Simulación e Implementación en FPGA de un Esquema de Codificación del Canal sujeto al Estándar de Wimax

José Andrés Marzo Icaza[1], Rebeca Estrada[2]
Facultad de Ingeniería en Electricidad y Computación (FIEC)
ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL (ESPOL)
Campus Gustavo Galindo, vía Perimetral Km. 30.5, Apartado 09-01-5863, Guayaquil, Ecuador
andres_marzo25@hotmail.com[1], restrada@espol.edu.ec[2]

Resumen

El presente trabajo describe el diseño, simulación e implementación en una FPGA de un Codificador de Canal para Wimax, enfocándose en el estándar IEEE 802.16-2004 el cual representa la implementación fija y forma parte de la investigación en el campo de redes de acceso fijo inalámbrico de banda ancha sin línea de vista. El trabajo presenta las principales características usadas en Wimax para la transmisión y recepción además del funcionamiento de cada uno de los bloques usados para la corrección de errores.

Este proyecto presenta una implementación en FPGA empleando diseño basado en modelo, usando el software System Generator junto a Matlab y Simulink, para poder obtener los datos que nos permitan comprobar el funcionamiento del diseño propuesto de acuerdo a las especificaciones del estándar de Wimax y además analizar la capacidad de corrección de errores mediante el uso de curvas BER vs SNR y de las constelaciones a la salida del canal para justificar el uso del sistema diseñado.

Palabras Claves: Wimax, FEC, codificador de canal, FPGA, System Generator.

Abstract

This paper describes the design, simulation and implementation on a FPGA of a channel encoder for Wimax, focusing on the IEEE 802.16-2004 standard which represents the fixed implementation and it is part of research in the field of fixed wireless access networks Broadband without line of sight. The paper presents the main features used in Wimax for transmitting and receiving also the operation of each of the blocks used for error correction.

This project presents an implementation using FPGA-based design model, using the System Generator software with Matlab and Simulink, to obtain data that allow us to verify the operation of the proposed design according to the WiMAX standard specifications and also analyze the ability for error correcting using BER vs. SNR curves and constellations to the channel output to justify the use of the system design.

Keywords: *Wimax, FEC, channel encoder, FPGA, System Generator.*

1. Introducción.

Desde que comenzaron a usarse redes WLAN estos han evolucionado para poder soportar cada vez más aplicaciones y tasas de datos superiores. Como consecuencia de esta evolución nació Wimax (Worldwide Interoperability for Microwave Access), con el estándar IEEE-802.16 [1]. Wimax soporta comunicaciones con o sin línea de vista y resulta una solución para sistemas de última milla en lugares donde los costos de implementación y mantenimiento de tecnologías como DSL no justifiquen su inversión.

El uso de herramientas programables reconfigurables, como la FPGA, son un buen auxilio para el diseño de sistemas para Wimax. Las ventajas que ofrecen estas son las siguientes [2]:

- **Velocidad de procesamiento:** Wimax posee requerimientos superiores en lo que se refiere al manejo de las tasas de datos y la velocidad de procesamiento del sistema.
- **Flexibilidad:** Wimax es una tecnología que está en constante evolución, por lo que es necesario que la FPGA sea flexible para poder reprogramar el diseño hasta obtener el producto final.
- **Mercado:** Al ser Wimax una tecnología emergente, el tiempo que los productos lleguen al mercado es un factor clave. Las herramientas de desarrollo para las FPGA, y la existencia de Cores ya diseñados y optimizados, logran que el tiempo de desarrollo sea menor.

2. Fundamentos Teóricos.

El desarrollo de este proyecto se realizó en base al estándar 802.16-2004 donde las características a considerarse en el diseño son mostradas a continuación.

2.1. Etapas Generales del estándar

Para realizar este trabajo se considera WirelessMAN-OFDM que utiliza una transformada rápida de Fourier de 256 puntos y opera en la banda de frecuencia de 2 a 11 GHz. El estándar fijo de Wimax provee servicio para un área de 5 Km permitiendo una máxima tasa de datos de 70 Mbps con un ancho de banda del canal de 20MHz, ofreciendo a los usuarios conectividad de banda ancha sin la necesidad de tener línea de vista con la estación base.

La corrección de errores (Forward Error Correction, FEC), es realizada en 2 fases. Primero pasando por un codificador exterior Reed-Solomon y luego por un codificador interior Convolucional. El codificador Reed-Solomon se encarga de corregir errores de ráfaga a nivel de byte, lo cual es bien útil en la presencia de propagación en multicamino. El codificador convolucional corrige errores independientes de bits. Funcionalidades de puncturing se aplican en el codificador convolucional para variar la capacidad de corrección de errores.

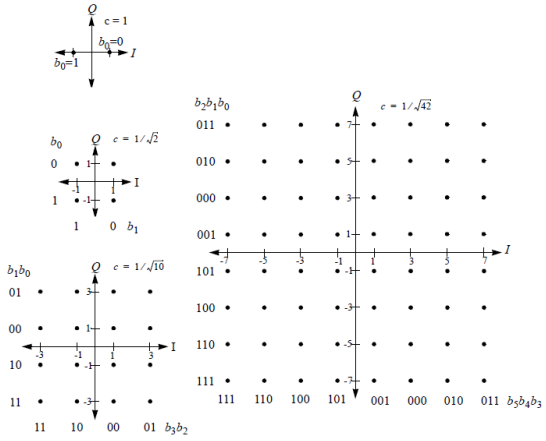

Figura 1. Constelaciones usadas en el estándar IEEE 802.16-2004

Tabla 1. Perfiles de modulación y codificación para el estándar IEEE-802.16-2004

ID	Modulación	Bloque de entrada (bytes)	Bloque de salida (bytes)	Tasa de codificación total
0	**BPSK**	12	24	1/2
1	**QPSK**	24	48	1/2
2	**QPSK**	36	48	3/4
3	**16-QAM**	48	96	1/2
4	**16-QAM**	72	96	3/4
5	**64-QAM**	96	144	2/3
6	**64-QAM**	108	144	3/4

Las modulaciones especificadas tanto para el enlace downlink como el uplink, son BPSK, QPSK, 16-QAM y 64-QAM y se muestran en la figura 1. Las opciones de FEC son emparejadas con cada esquema de modulación. El estándar especifica siete combinaciones de modulación y tasa de codificación, las cuales se seleccionan para dar comunicación a cada usuario, lo que es conocida como modulación adaptativa. La tasa de codificación se refiere a la relación de los datos sin codificar por los datos

codificados, por lo se tendrá que hacer una decisión si lo que se quiere es tener una tasa de datos alta o una mayor robustez del sistema. La tabla 1 muestra las combinaciones descritas.

2.2. Entorno de Diseño y Metodología.

Para llegar a la implementación del sistema se sigue el diseño basado en modelo a nivel de sistema como se ilustra en la figura 2 [3]. El entorno escogido para realizar la implementación de este proyecto, es el software System Generator de Xilinx en conjunto con Matlab y Simulink. Además, para realizar la experimentación en hardware, se utiliza la tarjeta de prototipado de Digilent Spartan 3E-Starter, la cual tiene la FPGA donde se va a implementar el diseño y donde se han realizado todas las pruebas para verificar su funcionamiento.

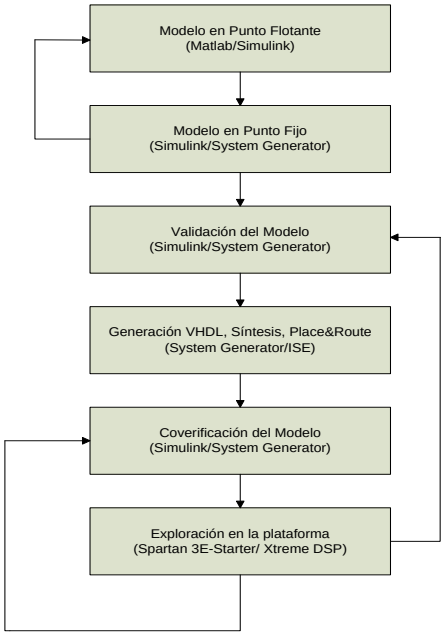

Figura 2. Metodología usada en el diseño basado en modelo

3. Codificación y decodificación del Canal.

El proceso de codificación del canal se encarga de tener una transmisión fiable en la comunicación de los datos, protegiendo a estos de degradaciones causadas por el canal. Para lograrlo se añade información redundante a los datos transmitidos de manera que el decodificador pueda interpretarlos para corregir los errores causados por el medio en que son transmitidos.

Esto se realiza por medio de un FEC el cual detecta y corrige la información recibida.

3.1. Codificación.

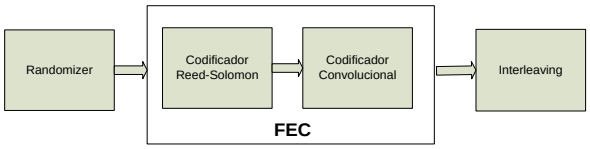

Figura 3. Diagrama de Bloques del Codificador del Canal

El estándar especifica el diagrama de bloques de la figura 3 para el codificador del canal.

El Randomizer se utiliza para evitar largas secuencias de unos y ceros. Se implementa utilizando un registro de desplazamiento con retroalimentación lineal (LFSR), utilizando el polinomio $1 + x^{14} + x^{15}$.

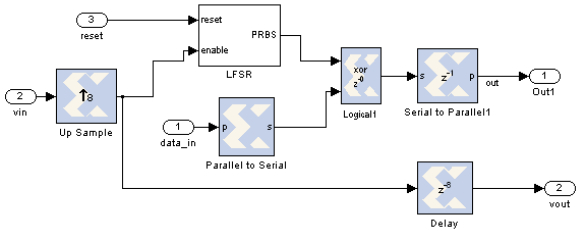

Figura 4. Implementación del Randomizer.

La salida del randomizer debe ser concatenada con un byte 0x00 (zero tailing) el cual se encarga de inicializar el codificador convolucional en cada ráfaga de datos.

Para la etapa del FEC, primero se tiene un codificador de Reed-Solomon el cual trabaja a nivel de bytes. Se utiliza un codificador nativo RS(255,239,8) y se utiliza shortening y puncturing para variar la capacidad de corrección del código en base al perfil de codificación usado, como se muestra en la tabla 2. Los polinomios generador del código y generador del campo son expresados a continuación:

$$g(x) = (x - \alpha^0)(x - \alpha^1)(x - \alpha^2) \dots (x - \alpha^{2t-1}) \quad \textbf{Ec. 1}$$

$$p(x) = x^8 + x^4 + x^3 + x^2 + 1 \quad \textbf{Ec. 2}$$

Tabla 2. Variaciones al Codificador de Reed-Solomon

id	Bloque de entrada k'	Bloque de salida n'	Símbolos que pueden ser corregidos t'
1	32	24	4
2	40	36	2
3	64	48	8
4	80	72	4
5	108	96	6
6	120	108	6

La implementación viene dada en la figura 5, donde el primer bloque se encarga del shortening y la codificación de los datos (figura 6), el segundo bloque realiza el proceso de puncturing (figura 7), y finalmente se reordenan los datos para transmitir los símbolos de paridad antes de los bytes de información[1].

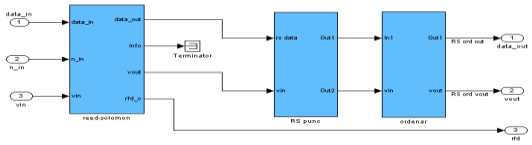

Figura 5. Codificación Reed-Solomon.

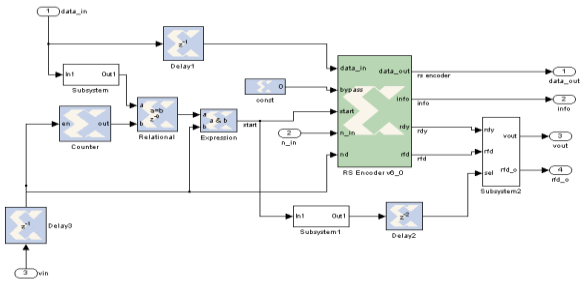

Figura 6. Codificación y Shortening.

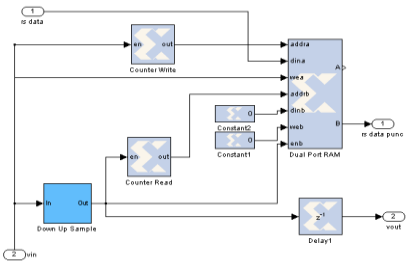

Figura 7. Puncturing del Codificador de Reed-Solomon.

[1] *Se utiliza el diseño usado en el interleaver para el reordenamiento de los datos.*

Para la etapa del puncturing se utiliza una memoria RAM de doble puerto haciendo que la velocidad de lectura sea diferente al de la escritura modificándola con un bloque de *Down Sample* seguido de un *Up Sample.*

Después de tener el bloque codificado, se convierten los datos de paralelo a serial para ser procesados por un codificador convolucional nativo con tasa de codificación ½ y una longitud de restricción de 7 utilizando los siguientes polinomios:

$$G_1 = 171_{oct} \quad para\ X \qquad \textbf{Ec. 3}$$

$$G_1 = 133_{oct} \quad para\ Y \qquad \textbf{Ec. 4}$$

Se utiliza puncturing variable para ajustar la capacidad de corrección de errores del codificador, para lo cual se elimina una cierta cantidad de bits tal como está especificado en la tabla 3, donde además se muestra la distancia libre del código d_{free} que representa por medio de la siguiente ecuación la capacidad de corrección de errores del codificador convolucional[4].

$$t = \frac{d_{free}-1}{2} \qquad \textbf{Ec. 5}$$

Tabla 3. Vectores de Punturing para el Codificador Convolucional

	Tasas de Codificación			
Tasa	$^1/_2$	$^2/_3$	$^3/_4$	$^5/_6$
d_{free}	10	6	5	4
X	1	10	101	10101
Y	1	11	110	11010
XY	X_1Y_1	$X_1Y_1Y_2$	$X_1Y_1Y_2X_3$	$X_1Y_1Y_2X_3Y_4X_5$

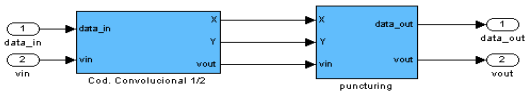

Figura 8. Codificación Convolucional.

La figura 8 muestra la implementación del codificador convolucional en System Generator, donde el primer bloque (figura 9) contiene el diseño del codificador nativo y el segundo bloque la etapa de puncturing (figura 10) que dependerá del perfil de codificación escogido.

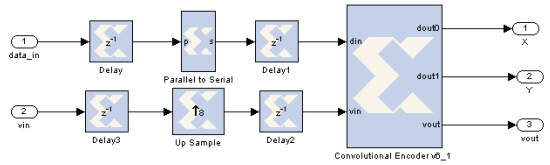

Figura 9. Codificador convolucional nativo.

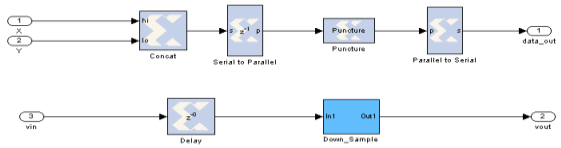

Figura 10. Puncturing usado en el codificador convolucional.

.

Luego de pasar los datos por la etapa del FEC, estos pasan por una etapa de interleaver (entrelazador) el cual se encarga de aleatorizar la posición en que se localizan los errores. Este proceso se realiza por medio de 2 permutaciones la cuales son generadas en Matlab y se muestran a continuación:

```
%interleaver
%Ncbps=192(BPSK),384(QPSK),768(16-QAM),1152(64-QAM)
%Ncpc=1BPSK),2(QPSK),4(16QAM),6(64QAM)
k = 0 : Ncbps - 1;
mk=(Ncbps/12)*mod(k,12)+floor(k/12);
s = ceil(Ncpc/2);
jk=s*floor(mk/s)+mod(s,mk+Ncbps-floor(12*mk/Ncbps));
[s1,int_idx]=sort(jk);
s = ceil(Ncpc/2);
%deinterleaver:
[s2,dint_idx] = sort(int_idx);
```

Figura 11. Permutaciones del interleaver en Matlab.

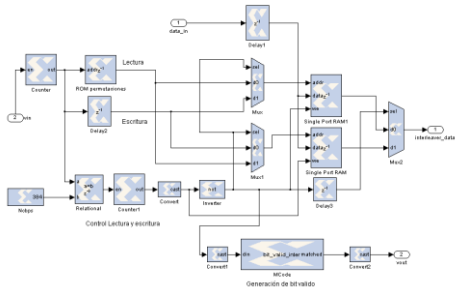

Figura 12. Diseño del Interleaver.

Para el interleaver se utilizan 2 memorias RAM, las cuales se alternan para realizar el proceso de escritura y lectura de los datos, donde este último paso se realiza leyendo la posición de los datos de una memoria ROM que contiene el vector generado en Matlab (int_idx) luego de realizar las 2 permutaciones descritas anteriormente.

3.2. Decodificación.

La decodificación se encarga de recuperar la información transmitida dado un bloque de símbolos que contiene información redundante más los errores causados por el canal. Se asume una demodulación de los datos usando decisión soft con 8 niveles de cuantización (3 bits). Se sigue el diagrama de bloques de la figura 13.

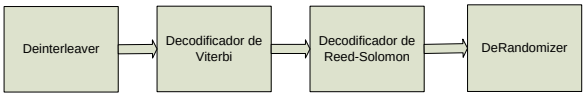

Figura 13. Diagrama de bloques del decodificador del canal.

Para el deinterleaver se sigue el mismo diseño descrito en la figura 12, donde el único cambio seria cambiar el vector de inicialización de la memoria ROM (dint_idx).

Para el diseño del decodificador de Viterbi se sigue el proceso inverso al usado en el codificador convolucional. En las figuras 14 y 15 se ilustra este proceso. Se utiliza el mismo vector de puncturing usado en la codificación, Añadiendo un símbolo que indique una mayor probabilidad de haber recibido un cero, y etiquetando ese símbolo nulo con un bit adicional para que pueda ser interpretado por el bloque del decodificador de Viterbi.

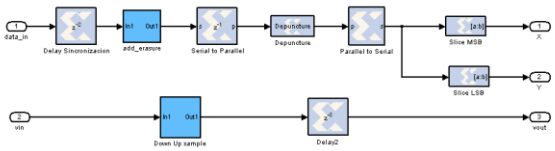

Figura 14. Depunturing usado en el decodificador de Viterbi.

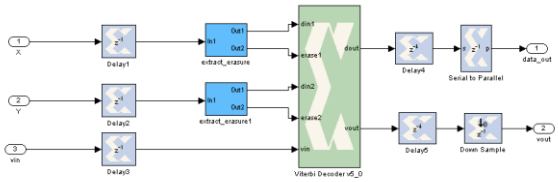

Figura 15. Decodificación de Viterbi.

Una vez que se decodifican los datos, se convierten los datos a un formato de 8 bits para que puedan ser procesados por el decodificador de Reed-Solomon.

Para la decodificación de Reed-Solomon se sigue el proceso inverso, donde para reordenar los símbolos solo se cambio el vector de inicialización de la ROM.

Para la etapa de depunturing se tiene que agregar bytes ceros para reemplazar los símbolos eliminados en el codificador y agregar además un bit en el instante de tiempo en que son agregados.

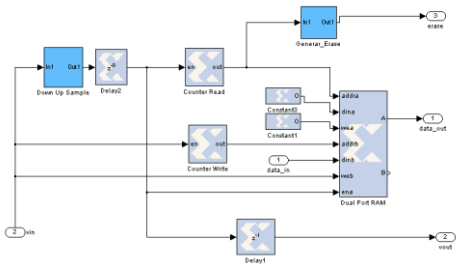

Figura 16. Depunturing del decodificador de Reed-Solomon.

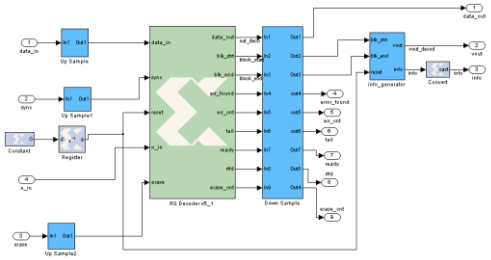

Figura 17. Decodificación de Reed-Solomon.

Finalmente, para el derandomizer, se utiliza el mismo diseño del codificador.

4. Implementación en FPGA.

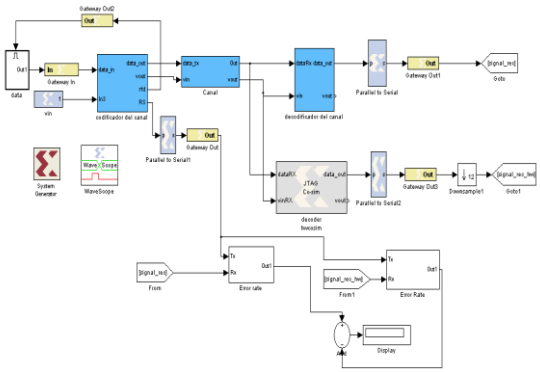

Figura 18. Verificación en FPGA.

Tanto el codificador como el decodificador fueron implementados en una spartan 3E xc3s500e-4fg320 y se analizó su funcionamiento usando hardware cosimulation (figura 18). Se presenta en la tabla 4 los recursos utilizados en hardware considerando el perfil de codificación 2 (QPSK ¾).

Tabla 4. Recursos de la FPGA usando perfil de codificación QPSK ¾

Etapa	Slice	Slice FF	LUT	IOB	BRAM
Randomizer	34	61	34	20	0
Reed-Solomon	214	285	285	20	4
Cod. Convolucional	66	115	74	12	0
Interleaver	26	36	51	5	3
Deinterleaver	22	34	43	9	3
Viterbi	1667	1516	2379	7	4
Dec. Reed-Solomon	1627	1678	2907	12	9

5. Análisis de Resultados.

5.1. Variación del BER según el SNR.

Para analizar la capacidad de corrección de errores del sistema diseñado se analizan las curvas de BER vs SNR en presencia o ausencia de codificación.

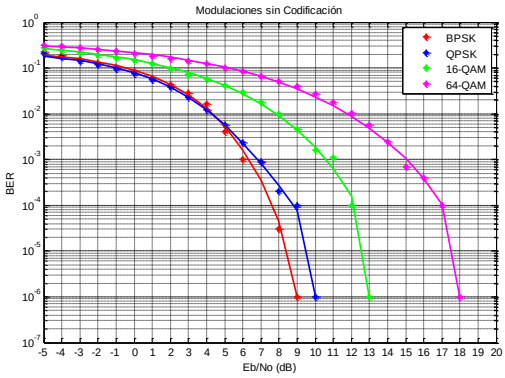

Figura 19. Comparacion de las modulaciones usadas en Wimax sin usar codificación.

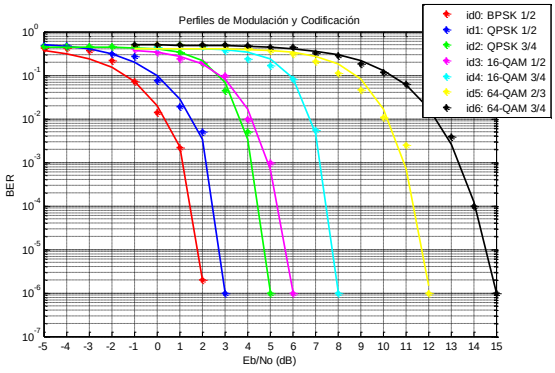

Figura 20. Resultados de la Simulación de los Perfiles de Codificación y Modulación.

Como se puede observar comparando las figuras 19 y 20, el uso de la codificación de canal mejora el BER para cada una de las modulaciones usadas.

5.2. Comparación de Constelaciones a la salida del canal.

Para analizar los beneficios del codificador del canal, se muestra en la figura 21 un grafico de la constelación de una señal 64-QAM a la salida de un canal AWGN con un SNR dado para tener un BER de 10^{-3}, donde el grafico de la izquierda representa el caso sin codificación y el de la derecha usa una codificación con tasa 2/3.

Se observa en la gráfica que con el uso de la codificación del canal se necesita un SNR menor para tener el mismo desempeño al que se tiene en el caso que no se usa codificación.

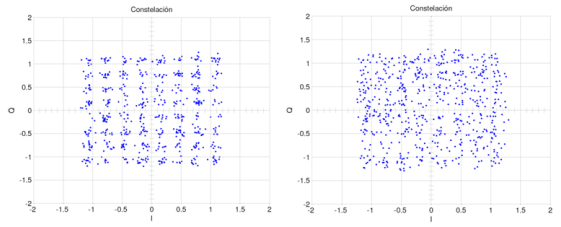

Figura 21. Comparación de constelaciones a la salida del canal.

5.3. Verificación del estándar.

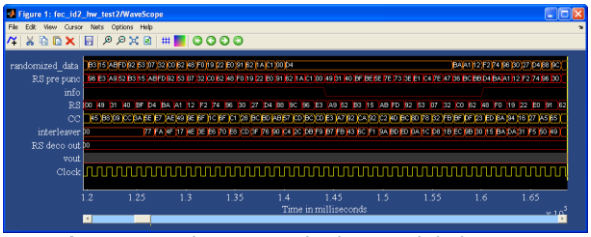

Figura 22. Diagramas de tiempo del sistema diseñado.

Para verificar que el diseño propuesto cumple las especificaciones del estándar IEEE 802.16-2004, se utilizaron los vectores de prueba que exige el mismo para analizar los datos a la salida de cada bloque, lo cual se muestra en el diagrama de tiempo de la figura 22.

6. Conclusiones.

Con el desarrollo de este trabajo se ha podido comprobar las ventajas de utilizar un ambiente de desarrollo de Hardware de alto nivel, con la ayuda del software System Generator de Xilinx. En base a las simulaciones realizadas usando Matlab y Simulink se concluye que el diseño del codificador cumple con las características especificadas en el estándar 802.16, al obtener la secuencia de bits deseados a la salida en base al vector de prueba utilizado como entrada. Con las graficas BER vs SNR obtenidas, se demuestra la capacidad de detección y corrección de errores del sistema. Así mismo al visualizar las constelaciones de las señales a la salida del canal, se puede tener una idea de que tan eficaz es el codificador usado.

7. Recomendaciones.

Se recomienda para poder usar este diseño en conjunto con otras partes de la transmisión y recepción de Wimax, el uso de una FPGA de mayores prestaciones.

8. Referencias.

[1] Instituto de Ingenieros Eléctricos y Electrónicos, Estándar IEEE-802.16-2004, http://standards.ieee.org, 2004

[2] Lopez F. Diseño de Transmisor y Receptor para redes Inalámbricas WMAN, Universidad de Málaga, 2005.

[3] Serra M. Prototipado Rápido de la Capa Física de OFDM: Hiperlan2, Universidad Autónoma de Barcelona, 2005.

[4] Sklar B. Digital Communications Fundamentals and Applications, Segunda Edición, Prentice Hall, 2001

[5] Roca A. Implementation of a WiMAX Simulator in Matlab, Eingereicht an der Technischen Universität Wien, 2006.