CODIFICADOR Y DECODIFICADOR DIGITAL REED-SOLOMON PROGRAMADOS PARA HARDWARE RECONFIGURABLE*

REED-SOLOMON DIGITAL ENCODER/DECODER FOR RECONFIGURABLE HARDWARE

*Cecilia E. Sandoval Ruiz***
*Antonio Fedón****

Resumen: en este artículo se presenta una recopilación de las bases teóricas empleadas para diseñar bloques funcionales del codificador/decodificador *Reed-Solomon* y una metodología de diseño orientada a tecnología *FPGA*. Inicialmente se presenta el diseño del algoritmo del codificador, luego se concibe la arquitectura y se captura el diseño de hardware mediante el empleo de VHDL y la herramienta de sintaxis *Xilinx ISE 6.1*. Finalmente se lleva a cabo la validación del comportamiento del codificador con *ModelSim 5.7* mediante simulaciones de los módulos. Las operaciones en los campos finitos de Galois, $GF(2^m)$, son la base de varios algoritmos en el área de corrección de errores y procesamiento digital de señales. Sin embargo, los cálculos requeridos demandan gran cantidad de tiempo al ser implementados a través de software; por razones de desempeño y seguridad es preferible implementar los algoritmos en hardware.

Palabras clave: hardware reconfigurable, códigos Reed-Solomon, comunicación digital.

Abstract: in this paper we present theory bases for Reed-Solomon Coders/Decoders building blocks, and a methodology to the basic-oriented design of Field Programmable Gate Arrays (FPGA). Initially,

* *Fecha de recepción: 22 de noviembre de 2006. Fecha de aceptación para publicación: 21 de marzo de 2007. El artículo se deriva del proyecto de investigación denominado "Diseño modular de un sistema de procesamiento y comunicación digital usando programación VHDL", financiado por la Universidad de Carabobo.*

** *Ingeniero electricista, Universidad de Carabobo, Valencia, Venezuela. Correo electrónico: cecisandova@yahoo.com*

*** *Ingeniero electricista, Universidad de Carabobo. Magíster en Ingeniería y Doctor en Ingeniería de Telecomunicaciones, University of Florida. Profesor, Universidad de Carabobo, Valencia, Venezuela. Correo electrónico: afedon@cantv.net*

the design of the Coder at the software level is presented, later the architecture and captures using VHDL, with Xilinx ISE 6.1 are showed. Finally, the simulations using ModelSim 5.7 are carried out. The operations in finite or Galois fields, GF(2^m), are the fundamentals for several algorithms in the fields of error-correction codes and digital signal processing. Nevertheless, the calculations involved are time-consuming, especially when they are performed by software. Due to performance and security reasons, it is rather convenient to implement algorithms by hardware.

Key words: re-configurable hardware, Reed-Solomon codes, digital communication.

1. INTRODUCCIÓN

La codificación para control de errores corresponde a una rama de las matemáticas aplicadas llamada teoría de la información (Shannon, 1948). Una aplicación específica corresponde a los códigos Reed-Solomon (Reed, 1960; Wicker, 1999). Los algoritmos que maneja esta aplicación pueden ser implementados tanto en software como en hardware. En vista de la creciente tendencia hacia el uso de dispositivos de lógica reconfigurable a alta escala de integración (Reed, 1960) y de los beneficios que esta tecnología ofrece a los diseñadores de sistemas digitales mediante el empleo de un lenguaje de descripción de hardware como VHDL, que permite configurar sistemas digitales según las especificaciones demandadas por los usuarios, ajustar cambios en la programación y optimizar los diseños tratándolos en forma modular, se plantea el diseño de estos módulos de codificación bajo esta tecnología.

Una importante característica de VHDL es su estandarización bajo la norma 1076 (Ashenden, 1990; Pérez, 2002). Es por ello que se ha seleccionado como lenguaje para la descripción del codificador de canal digital.

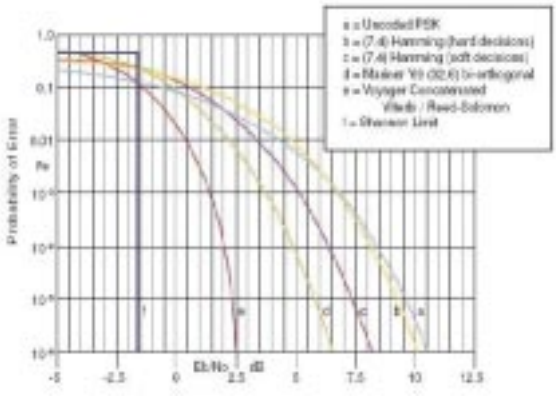
Debido a la gran importancia que tiene el campo de las comunicaciones digitales, por la ventaja que ofrece sobre las comunicaciones analógicas, resulta interesante diseñar un sistema compacto que incorpore las etapas de procesamiento y comunicación de datos, ya que debido a la gran flexibilidad de los FPGA (Field Programmable Gates Array) es factible para cualquier diseño, siempre y cuando se cuente con capacidad suficiente de memoria para almacenar la totalidad del hardware a implementar. Los circuitos FPGA son una solución reconfigurable y eficiente para implementar aplicaciones DSP (Vera, s.f.).

Se planteó entonces la necesidad de emplear la tecnología de programación en VHDL como soporte para el diseño del codificador Reed-Solomon, donde se presenta la posibilidad de subdividir el codificador, atendiendo a cada módulo según su función específica para lograr los objetivos planteados a través de la descripción de hardware (Sandoval, 2006) y realizar los ajustes que permitan optimizar el diseño en la etapa de programación.

2. BASES DEL CÓDIGO REED-SOLOMON

En el estudio de los codificadores de canal se presenta el código Reed-Solomon, el cual resulta ser el más ventajoso (Figura 1). Se puede observar que su probabilidad de error en relación con la señal a ruido está cercana al límite de Shannon (1948) y presenta mayor eficiencia sobre otros códigos correctores de error en cuanto a ganancia del código (Xilinx, s.f.). La clave para hacer del código Reed-Solomon una aplicación tecnológica fue la implementación de un algoritmo eficiente de decodificación desarrollado por Berlekamp (Wicker, 1999).

Figura 1. Característica de los codificadores de canal



Fuente: Xilinx (s.f.)

El código Reed-Solomon es un código corrector de errores basado en bloques en donde el codificador procesa un bloque de símbolos de datos, a los que agrega redundancia para producir un bloque de símbolos codificados. En la actualidad, los códigos Reed-Solomon se utilizan para corregir errores en varios sistemas incluyendo los dispositivos de almacenamiento –cintas, discos compactos, DVD, códigos de barras, etc.–, comunicaciones inalámbricas o móviles –telefonía celular, enlaces de microondas, etc.–, comunicaciones satelitales, televisión Digital/ DVB, módem de alta velocidad como ADSL, x DSL (Xilinx, s.f.).

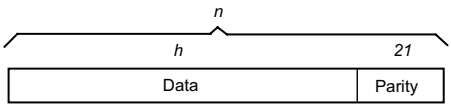
2.1 PROPIEDADES DE LOS CÓDIGOS REED-SOLOMON

El código Reed-Solomon es un subconjunto de los códigos BCH (Bose Chaudhuri Hocquenqhem), códigos cíclicos que presentan entre sus parámetros (n,k,t) una relación entre los símbolos de datos (k), del código total (n) y del número máximo de errores por ser corregidos (t), y son de bloques lineales. Un código Reed-Solomon se especifica como RS(n,k) con símbolos de s bits. Lo anterior significa que el codificador toma k símbolos de los s bits y añade símbolos de paridad para hacer una palabra de código de n símbolos. Existen n-k símbolos de paridad de s bits

cada uno. Un decodificador puede corregir hasta t símbolos que contienen errores en una palabra de código, donde 2t = (n-k) (López, 2005).

La Figura 2 muestra una típica palabra de código Reed-Solomon que se conoce como un código sistemático puesto que los datos se dejan inalterados y los símbolos de paridad se anexan.

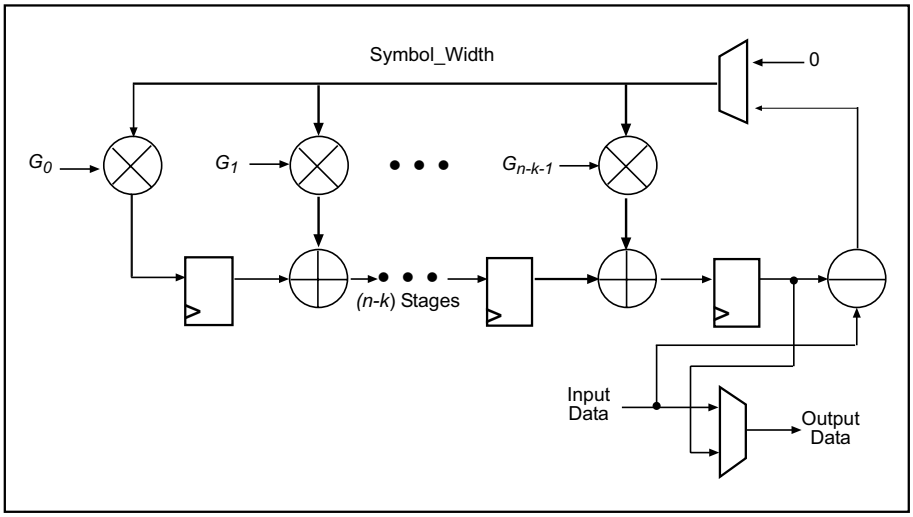
Figura 2. Palabra de Código Reed-Solomon



Fuente: Xilinx (s.f.)

Para codificar la trama con esta estructura se debe procesar a través de un circuito digital que opere bajo los fundamentos de campo finito de Galois. Este presenta una arquitectura en el codificador compuesta por los bloques funcionales mostrados en la Figura 3.

Figura 3. Arquitectura genérica de un codificador Reed-Solomon



Fuente: Xilinx (s.f.)

2.2 Campos de Galois aplicados a la codificación Reed-Solomon

Los códigos Reed-Solomon se basan en un área especializada de la matemática llamada campos de Galois o campos finitos. Un campo finito tiene la propiedad de que las operaciones aritméticas sobre elementos del campo siempre tienen un resultado en el campo. Un codificador o decodificador Reed-Solomon debe ser capaz de realizar estas operaciones aritméticas (López, 2005; Sandoval, 2006; Xilinx, s.f.).

2.3 Generador polinomial

Una palabra de código Reed-Solomon es generada usando un polinomio especial. Todas las palabras de código válidas son divisibles exactamente por el polinomio generador representado por la Ecuación 1.

$$g(x) = (x - \alpha^{i})(x - \alpha^{i+1})...(x - \alpha^{i+2t}) \quad (1)$$

La palabra de código se genera de c(x)=g(x)*i(x), donde g(x) es el polinomio generador, i(x) es el bloque de información, c(x) es una palabra de código válida y á se conoce como un elemento primitivo del campo (Cuervo, s.f.; López, 2005; Xilinx, s.f.).

El primer paso corresponde a la definición del campo de Galois para la codificación, el cual estará definido en función de la longitud del símbolo entiéndase *m* bits / símbolo, el cual permite conocer el polinomio reducible del campo de Galois $GF(2^m)$.

Las bases teóricas que sustentan este codificador están dadas por el polinomio en su forma general (Ecuación 2).

$$g(x) = \prod_{i=0}^{n-k-1} \left(x - \alpha^{hx(Generator\_start+i)}\right) \quad (2)$$

Al expandir el polinomio se obtiene la Ecuación 3.

$$g(x) = G_{n-k-1}x^{n-k-1} + G_{n-k-2}x^{n-k-2} + ... + G_1x + G_0 \quad (3)$$

Donde:

N: longitud de la palabra codificada (en símbolos)

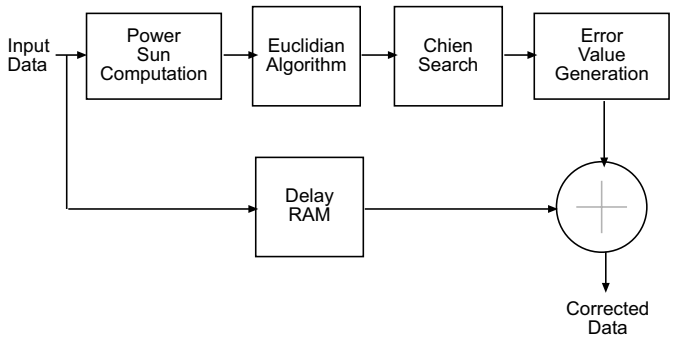
K: longitud del mensaje codificado (en símbolos)

M: longitud del símbolo (bits)

2.4 Bloques funcionales del decodificador

Para el estudio del decodificador se considera el diagrama de bloques presentado en la Figura 4.

Figura 4. Diagrama de bloques del decodificador Reed-Solomon



Fuente: Xilinx (s.f.)

2.4.1 Cálculo del síndrome

Se trata de un cálculo similar al cálculo de paridad. Un código de palabra Reed-Solomon tiene 2t síndromes que dependen solamente de los errores –no de la palabra transmitida–. Los síndromes pueden ser calculados al sustituir las 2t raíces del polinomio generador g(x) en r(x) (Ashenden, 1990).

2.4.2 Localizador de errores

Encontrar el lugar del símbolo erróneo implica resolver de forma simultánea ecuaciones con t incógnitas. Existen varios algoritmos rápidos para hacerlo, los cuales toman ventaja de la estructura matricial especial de los códigos Reed-Solomon y reducen de gran forma el esfuerzo computacional requerido.

2.4.3. Polinomio localizador de error

Este polinomio localizador de error se puede lograr utilizando el algoritmo Berlekamp-Massey o el algoritmo de Euclides. Para localizar los símbolos erróneos se debe seguir un procedimiento que implica resolver unos sistemas de ecuaciones, el primero de los cuales se representa mediante la fórmula general de la Ecuación 4.

$$S_{t+j} = f_1 * S_{t+j-1} + \ldots\ldots\ldots + f_{t-}1 * Sj_{+1} + f_t * S_j \quad (4)$$

2.4.4. Corrección de errores

Este paso también implica resolver ecuaciones con t incógnitas para poder encontrar los valores verdaderos que deberán ser sustituidos en las posiciones correspondientes, para así poder reproducir el mensaje correcto que se intentó enviar. Esto se hace con el algoritmo de búsqueda de Chien. Donde 1<= j <= t los f_i son las incógnitas y S_i los componentes del síndrome calculado. La posición en la que están los errores se obtiene de los exponentes de las raíces de un polinomio f(x) (Ecuación 5).

$$f(x) = x^t + f_1 x^t + \ldots\ldots + f_{t-1} x + f_t \quad (5)$$

2.4.5. Generador del error

A continuación se debe resolver un sistema de ecuaciones con tantas incógnitas como errores hayan sido detectados de acuerdo con la Ecuación 6.

$$Y_1 + Y_2 = S_1 \quad (6)$$

$$x_1 Y_1 + x_2 Y_2 = S_2$$

Donde:

Y_1: valor del error en el símbolo dado por n - exponente de x_1.

x_1: raíz de f(x).

Y_2: valor del error en el símbolo dado por (n - exponente de x_2)

x_2: otra raíz de f(x).

Finalmente se realizará la adición del módulo-2 (xor) entre el dato de entrada que fue recibido por el decodificador –que se mantiene registrado en una memoria RAM– y el error generado con lo que se obtiene el código recuperado.

3. Diseño de un Codificador - Decodificador Reed-Solomon (7,3)

A partir del desarrollo teórico anterior se seleccionó el código RS (7,3), en vista de que logra las ventajas de corrección de errores con una aplicación sencilla para programar en VHDL.

3.1. Diseño del Codificador RS (7,3)

Para comprender en forma práctica el cálculo del código se asume un campo GF(8), es decir, m=3 bits/símbolo. La representación de los elementos del campo estará comprendida por el polinomio irreducible $p(x)= x^3+x+1$, el cual tiene raíz á, por lo que al igualar a cero el polinomio se obtiene $á^3 + á + 1 = 0$, tal que $á^3 = á + 1$ será la representación para sustituir el elemento (8) que se desborda del campo, con lo que se puede convertir a un elemento perteneciente al campo.

A partir de la expresión de G(x), para un codificador RS(7,3) se obtuvo el polinomio generador de la forma (Ecuación 7).

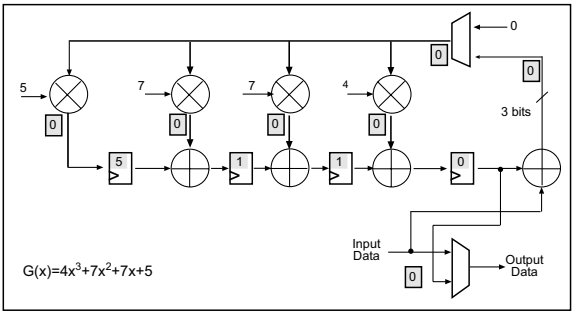
$$G(x) = á^2x^3 + á^5x^2 + á^5x + á^6 \quad (7)$$

el cual es sustituido en función de los valores de los elementos del campo de Galois (Ecuación 8).

$$G(x) = 4x^3 + 7x^2 + 7x + 5 \quad (8)$$

El campo de Galois permite obtener la palabra codificada compuesta por tres símbolos de datos y cuatro símbolos de paridad. Una vez obtenidos los coeficientes del decodificador su arquitectura queda representada como se muestra en la Figura 5.

Figura 5. Codificador RS (7,3)



Fuente: presentación propia de los autores.

Al aplicar los datos de entrada D=[1,3,7,0,0,0,0] se obtiene a la salida C=[1,3,7,0,1,1,5]. Se definieron igualmente los multiplicadores para el campo de Galois, los cuales son componentes que obtienen a su salida el resultado del producto en el campo en función de la entrada, donde el módulo en VHDL para la multiplicación del coeficiente 4 por los símbolos de entrada se muestra en la Tabla 1.

Tabla 1. Código en VHDL del multiplicador por 4 en GF(8)

```
with D_dato select
     dato1<= "000"  when  "000",  —4x0=0
             "100"  when  "001",  —4x1=4
             "011"  when  "010",  —4x2=3
             "111"  when  "011",  —4x3=7
             "110"  when  "100",  —4x4=6
             "010"  when  "101",  —4x5=2
             "101"  when  "110",  —4x6=5
             "001"  when  others;—4x7=1
end Behavioral;
```

Fuente: presentación propia de los autores.

El manejo de los registros y la XOR entre los elementos procesados se muestra en la Tabla 2.

Tabla 2. Código en VHDL de la arquitectura del RS (7,3)

```
Process (clk)
begin
if (clk'event and clik0'1') then
rdato4<=dato4;
rdato3<rdato4 xor dato 3;
rdato2<=rdato3 xor dato2;
rdato1<=rdato2 xor dato1;
D_dato <=rdato1 xor D_in;
end if;
end process;
d_out<=rdato1;
```

Fuente: presentación propia de los autores.

El cálculo del campo de Galois se realizó también usando la herramienta *Matlab*®; el comando para obtener el arreglo de Galois (gf) se aplicó a la matriz de datos a enviar y al número de bits por símbolo, en este caso tres bits por símbolo, es decir, msn = gf([matriz de datos],m), en tanto que la palabra de código se obtiene al aplicar el comando de codificación (rsenc) al arreglo de Galois (msn) y a la longitud de la palabra de código, con la longitud del mensaje. El código en *Matlab*® se muestra en la Tabla 3.

Tabla 3. Código en Matlab del RS (7,3)

```
>> m=3 % longitud del símbolo (en bits)
>> n=7 % longitud del codeword (en símbolos)
>> k=3 % longitud del mensaje (en símbolos)
>> i(x)= gf ( [1,3,7], m)
% aplica gf(2^m) al mensaje sin formato
>> c(x)= rsenc (i(x), n, k)
% aplica g(x).i(x) para obtener el codeword
>>c(x)=[1,3,7,0,1,1,5] %palabra de código obtenida
```

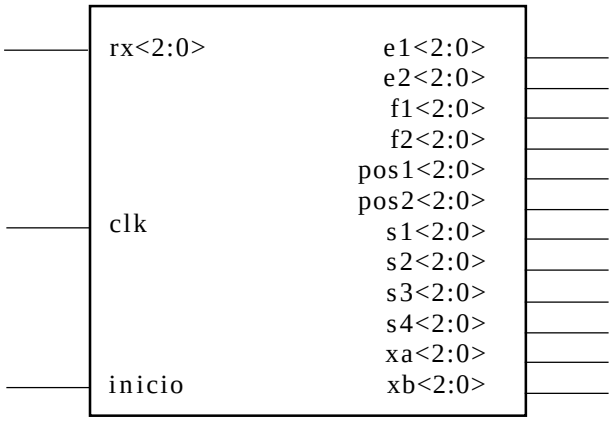
Fuente: presentación propia de los autores.

Esta validación del codificador con *Matlab*® permite comprobar la fiabilidad de su diseño.

3.2. Diseño del decodificador RS (7,3)

Se definen las señales de entrada y salida que están disponibles desde los pines del FPGA, lo que produce por parte del software Xilinx ISE 6.1 la entidad mostrada en la Figura 6.

Figura 6. Entidad del Decodificador RS



Fuente: presentación propia de los autores.

Con base en el diagrama de bloques de la Figura 4 se definieron los componentes representados por cada bloque funcional para los cuales se describe la estructura y sus funciones específicas. La descripción de la estructura bajo la sintaxis VHDL se muestra en la Tabla 4.

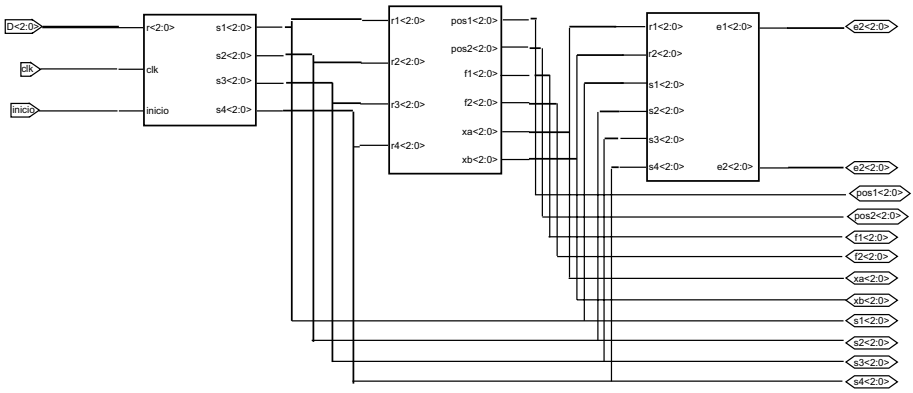
Tabla 4. Código en VHDL de los componentes

```
U1: calc_sindrome  port map  (rx,inicio,clk,s1,s2,s3,s4);
U2: localiz_e      port map  (s1,s2,s3,s4,f1,f2,xa,xb,pos1,pos2);
U3: valor_e        port map  (s1,s2,s3,s4,xa,xb,e1,e2);
end Behavioral,
```

Fuente: presentación propia de los autores.

Cada uno de los módulos internos se describe como componentes. El diagrama RTL a nivel de registros de transferencia para la implementación se muestra en la Figura 7.

Figura 7. Estructura del Decodificador RS



Fuente: presentación propia de los autores.

4. METODOLOGÍA

Entre los pasos que se siguieron para la programación se realizó el modelo del codificador y decodificador a diseñar RS(7,3) con el número de símbolos m=3, con lo cual se logró describir su comportamiento; este fue descrito bajo la sintaxis de lenguaje descriptor de hardware estandarizado VHDL (Chang, 1999; Nazar, 2004; Suardíaz, 2004), y se creó un archivo de asignación de pines para así construir el mapa sobre el dispositivo de hardware reconfigurable seleccionado para esta aplicación. Con el dispositivo FPGA de Xilinx, Spartan IIe 208 pq, se obtuvo como resultado del proceso de programación un archivo de extensión .bit, el cual fue descargado en el dispositivo. Adicional a la fuente del codificador se crearon las fuentes de interfaz para lograr la visualización de los datos en el *display* del banco de prueba DIO1 (Digilab DIO2 Reference Manual, 2002). El archivo .vhdl correspondiente se presenta en la Tabla 5.

Tabla 5. Convertidor de elementos del campo finito a *display* siete segmentos (archivo driver.vhdl)

```
with OCT eelect
display<="1111001" when "001", —1
"0100100" when "010", —2
"0110000" when "011", —3
"0011001" when "100", —4
"0010010" when "101", —5
"0000010" when "110", —6
"1111000" when "111", —7
"1000000" when others; —0
```

Fuente: presentación propia de los autores.

El archivo driver.ucf con la asignación de pines se muestra a su vez en la Tabla 6. De igual manera se configuraron los pines de entrada de datos para la recepción de la trama.

Tabla 6. Asignación de pines al *display* del banco de prueba DIO1 (archivo driver.ucf)

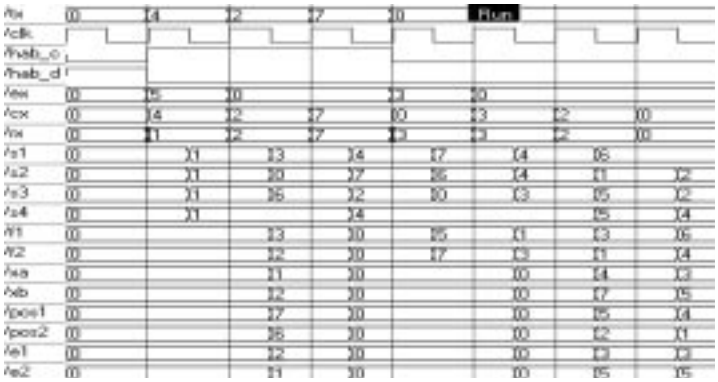
```
NET    "A"   LOC  =   "P17" ;
NET    "a1"  LOC  =   "P45" ;
NET    "a2"  LOC  =   "P47" ;
NET    "a3"  LOC  =   "P49" ;
NET    "a4"  LOC  =   "P56" ;
NET    "B"   LOC  =   "P20" ;
NET    "C"   LOC  =   "P22" ;
NET    "D"   LOC  =   "P24" ;
NET    "E"   LOC  =   "P29" ;
NET    "F"   LOC  =   "P31" ;
NET    "G"   LOC  =   "P34" ;
```

Fuente: presentación propia de los autores.

5. PRESENTACIÓN DE RESULTADOS

Los resultados obtenidos a través de la simulación del codificador y del decodificador corresponden a los mostrados en la Figura 8. *tx* representa la trama de entrada de datos que se ha codificado y que genera la trama *cx*. A esta nueva trama de siete símbolos se le ha adicionado el ruido del canal sobre dos de sus símbolos cualesquiera, *ex*, con lo que se obtiene la trama recibida *rx*. Estos símbolos son procesados y se obtiene s_1,s_2,s_3,s_4 los resultados del síndrome. Estos son procesados a través del sistema de ecuaciones y generan los coeficientes de f(x), en donde f_1,f_2 son entrada a la matriz de conversión para obtener x_a, x_b, que son raíces del polinomio f(x) y con las cuales se calculan los valores de los errores e_1, e_2, los que a su vez son sumados en las posiciones localizadas con la entrada al receptor *rx*, obteniéndose el conjunto de datos corregido.

Figura 8. Simulación del Codec RS (7,3)



Fuente: presentación propia de los autores.

En la implementación del diseño sobre el FPGA se definió la trama de datos a enviar, como en la simulación (D= 4,2,7), correspondientes a la secuencia de datos (D=100,010,111) que son codificados para la transmisión. Así mismo, se presenta la salida de datos a transmitir a través de los *display* con siete segmentos del banco de prueba del sistema de desarrollo del FPGA Spartan IIe 208 pq, Xilinx, tarjeta Digilab DIO2. Este procedimiento se realizó con el fin de completar el proceso de implementación que se muestra en la Figura 9.

Figura 9. Prueba sobre la tarjeta Digilab DIO2

Fuente: presentación propia de los autores.

La síntesis consiste en reducir el nivel de abstracción del circuito hasta convertirlo en una definición estructural, donde los componentes están soportados por la librería de componentes del software. Al final

del proceso de síntesis se obtiene un circuito que funcionalmente se comporta igual que la descripción mostrada. En las Tablas 7 y 8 se aprecian los resultados cualitativos del proceso de síntesis de los módulos.

Tabla 7. Reporte de síntesis del codificador sobre el FPGA

Device utilization summary:				
Selected device: 2s200epq208-6				
Number of Slices:	13	out of	2352	0%
Number of Slice Flip Flops:	12	out of	4704	0%
Number of 4 input LUTs:	23	out of	4704	0%
Number of bonded IOBs:	22	out of	146	15%
Number of GCLKs:	1	out of	4	25%

Fuente: presentación propia de los autores.

Tabla 8. Reporte de síntesis del decodificador en el FPGA

Device utilization summary:				
Selected device: 2s200epq208-6				
Number of Slices:	101	out of	2352	4%
Number of Slice Flip Flops:	27	out of	4704	0%
Number of 4 input LUTs:	174	out of	4704	3%
Number of bonded IOBs:	43	out of	146	29%
Number of GCLKs:	1	out of	4	25%

Fuente: presentación propia de los autores.

En el reporte se muestra el porcentaje de operadores del FPGA utilizados; allí se muestra que se cuenta con una capacidad disponible para expansión y concatenación de módulos de comunicación sobre el mismo dispositivo, ya que se ha empleado sólo un 29% de los pines I/O disponibles en el empaquetado del FPGA seleccionado.

Es importante destacar los beneficios de implementar el codificador-decodificador Reed-Solomon en un FPGA, ya que permite el procesamiento paralelo; adicionalmente, los proveedores del paquete VHDL deben seguir las normas de programación IEEE-1076 a fin de que los diseños sean adaptables a cualquier plataforma (Alvarado, 2004; Agatep, 2000; Arriagada, 2001; Carpio, 1997; IEEE Computer Society, 2002).

6. Conclusiones y trabajos futuros

El diseño descrito es una aplicación práctica sencilla que permite ilustrar el funcionamiento y validar el código en VHDL que puede ser extendido para cualquier codificador RS *(n,k)*. Los módulos programados permiten codificar y decodificar respectivamente los datos para

aplicaciones en sistemas de comunicación, ofreciendo alto nivel de paralelismo en el procesamiento a través de hardware, lo que se traduce en una reducción del tiempo de procesamiento; así mismo, los diseños pueden ser ajustados de acuerdo con los requerimientos particulares, teniendo en cuenta que el comportamiento de los módulos diseñados ha sido validado a través de la simulación y que se realizó el mapa del diseño sobre el FPGA y se obtuvo una programación exitosa.

Entre los aportes realizados se presenta el desarrollo de un modelo de codificación Reed-Solomon bajo sintaxis VHDL con una aplicación de la teoría de campos finitos de Galois.

El diseño de un dispositivo para implementar un código Reed-Solomon RS (255,223) con símbolos de 8 bits es posible gracias a que se cuenta con recursos de hardware suficientes y con el desarrollo base para la expansión; cada palabra de código contiene 255 bytes de palabra de código, de los cuales 223 bytes son datos y 32 bytes son paridad.

Una de las alternativas para trabajos futuros es el posible uso de una cadena concatenada semejante a la propuesta en la norma IEEE 802.16.

REFERENCIAS

AGATEP, Antolin. *Reed-Solomon Solutions with Spartan-II FPGA, WP110 (v1.1)*. [Documento en línea]. February 10, 2000. <http://direct.xilinx.com/bvdocs/whitepapers/wp110.pdf> [Consulta: 10-3-2006].

ALVARADO, Raúl. Códigos para detección y corrección de errores en comunicaciones digitales. *Ingenierías*. 2004, vol. VII, núm. 25, p. 52-60. ISSN 1405-0676.

ARRIAGADA, Álvaro. *FEC (Forward Error Correction) y Código Reed-Solomon*. Universidad de Concepción, 2001.

ASHENDEN, Peter J. *The VHDL Cookbook*. 1st ed. Australia: Dept. Computer Science, University of Adelaide, South Australia, 1990. <http://www.ashenden.com.au>

CARPIO, Fernando. *VHDL Lenguaje para descripción y modelado de circuitos*. Ingeniería Informática. Universidad de Valencia, 1997.

CHANG, K. C. *Digital Systems Design with VHDL and Synthesis, An Integrated Approach*. Los Alamitos: IEEE Computer Society, 1999. 0-76950023-4

CUERVO, Efrén C. *Construcción de un decodificador Reed-Solomon en VHDL*. s.f.

DIGILENT. *Digilab DIO2 Reference Manual*. [Documento en línea]. 2002. <http://www.digilentinc.com>.[Consulta: 10-7-2006].

IEEE COMPUTER SOCIETY. *IEEE Standard VHDL Language Reference Manual*. Los Alamitos: IEEE Computer Society, 2000. ISBN 0-7381-1948-2.

LÓPEZ MARTÍNEZ, Fco. Javier. *Diseño de transmisor y receptor para redes inalámbricas W-MAN*. Tesis de grado. Escuela Técnica Superior de Ingeniería de Telecomunicación. Universidad de Málaga, 2005.

NAZAR A., Saqib. *Implementación eficiente de algoritmos criptográficos en dispositivos de hardware reconfigurable*. Tesis Doctoral. México: Centro de Investigación y de Estudios Avanzados, Instituto Politécnico Nacional, 2004.

PÉREZ L., Serafín A. *et al. Diseño de sistemas digitales con VHDL*. España, 2002. http://www.dte.uvigo.es/vhdl/. [Consulta: 14-10- 2006].

REED, I. S. y SOLOMON, Polynomial Codes over Certain Finite Fields, *Journal of the Society for Industrial and Applied Mathematics*. 1960, vol. 8, núm. 2, p. 300-304.

SANDOVAL, C. Transmisión de datos usando códigos Reed-Solomon e intercalado convolucional implementado sobre FPGA. *Comunicación de Datos*. 2006, vol. 4, pp. 83-89.

SHANNON, C. E. A Mathematical Theory of Communication. *Bell System Technical Journal*. 1948, vol. 27, p. 379-423.

SUARDÍAZ, Juan. Control electrónico mediante telefonía móvil digital basada en la red GSM. *Revista Tecnología y Desarrollo*. 2004, vol. 2.

VERA, M. y VEJARANO, G. *Diseño de funciones DSP usando VHDL y FPGAs*. [Documento en línea] Cali, Colombia: Grupo de Bioelectrónica y Nanoelectrónica, EIEE, Universidad del Valle. <http://www.iberchip.org/IX/Articles/POST-082.pdf>. [Consulta: 15-10-2006].

WICKER, S. B. y BHARGAVA, V. K. *Reed-Solomon Codes an Their Applications*. Wiley-IEEE Press. 1999. 336 p. ISBN: 0-7803-5391-9.

XILINX. *Reed-Solomon Solutions with Spartan-II FPGA. Xilinx System Generator v2.1 for Simulink*. s.l.: s.d.