Artículo de Investigación. Revista Ciencia e Ingeniería. Vol. 38, No. 1 pp. 3-16, diciembre - marzo, 2017.
ISSN 1316-7081. ISSN Elect. 2244-8780 Universidad de los Andes (ULA)

Análisis de Circuitos Fractales y Modelado a través de Sistema de Funciones Iteradas para VHDL

Caso de estudio: Codificador Reed Solomon

Analysis of Fractal Circuits and Modeling through Iterated Functions System for VHDL

Case Study: Reed Solomon Encoder

[1] Sandoval-Ruiz, Cecilia E.
[1]Facultad de Ingeniería / Dirección de Postgrado
Universidad de Carabobo, Venezuela.
[*]cecisandova@yahoo.com

Resumen

La presente investigación corresponde al modelado de circuitos para hardware reconfigurable, basado en la teoría fractal, que aplica conceptos de funciones iteradas para la generación del código de descriptción en VHDL, lo que es un tratamiento novedoso como método de configuración de hardware. El método empleado consisten en el análisis de la estructura fractal de los circuitos: generadores de código Reed Solomon, y los multiplicadores de campos finitos GF, estudiando la arquitectura generalizada del componente LFSR (registros desplazamientos con realimentación lineal), presente en estos. Para futuras aplicaciones, se pretende incluir este desarrollo en control adaptativo y control avanzado aplicando funciones iteradas, con el manejo fractal de los circuitos de implementación. Se obtuvo un conjunto de ecuación con funciones iteradas que reproducen su comportamiento, basados en la estructura fractal del circuito, del modelo se estimó el consumo de recursos hardware, a nivel de operadores lógicos, con lo que el método propuesto muestra eficiencia para su aplicación en sistemas lógicos avanzados. El aporte principal es el manejo matemático de una aplicación experimental, donde se pueden extrapolar los avances e incorporar estos conceptos para nuevos enfoques de diseño. Lo que permite concluir, que el desarrollo optimiza el diseño de circuitos en VHDL, a partir del modelo de funciones iteradas obtenido, simplificando la descripción de hardware parta circuitos paralelos, de alta eficiencia aplicando modelado SFI, siendo una base para la generación de conocimientos en esta rama de la Ingeniería.

Palabras Clave: Estructura Fractal, SFI, LFSR, VHDL, códigos RS, Hardware Reconfigurable.

Abstract

The present research corresponds to circuit modeling for reconfigurable hardware, based on fractal theory, which applies concepts of iterated functions for the generation of the description code in VHDL, which is a novel treatment as a method of hardware configuration. The method used consists of the analysis of the fractal structure of the circuits: Reed Solomon code generators, and the finite field multipliers GF, studying the generalized architecture of the LFSR component (linear displacement records), present in these. For future applications, it is intended to include this development in adaptive control and advanced control handling iterated functions, with the fractal management of the implementation circuits. It was obtained a set of equations with iterated functions that reproduce their behavior, based on the fractal structure of the circuit, the model was estimated the consumption of hardware resources, at the level of logical operators, with which the proposed method shows efficiency for its application in Advanced logic systems. The main contribution is the mathematical management of an experimental application, where you can extrapolate the advances and incorporate these concepts for new design approaches. This allows to conclude that the development optimizes the design of circuits in VHDL, from the iterated function model obtained, simplifying the hardware description for parallel circuits, high efficiency by applying SFI modeling, being a basis for the generation of knowledge in This branch of Engineering.

Keywords: Fractal Structure, SFI, LFSR, VHDL, RS codes, Reconfigurable Hardware.

1. Introducción

Los fractales además de caracterizarse por la belleza de sus formas, son herramientas de gran potencial para el estudio de diferentes áreas del conocimiento (Molero 2011), como pueden ser la compresión de imágenes, audio y vídeo, modelado del tráfico en redes, estudios de sistemas dinámicos, análisis de patrones, etc..

La geometría fractal y la teoría de los sistemas dinámicos están íntimamente ligadas (Magaña y col. 2011), dando paso a nuevas alternativas científicas para la solución y optimización de estos sistemas. En los casos del codificador Reed Solomon, que opera sobre datos transmitidos a través de un canal dinámico, o en el control adaptativo, que modifica sus parámetros en respuesta a la dinámica del sistema, resulta de interés su tratamiento a través de conceptos de teoría fractal.

Dichos conceptos han permitido identificar estructuras y modelos para el análisis de procesos dinámicos, sistemas complejos, incluyendo circuitos eléctricos (Rivera y col., 2012). Considerando el modelado de sistemas circuitales de hardware reconfigurable, para la implementación a través de lenguaje descriptor de hardware VHDL – *VHSIC Hardware Description Language*.

En esta materia se puede notar un campo de aplicación que resulta propicio por la naturaleza del diseño hardware mediante lenguaje VHDL, que se aplica en la etapa de configuración métodos de simplificación y optimización, para la implementación sobre tecnología *Field Programmable Gate Array* – FPGA. Donde es importante reconocer estructuras similares entre los elementos componentes de un sistema y desarrollar el modelo matemático, basado en esta geometría, que se puede tratar como un modelado fractal de hardware.

En la actualidad existen numerosos modelos matemáticos que permiten definir estructuras fractales asociados con problemas particulares. Uno de los modelos matemáticos más popular para crear objetos fractales es el conocido como sistema de funciones iteradas –SFI (Magaña y col., 2011). Así mismo, destacan trabajos de investigación (Adame 2005), en los cuales se presenta el estudio de estos para la obtención del fractal asociado. Partiendo de esto se ha considerado de manera recíproca una estructura circuital fractal que puede ser modelada a través de un SFI.

Así estos pueden aplicarse para casos particulares de sus componentes y sus resultados ser extrapolados para la generalización de circuitos eficientes. En el cual, el tratamiento del modelo se puede desarrollar en base a la algoritmia característica de las estructuras fractales identificadas.

Se ha seleccionado así el caso de estudio del codificador Reed Solomon, siguiendo la tesis de modelado a partir de la estructura de registros de desplazamiento realimentados (Sandoval 2013), donde se puede observar que tanto el codificador, como el reductor modular del multiplicador en campos finitos de Galois, generan una secuencia pseudo-aleatoria, lo que es una característica de los generadores con estructura LFSR – *Linear Feedback Shift Register*, estos generadores están compuestos por un registro desplazamiento y una función de realimentación.

[1] Muchas investigaciones actualmente, se orientan al desarrollo de modelos VHDL para la descripción de aplicaciones, siendo de vital utilidad un modelo matemático que permita generar el código VHDL, a partir de ecuaciones que manejen funciones iteradas para su codificación. Estas apuntan hacia el estudio de las estructuras circuitales que usan LFSR (Sandoval 2012, 2010), a fin de configurar el circuito de forma concurrente, sobre hardware reconfigurable y obtener versiones eficientes de los mismos (Sandoval y col., 2014).

En este trabajo se plantea el reconocimiento de una estructura fractal y un patrón iterativo en las funciones que describen el codificador RS, que permita modelar el comportamiento del hardware para el circuito del codificador RS, considerando de interés el estudio de las características identificadas en las estructuras, considerando el análisis fractal y los sistemas de funciones iterativos que se presentaron en la descripción espacio – tiempo.

2. Análisis de Circuitos Fractales

El análisis fractal de circuitos generadores de códigos, filtros adaptativos, control reconfigurable, resulta propicio por las características de estos. Lo que puede permitir un nuevo enfoque en la configuración de sistemas con características auto-similares, así como algoritmos recurrentes, lo cual ha sido el punto de partida de la presente investigación, en la cual se ha seleccionado como caso de estudio los codificadores Reed Solomon.

Entre las propiedades de los fractales destacan la autosemejanza (Molero 2011). En general, una estructura se dice que cumple este criterio, si puede ser construida como una reunión de estructuras, cada una de las cuales es una copia de la original pero a tamaño reducido.

Este concepto de *Auto-similitud* o Auto-semejanza, se puede enunciar como la propiedad de una estructura en el que ésta es exacta o aproximadamente semejante a sus componentes, es decir, el conjunto se puede descomponer en un número finito de partes de modo que una de ellas es idéntica, salvo escala, al todo. Dicha propiedad de simetría de escalamiento se encuentra presentes en diversas ramas de la ciencia y la tecnología (Lee y col., 2010).

Otra propiedad establece que para generar un fractal basta con muy poca información, la clave se encuentra en la iteración, que consigue generar una gran cantidad de estructuras a partir de esa información inicial. De allí que un método para generar estructuras fractales es el *Sistemas de Funciones Iteradas – SFI*, desarrollado en los años 80 fundamentalmente por J. E. Hutchinson y M. Barnsley. Se empleó estos sistemas para modelar objetos que a diferentes

escalas presentan una relación de semejanza con la estructura completa (Molero 2011).

El método consiste en establecer una semilla inicial y ejercer sobre ella una serie de transformaciones. Este resultado de sucesivas iteraciones recibe el nombre de atractor del sistema. El conjunto de transformaciones que se aplican para llegar hasta éste, es lo que se denomina Sistema de Funciones Iteradas (SFI).

Es así como existen los fractales definidos por ecuaciones, donde una de las formas más populares para generarlos es a través de una construcción matemática usada para representar, de manera simple, ciertos conjuntos fractales que presentan auto-similitud. Basados en este concepto, se pretende reconocer esta propiedad en los circuitos generadores de secuencia asociados al codificador Reed Solomon, para obtener las ecuaciones matemáticas que permitan su descripción para VHDL.

En diversos trabajos previos (Rivera y col., 2012) (Ramírez y col., 2012), las investigaciones apuntan en reconocer las propiedades fractales de las secuencias Fibonacci. Cuya definición viene dada por: $f_{k+2} = f_{k+1} + f_k$, para el caso del generador de secuencias de Fibonacci se muestra un circuito recursivo, que se puede implementar a través de funciones iteradas. Es importante destacar la similitud entre los circuitos generadores de secuencia Fibonacci y Galois, estos últimos propios del procesamiento asociado al codificador Reed Solomon y su componente auto-similar la reducción modular del multiplicador en campos finitos de Galois.

2.1. Estructura del Generador de código RS

El código Reed Solomon es un subconjunto de los códigos BCH y son de bloques lineales. Un código Reed Solomon se especifica como RS(*n*,*k*) con símbolos de *m* bits. Lo anterior significa que el codificador toma *k* símbolos de los *m* bit y añade símbolos de paridad para hacer una palabra de código de n símbolos. Existen n-k símbolos de paridad. Esto lo realiza a través de un generador polinomial, cada palabra de código Reed Solomon es generada usando un polinomio especial, así todas las palabras de código válidas son divisibles exactamente por el polinomio generador.

En el codificador Reed Solomon (Sandoval y col., 2008), se puede identificar la arquitectura característica del generador de símbolos de redundancia, para la implementación de las ecuaciones que describen matemáticamente el codificador RS (Sandoval 2007), siendo su definición de comportamiento dado por la ecuación 1.

$$R(x) = D(x)\, x^{i} \bmod G(x) \qquad (1)$$

Esta arquitectura corresponde a un elemento circuital generador de secuencias pseudo-aleatorias conocido como *n*,*k*-LFSR, el cual es un circuito secuencial de *n-k* etapas de *m* bits cada una, donde los datos son operados (en algebra de campos finitos) con los coeficientes del polinomio generador del codificador *G(x)*.

Esto se lleva a cabo, a través de componentes multiplicadores en campos finitos de Galois GF(2^m), los cuales son los responsables de obtener los elementos a operarse en las compuertas XOR y almacenarse en los elementos de memoria, como se muestra en la figura 1.

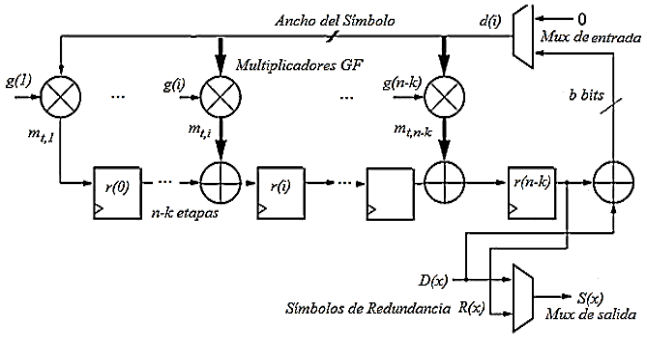

Fig. 1. Arquitectura del Codificador RS(*n*,*k*) Genérico

El generador de símbolos de redundancia del codificador recibe el primer símbolo de la entrada de datos, este símbolo pasa a la salida sin alterar, a través del componente selector del símbolo de salida (*Mux de salida)*, a la vez es operado con el dato almacenado en el registro menos significativo *r(0) a través de la xor de entrada*, generando un símbolo dado por: *D(x) xor r(0)*, éste es ingresado a través del siguiente multiplexor (*Mux de entrada)* a la realimentación del circuito LFSR y se realiza la operación de éste con cada uno de los coeficientes del polinomio generador *G(x)*, a través de la multiplicación en algebra de campos finitos de *Galois*.

El cálculo parcial del producto se opera a través de una *xor* de *m* bits, con la salida del registro precedente, al recibir la señal de reloj, el dato es almacenado en el registro a la salida de la *xor*. Este proceso se realiza para los *k* símbolos de datos, donde los resultados dependen de la secuencia de símbolos de entrada. Alcanzados los *k* símbolos, el *mux de entrada* conmuta a la selección de la entrada cero y el multiplexor de salida selecciona el símbolo del *r(0)* durante *(n-k)* pulsos de reloj, los cuales corresponden a los símbolos de redundancia generados del proceso, bajo esta secuencia de dos etapas se genera la palabra de código Reed Solomon.

En el caso del codificador Reed Solomon, que opera sobre los datos, transmitidos, a través de un canal dinámico, resulta de interés estudiar a éste como un elemento con características fractales. De esta manera, se estudia una interpretación fractal de los modelos circuitales que describen el comportamiento de un codificador Reed Solomon (Sandoval y col., 2013), en el cual se ha identificado una similitud entre la estructura del componente multiplicador y el codificador RS.

2.2. Aplicaciones del LFSR

En (Moon 2005) se presenta todo el tratamiento de los LFSR para procesamiento de datos en base polinómica. Entre las aplicaciones del circuito encontramos elementos de procesamiento binario, como lo son los multiplicadores de campos finitos de Galois (Sandoval 2010), donde el ancho del bus de datos corresponde a un solo bit y cuyos coeficientes vienen dados por los polinomios característicos del campo *P(x)*. Así mismo, encontramos elementos de procesamiento no binario, dado por símbolos de *m* bits, entre los que destacan los codificadores Reed Solomon (Sandoval y col., 2013), en este último los coeficientes de la estructura están dados por el polinomio generador del código *G(x)*.

Dada la importancia de estos circuitos, por su capacidad de generar secuencias con buenas propiedades estadísticas (Ekdahl 2003), resulta pertinente un análisis de la estructura generalizada. Estos circuitos pueden ser lineales o no lineales – NLFSR, de acuerdo a los operadores en la función de realimentación, tomada de cualquier etapa.

Estos circuitos basados en registro desplazamiento con realimentación lineal – LFSR, han sido objeto de estudio (Dubrova 2008), (Pérez 2009), sus aplicaciones (Alvarez 2005) y arquitecturas han sido estudiadas, y a partir de su representación de Galois han generado una arquitectura alternativa, conocida como FCSR – *Feedback with carry shift registers* (Goresky y col., 2002), donde se evidencia la importancia de nuevos modelos generalizados, para esta estructura.

Para el modelado del LFSR es necesario expresar el comportamiento del circuito de forma genérica, a fin de describir la estructura de forma reconfigurable. Siendo necesario definir los parámetros de escalabilidad del circuito en función del número de etapas y tamaño del bus. En base a esta necesidad, se han encontrado desarrollos en técnicas de factorización de las funciones y operaciones comunes, que dan paso al concepto de LFSR reconfigurables (Alaus y col., 2008), lo que permite la reutilización de hardware del circuito R_LFSR.

La generalización del LFSR será aplicada en la configuración de hardware en VHDL, con un tratamiento paramétrico del vector de memoria, el tamaño del bus, la capacidad de los operadores y los coeficientes del polinomio generador. De tal manera, que realizando la concatenación de componentes del circuito se puede tratar el diseño de forma modular con habilitadores de etapas, a fin de emplear el LFSR reconfigurable, como modelo base, en este caso se emplea una generalización para manejar la estructura y los coeficientes.

Destacando que las aplicaciones estudiadas, realizan su procesamiento bajo arreglos particulares de una arquitectura circuital común (LFSR), se busca mantener la estructura con selección de las condiciones específicas, para establecer un modelo generalizado de las estructuras auto-similares.

Profundizando en el estudio del codificador RS para su implementación eficiente, se encuentra que estos emplean aritmética de campos finitos de *Galois* GF(2^m), por producir resultados con longitud fija. Donde se debe tener en cuenta que la eficiencia computacional de las operaciones aritméticas en campos finitos está estrechamente relacionada con la forma particular en la cual los elementos del campo son presentados (Kim y col., 2002). Por lo que se ha detectado la necesidad de estudiar alternativas de solución para la implementación de operaciones aritméticas GF(2^m), basadas en los fundamentos matemáticos de los campos finitos.

Los *Campos Finitos de Galois* (GF) constituyen un área específica de la matemática desarrollada por E. Galois. Donde el campo es especificado a través de un elemento primo *p*, base del campo; y un entero positivo *m*, longitud del elemento del campo. Se cumple que p^m corresponde al número de elementos del campo, y las operaciones aritméticas sobre el campo finito da como resultado un elemento que pertenece al mismo campo (Saqib 2004). Estas propiedades son utilizadas en aplicaciones de codificación, decodificación Reed-Solomon (Xilinx 2011) y criptografía. Una presentación ampliamente utilizada es la forma polinomial, en la cual se define un polinomio generador del campo, conocido como polinomio irreducible *P(x)*, el cual se operará módulo con los resultados de las operaciones para llevar el resultado a la longitud fija definida para el campo.

2.3. Multiplicadores de Campos Finitos de Galois

El multiplicador en campos finitos es generalmente más complejo que un multiplicador convencional, estos han sido un tópico de investigación desde 1960 hasta la actualidad, por ello la relevancia de interpretar la lógica circuital de soporte para estos módulos y establecer un modelo eficiente para su diseño sobre dispositivos de hardware FPGA.

Los multiplicadores aritméticos sobre campos finitos de *Galois*, tienen como base la multiplicación de dos elementos del cuerpo finito y la reducción del resultado mediante un polinomio irreducible *P(x)* de grado *m*. Estos pueden ser implementados a través de un modelo algorítmico, el cual reproduce el comportamiento del multiplicador de forma secuencial o modelos paralelos, según la necesidad de que estos módulos aritméticos operen a frecuencias elevadas y ocupen el menor área posible, lo que demanda un equilibrio entre estos factores.

2.4. Reducción Modular GF

Si *P(x)* es el polinomio irreducible, entonces la multiplicación de dos elementos del campo, representados como los polinomios *A(x)* y *B(x)*, es el producto algebraico de los dos polinomios, y la reducción modular con *P(x)*, como se presenta en la ecuación 2.

$$C(x) = A(x) \times B(x) (mod\, P(x)) \quad (2)$$

Donde *A(x)* corresponde en el codificador Reed Salomon a un coeficiente del polinomio generador del código,

P(x) el polinomio irreducible del campo de Galois, y *B(x)* la entrada de datos a multiplicar, donde la operación módulo del polinomio *P(x)*, correspondiente a la reducción modular, *red_mod = A(x) x^i mod P(x)*, es realizada bajo la arquitectura de un circuito LFSR (ver figura 2).

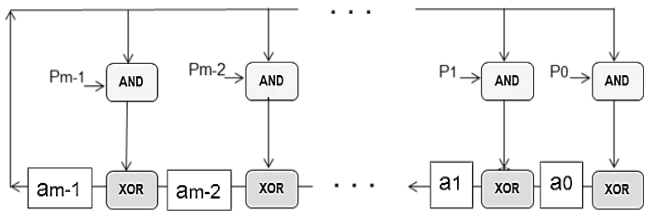

Fig. 2. Arquitectura del módulo LFSR *A(x) x^i mod P(x)*

Esta operación, originalmente es implementada a través de registros para obtener los *n* vectores con corrimiento de los bits. En esta configuración se basa el análisis del comportamiento del circuito en *m* ciclos de reloj, donde *m* es igual al número de bits de la palabra. En este punto se puede observar que el comportamiento de los circuitos estudiados guardan similitud, en el procesamiento de los datos, la estructura circuital y la lógica que lo define.

2.5. Estructura Auto-similar del Circuito y sus componentes

Llegado a este punto de la disertación, se han estudiado de forma separada las estructuras componentes de los elementos del codificador Reed Solomon. Identificando una arquitectura común entre el circuito generador de símbolos de redundancia y el circuito reductor modular del multiplicador de campos finitos, lo que resulta de vital interés para establecer una descripción VHDL.

Es importante señalar que el circuito codificador RS, presenta un componente con la misma arquitectura, que difiere en escala con respecto al ancho de bus de datos y a los operadores: *and* por *prod_GF* y *xor* binario por *xor* entre símbolos.

Así se identifica un LFSR cuya función es implementada, a través de un LFSR interno, lo que permite observar el *Isomorfismo* estructural entre el componente de reducción modular *A(x) x^i mod P(x)* y el generador de símbolos de redundancia del codificador *D(x) x^i mod G(x)*. Donde el polinomio generador del campo *P(x)* y el polinomio generador del código *G(x)* definen los coeficientes de entrada de los multiplicadores en la función de realimentación para cada caso.

Dado que un método para el modelado son los SFI, se considera este concepto de sistemas de funciones iteradas, para modelar los circuitos auto-similares, que constituyen el generador de código. La función característica de la estructura común del generador de código, modifica un elemento pero mantiene su tamaño. En el caso de la función multiplicación en campos finitos GF, se obtiene un elemento con iguales características, se puede así establecer una función recursiva sobre la misma función.

De esta manera se ha identificado que el procesamiento de los datos *d(x)*, al aplicar una función LFSR para generar las secuencias del multiplicador GF y del codificador RS, por lo que se puede expresar, en función de los correspondientes polinomios generadores de la secuencia del LFSR, siendo *G(x)*, el polinomio generador de código, característico del LFSR externo y *P(x)*, el polinomio característico del campo GF, asociado al LFSR interno, obteniendo la ecuación 3.

$$R(x) = lfsr\_G\,[lsfr\_P\,[d(t)]] \quad (3)$$

3. Modelado en VHDL de circuitos con Sistemas de Funciones Iteradas – SFI.

El método de modelado basado en funciones iteradas, comprende la adaptación de las expresiones matemáticas que definen el comportamiento del codificador Reed Solomon y sus componentes, a fin de realizar la descripción en VHDL de su comportamiento, expresando las iteraciones necesarias para la establecer el código que permita la configuración del circuito sobre hardware. En una primera etapa se realiza de una tabulación de operaciones circuitales, seguido de la descripción en VHDL del sistema, hasta obtener las ecuaciones para la implementación.

En primer lugar se analizaron las características de los módulos componentes, se encontró una función que modifica un elemento pero mantiene su tamaño, corresponde a la función multiplicación en campos finitos GF, cuyo método de generación comprende una realimentación del resultado al sistema de procesamiento, lo que se puede interpretar como un *sistema iterado con realimentación*. Donde las ecuaciones descriptivas pueden ser re-formuladas en función de los correspondientes polinomios generadores de la función iterada, que genera el resultado parcial.

Por otra parte, se aplica el método de paralelización del componente multiplicador, basado en un circuito de realimentación LFSR, a través de un tratamiento de los elementos de memoria almacenan los datos procesados, en función de la posición de los elementos *i* en un momento dado de

tiempo *t* (iteración), lo que nos permite observar que estamos en presencia de una función iterada, para la que se desarrollará el modelado por el método de descripción espacio – temporal, de los componentes parciales de la secuencia.

De esta manera, se logra definir las ecuaciones para el modelo concurrente del multiplicador de campos finitos, el cual ha sido extendido para el componente similar correspondiente al generador de símbolos del codificador RS. El método de diseñado, consistió inicialmente en describir la función del LFSR, sobre los datos de entrada, esto comprende una descripción en VHDL de cada uno de los elementos generados, para cada instante de tiempo.

A partir de la observación de la arquitectura del generador de símbolos de redundancia: $D(x)\ x^i \bmod g(x)$ del codificador Reed Solomon, se reconoció un arquitectura similar con el circuito de reducción modular: $A(x)\ x^i \bmod P(x)$, encontrado en el multiplicador GF, donde existe una correspondencia entre los coeficientes del polinomio generador $G(x)$, que define el código Reed Solomon y los coeficientes del polinomio irreducible $P(x)$, que define el campo de Galois GF.

Motivo por el cual se consideró evaluar la extensión del modelo concurrente del LFSR desarrollado en el multiplicador (en campos de Galois), al circuito generador del código Reed Solomon (*n*,*k*). Así la relación lógica-matemática, para cada uno de los elementos generados por el componente LFSR, se describe cada función de operadores asociados a la longitud del elemento a operar, donde el operador $\otimes$ corresponde a la operación producto GF, entre símbolos, y el operador $\oplus$, representa la suma.

A partir de la descripción, se identifican los operandos comunes en la secuencia de generación y los parámetros que varían a lo largo del algoritmo de descripción, con el fin de analizar su comportamiento (tabla 1).

Tabla 1. Cálculo de coeficientes $A(x)x^i mod P(x)$ *para* $GF(2^8)$

t	*i=7*	…	*i=0*
1	$a_{1,7}$	…	$a_{1,0}$
2	$a_1(6) \oplus a_1 \otimes p_7$	…	$a_{1,7} \otimes p_8$
3	$a_2(6) \oplus a_{2,7} \otimes p_7$	…	$a_{2,7} \otimes p_8$
4	$a_3(6) \oplus a_{3,7} \otimes p_7$		$a_{3,7} \otimes p_8$
5	$a_4(6) \oplus a_{4,7} \otimes p_7$		$a_{4,7} \otimes p_8$
…	…	…	…
8	$a_7(6) \oplus a_{7,7} \otimes p_7$	…	$a_{7,7} \otimes p_8$

Siendo $a_{t,i}$ el coeficiente del polinomio operado en la iteración *t*, en la posición *i* de la estructura LFSR del multiplicador. Es así como, en la tabla 1 se han representado las expresiones que permiten calcular los valores de dichos coeficientes en función de la posición espacial del LFSR, para este caso de estudio con 8 elementos definidos por *i* de 0 a 7, en cada una de las iteraciones *t*, correspondientes al cálculo del producto en campos finitos, estos términos definen un componente hardware para su implementación circuital.

De la generalización del método para el cálculo de los coeficientes $a_{t,i}$, se obtuvo la ecuación, para generar cada elemento, correspondiente a aplicar el circuito LFSR sobre un polinomio $A(x)\ x^i$, estos dependen del instante de tiempo y la posición del elemento en el polinomio, el coeficiente p_i está dado por la posición del coeficiente en el polinomio generador del campo $P(x)$, el cual será fijo en el tiempo.

Ahora bien, la recursividad en las operaciones en campos finitos, hace que el mismo circuito opere los datos realimentados, para lo cual los resultados parciales pueden ser renombrados y operados con la misma función. Estamos en presencia de una función iterada, de manera de presentarse circuitos operadores que generan resultados que serán operados por la misma estructura, estamos en presencia así de circuitos fractales del mismo nivel.

De forma alternativa, la descripción del multiplicador, se puede particionar la implementación en componentes LFCS (LFSR concurrentes), logrando una representación RTL multinivel por elementos circuitales, para describir sub-funciones reutilizables. Este concepto de lógica multinivel (Imaña y col., 2002), permite definir componentes circuitales interconectados, a través de capas, para generar las salidas.

Es el caso de la estructura LFCS, las señales pueden atravesar un número arbitrario de compuertas en el proceso combinacional, pero la reutilización de los elementos circuitales es una característica que permite disminuir la complejidad lógica, donde se presenta una red de componentes, para representar los *ciclos iterados*, una solución que simplifica la descripción de manera eficiente, logrando que se obtenga a través de una expresión lógica, el modelo concurrente (ver figura 3).

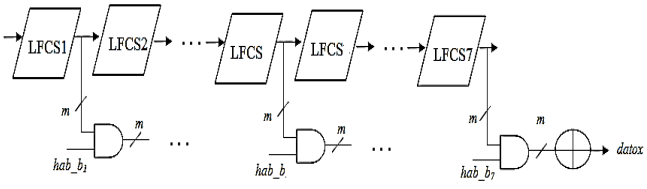

Fig. 3. Lógica Multinivel de componentes LFCS

Como resultado del análisis, en función de los parámetros, se obtiene un esquema para la generación del código de descripción de los componentes en VHDL, como se presenta en la tabla 2, donde se asignan los puertos de los símbolos de entrada al módulo LFCS y se obtiene cmo resultado su operación *red_mod*, luego estos resultados parciales son operados en el producto final.

Tabla 2. Descripción VHDL de lógica multinivel

```
entity mult is
--(Las declaraciones idénticas al multiplicador)
Port (a: in std_logic_vector (7 downto 0);
      datox : out std_logic_vector (7 downto 0));
end mult;
architecture Behavioral of mult is
signal
b,b1,b2,b3.b4,b5,b6,b7,b8,c1,c2,c,3,c4,c5,c6,c7,c8
: std_logic_vector (7 downto 0);
Begin
b<= D_dato; -- dato de entrada para la multiplica-
ción
a1<= coef; --a1<= coef;
p<="100011101";   --   Primitive   polynomial   =
D^8+D^4+D^3+D^2+1 (285)
u1: LFCS  port map  (a1,a2);
u2: LFCS  port map  (a2,a3);
...
ui: LFCS  port map  (ai,ai+1);
for i=m-1 to 0
-- Construcción del operador bt:
b1<= b(0) & b(0) & b(0) & b(0) & b(0) &… &b(0);
...
b8<= b(7) & b(7) & b(7) & b(7) & b(7) & … & b(7);

-- Productos Parciales: Lógica AND del multiplica-
dor
c1<=a1 and b1;
...
c8<=a8 and b8;
-- ct <= at and bt
-- Acumulador de Productos Parciales:
datox<=c1 xor c2 xor c3 xor c4 xor c5 xor c6 xor
c7 xor c8;
end Behavioral;
```

El componente LFCS, al cual se hace referencia como un módulo de operación en hardware en la descripción del multiplicador, corresponde a la función iterativa que describe el comportamiento de la reducción modular en el campo definido por *P(x)*. Este componente es descrito en la tabla 3, en la cual se especifican los elementos no nulos del polinomio irreducible en las ramas a operar.

Tabla 3. Descripción VHDL del componente LFCS

```
entity LFCS is
port     (a: in std_logic_vector (7 downto 0);
          r: out std_logic_vector (7 downto 0));
end LFCS;
architecture Behavioral of LFCS is
signal p: std_logic_vector (8 downto 0);
begin
a<= coef;
p<="100011101";--polynomial=D^8+D^4+D^3+D^2+1(285)
-- Generación de términos del LFCS:
-- ut:at+1 <= at(i) xor (at(m-1) and p(i) & ...

u: r   <= a(6 downto 4)&(a(3)xor a(7))&(a(2)xor
a(7))&(a(1)xor a(7))&a(0)& a(7);
end Behavioral;
```

A partir de esto, se realiza la descripción en VHDL del comportamiento del codificador, bajo un tratamiento de concreción de las ecuaciones en forma de funciones iteradas, como se presenta en las tabla 4.

Tabla 4. Descripción VHDL del Codificador Reed Solomon

```
entity encoder is
    generic(length:integer:= 17;    --length n-k+1
            width :integer:=7);
    Port ( D_in : in std_logic_vector(6 downto 0);
           clk,hab : in std_logic;

architecture Behavioral of encoder is
type  memoria  is  array  (0  to  length-1)  of
std_logic_vector(width-1 downto 0);
signal ...
component mult is
port  (D_dato: in std_logic_vector (6 downto 0);
       coef: in std_logic_vector (6 downto 0);
       datox: out std_logic_vector (6 downto 0));
end component;

begin
C01:   mult  port map  (D_dato,"0001101",dato1);
C02:   mult  port map  (D_dato,"0110100",dato2);
C03:   mult  port map  (D_dato,"1101000",dato3);

...
C16:   mult  port map  (D_dato,"1001101",dato16);
process (clk)
...
if (clk'event and clk='1')then
  memoria_v(0) :=memoria_v(1)xor dato1;
  memoria_v(1) :=memoria_v(2)xor dato2;
  memoria_v(2) :=memoria_v(3)xor dato3;
  memoria_v(3) :=memoria_v(4)xor dato4;

...
 memoria_v(15):=dato16;
end if;
if hab='1' then
  salida<=D_in;
else
  salida<=memoria_v(0);
end if;
end process;
end Behavioral;
```

De la descripción en VHDL se puede observar que se ha definido el codificador, basado en componentes: (1) el componente multiplicador (mult), que presenta el circuito de multiplicación en campos finitos, al cual se le ingresan los operandos por los puertos correspondientes a **D_dato**, dato a operar y el coeficiente de *G(X)*, para cada rama, (2) los rgistros definidos como memoria y sus operaciones xor, donde se realiza el corrimiento de los datos resultantes, funcionando como un registro desplazamiento, con la realimentación respectiva. A partir de la descripción VHDL, se puede definir el diagrama para la generación del código (ver figura 4).

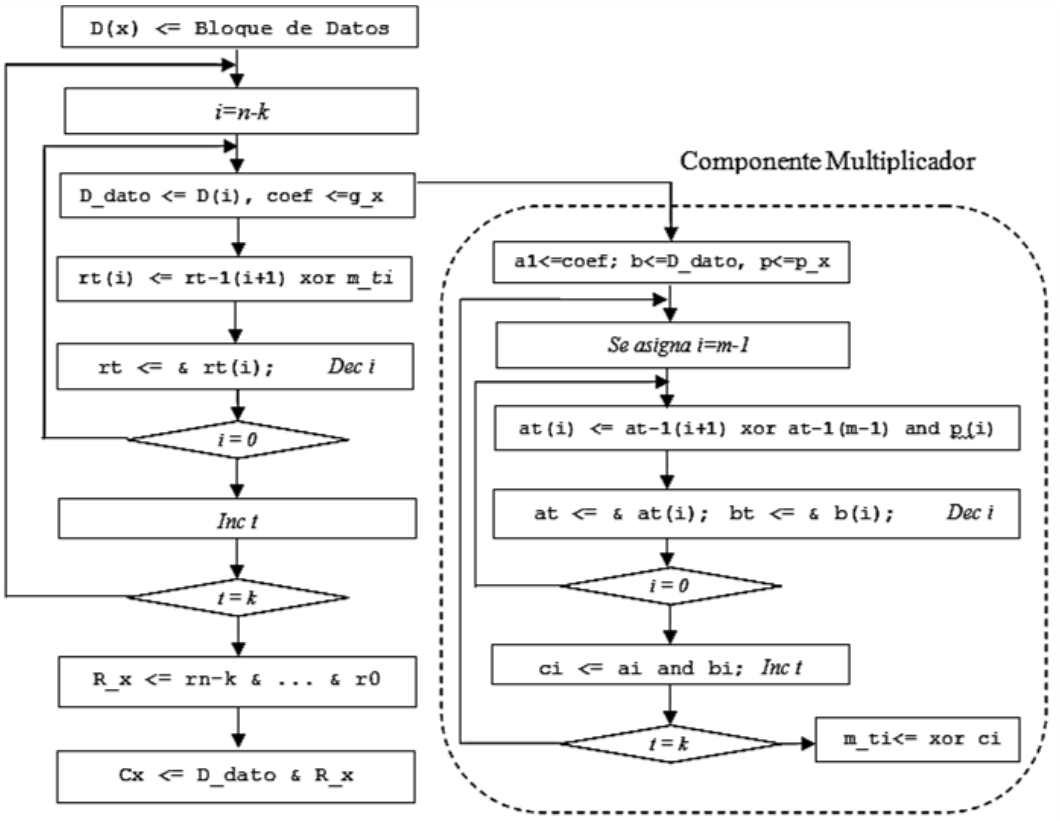

Fig. 4. Esquema del procedimiento para Generación de Código VHDL de un Codificador Reed Solomon (*n*,*k*)

El diagrama ilustra el proceso iterativo para la generación del código en VHDL asociado a la implementación del codificador, el comportamiento en el cual las funciones son implementadas de forma concurrente, a través de la definición del componentes *Multiplicador GF*, que a su vez presenta el mismo comportamiento, con el algoritmo de generación de código para la descripción de hardware propuesto.

El número de iteraciones para generar cada término, que implementa cada sección de hardware ha sido definido a través de los parámetros que son confirmados en los bloques condicionales, mientras las operaciones lógicas son descritas a partir de la descripción VHDL desarrollada.

De esta manera, se han identificado funciones recurrentes en el caso de la aplicación del generador de secuencia LFSR, tanto para el componente del multiplicador en campos finitos de Galois, como para el generador de redundancia del codificador Reed Solomon, haciendo una auto-llamada al módulo LFSR del multiplicador.

4. Resultados

Una vez modelado en VHDL la aplicación y reconocidas las características de las estructuras y su comportamiento, que ha sido tratado en el análisis circuital a través de la disertación, se destaca que la aplicación estudiada presenta: (1) Funciones Iteradas, las cuales se pueden modelar para la implementación concurrente del circuito secuencial, (2) Auto-similitud entre los componentes LFSR de las estructuras del multiplicador GF y del codificador RS, de las mismas característica.

Es de hacer notas que se ha sustituido el circuito del multiplicador (con la etapa de reduccuón modular) en la estructura del generador de código RS, a fin de ilustrar su correspondencia a diferente escala, destacando que los operados binarios del multiplicador son extrapolados en operadores de 8 bits, para la operación de los símbolos, por lo que se puede identificar la posibilidad generalizar la aplicación, como un circuito fractal en VHDL, cuya estructura se presenta en la figura 5.

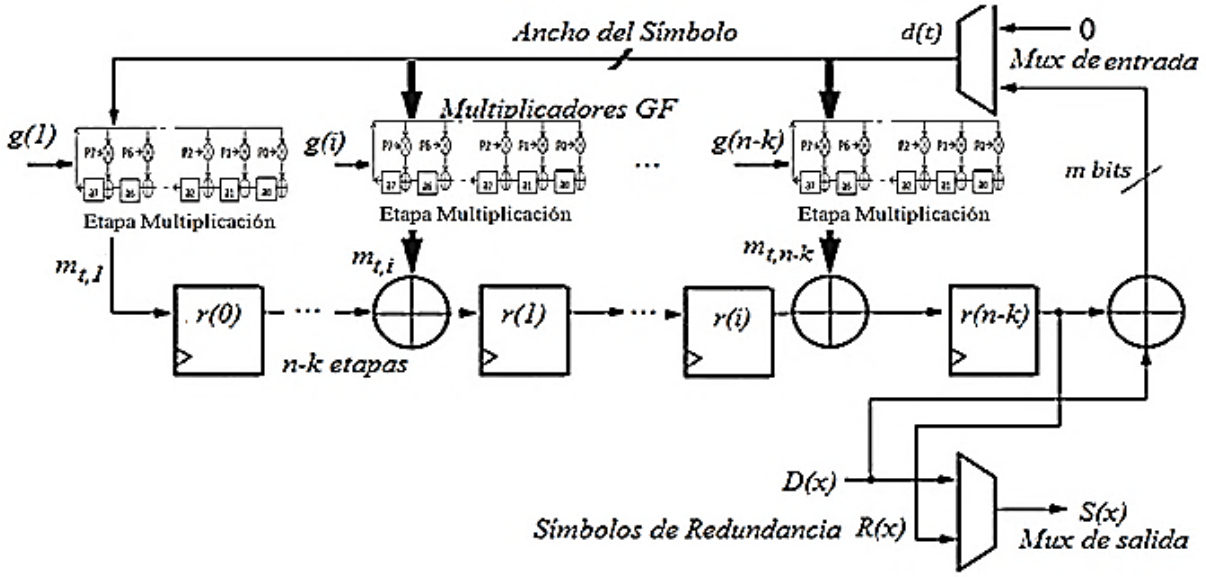

Fig. 5. Modelo Fractal del Circuito de generación del Código RS

A continuación se definen las ecuaciones del sistema de funciones iteradas, que describen el modelo del codificador Reed Solomon, correspondiente a la implementación del circuito LFSR. En la ecuación 3 se tiene la descripción comportamental del componente LFCS (de reducción modular) en el multiplicador de campos finitos GF(2^m), a través de la operación concatenación y los operados lógicos. Esta expresión ha sido obtenida a partir de la descripción VHDL (tabla 3)

$$a_t = \&_{i=0}^{m-1} a_{t-1}(i-1) \oplus (a_{t-1}(m-1) \text{ and } p(i)) \quad (4)$$

Este elemento a_t, operando A reducido al campo F, se opera and con b_t, correspondiente al bit *t* del operando B, siendo *t* el parámetro para identificar cada uno de los bits del símbolo y *p(i)*, el coeficiente *i* del polinomio *P(x)*. Es de hacer notar la naturaleza recursiva de la implementación, ya que para el cálculo de uno de los elementos correspondientes a la salida de la rama en la posición *i*, estará dada por el resultado del término correspondiente *i-1*. Esta característica permite definir los elementos circuitales necesarios para su generación concurrente, a partir de un arreglo combinacional.

En tanto que, para el generador de código RS, se obtiene considerando la ec. 1 y la concatenación de términos generados por la implementación de las ramas del LFSR del codificador descritos en VHDL en la tabla 4.

De esta manera se expresa el modelo del codificador RS, a través de la ecuación 5.

$$r_t = \&_{i=n-k}^{0} r_{t-1}(i) \oplus (d(t) \otimes g(i)) \quad (5)$$

Con r_t, símbolo de redundancia; *d(t)*, símbolo de dato a codificar y *g(i)*, coeficiente en la posición *i* del polinomio *g(x)*. Este modelo ha sido empleando como técnica de paralelización la concatenación de términos, la cual está basada en una descripción tiempo – espacio de las funciones generadoras de los elementos componentes.

La ecuación 4 puede ser sustituida en la ecuación 5, donde se define la función iterada, con respecto al espacio, tal como se observa en la ecuación 6.

$$r_t = \&_{i=n-k}^{0} r_{t-1}(i) \oplus (\&_{i=0}^{m-1} a_{t-1}(i-1) \text{ xor } (a_{t-1}(m-1) \text{ and } p(i) \text{ and } b_t) \quad (6)$$

Para la descripción en VHDL se reconoció la similitud entre las ecuaciones, lo que permite definir un sistema de funciones iteradas – *SFI*, aplicable a la generación del código VHDL, para el diseño en forma concurrente. Los resultados del modelo en forma matricial, pueden ser consultados en el modelo optimizado del codificador Reed-Solomon *(255,k)* en VHDL a través de un LFSR paralelizado (Sandoval 2013).

Este modelo VHDL permite el análisis de consumo de recursos basado en la arquitectura del multiplicador, el cual ha sido paralelizado, a través de la descripción concurrente la etapa de reducción modular. Esto representa un importante avance en el modelo propuesto del multiplicador con el componente LFSR concurrente. Donde se hace notar que los avances en el modelo paralelo del componente, puede ser aplicado para el codificador.

Al analizar la estructura del modelo VHDL, se obtienen los operandos directos asociados y se puede realizar una estimación de recursos de hardware a nivel de compuertas (tabla 5).

Tabla 5. Estimación de Recursos por operaciones

Función / Op.	*AND*	*XOR*
$A(x)x^i \bmod P(x)$	*0*	*p.m-p-2m+2*
rm(x) and B(x)	n^2	*0*
XOR C(x)	*0*	m^2 - *m*
LFCS_Mult.GF	n^2	m^2 + *p.m-p-3m+2*
LFSR_Cod.RS	n^2	*(n-k*b)* (m^2 + *p.m-p-3m+2)*

Donde *m* corresponde al número de bits de cada palabra del campo y *p* corresponde al número de bits no nulos del polinomio irreducible *P(x)*, en este caso la optimización corresponde a la simplificación de operaciones, en función del número de coeficientes *P(x)* no nulos.

Del mismo modo para este modelo se obtuvo el consumo de recursos del FPGA, a través de la herramienta de desarrollo ISE 11, obteniendo el consumo de Slice y LUTs Tables en función del número de bits del multiplicador (ver figura 6).

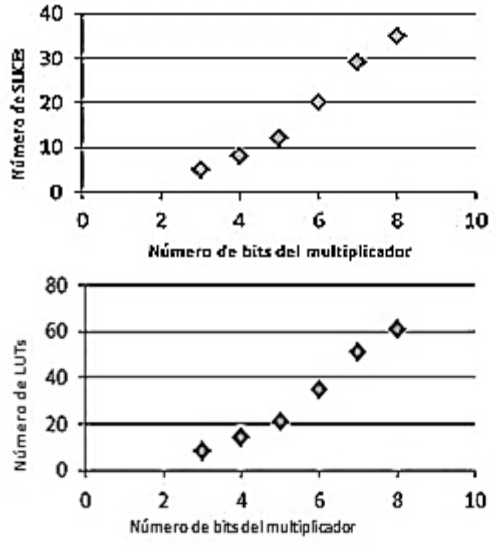

Fig. 6. Consumo de Recursos del Multiplicador por bits de símbolos, implementado sobre Tecnología FPGA

Se presentan en trabajos previos, el contraste entre los resultados de consumo de recursos de hardware, consumo de potencia y retardos asociados a la implementación (Sandoval y col., 2014), detallando la optimización de potencia (Sandoval 2014), el reporte generado para este caso se presenta en la tabla 6.

Tabla 6. Análisis de Consumo de Recursos y Potencia del Codificador Reed Solomon diseñado

Project File:	LFCS.ise	Implementation State:	Placed and Routed
Module Name:	LFCS	▪ Errors:	
Target Device:	xc5vlx30t-3ff323	▪ Warnings:	
Product Version:	ISE 11.1	▪ Routing Results:	All Signals Completely Routed
Design Goal:	Balanced	▪ Timing Constraints:	
Design Strategy:	Xilinx Default (unlocked)	▪ Final Timing Score:	0 (Setup: 0, Hold: 0) (Timing Report)

Device Utilization Summary

Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice LUTs	44	19,200	1%	
Number used as logic	44	19,200	1%	
Number using O6 output only	44			
Number of occupied Slices	17	4,800	1%	
Number of occupied SLICEMs	0	1,280	0%	
Number of LUT Flip Flop pairs used	44			
Number with an unused Flip Flop	44	44	100%	
Number with an unused LUT	0	44	0%	
Number of fully used LUT-FF pairs	0	44	0%	
Number of slice register sites lost to control set restrictions	0	19,200	0%	
Number of bonded IOBs	24	172	13%	
Average Fanout of Non-Clock Nets	3.65			

Name	Value	Used	Total Available	Utilization (%)
Logic	0.00004 (W)	44	19200	0.2
Signals	0.00035 (W)	60	—	—
IOs	0.01682 (W)	24	192	12.5
Total Quiescent Power	0.40164 (W)			
Total Dynamic Power	0.01721 (W)			
Total Power	0.41885 (W)			
Junction Temp	50.0 (degrees C)			

Por otra parte, el análisis de la aplicación permitió establecer la correspondencia entre las estructuras componentes y las funciones que definen su comportamiento matemático, en el procesamiento de datos, según la longitud del elemento a operar, como se presenta en la tabla 7.

Tabla 7. Reconocimiento de estructura circuital fractal

Componente LFSR	Codificador RS	Mult. GF
Elemento de memoria	*Símbolos (m bits)*	*bit*
Operador de ramas	*Mult_GF*	*and*
Expresión de función	*D(i) lfsr G(i)*	*A(i) lfsr P(i)*
Polinomio Generador	*G(x)*	*P(x)*

Finalmente, los reportes obtenidos demuestran la aplicabilidad del modelo iterado para descripción de hardware en el caso de circuitos fractales como el codificador Reed Solomon, el proceso de validación de comportamiento y desempeño fueron analizados anteriormente (Sandoval y col., 2014), destacando que la aplicación del método de diseño resulta eficiente.

Es importante señalar que los resultados presentados corresponden a las expresiones matemáticas de las funciones iteradas que permiten generar el código para la descripción en VHDL, del comportamiento del codificador Reed Solomon y el multiplicador GF, destacando que se encontró la relación de auto-similitud entre las estructuras estudiadas, que solo varía en cuanto a dimensiones de operandos y operadores, siendo que esto no se había encontrado en literatura previa, la cual se puede extrapolar igualmente para codificadores RS paralelos (Sandoval 2017) con n-dimensiones, mostrado en la figura 7**.**

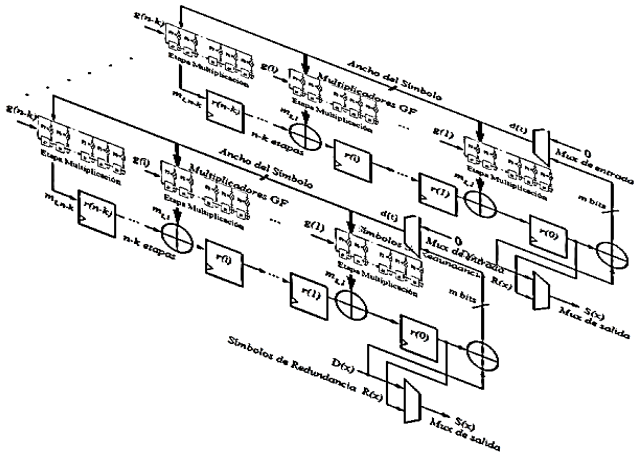

Fig. 7. Esquema del Codificador n-RS concatenado paralelo

Aplicaciones del Modelado con Sistemas de Funciones Iteradas para VHDL

Un área de interés para configurar sistemas con este método e diseño corresponde a los filtros adaptativos (Castellanos y col., 2014), considerando la estructura circuital del filtro transversal que coincide con un LFSR Fibonacci. De manera que el modelo fractal acá desarrollado puede ser aplicado para el modelado SFI de estos sistemas.

Estos avances igualmente pueden ser aplicados para control avanzado reconfigurable, donde se utilicen estructuras similares como es el caso de control adaptativo, con operadores definidos como componentes de aproximaciones sucesivas, los cuales presentan el modelo autosimilar, donde los componentes del sistema de control pueden reconfigurarse de acuerdo a los parámetros definidos por los diseñadores.

Una de las aplicaciones en esta área corresponde a los sistemas eco-adaptativos (Sandoval 2015), estando estos basados en control adaptativo con tecnología FPGA, que pueden configurarse de acuerdo a la dinámica del sistema, dejando que los parámetros se estimen a través de circuitos iterados. De tal manera, que todos los conceptos acá analizados y el modelo desarrollado pueden ser ajustados para obtener los términos requeridos en configuración de diversas aplicaciones hardware, aportando un método de modelado para hardware reconfigurable.

5. Conclusiones

Gracias al análisis del sistema objeto de estudio, se ha podido reconocer una estructura fractal en el circuito del generador de código Reed Solomon, dado por la similitud entre los componentes LFSR, presentes en el generador de redundancia y el componente multiplicador GF. Así mismo, se ha realizado el modelado de la aplicación, alcanzando definir la descripción concurrente en VHDL, del comportamiento de la función LFSR, aplicable para ambos casos, reconociendo una función generadora de términos, que mantiene relación con la secuencia temporal, la cual se modelo de forma concurrente, a partir de un SFI.

Del modelo se obtiene la descripción del hardware bajo un tratamiento fractal-modular, generado por iteraciones para la construcción de los términos en VHDL. La presente investigación incluye nuevos conceptos al área de conocimiento de descripción de hardware. Observando que en los componentes auto-similares se presenta una estructura anidada, donde se reconoce una función iterada. Por otra parte, se analizó la realimentación precalculada para sustituir los ciclos, a fin de definir un tratamiento paralelo.

Igualmente, se pueden aplicar los principios de comportamiento analizados, como base para el diseño de otros sistemas complejos, reconociendo los patrones asociados en la estructura fractal. El principal aporte de este trabajo consiste en aplicar la teoría fractal para desarrollar un modelo de funciones iteradas en el espacio que permite el tratamiento de un sistema recursivo en el tiempo, siendo un enfoque novedoso, que no se ha manejado por la comunidad científica y que presenta múltiples aplicaciones en el área de control y comunicaciones.

Observando las alternativas de co-diseño hardware / software, permite hablar de una correspondencia entre un modelo con funciones iteradas (software) para la generación del código en VHDL para la configuración de estructuras circuitales con características fractales (hardware), como resultado de integrar ambos conceptos en una propuesta innovadora, se ha propuesto avanzar en este método de modelado, para la optimización de los diseños VHDL, el cual puede ser generalizado en aplicaciones con características similares.

Se aportó simplicidad al método de descripción de hardware, lo que permite un ajuste dinámico con gran flexibilidad en su configuración, la propuesta permite el desarrollo del hardware, a partir de las ecuaciones obtenidas y logra que el diseño reutilice los compontes definidos, orientado a la implementación concurrente, dando un nuevo enfoque al diseño libre (basado en ecuaciones

de definición para su implementación), desde un conjunto matemático que permite su configuración en hardware.

Referencias

Adame SEA, 2005, Sistemas de funciones iteradas y los fractales, Fundación Universitaria Konrad Lorenz.

Alaus LA, Oguet DN, Alicot JP, 2008, Extended reconfigurable linear feedback shift register operators for software defined radio. Gestion, Spread Spectrum Techniques and Applications, 2008 IEEE 10th International Symposium, pp. 1–5.

Alvarez I, 2005, Aplicaciones de las Matrices por Bloques a los Criptosistemas de Cifrado en Flujo, Tesis Doctoral. Universidad de Alicante.

Castellanos J, Sandoval C, Azpurua M, 2014, A FPGA implementation of a LMS adaptative algorithm for smart antenna arrays. Revista Técnica de La Facultad de Ingenieria de La Universidad de Zulia, Venezuela, 37(3), pp. 270–278.

Dubrova E, Teslenko M, Tenhunen H, 2008, On Analysis and Synthesis of (n,k) -Non-Linear Feedback Shift Registers. Design, Automation and Test in Europe Transition, pp. 1286–1291.

Ekdahl P, 2003, On LFSR based Stream Ciphers-analysis and design. Department of Information Technology, Lund University, Sweden.

Goresky M, Klapper A, 2004, Fibonacci and Galois representations of feedback with carry shift registers, 1–31.

Goresky M, Klapper AM, 2002, Fibonacci and Galois representations of feedback-with-carry shift registers. IEEE Transactions Information Theory, 48(11), pp. 2826–2836.

Imaña JL, Sánchez J, Fernández M, 2002, Método de multiplicación canónica sobre campos GF (2m) generados por AOPs orientado a hardware reconfigurable. In II Jornadas sobre Computacion Reconfigurable y Aplicaciones JCRA2002.

Kim CH, Oh S, Lim J, 2002, A new hardware architecture for operations in GF (2m). Computers, IEEE Transactions on, 51(1), pp. 90–92.

Lee MYI, Prieto FR, 2010, Generación de Fractales a partir de Transformaciones Afines. In III REPEM - Reunión Pampeana de Educación Matemática pp. 520–530.

Magaña del Toro R, Hermosillo-Arteaga AR, Romo-Organista MP, Carrera-Bolaños J, 2011, Análisis con elemento finito y remalleo fractal en geotecnia Finite Element Analysis and Fractal Remeshing in Geotechnics. Ingeniería, Investigación Y Tecnología, XII(I), pp. 103–118.

Molero J, 2011, Conjunto de Julia y Mandelbrot. Sistenas de Funciones Iteradas. Aplicaciones, Universidad Politécnica de Cartagena, España.

Moon TK, 2005, Error Correction Coding: Mathematical Methods and Algorithms. (Wiley, Ed.), Moon. N.J., USA: Wiley.

Pérez M, 2009, Generación y Correlación Eficiente de Códigos Binarios derivados de conjuntos de Secuencias Complementarias para sistemas ultrasónicos, Tesis Doctoral. Universidad de Alcalá, España.

Ramírez JL, Rubiano GN, 2012, Generación de curvas fractales a partir de homomorfismos entre lenguajes. Revista Integración, 30(2), pp.129–150.

Rivera E, López HR, 2012, Evidencia de propiedades fractales en la sucesión de Fibonacci usando wavelets. Scientia et Technica, 17(52), pp.122–128.

Sandoval-Ruiz C, 2012, Codificador RS (n,k) basado en LFCS : caso de estudio RS (7,3). Rev. Fac. Ing. Univ. Antioquia, (64), pp. 68–78.

Sandoval-Ruiz C, 2013, Modelo Optimizado del Codificador Reed-Solomon (255,k) en VHDL a través de un LFSR paralelizado. Tesis Doctoral, Dirección de Postgrado, Universidad de Carabobo, Venezuela.

Sandoval-Ruiz C, 2015, Sistema Eco-Adaptativo integrado en elementos arquitectónicos con tecnología sostenible. Revista Electrónica Científica Perspectiva, 8(4), pp. 96–109.

Sandoval-Ruiz CE, 2017, Logical-Mathematical Model of Encoder 2D-RS for Hardware Description in VHDL. Revista Ingeniería UC, 24(1), pp. 28–39.

Sandoval-Ruiz, CE, Fedón-Rovira A, 2013, Codificador RS (255,k) en hardware reconfigurable orientado a radio cognitivo. Ingeniería y Universidad, 17(1), pp. 77–91.

Sandoval-Ruiz C, Fedón-Rovira A, 2013, Modelo fractal de un codificador Reed Solomon. VIII Congreso Nacional y 2do Congreso Internacional de La UC, pp. 1–12.

Sandoval-Ruiz C, Fedón-Rovira A, 2014, Efficient RS (255,k) encoder over reconfigurable systems. Rev.Téc.Ing.Zulia, 37(2), pp. 151–159.

Sandoval-Ruiz C, 2007, Diseño Modular de un Sistema para Procesamiento y Comunicación Digital en Banda Base usando Programación en VHDL. Trabajo de Grado, Universidad de Carabobo, Venezuela.

Sandoval-Ruiz C, 2010, Multiplicador paralelo en campos finitos de Galois GF (2m). Congreso de Investigación UC, (1), pp. 1706–1711.

Sandoval-Ruiz C, Fedón A, 2008, Programación VHDL de algoritmos de codificación para dispositivos de hardware reconfigurable. Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería *(RIMNI)*, 24(1), pp. 3–11.

Sandoval-Ruiz C, 2014, Power Consumption Optimization in Reed Solomon Encoders over FPGA. Latin American Applied Research, 44(1), pp. 81–85.

Saqib A, 2004, Implementación Eficiente de Algoritmos Criptográficos en Dispositivos de Hardware Reconfigurable, Tesis Doctoral, Cinvestav del Instituto Politécnico Nacional, México.

Xilinx, 2011 DS251, LogiCORE IP Reed-Solomon Encoder v7.1 Data Sheet. Cadence, pp. 1–16.

Recibido: *2 de noviembre de 2016*

Aceptado: *28 de noviembre de 2016*

Sandoval-Ruiz, Cecilia E. *Profesora en Postgrado de Ingeniería UC, egresada de la Universidad de Carabobo de Ingeniero Electricista en 2002, Magister en Ingeniería Eléctrica en 2007 y Doctora en Ingeniería en 2014. Investigadora acreditada en el PEII - Nivel C, áreas de investigación: diseño sostenible y configuración de hardware. cecisandova@yahoo.com*