Ingeniare. Revista chilena de ingeniería, vol. 16 Nº 2, 2008, pp. 310-317

DESCRIPCIÓN MODULAR DE UN ESQUEMA DE CODIFICACIÓN CONCATENADO PARA CORRECCIÓN DE ERRORES CON PROGRAMACIÓN DE HARDWARE

MODULAR DESIGN OF SCHEME CODING CONCATENATED FOR CORRECTION ERROR WITH PROGRAMMING OF HARDWARE

Cecilia E. Sandoval Ruiz[1] **Antonio Fedón** [1]

Recibido el 10 de enero de 2007, aprobado el 4 de abril de 2008
Received: January 10, 2007 Accepted: April 4, 2008

RESUMEN

Las comunicaciones inalámbricas requieren el empleo de métodos de corrección de errores sobre los datos transmitidos, usándose generalmente técnicas de codificación Reed-Solomon & Viterbi, por razones de desempeño y seguridad es preferible implementarlos sobre hardware. En este trabajo se presenta el diseño modular de la etapa de codificación de estos códigos para su concatenación usando VHDL (VHSIC Hardware Descriptor Language), orientado a la implementación sobre tecnología de matriz de compuertas programadas por campo (FPGA). Se inicia con una revisión de los conceptos asociados a la definición de los componentes, y el modelo, descripción del comportamiento, luego la arquitectura es diseñada usando la sintaxis en VHDL y es capturado el diseño de hardware, finalmente se presentan los resultados de síntesis.

Palabras clave: VHDL, hardware reconfigurable, codificadores, comunicación digital.

ABSTRACT

The wireless communication medium requires employing forward error correction methods on the data transferred, where Reed-Solomon & Viterbi coding techniques are generally utilized, because of performance and security reason. In this paper we present a modular design of phase encoding these codes for concatenation using VHDL (VHSIC Hardware Descriptor Language) and oriented to implementation with field programmable gate arrays (FPGA). The work begins with a review of code concept and the definition of the components and the model and the description of the behavioral. Later, the architecture is designed and captured using syntax in VHDL, and finally presents the results of synthesis.

Keywords: VHDL, hardware re-configurable, coders, communication digital.

INTRODUCCIÓN

En los procesos de transmisión de datos se deben emplear técnicas de codificación con corrección de error para garantizar la confiabilidad de la información. Entre las posibles soluciones para contrarrestar los efectos del canal pueden mencionarse algunas metodologías, como por ejemplo el uso de diversidad temporal, mediante códigos robustos, es decir, usando una cadena concatenada de códigos.

Entre la codificación de canal [9,11], se presentan los esquemas empleados con mayor frecuencia, los cuales serán considerados con el objetivo de identificar los módulos a definir para realizar la descripción estructural.

En particular se revisan los códigos concatenados de forma serial compuestos por Reed-Solomon (como codificador externo) y códigos convolucionales (como codificador interno) y los intercaladores presentados en [17] para disminuir los errores por efecto ráfaga, siendo este esquema de codificación concatenado una alternativa eficiente, cuya ganancia se acerca al límite de Shannon, con respecto al empleo de los códigos de forma separada.

La concatenación entre códigos fue introducida por Forney como una técnica muy práctica para obtener un código de longitud suficientemente alta y una capacidad correctora extremadamente elevada. Eso se logra utilizando múltiples niveles de codificación, con

[1] Universidad de Carabobo/Dirección de Postgrado. Facultad de Ingeniería Naguanagua sector Bárbula Venezuela. Teléfonos directos (0241) 8672829 /8674268 ext. 102, fax: (0241) 8671655. Código postal: 2008. E-mail: csandoval1@uc.edu.ve, afedon@uc.edu.ve

el fin de lograr unas ganancias de codificación grandes, cercanas al límite de Shannon [1], por la combinación de dos códigos componentes relativamente simples. El esquema de codificación resultante es muy potente y está dotado de una estructura que permite una decodificación sencilla.

El propósito principal de este artículo es presentar una estrategia general de describir los módulos en VHDL (lo cual presenta una cantidad importante de ventajas sobre los circuitos integrados estándar) de un código concatenado para mitigar los efectos del canal, a través de las características del lenguaje descriptor de hardware que ofrece alto nivel de paralelismo para diseños específicos [4-5,16-17]; para esto se realizará una descomposición de los bloques que constituyen dichos esquemas, con el fin de ilustrar los aspectos relevantes de la técnica de descripción de las fases. Siendo estos desarrollos herramientas que se pueden tomar como referencia para posteriores aplicaciones. En tal sentido; es oportuno mencionar que estos componentes han sido trabajados por diferentes autores para aplicaciones definidas, en tanto que acá se busca dejar propuestos los componentes básicos que puedan adaptarse a los diferentes esquemas concatenados.

BASES TEÓRICAS

Existen esquemas estandarizados, como es el caso de la configuración recomendada por el CCSDS (Consultative Committee for Space Data Systems) compuesta por un Reed-Solomon (255,223,32), un entrelazado bloque y un código convolucional (2,1,7) y la propuesta por el estándar IEEE 802.16, compuesta por un Reed-Solomon (204,188,16), un intercalador convolucional y un código convolucional con diversas tasas (desde 1/2 a 7/8). Estas estructuras permitirán identificar los componentes de interés que deben ser analizados y modelados mediante la descripción en VHDL para su implementación en hardware.

Una probabilidad de error muy baja puede obtenerse en el receptor gracias al efecto combinado de los dos códigos. En teoría, la concatenación en dos niveles no impide elegir cualquier pareja de códigos ni tampoco su posición dentro del esquema total.

La flexibilidad particular de un concepto de este tipo dejaría una libertad casi absoluta sobre la configuración que se puede elegir (Reed-Solomon & Viterbi). En la realidad, la configuración más utilizada es sobre dos niveles, porque ya con este tipo de concatenación, que es la más simple, se logran resultados más que suficientes para un número muy elevado de aplicaciones. El concepto básico de un sistema concatenado se ilustra en la figura 1.

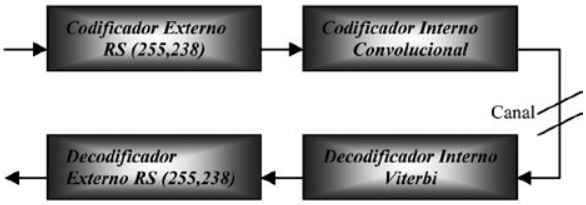

Figura 1. Codificación concatenada.

Originalmente el código Reed-Solomon está diseñado para la operación sobre campos de Galois GF (2^m), y la longitud de estos códigos es $n=2^m-1$. Una interesante aplicación de estos códigos es el diseño de códigos RS sobre una longitud del campo Galois $GF(2^8) = GF(256)$, porque cualquiera de los elementos de un vector en estos códigos es en sí mismo un vector de 8 bits, o de un byte. Un código RS diseñado sobre este campo, en condiciones para corregir cualquier error patrón de tamaño t = 2 o menos, es el código RS(255, 251,4)

Es de hacer notar que una tabla de las palabras de este código sería enorme. Sin embargo, la primera página de la tabla contendría ceros en las posiciones más significativas del mensaje. Por lo tanto, un código abreviado RS puede ser generado de la supresión de las posiciones que son cero en todas las palabras de código en esta página. Más concretamente, un código abreviado RS puede ser construido mediante la fijación de los símbolos de mensaje sRS a cero, en donde $1 \leq sRS <$k. Entonces, la longitud del código es n-sRS símbolos, el número de símbolos de los mensajes es k-sRS y el número de símbolos de la paridad es $n - k$ como antes, en la que todos los símbolos permanecen en GF (2^m). El polinomio generador y la capacidad de corrección de errores del código abreviado es el mismo que el del código recortado, pero el código ya no es cíclico, ya que no todos los cambios cíclicos de palabras de código abreviado en el código también se encuentran en el código acortado. Por esta razón, el código abreviado se dice que es casi cíclico, según lo enuncia [11].

A partir de la aplicación de códigos RS abreviados se presenta el esquema de codificación CIRC (Cross Interleaving Reed-Solomon Code), el cual está compuesto de tres niveles de intercalado y dos etapas de codificadores Reed-Solomon, siendo éste un código corrector de errores basado en bloques con un amplio rango de aplicaciones en comunicaciones digitales y almacenamiento, empleándose en este esquema comúnmente un codificador externo

RS(28,24,4) y un codificador interno RS(32,28,4), esquema que se muestra en la figura 2.

Figura 2. Esquema de codificación CIRC.

CODIFICADOR REED-SOLOMON

El codificador Reed-Solomon cuenta con un comportamiento definido en el campo de Galois; una palabra de código Reed-Solomon es generada usando un polinomio especial [2-3]. Todas las palabras de código válidas son divisibles exactamente por el polinomio generador, cuya forma general corresponde a la mostrada en la ecuación 1:

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

El diseño de un dispositivo para implementar un código Reed-Solomon RS(255,239,16) con símbolos de 8 bits, en el cual cada palabra de código contiene 255 bytes de palabra de código, de los cuales 239 bytes son datos y 16 bytes son paridad, con símbolos de 8 bits de longitud, consta de una serie de pasos.

El procedimiento seguido para el diseño del codificador Reed-Solomon RS(255,239,16) comprendió la definición de los coeficientes del codificador dentro del campo finito [15], esto a través de la ecuación (1), donde se obtuvo el polinomio generador de la forma:

G(x) =[1 59 13 104 189 68 209 30 8 163 65 41 229 98 50 36 59]

Donde el polinomio primitivo corresponde a la ecuación 2:

$$p(x) = x^8 + x^4 + x^3 + x^2 + 1 \ \textit{(285 decimal)} \qquad (2)$$

Siendo éste el polinomio empleado para dividir los productos a fin de que todos los resultados queden dentro del campo.

Estos polinomios estarán dados por el número de bits definidos por símbolo [8], como se ilustra en la tabla 1.

Una vez establecidos el polinomio y los coeficientes, éstos son ajustados en el esquema del codificador para iniciar el diseño conceptual y así definir los parámetros que intervienen en la descripción funcional del codificador bajo el lenguaje de descripción de hardware VHDL, como se muestra en la figura 3.

Tabla 1. Polinomios de n símbolos.

Symbol Width	Default Polynomials	Array Representation
3	$x^3 + x + 1$	[1011]
4	$x^4 + x + 1$	[10011]
5	$x^5 + x^2 + 1$	[100011]
6	$x^6 + x + 1$	[1000011]
7	$x^7 + x^3 + 1$	[10001001]
8	$x^8 + x^4 + x^3 + x^2 + 1$	[100011101]
9	$x^9 + x^4 + 1$	[1000010001]
10	$x^{10} + x^3 + 1$	[10000001001]
11	$x^{11} + x^2 + 1$	[100000000101]
12	$x^{12} + x^6 + x^4 + x + 1$	[1000001010011]

Fuente: System Generator Xilinx (2007).

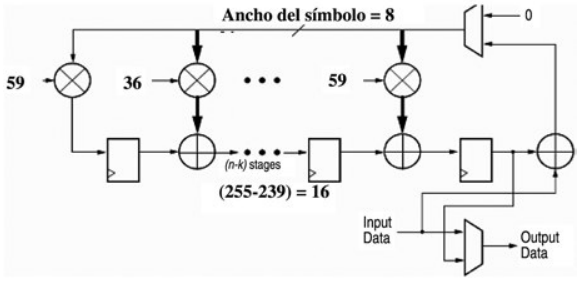

Figura 3. Estructura del codificador RS(255,239,16).

Donde se observa cómo quedaría la arquitectura a configurar para tal aplicación donde la longitud de los registros de memoria será de 8 bits, se crearán 16 etapas para el RS(255,239,16) y se programarán las tablas (LookUp Table) que contienen la información de los productos en el campo de Galois para efectuar la operación entre los coeficientes y los datos de entrada, de esta manera agregando las líneas de código correspondiente en VHDL se extiende la aplicación con base al desarrollo realizado por [15], siguiendo de esta manera la metodología aplicada.

A partir de dichos coeficientes se establece la matriz generatriz, que comprende una serie de pasos que se detallan a continuación. En primer lugar se establecen los coeficientes de las filas, usando el polinomio generador, a partir de la presente expresión: $x^{n-i}/x^k + g(x)$, de donde aplicando la fórmula genérica se tiene la ecuación 3.

$$x^{n-i} \quad \Big| x^k + x^6 + g_5 x^5 + g_4 x^4 + g_3 x^3 + g_2 x^2 + g_1 x^1 + g_0 \qquad (3)$$
$$r_{n-i}(x)$$

Donde los resultados del residuo de la división serán sumados al dividendo y los coeficientes determinarán

la fila para cada valor de i, como se muestra en la ecuación 4.

$$f_i = x^{n-i} + r_{n-i}(x) \tag{4}$$

A partir del cálculo de cada fila, se construye la matriz generatriz, la cual sigue la estructura de la ecuación 5.

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & ... & 0 & r_1 \\ 0 & 1 & 0 & 0 & ... & 0 & r_2 \\ 0 & 0 & 1 & 0 & ... & 0 & r_3 \\ ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & 0 & 1 & r_n \end{bmatrix} \tag{5}$$

Para la implementación se programan las tablas (LutTable) correspondientes a los productos de los coeficientes, por ejemplo, los productos de todos los posibles valores del campo GF (2^8) por el primer coeficiente del diseño (59) serán: 0 59 118 77 236 215 154 161 197 254 179 …. 106, el cual se describe en VHDL; la tabla 2 resume el componente producto para uno de los coeficientes.

Tabla 2. Tabla producto coeficiente 59 en VHDL.

```
with D_dato select
d<=     "00000000"  when "00000000", --59x0=0
        "00111011"  when "00000001", --59x1=59
        "01110110"  when "00000010", --59x2=118
        "01001101"  when "00000011", --59x3=77
        "11101100"  when "00000100", --59x4=236
        "11010111"  when "00000101", --59x5=215
        "10011010"  when "00000110", --59x6=154
        .
        .
        .
        "10000001"  when others;     --59x255=106
```

Para el proceso de decodificación se procede a aplicar el procedimiento inverso usando la matriz transpuesta.

Se pueden obtener los errores del código de la columna respectiva al símbolo alterado multiplicando la matriz H por la trama de datos recibida, calculando el error a través de la ecuación 6.

$$E = R * G^{-1} \tag{6}$$

Finalmente se suma en módulo-2 (XOR) el error al símbolo detectado como dañado y se logra corregir la data recibida, con la ecuación 7.

$$RC = R \oplus E \tag{7}$$

Por último, se tiene la corrección tal como se esperaba; el procedimiento puede ser detallado en el trabajo [7], donde se presenta el diseño de un decodificador RS(7,3,3), con fines didácticos. Para cualquier código general RS(n,k) se pueden aplicar los módulos diseñados para la implementación sobre hardware reconfigurable de módulos de operación en álgebra de campos finitos de Galois, GF (2^m), presentados en [3], según el ancho de los símbolos y el número de símbolos de redundancia variarán en la síntesis los recursos utilizados.

CODIFICADOR CONVOLUCIONAL

El codificador convolucional tiene como característica la inserción de k bits de redundancia asociado a la combinación de n bits consecutivos. Un codificador convolucional (n,k,K) para k entradas que ingresan a un circuito secuencial, con n salidas que se implementa a través de K niveles de memoria [10]. De forma tal que si se considera un codificador convolucional (*2,1,3*), la entrada serial de 3 bits de datos se puede obtener el bit generado resultante que será insertado en la palabra de código para la comprobación de la paridad. El circuito para la implementación se muestra en la figura 4.

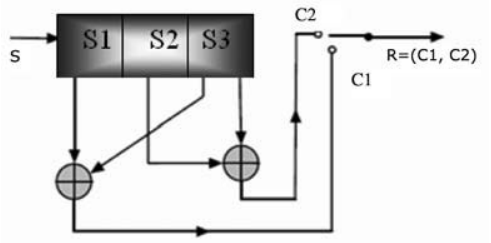

Figura 4. Esquema del codificador convolucional (*2,1,3*).

Para la implementación del codificador convolucional se requieren K=3 elementos de memoria (en forma de cascadas de registros desplazamiento), donde el código contendrá un (1) bit de redundancia por cada bit de datos s, lo que se traduce en el doble de bits transmitidos, en comparación a un sistema sin codificación. La generación del código de r bits estará dada por la concatenación de los elementos c resultado de la suma módulo dos (adder mod-2 o xor); para el caso de r = 2 bits se tendrá la ecuación 8, donde s_1,s_2,s_3 tres bits consecutivos del código fuente. Es importante mencionar que para un canal con una velocidad de transmisión alta esta técnica resulta eficiente.

$$\begin{bmatrix} c_1 = s_1 \oplus s_3 \\ c_2 = s_1 \oplus s_2 \end{bmatrix} \tag{8}$$

Para la función generadora se tiene que solo existe dada la llegada de dos bits codificados una posibilidad de bit fuente transmitido, por tal motivo se realizará para el caso de la decodificación un estudio de posibilidades de bit transmitido de acuerdo al par de bits recibido, estableciendo un diagrama de flujo que obtenga el proceso inverso a la codificación.

El diagrama de estado de la figura 5 muestra el algoritmo del decodificador de Viterbi, el cual es una representación de la evolución de la secuencia de estados para estos códigos [8], una vez que se limpian (reset) los elementos de memoria se obtiene el primer estado A; el código posible es R=00 para un código fuente s=0 y R=11 para un código de entrada s=1, cualquier otra condición corresponde a un error y se debe estudiar el próximo par de bits y determinar qué código fuente fue afectado por el ruido en el canal. En función de la descripción funcional de ese codificador y su respectivo algoritmo de decodificación (decodificador de Viterbi) se puede modelar esta etapa para protección de la data en el canal.

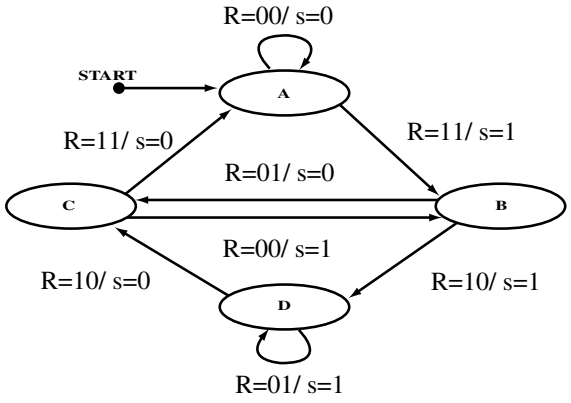

Figura 5. Diagrama de estado del decodificador Viterbi.

Para la programación se consideró el diseño de una máquina de estado que satisfaga el algoritmo del decodificador Viterbi, antes enunciado, de tal manera que se obtiene el código en VHDL que se presenta en la tabla 3, donde, siguiendo el diagrama de flujo, se establecen los estados para conocer los bits decodificados según las entradas *R* y el estado actual.

Al hacer la simulación del decodificador de Viterbi diseñado se observa la salida (trama recuperada) en función de los datos con los bits de redundancia insertados. En la simulación del comportamiento del decodificador Viterbi se puede observar que para la entrada *r*, señal de 2 bits de longitud que corresponde a los datos recibidos retrasados en el tiempo, y la entrada *prox*, que corresponde a la entrada recibida actualmente, se establece una secuencia en función del diagrama de estados del decodificador Viterbi, en el cual se nota en activo solo uno de los estados correspondiente a la transición de la data recibida y la próxima recibida, así se obtiene una salida decodificada, la cual debe corresponder con la fuente de datos transmitida, *s*, señal decodificada de salida. Como se muestra en la figura 6.

Tabla 3. Algoritmo del decodificador Viterbi.

```
IF (( R="00" AND  A='1') OR ( R="11" AND C='1')) THEN
next_A<='1';
ELSE
next_A<='0';
END IF;
IF (( R="11" AND A='1') OR ( R="00" AND C='1')) THEN
next_B<='1';
ELSE
next_B<='0';
END IF;
IF (( R="01" AND B='1') OR ( R="10" AND D='1')) THEN
next_C<='1';
```

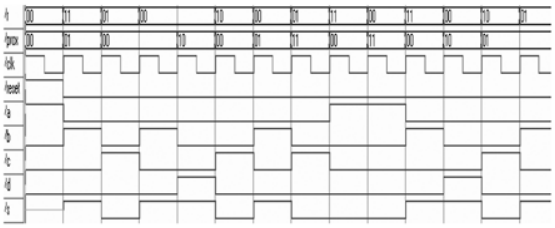

Figura 6. Simulación del decodificador Viterbi.

Se ha logrado la correcta decodificación de la data, incluso se ha introducido a propósito un error en los datos recibidos posición 5 y 12 y de acuerdo al código creado éste ha sido detectado y corregido, lo que permite validar el diseño.

Al analizar el comportamiento del decodificador de Viterbi podemos observar que para el quinto dato, *r=00*, el estado corresponde a *d*, lo cual detecta un error puesto que no está entre las entradas para la decodificación de este estado sino las opciones *r=01* y *r=10*. Es en base al dato *prox=10* que se puede definir que el dato correcto a decodificar corresponde a *s=1*, con lo cual se corrige el error en la trama transmitida.

Finalmente se obtiene una tabla de los recursos utilizados por la implementación del diseño, sobre un FPGA Spartan IIe 208 pq, Xilinx, tarjeta Digilab DIO2 [14], como se muestra en la tabla 4.

Tabla 4. Resumen de recursos sintetizados del codificador convolucional y decodificador Viterbi.

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	4	6,144	1%
Number of 4 input LUTs	17	6,144	1%
Logic Distribution			
Number of occupied Slices	9	3,072	1%
Number of Slices containing only related logic	9	9	100%
Number of Slices containing unrelated logic	0	9	0%
Total Number of 4 input LUTs	17	6,144	1%
Number of bonded IOBs	10	178	5%
IOB Flip Flops	1		
Number of GCLKs	1	4	25%
Number of GCLKIOBs	1	4	25%
Total equivalent gate count for design	142		

En primer lugar, han sido utilizados 9 slices, lo que supone el 1% de la capacidad de la FPGA. Xilinx recomienda una ocupación de estos dispositivos inferior al 80%. Teniendo en cuenta este dato, se puede decir que la elección de la FPGA ha sido correcta, donde se dispone de suficiente capacidad del dispositivo para implementar otros módulos.

Por otro lado, de los 140 pines de entrada/salida disponibles, únicamente han sido utilizados 10. Esto corresponde con el 5% del total, lo cual indica que esta característica de la FPGA está siendo infrautilizada. Esto permitiría una gran ampliación, pudiendo dotar al sistema de un gran número señales de control y variables de entrada adicionales.

INTERCALADOR CONVOLUCIONAL

La técnica consiste en enviar al receptor réplicas independientes de la secuencia transmitida, transformando los errores correlacionados en errores aleatorios [17]. Más específicamente, consiste en eliminar la correlación entre los datos en recepción. En sistemas concatenados, el entrelazado convolucional es necesario entre el codificador RS y el convolucional, porque la codificación RS trabaja mejor cuando los símbolos erróneos no están correlacionados. Sin el entrelazador o intercalador convolucional los símbolos erróneos tienden a correlacionarse y se agrupan en ráfagas producidas por el decodificador de Viterbi.

El principio de funcionamiento del intercalador convolucional de Forney consiste en un conjunto de retardos de línea estructurados a través de registros desplazamientos. Los símbolos de entrada serializados son distribuidos por medio de un multiplexor (conmutador) a cada una de las ramas del circuito, tal como se observa en la figura 7.

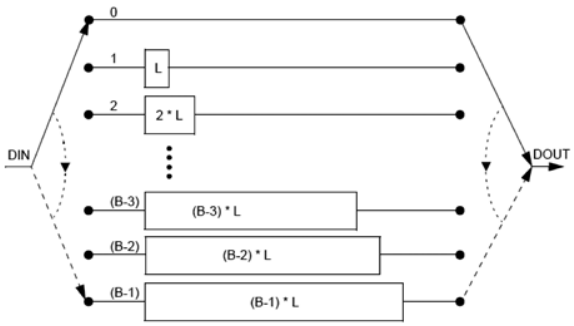

Figura 7. Intercalador convolucional Forey.

La figura 8 muestra los resultados de la simulación; la señal de entrada al intercalador *entrada* es procesada a través de los arreglos de retrasos (delay) en cada rama, para lo cual se obtiene una reorganización de los símbolos. Esta nueva señal generada del módulo intercalador *trans* presenta la data en forma desordenada, ésta ha sido suministrada al desintercalador; por ser este módulo inverso en arquitectura al intercalador produce la reestructuración de los símbolos al formato original con retraso de un símbolo. En la señal recuperada *salida* estos resultados validan el diseño obteniendo la respuesta esperada.

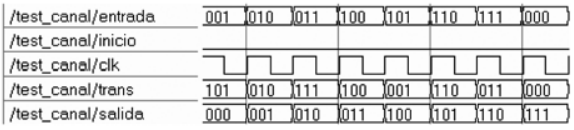

Figura 8. Simulación de la etapa intercalador-desintercalador.

Tabla 5. Resumen de recursos sintetizados del intercalador/desintercalador.

Logic Utilization	Used	Available	Utilization
Total Number Slice Registers	17	4,704	1%
Number used as Flip Flops	14		
Number used as Latches	3		
Number of 4 input LUTs	20	4,704	1%
Logic Distribution			
Number of occupied Slices	18	2,352	1%
Number of Slices containing only related logic	18	18	100%
Number of Slices containing unrelated logic	0	18	0%
Total Number of 4 input LUTs	32	4,704	1%
Number used as logic	20		
Number used as Shift registers	12		
Number of bonded IOBs	10	142	7%
IOB Latches	3		
Number of GCLKs	1	4	25%
Number of GCLKIOBs	1	4	25%
Total equivalent gate count for design	1,816		

Una vez realizada la simulación se obtiene la síntesis del circuito sobre el FPGA, como se muestra en la tabla 5, donde podemos ver que solo se han empleando un 1% de los slices, solo 1% de las Lut Tables, 7% de los pines de entrada/salida y 25% de las I/O de reloj.

CONCLUSIONES

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.

Estos diseños pueden ser programados en VHDL y ser sintetizados en un FPGA, lo que permite diseños "Sistemas sobre Chip" donde múltiples módulos pueden ser combinados en un solo circuito integrado [18].

El análisis matemático es un excelente complemento didáctico para la enseñanza de la teoría de las comunicaciones digitales, ya que al emplear este tipo de fundamentos logra consolidar la comprensión de los conceptos básicos y las características principales de los códigos.

Podemos observar el diseño modular en la descripción de los componentes, empleado previamente por distintos autores entre ellos [12-13], donde el algoritmo de decodificación comprende módulos de funciones matemáticas [6]. La descripción en VHDL de estos módulos es apropiada para implementar aplicaciones tanto para códigos correctores de errores simples como concatenados, dando la alternativa de codiseño modular, ampliaciones y optimización.

REFERENCIAS

[1] C.E. Shannon. "A Mathematical Theory of Communication". Bell System Technical Journal. Vol. 27, pp. 379-423. 1948.

[2] I.S. Reed-Solomon. "Polynomial Codes over Certain Finite Fields". Journal of the Society for Industrial and Applied Mathematics. Vol. 8 Nº 2, pp. 300-304. 1960.

[3] C. Sandoval y A. Fedón. "Programación VHDL de algoritmos de codificación para dispositivos de hardware reconfigurable". Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería. Vol. 24 Nº 1. Marzo 2008.

[4] S.A. Pérez, E. Soto y S. Fernández. "Diseño de sistemas digitales con VHDL". Editorial Thomson. España. 2002.

[5] IEEE Computer Society. "IEEE Standard VHDL Language Reference Manual". The Institute of Electrical and Electronics Engineers, Inc. Mayo 2002.

[6] M. Vera, G. Verajano y J. Velasco-Medina. "Diseño de funciones DSP usando VHDL y FPGAs". Grupo de Bioelectrónica y Nanoelectrónica. EIEE. Universidad del Valle. Colombia.

[7] C.E. Sandoval Ruiz y Antonio Fedón. "Codificador y decodificador digital Reed-Solomon programados para hardware reconfigurable". Ingeniería y Universidad. Vol. 11 Nº 1, pp.17-31. Junio 2007.

[8] Xilinx Blockset Reference Guide. Xilinx System Generator v2.1 for Simulink. 2007.

[9] F.J.M. López. "Diseño de transmisor y receptor para redes inalámbricas W-MAN". Tesis para optar al grado de doctor. Departamento de Comunicaciones. Málaga. 2005. URLs: http://www.coit.es/pub/ficheros/p068_resumen_vodafone_d51dc759.pdf

[10] IEEE802-Compatible Viterbi Decoder v1.1, 2004.

[11] M.J. Castiñeira y G. Farrell Patrick. "Codificación para el control de errores". 2004.

[12] A. Saqib Nazar. "Implementación Eficiente de Algoritmos Criptográficos en Dispositivos de Hardware Reconfigurable". Tesis para optar al grado de doctor. Departamento de Ingeniería Eléctrica. México. 2004.

[13] K.C. Chang. "Digital Systems Design with VHDL and Synthesis, An Integrated Approach". IEEE Computer Society. USA. 1999.

[14] "Digilab DIO2 Reference Manual". 2002. URLs: www.digilentinc.com

[15] A. Agatep. "Reed-Solomon Solutions with Spartan-II FPGA". WP110 (v1.1). February 10, 2000.

[16] F. Carpio. "VHDL Lenguaje para descripción y modelado de circuitos". Departamento de Ingeniería Informática. Universidad de Valencia. España. 1997.

[17] Ulloa V., Fernando. "Contribución al estudio de un modelo de canal aeronáutico para sistemas de radiocomunicación digital terrestre basados en plataformas estratosféricas HAPS". Tesis para optar al grado de doctor. Barcelona, España. 2003.

[18] A. Arriagada. "FEC (Forward Error Correction) y Código Reed-Solomon". Universidad de Concepción. Concepción, Chile. 2001.