Diseño de un circuito sumador completo usando evolución intrínseca de hardware

José M. Luna[1]
Christian J. Devia[2]
Álvaro Betancourt U.[3]

RESUMEN

Este artículo describe la utilización de estrategias bioinspiradas en el diseño de un circuito digital, que implementa un sumador completo con acarreo de entrada en un dispositivo lógico programable. Además, se presenta el desarrollo de una plataforma experimental en hardware evolutivo utilizada en la ejecución del experimento.

Palabras Clave: Algoritmos Genéticos, FPGA, Hardware Evolutivo, Sumador Completo.

ABSTRACT

This paper describes the use of bioinspired strategies in the design of a digital circuit, which implements a full adder with carry input on a programmable logic device, as well as presenting the development of an experimental evolvable hardware platform used in the experiment execution.

Key Words: Evolvable Hardware, Full Adder, FPGA, Genetic Algorithms.

I. INTRODUCCIÓN

EL hardware evolutivo (Evolvable Hardware, EHW) consiste en la aplicación de las estrategias evolutivas de la naturaleza en la resolución de problemas de diseño electrónico de alta complejidad, aprovechando las ventajas de autoorganización y adaptabilidad de los sistemas evolutivos computacionales [1].

Esta aproximación de diseño es mejor expresada como una vista en caja negra del circuito, donde lo realmente importante es la tarea que realiza y nó la manera de llevarla cabo [2].

El EHW se divide en *extrínseco* e *intrínseco* [3]. En el primer caso, el proceso evolutivo se ejecuta sobre un modelo simulado de una matriz de celdas evolutiva. Es decir, que un algoritmo evolutivo es ejecutado y la evaluación de aptitud de los individuos que conforman la población se lleva a cabo en simulación, lo cual presenta ventajas en cuanto a velocidad de procesamiento de los algoritmos, generalidad y estabilidad de los circuitos obtenidos. Una vez converge la evolución, el mejor individuo es descargado en el dispositivo lógico programable implementando el circuito requerido.

La evolución intrínseca se lleva a cabo directamente sobre el dispositivo, es decir, que durante la ejecución de un algoritmo evolutivo, cada uno de los individuos de la población es descargado en el dispositivo configurable para la evaluación de su aptitud. Este tipo de evolución presenta ventajas en cuanto a variables de diseño que son muy difíciles de manejar, como cambios de temperatura o radiaciones externas que afectan al circuito físicamente produciendo algunos niveles de entropía difíciles de superar por las técnicas de modelado tradicional. En ese sentido, los circuitos resultan muy robustos pero poco generales, ya que dichos circuitos se ajustan muy bien al dispositivo sobre el que se evolucionan, pero cabe la posibilidad de que su implementación sobre otro dispositivo diferente no se desempeñe de la misma manera.

En este artículo producto de un trabajo de investigación y desarrollo, se evalúa el potencial del EHW en el diseño de circuitos digitales combinacionales mediante la experimentación con un circuito sumador completo de un bit con acarreo de entrada. Con el fin de describir la metodología de experimentación, así como los ensayos realizados, el artículo se organiza de la siguiente manera: En la sección II se hace una descripción general de la plataforma experimental desarrollada y se exponen las principales características de los módulos de hardware y software implementados. En la sección III se presenta el experimento de evolución del circuito sumador completo y se realizan comparaciones cualitativas con experimentos similares registrados en la literatura. Se exponen los resultados obtenidos y se ilustran estadísticas de los procesos evolutivos llevados a cabo. En la sección IV se presentan las conclusiones.

II. PLATAFORMA EXPERIMENTAL

La plataforma experimental DEEP (del inglés *Development of an Experimental Evolvable Hardware Platform*) utilizada, se especializa en evolución intrínseca de hardware utilizando algoritmos genéticos (AG), y se compone de dos módulos que operan de manera conjunta [4]. Dichos módulos comprenden a) el desarrollo de una biblioteca de instrumentos virtuales (VI, del inglés *Virtual Instruments*) para AG denominada VIGA (del inglés *Virtual Instruments for Genetic Algorithms*) y b) el diseño de una unidad estructural mínima configurable denominada celda evolutiva.

A continuación, se describen los módulos enunciados.

[1] Miembro Grupo de Investigación LAMIC, Universidad Distrital Francisco José de Caldas.
[2] Miembro Grupo de Investigación LAMIC, Universidad Distrital Francisco José de Caldas.
[3] Director Grupo de Investigación LAMIC, Universidad Distrital Francisco José de Caldas.

Artículo recibido en Septiembre de 2004
Aceptado en Noviembre de 2004

Si se compara este funcionamiento con el de la celda evolutiva propuesta en el presente artículo, se tiene que en este caso, no solo se involucra la conducta de una compuerta en cada celda, sino de múltiples compuertas funcionando de manera conjunta en el mismo espacio.

A. Biblioteca de Instrumentos Virtuales para Algoritmos Genéticos

Cómo se mencionó anteriormente, para la ejecución de los experimentos en evolución intrínseca de hardware, se implementó una biblioteca de operadores genéticos y evolutivos orientada al diseño y ejecución de AG denominada VIGA. Además, se consideró conveniente monitorear el desempeño de los AG de manera concomitante al proceso evolutivo, por lo que se escogió la herramienta de instrumentación virtual Labview®, que ofrece una interfaz gráfica de fácil operación para el diseñador, y además permite realizar el seguimiento del desempeño de los algoritmos a través de gráficas de formas de onda e indicadores digitales, que se actualizan en pantalla mientras el ciclo evolutivo es llevado a cabo.

Dicha biblioteca consta de un grupo de 31 VI que facilitan el proceso de diseño de un AG y la comunicación por puerto paralelo tipo D con un dispositivo lógico programable, que en este caso es una FPGA XC4000.

La herramienta no solo está limitada a resolver experimentos de evolución de hardware, sino que permite solucionar una variedad de problemas clásicos de software, entre los cuales se pueden mencionar problemas de optimización combinatorios (*combinatorial optimization problems*) como el problema del agente viajero (TSP del inglés *Traveling Salesman Problem*), problemas de estructuras de árbol como el MST (del inglés *Minimun Spanning Tree Problem*), y el problema de la Mochila o del Furgón (*Knapsack Problem*) entre otros [1], [5]-[7]. Asimismo, permite resolver problemas de optimización de disponibilidad de dispositivos (*Reliability Optimization Problem*) y problemas de secuencias de máquinas (*Flow-Shop Sequencing Problem*).

Fue necesario realizar una revisión de los operadores genéticos y evolutivos más utilizados y reportados en la literatura. De estos operadores se seleccionaron los que se consideraron más relevantes y se modelaron los algoritmos de tal manera que se aprovecharan las facilidades gráficas y operativas de Labview® en la comunicación con la FPGA durante el proceso evolutivo.

B. Celda Evolutiva

Como antes se había mencionado, es la unidad estructural mínima configurable, es decir, el mínimo bloque funcional en la construcción de un circuito electrónico digital durante el proceso evolutivo. La celda propuesta está conformada por tres unidades lógicas, un flip-flop tipo D como elemento de almacenamiento, y un conjunto de multiplexores que determinan la conectividad del circuito obtenido en dicha celda, como se ilustra en Fig. 1 [8].

Estas celdas interconectadas, forman un arreglo normalmente de tipo rectangular como el que se ilustra en la Fig. 2, que se conoce como *Matriz de Celdas Evolutiva* [4] - [9]. De manera previa a la ejecución de cada experimento las conexiones entre cada celda son configuradas por el usuario, de tal forma, que a diferencia de los experimentos planteados por Kalganova y Miller [10] la geometría del circuito permanece aparentemente constante. A esta invariancia en la disposición geométrica se le conoce como *disposición homogénea del circuito*.

Se decidió no variar la geometría del circuito dada la complejidad de la celda evolutiva diseñada, pues visualizando las marcadas diferencias entre la celda propuesta por Miller [11] en algunos experimentos ejecutados con evolución extrínseca, y la nuestra, se tiene que en dichos experimentos, cada celda solamente ejecuta una operación lógica combinacional a la vez, como una compuerta AND, OR, NOT, XOR, o un multiplexor de dos entradas.

Si se compara este funcionamiento con el de la celda evolutiva propuesta en el presente artículo, se tiene que en este caso, no solo se involucra la conducta de una compuerta en cada celda, sino de múltiples compuertas funcionando de manera conjunta en el mismo espacio, es decir, el equivalente a tener más de una celda de las propuestas por Miller interconectadas en la matriz. Además, la implementación de multiplexores se lleva a cabo de manera implícita con cada uno de los multiplexores contenidos en el modelo de celda propuesto en la Fig. 1.

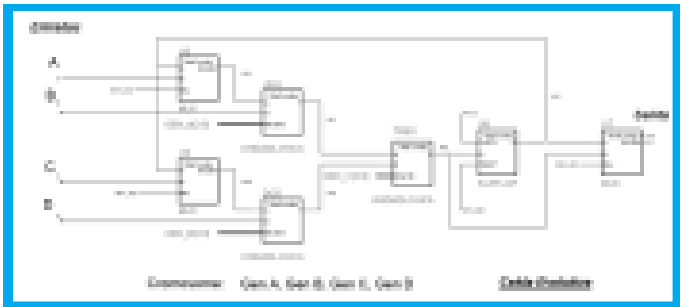


Figura 1. Celda Evolutiva implementada en la evolución del circuito sumador completo de un bit con acarreo de entrada. Nótese el elemento de almacenamiento (flip-flop) en la salida.

Esto resulta en un comportamiento semejante a la *disposición heterogénea del circuito* (que en oposición a la disposición homogénea, implica la variación ocasional de la geometría del mismo en la evolución). No obstante, es necesario aclarar que dicha variación no se lleva a cabo, ni se considera en el cromosoma de control descargado a la FPGA durante el proceso evolutivo, por lo cual, es errado afirmar que en este ensayo la geometría cambie, puesto que esto tiene otras implicaciones de fondo tales como cromosomas de longitud variable, cambios fortuitos en la distribución de filas y columnas de la matriz, y la modificación en el número de recursos utilizados en el circuito, que como puede notarse, no se toman en consideración en el presente experimento.

Es habitual encontrar que los experimentos de evolución intrínseca de hardware sean ser de mayor duración que los de evolución extrínseca. Esto se explica, porque los procesos de EHW intrínseco en su mayoría se suelen ejecutar sobre FPGA que admiten configuración parcial como la XC6200 o la Virtex de Xilinx. Como suele trabajarse con las celdas lógicas que dichos dispositivos tienen de fábrica, y estas celdas son de muy bajo nivel, la implementación de una compuerta con estos recursos compromete un tiempo relativamente extenso en cada ciclo del AG.

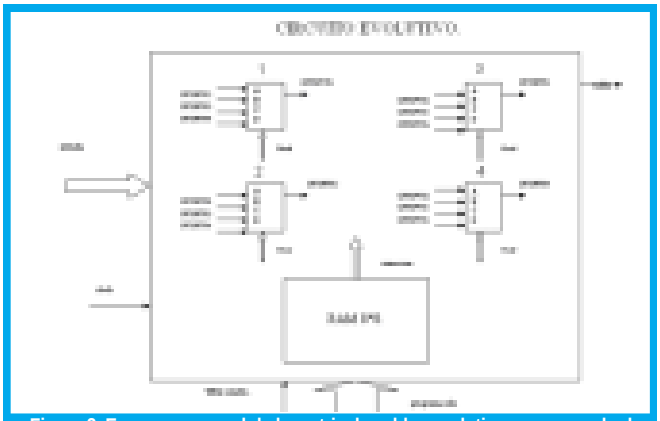


Figura 2. Esquema general de la matriz de celdas evolutiva en un arreglo de 2x2. Nótese la memoria RAM en la cual se almacenan los individuos de control.

Miller [11] escribe refiriéndose a las FPGA XC6200 y Virtex: "...Estos dispositivos son conocidos como FPGA granulares finas, lo cual, significa que cada uno está compuesto por un arreglo grande (típicamente cuadrado) de celdas lógicas muy simples. Esto, en contraposición con las mejores conocidas Xilinx XC3000 y XC4000, las cuales, son mucho más grandes, y en consecuencia, poseen celdas más complejas (que son conocidas como bloques lógicos configurables o CLB)...". De donde se concluye que los experimentos de este tipo se prefieren sobre FPGA granulares finas ya que las FPGA XC3000 y XC4000 son muy densas en componentes de celdas y además no permiten configuración parcial.

La propuesta de hacer un mapeo de nuestra propia celda lógica sobre la FPGA XC4000 contrarresta el efecto de tener que utilizar CLB de fábrica, y permite diseñar una FPGA con celdas apropiadas para EHW intrínseco, reduciendo los tiempos de configuración de estos dispositivos.

Una de las razones por la que se evitó al máximo la variación en la geometría del circuito es su incidencia sobre la duración de cada generación en el AG, puesto que se agrega un operador genético más en cada iteración. Con esto, se hace innecesaria la implementación de algunos recursos evolutivos propuestos en los experimentos de Kalganova [10], como el operador de *Mutación de Geometría* que existe además del operador de *Mutación del Circuito* [10], y que consiste en el cambio fortuito del número de filas o de columnas en la matriz, es decir, que se ejecutan dos tipos de mutaciones independientes con un incremento en los lapsos entre generaciones. Además, se evitan también algunos algoritmos de reparación de conexiones inválidas que regularmente se obtienen a través de la supresión o agregación de filas o columnas, que imprimen más demora a la convergencia del proceso.

Otro factor de suma importancia, es el sesgo que normalmente se le da a los procesos evolutivos, es decir, que teniendo un conocimiento previo del problema, es posible ayudar al AG en el proceso de optimización involucrando algo de dicho conocimiento en la solución. Es así como Gordon, Kalganova y Miller, [2], [10], [11], aseguran de que la solución sea de tipo combinacional evitando realimentaciones entre las salidas y las entradas de celdas ya interconectadas. Es decir, que la matriz solo permite topologías de circuitos de tipo unidireccional en este caso, por lo cual, es necesario un algoritmo de reparación después de la ejecución de un operador de cruce, que adiciona tiempo al proceso de convergencia.

Esta característica unidireccional en los circuitos evolucionados es ventajosa, puesto que se evitan posibles inconvenientes a partir de las realimentaciones, que en ocasiones forzan a las salidas de los dispositivos de la celda a entregar una corriente grande ocasionando un posible daño físico al dispositivo lógico programable. Además se garantiza el comportamiento combinacional del sumador, partiendo del diseño convencional de un sumador completo, como el que se ilustra en la Fig. 3.

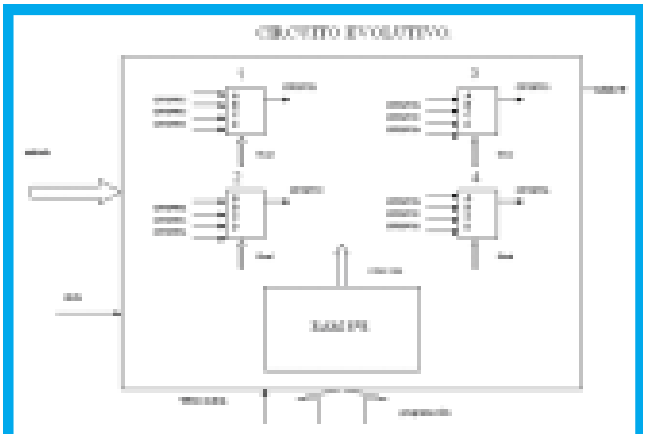


Figura 3. Circuito sumador completo convencional. Es un circuito puramente combinacional.

En la presente investigación y experimentación, se evitan las conexiones dañinas mediante la realimentación de la salida a la entrada de la celda, empleando los multiplexores mostrados en la Fig. 1, además, las conexiones fijas entre celdas contribuyen también a evitar realimentaciones indeseadas en la matriz.

Los sesgos implantados en la evolución operan bajo una filosofía diferente a las propuestas por Gordon y Kalganova [2], [10], que consiste en que las conexiones estáticas entre celdas impuestas en la etapa de acondicionamiento (ver Fig. 4), garanticen

la factibilidad del circuito sumador, con la presencia de un número límite de compuertas en la matriz. Dicho número debe satisfacer por lo menos uno de los muchos circuitos conocidos diseñados por métodos convencionales. Vale recalcar que a diferencia de Kalganova y Miller [10], [11], no se desactivaron los elementos de almacenamiento (flip-flops) en las celdas, ni se inhibieron las conexiones de realimentación, puesto que como se puede observar en el modelo de la celda (ver Fig. 1) no hay peligro de daño en el circuito, y además (aunque en los ensayos realizados no se presentó), tampoco se descartó un posible comportamiento secuencial que fuese de utilidad en el circuito a obtener.

C. Operación Conjunta de Módulos

La operación conjunta de los módulos de hardware y software descritos y justificados en los párrafos anteriores, se explica mediante lo que hemos denominado el *ciclo evolutivo* [9], el cual, se lleva a cabo durante cada experimento y se resume en el esquema ilustrado de la Fig. 4.

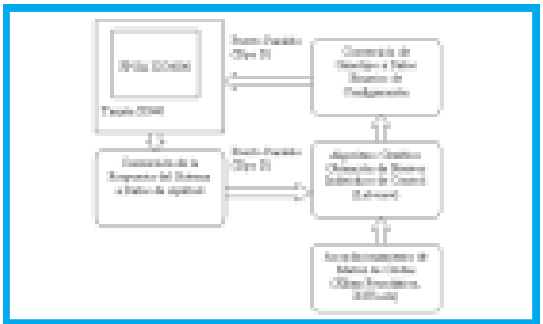


Figura 4. Esquema general del ciclo evolutivo. Ilustra el funcionamiento conjunto de los módulos software y hardware de la plataforma DEEP [9].

Antes de la ejecución de un experimento, es necesario establecer las condiciones correctas para la evolución. Dichas fases preliminares, están relacionadas con la selección de los parámetros apropiados del AG, el número de celdas evolutivas y sus respectivas conexiones.

El primer paso, es la selección del número de celdas a utilizar. Dicha cantidad se elige en proporción a la complejidad del experimento. Es posible, que un número excedente de celdas aumente el espacio solución, lo que en algunos casos resulta en muchas configuraciones que implementan el mismo circuito.

La idea básica del experimento es darle libertad a la evolución para converger a una solución combinacional o secuencial indistintamente.

Como se había mencionado antes, las conexiones entre celdas, son planeadas de manera intuitiva o partiendo de un conocimiento previo del problema por parte del experimentador, y se espera, que dicho conocimiento ayude a la obtención de la solución.

Posteriormente, se escoge la estrategia evolutiva apropiada. Al utilizar VIGA se pueden personalizar los pasos básicos de un AG, seleccionando los operadores genéticos y evolutivos necesarios. Asimismo, se configuran los parámetros del AG, como las probabilidades de cruce, de mutación, el método de selección y demás propiedades de la estrategia evolutiva seleccionada.

Una vez se finaliza con esta etapa inicial de acondicionamiento, se procede a ejecutar el AG. Durante la evaluación de aptitud de los cromosomas se debe realizar la descarga de cada individuo en la FPGA. Previo a esta descarga se hace la conversión del genotipo a palabras de control o configuración de la matriz de celdas evolutiva.

Posteriormente, se evalúa la salida de cada palabra de control en respuesta a una excitación del circuito por entradas digitales, provenientes de una fuente externa que puede ser un circuito generador de estímulos, o señales enviadas a través del puerto paralelo. Estas salidas son confrontadas con la tarea que el circuito debe realizar, y de acuerdo a la definición de la función aptitud que el diseñador proponga, se procede a ejecutar los operadores de selección, cruce y mutación inherentes al AG.

El ciclo evolutivo se repite hasta que se cumpla la condición de finalización que bien puede ser un número de generaciones determinadas, un punto de error mínimo o un máximo de una función, entre otras opciones que el diseñador considere.

III. EVOLUCIÓN DE UN SUMADOR COMPLETO

A continuación se describe el experimento de diseño de un circuito sumador completo con acarreo de entrada, utilizando EHW intrínseco. Este circuito es de tipo combinacional, es decir, que es un sistema sin memoria, en el cual, a cada combinación de datos binarios de entrada, le corresponde una salida que es independiente del estado anterior como en el caso de la Fig. 3.

La idea básica del experimento es darle libertad a la evolución para converger a una solución combinacional o secuencial indistintamente. Un circuito secuencial, es un sistema con memoria donde cada salida es producto de las combinaciones de datos a la entrada y del estado inmediatamente anterior.

La configuración de la matriz de celdas evolutiva utilizada y sus conexiones fijas se ilustran en la Fig. 5.

Obsérvese de la Fig. 5, el lugar donde se encuentran conectados los operandos de entrada A y B, el acarreo de entrada C_{in}, la suma S_{out} y el acarreo de salida C_{out}.

A. Codificación

Contrastando con los experimentos similares ejecutados por Kalganova y Miller [10], [11], la codificación implementada en este problema de optimización condicionada es de tipo binaria, la cual,

es muy común en el AG simple. Se decidió trabajar de esta forma, ya que las complicaciones que surgen de imponer restricciones de tipo combinacional se simplifican en alguna medida.

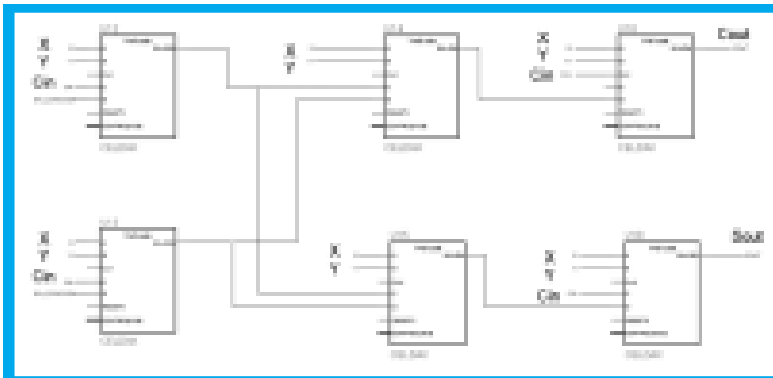


Figura 5. Matriz de celdas evolutiva utilizada en el experimento de evolución del circuito sumador completo con acarreo de entrada. Obsérvense los enlaces fijos entre celdas.

Entre las complicaciones a las que se ha hecho mención se tienen: impedir realimentaciones entre compuertas, haciendo que el circuito opere de manera unidireccional, variar la geometría del circuito de manera fortuita durante cada generación y utilizar una convención de codificación, en la cual, se emplean tripletas de números enteros, de los cuales dos números determinan las dos celdas conectadas a las entradas de la celda bajo referencia y el tercer término determina la operación lógica [10], [11].

En conclusión, cada celda evolutiva es controlada por una palabra de 16 bits, y al haberse utilizado un arreglo de 2x3 como se ilustra en la Fig. 5, la palabra de control (cromosoma) posee una longitud de 96 bits, que difiere mucho de las longitudes de cromosomas utilizadas por Gordon [2] con un valor regular de 604 genes para un arreglo de celda de 2x2. Las longitudes cortas resultan ortodoxas si se tiene en cuenta que el teorema del esquema establece que "...los esquemas entre menor longitud tengan, menor orden y mayor aptitud, incrementan el número de sus instancias de manera exponencial en el transcurso de las generaciones" [9], de donde se tiene que si la longitud de un cromosoma es corta, cabe la posibilidad de que la longitud del esquema también sea corta, lo cual, favorece el proceso de selección de esquemas útiles durante la evolución.

B. Algoritmo Genético

La Fig. 6 ilustra el AG diseñado utilizando la biblioteca VIGA. Nótese la facilidad gráfica que presta Labview® en el diseño de este tipo de algoritmos, y la simplicidad del mismo en el proceso de conexión de las variables y elementos que lo componen.

De la misma manera debe tenerse en cuenta que el manejo de puertos resulta práctico utilizando los VI *in port* y *out port* del menú de funciones de Labview®, por lo que el diseño de las funciones de aptitud para cada AG es relativamente sencillo.

Describiendo el AG y la estrategia evolutiva adoptada se tiene en la Fig. 6 de izquierda a derecha un primer VI denominado *Generación de Cromosomas Enteros*, que construye una población aleatoria de individuos codificados en arreglos binarios. Esta primera generación de individuos ingresa al ciclo *For* representado por el rectángulo que contiene el resto de VI involucrados en el AG. Posteriormente, la población ingresa al VI Fitness EHW, que evalúa la aptitud de cada uno de los individuos. Su operación interna se describe en párrafos posteriores. El VI siguiente se conoce como *Ordenar Arreglo* y pertenece al grupo de *VI No Operadores*, puesto que su tarea no representa un operador genético o evolutivo como tal, sin embargo, en los AG es muy útil ya que organiza la población ya sea en orden creciente o decreciente de aptitud. Para este caso particular, los individuos son organizados en orden decreciente de aptitud.

Una vez ejecutado el ordenamiento, se lleva a cabo un proceso de escalamiento dinámico denominado por los autores como *Escalamiento Exponencial Dinámico*. El término dinámico se debe a que varía la presión selectiva en cada generación dependiendo de la diferencia entre las aptitudes del mejor y el peor individuo, de tal forma que entre mayor sea dicha diferencia, mayor presión selectiva se imprime al proceso en dicha generación. Está basado en el *Escalamiento Lineal Dinámico* propuesto por Grefenstette y Baker señalado en[7]. Nuestra propuesta es agregar un exponente α en (1):

$$A'_k = a \times A_k + b_t \qquad (1)$$

resultando (2): $$A'_k = \left(a \times A_k + b_t\right)^{\alpha} \qquad (2)$$

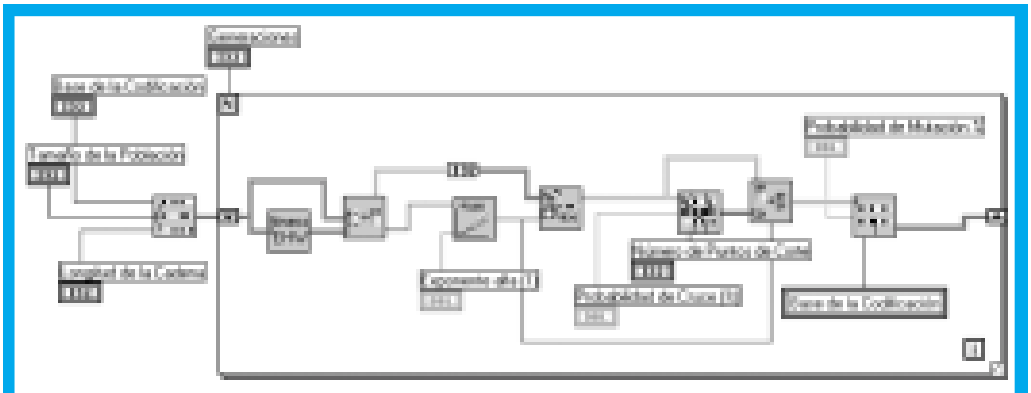


Figura 6. Algoritmo genético implementado en Labview®, utilizando la biblioteca VIGA.

De esta forma se imprime un grado de no linealidad a la función de escalamiento, y se brinda un grado de libertad adicional al mismo. Entre varias metodologías de escalamiento implementadas en el experimento, las de carácter dinámico fueron las que mostraron mejor desempeño en cuanto a los tiempos de respuesta, y un marcado crecimiento de la función de evolución de aptitud.

En este caso particular, para imprimir una mayor dinámica en las variaciones de la población se ha implementado una selección de tipo determinista que complementa la selección estocástica antes descrita, y se conoce como *Selección por Reemplazo Generacional*.

Posteriormente, dicha población con sus funciones de aptitud escaladas, se somete a una selección de tipo estocástica conocida como *Muestreo Universal Estocástico* propuesto por Baker que se describe en [7], y que consiste en asignar un número de copias a cada individuo de manera proporcional a su probabilidad de selección, pero imponiendo como límite superior su valor esperado de copias, evitando así la proliferación de "supercromosomas" y evadiendo el posible estancamiento del AG.

Una vez realizada la selección estocástica se realiza el cruce P_c de individuos, con una probabilidad de cruce determinada por el usuario. El VI utilizado se denomina *Cruce por Puntos de Corte Aleatorio*. Este VI selecciona los padres y realiza el intercambio de genes entre ellos aplicando un número determinado de puntos de corte según el usuario especifique. Los lugares de los puntos de corte se asignan de manera aleatoria, de tal forma, que se obtiene mayor diversidad de soluciones, que si se establecen de manera determinista, contribuyendo a una mejor exploración del espacio solución.

En este caso particular, para imprimir una mayor dinámica en las variaciones de la población se ha implementado una selección de tipo determinista que complementa la selección estocástica antes descrita, y se conoce como *Selección por Reemplazo Generacional*. Consiste en reemplazar a los peores padres de la población por los hijos obtenidos del cruce. Es decir, que se implementan dos tipos de selección en el AG, una de tipo estocástico y otra de tipo determinista.

Por último, se ejecuta el VI de *Mutación de Cromosomas Enteros*, que consiste en que de acuerdo a una probabilidad de mutación P_m determinada por el usuario (que suele ser pequeña), un número aleatorio de genes se escoge entre la población y se modifica su valor. Los individuos mutados también son reemplazados.

Por último, la nueva población de individuos pasa al siguiente ciclo a través de un registro de corrimiento, representado por una flecha que apunta hacia arriba en el borde derecho del ciclo *For*. De esta manera, se corren las iteraciones hasta cumplir con la condición de finalización , que en este caso obedece al número de generaciones impuestas por el usuario.

IV. RESULTADOS EXPERIMENTALES

A continuación se ilustran los principales resultados experimentales obtenidos de 9 repeticiones del experimento de diseño propuesto.

A. Parámetros Básicos del Algoritmo Genético

Los valores de los parámetros del AG se ilustran a continuación:

- Población de individuos = 10.
- Probabilidad de cruce = 0.3%.
- Puntos de corte = 10.
- Probabilidad de mutación = 10%.

Los valores dados son inusuales, ya que los porcentajes son muy bajos para el común de los experimentos en EHW. Esto ocurre, porque la secuencia de operadores genéticos y evolutivos, y la manera en que se están ejecutando facilitan mucho el buen desempeño del algoritmo, a tal punto que el operador fundamental es la mutación [4], [12]. El cruce es utilizado para evitar que la mutación induzca demasiado ruido en el proceso evolutivo.

B. Función de Aptitud

A continuación se describe la operación que cumple el VI *Fitness* EHW en el AG descrito por la Fig. 6.

La aptitud de los individuos es evaluada de modo proporcional al número de aciertos en la salida del sumador al ser excitado por una secuencia aleatoria de los operandos A, B y el acarreo de entrada C_{in}. Se compara dicha salida con el valor correcto de la suma que se encuentra previamente tabulado. La función de aptitud se define según la ecuación (3).

$$A(x) = \sum_{i=0}^{7} N_i \qquad (3)$$

Donde:

$A(x) \equiv$ Es la aptitud del individuo x.
$N \equiv$ Es el número de aciertos en la salida del circuito determinado por el cromosoma x.
$i \equiv$ Es el índice de los datos aleatorios de entrada que excitan al circuito.

Para la evaluación de aptitud del AG, se utilizó el escalamiento exponencial dinámico descrito por la ecuación (2) con α =2, α =1 y b =0.

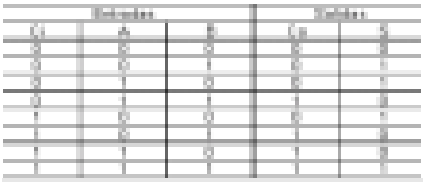
Los datos de entrada se alimentan de manera aleatoria al circuito a través del puerto paralelo tipo D, de esta forma se evita la posibilidad de evolucionar un circuito que solamente responda a una secuencia de entrada determinada. Es decir, que con esto se trata de garantizar que con cualquier secuencia de datos posible el circuito funcionará de manera correcta.

C. Experimento

Las entradas y salidas del circuito sumador completo a evolucionar, se ilustran en la Tabla 1.

Es un circuito con tres entradas, dos salidas que opera de modo combinacional, como puede deducirse de su función de entrada salida.

Tabla 1. Tabla de verdad del circuito sumador completo



El experimento se corrió nueve veces bajo las mismas condiciones. En la Fig. 7, se ilustra una de las

gráficas de la función evolución de aptitud del mejor, el peor y la media aritmética de toda la población. Nótese la convergencia del experimento en la generación 832, donde la aptitud máxima sin escalar es ocho, puesto que son ocho sumas las que deben hacerse de manera correcta como se ilustra en la Tabla 1.

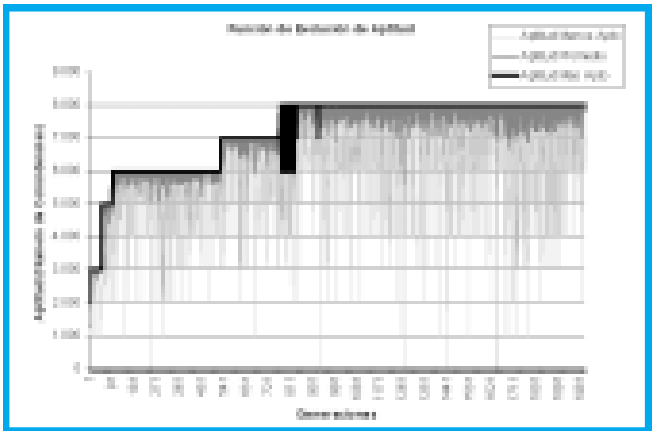


Figura 7. Función evolución de aptitud del mejor y el peor individuos a lo largo de la evolución. También se ilustra la aptitud promedio de la

El cromosoma del mejor individuo del proceso evolutivo en codificación hexadecimal es: D536ABCF973F77031D6F6B9E.

En la Tabla 2 se ilustra la media aritmética de la aptitud de los mejores individuos en la generación 774, que corresponde a la media aritmética de las generaciones de convergencia de cada experimento.

Cabe resaltar que en la generación indicada, la dispersión de las aptitudes de los individuos es muy pequeña, lo cual, da cuenta de la capacidad de convergencia del AG en corto tiempo, ya que en promedio una corrida de 10000 generaciones tarda cerca de 31,64 segundos, es decir, que cada generación tarda en correr aproximadamente 3,164 ms. Al cabo de 774 generaciones que toman cerca de 2,35 s, es posible afirmar que la evolución intrínseca de hardware en el problema planteado, trabaja en tiempos que dependiendo de la aplicación pueden llegar a ser bastante apropiados.

Tabla 2. Datos estadísticos de los mejores individuos y de los tiempos de convergencia

No obstante, observando la dispersión existente en las generaciones de convergencia (358,75), es posible afirmar que el tiempo medio de convergencia no establece un intervalo de confianza lo suficientemente pequeño como para limitar el número de generaciones a un rango pequeño y confiable.

Aunque 832 parezca una cifra muy grande, debe notarse que este experimento muestra un espacio solución relativamente reducido, en el cual, no se trata de hallar una salida determinada para una señal de entrada, sino que coincidan ocho salidas para ocho entradas diferentes aleatorias, y que el mismo circuito cumpla con dichas relaciones entrada-salida sin errores. Además debe tenerse en cuenta que estas 832 generaciones tardaron 2.63 s en correr, que para algunos propósitos específicos es poco tiempo. Debe tenerse presente que los experimentos reportados por Gordon, Kalganova y Miller [2], [10], [11] en evolución extrínseca tardaron entre 1000 y 3000 generaciones promedio en converger.

Vale aclarar que sin tener en cuenta la media del número de generaciones de convergencia, todos los experimentos convergieron, de modo que todos los individuos más aptos de cada ensayo presentaban una aptitud de ocho con desviación estándar cero.

Es muy difícil establecer parámetros cuantitativos entre evolución intrínseca y extrínseca como se ha podido observar en el presente documento ya que cada tipo de evolución tiene sus ventajas y desventajas. Por ejemplo, según Gordon [2], al evolucionar un circuito sumador completo de dos bits con acarreo de entrada, el experimento de evolución extrínseca tardó cerca de cuatro minutos en converger con arreglos de celdas lógicas de 3x3, 3x4 y 4x4 utilizando un microprocesador celeron de 433 Mhz. El mismo experimento fue ejecutado en evolución extrínseca sobre una FPGA Virtex, evitando utilizar codificación binaria y tardó cerca de 3 horas con 45 minutos en un microprocesador celeron de 433 Mhz.

Teniendo en cuenta que el salto tecnológico es bastante grande entre el microprocesador utilizado por Gordon en sus experimentos y el microprocesador empleado por los autores de la presente publicación en la evolución del sumador completo de un bit con acarreo de entrada, el cual, es un atlon de 1.2 Ghz, que casi triplica la velocidad de procesamiento, no da lugar a comparaciones. Además el sumador completo descrito por Gordon es de palabras de dos bits de entrada, un tanto más complejo que el sumador presentado en este documento.

En la Fig. 8, se esquematiza un diagrama de compuertas del circuito evolucionado. Ilustrando las funciones de salida se tiene que:

$$C_o = \overline{(A+B)} \oplus \left[\overline{(A \oplus B)} + C_i\right] \qquad (4)$$
$$C_o = AB + C_i A + C_i B$$

y

$$S = \overline{(A \oplus B) \oplus \overline{C_i}} \qquad (5)$$
$$S = A \oplus B \oplus C_i$$

Nótese que las funciones lógicas obtenidas al ser simplificadas por métodos convencionales tienen el comportamiento del sumador completo buscado, lo cual garantiza el correcto funcionamiento del circuito.

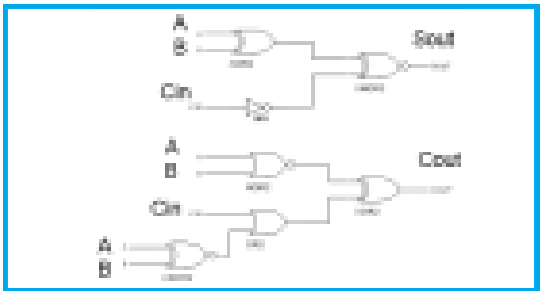


Figura 8. Diagrama de compuertas del circuito sumador completo obtenido.

V. CONCLUSIÓN

Se describió el procedimiento experimental llevado a cabo en la implementación de un circuito sumador completo, diseñado por medio de estrategias bioinspiradas sobre una FPGA XC4000 de Xilinx.

Se propuso una estrategia evolutiva para dicho problema, tomando elementos de diversos experimentos en evolución extrínseca reportados en la literatura. Dada la diferencia insalvable existente entre la evolución intrínseca y la extrínseca, muchas de las características de complejos procesos de diseño debieron ser optimizadas, algunas replanteadas y otras omitidas, con el fin de minimizar los retardos que implica la evolución intrínseca.

Uno de los puntos principales fue prescindir de la utilización de cromosomas de longitud variable, y con esto la variación en la geometría del circuito y el operador de mutación de geometría. El mecanismo de crear celdas complejas con conexiones variables dejando algunas conexiones entre celdas fijas, es una manera de variar la conectividad e indirectamente la geometría del circuito, dando buenos resultados, ya que se obtuvo una gran diversidad de circuitos sumadores.

En cuanto a la manera de evitar circuitos redundantes, se habilitaron matrices pequeñas que limitan el número de compuertas a utilizar. Aunque esto no garantiza la falta de redundancia de los circuitos, el número de recursos empleados fueron reducidos. Para trabajo futuro se implementarán funciones de penalización o funciones de aptitud multiobjetivo como las propuestas por Kalganova [10] que en resumidas cuentas es un modo de penalización para los individuos que comprometen más recursos.

A diferencia de los experimentos reportados por Kalganova y Miller [10], [11], se puso a prueba la capacidad de discriminación entre circuitos secuenciales y netamente combinacionales, ya que los elementos de almacenamiento no fueron deshabilitados de las celdas. Sin embargo, los circuitos obtenidos resultaron combinacionales y todos los flip-flops de las celdas se desactivaron de forma espontánea al final de los nueve ensayos realizados.

Los tiempos de convergencia resultaron bastante prometedores, ya que la generación promedio fue la 774, que comparada con los trabajos consultados cuyo rango de convergencia oscila entre 1000 y 3000 generaciones dan cuenta de un buen rendimiento en la convergencia.

La generación promedio fue la 774, que comparada con otros trabajos cuyo rango de convergencia oscila entre 1000 y 3000 generaciones dan cuenta de un buen rendimiento en la convergencia.

REFERENCIAS BIBLIOGRÁFICAS

[1] J. MARTINEZ y S. ROJAS, Introducción a la informática evolutiva: Un nuevo Enfoque para Resolver Problemas de Ingeniería, Santafé de Bogotá: Universidad Nacional de Colombia., 1999, pp. 185.

[2] T. G. W. GORDON y P. J. BENTLEY, "On Evolvable Hardware," Soft Computing in Industrial Electronics, S. Ovaska and L. Sztandera Eds. Physica-Verlag, Heidelberg 2002.

[3] X. YAO y T. HIGUCHI, "Promises and Challenges of Evolvable Hardware," IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews, 1999.

[4] J. M LUNA, C. DEVIA y A. BETANCOURT, "Divisor de Frecuencias en Hardware Evolutivo," en: Revista Científica, No 5, Vol. 1, Agosto 2003, pp. 79-91.

[5] M. CERRALOZA, W. ANNICCHIARICO, Algoritmos de optimización estructural basados en simulación genética. Caracas : Universidad Central de Venezuela, 1996, pp. 163. (Colección Monografías ; no. 47)

[6] E. FALKENAUER, Genetic Algorithms and Engrouping Problems. New York : John Wiley, 1998, pp. 220.

[7] M. GEN y R. CHENG, Genetic Algorithms and Engineering Problems. New York : John Wiley, 1998, pp. 411.

[8] J. LUNA, C. DEVIA. Desarrollo de una Plataforma Experimental en Hardware Evolutivo - DEEP. Bogotá DC, 2004, 154p. Tesis (Pregrado en Ingeniería Electrónica): Universidad Distrital Francisco José de Caldas, Facultad de Ingeniería.

[9] J. LUNA, C. DEVIA. Diseño de una máquina de estados finitos en hardware evolutivo. En: CONGRESO INTERNACIONAL DE LA REGIÓN ANDINA. (2º : 2004 : Bogotá). Memorias del II Congreso Internacional de la Región Andina - ANDESCON 2004. Bogotá : IEEE, 2004. 6p.

[10] T. KALGANOVA, J. MILLER. Evolving More Efficient Digital Circuits By Allowing Circuit Layout Evolution and Multi-Objective Fitness. The First NASA/DoD Workshop on Evolvable Hardware. 1999. http://citeseer.ist.psu.edu/kalganova99evolving.html

[11] J. F. MILLER, P.THOMSON, T. FOGARTY. Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study. Dept. of Computer Studies, Napier. Genetic Algorithms and Evolution Strategy in Engineering and Computer Science. 1997. http://citeseer.ist.psu.edu/miller97designing.html

[12] ATMAR, Wirt. Notes on the simulation of evolution. En: Transactions on neural networks. Vol. 5, No. 1 (1994) p. 130-148.

José. M. Luna
Estudiante de Ingeniería Electrónica de la Universidad Distrital Francisco José de Caldas. Miembro del Grupo de Investigación LAMIC, **jmarcio_1@yahoo.com**

Christian. J. Devia
Estudiante de Ingeniería Electrónica de la Universidad Distrital Francisco José de Caldas. Miembro del Grupo de Investigación LAMIC, **christiandevia@hotmail.com**

Alvaro Betancourt Uscátegui
Ingeniero Electrónico, Universidad Distrital Especialista en Telecomunicaciones Móviles, Universidad Distrital, Msc. Ciencias Financieras y de Sistemas, Universidad Central, Magister en Ingeniería, Informatique Appliquée, Ecole Polytechnique Université de Montreal, Canada, Profesor Facultad de Ingeniería, Universidad Distrital, Director del Grupo de Investigación LAMIC - U. Distrital. **abetancourt@udistrital.edu.co**