Nexo Revista Científica

Vol. 21, No. 01, pp. 02-10 / Junio 2008

ISSN 1818-6742
Impreso en Nicaragua
www.uni.edu.ni/Nexo

Diseño de un codificador y decodificador digital Reed-Solomon usando programación en *VHDL*

C. Sandoval* y A. Fedón

Facultad de Ingeniería, Universidad de Carabobo
PO Box 2008, Naguanagua, Sector Bárbula, Venezuela
E-mails: {csandoval1; afedon}@uc.edu.ve

(recibido/received: 14-Nov-2006; aceptado/accepted: 28-Mayo-2007)

RESUMEN

En este artículo se presenta un procedimiento práctico para el diseño de un codificador/decodificador Reed-Solomon a través de la descripción funcional usando lenguaje descriptor de hardware (*VHDL*) con la herramienta de programación *Xilinx ISE 9.2i*. Este trabajo propone un diseño que usa los beneficios que presenta la programación *VHDL*, su característica de modularidad, y la estrategia de seccionar el diseño en componentes menos complejos para facilitar el proceso. Además, se detalla la metodología del diseño del decodificador a través de procesamiento paralelo. Para la validación del comportamiento del codificador/decodificador, se realizaron simulaciones con el programa *ModelSim XE 5.7c*.

Palabras claves: codificador Reed-Solomon; diseño de hardware; *VHDL*

ABSTRACT

This paper presents a practical procedure for designing a Reed-Solomon encoder/decoder through the functional description using hardware descriptor language (VHDL) with the programming tool *Xilinx ISE 9.2i*. We propose a design that use the benefits introduced by VHDL programming, its modularity feature, and the strategy for dividing the design in less complex components to facilitate the process. In addition, the methodology of the decoder design using parallel processing has been detailed. Simulations with *ModelSim XE 5.7c* software were carried out in order to validate the encoder/decoder behaviour.

Keywords: hardware design; Reed-Solomon coder; VHDL

*Autor para la correspondencia

INTRODUCCIÓN

En la actualidad se evidencia una creciente tendencia hacia el uso de dispositivos de lógica reconfigurable a alta escala de integración (Wakerly, 2001), en función de los beneficios que esta tecnología ofrece a los diseñadores de sistemas digitales, estos dispositivos programables a través de un lenguaje de descripción de hardware (VHDL, por sus siglas en inglés), los cuales permiten configurar sistemas digitales según las especificaciones demandadas por los usuarios, ajustar cambios en la programación, optimizar los diseños tratándolos en forma modular.

En este estudio se introduce el tema del lenguaje descriptor de hardware (estandarizado) VHDL, el cuál tiene entre sus características el hecho de ser estructural, lo que permite manejar los aspectos de la programación por módulos, e incluso podemos desarrollar distintos niveles de abstracción dentro de la programación de los dispositivos lógicos programables (CPLD, por sus siglas en inglés).

En este orden de ideas, es fundamental avanzar en dirección al diseño de hardware con programación VHDL, por contar con una normalización IEEE y tener características de modularidad, adquiriendo experiencia en el desarrollo de sistemas de comunicaciones. Por lo anterior expuesto, se estableció como propósito de este proyecto realizar un diseño de un codificador de canal para corrección de errores y sus componentes en los sistemas de comunicaciones digitales en banda base, orientado a la implementación sobre dispositivos programables de alta escala de integración.

De esta manera que se presenta una propuesta didáctica para el diseño de un codificador y su decodificador Reed-Solomon, por presentar estos la capacidad para corregir errores y su utilidad en varios sistemas incluyendo: dispositivos de almacenamiento (cintas, discos compactos, DVD, códigos de barras), comunicaciones inalámbricas o móviles (telefonía celular, enlaces de microondas, etc.), comunicaciones satelitales, televisión digital/DVB, módem de alta velocidad como ADSL, x DSL. (cuervo, E), en vista de lo cual resulta de interés realizar la descripción funcional de estos, a fin de implementarlos a nivel de programación VHDL.

TEORÍA

Programación en VHDL

El auge y la versatilidad que ofrecen los dispositivos lógicos programables han requerido de un lenguaje descriptor de hardware, que permita al diseñador, realizar la configuración del dispositivo de acuerdo a su función especifica, a través de programación estructurada de alto nivel, para así describir el comportamiento de los componentes del sistema digital, sintetizar y finalmente descargar la programación en el dispositivo sea PLD, CPLD, FPGA, a fin de configurar el hardware de manera conveniente al propósito del diseño.

El diseño se puede descomponer jerárquicamente:

1. Definición de diagrama de bloques modular.
2. Identificar los puertos de entrada y salida.
3. Identificar las señales del diseño modular.

Es de hacer notar que el diseño cuenta con una etapa cognoscitiva, una vez realizada la modelación del sistema digital de interés, el próximo paso estará dado por el cumplimiento de las etapas de programación en VHDL.

Codificador de canal Reed-Solomon

En esta etapa se presenta el tratamiento de la trama de datos para formar la palabra de código que será transmitida incluyendo un mecanismo para corrección de errores, en este estudio se considerará un código Reed-Solomon.

Este es un código corrector de errores basado en bloques con un amplio rango de aplicaciones en comunicaciones digitales y almacenamiento. El código fue inventado por Irving S. Reed y Gustave Solomon, miembros del MIT Lincoln Laboratory (Reed y Solomon, 1960). Sin embargo, la clave para hacer del código Reed-Solomon una aplicación tecnológica fue la

implementación de un algoritmo eficiente de decodificación el cuál fue realizado por Elwyn Berlekamp de la universidad de Berkeley (Suardíaz y Al-Hadithi, 2004).

Estos códigos pertenecen a la familia de códigos de bloque, en que el codificador procesa un bloque de símbolos sin codificar, a los que agrega redundancia para producir otro bloque, de mayor longitud, de símbolos codificados, de ahí el interés en desarrollar un proyecto en donde se aplique este tipo de algoritmo de "corrección de errores" en VHDL.

a) Estructura de los códigos Reed-Solomon

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*s y añade símbolos de paridad para hacer una palabra de código de "n" símbolos. Existen n-k símbolos de paridad de "m" *bits* cada uno. Un decodificador puede corregir hasta "t" símbolos que contienen errores en una palabra de código, donde 2t = (n-k) (Xilinx, 2002). La palabra de código presenta la estructura mostrada en la Fig. 1.

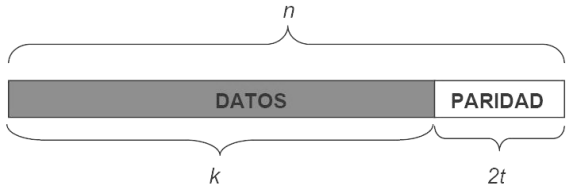

Fig. 1 Palabra de código Reed- Solomon (Clemente, 2004).

b) 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. La forma general de este polinomio es:

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

La palabra de código se genera de:

$$c(x)=g(x)\times i(x) \qquad (2)$$

donde g(x) es el polinomio generador, i(x) es el bloque de información, c(x) es una palabra de código válida y "α" se conoce como un elemento primitivo del campo (Agatep, 2000). 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:

$$g(x)=\prod_{i=0}^{n-k-1}\left(x-\alpha^{h\times(Generator\_Start+i)}\right) \qquad (3)$$

Al ser expandido, se obtiene:

$$g(x)=G_{n-k-1}x^{n-k-1}+G_{n-k-2}x^{n-k-2}+ \\ ....+G_1x+G_0 \qquad (4)$$

donde n es la longitud de la palabra codificada (en símbolos), k la longitud del mensaje codificado (en símbolos) y m la longitud del símbolo (*bits*) (Agatep, 2000).

Diseño del codificador RS (7,3)

Se ha seleccionado un codificador de orden 3, ya que se requería la transmisión de paquetes de 3 símbolos de 3 *bits* cada símbolo, siendo el RS(7,3) apropiado para la aplicación y con base al desarrollo teórico estudiado se realizó la descripción de cada uno de los módulos, que constituyen dichas etapas de codificación y decodificación del canal.

Aplicando los parámetros establecidos, se puede calcular el polinomio primitivo y el correspondiente polinomio generador, con cuyos coeficientes se implementará el diseño.

```
N=7; % tamaño del mensaje (7 simbolos)
K=3; % numero de simbolos de datos
m=3; %numero de bits por simbolo


B=0; %corresponde al valor de i
GENPOLY = RSGENPOLY(N,K,11,B)
Array elements =

 1     4     7     7     5
```

El cual permite obtener la palabra codificada compuesta por 3 símbolos de datos y 4 símbolos de paridad.

```
**************************************
msg = gf([1,3,7],m)
code = rsenc(msg,N,K,GENPOLY)
**************************************
```

Lográndose implementar el codificador de la forma:

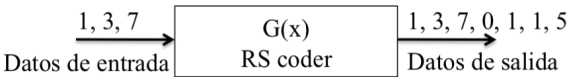

Fig. 2 Codificador RS (7,3).

Al aplicar los datos de entrada D = [1, 3, 7], el polinomio generador se encarga de procesar los datos, lo cual se describe en VHDL a través de una serie de componentes y así se obtiene a la salida C = [1, 3, 7, 0, 1, 1, 5]. La definición jerárquica de los componentes que conforman el codificador se presenta en la Fig. 3.

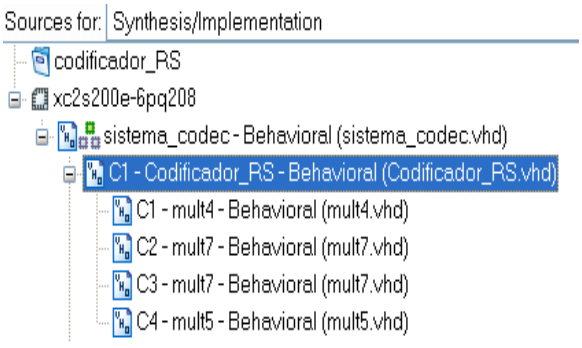

Fig. 3 Jerarquía de componentes del codificador.

El procesamiento y manejo serial de los datos de entrada se realiza a través del siguiente código, presentado en la Tabla 1, correspondiente al módulo principal Codificar_RS, donde es importante destacar que se obtiene una salida serial codificada.

```
process (clk)
variable memoria_v:memoria:=(others=>"000");
begin
if (hab='1')  then
D_dato<=( memoria_v(0) xor D_in);
else
D_dato<="000";
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;
memoria_v(3):=dato4;
s4<=memoria_v(3);
s3<=memoria_v(2);
s2<=memoria_v(1);
s1<=memoria_v(0);
end if;
```

Tabla 1 Código en VHDL de la arquitectura del RS (7,3).

Este diseño es una aplicación práctica sencilla que permite ilustrar el funcionamiento y describir el código en VHDL del codificador RS (n,k).

Diseño del decodificador RS (7,3)

La palabra de código recibida es $r(x) = c(x) + e(x)$, donde se encuentra el código original con los errores adicionales.

En la Fig. 4, se presenta el diagrama de los módulos funcionales que intervienen en el proceso de decodificación, para establecer la descripción del bloque encargado de operar la trama de entrada recibida para obtener el cálculo del síndrome del decodificador RS (7,3).

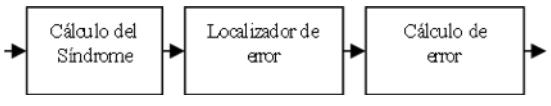

Fig. 4 Módulos del decodificador RS (*n,k*).

a) Función síndrome en el decodificador

Se realiza el procedimiento de calculo de los símbolos de paridad conocido el polinomio generatriz G(x) del codificador y H(x) del decodificador RS(n,k), a fin de evaluar la palabra de código del lado del receptor lo que produce un vector nulo en caso de no haber error en la transmisión y un vector S (síndrome) que permite conocer los símbolos con error.

b) Polinomio localizador de error

Para encontrar un polinomio localizador de error, lo que se puede lograr utilizando el algoritmo Berlekamp-Massey o el algoritmo de Euclides. El algoritmo de Euclides tiende a ser el más utilizado en la práctica debido a que es más fácil de implementar, sin embargo el algoritmo Berlekamp-Massey tiende a llevar a una implementación hardware y software más eficientes. Este algoritmo fue estudiado y tomado como soporte para el diseño del componente en VHDL. Además sus únicas operaciones sobre los datos son la suma polinomial que en hardware son realizadas con compuertas XOR y las multiplicación y división por potencias de 2, que por ser aritmética binaria se traducen en hardware como corrimientos de *bits*.

Ahora se debe localizar los símbolos erróneos, para ello debemos seguir un procedimiento que implica resolver uno sistemas de ecuaciones, el primero de ellos sigue la fórmula general:

$$S_{t+j} = f_1 \times S_{t+j-1} + \ldots\ldots\ldots + f_{t-1} \times S_{j+1} + f_t \times S_j \qquad (5)$$

donde 1< = j < = t los f_i son las incógnitas y S_i los componentes del síndrome calculado.

c) Búsqueda de Chein

La posición en la que están los errores se hace con el algoritmo de búsqueda de Chien, este consiste en obtener de los exponentes de las raíces de un polinomio f(x) cuya fórmula general es:

$$f(x) = x^t + f_1 x^t + \ldots\ldots + f_{t-1} x + f_t \qquad (6)$$

c) Generador del error

Debemos resolver un sistema de ecuaciones con tantas incógnitas como errores hayamos detectado y tiene la forma para nuestro ejemplo:

$$\begin{aligned} Y_1 + Y_2 &= S_1 \\ x_1 Y_1 + x_2 Y_2 &= S_2 \end{aligned} \qquad (7)$$

donde Y_1 indica el valor del error en el símbolo dado por (n - exponente de x_1), x_1 es la raíz de f(x), Y_2 indica el valor del error en el símbolo dado por (n - exponente de x_2), x_2 es la otra raíz de f(x). Finalmente se realizará la adicción entre modulo-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.

El diseño del decodificador que implementa la función inversa del codificador previamente programado. Fue programado a partir del modelo matemático se estructuró para la implementación en hardware. Definido tanto los puertos de entrada y salida, como los componentes del circuito decodificador, como se muestra en la Fig. 5.

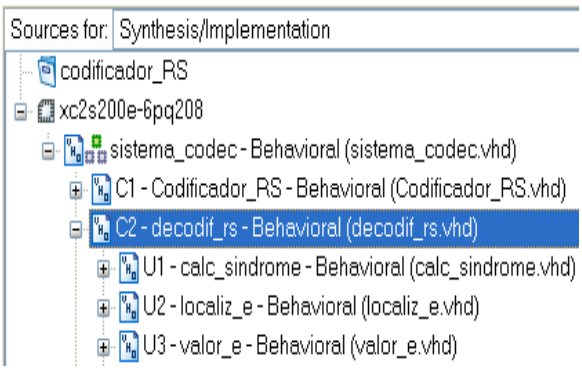

Fig. 5 Definición de componentes del decodificador.

Se procede a calcular los valores de S, aplicando la ecuación (5):

$S_1 = 1 + \alpha^2 + \alpha^5 + 1 + \alpha^2 + \alpha^6 = \alpha$
$S_2 = 1+\alpha^2+\alpha^2+\alpha^3+\alpha^6 = 1+\alpha^2+\alpha^2+\alpha^3 = 1+ \alpha +1 = \alpha$
$S_3 = \alpha^5 + \alpha^5 + \alpha^6 + \alpha^4 + \alpha^4 + \alpha^6 = 0$

$S_4 = \alpha^4+\alpha^3+\alpha^{10}+\alpha^6+\alpha^5+\alpha^6=\alpha^2+\alpha+\alpha^2+\alpha+1=1$

donde se obtiene el síndrome como:

$S = [\ \alpha\ \alpha\ 0\ 1]$

Para el diseño del modulo de cálculo de síndrome en el decodificador Reed-Solomon se debe seleccionar la arquitectura del circuito que realice las operaciones de producto y suma dentro del campo finito del diseño GF (2^3) a fin obtener los síndromes a partir de evaluar la palabra de código recibida r(x) donde $x = \alpha^j$, para $j = 1 \ldots 2t$.

Para localizar la posición del error se tiene el sistema de ecuaciones generado de la ecuación 7, en el caso particular de $n\text{-}k = 4 \rightarrow t = 2$, se tiene $1 <= j <= 2$ por lo que el sistema de ecuaciones queda:

$$S_3 = f_1 \times S_2 + f_2 \times S_1$$

$$S_4 = f_1 \times S_3 + f_2 \times S_2$$

por Kramer pero teniendo en cuenta la aritmética en un campo finito:

$$f_1 = \begin{vmatrix} S_2 & S_1 \\ S_3 & S_2 \end{vmatrix} = S_2.S_2 + S_1.S_3 = \alpha\alpha + \alpha 0 = \alpha^2$$

$$f_1 = \frac{\begin{vmatrix} S_3 & S_1 \\ S_4 & S_2 \end{vmatrix}}{\alpha^2} = \frac{0.\alpha + \alpha 1}{\alpha^2} = \frac{\alpha}{\alpha^2} = \frac{\alpha^8}{\alpha^2} = \alpha^6$$

$$f_2 = \frac{\begin{vmatrix} S_2 & S_3 \\ S_3 & S_4 \end{vmatrix}}{\alpha^2} = \frac{\alpha 1 + 0}{\alpha^2} = \frac{\alpha}{\alpha^2} = \frac{\alpha^8}{\alpha^5} = \alpha^6$$

Al sustituir los coeficientes f_1, f_2 que en este caso particular, se obtiene:

$$f(x) = x^2 + f_1 x + f_2$$

que sustituyendo los valores, se obtiene:

$$f(x) = x^2 + \alpha^6 x + \alpha^6$$

Para encontrar las raíces del polinomio anterior se emplea el algoritmo de búsqueda de Chien. Encontrando los valores del símbolo erróneo. Nuevamente, esto implica resolver ecuaciones con t incógnitas. Un algoritmo usado ampliamente es el algoritmo de Forney. Y para obtener las raíces de f(x) se procede por tanteo.

Este bloque se diseño a través de la evaluación del polinomio en función de los valores de f_i previamente calculados, este procedimiento se aplicó en forma secuencial, se realizó un conjunto de simulaciones para recabar los datos resultantes y así establecer para el diseño en forma paralela una tabla de búsqueda programada para los diferentes valores de f_i obtener el respectivo valor de la raíz x_i. De lo cual se obtiene la Tabla 2.

Tabla 2 Relación de coeficientes f_1,f_2 con las raíces r_1,r_2.

f_1	f_2	r_1	r_2
2	6	5	7
3	3	5	6
6	4	3	5
1	2	4	5
7	1	2	5
4	5	1	5
1	4	6	7
4	2	3	7
3	1	4	7
5	5	2	7
6	7	1	7
5	1	3	6
2	5	4	6
4	7	2	6
7	6	1	6
7	7	3	4
1	6	2	3
2	3	1	3
6	3	2	4
5	4	1	4
3	2	1	2

La posición de los símbolos erróneos se obtiene a partir de los exponentes de las raíces.

Este modelo permitió definir las fuentes que forman parte del proyecto en Project Navigator

versión 9.2i, donde se programará el decodificador Reed-Solomon, cada uno de los bloques, fueron definidos como componentes y se adicionaron como *fuentes .vhdl*, así lo ilustra la Fig. 6.

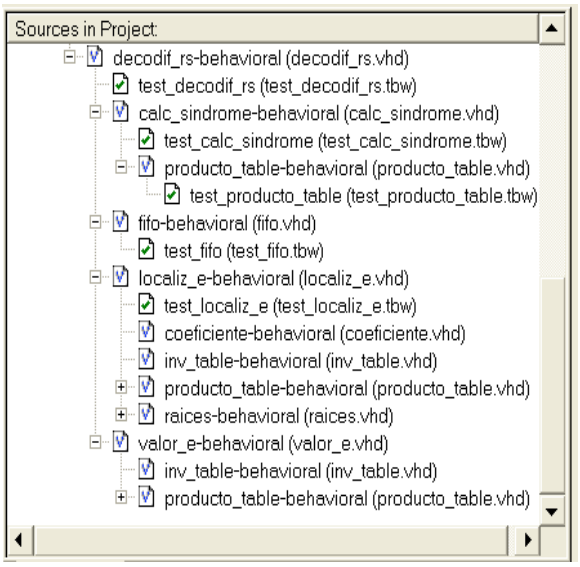

Fig. 6 Jerarquía de las fuentes del decodificador Reed-Solomon.

Y se procede a la descripción del comportamiento. En la Tabla 3 se presenta el código que describe el módulo de localizador de errores, donde se obtienen las posiciones de los símbolos errados, con la evaluación de las raíces obtenidas del sistema de ecuaciones resultantes de aplicar la ecuación (6).

Tabla 3 Código del localizador de errores.

```
M1: producto_table port map (s2,s2,dd1);
M2: producto_table port map (s1,s3,dd2);
det <= dd1 xor dd2;--+
M3: inv_table port map (det,inv_det);
M4: producto_table port map (s3,s2,d11);
M5: producto_table port map (s1,s4,d12);
mat1<= d11 xor d12;
M6: producto_table port map (mat1,inv_det,f1);
M7: producto_table port map (s2,s4,d21);
M8: producto_table port map (s3,s3,d22);
mat2<= d21 xor d22;
M9: producto_table port map (mat2,inv_det,f2);
M12: RAICES port map (f1,f2,xa,xb);
M13: coeficiente port map (xa,med1);
M14: coeficiente port map (xb,med2);
```

En la Fig. 7, se muestra el proceso de programación.

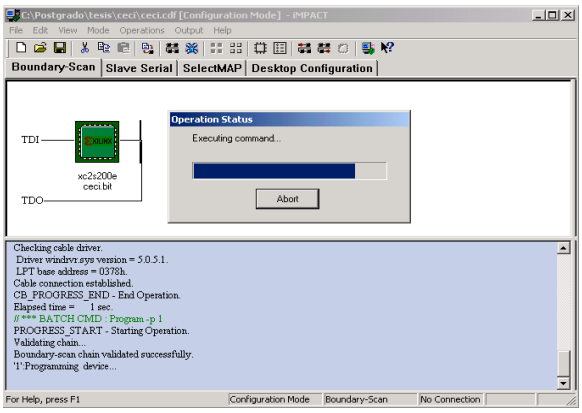

Fig. 7 Proceso de programación del FPGA.

RESULTADOS Y DISCUSIÓN

Los resultados obtenidos a través de la simulación del codificador y del decodificador corresponde a las Figs. 8 y 9, respectivamente.

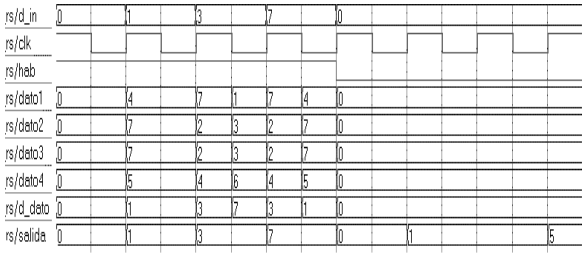

Fig. 8 Simulación del codificador RS.

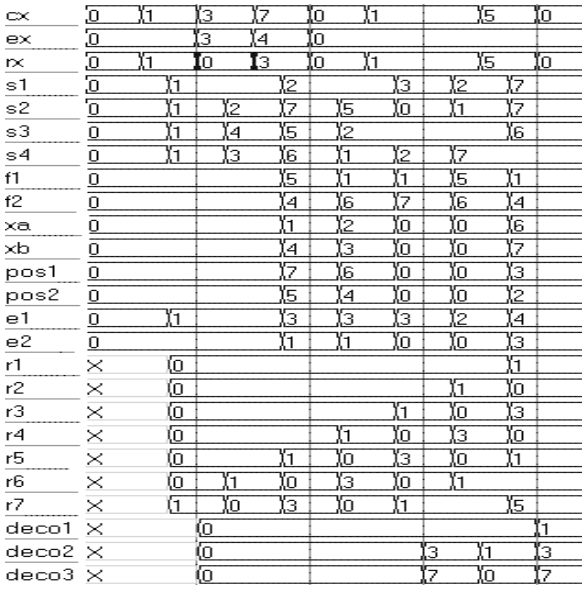

Fig. 9 Simulación del decodificador RS.

En la simulación del comportamiento del codificador Reed-Solomon, se estableció el procesamiento de los datos recibidos (*d_in*). En la Fig. 9, se puede observar la salida del codificador, será la señal de entrada *cx*, es la trama de entrada de datos, a esta nueva trama de 7 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 produciendo los coeficientes de f(x) entiéndase f_1, f_2 estos son entrada a la matriz de conversión para obtener x_a, x_b estas ultimas raíces del polinomio f(x) y con los cuales se calculan los valores de los errores e_1, e_2, los cuales son sumados en las posiciones localizadas con la entrada al receptor *rx* obteniéndose los datos corregidos.

El resultado exitoso del diseño y las características de VHDL, que siguen las norma de programación IEEE-1076, hace que los diseños sean portátiles a cualquier plataforma (Chang, 1999).

CONCLUSIONES

El estado actual sobre codificación Reed-Solomon. Se ha incluido un compendio de la teoría de codificación Reed-Solomon. Así como, un desarrollo de un modelo de codificación Reed-Solomon. Una de las mejores alternativas es el posible uso de una cadena concatenada semejante a la propuesta en la norma IEEE 802.16.

La arquitectura diseñada fue optimizada para dispositivos reconfigurables (*FPGAs*) tomando ventaja del máximo nivel de paralelismo que se puede explotar en la multiplicación y operaciones de la aritmética de campos finitos de Galois GF(2*m*) mediante un uso eficiente de loa recursos de hardware (Sandoval, 2006).

Se ha podido diseñar un codificador-decodificador Reed-Solomon que puede corregir errores en dos símbolos, con las facilidades de una descripción a través de un programa de programación de hardware.

TRABAJOS FUTUROS

El diseño de un dispositivo para implementar un código Reed-Solomon RS(255,223) con símbolos de 8 *bits*. En el cuál, el decodificador pueda corregir cualquier error de 16 símbolos en la palabra de código, es decir, errores de hasta 16 *bytes* en cualquier lugar de la palabra pueden ser automáticamente corregidos. Una vez establecido el polinomio g(x), los síndromes pueden ser calculados al sustituir las 2t raíces del polinomio generador g(x) en r(x).

REFERENCIAS

Agatep, A. (2000). *Reed-Solomon Solutions with Spartan-II FPGAs*. White paper: Spartan-II Family.http://direct.xilinx.com/bvdocs/whitepapers/wp110.pdf

Chang, K.C. (1999). *Digital Systems Design with VHDL and Synthesis: An Integrated Approach*. Wiley-IEEE Computer Society Pr. New Jersey, USA.

Clemente, E. (2004). *Construcción de un Decodificador Reed-Solomon en VHDL*. Informe Técnico. Centro de Investigación y de Estudios Avanzados. Ciudad de México, México.

Sandoval, C. (2006). *Transmisión de Datos usando Códigos Reed-Solomon e Intercalado Convolucional Implementado sobre FPGA*. Comunicación de Datos, Vol. 4, pp. 83-89.

Reed, I.S. y G. Solomon (1960). *Polynomial Codes over Certain Finite Fields*. SIAM Journal on Applied Mathematics, Vol. 08, No. 02, pp. 300-304.

Suardíaz, J. y B.M. Al-Hadithi (2004). *Control Electrónico mediante Telefonía Móvil Digital basada en la Red GSM*. Tecnología y Desarrollo. Revista de Ciencia, Tecnología y Medio Ambiente, Vol. 02.

Wakerly, J. (2001). *Diseño Digital: Principios y Prácticas*. 3ra Edición. Editorial Pearson. Ciudad de México, México.

Xilinx, Inc. (2002). *Xilinx System Generator v2.1 for Simulink: Basic Tutorial*. The Xilinx Design Series. Disponible en línea: http://www.xilinx.com/support/library.htm

Cecilia Sandoval es Ingeniero Electricista y Magíster en Ingeniería Eléctrica en la Universidad de Carabobo, Venezuela. Siendo su área de investigación procesamiento digital de datos con énfasis en aplicaciones de hardware reconfigurable. Docente-Investigador en la Universidad Nacional Experimental de las Fuerzas Armadas (UNEFA), Venezuela.