Decodificador Turbo MAP de varias iteraciones

Richard Chávez*, Gabriel Gamarra*, Mauricio Pardo**

Resumen

Este artículo presenta la descripción de un decodificador Turbo usando algoritmos MAP en una FPGA mediante el uso de VHDL. El objetivo general de este trabajo se relaciona con lograr la síntesis física del algoritmo en términos de densidad lógica y velocidad de procesamiento y, con ello, mostrar la respuesta de dicho decodificador ante la variación de dos parámetros básicos, que son el número de iteraciones de decodificación y el tamaño de la trama de datos. Se analizan las opciones del algoritmo MAP y se exponen los resultados de síntesis obtenidos de la herramienta Quatrus II de Altera Corp. y se presentan curvas de rendimiento del decodificador bajo la influencia de un canal simulado donde la fuente de perturbación elegida es ruido blanco gaussiano aditivo. Finalmente, se presentan las conclusiones y recomendaciones derivadas del proyecto.

Palabras claves: Algoritmo Log-MAP, algoritmo Bahl, Cocke, Jelinek y Raviv (BCJR), Decodificador SISO (*Soft Input Soft Output*), Decodificación Turbo, Razón de Verosimilitud Logarítmica (LLR).

Abstract

This article presents the description of a Turbo decoder with MAP algorithm in an FPGA using VHDL. The main objetive is related with achieving the algorithm synthesis in terms of logic density and processing speed and with this, show the decoder's response to the variations of two basics parameters as the number of decoding iterations and the size of the data frames.The options of the MAP algorithm have been analysed and the results of the synthesis obtained with Quartus II are exposed as well as the performance curves of the decoder under the influence of Additive White Gaussian Noise. Finally, the conclusions and recommendations derived from the project are enunciated.

Keywords: Log-MAP algorithm, Bahl, Cocke, Jelinek and Raviv algorithm (BCJR), SISO (*Soft Input Soft Output*) decoder, Turbo Decoding, Log Likelihood Ratio (LLR).

Fecha de recepción: 12 de octubre de 2005
Fecha de aceptación: 27 de abril de 2006

*Estudiante de Ingeniería Electrónica, Universidad del Norte.
**Ingeniero Electrónico, Universidad del Norte.
Dirección: Universidad del Norte, Km 5 vía a Puerto Colombia, A.A. 1569, Barranquilla (Colombia).

1. INTRODUCCIÓN

Con la aparición de los códigos Turbo se logró disminuir aún más la brecha existente entre el límite teórico de codificación de canal propuesto por Shannon y el desempeño de los dispositivos reales. La clave de este tipo de sistemas es utilizar esquemas híbridos (parte sistemática y parte no sistemática convolucionada) y entrelazados para aleatorizar los parámetros de ruido en medios inalámbricos [4].

El algoritmo de máxima probabilidad *a posteriori* (MAP), conocido también como algoritmo BCJR por las iniciales de sus autores, fue formalmente presentado en 1974 por Bahl *et al.* como una alternativa para la decodificación de códigos convolucionales. La decodificación con algoritmos MAP ha tenido un resurgimiento desde el descubrimiento de los codificadores Turbo en 1993. Los decodificadores MAP realizan una decisión *símbolo a símbolo* óptima, y además proveen "salidas suaves", que consisten en una versión cuantificada de la decisión del decodificador que puede ser vista como la probabilidad de dicha decisión, características necesarias en sistemas de decodificación concatenada como los Turbo [2].

La decodificación Turbo se realiza de forma iterativa, es decir, la salida suave del proceso es realimentada y utilizada como entrada en la siguiente iteración de decodificación. Las iteraciones, aunque aumentan la latencia del sistema, reducen considerablemente la potencia requerida para alcanzar un mejor desempeño y aumentar la eficiencia de los receptores.

En la Universidad del Norte (Barranquilla, Colombia) se desarrolló con anterioridad una tesis en la que se sintetizó, en *hardware*, un decodificador Turbo, y se obtuvieron buenos resultados desde el punto de vista de un primer prototipo, que, sin embargo, poseía limitantes como la velocidad de procesamiento y algunas características que no permitían obtener valores óptimos en la decodificación. Con la adopción de algoritmos MAP como alternativa de decodificación se busca suplir algunas de las deficiencias del esquema previo.

La sección 2 presenta la metodología utilizada en el diseño del decodificador y el enfoque del mismo. La 3 enuncia las especificaciones tenidas en cuenta en el desarrollo del diseño. La 4 describe los componentes del prototipo, tanto la parte *software* como la *hardware*. En la 5 se presentan los resultados obtenidos en las pruebas del dispositivo, así como los resultados de la síntesis física. Finalmente se dan a conocer las conclusiones realizadas con base en las pruebas realizadas y la comparación con las curvas teóricas.

2. ENFOQUE Y METODOLOGÍA

El proyecto fue concebido para realizarse en *software* y *hardware*. La parte *software* consiste en una interfaz de usuario que incluye un codificador Turbo, un simulador de canal ruidoso y permite modificar algunos parámetros de dichos módulos. Además cuenta con un cuantificador que entrega los datos cuantificados a la parte *hardware*. Finalmente, permite observar el rendimiento del decodificador a través de un medidor de curvas de desempeño. La parte *hardware* consiste en la tarjeta con el decodificador Turbo.

El decodificador Turbo se realiza con base en el algoritmo MAP. Este algoritmo es un esquema de decodificación de códigos convolucionales, símbolo a símbolo, que utiliza las probabilidades de los datos sistemáticos y de paridad para estimar si un dato es un uno o un cero a través de la razón de verosimilitud logarítmica.

La metodología utilizada consistió en dividir el proyecto en varias etapas de acuerdo con la herramienta utilizada. MATLAB[1] permitió la puesta a prueba del sistema para probar el rendimiento experimental de forma computacional antes de realizar la síntesis física. HDL Design Series[2] permitió la descripción a nivel de diagrama de bloques para luego ser convertida a VHDL. La etapa de síntesis física junto con la de simulación y programación del dispositivo se hizo mediante Quartus II de Altera[3]. Esta herramienta permite comprobar los parámetros de tiempo que requiere el sistema y la asignación de pines. Además se usó C++ Builder de Borland[4] para realizar el desarrollo del codificador y la interfaz gráfica que controla los parámetros requeridos y genera las gráficas de BER Vs Eb/No con los resultados obtenidos.

Para el desarrollo del decodificador se han considerado los recursos disponibles en la Universidad del Norte, los cuales permitirán la verificación y la realización de la síntesis física del decodificador. Dada la necesidad de fabricar una tarjeta, también se debe tener en cuenta las tecnologías disponibles, además de su costo. Se ha escogido la FPGA CYCLONE (EP1C12Q240C6) de Altera, que ofrece 12.060 elementos lógicos y 52 bloques de memoria de 4 kbits, soporta hasta 300MHz de frecuencia y tiene un bajo costo. Las características básicas del sistema se han desarrollado teniendo en cuenta que se deseaba que la

[1] www.mathworks.com
[2] www.menthor.com
[3] www.altera.com
[4] www.borland.com

síntesis se pudiera realizar en un único chip, y se logró que las unidades de almacenamiento necesarias quedaran como bloques internos para favorecer el desempeño del decodificador. Se escogió Altera por la disponibilidad de herramientas en los laboratorios de la Universidad y el soporte en línea que incluye *white papers* y *MegaCore Functions.*

3. ESPECIFICACIÓN DEL SISTEMA

La etapa de codificación de canal utilizada en el proyecto implementa un codificador Turbo con razón de 1/3, a cuya salida le es aplicada ruido generado con un simulador de canal AWGN. En la etapa de decodificación se utiliza un cuantificador de 4 bits, el cual entregará al sistema la información suave proveniente de los datos recibidos. Todos estos bloques son desarrollados en *software*.

El sistema posee una interfaz gráfica en la que se escogen las diferentes opciones, ya sea tamaño del bloque, el número de iteraciones o el nivel de ruido del canal. Es en esta misma interfaz que se puede observar el rendimiento de la decodificación con base en el número de iteraciones de decodificación y el nivel de potencia seleccionados. También es posible especificar el número de veces que se desea realizar las pruebas, esto con el fin de alcanzar un número que permita dar validez estadística a los resultados.

La etapa de decodificación se realiza a través de la aplicación de un algoritmo Log-MAP, que es la parte que se sintetiza físicamente. El entrelazador utilizado se realizó de acuerdo con la especificación 25.222 del 3GPP, el cual es un entrelazador pseudoaleatorio. El sistema maneja tres tamaños de bloques de procesamiento que son 64, 128 y 256 bits. Además es capaz de realizar de 1 hasta 10 iteraciones.

La medición del BER se realiza de vuelta en el PC cuando el decodificador entrega los datos.

4. DESCRIPCIÓN DEL DISEÑO

En la figura 1 se observa el diagrama general donde se muestran los módulos *hardware* y *software* que constituyen el sistema.

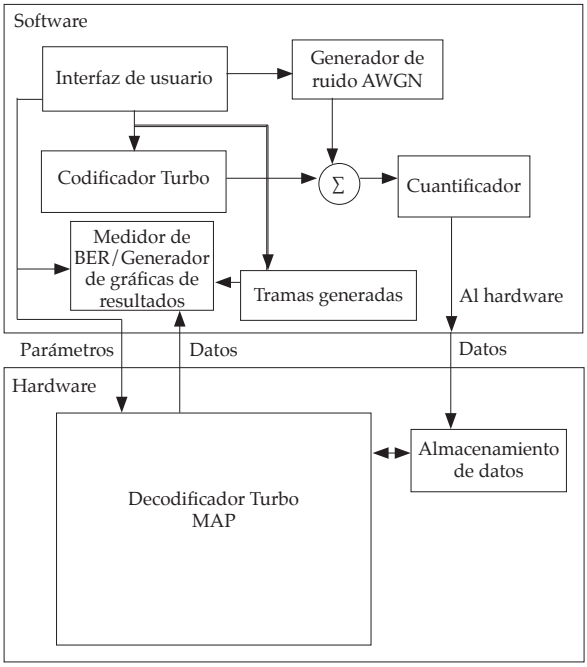

Figura 1. Diagrama general del sistema

A. *Codificador*

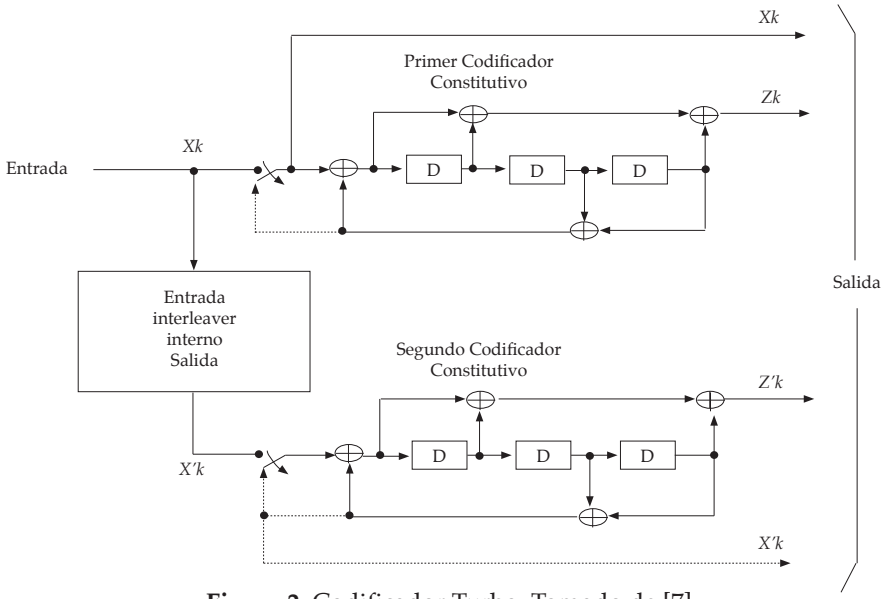

Figura 2. Codificador Turbo. Tomado de [7]

El codificador Turbo consta de dos codificadores convolucionales conectados a través de un entrelazador. La salida del codificador tiene tres componentes: los datos sistemáticos, los datos convolucionales del primer codificador y los datos convolucionales del segundo codificador. Estos últimos se obtienen al colocar en la entrada de ese codificador los datos sistemáticos entrelazados.

B. *Generador de ruido*

El ruido agregado a la señal codificada se asume blanco aditivo gaussiano, y se realizó con base en un generador de números aleatorios con distribución gaussiana, media cero y varianza

$$\alpha^2 = \frac{N_0}{2E_b r}$$

Donde r es la tasa de codificación utilizada, en este caso 1/3, y Eb/No se expresa en unidades lineales. La selección del ruido blanco gaussiano permite generalizar la respuesta del decodificador, pues a pesar de que los decodificadores Turbo están orientados a la corrección de errores tipo ráfaga, la presencia de los entrelazadores permite aleatorizar los patrones de ruido.

C. *Cuantificador*

Esta es la primera etapa de la recepción. La etapa de cuantificación toma los valores obtenidos a la salida del generador de ruido y le asigna un valor entre 16 posibles. De esa manera se obtiene una estimación suave del dato manipulado.

D. *Decodificador*

Los bloques constitutivos del decodificador se muestran en la figura 3. Cabe anotar que la sincronización de la comunicación entre los diversos bloques se realiza desde la unidad de control del sistema.

El decodificador Turbo consta de dos decodificadores *Soft Input Soft Output* (SISO) trabajando en paralelo y conectados a través de un bloque entrelazador y otro de-entrelazador.

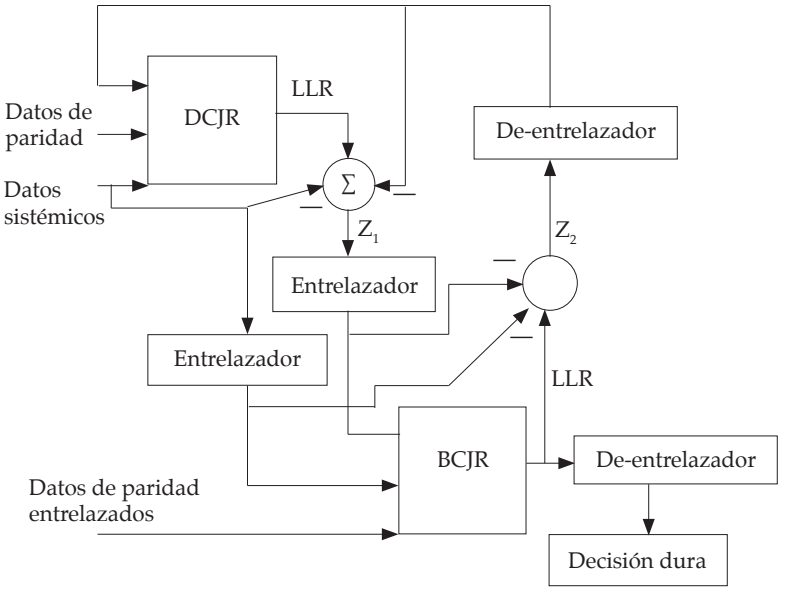

Figura 3. Esquema del decodificador Turbo

Cada bloque SISO entrega un estimado de la trama recibida. Dicho estimado es conocido como información extrínseca, y es utilizada por el otro decodificador como un parámetro de referencia en su decisión en la siguiente iteración de decodificación.

El primer bloque SISO recibe los datos sistemáticos junto con los de paridad y la información extrínseca del segundo decodificador. Éste recibe los datos de paridad correspondientes a los datos sistemáticos entrelazados, la información extrínseca del primer decodificador y la salida del entrelazador de datos sistemáticos.

E. *Maximum a Posteriori Probability (*MAP*)*

El bloque decodificador MAP está constituido por cuatro funciones principales: *alfa, beta, gamma, lambda*. La función *Gamma* realiza el cálculo de la métrica de rama, que consiste en estimar la probabilidad de que el dato recibido pertenezca a una de las transiciones del *trellis*. *Alfa* realiza la estimación hacia delante de la métrica de estados, es decir, calcula la probabilidad de la relación del estado actual del *trellis* con el estado anterior. *Beta* realiza la estimación hacia atrás de la métrica de estados, calculando las relaciones entre los estados futuros y los actuales del *trellis*, y *Lambda* calcula la razón de verosimilitud logarítmica (LLR),

que consiste en la relación entre la probabilidad de que el dato recibido sea un uno y la probabilidad de que sea un cero, para, con base en ella, calcular la información extrínseca del decodificador. Un esquema del decodificador MAP se puede observar en la figura 4.

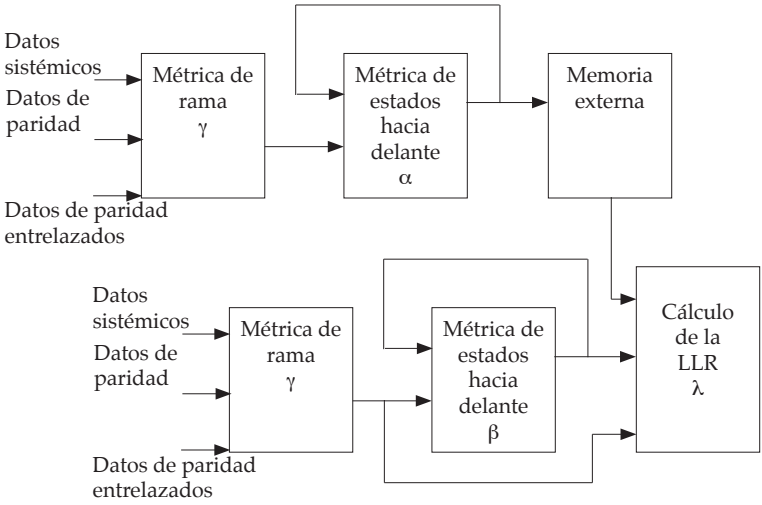

Figura 4. Diagrama de bloques del algoritmo MAP

Para este diseño se utilizó una versión simplificada del algoritmo MAP llamada MAX-*LOG*-MAP, la cual trabaja en el dominio logarítmico, permitiendo así reducir las multiplicaciones incluidas en el proceso en sumas, y las adiciones en simples operaciones de maximización, con lo que se consigue una reducción muy significativa de la complejidad del sistema. Si bien existen otras versiones del algoritmo MAP, se prefiere la opción que requiere las operaciones matemáticas más simples, pues la síntesis física se realiza en dispositivos lógicos programables.

El bloque *Gamma* calcula la métrica de rama con base en los datos sistemáticos, los datos de paridad y la información extrínseca proveniente del otro decodificador.

Los bloques *Alfa* y *Beta* están constituidos por unidades básicas de suma, comparación y selección (SCS). Esta unidad se encarga de seleccionar el mayor entre las dos cantidades comparadas. Las estimaciones hacia delante y hacia atrás de la métrica de estados se realizan de forma recursiva con base en la métrica de rama y la métrica de estado del estado anterior.

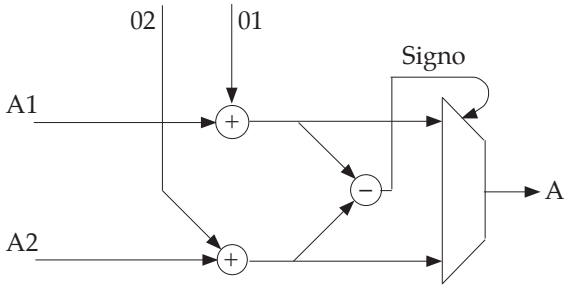

Figura 5. Unidad SCS (Suma, Comparación, Selección)

Cada SCS ocupa un máximo de 56 celdas lógicas. Con ello se consigue que los bloques *Alfa* y *Beta* consuman 390 y 300 celdas lógicas respectivamente. La diferencia entre ellos se debe al valor inicial que debe tener la estimación hacia delante.

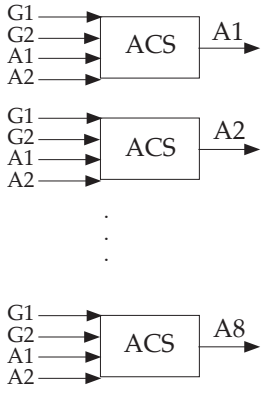

Figura 6. Estimación hacia delante constituida por unidades SCS

Una vez realizada la estimación hacia delante, los resultados deben almacenarse en memoria mientras se realiza la estimación hacia atrás. Para agilizar el proceso, a medida que se va realizando la estimación hacia atrás se lee la memoria en la que está almacenado el resultado de la estimación hacia delante, en forma invertida, es decir, primero el último dato, luego el antepenúltimo, etc. De esta manera se puede calcular la LLR de manera casi simultánea con el cálculo de la estimación hacia atrás, con lo cual se evita almacenar los resultados de dicha estimación.

El bloque del cálculo de la LLR es el más crítico del diseño en cuanto a consumo de elementos lógicos. Éste ocupa 1.063 celdas lógicas. Su tamaño se debe en gran parte a la resolución utilizada en el diseño con el fin de no perder información que pudiera resultar valiosa en la decodificación. Su función es hallar el mayor entre las sumas de las métricas relacionados con las transiciones de estado generadas por un uno y aquellas generadas por un cero, y calcular la diferencia entre ellas.

La información extrínseca de cada decodificador se obtiene al restar de la razón de verosimilitud logarítmica el dato sistemático correspondiente y la información extrínseca del decodificador correspondiente a la iteración anterior.

La información extrínseca es utilizada para la estimación de la métrica de rama en el otro decodificador en la siguiente iteración de decodificación.

F. *Entrelazador y De-entrelazador*

El bloque entrelazador se realizó con base en la especificación 25.222 del 3GPP. Para ello se calcularon las posiciones finales de los datos entrelazados y se almacenaron en un vector. Dicho vector de posiciones se utilizó para escribir los datos en una memoria de forma que el almacenamiento permitiera ordenar los datos de la misma manera que terminarían al someterlos al proceso de entrelazado, pero sin los retardos que implica el cálculo de las permutaciones propias del proceso. El de-entrelazador se realizó de forma similar, sólo que en ese caso el vector de posiciones utilizado contiene las posiciones finales del vector desentrelazado. De esa manera se logra que los bloques de entrelazador y de-entrelazador consuman solamente recursos de memoria y no elementos lógicos del dispositivo programable. El proceso de escritura y lectura de dichas memoria debe ser administrado por una unidad de control, que maneje la sincronización de todos los procesos incluidos en el sistema.

5. RESULTADOS

Las pruebas del prototipo se realizaron utilizando la interfaz de usuario. A través de ella se seleccionaron la relación $\mathbf{E_b/N_0}$ del canal, el tamaño de las tramas y el número de iteraciones de decodificación. También es posible seleccionar el número de tramas que se va a enviar con el fin de conseguir medidas estadísticamente válidas.

La intención de las pruebas es demostrar la influencia de la variación de la longitud del bloque codificado y del número de iteraciones de decodificación sobre el rendimiento del decodificador.

A. *Rendimiento del decodificador*

La tabla 1 muestra los resultados obtenidos al decodificar tramas de 64 bits de longitud. La tasa de error de bit (BER) correspondiente se muestra en la tabla 2, acompañados del error de la medición.

Tabla 1
Número de errores de decodificación para bloques de 64 bits

E_b/N_o	Varianza	Número de bits	Número de iteraciones				
0	1,50	1280000	341863	340754	339413	337991	333308
1	1,19	1280000	235071	201345	193516	184189	170211
2	0,95	2560000	198566	83856	65752	52083	38987
3	0,75	5120000	76290	6544	2912	1853	1307
4	0,60	10240000	27557	238	68	64	48

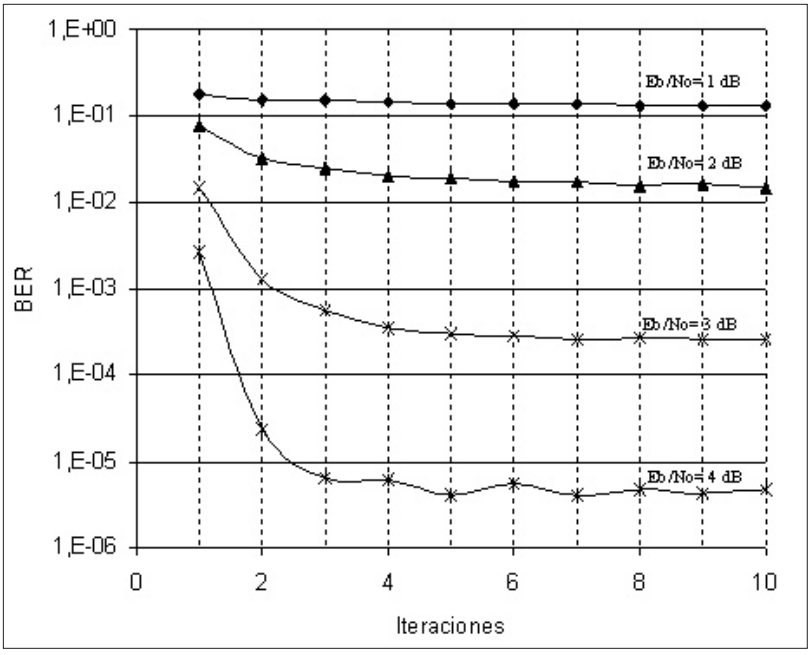

Figura 7. Convergencia de las iteraciones de 64 bits

En la figura 7 se aprecia que el incremento en el número de iteraciones para E_b/N_o bajas no mejora el desempeño del decodificador. También se observa que a partir de la tercera y la cuarta iteración no hay una disminución significativa en el BER.

En la figura 8 se puede observar el efecto de las iteraciones en el desempeño del decodificador. A medida que aumenta el número de iteraciones la curva de desempeño se desplaza a la izquierda, lo cual quiere decir que se puede alcanzar un BER determinado a una menor E_b/N_o.

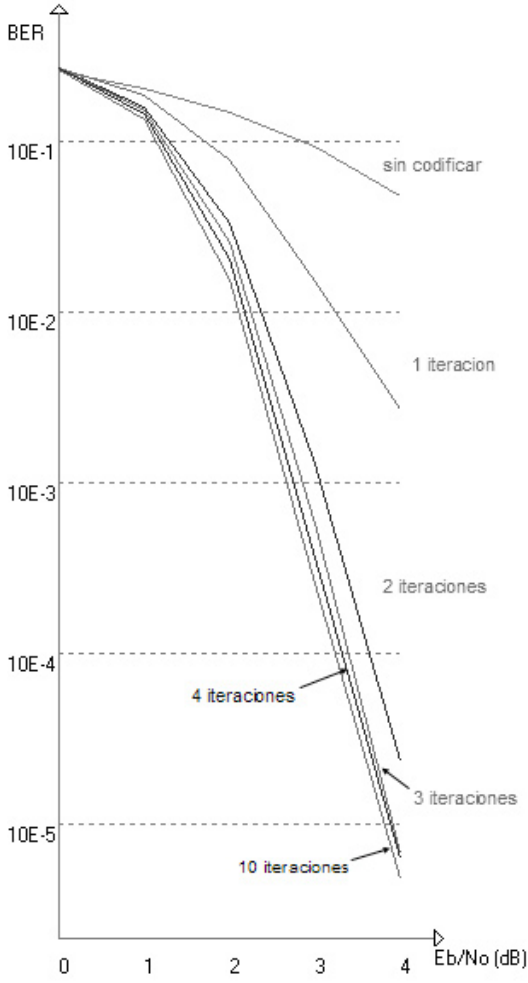

Figura 8. Resultados para bloques de 64 bits

Tabla 2

Tasa de error de bit y error porcentual de la medición para bloques de 64 bits

E_b/N_o	Número de bits	Número de iteraciones									
		1		2		3		4		10	
		BER	Error (%)	BER	Error (%)	BER	Error (%)	BER	Error (%)	BER	Error (%)
0	1280000	2,7 E-01	0,15	2,7 E-01	0,15	2,7 E-01	0,15	2,6 E-01	0,15	2,6 E-01	0,15
1	1280000	2,8 E-01	0,19	1,6 E-01	0,20	1,5 E-01	0,21	1,4 E-01	0,22	1,3 E-01	0,23
2	2560000	7,8 E-02	0,22	3,3 E-02	0,34	2,6 E-02	0,38	2,0 E-02	0,43	1,5 E-02	0,50
3	5120000	1,5 E-02	0,36	1,3 E-03	1,24	5,7 E-04	1,85	3,6 E-04	2,32	2,6 E-04	2,77
4	10240000	2,7 E-03	0,60	2,3 E-05	6,48	6,6 E-06	12,13	6,3 E-06	12,50	4,7 E-06	14,43

La tabla 3 muestra los resultados obtenidos con bloques de 256 bits. La tasa de error de bit se presenta en la tabla 4.

Tabla 3

Número de errores de decodificación para bloques de 256 bits

E_b/N_o	Varianza	Número de bits	Número de iteraciones				
			1	2	3	4	10
0	1,50	1280000	350482	354984	354424	353230	351844
1	1,19	1280000	239806	212450	206192	198739	183004
2	0,95	2560000	191949	61035	30037	17552	6140
3	0,75	5120000	58506	1495	246	99	71
4	0,60	10240000	10312	21	13	14	15

En la figura 9 se observa un comportamiento similar al de la figura 7. La figura 10 muestra los resultados correspondientes a la tabla 4. En esta gráfica se puede apreciar una mayor ganancia entre iteraciones que la visible en la figura 8. Además, las iteraciones alcanzan un BER más bajo que en el caso de 64 bits.

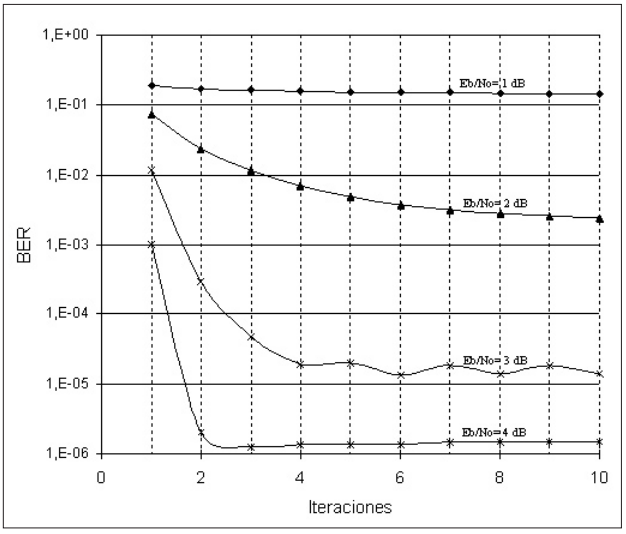

Figura 9. Convergencia de las iteraciones para 256 bits

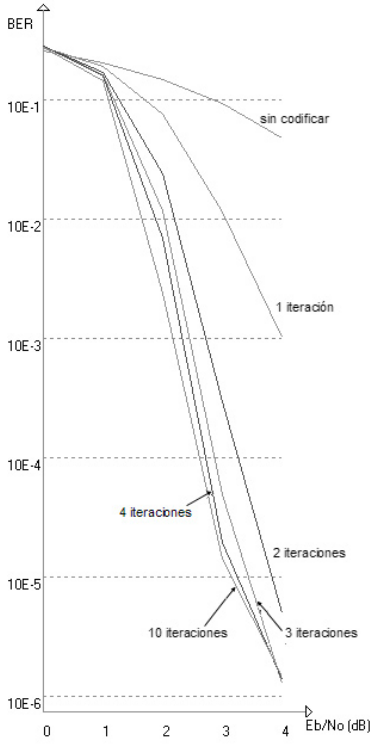

Figura 10. Resultados para bloques de 256 bits

Tabla 4
Tasa de error de bit para bloques de 256 bits

E_b/N_o	Número de bits	Número de iteraciones									
		1		2		3		4		10	
		BER	Error (%)	BER	Error (%)	BER	Error (%)	BER	Error (%)	BER	Error (%)
0	1280000	2,7 E-01	0,14	2,8 E-01	0,14	2,8 E-01	0,14	2,8 E-01	0,14	2,7 E-01	0,14
1	1280000	1,9 E-01	0,18	1,7 E-01	0,20	1,6 E-01	0,20	1,6 E-01	0,21	1,4 E-01	0,22
2	2560000	7,5 E-02	0,22	3,4 E-02	0,40	1,2 E-02	0,57	6,9 E-03	0,75	3,4 E-03	1,27
3	5120000	1,1 E-02	0,41	2,9 E-04	2,59	4,8 E-05	6,38	1,9 E--05	10,05	1,4 E-05	11,87
4	10240000	1,0 E-03	0,98	2,1 E-06	21,82	1,3 E-06	27,73	1,4 E-06	26,73	1,5 E-06	25,82

Se puede observar que al aumentar el tamaño del bloque codificado también mejora el desempeño del decodificador. Esto se debe al aumento en la aleatoriedad del código producida por el entrelazador. Este hecho se puede apreciar en la figura 11.

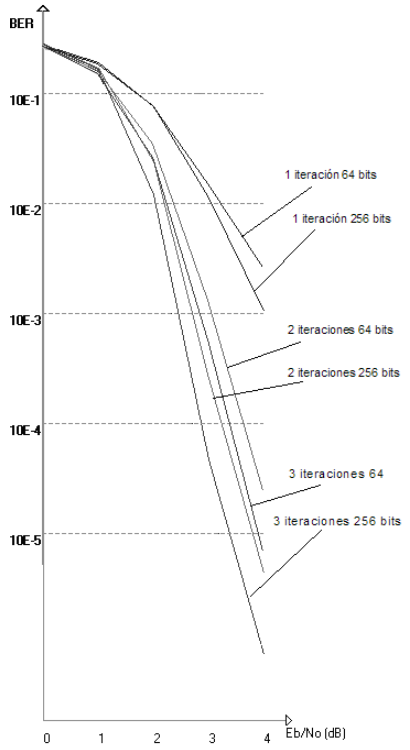

Figura 11. Comparación de resultados

Los resultados obtenidos fueron comparados con simulaciones como la presentada en la figura 12, obtenida por Dr. Mathew Valenti en [4]. Esta gráfica presenta resultados obtenidos para diferentes longitudes de tramas usando el mismo codificador. La decodificación se realizó con 14 iteraciones. Los resultados obtenidos se presentan en la tabla 5.

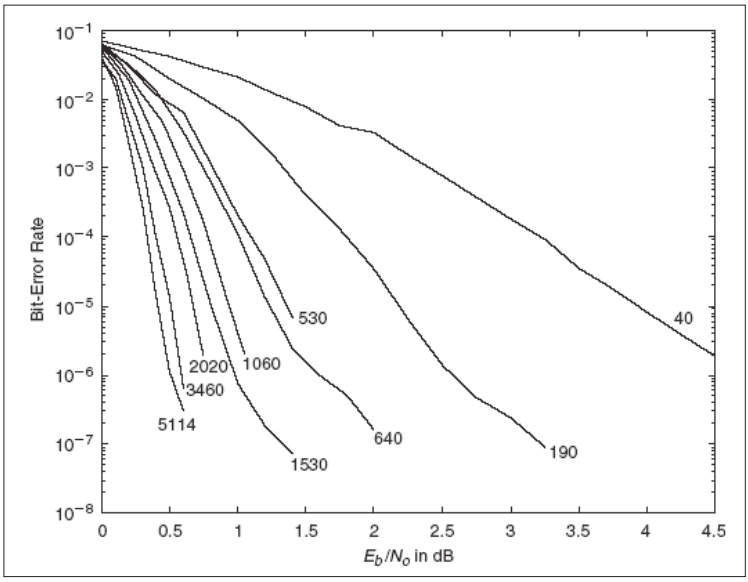

Figura 12. Desempeño del decodificador Turbo para diferentes longitudes de trama. Tomada de [4]

Tabla 5
Comparación de valores simulados y valores obtenidos

Longitud de la trama	BER objetivo=10E-3	BER objetivo=10E-5
40	2.41	3.93
64	**2.50**	**3.40**
190	1.34	2.18
256	**2.10**	**3.00**
530	0.82	1.36
640	0.75	1.24
1060	0.59	0.94
1530	0.48	0.80
2020	0.38	0.68
3460	0.30	0.51
5114	0.24	0.42

Al superponer las curvas obtenidas para 10 iteraciones con los resultados de la simulación (figura 13), se observa que los resultados obtenidos conservan una tendencia similar a la de los teóricos, a pesar de que la gráfica de 256 bits no consigue superar el resultado de la de 190 bits. Esto se debe a que además de que el estudio mencionado es una simulación por computador y este proyecto es una implementación *hardware*, en éste se utiliza una versión menos precisa del algoritmo MAP que puede introducir pérdidas de alrededor de 0.5 dB hasta 1 dB [6].

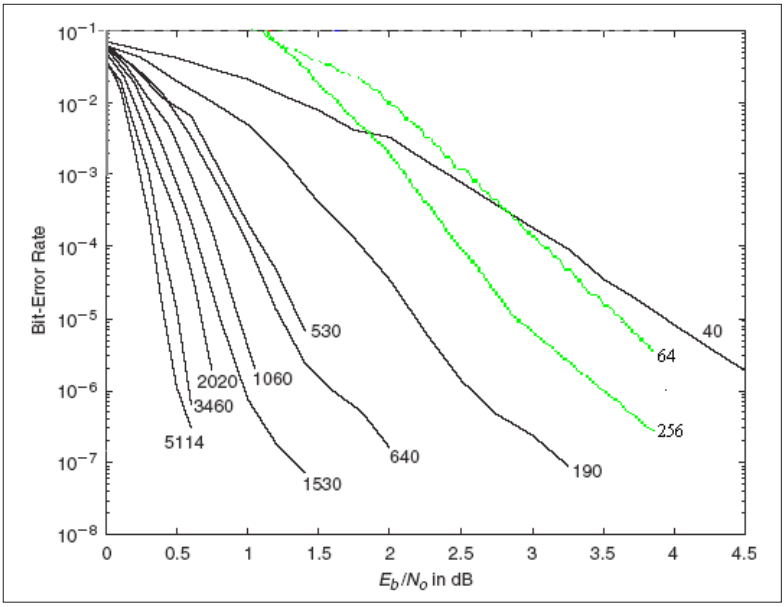

Figura 13. Comparación con valores simulados

B. *Resultados de síntesis*

La síntesis con Quartus II de Altera produjo los siguientes resultados:

Tabla 6
Reporte del ajustador de Quartus II

Recurso	Usado	Disponible	Porcentaje
Pines de E/S	45	173	26%
Elementos lógicos	6.049	12.060	50%
Bits de memoria	72000	239.616	30%
M4Ks	34	52	65%

Máxima frecuencia de reloj: 71.15 MHz.

El algoritmo BCJR ocupa 1.711 celdas lógicas, mientras que la unidad de control ocupa 2.316. El número de bits de memoria se debe principalmente al almacenamiento de la métrica de estados hacia delante. Ésta requiere un total de 57.344 bits de memoria.

El proceso de decodificación tarda aproximadamente 10 us por iteración para el caso de la trama más larga, 256 bits. Es decir que para realizar 10 iteraciones se necesitan 100 us, utilizando un reloj de 50 MHz.

CONCLUSIONES

Se ha demostrado que el desempeño del decodificador Turbo puede acercarse más al límite de Shannon siempre y cuando el número de iteraciones de decodificación aumente. De lo contrario, si el número de iteraciones se mantiene bajo, como fue el caso de [5], en el que el número de iteraciones era uno, el desempeño del decodificador alcanza una tasa de error aceptable a un nivel de E_b/N_o relativamente alto.

Se comprueba que el tamaño del entrelazador afecta en gran medida la calidad de la decodificación, siendo que a una longitud mayor se alcanza la misma tasa de error de bit a un nivel de potencia más bajo [8].

El desempeño del decodificador mejora a medida que aumenta el número de iteraciones de decodificación. Sin embargo, la ganancia de una iteración a la siguiente disminuye a medida que el número de iteraciones se incrementa.

Teniendo en cuenta que cada iteración introduce un retardo en la decodificación se puede establecer un número mínimo de iteraciones que permita conseguir una ganancia determinada en la decodificación. Por ejemplo, tres

iteraciones de decodificación son suficientes para alcanzar una ganancia de 1.5 dB con respecto a los resultados de la primera iteración.

Referencias

[1] BOUTILLON, E, GROSS, W. and GULAK, G. VLSI architectures for the MAP algorithm.
[2] GROSS, WJ. and GULAK P., G. Simplified MAP Algorithm Suitable for Implementation of Turbo Decoders. www.eecg.toronto.edu/~wjgross/wjg/MAP.pdf
[3] HAYKIN, S. *Communication Systems*. 4ª ed. (cap. 11, pp. 675-715, 721-722). New York, John Wiley & Sons, 1993.
[4] VALENTI, M. and SUN, J. *Handbook of RF and Wireless Technologies*.
[5] DITTA, Y. y PARDO, M. Codificación de canal para UTRA-TDD. Tesis de grado, 2001.
[6] VALENTI, M. *Iterative Detection and Decoding for Wireless Communication*.
[7] Third Generation Partnership Project (3GPP). Multiplexing and channel coding (TDD) [online]. Valbonne, Francia: 3GPP, 1999, update May 2000. 37 p.: il. (3G TS 25.222). Disponible en Internet: <ftp://ftp.3gpp.org/Specs> en Adobe Portable Document Format y documento Word.
[8] BERROU, C., GLAVIEUX, A. and THITIMAJSHIMA, P. Near Shannon limit. Error correcting coding and decoding [online]. <http://gladstone.systems.caltech.edu/EE/Courses/EE127/EE127B/handout/berrou.pdf>