Rev. Téc. Ing. Univ. Zulia. Vol. 37, Nº 2, 151 - 159, 2014

Efficient RS (255, k) encoder over reconfigurable systems

Cecilia Esperanza Sandoval Ruiz, Antonio S. Fedón Rovira

Facultad de Ingeniería, Universidad de Carabobo/Dirección de Postgrado. Naguanagua sector Bárbula, Carabobo, Venezuela. Teléfonos (0241) 8672829 /8674268 ext. 102. Fax: (0241) 8671655csandoval1@uc.edu.ve, afedon@uc.edu.ve

Abstract

This study is based on obtaining an optimized model for the Reed Solomon encoder design and efficiency analysis. First comes the introduction of the mathematical model of the encoder, followed by a description of the object of study encoder RS (255, k) in hardware description language, VHDL (hardware description language VHSIC) with the functional description of each one of its components, including the multiplier finite Galois field GF (2^m) and LFSR (Linear Feedback Shift Register) based on the equations as a result of analyzing the structure and behavior of the encoder optimized finally synthesized the proposed encoder IDE11 Xilinx tool. Finishing, with the comparative analysis of resource consumption, validation of the proposed optimization model. Among the results are the mathematical equations were derived for efficient encoder over reconfigurable systems, validating the behavior of simulated design and synthesis that demonstrate the effectiveness of the design compared to other studies. Thus we can conclude that it was parallel processing of multiple components and saves hardware resources in the system through the proposed model.

Keyword: efficient encoder over reconfigurable systems, VHDL, FPGA, Reed-Solomon (255,k).

CESR-Codificador RS (*255,k*) eficiente para sistemas reconfigurables

Resumen

La presente investigación está basada en la obtención de un modelo optimizado para el diseño del Codificador RS-Reed Solomon, orientado a sistemas reconfigurables. En primer lugar se realizó la descripción del codificador objeto de estudio RS*(255,k)*, bajo Lenguaje Descriptor de Hardware, VHDL (*VHSIC hardware description language)*, con la descripción funcional de cada uno de sus componentes, destacando entre ellos el multiplicador en campos finitos de Galois - GF(2^m) y el LFSR (Linear Feedback Shift Register), analizando su estructura y comportamiento para hallar un modelo del Codificador optimizado, finalmente, se sintetizó el diseño del codificador a través del IDE Xilinx 11, para el análisis comparativo del consumo de recursos Hardware, validando la optimización del modelo propuesto. Entre los resultados podemos mencionar que, se obtuvieron las ecuaciones matemáticas que soportan el modelo del CESR-Codificador Eficiente para Sistemas Reconfigurables, la validación del comportamiento del diseño simulado y los reportes de síntesis que evidencian la eficiencia del diseño respecto a otros estudios, de lo que podemos concluir que se alcanzó un procesamiento paralelo del componente multiplicador y un ahorro de recursos de hardware en el sistema.

Palabras clave: VHDL, hardware re-configurable, codificadores, comunicación digital.

1. Introducción

Desarrollos actuales de codificadores Reed Solomon –RS(n,k)– presentan sus avances en diseños cada vez más eficientes, a través del reporte de síntesis de los módulos, para la implementación sobre hardware, tal es el caso de los *IPcores* de las casas fabricantes de dispositivos FPGA [1-3], al igual que los presentados por investigaciones recientes en el campo de optimización de codificadores Reed Solomon [4-6], siendo el objetivo común, ofrecer a la comunidad científica modelos de codificadores Reed Solomon con mejores prestaciones.

Inicialmente, se pueden considerar diversos aspectos para la optimización del diseño de codificadores, como pueden ser, la velocidad de respuesta, el consumo de recursos hardware y consumo de potencia. Es importante resaltar que la potencia consumida por el dispositivo está altamente relacionada con la ocupación de área del dispositivo FPGA [7]. Por otra parte, un factor de interés corresponde a la implementación eficiente de la aritmética en campos finitos, la cual es medida en términos asociados a la complejidad espacial y temporal de la implementación, donde la complejidad espacial es definida por la ocupación de recursos de hardware, para la implementación del circuito y la complejidad temporal medida a través de los retardos totales de la respuesta del circuito diseñado [6].

Observando estos aspectos, es necesario el estudio de los modelos matemáticos de los componentes del codificador, a fin de optimizar la descripción VHDL de los componentes del codificador RS, como lo son el multiplicador en campos finitos de Galois-GF (*del Inglés, Galois Field*) y el LFSR (Linear Feedback Shift Register), para lo cual resulta importante analizar su estructura y comportamiento, generando un modelo concurrente en VHDL de la arquitectura del multiplicador, a fin de implementarlo de forma paralela, y así hallar un modelo del codificador RS con mayor velocidad de procesamiento.

Se presenta como objetivo de esta investigación el diseño de codificadores eficientes orientados a sistemas reconfigurables, a los cuales denominaremos CESR, considerando como caso de estudio la gama de codificadores RS(255,k), para así realizar un estudio comparativo de eficiencia entre los CESR desarrollados y los codificadores previos.

En tal sentido, es importante mencionar que para la optimización del consumo de potencia, se deben considerar dos clases de consumo en los dispositivos: ***consumo de potencia estática***, la cual está relacionada con la fabricación del dispositivo, sobre la cual el fabricante se mantiene desarrollando mejoras referentes a la tecnología [8] y ***consumo de potencia dinámica***, asociada al proceso de diseño, de acuerdo al nivel de abstracción en el diseño, la cual puede ser manejada por alguna herramienta como *ISE Design Tool Automated Intelligent Clock Gating* [8], o directamente por el diseñador, a través de métodos como: aplicación de pipeline, equilibrar los retardos de los caminos, distribución de tablas de búsqueda LUTs (*del Inglés, Look Up Table*), entre otras [9]. Los diseñadores de sistemas basados en FPGAs cuentan con herramientas para optimizar los circuitos en área y velocidad, a través de los IDE (*Integrated Development Environment*) proporcionados por los fabricantes de FPGAs, sin embargo no proveen software de diseño para bajo consumo de potencia [10]. En tal sentido, el presente diseño estará orientado fundamentalmente a la optimización del consumo del área del dispositivo y velocidad, lo cual está relacionado con la optimización de potencia, esto basado en las mediciones obtenidas por Angarita, donde concluyen que el consumo total es menor en la versión paralela [11]. En función de estos datos se seleccionó para este trabajo una optimización a través de paralelización de los componentes del codificador RS, específicamente los multiplicadores de campos finitos de Galois GF(2^m) y una descripción en VHDL uniforme y simplificada, a fin de disminuir el consumo de recursos de hardware, para lo cual se generó un conjunto de ecuaciones matemáticas que definen la arquitectura concurrente de los componentes multiplicadores para su descripción en VHDL, haciendo uso de una filosofía de diseño modular.

2. Códigos Reed Solomon y sus implementaciones

Los codificadores Reed-Solomon, desarrollados por los ingenieros-matemáticos Irving S. Reed y Gustave Solomon, realizan la codificación a través de un procesado sobre bloques de datos, estos demandan una gran capacidad de cómputo en su implementación, a fin de obtener una alta

velocidad de procesamiento. De manera tal que la velocidad de respuesta en la salida del codificador siempre estará limitada por el hardware sobre el cual se implementa el circuito, sus operaciones se realizan sobre algebra de campos finitos, resultando de interés la implementación concurrente sobre hardware de estos codificadores.

Para modelar el sistema, resulta importante estudiar el comportamiento del codificador RS(n,k), donde n es el número de símbolos del código y k, es el número de símbolos de datos, el codificador Reed Solomon es un elemento encargado de construir la palabra de código $C(x)$, de n símbolos de m bits cada uno, la cual genera a través de un circuito asociado con el polinomio generador $G(x)$, estos polinomios estandarizados son presentados de acuerdo a los parámetros n y k, y la palabra de datos $D(x)$, su representación matemática está dada por: $C(x) = G(x) * D(x)$. De este modo, la palabra codificada corresponde a los k símbolos de datos, de $D(x)$, concatenados con n-k símbolos de redundancia. Así se puede observar que la principal función del codificador consiste en generar los símbolos de redundancia, a través de una arquitectura específica.

Estos codificadores dadas sus características de desempeño son frecuentes en implementaciones, tales como; Digital Video Broadcast (DVB). Digital Satellite Broadcast (ETS 300-421satellite, ETS 300-429 cable), Intelsat Earth Stations (IESS-308), Space Telemetry Systems (CCSDS,. ADSL Transceivers (ITU G.992.1), Wireless Broadband Systems (IEEE 802.16), 2.5G, 10G AND 40G Optical Networks (ITUT G.795).

Teoría del modelo matemático del Codificador RS(*255,k*)

En esta sección se presenta el estudio del codificador Reed Solomon, identificando en su arquitectura componentes para su descripción modular, en primer lugar la arquitectura del codificador está basada en una estructura LFSR y multiplicadores GF, los cuales pueden ser implementados a través de un modelo algorítmico, el cual reproduce el comportamiento del multiplicador de forma secuencial [12] o usando modelos paralelos, entre los cuales se encuentra el modelo del multiplicador propuesto por Karatsuba-Ofman [13] para polinomios en GF(2m), el cual consiste en una técnica de sub-división modular del componente multiplicador, modelos combinacionales [14-19] y uso optimizado de tablas [6], la cual presenta un aporte en reducción de recursos en base a la implementación original de tablas de 255×255, equivalente a 65025 símbolos de 8 bits, a una reducción a 256 símbolos de 8 bits para el multiplicador, sin embargo esta reducción sigue demandando 512LUTs de 4 entradas al hacer la expansión para cubrir la tabla mencionada.

Los investigadores en el área de codificación, criptografía y otras aplicaciones, buscan disminuir el tiempo de procesamiento y el costo computacional de los multiplicadores GF [20, 21]. De lo que encontramos que para obtener mejor rendimiento en nuestra aplicación del codificador RS, se debe solventar el problema de establecer un modelo eficiente orientado a sistemas reconfigurables. Para la implementación de los multiplicadores, se estudió el modelo matemático de la teoría de multiplicación en campos finitos, lo cual sirvió de fundamento en el desarrollo del modelo circuital propuesto.

Multiplicación. 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 operación módulo del polinomio $P(x)$, *también conocido* como reducción modular, es el mostrado en la ecuación 1.

$$C(x) = A(x).B(x) \Leftrightarrow C(x) = \mathrm{A}(x) \times B(x) \ (mod\, p\,(x)) \quad (1)$$

La multiplicación de polinomios es asociativa, conmutativa y distributiva con respecto a la adicción por lo cual se obtienen las ecuaciones 2.

$$C(x) = B(x)(\sum_{i=0}^{m-1} A_i x^i) \bmod p(x) \rightarrow$$

$$C(x) = \sum_{i=0}^{m-1} B_i\,(A(x)x^i) \bmod p(x) \quad (2)$$

donde $A(x)$ corresponde en el codificador RS 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, entiéndase símbolos de información a la entrada del codificador. La ecuación 2, será descrita en VHDL para su representación circuital, definien-

do los componentes para la implementación de los correspondientes factores, donde la reducción modular, *A(x) mod p(x)*, será realizada bajo la arquitectura de un circuito LFSR, como el mostrado en la Figura 1.

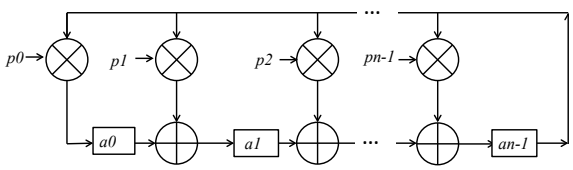

Figura 1. Arquitectura del módulo LFSR para implementación de *A(x) mod p(x).*

3. Implementación propuesta del codificador Reed Solomon

El estudio detallado de la arquitectura del multiplicador [22], originalmente implementado a través de registros para obtener los *m* vectores, a través de corrimiento de los bits, es el fundamento para el análisis del comportamiento del circuito en *m* ciclos de reloj, donde *m* es igual al número de bits de la palabra, esto permite la obtención del modelo matemático genérico para definir el comportamiento del multiplicador sobre campo finito [23], para su descripción en VHDL. Con base a este modelo se logra una versión paralelizada del multiplicador, para la cual se han eliminado los componentes secuenciales y estos han sido remplazados por la operación de concatenación que permite una versión concurrente del circuito.

A partir de la Figura 1 se puede definir el cálculo de cada uno de los términos del vector *a* en función de la posición *i,* para un instante *t*, cada término está dado por la función lógica definida por $a_{t-1}(i-1)\,xor\,(a_{t-1}(m-1)\,and\,p(i))$ para un instante *t-1*, donde el vector corresponde a la concatenación de los *m* términos, obteniendo así la ecuación 3.

$$A(x)_t = \&_{i=0}^{m-1} a_{t-1}(i-1)\,xor\,(a_{t-1}(m-1)\,and\,p(i)) \quad (3)$$

Fuente: Propia del autor.

Siendo $A(x)_t$, un elemento generado de aplicar al residuo, en instante *t*, de la reducción modular dada por *A(x) mod p(x)*, éste se obtiene a partir de la concatenación de la operación lógica AND entre el coeficiente *p*, en la posición *i*, y el elemento a_{t-1} en la posición más significativa del símbolo actual, este operado a través de una *XOR* (*OR-exclusica*) con a_{t-1}, en la posición *i-1*. A partir de la ecuación 3, se ha calculado en función del número de bits del símbolo el número de compuertas, resultando el término m^2-2m+1 de compuertas AND y *XOR*, para generar los *m* elementos.

Por otra parte, el producto de *(A(x) mod p(x))*B(x)*, corresponde a la operación AND de cada elemento de *B(x)* por cada a_t y la sumatoria *XOR* de estos, de acuerdo al circuito multiplicador [24], de donde se obtiene la ecuación 4.

$$c = \oplus_{t=1}^{m} a_t\,and\,b_i \quad (4)$$

Fuente: Propia del autor

donde se puede deducir que se requieren *(m-1)*(m-1)* compuertas AND y *(m-1)* compuertas *XOR*, de lo cual se tiene un total de compuertas para el multiplicador general. Una vez asignado el polinomio generador del campo *P(x)*, se puede simplificar la ecuación 3, particularizando la ecuación de consumo de recursos en función *p* el número de coeficientes no nulos del polinomio *p(x)*. De allí que, tomando el polinomio generador del campo *P(x)=100011101*, se obtiene el modelo particularizado, mostrado en la Figura 2.

Este circuito simplificado fue implementado en el componente de reducción modular, para la descripción del multiplicador en campos finitos, siendo un importante aporte en la optimización basado en el modelo matemático-teórico.

Se realizó el diseño modular de las etapas del codificador RS(*n,k*) de longitud ajustable, a

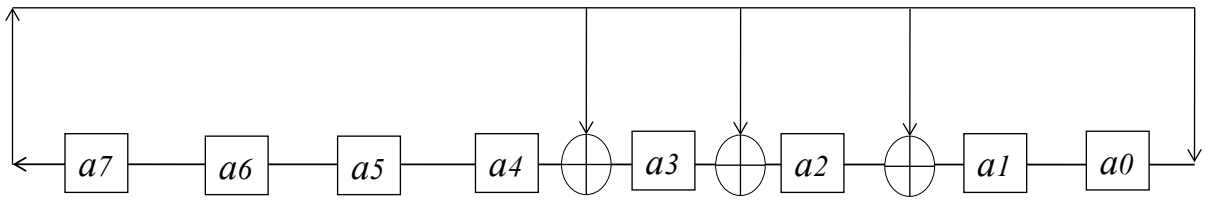

Figura 2. Modelo del LFSR para p(x)=285.

través del parámetro *k*. Para obtener los coeficientes del polinomio generador, se fijó el valor de *k* para cada caso particular del codificador RS(255,k), tomando los valores k=247, 239 y 223 respectivamente.

El primer paso considerado fue definir las características del código RS(*n*,*k*) con *b*-bit por símbolos y con *n* bytes por cada palabra de código (*k* de los cuales son bytes de datos y *n*-*k* son bytes de paridad). Para la programación se empleó el software ISE 11 de Xilinx, se inició por la definición de la entidad para los módulos RS con longitud ajustable, usando lenguaje descriptor de hardware VHDL. La descripción consistió en la aplicación del modelo propuesto, donde el número de etapas depende del valor *k* seleccionado, el cual se declaró de forma genérica, por tanto se definió *length* como un parámetro ajustable en función de *n*-*k*.

Por otra parte, se definió el número de bits por símbolo como *width*. Seguidamente, en la arquitectura del codificador se definió una memoria FIFO de *length* etapas de *b* bits de ancho (*width*) cada una. La entrada *D_dato* fue definida como un vector de *width-1* a *0* y fue descrito un circuito para la operación entre las etapas de la memoria FIFO definida de *0* a *length-1*. Finalmente, se definió el componente *mult* módulo básico en el *codificador_RS*, que procesa la entrada *D_dato* con el coeficiente del polinomio generador *p(x)*, obteniéndose a la salida el producto en *datox*, la sintaxis en VHDL se muestra en la Tabla 1.

Una vez definida la arquitectura y los componentes, se procesan las salidas de los componentes *mult*, donde se presenta la descripción del comportamiento del componente de reducción modular, para obtener los vectores *a1....a8*, resultantes como residuos parciales. La implementación lógica se reduce al uso de compuertas *AND* entre cada coeficiente de *p(x)* con la realimentación lineal del término más significativo del vector *a(m-1)*, y el uso de compuertas *XOR* que opera con los resultados de los coeficientes parciales, todo esto en el mismo pulso de reloj. La descripción en VHDL del modelo circuital concurrente, que emplea el multiplicador para el CESR, está basada en la ecuación 3, al aplicarla se obtiene la configuración presentada en la Tabla 2.

Como se observa en la Tabla 2 el circuito puede ser simplificado con la cancelación directa de las ramas cuyos coeficientes son "0" en el polinomio irreducible. El último paso para obtener el producto corresponde a la multiplicación de los vectores parciales obtenidos, donde *i* es el sub-índice para identificar cada uno de los bits del polinomio *B(x)*, de este modo se generan los productos parciales "c_i". Los productos necesitan implementarse con *AND* entre las "a_i" obtenidas (un vector de *m* elementos) con el bit correspondiente del dato de entrada "b_i", en forma paralela bit a bit, por lo cual es definido un vector *Bi* (igual a la concatenación del elemento *bi* *m* veces). Este procesamiento obedece al modelo descrito en la ecuación 4, una vez aplicada se obtien el código presentado en la Tabla 3, donde se observa la descripción VHDL de cada uno de los vectores resultantes c_i y la sumatorio XOR de estos productos parciales C definida como *datox*.

Seguidamente, se describió en VHDL el circuito LFSR (Linear Feedback Shift Register)

Tabla 1
Descripción en VHDL del Codificador_RS

```
generic(length: integer:= 16; --length n-k
width:integer:=8); -- width of the symbol
Architecture Behavioral of Codificador_RS is
type memoria is array (0 to length-1) of std_logic_vector(width-1 downto 0);
component mult is
port (D_dato: in std_logic_vector (width-1 downto 0);
      coef: in std_logic_vector (width-1 downto 0);
      datox: out std_logic_vector (width-1 downto 0));
end component;
C1: mult port map (D_dato,coef1,dato1);
C2: mult port map (D_dato,coef2,dato1);
C3: mult port map (D_dato,coef3,dato2);
```

Tabla 2
Descripción VHDL concurrente del Componente Divisor basado en LFSR

```
p<="100011101"; -- Primitive polynomial = D^8+D^4+D^3+D^2+1 (285)
u1: a2<=a1(6 downto 4)&(a1(3)XOR a1(7))&(a1(2)XOR a1(7))&(a1(1)XOR a1(7))&a1(0)& a1(7);
u2: a3<=a2(6 downto 4)&(a2(3)XOR a2(7))&(a2(2)XOR a2(7))&(a2(1)XOR a2(7))&a2(0)& a2(7);
...
u7: a8<=a7(6 downto 4)&(a7(3)XOR a7(7))&(a7(2)XOR a7(7))&(a7(1)XOR a7(7))&a7(0)& a7(7);
```

Tabla 3
Descripción del Código VHDL del Producto *A(x).B(x) mod p(x)*

```
B1<= b1(0) & b1(0) & b1(0) & b1(0) & b1(0) & b1(0) & b1(0) & b1(0);
c1<=a1 AND b1;
datox <=c1 XOR c2 XOR c3 XOR c4 XOR c5 XOR c6 XOR c7 XOR c8;
```

del codificador, con *n-k* etapas, estas etapas fueron representadas por elementos de memoria *memoria_v(i).* Las salidas del circuito LFSR que serán transmitidas por el canal de comunicación, usará un multiplexor para seleccionar las *k* palabras de datos y las *n-k* palabras de redundancia, de acuerdo a la señal de control *hab*, siendo este circuito de comportamiento secuencial, por contar con registros para almacenamiento de datos accionados con un *clk*, comportamiento descrito en la Tabla 4.

Adicionalmente, se propone el empleo del *clock_gating*, técnica empleada para el control de la señal de reloj [4], para la disminución de transiciones innecesarias y así optimizar el consumo de potencia lo que corresponde a colocar en la condición (*clk'event AND clk='1'*) AND *h_clk='1'*, con lo cual la señal de reloj depende de la habilitación de reloj *h_clk*.

Finalmente, se crearon bancos de prueba para verificar el funcionamiento del multiplicador, validando su comportamiento y obteniendo las salidas esperadas en los símbolos de redundancia generados, usando ModelSin XE III 6.3c para probar la simulación de cada módulo diseñado.

Tabla 4
Descripción del LFSR del codificador RS(255, k)

```
process (clk)
variable memoria_v:memoria:
=(others=>"0000000");
begin
if hab='1' then
D_dato<= memoria_v(0) XOR D_in;
else
D_dato<="0000000";
end if;
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;
-- dado por la longitud del LFSR
memoria_v(15):=dato16;
end if;
if hab='1' then
salida<=D_in;
else
salida<=memoria_v(0);
end if;
end process;
end Behavioral;
```

4. Resultados

En la simulación, se obtienen los resultados de uno de los codificadores implementados, RS(255,223), donde se puede observar que el comportamiento del circuito coindice con el esperado teóricamente, redundancia_code=[173 69 254 212 67 87 70 169 130 39 34 115 90 135 70 219 177 10 253 16 80 113 13 233 41 145 93 81 208 213 106 197], tal como se presenta en la Figura 3.

Validado el desempeño funcional, se presenta la comparación de consumo de recursos, donde la simplificación desarrollada al definir el polinomio generador del campo *p(x)*, permitió re-

ducir la complejidad del circuito significativamente, quedando el número de compuertas en función del número de bits del campo *m* y número de elementos no nulos del polinomio generador *p*. La comparación entre ambos modelos es presentada en la Tabla 5.

En [4] se presenta un análisis de los modelos previos de multiplicadores GF, estos han sido comparados con los resultados del multiplicador paralelo desarrollado en esta investigación, resumiendo los resultados en la Tabla 6, a nivel de compuertas, y en la Tabla 7, a nivel de recursos de hardware del FPGA, para los parámetros *m=8* y *p=5*, donde se puede observar, que se logró optimizar el consumo de recursos hardware del multiplicador. En el diseño del código eficiente para sistemas reconfigurables CESR, los resultados de la síntesis del codificador RS [26] con el modelo de LFSR concurrente se resumen en la Tabla 8.

De estos resultados podemos observar una optimización en consumo de recursos del hardware del FPGA. Tenemos el consumo de recursos menor al de los IPCores, 2011 [1-3], lo que se puede interpretar como un avance significativo en la

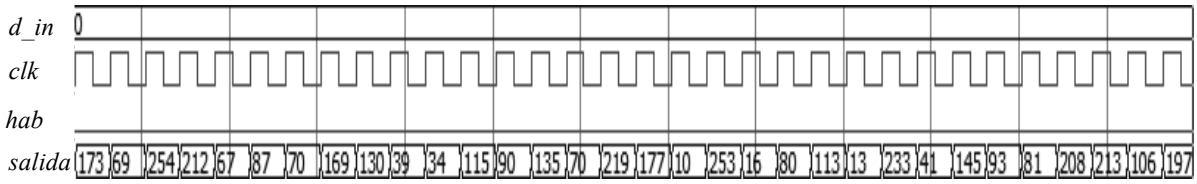

Figura 3. Simulación del RS(255,223) con tiempo total de t=2650.178ns.

Tabla 5
Compuertas utilizadas para el Multiplicador Concurrente

Compuertas del LFSR	AND2	XOR2	AND8	XOR8
Multiplicador general	m^2-2m+1	m^2-2m+1	n	m-1
Multiplicador del CESR	0	pm-p-2m+2	n	m-1

Tabla 6
Compuertas empleadas por el multiplicador GF(2^m) con m=8

Compuertas	AND2	XOR2	8bit XOR8
Mastrovito [14-15]	64	84	0
Paar [16-17]	48	62	0
Karatsuba Clásicos [13]	48	59	0
Multiplicador del CESR	64	21	7

Tabla 7
Recursos empleados por el multiplicador GF(256)

Componentes	Slice LUTs	LUTs	Reg. FF	Consumo de Potencia (mW)
Ahlquist [18-19]	53	-	-	-
Mastrovito [4,25]	64	58	-	2,75
Paar [4,25]	48	53	-	3,54
Multiplicador del CESR	17	44	-	0,39 [1,2]

[1]Se tomó para la comparación el polinomio irreducible P(x)=100011101.

[2]Consumo de Potencia Dinámica del diseño.

Tabla 8
Recursos empleados RS(n,k)

Componentes	n	k	CLB / Slice	LUTs	Reg. FF	$F_{máx}$ (MHz)	Delay (ns)
Altera 2.5G [1]	255	239	374	40	186	>160	-
Lattice DVB [2]	-	-	129	254	201	400	-
Xilinx G.709 [3]	255	239	195	192	186	441/596	-
Xilinx ETSI [3]	255	239	199	194	186	442/580	-
Xilinx CCSDS [3]	255	223	337	329	316	245/316	-
Gianni [5]	255	239	-	353	170	173	5,766
CESR RS-239	255	239	128	159	128	426,457	3,897
CESR RS-223	255	223	256	305	256	418,712	3,811
CESR RS-247	255	247	64	142	64	429,646	3,883

Fuente: Altera, Lattice, Xilinx, autor 2011.

optimización bajo el concepto y modelo estudiado para la implementación.

5. Conclusiones

Se ha analizado la simplicidad algorítmica del divisor polinómico secuencial, los cuales presentan resultados satisfactorios en área, pero con costo en tiempo de respuesta debido a las características secuenciales de éste, simultáneamente se han desarrollado multiplicadores paralelos basados en tablas ó en circuitos combinacionales, siendo eficientes en tiempo pero con un mayor consumo de hardware. Es en tal sentido que se ha propuesto avanzar en esta optimización considerando un modelo híbrido concurrente, a fin de crear un codificador altamente paralelo usando el menor número de recursos hardware, teniendo como referencia los reportes de síntesis de los modelos diseñados en las aplicaciones de propiedad intelectual de los fabricantes de dispositivos FPGA e investigaciones afines.

Se obtuvieron las ecuaciones generales que describen el código, con parámetros simplificables y adaptables para los casos particulares, ya que se determinó que los parámetros una vez asignados permiten optimizar el diseño en gran manera, motivo por el cual los resultados reales estarán dados en función de los coeficientes que definen el polinomio generador del código, por tratarse de coeficientes constantes para un RS dado, podemos disminuir el consumo de recursos de hardware en el código RS, se logró, igualmente, la descripción hardware de módulos genéricos de interés y finalmente el código VHDL de los componentes del código CESR, para su implementación en Hardware libre, es decir, que puede ser adaptado para nuevos diseños, ya que permite extender el modelo acá propuesto para una versión de codificador Reed Solomon paralelizado.

Gracias a la descripción VHDL del codificador Reed-Solomon propuesto, empleando el multiplicador concurrente de acuerdo al modelo circuital desarrollado, se logró obtener un codificador eficiente, que considera de forma equitativa la reducción de consumo de hardware, consumo de potencia y tiempo de procesamiento, a través del diseño del Código Eficiente sobre Sistema Reconfigurable y se han desarrollado los conceptos y estructuras de soporte para su óptima implementación.

Referencias bibliográficas

1. Altera, “RS Encoder” (2011).
2. Lattice, “RS Encoder” (2011).
3. Xilinx, “IPCore RS Encoder” (2011).
4. Allen J, “Energy Efficient Adaptive Reed-Solomon Decoding System”, Thesis, University of Massachusetts, February, 2008.
5. Gianni P., Di Claudio, G., Corteggiano, F., Del Barco, M., “Implementación en FPGA de un Código Reed Solomon RS(255,239)”, Actas de la Escuela Argentina de Microelectrónica, Tecnología y Aplicaciones, (2007) 77-81.
6. Almeida, G. “A Reed Solomon algorithm for FPGA area optimization in space applications” Proceeding AHS ‘07 Proceedings of the

Second NASA/ESA Conference on Adaptive Hardware AND Systems IEEE Computer Society Washington, DC, USA (2007) 243-249.

7. Nazar A., Saqib, "Implementación eficiente de algoritmos criptográficos en dispositivos de hardware reconfigurable", Tesis Doctoral, (CINVESTAV) del Instituto Politécnico Nacional (IPN), Unidad Zacatenco, México, 2004.
8. Jameel Hussein, Matt Klein, AND Michael Hart, "Lowering Power at 28 nm with Xilinx 7 Series FPGAs", WP389, Vol. Nº 1, June 13(2011).
9. Sutter, G. y Boemo, E., "Experiments in low power FPGA design". Lat. Am. appl. res., Vol.37, Nº1, (2007) 99-104.
10. Todorovich, E., Sutter, G., Acosta, N., Boemo, E. y Lopez-Buedo, S., "Relación entre Velocidad y Consumo en FPGAs", Proc. Iberchip 2001 Workshop, Montevideo, Uruguay, (2001).
11. Angarita, F., Marin-Roig, J., Todorovich, E., AND Boemo, E., "Relación área-potencia en la implementación con aritmética distribuida de un Filtro FIR en FPGA", Proc. JCRA 2005, ISBN: 84-9732-439-0, Granada, Septiembre (2005) 59-63.
12. Cruz, J., "Multiplicación Escalar en Curvas de Koblitz: Arquitectura en Hardware Reconfigurable", tesis de Maestría, 2005.
13. Machhout M., Zeghid M., El hadj youssef W., Bouallegue B., Baganne A., AND Tourki, R., "Eficient large numbers karatsuba-ofman multiplier designs for embedded systems". International Journal of Electronics, Circuits AND Systems, 3:123-133, (2009).
14. Mastrovito, E. D., "VLSI Design for Multiplication over Finite Fields GF(2m)". Proc. of Sixth International Applied Algebra, Algebraic Algorithms, AND Error Correcting Codes, (1988) 297-309.
15. Mastrovito, E, "VLSI Architectures for Computation in Galois Fields". PhD thesis, Linköping University, Dept. Electr. Eng., Linköping, Sweden, 1991.
16. Paar, C. A., "New Architecture for a Parallel Finite Field Multiplier with Low Complexity based on Composite Fields". IEEE Transactions on Computers, Vol. 45(7) (1996) 856-861.
17. Paar, C. "Efficient VLSI Architectures for Bit Parallel Computation in Galois Fields", PhD thesis, Institute for Experimental Mathematics, University of Essen, Essen, Germany, June 1994.
18. Ahlquist, G., AND others, "Design AND Synthesis of Small AND Fast Finite Field Multipliers for FPGAs", (2003).
19. G. Ahlquist, B. Nelson, AND M. Rice. "Optimal Finite Field multipliers for FPGAs". In J. Irvine P. Lysaght AND R. Hartenstein, editors, 9th International Workshop on Field Programmable Logic AND Applications (FPL 99), Springer-Verlag. Glasgow, UK, Aug (1999) 51-–61.
20. Markus H., Johann G., Guy-Armand K., "A Versatile AND Scalable Digit-Serial/Parallel Multiplier Architecture for Finite Fields GF(2^m)", 4TH International Conference On Information Technology: Coding AND Computing (2003) 692-700.
21. Sava E. ¸ Tenca A. F., Koç Ç. K., "A Scalable AND Unified Multiplier Architecture for Finite Fields $GF(p)$ AND $GF(2^m)$", In Cryptographic Hardware AND Embedded Systems-CHES 2000.
22. García-M. Mario A., Morales-L G. y Rodríguez-H. F., "Implementación en FPGA de un Multiplicador Eficiente para Campos Finitos Gf(2m)", CINVESTAV, IPN, México, s/f.
23. Sandoval R., Cecilia, Multiplicador Paralelo en Campos Finitos de Galois GF (2m) Aplicado a Códigos Reed Solomon con longitud ajustable sobre FPGA, Congreso Internacional de Investigación UC (2010).
24. Tejeda-calderón, V. C., García-martínez, M. A., & Posada-gómez, R. (n.d.). "Implementación en FPGA de un Multiplicador por digitos sobre Campos Finitos GF (2^m)", División de Estudios de Postgrado Orizaba, Veracruz, 2-5.
25. Mursanto, P., "Comparison of Galois Field Multipliers in Standard AND Composite Field Architectures", Conference on Computer Science & Information Technology (2007) 266-270.
26. Sandoval-Ruiz, Cecilia. "Codificador RS (n, k) basado en LFCS: Caso de Estudio RS (7,3)" Revista Facultad de Ingeniería Universidad de Antioquia Nº 64, (2012) 68-78.

Recibido el 10 de Febrero de 2013

En forma revisada el 12 de Mayo de 2014