligentes, su comportamiento emerge de la cooperación entre la especie, una sola hormiga no es inteligente [1]. Su sistema comportamental tiene como fin la recolección de alimento, para lo cual emplea un proceso de riego y detección de feromona que le permite crear y mejorar trayectos de recolección. Este fenómeno tiene su propia técnica de optimización en aplicaciones computacionales, el llamado ACO (Ant Colony Optimization) [2].

En la actualidad los algoritmos de colonias de hormigas se han orientado hacia la solución de problemas computacionales que pueden reducirse a encontrar la mejor ruta en un grafo [3]. Fueron inicialmente propuestos por Dorigo en 1992 [4] y motivo de diversas investigaciones en esta década [5], [6]. Dichos algoritmos han tenido un gran campo de aplicación sobre la tecnología de hardware configurable FPGA [7], [8] debido a factores como el paralelismo del hardware, los tiempos de respuesta y la facilidad de mantenimiento [9]. No obstante, las actuales investigaciones se alejan un poco del estudio de las colonias enfocándose en el marcado campo de la optimización de grafos.

El presente trabajo aborda una propuesta arquitectural de hardware y su correspondiente implementación, que permite retomar el estudio de las colonias de hormigas por medio de un entorno gráfico experimental basado en la interacción de hormigas artificiales. En el centro de la propuesta se encuentra que el comportamiento de una sola hormiga puede ser capturado por medio de una máquina de estados algorítmica a partir del algoritmo ACO Ant-Cycle System. La colonia de hormigas se concibe entonces como un conjunto de máquinas de estado que interactúan entre sí por medio de una memoria distribuida asociada al espacio de recolección de comida de las hormigas; en este sentido se estaría hablando de una arquitectura de procesamiento con capacidades de cómputo emergente, de ahí que sea denominado un procesador y no un emulador.

En esta línea de investigación existe un modelo para emular colonias de hormigas diseñada sobre el software Netlogo [10]; sin embargo, aunque presenta varios beneficios que serán punto de partida en el diseño propuesto en el artículo, no permite redefinir factores como la posición o cantidad de alimento, lo cual es un gran problema en el momento de formular experimentos. El hecho de proponer una implementación hardware sobre FPGA, en vez de una software, es una gran mejora en la medida que es posible supervisar cualquier señal (variable) del sistema, gracias a la herramienta Scope® incluida en la interfaz ISE® de Xilinx©. Agregando a esto las ventajas intrínsecas de la tecnología, mencionadas anteriormente.

En el campo de implementaciones ACO sobre FPGA se encuentran aplicaciones similares pero orientadas a grafos [6,7], la principal diferencia con respecto al desarrollo más reciente [7] es el modelo de feromona. La arquitectura propuesta en el presente artículo permite que el rastro de feromona sea depositado, tanto en los nodos de referencia para decisión de movimiento i y j como en el camino intermedio entre estos nodos, a diferencia de [7], en donde el rastro de feromona se deposita únicamente sobre los nodos i y j; se inserta este modelo con el objetivo de replicar el comportamiento de la colonia de hormigas lo más cercano posible a la realidad [15]. La organización global del flujo de acciones en la ejecución del algoritmo y la variante de ACO implementada coinciden para ambos desarrollos.

El modelamiento de colonias de hormigas se incluye dentro de la rama inteligencia de enjambres la cual ha generado un especial interés en aplicaciones de ingeniería. A través de los métodos de cooperación, auto organización y control descentralizado abstraídos del comportamiento de las hormigas se han obtenido resultados y aportes en la mejora de la precisión en el diseño de sistemas difusos [11], búsqueda de rutas de bajo consumo energético para robots móviles [12], optimización en rutas de abastecimiento de agua para embalses de emergencia [13] y diseño de sistemas de recolección de basura multi-robótica [14].

En busca de soluciones a distintas aplicaciones, en el estudio de los algoritmos de hormigas se hace necesario realizar pruebas bajo un simulador. En la mayoría de casos estas pruebas se realizan sobre mapas de grafos, los cuales son una aproximación muy útil al comportamiento de las hormigas, pero distan en diferentes aspectos del comportamiento real [15]. El desarrollo que aquí se presenta se muestra como una primera posibilidad de acercarse a una solución para este problema.

El artículo se organiza de la siguiente manera: en primer lugar se define un procesador basado en la interacción de ASM's que modela el flujo de decisiones y acciones de las hormigas y su entorno, luego se presenta el diseño de una unidad gráfica y sus correspondientes acondicionamientos para el procesador anterior. Con estos dos procesadores trabajando en paralelo se introduce el comportamiento basado en heurística ACO, adaptándolo a los recursos disponibles en hardware. Finalmente se analizan los resultados esperados y obtenidos haciendo énfasis en la discusión sobre la complejidad emergente en el sistema.

2. Modelo complejo basado en ASM's para la colonia de hormigas

2.1. Máquinas de estados algorítmicas

Una Algorithmic State Machine (ASM, por sus siglas en inglés) se define como el conjunto de estados, transiciones, entradas y salidas que permiten modelar la dinámica de un sistema de procesamiento digital [16]. La ASM permite especificar la secuencia de tareas y decisiones que contiene el algoritmo solución de determinado problema y se encarga de traducir el problema planteado a un diagrama de información secuencial sujeto a las condiciones de ejecución. Para lograr este propósito divide las tareas en un número finito de estados de acción y describe las condiciones para pasar de un estado a otro; cada estado, a excepción del estado inicial, debe tener un estado pasado y uno futuro, puede darse el caso en que el mismo estado sea su anterior o siguiente. En cada estado se guarda concurrencia en las operaciones realizadas, dado que contiene circuitos combinacionales y, además, se guarda memoria del estado anterior, pues contiene circuitos secuenciales.

2.2. Planteamiento del problema

En una colonia de hormigas se diferencian tres grupos: reina, obreras y zánganos. El grupo de interés son las obreras, específicamente su proceso de recolección de alimento. Las hormi-

gas buscan alimento estocásticamente, seleccionan su trayectoria con base en la información de feromona contenida en el espacio de búsqueda. Las trayectorias varían, puesto que cada una deposita feromona apoyada en su conocimiento específico del entorno, es decir, la información que haya adquirido. Como resultado de este proceso, indirectamente cada hormiga puede comunicarse con las demás, esta comunicación les permite a las hormigas ser atraídas hacia las mejores soluciones. La búsqueda eficiente se logra gracias a la intensificación del camino más corto por la realimentación positiva de la información de feromona por selección estocástica; la realimentación positiva se produce porque el camino solución se refuerza de feromona más rápido que los otros, al requerir un menor tiempo para ser completado. Este camino se convierte en el único, pasado determinado tiempo [1].

2.3. Modelo basado en ASM's

Para resolver el problema, se diseñan dos ASM. La primera modela una sola hormiga, la segunda usa la ASM anterior para cada una de las demás. Una ASM tipifica el cerebro de la hormiga y la otra reutiliza este cerebro para todas las hormigas de la colonia; esta propuesta requiere almacenar la información necesaria de cada hormiga, como su posición, dirección, actualización de feromona y objetivo actual en una memoria.

2.3.1. ASM para modelar el comportamiento de la hormiga obrera

En la figura 1 se muestra el diagrama de estados que especifica las acciones que debe realizar la hormiga obrera en el proceso de recolección de alimento.

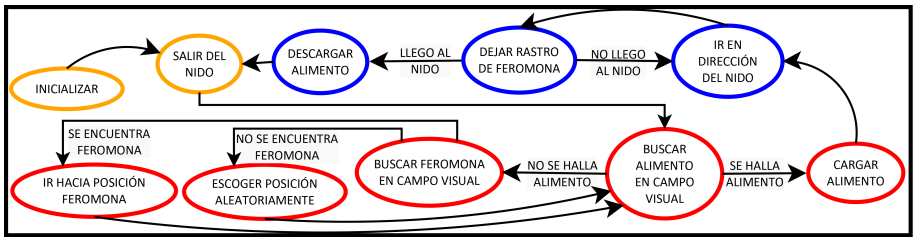

Figura 1. Diagrama funcional de la hormiga obrera.

Dentro del flujo de estados se diferencian dos grupos importantes, que dependen básicamente de tener o no tener comida. Si la hormiga no tiene comida realiza todo el proceso de búsqueda de la misma (estados de borde rojo); en caso contrario, realiza todo el proceso de búsqueda del nido (estados de borde azul). Este dato indica qué secuencia de acciones debe realizar la hormiga. En la ASM número dos, se detallan las modificaciones realizadas para que las hormigas recolecten alimento al mismo tiempo usando una única máquina de estados.

2.3.2. ASM para reutilizar el comportamiento de la hormiga obrera

Para reutilizar el diagrama de la figura 1 en todas las hormigas, se modifica el flujo de acciones agrupando los estados de igual color, si la hormiga tiene comida realiza el flujo

de estados de color rojo únicamente y en caso contrario el flujo de estados de color azul. Se agregan estados que admiten que cada una de las hormigas obreras pueda realizar una única acción simultáneamente. Esto permite, dada la velocidad con que funciona el ASM, crear la sensación de que todas trabajan simultáneamente; el selector de hormigas hace que pase hormiga por hormiga y realice su correspondiente acción, cuando cada hormiga haya realizado alguno de los flujos de acciones, se procede a deteriorar el rastro de feromona, si es que existe, y se entra en un bucle de espera, en seguida se reinicia el proceso, actualizando la información almacenada de cada hormiga.

El diagrama propuesto para modelar la segunda ASM se presenta en la figura 2.

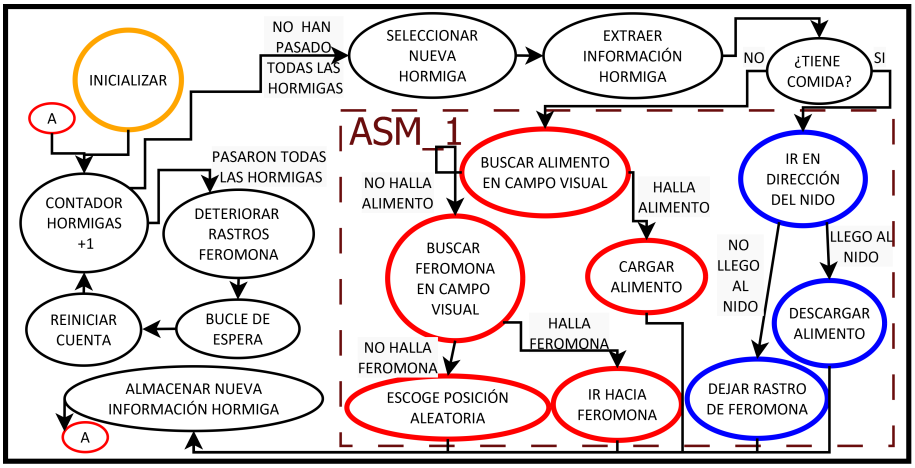

Figura 2. Diagrama funcional para la colonia de hormigas.

2.4. Acondicionamiento de las acciones básicas de la hormiga para la interfaz gráfica

A continuación se describen los diseños realizados para crear la interfaz gráfica de la hormiga, que se puede observar en [21 y 22].

A fin de generar la interfaz gráfica se crea una matriz 256 X 256 que se visualiza en pantalla y tiene dieciséis colores posibles por elemento de la matriz. La colonia se ubica en el centro del entorno, el lugar de la comida es aleatorio y se actualiza su cantidad a medida que se desabastece el entorno.

En la figura 3.a. se evidencia que toda hormiga dentro de esta matriz 256 X 256 tiene coordenada X, Y, que permite conocer su ubicación y una dirección de ocho posibles. La gráfica de la hormiga se forma por cuatro tonalidades de rojo.

En la figura 3.b. se pone de manifiesto que para realizar un movimiento la hormiga elige entre tomar la misma dirección, siguiente o anterior en la secuencia. Con el propósito de elegir la nueva coordenada se le suma a la actual el valor necesario para ir en la dirección seleccionada; por ejemplo, si su dirección es 0 puede tomar dirección 0, 1 o 7 afectando las coordenadas por la suma vectorial de la actual y (-6,0) para dirección 0 (-6, 3) para dirección 7 o (-6,-4) para dirección 1.

Al encontrar alimento inicia un proceso de regreso a la colonia. Conoce con cierto margen de error la posición de su colonia, se ubica en dirección a ella y toma caminos que le permitan llegar lo más rápido posible.

En la figura 3.c. se ve que en paralelo al proceso anterior la hormiga debe depositar el rastro de feromona. La feromona se forma por ocho tonalidades, que van desde el blanco hasta el negro, pasando por matices de verde. Para generar la gráfica, se reúsa la forma de la hormiga, cambiando las cuatro tonalidades de rojo por las cuatro tonalidades más cercanas al blanco de la feromona. Para degradar el rastro, cada dos movimientos en la colonia de hormigas, se cambia la tonalidad de los pixeles con feromona al inferior en secuencia de blanco a negro, de tal manera que luego de doce movimientos de la hormiga desaparezca por completo el rastro dejado inicialmente en un movimiento.

En el proceso de detección de feromona, figura 3.d., se considera un campo de visión para la hormiga de aproximadamente el doble de su tamaño, en la dirección de su movimiento. La zona con mayor concentración de feromona se elige como nueva dirección actualizando la coordenada de ubicación, como se detalló anteriormente.

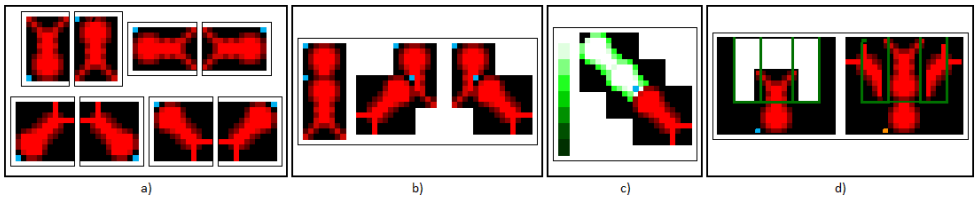

Figura 3. a) Direcciones posibles para la hormiga obrera. b) Direcciones que puede elegir la hormiga para su siguiente movimiento. Solo puede escoger nuevas direcciones con diferencia de 45° o 0° con respecto a la actual. c) Rastro de feromona depositado por la hormiga obrera. d) Campo de visión para detección de feromona. Se encuentra dividido en tres zonas delimitadas por los recuadros verdes.

2.5. Metaheurística ACO

La Metaheurística ACO está inspirada en la observación del comportamiento de hormigas reales considerándose un algoritmo de inteligencia colectiva, estos algoritmos se hallan compuestos de individuos simples que cooperan sin ninguna forma de control directo a través de la auto-organización [2].

El componente central de un algoritmo ACO es el modelo de feromona. Las hormigas artificiales construyen una solución para el problema de recolección en el entorno, llamado grafo de construcción; dicho grafo se encuentra formado por un conjunto de vértices y arcos, las hormigas se mueven de un vértice a otro a lo largo de los arcos del grafo construyendo incrementalmente una solución parcial. En estos desplazamientos depositan una cierta cantidad de feromona sobre los arcos y vértices que han visitado, la cantidad de feromona depositada puede depender de la calidad de la solución encontrada.

Entre los principales algoritmos ACO disponibles se encuentran Ant System, MAX-MIN Ant System y Ant Colony System. Se hace uso del modelo Ant System puesto que provee un mayor acercamiento al comportamiento de una colonia de hormigas que a la modelación de problemas de optimización combinatoria; Ant System presenta tres variantes que distan en

la manera de actualizar los rastros de feromona, se escoge Ant-Cycle porque a diferencia a los otros dos deposita feromona luego de encontrar una solución y no en la búsqueda de esta [17,18].

En AS k hormigas buscan una trayectoria solución concurrentemente. La hormiga k aplica una regla de elección de acción estocástica para decidir cuál debe ser su siguiente movimiento o vértice a visitar, la probabilidad con la cual dicha hormiga, estando en el vértice i, elige visitar el vértice j es:

$$P_{(i,j)}^{k} = \frac{[T_{i,j}]^{\alpha} * [\eta_{i,j}]^{\beta}}{\sum_{l \in N_i^k} [T_{i,l}]^{\alpha} * [\eta_{i,l}]^{\beta}} \tag{1}$$

Los parámetros α y β establecen la importancia relativa de los rastros de feromona y de la información heurística respectivamente. Los N_i^k corresponden a los vértices que puede visitar la hormiga.

Además, $\eta_{i,j}$ corresponde al inverso de la distancia entre i y j un valor heurístico disponible *a priori*.

$$\eta_{i,j} = \frac{1}{d_{i,j}} \tag{2}$$

La información de feromona se deposita una vez que todas las hormigas han hallado alguna fuente de alimento, el rastro de feromona global se evapora y se actualiza como sigue:

$$T(i,j) \leftarrow (1-p) * T_{(i,j)} + \sum_{k=1}^{m} \Delta T_{i,j}^{k} \tag{3}$$

Donde p corresponde a la tasa de evaporación. La cantidad de feromona que la hormiga k deposita en su recorrido de regreso a la colonia desde la fuente de alimentación se define como:

$$\Delta T_{i,j}^{k} = \frac{Q}{L_k} \tag{4}$$

Donde Q es una constante y L_k es la longitud del recorrido realizado por dicha hormiga.

2.6. Modelo con acondicionamiento ACO (algoritmo ACO orientado a hardware)

El algoritmo ACO introducido debe ser acondicionado para los requerimientos de la implementación hardware.

Una hormiga solo puede escoger entre tres nuevas posiciones, si la nueva posición está en la misma dirección y se denota esta distancia como 1, las otras dos posiciones estarán a una distancia $\sqrt{2}$ de la actual, como se muestra en la figura 4.

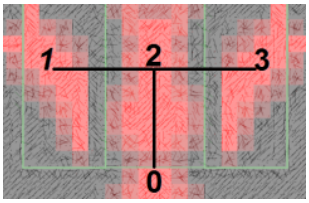

Figura 4. Distancias en los movimientos de la hormiga.

$$\eta_{i,j_1} = \frac{1}{\sqrt{2}}, \;\; \eta_{i,j_2} = 1, \;\; \eta_{i,j_3} = \frac{1}{\sqrt{2}} \tag{5}$$

Los valores típicos de α y β se encuentran entre 1 y 5, para este caso se escoge 1 para α y 4 para β. Estos son los valores que mejor se acomodan a las iteraciones experimentales, además, un valor par para β facilita la implementación en hardware. De esta manera la ecuación para escoger la nueva posición de la hormiga será.

$$P^k_{(0,1)} = \frac{\frac{T_{0,1}*1}{4}}{\frac{T_{0,1}*1}{4} + T_{0,2} + \frac{T_{0,3}*1}{4}} \tag{6}$$

$$P^k_{(0,2)} = \frac{T_{0,2}}{\frac{T_{0,1}*1}{4} + T_{0,2} + \frac{T_{0,3}*1}{4}} \tag{7}$$

$$P^k_{(0,3)} = \frac{\frac{T_{0,3}*1}{4}}{\frac{T_{0,1}*1}{4} + T_{0,2} + \frac{T_{0,3}*1}{4}} \tag{8}$$

Para poder almacenar estas probabilidades en registros se multiplica por cien al denominador de cada expresión, antes de realizar la división. Se almacena el valor entero. $P^k_{(0,1)}$ representa la probabilidad de que la hormiga k genere su movimiento en la dirección anterior en la secuencia a la que lleva actualmente, $P^k_{(0,2)}$ en la misma dirección y $P^k_{(0,3)}$ en la dirección siguiente en la secuencia. La información de feromona se recolecta sumando pixel a pixel la intensidad de los rastros contenidos en cada una de las zonas delimitadas por i y j.

Para depositar y deteriorar el rastro de feromona se debe tener en cuenta que esta operación se realiza pixel a pixel, no se evalúa la expresión del nodo i al nodo j sino puntualmente en cada coordenada (x, y) que se vea afectada en el proceso. En el caso puntual de depositar, las coordenadas (x, y) que se ven afectadas se encuentran entre los nodos i y j, para el caso de deteriorar se afecta a la matriz en general. Como se detalla en la sección 2.4 un rastro de feromona puede tener ocho niveles diferentes; las ecuaciones que rigen el cambio de estos niveles al depositar y deteriorar respectivamente, se enuncian a continuación.

$$N_{T_{x,y_1}} = N_{T_{x,y_0}} + \sum_{k=1}^{m} \Delta N^k_{T_{x,y_0}} \tag{9}$$

$$N_{T_{x,y_1}} = \begin{Bmatrix} N_{T_{x,y_0}} - 1, & N_{T_{x,y_0}} \neq 0 \\ 0, & N_{T_{x,y_0}} = 0 \end{Bmatrix}, \;\; N_{T_{x,y}} \in Z, 0 \leq N_{T_{x,y}} \leq 8 \tag{10}$$

$N_{T_{x,y}}$ representa el nivel de feromona contenido en el pixel con coordenada x, y. La hormiga k deposita inicialmente un rastro de feromona en el cual $N_{T_{x,y}} = 2$. Se considera un pixel

saturado si el nivel de intensidad en este llega a ocho, puesto que a partir de este nivel no se detecta el crecimiento en la información de feromona. En el ajuste que se realiza al algoritmo para su implementación hardware se debe sacrificar el crecimiento lineal o el decrecimiento exponencial de feromona por la limitación de recursos. Las iteraciones, demuestran que el decrecimiento lineal de feromona permite encontrar rutas óptimas, además de requerir un tamaño en los registros considerablemente inferior al de la segunda opción. Por tal razón, se considerará para efectos prácticos que la feromona crece y decrece linealmente. Además, para controlar la velocidad de decrecimiento de la feromona, se deteriora el rastro de feromona cada tres movimientos de las hormigas de la colonia.

3. Desarrollo en hardware

En esta sección se especifican las etapas mediante las cuales se implementa la ASM sobre el hardware. En la implementación se debe crear un entorno gráfico por medio del cual se evidencie el proceso de recolección de alimento. El diseño requiere un procesador gráfico y algunos circuitos auxiliares.

3.1. Descripción general

En la figura 5 se muestra la arquitectura del circuito que simula la colonia de hormigas, este permite la interacción entre el procesador de hormigas, el procesador gráfico y periféricos de entrada y salida. El circuito cuenta con tres memorias RAM; la RAM en bloque empotrada permite fraccionar el total de su capacidad en un considerable número de memorias de menor tamaño. Usar tres memorias facilita el flujo de información de una a otra (en menos ciclos de reloj que si fuera solo una memoria) y manejar tamaños diferentes para almacenar dicha información (según se requiera).

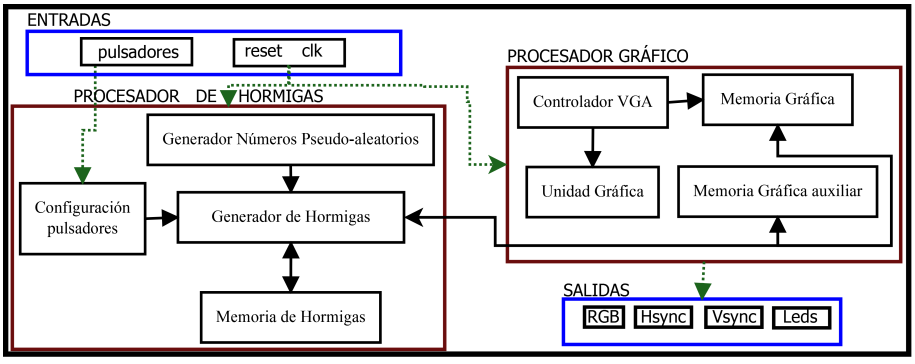

Figura 5. Diagrama de bloques de la especificación VHDL.

El procesador necesita un reloj (clk) externo ya que ha sido modelado con máquinas de estados, se agregan pulsadores para iniciar y resetear el circuito, el propósito general es controlar un display VGA. La señal RGB representa los colores mientras que Hsync y Vsync permiten la sincronización con el display VGA, los LED permiten supervisar la cantidad de alimento disponible en el entorno para ser recolectada por las hormigas.

3.2. Procesador de hormigas

El procesador de hormigas tiene como propósito escribir la memoria RAM gráfica utilizando dos memorias RAM auxiliares, valiéndose de un generador de números pseudo-aleatorios para tomar ciertas decisiones. La memoria gráfica auxiliar hace pequeños *backups* de la memoria gráfica, según se requiera, esta memoria cuenta con 128 posiciones con una longitud por posición de 4 bits, RAM 128X4. La memoria de hormigas permite guardar información característica de cada hormiga en 32 posiciones de una longitud por posición de 39 $bits$, RAM 32X39. Se abastece de alimento el entorno cuando la cantidad disponible de este es cero, se agregan componentes que permiten visualizar la colonia de hormigas y sus interacciones. El controlador de acciones decide el orden de operación del controlador de hormigas, selector de posición, generador de gráfica de la hormiga, controlador de alimento y controlador de feromona. El temporizador permite definir y ajustar el tiempo de duración de un movimiento de las hormigas mediante un bucle de espera. Si no se ajusta esta velocidad, el proceso sería imperceptible visualmente, todo el proceso es sincrónico.

La figura 6 muestra el diagrama de bloques del generador de hormigas.

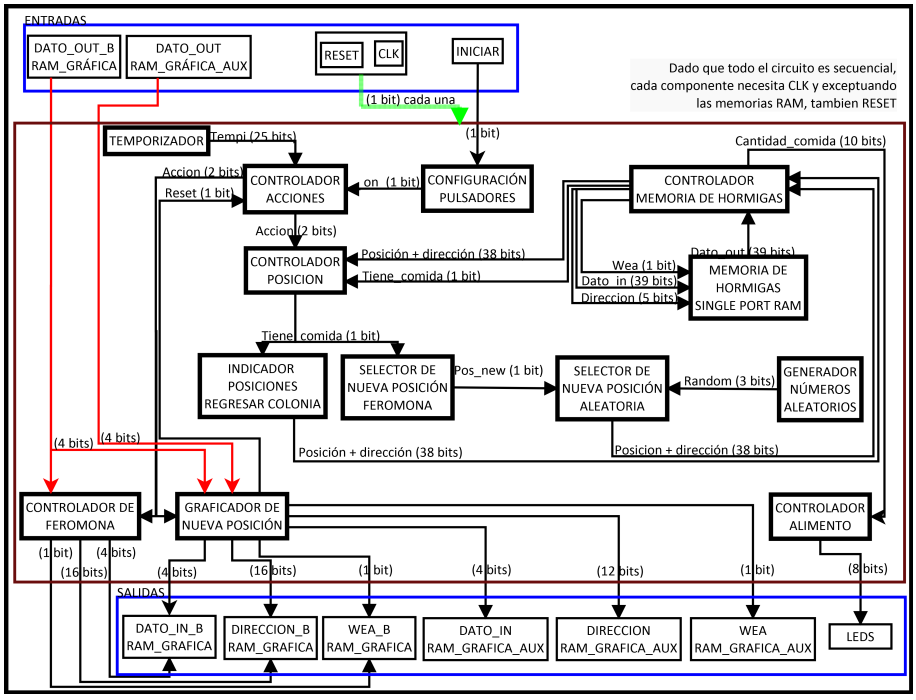

Figura 6. Diagrama de bloques del procesador de hormigas.

3.3. Circuitos auxiliares

Se diseña un procesador gráfico para controlar la pantalla VGA, se utiliza la técnica de visualización dinámica. El GPU está formado por tres componentes básicos: el controlador VGA, la unidad gráfica y la memoria RAM de doble puerto [12]. Permite visualizar dieciséis colores en pantalla. Adicionalmente se diseña un generador de números pseudoaleatorios, que

a diferencia de los aleatorios se repiten cada determinado periodo, este generador produce una secuencia de 256 números, el generador de hormigas los toma en orden diferente y aleatorio.

4. Resultados

4.1. Circuitales

Para validar la herramienta propuesta se configura la FPGA por medio del lenguaje de alto nivel definido por el IEEE, VHDL (Very-High-Speed Integrated Circuits Hardware Description Language). Se formula un experimento en el cual interactúan 32 hormigas en la búsqueda de fuentes de alimento con posición variable; implementar el circuito propuesto demanda los recursos presentados en la tabla I.

Tabla I. Recursos demandados por la implementación de la colonia de hormigas sobre la FPGA

Ítem	Usado	Disponible	Utilizado		
			GPU	P. Hormigas	Total
Slices	997	5888	1 %	15 %	16 %
Slice flip flops	486	11776	1 %	3 %	4 %
4 Input LUTs	1899	11776	2 %	14 %	16 %
IOBs (Input/Output block)	25	372	4 %	2 %	6 %
BRAMs	18 (82Kb)	20 (92Kb)	85 %	5 %	90 %
GCLKs	1	24	3 %	1 %	4 %

Como se puede observar en la tabla I el uso de recursos de la FPGA es mínimo, exceptuando la memoria RAM a causa del procesador gráfico (GPU), quien utiliza casi por completo este recurso. A pesar de la complejidad del circuito diseñado se logra implementar usando unidades básicas como Slices, Flip Flops, Luts, IOBs y memoria RAM; este hecho hace aún más evidente la gran ventaja que conlleva usar circuitos lógicos programables para diseñar procesadores. La implementación sobre FPGA tiene serias limitaciones de tamaño en memoria RAM, para solucionarlo se debe pensar en utilizar memorias externas incluidas en módulos de desarrollo o acopladas de forma manual cuando se requiera modelar colonias de mayor tamaño (escalar el sistema).

Los datos descritos en la tabla II son válidos bajo experimentos sobre una colonia conformada por 32 miembros, que realiza cinco movimientos por segundo. Considerando que el circuito realiza un movimiento de las hormigas y actualización del rastro de feromona en aproximadamente 1.7 mili segundos, el procesador de hormigas está inactivo el 99.15 % del tiempo de implementación para los experimentos propuestos.

Tabla II. Resultados de la implementación de la colonia de hormigas

Duración de Ítem	Mínimo [ms]	Máximo [ms]	Máximo escalamiento de Ítem	Valor
Un movimiento hormiga	0.0301	0.0332	Movimientos de la colonia por segundo (32 hormigas)	591
Actualización feromona global	5.2763	5.2763		
Promedio un movimiento colonia	1.6912	1.6974	Cantidad de hormigas (cinco movimientos por segundo)	6004
Procesador de hormigas activo	8.456	8.4870		

El tiempo de síntesis, implementación y generación del archivo de programación para la FPGA no supera los dos minutos, procesos que se llevan a cabo en el entorno de desarrollo ISE® Web Pack. La configuración del dispositivo de destino se realiza en la herramienta iMPACT® de Xilinx©.

En la tabla III se presenta una comparación entre las principales características de la herramienta Netlogo y el procesador de hormigas. Una distinción inmediata de estos datos, es que el procesador es capaz de realizar modificaciones referentes a la posición y cantidad de alimento en el entorno.

Tabla III. Comparación herramienta Netlogo y procesador de hormigas

Ítem	Netlogo	Procesador hormigas
Máxima cantidad de hormigas	200	6004
Máxima cantidad movimientos colonia por segundo	321	591
Cambios de dirección	22.5°	45°
Modificar radio de difusión de feromona	Sí	Sí
Modificar frecuencia de evaporación de feromona	Sí	Sí
Modificar cantidad de comida en el entorno	No	Sí
Modificar posición de comida en el entorno	No	Sí
Habilitar auto-inserción de comida en el entorno	No	Sí
Supervisar cantidad de comida en el entorno	Sí	Sí
Supervisar variables características de las hormigas (posición, dirección, estado)	No	Sí
Variar parámetros del algoritmo de hormigas	Sí	Sí

Los valores asociados a la variación en la cantidad de hormigas y los movimientos por segundo de estas son inversamente proporcionales, es decir, si uno de los dos se configura en su máximo valor el otro se situará en el valor inicial de escalamiento; para el experimento propuesto, 32 hormigas y 5 movimientos por segundo, dicha dependencia se puede aproximar a la siguiente ecuación:

$$y_{max}\,[k] = \frac{1}{33{,}2 * 10^{-6} * k + 659{,}59 * 10^{-6}}\ , \tag{11}$$

Donde k representa el número de hormigas en la colonia y y el máximo número de movimientos que las k hormigas pueden realizar en un segundo.

4.2. Cómputo emergente: una aproximación hacia la complejidad

Funcionalmente el procesador diseñado replica correctamente las características principales del proceso de recolección de alimento en una colonia de hormigas en una interfaz gráfica de resolución 256X256 [19]. En este sentido el procesador logra recrear la dinámica emergente de un sistema complejo sobre la base de una arquitectura computacional relativamente sencilla dados los costos de la implementación FPGA, podría entonces pensarse en escalar este sistema a la implementación de colonias más grandes. El crecimiento de recursos del procesador en la medida que la colonia se hace más grande solo afectaría el tamaño de las memorias RAM, especialmente la gráfica, tal crecimiento en la cantidad de recursos demandados obedecería una relación exponencial decreciente y para colonias más grandes casi que no sería notoria, así que la apreciación sobre la emergencia de comportamiento y simplicidad arquitectural no se ve limitada al caso particular aquí tratado.

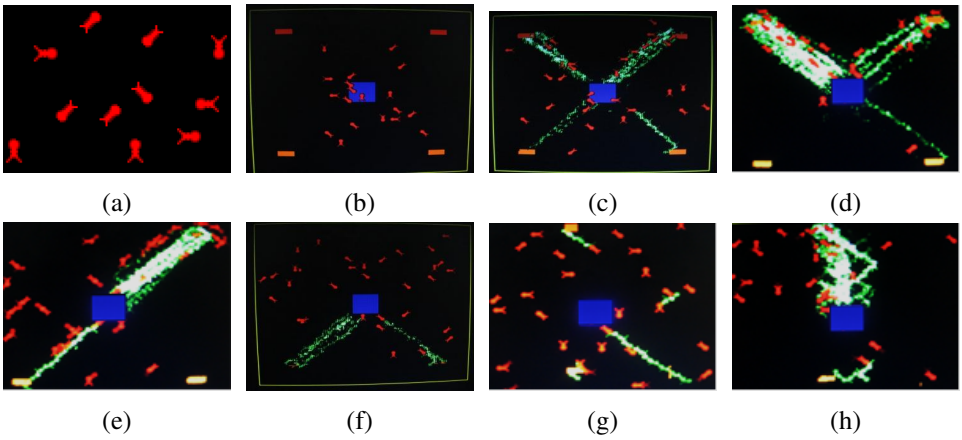

Figura 7. Resultados de la implementación hardware para simular una colonia de hormigas. a) Diseño gráfico de las hormigas. b) Hormigas abandonando la colonia en busca de alimento. c) Hormigas construyendo caminos de feromona. d) Rastro de feromona saturado. e) Desaparición del primer rastro de feromona saturado. f) Alimento recolectado por completo. g) Inserción de nuevo alimento en el entorno. h) Creación de nuevos caminos de feromona.

El procesador modela la sucesión de acciones para cada hormiga y el entorno en el cual estas se desarrollan. Como se analizaba en la sección 3, se espera que la interacción de estas acciones a través de la información de feromona, den lugar al proceso de recolección de alimento; el proceso demuestra auto-organización, como se evidencia en la figura 7, en un principio las hormigas vagan desorientadas y la primera fuente de alimento que hallan se desabastece exponencialmente, así con las otras tres, una a una, al acabarse la comida e introducirse una nueva fuente de esta, se repite el proceso de desabastecimiento exponencial. La aproximación al decrecimiento y actualización de la información de feromona no afecta la creación de caminos óptimos entre la colonia y las fuentes de alimento. La selección heurística de las

nuevas posiciones permite a las hormigas localizar rápidamente las fuentes de alimento encontradas por compañeras, en la implementación es posible variar el número de hormigas en la colonia desde 1 hasta 32. La cantidad de alimento recolectado en un tiempo determinado crece exponencialmente conforme a la cantidad de hormigas que estén involucradas en el proceso, tales fenómenos son totalmente emergentes y es comprobable experimentalmente con colonias reales [19].

5. Conclusiones y expectativas futuras

Se ha presentado un procesador de propósito específico que permite desarrollar experimentos sobre el proceso de recolección de alimento en una colonia de hormigas evidenciable gráficamente por medio de una pantalla VGA. El acondicionamiento aplicado al algoritmo Ant-Cycle System para su implementación sobre FPGA permitió describir correctamente el comportamiento de una colonia de hormigas, comprobando así el uso de la feromona para encontrar el camino más corto hacia los alimentos y la rapidez con la que se abastece la colonia luego de que una hormiga encuentra alimento por primera vez [19].

Se demuestra entonces que un sistema estructuralmente sencillo, dado los recursos necesarios para su implementación sobre FPGA (tabla I), es capaz de simular la complejidad del proceso de recolección de alimento de una colonia de hormigas. El hardware configurable facilita el diseño e implementación puesto que permite definir el tamaño de los registros y operadores.

Como se muestra en la tabla III, el procesador diseñado supone una mejora en el campo de simulaciones de colonias de hormigas con respecto a la herramienta de Netlogo [10], en la medida en que permite variar una cantidad superior de parámetros en la formulación de experimentos y supervisar una cantidad superior de variables en su implementación.

El desarrollo del trabajo utiliza dos procesadores en paralelo requiriendo así el uso de una memoria RAM de doble puerto. Para pensar en un escalamiento bajo el mismo kit de desarrollo una opción sería operar estos dos procesadores en serie. Un módulo adicional se encargaría de dar acceso a memoria al uno u otro y de esta manera poder utilizar una memoria RAM de puerto único, el kit de desarrollo utilizado incluye una memoria DDR2 SDRAM de 512Mbit (32M x 16) la cual sería usada para reemplazar la memoria distribuida interna de la FPGA. Lo anterioro solucionaría la limitación en memoria RAM de la implementación realizada permitiendo escalar de una matriz gráfica 256x256 a 4096x4096 (un campo de experimentos 256 veces más grande que el actual) y de almacenar dieciséis datos posibles a 65536.

Un estudio interesante que podría desprenderse, sería realizar el proceso propuesto agregando un depredador en el entorno, puesto que las hormigas deberían encontrar una ruta a la comida evitando a dicho depredador. Esta variación en el entorno aumentaría la complejidad del proceso de recolección de alimento.

Otra idea podría orientarse a la competencia entre hormigas, más exactamente entre diversas especies (el presente artículo solo se trató el tema de cooperación). Existen especies de hormigas esclavistas, que en competencia con otras especies buscan no solo eliminar sino, si es posible, subyugar a otras para su colonia. En la lucha por abastecerse de alimento, cada es-

pecie desarrolla estrategias para sobrevivir, fenómeno que permite un análisis de las colonias en su rol de competencia.

Una vía de investigación alternativa puede referirse a la comparación entre algoritmos de colonias de hormigas y algoritmos genéticos. Si se trata a la colonia de hormigas como un problema de optimización combinacional sería posible un eventual modelamiento bajo un algoritmo genético, de esta manera se avanzaría en la descripción del comportamiento de las hormigas. La aplicación es viable en la medida en que los algoritmos de hormigas aún tienen muchos vacíos en la descripción del comportamiento global. Por ejemplo, para este tipo de algoritmos aún no existe una directriz clara de cómo las hormigas vuelven a la colonia. Esta iniciativa tendría como soporte las investigaciones realizadas en [20].

Referencias

[1] Patricia J. Folgarait y Alejandro G. Farji-brener, Un mundo de hormigas. 1ª edición, Buenos Aires, Argentina, 2005, pp. 15-18.

[2] M. Dorigo, Marco Birattari and Thomas Stutzle, University Libre Bruxelles, Belgium, "Ant Colony Optimization, Artificial Ants as a Computational Intelligence Technique", IEEE Computational Intelligence Magazine, Vol. 1, No.4, November, 2006.

[3] S. Das, S. Vaze, G.Singh and E., Buehler, University Libre Bruxelles, Belgium, "Ant Colony Optimization, Artificial Ants as a Computational Intelligence Technique", IEEE Computational Intelligence Magazine, Vol. 1, No.4, November, 2006, pp. 28-39.

[4] M. Dorigo. Optimization, Learning and Natural Algorithms. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, 1992.

[5] Xiang-yang Deng, Wen-long Yu and Li-min Zhang, A new ant colony optimization with global exploring capability and rapid convergence. Intelligent Control and Automation (WCICA), 2012, 10th World Congress on IEEE, July, 2012, pp. 2055-2057.

[6] H.Y. Markid, B.Z. Dadaneh and M.E. Moghaddam, Bidirectional ant colony optimization for feature selection. Artificial Intelligence and Signal Processing (AISP), 2015 International Symposium on. IEEE, March, 2015, pp. 53-58.

[7] Li, Shih-An, Yang, Min-Hao, Weng, Chung-Wei, Chen Yi-Hong, Lo Chia-Hung and Wong Ching-Chang, Ant colony optimization algorithm design and its FPGA implementation. Intelligent Signal Processing and Communications Systems (ISPACS), 2012 International Symposium on. IEEE, November 2012, pp. 262-265.

[8] Huang, Hsu-Chih and A Taguchi-Based, Heterogeneous Parallel Metaheuristic ACO-PSO and Its FPGA Realization to Optimal Polar-Space Locomotion Control of Four-Wheeled Redundant Mobile Robots. Industrial Informatics, IEEE Transactions on. 2015, pp. 1-8.

[9] Clive Max Maxfield, FPGAs: Instant Access, Elsevier, 2008, 1-12.

[10] U. Wilensky, NetLogo, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1999, disponible en: http://ccl.northwestern.edu/netlogo/

[11] Chia-Feng Juang, Chi-Wei Hung and Chia-Hung Hsu, Rule-Based Cooperative Continuous Ant Colony Optimization to Improve the Accuracy of Fuzzy System Design. Fuzzy Systems, IEEE Transactions on. IEEE,, Volumen, 22, Edición 4, August, 2014, pp. 723-735.

[12] O. Wongwirat and A. Anuntachai, Searching energy-efficient route for mobile robot with ant algorithm. Control, Automation and Systems (ICCAS), 2011, 11th International Conference on. IEEE, August, 2011, pp. 1071-1075.

[13] Du Xiang-run, Zhang Jian-long and Feng Min-quan, Water supply route optimization for reservoir emergency based on improved ant colony algorithm, Control and Decision Conference (2014 CCDC), The 26th Chinese. IEEE. June, 2014, pp. 1354-1359.

[14] D.O. Sales, M. A. Dias and F. S. Osorio, Grid Ant Colony Optimization Applied to a Multi-robotic Garbage Collection System. Robotics: SBR-LARS Robotics Symposium and Robocontrol (SBR LARS Robocontrol), Joint Conference on. IEEE, October, 2014, pp. 187 - 192.

[15] Xing Wei, Zhiyuan Li, Jingjing Qu. Research on parameters optimization and simulation of the Ant Colony Algorithm. Cyberspace Technology (CCT 2014), International Conference on. IEEE. 2014. Páginas: 1 – 4.

[16] S. Brown, Fundamentos de lógica digital con diseño VHDL, 2ª edición, McGraw Gil, Ciudad de México, México, 2006, Cap 10, sec 10.2.

[17] Jinbiao Wang and Kaichi Wang, Union-Intersection Ant System. Theories and Applications (BIC-TA), 2010 IEEE, Fifth International Conference on. IEEE, September, 2010, pp. 364-371.

[18] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26(1):29–41, 1996.

[19] J.-L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. The self-organizing exploratory pattern of the argentine ant, Journal of Insect Behavior, 3(2), 1990, pp. 159-168.

[20] Sergio Rojas Galeano, Optimización de rutas mediante computación bioinspirada, un paralelo entre hormigas artificiales y algoritmos genéticos, Revista de Ingeniería Universidad Distrital Francisco José de Caldas, Vol. 7, Núm. 14, 2004, pp. 97-104.

[21] Funcionamiento de la colonia de hormigas, S.f., disponible en: https://www.dropbox.com/s/tt7dk57yo47gfks/Evidencia_colonia_hormigas.AVI?dl=0

[22] Archivos de descripción VHD para cada componente circuital del proyecto, S.f., disponible en: https://www.dropbox.com/s/jf9x3npcc9gt3vv/proyecto_colonia_hormigas.rar?dl=0

Cristian David Rodríguez Rodríguez

Estudiante Ingeniería Electrónica de la Universidad Distrital Francisco José de Caldas, Bogotá, D.C., Colombia; miembro del Laboratorio de Automática e Inteligencia computacional LAMIC.
e-mail: cridrodriguezr@correo.udistrital.edu.co

Miguel Alberto Melgarejo Rey

Ingeniero Electrónico, Universidad Distrital Francisco José de Caldas: magíster en Ingeniería Electrónica y Computadores, Universidad de los Andes: doctorando en Ingeniería, Pontificia Universidad Javeriana; profesor asociado, Facultad de ingeniería Universidad Distrital Francisco José de Caldas; miembro del Laboratorio de Automática e Inteligencia Computacional (LAMIC).
e-mail: mmelgarejo@udistrital.edu.co