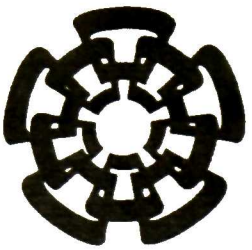
BC-649
Don.-2011

xx(179112.1)



Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional
Unidad Guadalajara

Diseño e implementación Digital del Turbodecodificador para el estándar de comunicaciones celulares 3GPP

Tesis que presenta:

Joaquin Adrian Espino Orozco

para obtener el grado de:

Maestro en Ciencias

en la especialidad de:

Ingeniería Eléctrica

Directores de Tesis

Dr. Ramón Parra Michel
Dr. Luis Fernando González Pérez

CINVESTAV del IPN Unidad Guadalajara, Guadalajara, Jalisco, Febrero de 2011

Diseño e implementación Digital del Turbodecodificador para el estándar de comunicaciones celulares 3GPP

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
Joaquin Adrian Espino Orozco
Ingeniero Electrónico
Instituto Tecnológico de Morelia 2003-2007

Becario de Consejo Nacional de Ciencia y Tecnologia , expediente no. 219058

Directores de Tesis
Dr. Ramón Parra Michel
Dr. Luis Fernando González Pérez

CINVESTAV del IPN Unidad Guadalajara, Febrero de 2011.

Agradecimientos:

A mis padres por su cariño.

A mi hermano por sus concejos y ayuda.

A mis asesores Ramón Parra y Luis Fernando González por su apoyo y paciencia.

Al CINVESTAV por el uso de sus instalaciones.

A CONACYT por el apoyo económico.

Resumen

Los turbo códigos son una técnica de codificación de canal que nos permite proteger la información que es enviada a través de un canal de comunicaciones. Este esquema de codificación genera una baja tasa de errores, muy cercana a los límites teóricos propuestos por C. Shannon [1]. Por esta razón, en la actualidad han entrado en su etapa de implementación y comercialización al estar siendo integrados en estándares de comunicaciones inalámbricas y móviles. En otras palabras, los turbo códigos han alcanzado ya una madurez tal que se empiezan a integrar en dispositivo y estándares comerciales.

En este trabajo de tesis se presenta el diseño e implementación en hardware de un turbo decodificador reconfigurable, para el estándar WCDMA. Se presenta una propuesta de arquitectura utilizando ventanas deslizantes, optimizando el uso de memoria. Además por medio de modificaciones matemáticas se propone la eliminación de un módulo interleaver, que representa una reducción importante en la estructura del hardware.

Durante el desarrollo de la tesis, se presentan diferentes algoritmos utilizados en la turbo decodificación, se analizan las características del estándar a implementar, se generan los requerimientos funcionales de acuerdo con el estándar, además de los requerimientos no funcionales, posteriormente se describe cada uno de los componentes de la arquitectura propuesta, por último se realizan el proceso de verificación y validación.

Los resultado del desempeño son comparados con los turbo decodificadores comerciales.

Abstract

The turbo Codes are a channel coding method, that allows us to protect information that is sent through communications channel. This coding scheme achieves a low error rate, very close to Shannon [1] theoretical limits. At present time, they are in the implementation and commercialization stage, being integrated into current and emerging wireless and mobile communications standards. In other words, turbo codes have enough maturity to be integrated into devices and commercial standards.

In this thesis work we present the design and implementation hardware of a reconfigurable turbo decoder for WCDMA standard. The architecture is based on the memory efficient sliding windows technique. In addition, using mathematical modifications to the turbo decoding algorithm, one of the interleaver blocks can be eliminated, archiving an important reduction in the hardware complexity with very small performance degradation.

In the development of this work, different variants of the algorithms were studied; we analyzed the characteristics of the standard, to generate the technical design specifications as well as the non-functional requirements, to subsequently describe each of the components in the architecture. Finally the verification and validation processes are carried out successfully.

Performance results are compared with commercial turbo decoders.

Contenido

Índice de Figuras

Índice de Tablas

Lista de Acrónimos

3G	3rd Generation
3GPP	3rd Generation Partnership Project
3GPP2	3rd Generation Partnership Project 2
ADC	Analog to Digital Converter
ASIC	Application Specific Integrated Circuit
APP	A Posteriori Probability
ARQ	Automatic Repeat Request
AWGN	Additive White Gaussian Noise
AMR	Adaptive Multi Rate
BEC	Backward Error Correction
BER	Bit Error Rate
BM	Branch Metric
BPSK	Binary Phase Shift Keying
BW	Backward
CDMA2000	Code Division Multiple Access 2000
DUT	Device Under Test
ECC	Error Correction Code
FDD	Frequency Division Duplexing
FEC	Forward Error Correction
FIFO	First Input First Output
FPGA	Field Programable Gates Array
FW	Forward
GSM	Global System for Mobile Communications
HDL	Hardware Description Language
HPSK	Hybrid Phase Shift Keying
IEEE	Institute of Electrical and Electronics Engineers
IP	Intellectual Property

LLR	Log Likelihood Ratio
LUT	Lookup Table
MAP	Maximum A posteriori Probability
PCCC	Parallel Concatenated Convolutional Codes
QPSK	Quadrature Phase Shift Keying
RSC	Recursive Systematic Convolutional
SISO	Soft Input- Soft Output
SNR	Signal to Noise Ratio
UMTS	Universal Mobile Telecommunications System
VHDL	VHSIC Hardware Description Language
WCDMA	Wideband Code Division Multiple Access
WiMAX	Worldwide Interoperability for Microwave Access

Capítulo I

Introducción

1.1. Justificación

La llegada de la comunicación remota por medios eléctricos y electrónicos, trajo nuevas formas de comunicarse, permitiendo de algún modo acortar las distancias, dando como resultado que la información de la vuelta al mundo en cuestión de segundos. Estos sistemas de comunicación no serían de gran utilidad si los mensajes que se trasmiten no fueran correctos o poco confiables; hace apenas unas décadas esta confiabilidad se obtenía aumentando la potencia de transmisión, utilizando más recursos de los necesarios (más ancho de banda) o bien disminuyendo la velocidad de transmisión.

Estas pérdidas de potencia y ancho de banda, ha llevado a buscar formas de proteger la información del ruido y las distorsiones que pueda sufrir durante la transmisión. De esta forma, enviar la mayor cantidad de información utilizando la menor cantidad posible de recursos, nos trae como consecuencia comunicaciones económicas y accesibles. En la figura 1.1 se muestra un modelo básico de un sistema de comunicación, en donde se puede observar que el canal introduce perturbaciones al mensaje.

El estadounidense Claude Elwood Shannon, reconocido como el padre de la teoría de la información en su trabajo Teoría Matemática de la Comunicación (*The mathematical theory of communication*) publicado en 1948 [1], establece que a todas las fuentes de información se las puede dimensionar mediante un parámetro llamado cantidad de información. A esta cantidad de información se le puede relacionar con la capacidad de un canal de comunicación. Si la cantidad de información de la fuente no sobre pasa la capacidad del canal, una comunicación confiable es posible. Con esto se dieron los primero pasos para los métodos de corrección de errores.

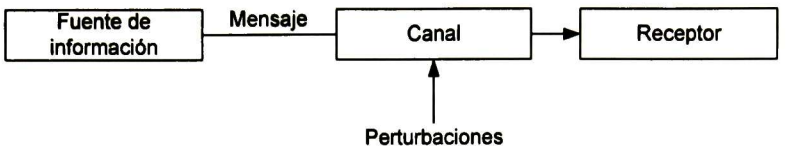


Figura 1.1 Modelo básico de un sistema de comunicación

La codificación de canal permite realizar la detección y/o corrección de errores, los cuales permiten acercarse a los límites teóricos establecidos por Shannon, lo que significa una menor cantidad de errores en la transmisión, usando la menor potencia posible. La codificación de canal consiste en agregar información (redundancia) a los datos a transmitir; esta información adicional será utilizada por el receptor para detectar y/o corregir los errores generados durante la transmisión. Con esto es posible cometer un menor número de errores, o bien tener la capacidad de detectarlos [2].

Existen dos métodos de codificación de canal conocidos como:

- BEC (Backward Error Correction): corrección de error hacia atrás también conocido como ARQ (Automatic Repeat Request), que consiste en la detección del error y solicitar la repetición de la información.
- FEC (Forward Error Correction): corrección de errores hacia adelante, que consiste en corregir la información errónea solamente a partir de la información recibida.

Dentro del método FEC se encuentran los códigos en bloques, códigos convolucionales y los códigos concatenados, dentro de los cuales se encuentran los turbo códigos.

Los turbo códigos fueron propuestos por Claude Berrou y Alain Glavieux a principios de los años 90 [4]. Este nuevo tipo de codificación consiste en la concatenación de dos codificadores sistemáticos recursivos (RSC, Recursive Systematic Convolutional) unidos entre sí por un entrelazador mejor conocido como interleaver. En el caso del decodificador se tiene el mismo esquema en donde cada decodificador toma ventaja del trabajo realizado por el decodificador anterior. El desempeño de este tipo de decodificadores depende en gran parte de la capacidad de entrelazado, así que para un entrelazado complejo en un bloque grande, el desempeño de un turbo decodificador se acerca a los límites teóricos expuestos por Shannon. El impacto de los turbo códigos es tan

grande que se ha creado una era en lo referente a las técnicas usadas en la codificación de canal.

Los turbo códigos se han implementado en los sistemas de telefonía móvil en la llamada tercera generación, también conocidos como el Sistema de Telecomunicaciones Móviles Universal (UMTS, Universal Mobile Telecommunication System), en los cuales se encuentra el estándar americano CDMA2000 (TIA/EIA-2002.2) y el estándar europeo GSM ó WCDMA (3GPP TS 25.212).

Como ya se mencionó, las aplicaciones de los turbo códigos son muy extensas, esto genera una gran necesidad sobre su investigación, desarrollo e implementación de este tipo de códigos. En la actualidad es posible encontrar estándares de turbo códigos implementados en Hardware (ASIC ó FPGA) como propiedad intelectual IP (Intellectual Property) de diferentes fabricantes.

1.2. Antecedentes

1.2.1. Teoría de la información

Las bases para las comunicaciones digitales modernas fueron establecidas por Shannon, con su llamada teoría de la información, en la cual trata de cuantificar la información y las capacidades del canal por donde se va a transmitir dicha información. Él dimensionó el canal de comunicación como un ducto de tres dimensiones, la relación de señal a ruido o potencia de transmisión, la velocidad de transmisión y el ancho de banda. Con estos tres factores Shannon fue capaz de darle al canal de comunicación una capacidad, que representa los límites de la cantidad de información que se puede enviar. Esta relación se expresa mediante la fórmula de Shannon-Hartley [1].

$$C = B\, log_2\left(1 + \frac{S}{N}\right) \frac{bits}{sec}. \qquad (1.1)$$

En la ecuación 1.1, C, representa la velocidad máxima a la que se puede efectuar la transmisión, B es el ancho de banda que tiene disponible el canal de comunicación y S/N es la relación señal a ruido o SNR (Signal Noise Ratio). Shannon mencionó que era posible recibir la información enviada con un índice de errores tan bajo como se desee, siempre y cuando la velocidad de transmisión (r_b) fuese menor que la capacidad del canal (C). De lo contrario no será posible realizar una transmisión con baja probabilidad de error. Además Shannon mencionó que la tendencia de estos límites se cumplen mediante el uso de

códigos que permitían al receptor tener una menor probabilidad de error. A esto se le llamo codificación de canal, que es la parte en donde se centra este trabajo de tesis.

En la figura 1.2, se ilustra un diagrama de comunicación digital en donde se señalan los bloques pertenecientes a la codificación de canal, así como la interacción con el resto de los bloques del sistema de comunicación. Los datos que entran al codificador de canal pueden venir de un codificador de fuente; el trabajo de este bloque es justo lo contrario al codificador de canal, trata de eliminar la información redundante que contengan los datos a transmitir; de este modo se requieren menos bits para enviar el mismo mensaje [3].

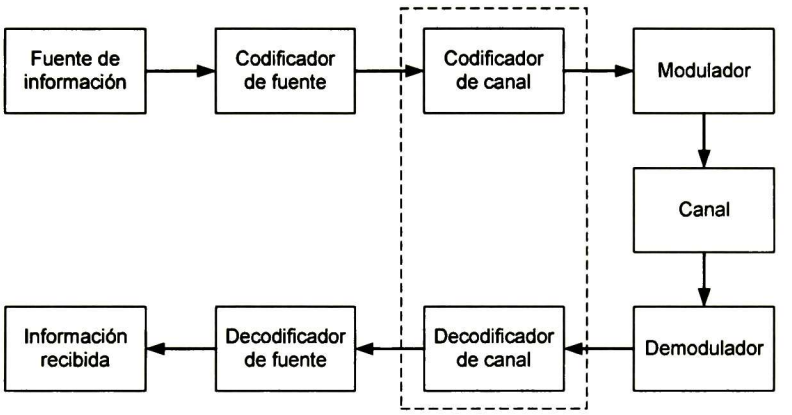


Figura 1.2 Modelo de un sistema de comunicación digital

En donde la fuente de información es el origen de los datos, estos pueden ser una señal de audio o una imagen de video digitalizada. El codificador de fuente tiene como propósito eliminar información redundante, de alguna forma comprime la información a ser transmitida, del mismo modo durante la recepción de la información es necesario descomprimir la información para poder ser entregada en su estado original, por lo que se requiere un decodificador de fuente [3].

1.2.2. Codificación de canal

La codificación de canal consiste en agregar información redundante de forma controlada de tal manera que cuando llegue al decodificador de canal éste pueda detectar y corregir los errores generados por el canal de comunicación. Este método también se le conoce como código corrector de error ECC (Error Correction Code).

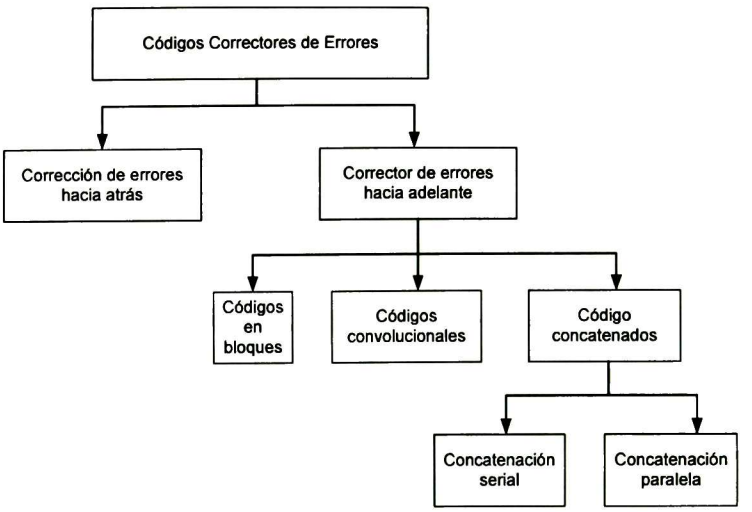


Figura 1.3 Clasificación de los códigos correctores de error

Como se muestra en la figura 1.3 se puede observar dos grandes grupos de códigos correctores de error [5]:

- **Detector de errores**: Éste usa los bits paridad, para detectar los errores, generando de manera automática una petición de repetición, a este tipo de ECC también se le conoce como ARQ (Automatic Repeat Request System), sin embargo este método para la corrección de errores obliga tener una conexión dúplex, además del retraso generado por la retrasmisión.
- **Detector y corrector de errores:** Para este caso los bits de paridad se utilizan no sólo para detectar los errores, también se utilizan para la corrección de estos, los principales códigos utilizados para este esquema son [6] :
  - Códigos a bloques
  - Códigos convolucionales
  - Códigos concatenados

  Los códigos concatenados se pueden dividir en concatenación serial y concatenación paralela, en este último se encuentran los turbo códigos [5].

1.3. Planteamiento del problema

El principal problema que tienen los sistemas de comunicación, consiste en poder enviar información a través de un canal de comunicaciones y ser capaz de recuperar dicha información sin ningún tipo de corrupción o pérdida de datos, por otro lado se tiene la necesidad fundamental de optimizar los recursos, por tanto se requiere hacer una transmisión confiable con una probabilidad de error pequeña utilizando la menor potencia posible. Por esta razón los códigos correctores de error en especifico los turbo códigos que tienen un muy buen desempeño de corrección a una baja potencia de transmisión los hace óptimos para dispositivos de comunicación móviles, tal como los estándares de tercera generación de telefónica celular (3GPP2 C.S0002-E, 3GPP TS 25.212).

La integración de estos dispositivos sólo se encuentra disponible como IP de algunos fabricantes, por lo que se propone el estudio, diseño e implementación en hardware, de un turbo codificador reconfigurable de acuerdo a los estándares 3GPP TS 25.212 [7] y 3GPP2 C.S0002-E [8].

1.4. Objetivos

1.4.1. Objetivos Generales

Los objetivos generales de este proyecto de tesis consiste en diseñar e implementar un turbo decodificador multiestándar de acuerdo con los estándares WCDMA (3GPP TS 25.212) y CDMA2000 (3GPP2 C.S0002-E) descrito en la tesis [9], el cual debe ser reconfigurable, y eficiente en el uso de recursos (memoria, operadores aritméticos). Los objetivos también comprenden la validación y verificación del diseño.

1.4.2. Objetivos Específicos

Los objetivos específicos son:

- Realizar la especificación de requerimientos para el turbo decodificador del estándar WCDMA [7].
  Realizar un simulador del algoritmo implementado en lenguaje C.
  Implementación en plataformas FPGA (codificación HDL, simulación, síntesis, optimización e implementación de la cama de pruebas).
  Realizar un plan de verificación a fin de comprobar que el dispositivo funciona correctamente.
  Ejecutar en forma integral el plan de verificación.

1.5. Organización de la tesis

Esta tesis está divida en seis capítulos organizados de la siguiente manera:

Capítulo 1:

Se presenta la justificación de este trabajo, los antecedentes de la codificación de canal y los objetivos planteados para el desarrollo del turbo decodificador.

Capítulo 2:

En este capítulo se describirán los principios y el funcionamiento del algoritmo de turbo codificación, se explican cada uno de sus componentes, principios de operación, se presentan algunas variaciones en el algoritmo, se explica la forma de operación con el uso de ventanas deslizantes.

Capítulo 3:

Se presentan una descripción del estándar WCDMA (3GPP TS 25.212) además de la especificación de los requerimientos, así como la descripción de los modos de operación propuestos para el turbo decodificador a implementar y las restricciones en su funcionamiento.

Capítulo 4:

En este capítulo se muestra la estructura de la arquitectura propuesta, en la cual se plantea la eliminación de uno de los interleaver del algoritmo. Además se describe cada uno de los componentes del turbo decodificado, así como su funcionamiento y la forma con el que interactúa con los demás componentes.

Capítulo 5:

En este capítulo se describe el proceso de pruebas, mediante los cuales se evaluará el desempeño del turbo decodificador, del mismo modo se presentan los resultados de desempeño obtenidos mediante el proceso de pruebas.

Capítulo 6.

Se presentan las conclusiones y los resultados obtenidos en este trabajo.

Capítulo II

Turbo Codificación

Los turbo códigos fueron propuesto por Claude Berrou y Alain Glavieux en 1993 [4], quienes buscaban un método para disminuir la latencia generada por los códigos correctores concatenados. Este método consiste en colocar dos codificadores concatenados en paralelo (Parallel Concatenated Convolutional Codes, PCCC), poco a poco se convirtió en la vanguardia de las investigaciones en comunicaciones, generando muchas ideas y aplicaciones, una de ellas es su uso en la telefonía celular.

En este capítulo se definen las partes que conforman el turbo codificador, posteriormente se analizarán algunos algoritmos usados para la turbo decodificación.

2.1 Turbo codificador

En la figura 2.1 se puede observar el esquema de un turbo codificador, en donde se observa dos codificadores RSC conectados en paralelo separados por un interleaver. El trabajo del interleaver es entrelazar las secuencias de bits a trasmitir, en otras palabras mezclar el orden de los bits, de este modo el segundo codificador trabajará con la misma trama pero en orden diferente se representa como **c'**.

Una vez que los datos pasaron por los codificadores estos pasan al multiplexor y perforador (Puncturing). Para el caso de la figura 2.1 se dice que tiene una tasa de codificación de 1/3 ya que por cada bit de entrada **c** se tienen tres bits de salida $\mathbf{v_0}$, $\mathbf{v_1}$ y $\mathbf{v_2}$.

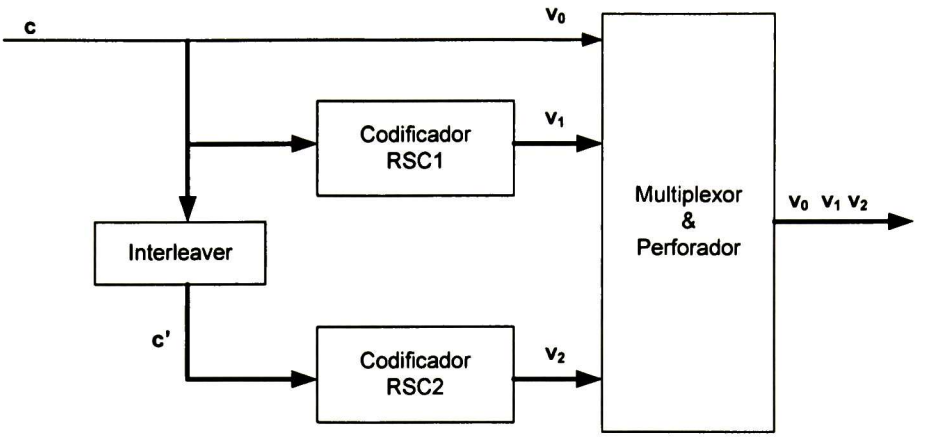


Figura 2.1. Turbo codificador.

El multiplexor consiste en unir los bits paralelos generados por cada uno de los codificadores en una secuencia única, la cual será transmitida a través del canal en una forma serial.

El perforado consiste en remover ciertos bits del total de la palabra de código, esto se realiza con el fin de disminuir el tamaño de la palabra de código, Este perforado se realiza de acuerdo a un patrón definido por una matriz de perforación [5].

2.1.1 Codificadores Componentes

Los códigos convolucionales son básicamente una máquina de estados, en donde sus salidas dependen del estado actual y la entrada. Están constituidos de flip-flops tipo D y sumadores complemento a 2 (compuertas XOR). Se pueden clasificar como sistemáticos y no sistemáticos. Los códigos sistemáticos mantienen los símbolos originales en la palabra codificada, además ésta puede ser reconocida fácilmente a la salida del codificador, mientras que los no sistemáticos no contienen los símbolos originales en su salida.

En la figura 2.2 se tiene un ejemplo de un turbo codificador sistemático, con una tasa de codificación (code rate) de 1/2, y una longitud de restricción (constraint length) de 3; **c** representa la cadena de bits de información, S_1 y S_2 representan los elementos de memoria, c_i y c_p es la salida del codificador, dividido en: bit sistemático y bit de paridad respectivamente.

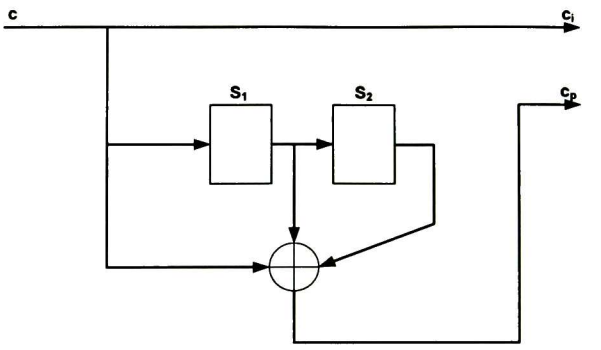


Figura 2.2. Estructura de un codificador convolucional sistemático.

Los codificadores convolucionales se definen por medio de los siguientes parámetros:

- *Tasa de codificación (Code rate)*: es la relación de bits de entrada o información y la cantidad de bits de salida o palabra codificada, ésta se expresa como el cociente de dos números, donde los bits de entrada (*k*) es el numerador y el número de bits de salida (*n*) es el denominador, *k/n*.

- *Número de elementos de memoria* (**v**)*:* Éste parámetro indica el número de registros que conforman al codificador, otra forma en la que se expresa este parámetro es la longitud de restricción (constraint length, **K**) se obtiene sumando uno al número de elementos de memoria **K = v + 1.** Son los que determinan cuántos bits anteriores se usan para generar la palabra de código actual.

  Los polinomios generadores: estos representan el modo de conexión entre los elementos de memoria y los operadores XOR, por medio de estas ecuaciones se describe el codificador. En otras palabras son los que introducen la protección en la palaba de código generada.

Como se observó en la figura 2.1 el turbo codificador está compuesto por dos codificadores RSC (Recursive Systematic Convolutional), en donde se apreció que la entrada a los elementos de memoria, no sólo depende de la entrada, sino también del estado actual, proporcionando una mayor distancia de Hamming entre las palabras del código.

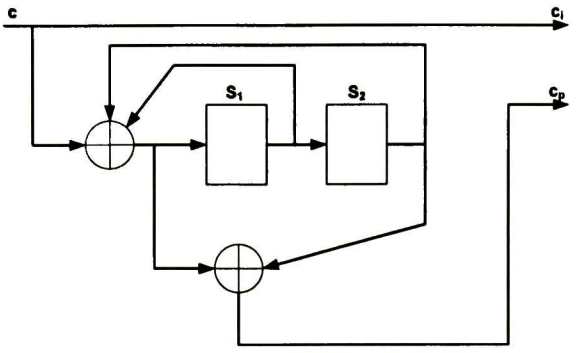


Figura 2.3. Codificador RSC (7,5)

En la figura 2.3 se tiene un codificador RSC (7,5), el número 7 en binario (1 1 1) representa la conexión del sumador complemento a dos para el polinomio de retroalimentación (feedback), en este caso la entrada y los dos elementos de memoria. El segundo número es representado en binario como (1 0 1) e indica la conexión para el polinomio de salida, que son el valor de entrada generado por el primer polinomio y el valor contenido por el último elemento de memoria.

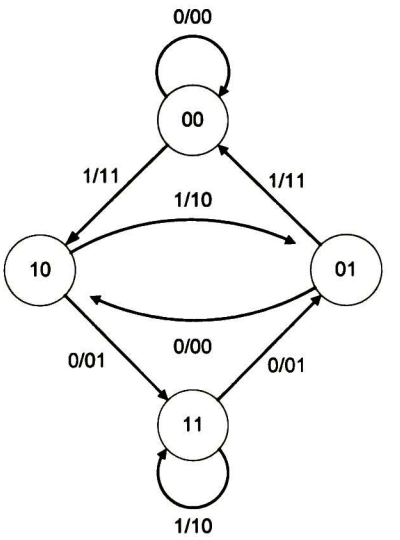


Figura 2.4. Diagrama de estados para un codificador RSC (7,5).

Los códigos recursivos se comportan como máquinas de estado cíclicas [10], por lo que se puede definir su comportamiento por medio de las entradas y el estado en el que se encuentra. En la figura 2.4 se tiene la representación de la máquina de estados para el codificador mostrado en la figura 2.3. Cada estado se representa por el contenido de los elementos de memoria del codificador, por tal motivo se tendrán tantos estados como la relación de 2^{v}, donde **v** son los elementos de memoria. La etiqueta dada a cada una de las transiciones es generada de la siguiente manera: el primer valor indica el bit de entrada, mientras que los otros dos bits indican la salida del codificador para la transición en curso.

Otra forma de expresar el comportamiento del decodificador es por medio de los diagramas de trellis, que consiste en ordenar la máquina de estados anterior en una agrupación temporal [10], en donde cada transición lleva al próximo estado pero en el siguiente tiempo, de esta manera es más sencillo realizar el análisis del comportamiento del codificador a través de toda una secuencia.

En la figura 2.5, se tiene el trellis que describe el comportamiento para el codificador RSC (7,5), en donde se puede observar que cuenta con el mismo número de estados que la máquina mostrada en la figura 2.4, de igual forma se tienen las mismas transiciones. Las líneas punteadas indican cuando el valor de entrada al codificador fue un 0, mientras que las líneas continuas indican cuando la entrada es 1. Mientras que S_{k-1} y S_k indican el estado previo y el estado presente respectivamente, para el bit decodificado en el instante k. Cada una de las uniones entre los estados del diagrama de trellis también se le conoce como *rama*. El número que acompaña a cada una de las ramas del diagrama se le conoce como etiqueta de rama, son las salidas que genera el codificador para cada transición, en este caso son (00 01 10 11) ya que el codificar sólo cuenta con dos salidas.

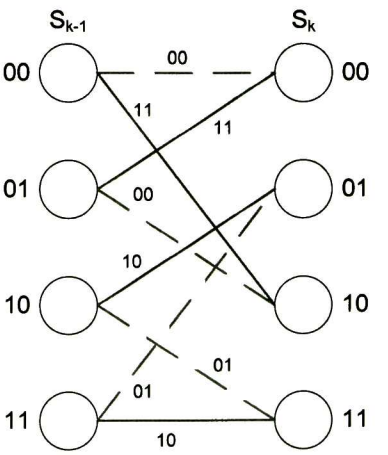


Figura 2.5. Diagrama de Trellis para un decodificador RSC.

2.1.2 Terminación del Trellis

Si dividimos la información a transmitir en bloques independientes, es necesario que cada uno de los bloques inicie en la posición cero del trellis, o bien que los elementos de memoria del codificador se encuentren en ceros, para el primer bloque esto no representa un problema, en los bloques siguientes no se tiene ninguna certeza del estado de inicialización del trellis, esto se podría solucionar agregando **v** bits en cero como cola de cada bloque, sin embargo para los códigos RSC esto no es posible ya que la retroalimentación afecta el contenido de los elementos de memoria, evitando el regreso al estado cero[10].

Una forma de forzar el regreso al estado cero para los codificadores RSC, es hacer que los bits de cola dependan del estado en el que se encuentre el codificador, para esto se le agrega un interruptor como el mostrado en la figura 2.6, este interruptor se mantiene en la posición **A** por los primeros **N** ciclos de reloj y en la posición **B** para los siguientes **v** ciclos, el valor que toma a_t siempre será cero cuando se encuentre en la posición **B** de este modo el contenido de los elementos de memoria regresaran a cero[11].

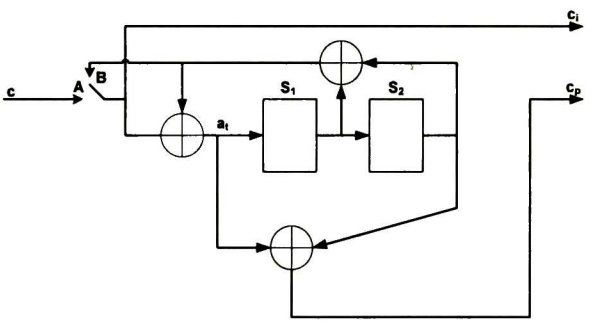


Figura 2.6. Codificador RSC con generador de bits de cola.

2.1.3 Interleaver

Este es un bloque mezclador pseudoaleatorio, definido por la permutación de N elementos sin ninguna repetición [10], en otras palabras cambia el orden de los bits entrantes. Su principal tarea es descorrelacionar las entradas de cada uno de los codificadores, de esta forma si un símbolo no puede ser corregido por el primer

codificador, es posible que el segundo codificador pueda corregirlo ya que este cuenta con información del mismo símbolo pero transmitida en un tiempo diferente [10].

Cuando se genera un esquema de codificación-decodificación de canal es necesario que se conozca el comportamiento del entrelazador, ya que será necesario ordenar la secuencia entrelazada a su forma original, a esto se le conoce como de-entrelazador (deinterleaver).

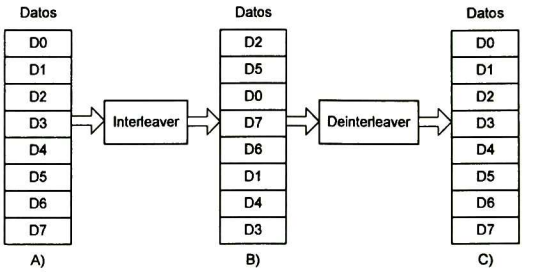


Figura 2.7. Efecto del Interleaver/Deinterleaver sobre un bloque de datos.

En la figura 2.7 se tiene un ejemplo del efecto que causa el interleaver sobre un bloque de 8 datos [10], en la columna A se tienen los datos en su orden original, mientras que en la Columba B se tienen los mismos datos pero en un orden diferente. Como se mencionó anteriormente el trabajo del deinterleaver es regresar un bloque entrelazado a su orden original, esto es lo ocurre entre la Columna B y la C.

2.2 Turbo decodificación

Hasta ahora hemos visto el proceso de turbo codificación del lado del emisor. Aún resta ver qué sucede durante la recepción, más específicamente durante la turbo decodificación. En esta sección se analizará el proceso de turbo decodificación realizada en el receptor, se analizarán las partes que conforman al algoritmo así como sus fundamentos.

En la transmisión de datos binarios digitales sólo es necesario enviar dos símbolos (0, 1), de esta forma el receptor sólo tiene que interpretar y escoger uno de estos dos

símbolos. Existen dos formas de realizar esta interpretación: Decisión dura y Decisión suave.

La decisión dura o (hard decision): consiste en colocar un umbral de decisión central, que separará los estados lógicos. El cual ayudará a tomar una decisión, ya que se tienen dos estados sólo será necesario un umbral, todo lo que logra agrupar ese umbral será tomado con el valor correspondiente, sin importar que tan lejos o cerca se encuentre del otro estado lógico. Este esquema también puede ser visto como un convertidor análogo digital o ADC (Analog to Digital Converter) de 1 bit. En la figura 2.8. a); se muestra la probabilidad de error de un canal AWGN (Affitive White Gaussian Noise), y el umbral mencionado al centro de la gráfica, este umbral divide los dos estados lógicos.

La decisión suave o (soft decision): este esquema propone aumentar el número de umbrales para tener un control más preciso del estado lógico correspondiente, por lo tanto esto requiere un mayor número de bits, definidos como 2^n umbrales, de esta forma es posible identificar cada uno de los umbrales divisores, de igual forma que el HD este es como un ADC de **n** bits, en la figura 2.8 b) se puede apreciar la forma en que se distribuye los umbrales a través de la gráfica de densidad probabilística.

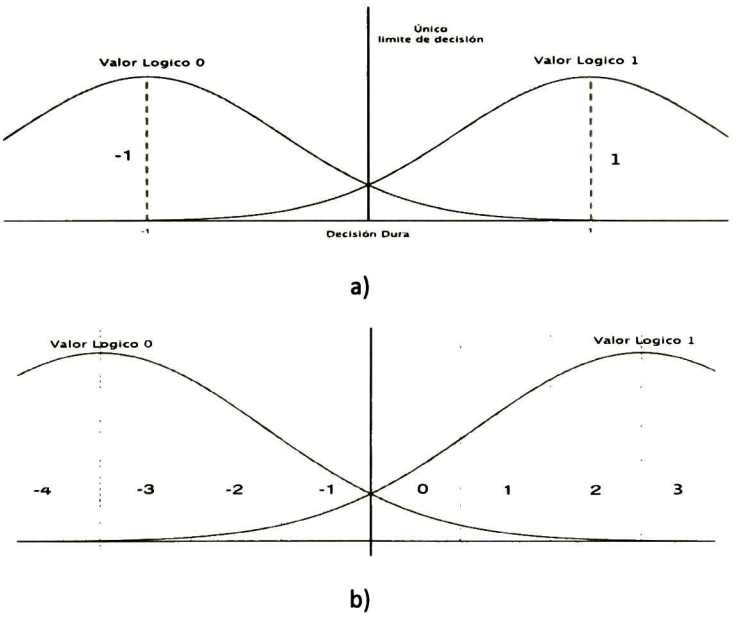


Figura 2.8 Graficas decisión para un canal AWGN; a) Hard Decision, b) Soft Decisión.

El esquema de turbo decodificación es parecido al de su codificador, se tienen dos decodificadores [12], cada uno de ellos es alimentado por la información suave proveniente del demodulador, que se encuentra dividida en Ys, Yp, Yp', valor sistemático, valor de paridad y valor de paridad entrelazado, respectivamente.

En la figura 2.9 se ilustra la estructura principal de un tubo decodificador iterativo, tal como se mencionó anteriormente se encuentra conformado por dos decodificadores SISO, (Soft Input Soft Output).

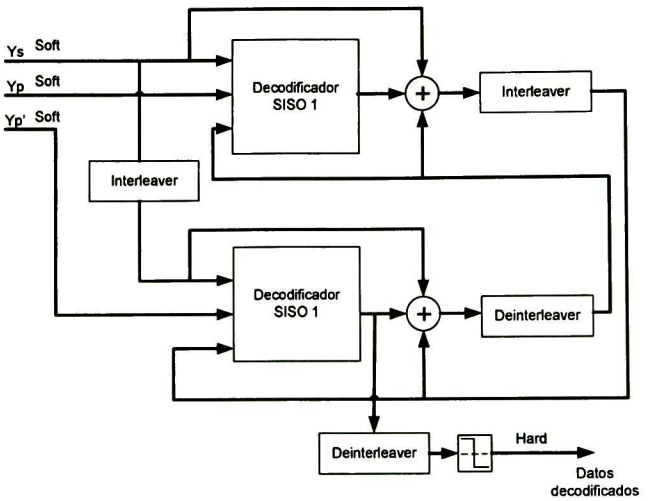


Figura 2.9. Esquema de un turbo decodificador.

Cada uno de los decodificadores SISO cuenta con tres entradas, las primeras dos son las provenientes del canal de comunicación, la otra entrada es la información a-priori proveniente del otro decodificador. Cada uno produce una salida suave que será entrelazada para generar la información a-priori del siguiente decodificador [10]. En el caso del segundo decodificador, sus entradas están compuestas por: la información sistemática entrelazada proveniente del canal, la segunda paridad y la información a-priori, generada por el primer decodificador. Así el decodificador funciona de una forma iterativa, en donde una iteración consiste en decodificar, las entradas una vez en cada uno de los decodificadores.

Conforme se aumenta el número de iteraciones el desempeño del turbo decodificador aumenta, de tal manera que el número de errores disminuye, en otras palabras el BER (Bit Error Rate) disminuye. Sin embargo existe un límite en el número de

iteraciones realizadas, ya que la disminución del BER es poco significativa después de un determinado número de iteraciones, normalmente se utilizan 8 iteraciones [12].

2.2.1 El Procesador SISO

La salida suave es conocida como la razón logarítmica de verosimilitud también conocida como LLR (Log Likelihood Ratio), el signo de este valor representa el bit decodificado, mientras que su magnitud indica la probabilidad o verosimilitud de haber hecho la decisión correcta sobre el signo del bit decodificado; por lo tanto, entre más grande es el valor del LLR mayor es la certeza de una correcta decodificación.

El LLR se puede representar para cada bit u_k como $L(u_k)$, y es definido como el logaritmo de la división de las probabilidades de que un bit tomó alguno de sus dos posibles valores.

$$L(u_k) \triangleq ln\left(\frac{P(u_k = +1)}{P(u_k = -1)}\right). \tag{2.1}$$

Se puede notar que los dos posibles valores que toma u_k son +1 y -1, que representan los valores binarios 1 y 0 respectivamente, la figura 2.10 muestra el comportamiento del $L(u_k)$ [12], contra la probabilidad de u_k = +1, de tal forma que el valor de $L(u_k) \approx 0$ sólo cuando $P(u_k = +1) \approx P(u_k = -1) \approx 0.5$, en este caso no podemos tener certeza del valor decodificado [12].

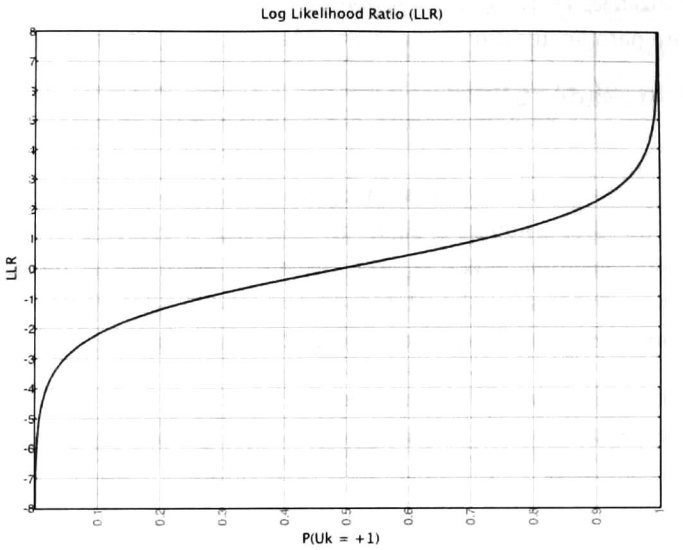


Figura 2.10. . LLR contra la probabilidad de Uk = +1.

El objetivo es que por medio del LLR se pueda calcular la probabilidad de que u_k = +1 ó bien u_k =-1, por lo que es necesario recordar que $P(u_k = -1) = 1 - P(u_k = +1)$, de éste modo se sustituye $P(u_k = -1)$, así como aplicar el exponencial en ambas partes de la ecuación 2.1, entonces ésta se podría escribir de la siguiente manera:

$$e^{L(u_k)} = \frac{P(u_k = +1)}{1 - P(u_k = +1)}, \tag{2.2}$$

De esta ecuación despejamos $P(u_k = +1)$

$$P(u_k = +1) = \frac{e^{L(u_k)}}{1+e^{L(u_k)}}$$
$$= \frac{1}{1 + e^{-L(u_k)}}, \tag{2.3}$$

De igual forma

$$P(u_k = -1) = \frac{e^{-L(u_k)}}{1 + e^{-L(u_k)}}$$

$$= \frac{1}{1 + e^{+L(u_k)}}, \tag{2.4}$$

De esta forma podemos escribir la ecuación general como:

$$P(u_k = \pm 1) = \left(\frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)}}\right) e^{\pm L(u_k)/2} \tag{2.5}$$

En la ecuación 2.5 se puede notar que el término entre paréntesis no mantiene dependencia con la probabilidad de que u_k tome valores de +1 o -1, por lo tanto puede ser tratado como una constante, del mismo modo que podemos tomar el valor $L(u_k)$ que se puede tomar como una probabilidad condicional, en este caso se trata de encontrar el valor del LLR atreves de las probabilidades condicionales, las cuales dependen de la información obtenida del canal de comunicaciones $\underline{y}$.

$$L\left(u_k \middle| \underline{y}\right) \triangleq ln\left(\frac{P\left(u_k = +1 \middle| \underline{y}\right)}{P\left(u_k = -1 \middle| \underline{y}\right)}\right). \tag{2.6}$$

La probabilidad condicional $(P(U_k = +1)|y)$ es conocida como la probabilidad a posteriori del bit a decodificar u_k, en otras palabras es nuestra entrada suave. Además de esta probabilidad condicional también es posible de obtener el valor del LLR basado en la probabilidad de que el bit enviado x_k sea el mismo que el bit recibido y_k. Esta relación es escrita de la siguiente manera:

$$L(y_k|x_k) \triangleq ln\left(\frac{P(y_k|x_k = +1)}{P(y_k|x_k = -1)}\right). \tag{2.7}$$

Podemos notar que la ecuación 2.6 y 2.7 se representan de forma similar, sin embargo manejan conceptos diferentes, estas dos ecuaciones son la teoría de la turbo decodificación. Si asumimos que la transmisión de nuestro bit de información x_k es enviado a través de un canal Gaussiano utilizando la modulación BPSK, podemos escribir la probabilidad de la coincidencia de los bits de la siguiente forma:

$$P(y_k|x_k = \pm 1) = \frac{1}{\sigma\sqrt{2\pi}}\, exp\left(-\frac{E_b}{2\sigma^2}(y_k \pm a)^2\right). \quad (2.8)$$

En donde E_b es la energía transmitida por cada bit, σ es la varianza del ruido y a es la amplitud, del mismo modo podemos escribir la ecuación 2.7 cuando es pasada por un canal gaussiano con modulación BPSK como:

$$L(y_k|x_k) \triangleq ln\left(\frac{exp\left(-\frac{E_b}{2\sigma^2}(y_k - a)^2\right)}{exp\left(-\frac{E_b}{2\sigma^2}(y_k + a)^2\right)}\right)$$

$$= \left(-\frac{E_b}{2\sigma^2}(y_k - a)^2\right) - \frac{E_b}{2\sigma^2}(y_k + a)^2$$

$$= \frac{E_b}{2\sigma^2} 4a\, y_k$$

$$= L_c\, y_k, \quad (2.9)$$

Donde:

$$L_c = 4a\frac{E_b}{2\sigma^2}. \quad (2.10)$$

v

Esta ecuación es conocida como la confiabilidad del canal, y depende sólo de la relación de señal a ruido (SNR) y la amplitud del desvanecimiento del canal. De esta forma podemos definir el LLR como la multiplicación de la salida suave proveniente del canal y_k con la confiabilidad del canal L_c.

2.2.1 El algoritmo de Maximum A-Posteriori Probability MAP

Este algoritmo propuesto por Bahl, Cocke, Jelinek y Raviv en 1974 [12, 13, 14], sirve para estimar las probabilidades a-posteriori de los estados y sus transiciones de una máquina de Markov, cuando está sujeta a ruido sin memoria. Sus autores demostraron cómo este algoritmo podría ser usado en la decodificación de códigos convolucionales y en bloques.

El algoritmo de Viterbi minimiza la probabilidad de tomar un camino del trellis incorrecto. Sin embargo el MAP examina cada una de las posibles ramas que puede tomar el trellis del decodificador convolucional, lo que resulta extremadamente complejo en la mayoría de las aplicaciones, obligándolo a ser poco usado hasta la llegada de los turbo códigos.

El MAP no sólo provee el estimado de la secuencia de bits, sino que también la probabilidad que tiene cada bit de haber sido correctamente decodificado. Esto resulta esencial para la decodificación iterativa aplicada a los turbo códigos, por lo que toma un papel esencial en los principios de la turbo decodificación, desde entonces se ha trabajado en métodos que simplifiquen este algoritmo.

A continuación se describirá el algoritmo MAP, utilizado en la turbo decodificación, en donde se hará uso de la regla de Bayes, que define la probabilidad condicional de la siguiente manera:

$$P(a \wedge b) = P(a|b) \cdot P(b), \tag{2.11}$$

Como consecuencia de la regla de Bayes tenemos que:

$$P(\{a \wedge b\}|c) = P(a|\{b \wedge c\}) \cdot P(b|c), \tag{2.12}$$

Con las ecuaciones 2.11, 2.12 y considerando que: $x \equiv a \wedge b$ y $y \equiv b \wedge c$ se puede deducir la siguiente ecuación:

$$P(\{a \wedge b\}|c) = P(x|c) = \frac{P(x \wedge c)}{P(c)}$$
$$= \frac{P(a \wedge b \wedge c)}{P(c)} \equiv \frac{P(a \wedge y)}{P(c)}$$

$$= \frac{P(a|y) \cdot P(y)}{P(c)} \equiv P(a|\{b \wedge c\}) \cdot \frac{P(b \wedge c)}{P(c)}$$
$$= P(a|\{b \wedge c\}) \cdot P(b|c). \quad (2.13)$$

Como se mencionó anteriormente el algoritmo MAP entrega para cada bit decodificado u_k, la probabilidad de que éste sea +1 ó bien -1, dada la secuencia recibida $\underline{y}$. Tal como se mostró en la ecuación 2.6, de la cual la regla de Bayes nos permite generar la siguiente expresión:

$$L\left(u_k \middle| \underline{y}\right) = ln\left(\frac{P\left(u_k = +1 \wedge \underline{y}\right)}{P\left(u_k = -1 \wedge \underline{y}\right)}\right). \quad (2.14)$$

Para continuar con el análisis de la turbo decodificación es necesario considerar las transiciones del diagrama de trellis mostrado en la figura 2.5, en donde se ilustran todas las posibles transiciones que pueden ocurrir en un código RSC de K = 3, el cual está conformado por cuatro estados, cada estado tiene dos rutas de transición, la cual depende de la cadena de bits transmitidos u_k.

$$L\left(u_k \middle| \underline{y}\right) = \ln\left(\frac{\sum_{(s',s)\Rightarrow u_k=+1} P\left(S_{k-1} = s' \wedge S_k = s \wedge \underline{y}\right)}{\sum_{(s',s)\Rightarrow u_k=-1} P\left(S_{k-1} = s' \wedge S_k = s \wedge \underline{y}\right)}\right) \quad (2.15)$$

En donde $(s', s) \rightarrow u_k = +1$ es la entrada que genera la transición del estado previo $S_{k-1} = s'$ al estado presente $S_k = s$ que puede ocurrir cuando el bit entrante $u_k = +1$, similar para $(s', s) \rightarrow u_k = -1$.

Ahora tomaremos la probabilidad individual $P\left(s' \wedge s \wedge \underline{y}\right)$ proveniente del numerador y el denominador de la ecuación 2.15. La secuencia recibida $\underline{y}$ puede ser separada en tres partes: la palabra de código asociada con la transición presente $\underline{y}_k$, la secuencia anterior $\underline{y}_{j<k}$ y la secuencia siguiente $\underline{y}_{j>k}$, por lo tanto la ecuación se escribe de la siguiente manera:

$$P\left(s' \wedge s \wedge \underline{y}\right) = P\left(s' \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k \wedge \underline{y}_{j>k}\right). \tag{2.16}$$

Por medio de la regla de Bayes y el hecho de que el canal de comunicaciones no cuenta con memoria, la próxima secuencia recibida $\underline{y}_{j>k}$ sólo dependerá del estado presente *s* y no de los estados previos *s'*, o de las secuencias presente y previa recibidas por el canal $\underline{y}_k$ podemos escribir:

$$\begin{aligned} P\left(s' \wedge s \wedge \underline{y}\right) &= P\left(\underline{y}_{j>k} \middle| s' \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k\right) \cdot P\left(s' \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k\right) \\ &= P\left(\underline{y}_{j>k} \middle| s\right) \cdot P\left(s' \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k\right). \end{aligned} \tag{2.17}$$

Nuevamente, usando la regla de Bayes y asumiendo que el canal es sin memoria podemos expandir la ecuación 2.17 como:

$$\begin{aligned} P\left(s' \wedge s \wedge \underline{y}\right) &= P\left(\underline{y}_{j>k} \middle| s\right) \cdot P\left(s' \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k\right) \\ &= P\left(\underline{y}_{j>k} \middle| s\right) \cdot P\left(\left\{\underline{y}_k \wedge s\right\} \middle| \left\{s' \wedge \underline{y}_{j<k}\right\}\right) \cdot P\left(s' \wedge \underline{y}_{j<k}\right) \\ &= P\left(\underline{y}_{j>k} \middle| s\right) \cdot P\left(\left\{\underline{y}_k \wedge s\right\} \middle| s'\right) \cdot P\left(s' \wedge \underline{y}_{j<k}\right) \\ &= \beta_k(s) \cdot \gamma_k(s', s) \cdot \alpha_{k-1}(s'), \end{aligned} \tag{2.18}$$

Donde:

$$\alpha_{k-1}(s') = P\left(S_{k-1} = s' \wedge \underline{y}_{j<k}\right). \tag{2.19}$$

Esta es la probabilidad de que el trellis esté en el estado *s'* en el tiempo *k-1* dada la secuencia recibida del canal antes de *k*, en otras palabras los símbolos de $\underline{y}_{j<k}$.

$$\beta_k(s) = P\left(\underline{y}_{j>k} \middle| S_k = s\right). \tag{2.20}$$

Es la probabilidad de que el trellis esté en el estado s en el tiempo k y que la siguiente secuencia recibida proveniente del canal sea $\underline{y}_{j>k}$ hasta el último elemento.

$$\gamma_k(s', s) = P\left(\left\{\underline{y}_k \wedge S_k = s\right\} \middle| S_{k-1} = s'\right). \tag{2.21}$$

La ecuación 2.21, es la probabilidad de que el trellis esté en el estado s' en el tiempo $k-1$ y se mueva al estado s, dado la secuencia recibida $\underline{y}_k$.

Dicho lo anterior es necesario encontrar un método para poder calcular estos valores y aplicarlos, esto se realiza por medio del algoritmo MAP el cual encuentra $\alpha_k(s)$ y $\beta_k(s)$ para todos los estados s a través del trellis con $k = 0,1, \ldots, N-1$ y $\gamma_k(s', s)$para todas las transiciones posibles del estado $S_{k-1} = s'$ al estado $S_k = s$ para toda $k = 0,1, \ldots, N-1$, todos estos valores son usados para encontrar los valores LLR $L\left(u_k \middle| \underline{y}\right)$ para cada bit u_k.

2.2.1.1 Cálculo de los valores de $\alpha_k(s)$

Considerando la ecuación 2.19, y realizando la sustitución de $\alpha_{k-1}(s')$ por $\alpha_k(s)$ [13], con lo cual permitimos realizar cálculo del valor actual basados en los valores anteriores, generados por la ecuación:

$$\begin{aligned} \alpha_k(s) &= P\left(S_k = s \wedge \underline{y}_{j<k+1}\right) \\ &= P\left(s \wedge \underline{y}_{j<k} \wedge \underline{y}_k\right) \\ &= \sum_{all\ s'} P\left(s \wedge s' \wedge \underline{y}_{j<k} \wedge \underline{y}_k\right). \end{aligned} \tag{2.22}$$

Esta ecuación permite calcular la probabilidad $P\left(s \wedge \underline{y}_{j<k+1}\right)$ en base a la sumatoria de las intercepciones de las probabilidades $P\left(s \wedge s' \wedge \underline{y}_{j<k+1}\right)$ sobre todos los estados previos s'. Por medio de la regla de Bayes y asumiendo que se trata de un canal sin memoria podemos reescribir la ecuación 2.22 como:

$$\alpha_k(s) = \sum_{all\ s'} P\left(\left\{s \wedge \underline{y}_k\right\} \middle| \left\{s' \wedge \underline{y}_{j<k}\right\}\right) \cdot P\left(s' \wedge \underline{y}_{j<k}\right)$$

$$= \sum_{all\ s'} P\left(\left\{s \wedge \underline{y}_k\right\} \middle| s'\right) \cdot P\left(s' \wedge \underline{y}_{j<k}\right) \tag{2.23}$$

Ahora se sustituyen las probabilidades expuestas en la ecuación 2.21 y la ecuación 2.19 tenemos que:

$$\alpha_k(s) = \sum_{all\ s'} \gamma_k(s', s) \cdot \alpha_{k-1}(s'). \tag{2.24}$$

De esta manera, una vez calculado los valores de $\gamma_k(s', s)$, los valores de $\alpha_k(s)$ son calculados por medio de recursiones, para esto es necesario asumir que el trellis siempre iniciara en el estado $S_0 = 0$, por lo tanto la condición inicial para esta recursión es:

$$\alpha_0(S_0 = 0) = 1,$$

$$\alpha_0(S_0 = s) = 0\ para\ toda\ s \neq 0. \tag{2.25}$$

2.2.1.2 Cálculo de los valores de $\beta_k(s)$

Del mismo modo los valores de $\beta_k(s)$ pueden ser calculados de una forma recursiva, basado en la ecuación 2.20, se separa la ecuación de una sola probabilidad, en la sumatoria de las intercepciones de las probabilidades, asumiendo un canal sin memoria y por medio de la regla de Bayes tenemos que:

$$\beta_{k-1}(s') = P\left(\underline{y}_{j>k-1} \middle| s'\right)$$

$$= \sum_{all\ s'} P\left(\left\{\underline{y}_{j>k-1} \wedge s\right\} \middle| s'\right)$$

$$= \sum_{all\ s'} P\left(\left\{\underline{y}_k \wedge \underline{y}_{j>k} \wedge s\right\} \middle| s'\right)$$

$$= \sum_{all\ s'} P\left(\underline{y}_{j>k} \middle| \{s' \wedge s \wedge \underline{y}_k\}\right) \cdot P\left(\{\underline{y}_k \wedge s\} \middle| s'\right)$$

$$= \sum_{all\ s'} P\left(\underline{y}_{j>k} \middle| \{s\}\right) \cdot P\left(\{\underline{y}_k \wedge s\} \middle| s'\right)$$

$$= \sum_{all\ s'} \beta_k(s) \cdot \gamma_k(s', s) \qquad (2.26)$$

De este modo una vez que los valores de $\gamma_k(s', s)$ son conocidos, una recursión hacia atrás es usada para el cálculo de los valores de $\beta_{k-1}(s')$ utilizando los valores de $\beta_k(s)$. Las condiciones de inicialización no son tan sencillas como en el caso de $\alpha_0(s)$, ya que esta condición es la probabilidad de que el futuro estado sea $\underline{y}_{j>k}$ dado el presente estado s. Sin embargo si suponemos que $k = N$, no existirá ninguna secuencia futura, por lo tanto no está claro cuál sería el valor de inicialización. Sin embargo parte del propósito del la terminación del trellis es garantizar que el estado inicial será $\beta_N(0) = 1$ y $\beta_N(s) = 0$ para toda $s \neq 0$ [13].

2.2.1.3 Cálculo de los valores de $\gamma_k(s', s)$

Ahora es necesario calcular los valores de $\gamma_k(s', s)$en base a la secuencia de datos recibidos por el canal de comunicaciones [12, 14], utilizando la definición dada por la ecuación 2.21 y la regla de Bayes se tiene que:

$$\begin{aligned} \gamma_k(s', s) &= P\left(\{\underline{y}_k \wedge s\} \middle| s'\right) \\ &= P\left(\underline{y}_k \middle| \{s' \wedge s\}\right) \cdot P(s|s') \\ &= P\left(\underline{y}_k \middle| \{s' \wedge s\}\right) \cdot P(u_k). \end{aligned} \qquad (2.27)$$

En donde u_k es la entrada de la secuencia de bits necesaria para causar una transición del estado $S_{k-1} = s'$ al estado $S_k = s$, y $P(u_k)$ es la probabilidad a-priori de este bit. Es posible escribir la ecuación 2.5 como:

$$P(u_k = 1) = \left(\frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)}}\right) \cdot e^{L(u_k)/2}$$
$$= C_{L(u_k)}^{(1)} \cdot e^{\frac{L(u_k)}{2}} \quad (2.28)$$

En donde

$$C_{L(u_k)}^{(1)} = \left(\frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)}}\right). \quad (2.29)$$

Sólo depende del $L(u_k)$ y no de la secuencia de entrada u_k. En la ecuación 2.27, sustituyendo el término $P\left(\underline{y}_k \middle| \{s' \wedge s\}\right)$por su equivalente $P\left(\underline{y}_k \middle| \underline{x}_k\right)$, en donde $\underline{x}_k$ es la palabra del código trasmitido del estado $S_{k-1} = s'$ al estado $S_k = s$, una vez más asumiendo que la información es trasmitida a través de un canal sin memoria tenemos que:

$$P\left(\underline{y}_k \middle| \{s' \wedge s\}\right) \equiv P\left(\underline{y}_k \middle| \underline{x}_k\right) = \prod_{l=1}^{n} P(y_{kl}|x_{kl}). \quad (2.30)$$

Donde x_{kl} y y_{kl} es la información individual para cada bit, que componen cada palabra de código y n es el número de bits, ahora asumiendo que la transmisión se realizó a través de un canal Gaussiano usando modulación BPSK, se puede escribir $P\left(\underline{y}_k \middle| \underline{x}_k\right)$ como:

$$P\left(\underline{y}_k \middle| \underline{x}_k\right) = \frac{1}{\sqrt{2\pi\sigma}} exp\left(-\frac{Eb}{2\sigma^2}(y_{kl} - ax_{kl})^2\right). \quad (2.31)$$

En donde *Eb* es la energía de transmisión por bit, σ^2es la varianza del ruido y a es la amplitud del desvanecimiento. Sustituyendo la ecuación 2.31 en la ecuación 2.30 se tiene:

$$P\left(\underline{y}_k \middle| \{s' \wedge s\}\right) = \prod_{l=1}^{n} \frac{1}{\sqrt{2\pi\sigma}} exp\left(-\frac{Eb}{2\sigma^2}(y_{kl} - ax_{kl})^2\right)$$

$$= \frac{1}{\left(\sqrt{2\pi\sigma}\right)^n} exp\left(-\frac{Eb}{2\sigma^2}\sum_{l=1}^{n}(y_{kl} - ax_{kl})^2\right)$$

$$= \frac{1}{\left(\sqrt{2\pi\sigma}\right)^n} exp\left(-\frac{Eb}{2\sigma^2}\sum_{l=1}^{n}(y_{kl}^2 + a^2x_{kl}^2 - 2ax_{kl}y_{kl})\right)$$

$$= C_{\underline{y}_k}^{(2)} \cdot C_{\underline{x}_k}^{(3)} \cdot exp\left(\frac{Eb}{2\sigma^2}2a\sum_{l=1}^{n}(x_{kl}y_{kl})\right), \tag{2.32}$$

Donde:

$$C_{\underline{y}_k}^{(2)} = \frac{1}{\left(\sqrt{2\pi\sigma}\right)^n} exp\left(-\frac{Eb}{2\sigma^2}\sum_{l=1}^{n}(y_{kl}^2)\right). \tag{2.33}$$

Depende sólo del SNR del canal y no de la magnitud de la secuencia recibida y_k, mientras que:

$$C_{\underline{x}_k}^{(3)} = \frac{1}{\left(\sqrt{2\pi\sigma}\right)^n} exp\left(-\frac{Eb}{2\sigma^2}a^2\sum_{l=1}^{n}(x_{kl}^2)\right) \tag{2.34}$$

Depende sólo del SNR del canal y de la amplitud de desvanecimiento. Por lo tanto podemos escribir la ecuación 2.27 como:

$$\gamma_k(s', s) = P\left(\underline{y}_k \middle| \{s' \wedge s\}\right) \cdot P(u_k)$$

$$= C \cdot e^{(u_k L(u_k)/2)} \cdot exp\left(\frac{Eb}{2\sigma^2}2a\sum_{l=1}^{n}x_{kl}y_{kl}\right)$$

$$= C \cdot e^{(u_k L(u_k)/2)} \cdot exp\left(\frac{L_c}{2}\sum_{l=1}^{n}x_{kl}y_{kl}\right) \tag{2.35}$$

Donde

$$C = C_{L(u_k)}^{(1)} \cdot C_{\underline{y}_k}^{(2)} \cdot C_{\underline{x}_k}^{(3)} \tag{2.36}$$

El termino C no depende del signo del bit u_k o de la palabra de código trasmitida $\underline{x}_k$ y ésta es una constante sobre la sumatoria en el numerador y denominador en la ecuación 2.15 por lo que se cancela.

Con lo anterior es posible escribir la ecuación 2.15, dado la secuencia recibida $\underline{y}_k$ como:

$$L(u_k|\underline{y}) = \ln\left(\frac{\sum_{\substack{(s',s)\Rightarrow \\ u_k=+1}} \alpha_{k-1}(s')\cdot\gamma_k(s',s)\cdot\beta_k(s)}{\sum_{\substack{(s',s)\Rightarrow \\ u_k=-1}} \alpha_{k-1}(s')\cdot\gamma_k(s',s)\cdot\beta_k(s)}\right). \tag{2.37}$$

2.2.1.4 Turbo decodificación iterativa

Considerando la ecuación 2.35, asumiendo que el primer bit de los n bits que conforman la palabra del código es la parte sistemática u_k, tenemos que:

$$\gamma_k(s',s) = C\cdot e^{(u_k L(u_k)/2)}\cdot exp\left(\frac{L_c}{2}y_{ks}u_k\right)\cdot exp\left(\frac{L_c}{2}\sum_{l=2}^{n}y_{kl}x_{yl}\right)$$

$$\gamma_k(s',s) = C\cdot e^{(u_k L(u_k)/2)}\cdot exp\left(\frac{L_c}{2}y_{ks}u_k\right)\cdot \mathcal{X}_k(s',s). \tag{2.38}$$

Donde y_{ks}es la versión del bit sistemático trasmitido y $x_{y1} = u_k$ y:

$$\mathcal{X}_k(s',s) = exp\left(\frac{L_c}{2}\sum_{l=2}^{n}y_{kl}x_{yl}\right). \tag{2.39}$$

Usando la ecuación 2.37 y recordando que $u_k = +1$ en el numerador para todos los términos en la sumatoria, en contraparte $u_k = -1$ en el denominador, se puede escribir la ecuación 2.37 como:

$$L(u_k|\underline{y}) = \ln\left(\frac{\sum_{\substack{(s',s)\Rightarrow \\ u_k=+1}} \alpha_{k-1}(s')\cdot e^{+L(u_k)/2}\cdot e^{+L_c y_{ks}/2}\chi_k(s',s)\cdot\beta_k(s)}{\sum_{\substack{(s',s)\Rightarrow \\ u_k=-1}} \alpha_{k-1}(s')\cdot e^{-L(u_k)/2}\cdot e^{-L_c y_{ks}/2}\chi_k(s',s)\cdot\beta_k(s)}\right)$$

$$= L(u_k) + L_c y_{ks} + \ln\left(\frac{\sum_{\substack{(s',s)\Rightarrow \\ u_k=+1}} \alpha_{k-1}(s')\cdot\chi_k(s',s)\cdot\beta_k(s)}{\sum_{\substack{(s',s)\Rightarrow \\ u_k=-1}} \alpha_{k-1}(s')\cdot\chi_k(s',s)\cdot\beta_k(s)}\right)$$

$$= L(u_k) + L_c y_{ks} + L_e(u_k), \tag{2.40}$$

Donde

$$L_e(u_k) = \ln\left(\frac{\sum_{\substack{(s',s)\Rightarrow \\ u_k=+1}} \alpha_{k-1}(s')\cdot\chi_k(s',s)\cdot\beta_k(s)}{\sum_{\substack{(s',s)\Rightarrow \\ u_k=-1}} \alpha_{k-1}(s')\cdot\chi_k(s',s)\cdot\beta_k(s)}\right). \tag{2.41}$$

En la ecuación 2.40, se puede observar que el cálculo para el valor a posteriori LLR $L(u_k|\underline{y})$ está conformado por la suma de tres métricas suaves en términos de $L(u_k)$, $L_c y_{ks}$ y $L_e(u_k)$. En donde el primer término representa la información a-priori que es la información relacionada con el bit a decodificar conocida previamente, en ocasiones ésta también es referida como información intrínseca, el segundo termino es la salida suave proveniente del canal de comunicación y el último termino es conocido como la información extrínseca, ésta se calcula por medio la información capturada por las recursiones y los bits de paridad recibidos a través del canal de comunicaciones [13].

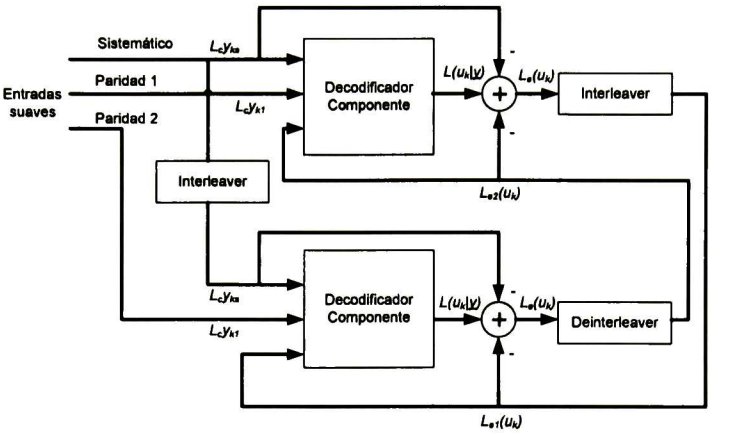


Figura 2.11. Turbo decodificador iterativo

Como se mencionó anteriormente, el signo de la información suave que proporciona el demodulador, indica el valor binario que toma la salida, por consiguiente cuanto más cercano esté a cero, más incertidumbre se tendrá en que la información recibida es la correcta. En la figura 2.12 se muestra un ejemplo en donde se codifico y decodifico una cadena bits compuesta de puros 0, esta prueba se realizó para diferente número de iteraciones, es posible observar que a mayor número de iteraciones, el conjunto de puntos que representan cada bit trasmitido, se aleja de la línea 0, generado mayor certidumbre en los datos recibidos. Esta cadena de bits fue trasmitida sobre una canal AWGN con un SNR de 0.2 dB, se utilizó el turbo decodificador para el estándar 3GPP, que será explicado posteriormente.

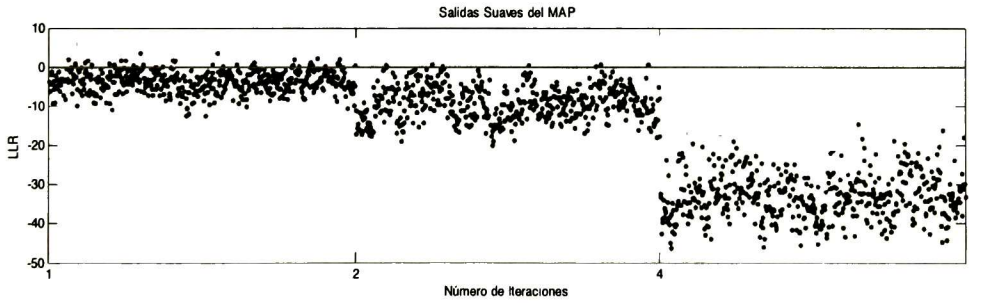


Figura 2.12. Turbo decodificador iterativo con todo un bloque en -1.

2.2.1.5 Max - Log - MAP

El uso del algoritmo MAP tratado en las secciones anteriores, resulta ser muy complejo para ser implementado, esto se debe a las grandes memorias que requiere así como a las operaciones que involucran exponenciales y multiplicaciones, por lo que este algoritmo no fue usado en sus inicios. Por esta razón surge el Max – Log – MAP que es una técnica que simplifica el algoritmo MAP transfiriendo las recursiones al dominio del tiempo, y haciendo uso de aproximaciones que disminuyen dramáticamente la complejidad del algoritmo [12].

El algoritmo Max-Log-MAP tiene como principal base la aproximación mostrada en la ecuación 2.42, en donde se pude observar que esta forma es usada en el cálculo de las recursiones $\alpha_{k-1}(s')$, $\beta_k(s)$ y del cálculo de $\gamma_k(s',s)$ tal como se muestra en la ecuación 2.35.

$$ln\left(\sum_i e^{x_i}\right) \approx \max_i(x_i). \tag{2.42}$$

En donde $\max_i(x_i)$es el valor máximo entre las x_i, Luego se definen los $A_k(s)$, $B_k(s)$ y $\Gamma_k(s',s)$ como:

$$A_k(s) \triangleq \ln\left(\alpha_k(s)\right)$$

$$= \ln\left(\sum_{all\ s'} \alpha_{k-1}(s')\gamma_k(s',s)\right)$$

$$= \ln\left(\sum_{all\ s'} exp[A_{k-1}(s') + \Gamma_k(s',s)]\right)$$

$$\approx \max_{s'}\left(A_{k-1}(s') + \Gamma_k(s',s)\right), \tag{2.43}$$

De manera similar se obtiene la recursión hacia atrás.

$$B_k(s) \triangleq \ln\left(\beta_k(s)\right)$$

$$= \ln\left(\sum_{all\ s'} \beta_k(s)\gamma_k(s',s)\right)$$

$$= \ln\left(\sum_{all\ s'} exp[B_k(s) + \Gamma_k(s', s)]\right)$$

$$\approx \max_{s'}\left(B_k(s) + \Gamma_k(s', s)\right). \tag{2.54}$$

Este mismo cambio se realiza para la obtención de las métricas de rama, escribiendo la ecuación como:

$$\Gamma_k(s', s) \triangleq \ln\left(\gamma_k(s', s)\right)$$

$$= \ln\left(C \cdot e^{(u_k L(u_k)/2)} \exp\left[\frac{Eb}{2\sigma^2} 2a \sum_{l=1}^{n} x_{kl} y_{kl}\right]\right)$$

$$= \ln\left(C \cdot e^{(u_k L(u_k)/2)} \exp\left[\frac{Eb}{2} \sum_{l=1}^{n} x_{kl} y_{kl}\right]\right)$$

$$= \hat{C} + \frac{1}{2} u_k L(u_k) + \frac{L_c}{2} \sum_{l=1}^{n} x_{kl} y_{kl}. \tag{2.55}$$

En donde $\hat{C} = \ln C$ y no depende de u_k de la palabra de código trasmitida $\underline{x}_k$ por tanto puede ser considerada como constante y omitida. Ahora bien aplicando el mismo procedimiento a la ecuación 2.37 es posible escribirla como:

$$L\left(u_k \middle| \underline{y}\right) = \ln\left(\frac{\sum_{\substack{(s',s)\Rightarrow \\ u_k=+1}} \exp\left(A_{k-1}(s') + \Gamma_k(s', s) + B_k(s)\right)}{\sum_{\substack{(s',s)\Rightarrow \\ u_k=-1}} \exp\left(A_{k-1}(s') + \Gamma_k(s', s) + B_k(s)\right)}\right)$$

$$\approx \max_{\substack{(s',s)\Rightarrow \\ u_k=+1}} \left(A_{k-1}(s') + \Gamma_k(s', s) + B_k(s)\right)$$

$$- \max_{\substack{(s',s)\Rightarrow \\ u_k=-1}} \left(A_{k-1}(s') + \Gamma_k(s', s) + B_k(s)\right). \tag{2.56}$$

Una vez que se ha utilizado esta modalidad del algoritmo se observó, que las operaciones logarítmicas y los multiplicadores, fueron remplazados por operadores más sencillos como sumas y cálculos de máximos.

2.2.1.6 Log MAP

La aproximación usada en el algoritmo de la variante Max-Log-MAP, resulta ser sub-óptima, y puede ser mejorada usando el logaritmo Jacobiano, el cual define la siguiente ecuación:

$$\begin{aligned} ln(e^{x_1} + e^{x_2}) &= \max(x_1, x_2) + \ln\left(1 + e^{-|x_1 - x_2|}\right) \\ &= \max(x_1, x_2) + \mathrm{f_c}(|x_1 - x_2|) \end{aligned} \tag{2.57}$$

En donde podemos observar que $\mathrm{f_c}(x)$puede ser vista como un factor de corrección que se le suma al máximo calculado.

El cual también es llamado MAX*, consiste en encontrar el máximo de los exponentes y sumarles un valor de compensación, el cual dependerá del valor absoluto de las diferencias de dichos exponentes.

2.2.2 Métodos de optimización de memoria

Con la sección anterior se analizaron métodos que permiten simplificar la complejidad de las operaciones utilizadas en el algoritmo MAP, sin embargo aún se tiene el problema del uso de la memoria [14, 15], ya que se requiere almacenar las métricas de rama, esto se debe a que éstas son utilizadas primero en el cálculo de cada una de las recursiones $(\alpha_k(s), \beta_k(s))$, y posteriormente para el cálculo del LLR $L\left(u_k \middle| \underline{y}\right)$, en el caso en donde sea necesario decodificar bloques muy grandes, estas memorias tendrían que ser igual mente grande para poder contener cada una de las métricas de rama.

2.2.2.1 Ventanas deslizantes

Este método de optimización de memoria consiste en seleccionar sólo una parte del bloque y procesarlo de forma independiente [10], de esta manera las memorias que

almacenan las métricas de rama serán reutilizadas por otra sección del mismo bloque, éste método es similar al utilizado en el algoritmo de Viterbi.

Las memorias del turbo decodificador pueden ser reducidas en su profundidad, de tal manera que éstas tengan tantas localidades como 6 veces los elementos de memoria contenidos en el codificador [12], para el caso del algoritmo de Viterbi. Otro punto importante es el estado inicial para cada una de las ventanas es desconocido, y esta información es requerida para el cálculo de las recursiones $(\alpha_k(s), \beta_k(s))$. Al inicio de la recursión la ruta generada con una inicialización incorrecta será muy diferente de la ruta original, sin embargo después de varias recursiones ésta se aproximará a la ruta correcta.

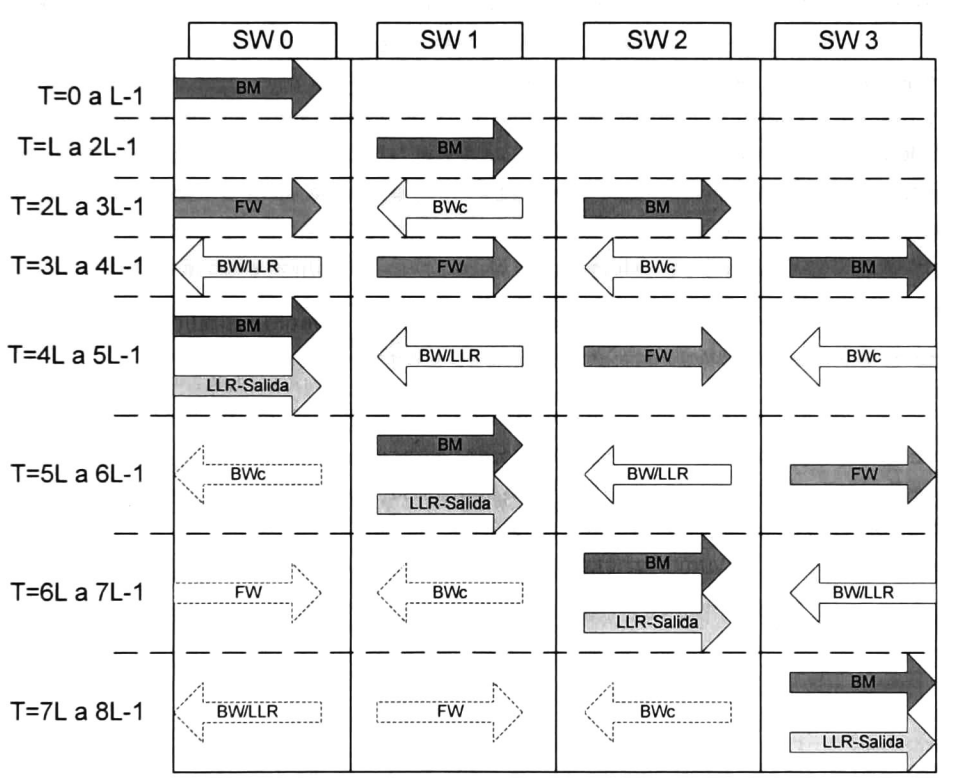


Figura 2.13. Manejo de memoria usando 4 ventanas deslizantes.

En la figura 2.13 se muestra un ejemplo del modo de operación de las ventanas deslizantes, en este caso se cuenta con 4 memorias, las cuales almacenaran las métricas de rama, para posteriormente ser usadas en el cálculo de las recursiones. El nombre que

tiene cada una de las flechas representa la operación que realiza y se describen de la siguiente forma:

- BM Métricas de Rama (Branch Metrics).
- BWc Recursión hacia atrás de convergencia (Backward).
- BW/LLR Recursión hacia atrás de decodificación y cálculo del LLR (Backward).
- FW Recursión hacia adelante (Forward).

El primer paso es calcular las métricas de rama y almacenarlas en la primera memoria T=0 a L-1, este mismo proceso se realiza para la segunda memoria en el tiempo T=L a 2L-1. En el tercer tiempo se realizan tres procesos en forma simultánea, que son el cálculo y almacenado de las métricas de rama, además del cálculo de las recursiones BWc y FW, en el cuarto tiempo se realizan las mismas operaciones que en el tiempo anterior además de eso se calcula la recursión BWd y el LLR, sin embargo este cálculo se realizó hacia atrás provocando que el orden de los bits de salida no se encuentren en el orden original, en el quinto tiempo se repite el mismo proceso además se ordenan y entregan los bit decodificados.

La diferencia del proceso BWc y BWd, es que el primero siempre inicia en el estado cero, mientras que BWd inicia en el último estado generado por BWc.

Con esta estrategia es posible realizar la decodificación de bloques muy grandes sin necesidad de extender las memorias. Otra ventaja que presenta el uso de ventanas deslizantes, es la capacidad de acelerar el procesamiento de los datos, ya que paraleliza los procesos que calculan las métricas de rama, las recursiones y el LLR.

Capítulo III

Especificación de Requerimientos

3.1 Introducción

La especificación de requerimientos para un módulo de hardware es el documento que plasma las necesidades técnicas del proyecto al grupo que diseñará el módulo. En este capítulo se definen claramente las características generales y técnicas del módulo a diseñar. Dicha especificación debe escribirse en un lenguaje claro y sin ambigüedades de forma que sea fácilmente comprensible.

3.2 Estándar WCDMA

El estándar W-CDMA (Wideband Code Division Multiple Access) hereda las ventajas de la tecnología DS-CDMA (Direct Sequence Code Division Multiple Access), además adiciona nuevas tecnologías como una mayor precisión en el TPC (Transmission Power Control), operación de células asíncronas y turbo códigos entre algunas otras mejoras [17].

3.2.1 Estructura

W-CDMA Introduce la operación asíncrona entre las células y un canal piloto asociado con cada canal de comunicación, la operación asíncrona da flexibilidad al desarrollo del sistema, mientras que el canal piloto permite realizar la cancelación de las interferencias y posteriormente permitirá el uso de arreglo de antenas.

Las características del W-CDMA son:

- Búsqueda rápida de células bajo operación intercelular asíncrona.
- Rastreo del código de difusión coherente.

- Rápido TPC en ambos enlaces.
- Estimador de canal coherente en ambos enlaces.
- Múltiples factores de propagación ortogonales durante la descarga.
- Tasa de transmisión variable con detección de tasas ciegas.

En la figura 3.1 se muestra el diagrama a bloques del receptor y transmisor de un sistema basando en W-CDMA [17].

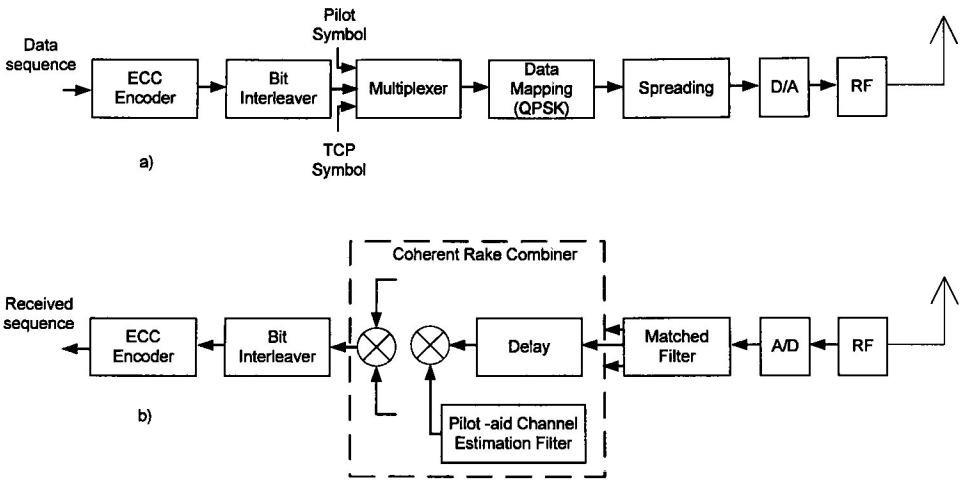


Figura 3.1. Diagrama a bloques del estándar W-CDMA para: a) Transmisor. b) Receptor.

3.2.1.1 Requerimientos básicos

En la siguiente tabla se ilustran las especificaciones básicas para el estándar W-CDMA.

Access scheme	Direct sequence CDMA
Duplex scheme	FDD
Bandwidth	5 MHz
Chip rate	3.84 Mcps
Carrier spacing	200kHz raster
Data speed	2Mbits/s
Frame length	10,20,40,80 msec
Forward error correction	Turbo Code, convolutional code
Data modulation	Downlink: QPSK, Uplink: BPSK
Spreading modulation	Downlink: QPSK, Uplink: HPSK
Spreading factor	4 – 512
Synchronization between base stations	Asynchronous (sync operation also

	possible)
Speech coding	AMR(1.95 k – 12.2 kbits/s)

Tabla 3.1. Especificaciones básicas para el estándar W-CDMA

Las especificaciones para los estándares 3GPP se centraron en la portadora de 5MHz discriminando las portadoras de 10 MHz y 20MHz, esto se debe a que la portadora de 5 MHz es suficiente para cubrir con la velocidad de transmisión 2Mbit/s.

El modo asíncrono entre las estaciones base permite mayor flexibilidad durante la implementación de la red, sin embargo el estándar también soporta la sincronía entre las estaciones base.

La longitud básica de cada trama es de 10 ms, pero también puede tomar los valores mostrados en la tabla 3.1 pasando a través del entrelazador.

El esquema de modulación es de Quadrature Phase Shift Keying (QPSK) para el enlace de descarga y Binary Phase Shift Keying (BPSK) para el de Subida, para el enlace de subida en modo esparcido se utiliza la modulación Hybrid Phase Shift Keying (HPSK).

3.2.2 Turbo codificación

De acuerdo con el estándar existen dos esquemas de codificación de canal que pueden ser usados:

- Codificación convolutiva.
- Turbo codificación

La turbo codificación es efectiva para la transmisión de videos u otras aplicaciones en donde se requiere una alta velocidad de transmisión y calidad en los datos, (tasa de codificación de 1/3, longitud de restricción de 4), en cambio se utiliza la codificación convolucional cuando la aplicación requiera la transmisión de caracteres o alguna otra fuente de datos de baja velocidad, (tasa de codificación de 1/2 y 1/3; longitud de restricción de 9).

En la Figura 3.2 se muestra el diagrama de los tipos de codificadores convolucionales usados en el estándar.

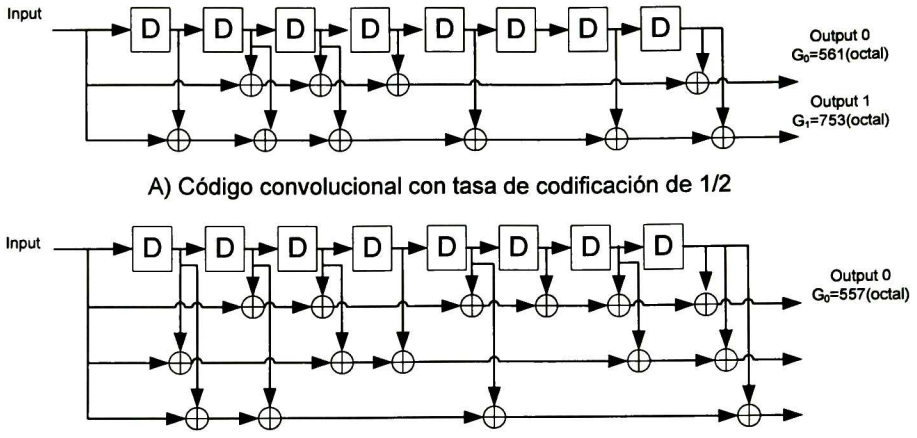


Figura 3.2. Configuraciones del turbo codificador convolucional para el estándar W-CDMA.

Cuando es necesario transmitir datos con una mayor velocidad se utiliza el esquema de codificación mostrado en la figura 3.3, en donde se ilustra el turbo codificador usado para el estándar.

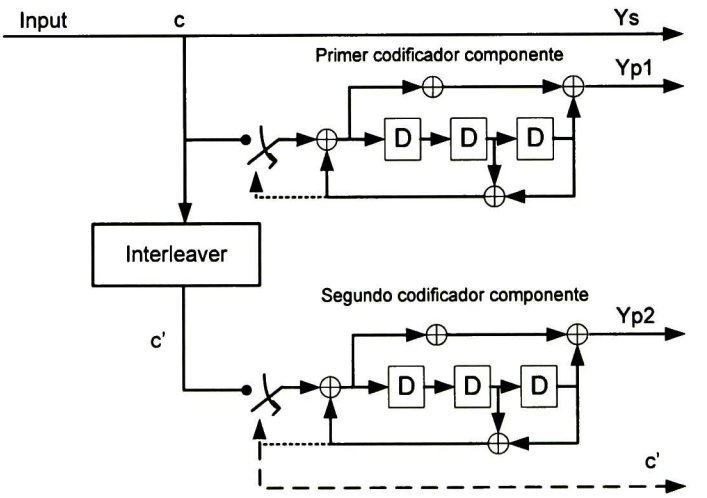


Figura 3.3. Configuración para el turbo codificador para el estándar W-CDMA.

3.3 Turbo decodificador WCDMA

3.3.1 Propósito

En el capítulo uno se habló sobre el propósito de la codificación de canal, se mencionó que existen dos métodos para el control de erres: el FEC y ARQ, los turbo códigos pertenecen al grupo de los FEC ya que no sólo tiene la capacidad de encontrar los errores sino también los pueden corregir.

El turbo decodificador está constituido por dos decodificadores componentes unidos entre sí por medio de un módulo entrelazador también conocido como Interleaver, tal como se mostró en la Figura 2.9.

3.3.2 Funciones del Turbo decodificador

La función del turbo decodificador, es encontrar el valor binario que se transmitió, reduciendo el número de errores generados por causa del ruido blanco gaussiano introducido por el canal de comunicaciones.

El turbo decodificador implementado en hardware, determina el posible valor enviado por el transmisor, a través de los valores entregados por el demodulador con salida suave, previamente codificados por el transmisor, basado en el estándar 3GPP TS-25.212 [7].

Las funcionalidades del turbo decodificador se enumeran en el siguiente grupo de tablas, en donde se da una pequeña descripción de cada una de las funciones.

FUN-TDEC-01	**Configuración de parámetros.**
Descripción	Para que el módulo pueda realizar una decodificación es necesario que sea previamente configurado, estos parámetros serán introducidos mediante señales externas, y guardados en los registros de configuración.
Importancia	Alta
Estado	Integración
Comentarios	Parámetros de configuración:

	• Estándar • Tamaño de bloque Número de iteraciones • Modo de operación Inicialización(Carga de parámetros)

Tabla 3.2. Descripción funcional FUN-TDEC-01

FUN-TDEC-02	**Disponibilidad del módulo.**
Descripción	Indica cuando el módulo fue configurado exitosamente además de eso indica cuando está listo para recibir un nuevo bloque de datos.
Importancia	Alta
Estado	Integración
Comentarios	-

Tabla 3.3. Descripción funcional FUN-TDEC-02

FUN-TDEC-03	**Decodificación**
Descripción	El módulo toma los datos válidos de las entradas del módulo y los almacena en las memorias de datos para posteriormente ser decodificados y entregados a la salida junto a un indicador de validación.
Importancia	Alta
Estado	Integración
Comentarios	Si el indicador de validez a la entrada de datos desaparece antes de la terminación del bloque se considerará que ocurrió un error.

Tabla 3.4. Descripción funcional FUN-TDEC-03

FUN-TDEC-04	**Decodificación sencilla**
Descripción	El módulo decodificará únicamente un bloque de datos y realizará las iteraciones indicadas en la configuración y sólo recibirá el próximo bloque de datos hasta que se concluya la decodificación

	en curso.
Importancia	Alta
Estado	Integración
Comentarios	

Tabla 3.5. Descripción funcional FUN-TDEC-04

FUN-TDEC-05	**Decodificación doble**
Descripción	El módulo decodificará dos bloques de datos y realizará las iteraciones indicadas en la configuración y sólo recibirá los próximos dos bloques de datos hasta que se concluya la decodificación en curso.
Importancia	Alta
Estado	Integración
Comentarios	En este modo SIEMPRE tienen que introducirse dos bloques. Sólo se aceptaran datos cuando la bandera de listo este activa.

Tabla 3.6. Descripción funcional FUN-TDEC-05

FUN-TDEC-06	**Decodificación con dispositivos concatenados**
Descripción	El módulo decodificará un bloque de datos y realizará únicamente una iteración pasando el resultado a un conjunto de módulos que realizaran los iteraciones posteriores.
Importancia	Alta
Estado	Integración
Comentarios	Se pueden colocar tantos módulos como iteraciones deseadas.

Tabla 3.7. Descripción funcional FUN-TDEC-06

3.3.3 Restricciones generales

1. El módulo implementado decodificará tramas, previamente codificadas en base al estándar WCDMA (3GPP TS-25.212).
2. Los buses de entrada de datos tendrán un ancho de palabra de 5 bits para cada uno de los valores sistemático y de paridad.
3. El módulo sólo aceptará datos una vez que el dispositivo indique que está disponible.
4. Las señales de entrada vendrán acompañadas de banderas de sincronización, que indiquen el inicio de la trama y la validez de los datos.
5. La señal de reloj es proporcionada por dispositivos externos.
6. Las señales de control del módulo serán generadas por un dispositivo externo.

3.4 Requerimientos específicos del Turbo decodificador WCDMA

3.4.1 Requerimientos funcionales

RF-TDEC-1. El módulo turbo decodificador, procesará tramas de acuerdo al estándar 3GPP TS 25.212.

RF-TDEC-2. El tamaño de bloque que manejará el turbo decodificador para el estándar WCDMA será de 40 a 5114 bits.

RF-TDEC-3. El módulo manejará un reset asíncrono.

RF-TDEC-4. El módulo será configurado mediante señales de configuración además de una señal de inicialización.

RF-TDEC-5. La decodificación se realizará implementando el algoritmo Log-MAP.

RF-TDEC-6. El Algoritmo Log-MAP, contará con métodos de optimización de memoria mediante el uso de ventanas deslizantes.

RF-TDEC-7. El dispositivo tendrá tres modos de operación:

a. Un sólo módulo podrá realizar hasta 15 iteraciones.
b. Un sólo módulo podrá realizar hasta 15 iteraciones manejando dos bloques por vez.
c. Los módulos podrán ser colocados en cascada para disminuir la latencia, de esta manera cada dispositivo realizará una iteración.

RF-TDEC-8. Los datos de salida del turbo decodificador deberán ir acompañadas de un señal de validez de datos.

RF-TDEC-9. El módulo será capaz de reconocer incongruencias en el tamaño de los bloques e indicarlo.

RF-TDEC-10. El módulo indicará a través de una señal cuando este se encuentre listo.

RF-TDEC-11. Deberá ser reconfigurables para tamaños de bloque de 40 a 5114 bits.

3.4.2 Requerimientos no funcionales

RNF-TDEC-1. El módulo debe ser implementado en el leguaje VHDL.

RNF-TDEC-2. El módulo descrito en VHDL deberá ser sintetizable, para ello se utilizará la herramienta de software Quartus II.

RNF-TDEC-3. El módulo será implementado en un FPGA y probado funcionalmente.

RNF-TDEC-4. El módulo tendrá la capacidad de almacenar la longitud de bloque máxima descrita por el estándar.

RNF-TDEC-5. Se reutilizarán componentes desarrollados en trabajos anteriores.

Capítulo IV

Arquitectura y Diseño

En éste capítulo se explica las aportaciones al diseño del turbo decodificador, así como la descripción de cada uno de los módulos que lo componen, el cual cumple con los requerimientos tratados en el capítulo anterior. Ya que la arquitectura y el diseño propuesto soporta múltiples estándares celulares, esta sección se realizo en forma conjunta con el trabajo [9].

4.1 Propuesta de arquitectura

En la figura 2.9, se presentó la arquitectura para el turbo decodificador, el cual está compuesto básicamente por dos decodificadores SISO y cuatro Interleavers, la arquitectura propuesta nos permite eliminar un interleaver tal como se muestra en la figura 4.1, lo que se traduce en la eliminación de una gran porción de elementos lógicos y de memoria, esto es posible de acuerdo al siguiente análisis.

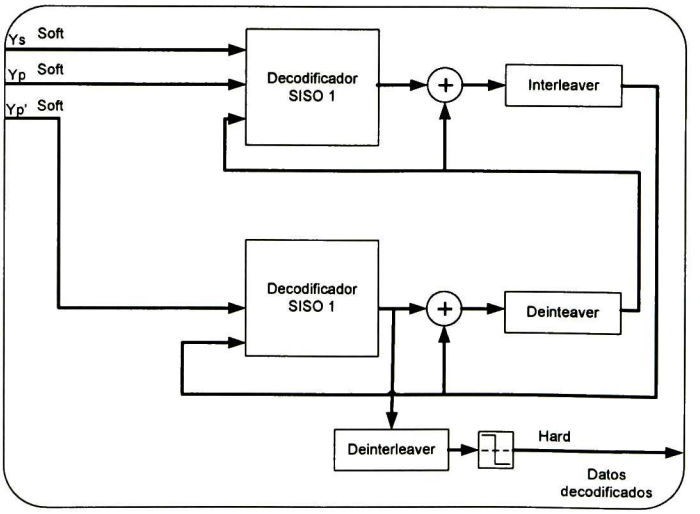


Figura 4.1. Esquema de un turbo decodificador propuesto

El cálculo del LLR depende de la recursión backward, forward y las métricas de rama, éstas últimas son calculadas por medio de la ecuación 2.55, la cual puede ser reescrita de la siguiente forma:

$$\Gamma_k(s',s) = \hat{C} + \frac{1}{2}u_k L(u_k) + \frac{L_c}{2}\sum_{l=1}^{n} x_{kl}y_{kl}$$

$$= \frac{1}{2}(u_k L(u_k) + L_c x_{k1}y_{k1} + \cdots + L_c x_{kn}y_{kn}). \quad (4.1)$$

En la ecuación 4.1 se puede observar que básicamente las métricas de ramas están calculadas en base a la información a-priori, el valor sistemático y las paridades, que difieren unas de otras dependiendo de los valores que puede tomar u_k (+1 o -1). La etiqueta de rama es representada por x_k, esto permite generar las siguientes ecuaciones tomando en cuenta que la información a-priori es extrínseca.

$$G_{0k} = \Gamma_k(s',s) = \frac{1}{2}(-L_e(u_k) - L_c y_{k1} - L_c y_{k2}) \quad (4.2\ a)$$

$$G_{1k} = \Gamma_k(s',s) = \frac{1}{2}(-L_e(u_k) - L_c y_{k1} + L_c y_{k2}) \quad (4.2\ b)$$

$$G_{2k} = \Gamma_k(s',s) = \frac{1}{2}(+L_e(u_k) + L_c y_{k1} - L_c y_{k2}) \quad (4.2\ c)$$

$$G_{3k} = \Gamma_k(s',s) = \frac{1}{2}(+L_e(u_k) + L_c y_{k1} + L_c y_{k2}). \quad (4.2\ d)$$

Ahora se analizará el cálculo de la información extrínseca, que como se explicó en el capítulo dos es calculada basándose en la ecuación 2.40.

$$L\left(u_k \left| \underline{y} \right.\right) = L(u_k) + L_c y_{k1} + L_e(u_k),$$

Despejando $L_e(u_k)$ y definiendo $L(u_k)$ como la información extrínseca de la iteración anterior $L_e'(u_k)$, se tiene.

$$L_e(u_k) = L\left(u_k \left| \underline{y} \right.\right) - L_e'(u_k) - L_c y_{k1}. \quad (4.3)$$

Una vez calculada la información extrínseca, se sustituye en el cálculo de las métricas de rama en el próximo decodificador. Hay que tomar en cuenta que el paso por el interleaver, no altera el contenido de la variable, sólo la mueve de posición con respecto a *k*, de tal forma que la coincidencia respecto a *k* se mantiene para toda la secuencia de símbolos, generando las siguientes ecuaciones.

$$G_{0k} = \Gamma_k(s', s) = \frac{1}{2}\left(-L_1\left(u_k \left| \underline{y}\right.\right) + \mathrm{L_e}'(u_k) + L_c y_{k1} - \mathrm{L_c y_{k1}} - \mathrm{L_c y_{k2}}\right) \quad (4.4\text{ a})$$

$$G_{1k} = \Gamma_k(s', s) = \frac{1}{2}\left(-L_1\left(u_k \left| \underline{y}\right.\right) + \mathrm{L_e}'(u_k) + L_c y_{k1} - \mathrm{L_c y_{k1}} + \mathrm{L_c y_{k2}}\right) \quad (4.4\text{ b})$$

$$G_{2k} = \Gamma_k(s', s) = \frac{1}{2}\left(+L_1\left(u_k \left| \underline{y}\right.\right) - \mathrm{L_e}'(u_k) - L_c y_{k1} + \mathrm{L_c y_{k1}} - \mathrm{L_c y_{k2}}\right) \quad (4.4\text{ c})$$

$$G_{3k} = \Gamma_k(s', s) = \frac{1}{2}\left(+L_1\left(u_k \left| \underline{y}\right.\right) - \mathrm{L_e}'(u_k) - L_c y_{k1} + \mathrm{L_c y_{k1}} + \mathrm{L_c y_{k2}}\right). \quad (4.4\text{ d})$$

Ahora es visible que la información sistemática $L_c y_{k1}$ se elimina dejando de aportar información al cálculo de las métricas de ramas. Por esto es posible eliminarla de la entrada del segundo decodificador MAP y ahorrar el uso de un interleaver, además de una segunda memoria encargada de aliviar la latencia generada por el procesador MAP, que tendría como función guardar la información sistemática, para ser restada posteriormente durante el cálculo de la información extrínseca.

Por otra parte, en el cálculo de la información extrínseca del segundo decodificador MAP, también es posible eliminar la información sistemática, tomando en cuenta la ecuación 4.3 y aplicándola al cálculo de la información extrínseca del segundo MAP, la ecuación puede ser escrita como:

$$L_{e2}(u_k) = L_2\left(u_k \left| \underline{y}\right.\right) - L_{e1}(u_k) - L_c y_{k1} \quad (4.5)$$

En donde $L_{e1}(u_k)$ está definido como:

$$L_{e1}(u_k) = L_1\left(u_k \left| \underline{y}\right.\right) - L_{e2}'(u_k) - L_c y_{k1} \quad (4.6)$$

Además $L_{e2}'(u_k)$ es el cálculo de la información extrínseca del MAP anterior. Ahora sustituyendo la ecuación 4.6 en la 4.5 tenemos:

$$L_{e2}(u_k) = L_2\left(u_k \left| \underline{y}\right.\right) - L_1\left(u_k \left| \underline{y}\right.\right) + L_{e2}'(u_k) + L_c y_{k1} - L_c y_{k1} \quad (4.6)$$

Una vez más la información sistemática se elimina, permitiendo dejarla fuera durante el cálculo de la información extrínseca, ahorrando una memoria más.

4.2 Diseño

La descripción se realiza por medio del modelo Top-Down, que consiste en establecer una serie de niveles de mayor a menor complejidad que den solución al problema [18]. En la figura 4.1 se muestra el diagrama de entrada y salidas del turbo decodificador, y en la tabla 4.1 se da la descripción de cada una de estas señales.

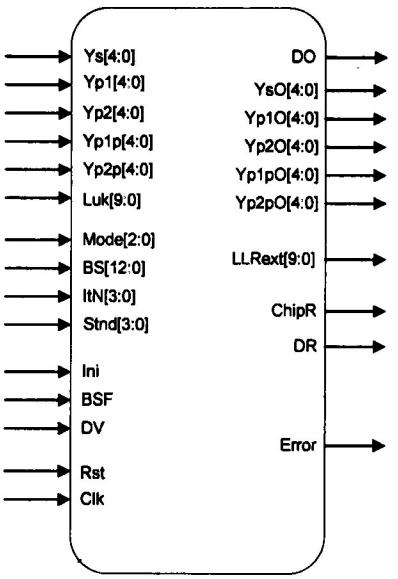


Figura 4.2. Diagrama de señales de entrada y de salida del turbo decodificador.

Como ya se mencionó esta tesis trata sobre el desarrollo de un turbo decodificador para el estándar WCDMA (3GPP TS 25.212), sin embargo la arquitectura fue planteada para soportar también el estándar CDMA2000 [8], por lo que algunas de las entradas no son requeridas.

Nombre	Descripción
Entradas	
Clk	Señal de reloj. Se considera el flanco de subida.
Rst	Señal de Reset
DV	Bandera de validez de datos.
BSF	Bandera de inicio de bloque.

Ini	Bandera de configuración del módulo.
Stnd	Selector de estándar.
itN	Selector de número de iteraciones.
BS	Selector del tamaño del bloque a decodificar, conforme a la especificación [Número de especificación].
Mode	Selector del modo de operación, conforme a la especificación [Número de especificación].
Luk	Valor de información a-priori, usada en el modo de dispositivos concatenados.
Ys	Valor sistemático.
Yp1	Valor de paridad uno.
Yp1p	Valor de paridad uno entrelazado.
Yp2	Valor de paridad dos.[1]
Yp2p	Valor de paridad dos entrelazado.[1]
Salidas	
DO	Dato binario decodificado
DR	Bandera de validez de datos
ChipR	Bandera indicadora de módulo listo
Error	Bandera indicadora de error detectado
LLRext	Señal de salida extrínseca o LLR (depende del modo de operación)
YsO	Valor sistemático (modo concatenado)
Yp1O	Valor de paridad uno (modo concatenado)
Yp1pO	Valor de paridad uno entrelazado (modo concatenado)
Yp2O	Valor de paridad dos (modo concatenado)[1]
Yp2pO	Valor de paridad dos entrelazado (modo concatenado)[1]

Tabla 4.1. Descripción de las señales de las señales de entrada y salida del turbo decodificador.

[1] Sólo son usados para el estándar CDMA2000 con una tasa de codificación de 1/4 y 1/5.

4.3 Descomposición Modular

La primera jerarquía que se analizará, es la división entre el bloque del control y la ruta de los datos, en donde el datapath es alimentado por los datos del exterior y controlado en su totalidad por el bloque de control, En la figura 4.2 se ilustra esta separación, así como la interacción que tiene el control, con el datapath.

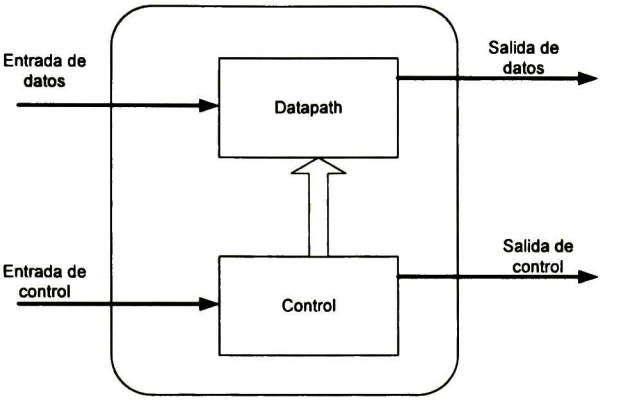


Figura 4.3. Descomposición modular general.

4.3.1 Datapath

Es la parte del diseño que se encarga de la operación y manipulación de los datos. Está compuesto por diferentes módulos, tal como se ilustra en la figura 4.4, estos módulos se encuentras descritos en la tabla 4.2, en donde podemos observar que el turbo decodificador se compone básicamente por dos decodificadores MAPpi y dos bloques de memoria. Los bloques MAPpi, se conforman por un decodificador MAP y un interleaver, esto se realizó con el fin de simplificar el análisis.

Es necesario el uso de dos bloques de memoria con el fin de tener la capacidad de procesar dos bloques por vez, este requerimiento es descrito en el capítulo anterior como requerimiento funcionales FUN-TDEC-05 y FUN-TDEC-06, esto se debe a que el segundo MAPpi depende del resultado del primer MAPpi, tal como se mencionó en secciones anteriores, la información a-priori generada por la iteración anterior es fundamental para el funcionamiento del algoritmo MAP.

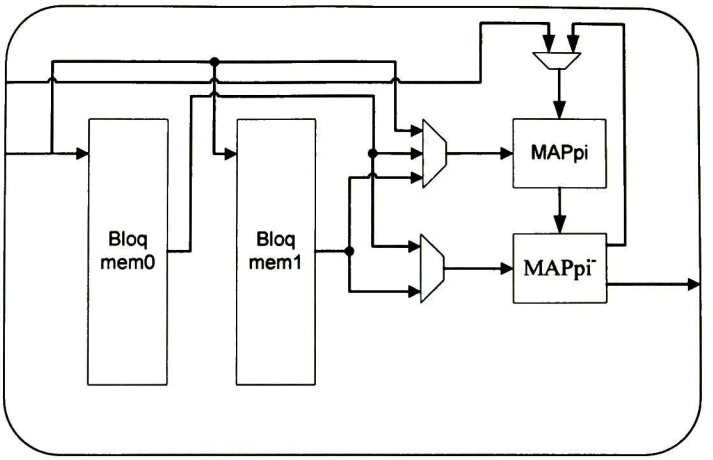


Figura 4.4. Flujo de datos de la arquitectura del turbo decodificador.

En la figura 4.5 se muestra el diagrama temporal del flujo de los bloques, en donde se puede apreciar la necesidad de implementar dos bloques de memoria, como parte fundamental del requerimiento FUN-TDEC-06, en donde se pide que el módulo tenga la capacidad de ser conectado en forma concatenada, de tal manera que se agilice la decodificación entre iteraciones.

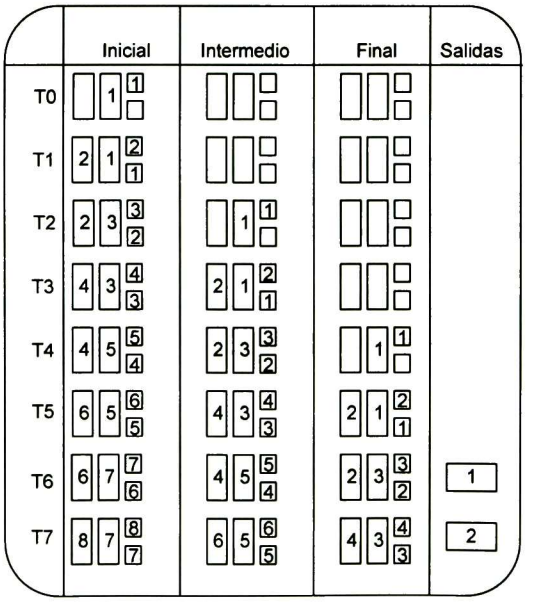


Figura 4.5. Diagrama de procesamiento de bloques.

Módulo	Descripción
Bloqmem0	Memoria de datos 1, almacena los datos de entrada y los mantiene durante la decodificación, para la primera secuencia.
Bloqmem1	Memoria de datos 2, almacena los datos de entrada y los mantiene durante la decodificación, para la segunda secuencia.
MAPpi	Decodificador componente, compuesto de un Módulo decodificador Log-MAP, y un interleaver-deinterleaver.

Tabla 4.2. Descripción de componentes.

Para T0 el primer bloque se recibe y graba en la primera memoria además se procesa en el primer MAPpi, en T2 el segundo bloque es procesador por el primer MAPpi y grabado en la segunda memoria, mientras que el primer bloque es procesado en el segundo MAPpi, para T3 el segundo dispositivo procesa el primer bloque, mientras el primer dispositivo procesa el tercer bloque en el primer MAPpi y el segundo bloque es procesado en el segundo MAPpi, ésta acción es repetida hasta completar todos los módulos concatenados. La posición de cada dispositivo es configurada en la entrada Mode, la cual define el modo en que operará el módulo, y se encuentra especificado en la tabla 4.3.

Mode[2:0]	Nombre
000	Un dispositivo un sólo bloque.
001	Dispositivos concatenados, parte inicial.
010	Dispositivos concatenados, parte intermedia.
011	Dispositivos concatenados, parte final.
1XX	Un dispositivo dos bloques.

Tabla 4.3. Descripción de modos de operación.

En el caso de dispositivos concatenados es necesario indicar que posición tiene, ya que realizarán diferentes funciones. Para el primer módulo éste tiene que leer los datos sistemáticos y de paridad, pero no espera recibir algún valor en la entrada del Luk, por lo contrario los dispositivos que se encuentran en medio esperan la entrada Luk, el dispositivo final a diferencia de los otros dos, entrega como resultado el LLR y no la información extrínseca.

4.3.1.1 Bloques de Memorias

Estos bloques almacenan en forma temporal la información sistemática y de paridad leídas en la entrada del módulo, con el fin de poder alimentar a los decodificadores MAPpi

en diferentes instantes de tiempo. Las entradas y salidas de este bloque se muestran en la figura 4.5.

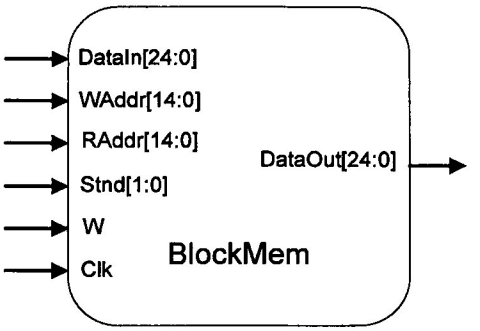


Figura 4.6. Diagrama de señales de entrada y de salida de un bloque de memoria.

En la tabla 4.4, se realiza una descripción de cada una de las entradas y salidas, además se especifica el destino y origen, se puede observar que los datos de entrada son separados, con el propósito de describir cada una de las señales que conforman toda la palabra de datos, la cual será almacenada en la memoria. Del mismo modo los datos que conforman la salida de la memoria son separados y descritos en forma independiente.

Nombre	Descripción	Conexión
Entradas		**Origen**
DataIn[5:0]	Valor sistemático.	Exterior
DataIn [6:9]	Valor de paridad uno.	Exterior
DataIn [11:14]	Valor de paridad dos.	Exterior
DataIn [17:19]	Valor de paridad uno primo.	Exterior
DataIn [22:24]	Valor de paridad dos primo.	Exterior
WAddr	Entrada de la dirección de escritura.	Control
RAddr	Entrada de la dirección de lectura.	Control
W	Señal de habilitación de escritura.	Control
Stnd	Selector del estándar a manejar.	Exterior
Salidas		**Destino**
DataOut[5:0]	Valor leído de la memoria correspondiente al dato sistemático.	Mux1
DataOut[6:9]	Valor leído de la memoria correspondiente al dato de paridad uno.	Mux1
DataOut[11:14]	Valor leído de la memoria correspondiente al dato de paridad dos.	Mux1

DataOut[17:19]	Valor leído de la memoria correspondiente al dato de paridad uno primo.	Mux2
DataOut[22:24]	Valor leído de la memoria correspondiente al dato de paridad dos primo.	Mux2

Tabla 4.4. Descripción de las señales de entrada y salida del un bloque de memoria.

El turbo decodificador está diseñado para soportar múltiples estándares, el estándar CDMA2000 [8], tiene la capacidad de manejar diferentes tasas de codificación, por esta razón se separó el manejo de las memorias de datos, con el fin de controlar la habilitación, dependiendo del tasa de codificación que se esté manejando. Este esquema puede ser observado en la figura 4.6, en donde la segunda memoria almacena los datos de las paridades adicionales para la tasa de codificación de 1/4 y 1/5.

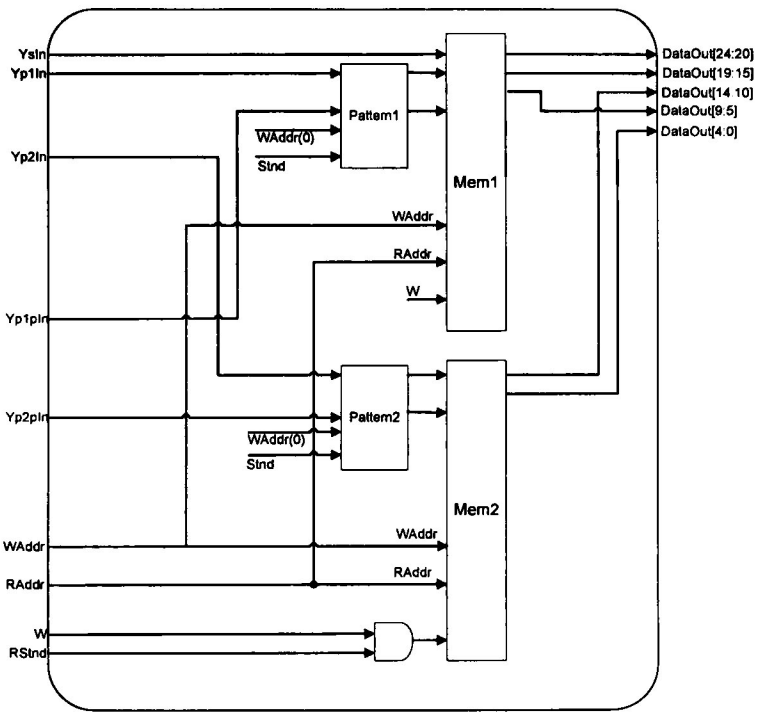


Figura 4.7. Flujo de datos de la arquitectura de un bloque de memoria.

4.3.1.2 MAPpi

Este bloque es propuesto con el fin de simplificar el análisis y el control del procesamiento. En el caso de los modos operación: bloques concatenados y bloques múltiples, en los cuales es necesario un manejo complejo en el flujo de datos. Además permite una estructura modular.

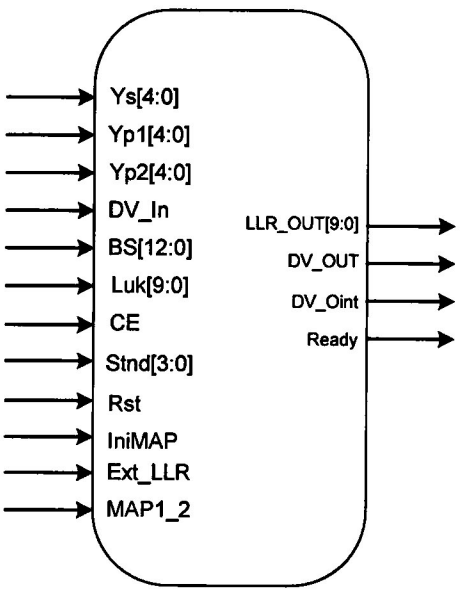


Figura 4.8. Diagrama de señales de entrada y de salida del bloque MAPpi

En la figura 4.8, se muestra las entradas y salidas utilizadas por este módulo, en la tabla 4.5 se describen e indican los puntos de interconexión, para los dos módulos MAPpi utilizados en el turbo decodificador.

Nombre	Descripción	Conexión M1	Conexión M2
Entrada		Origen	Origen
CE	Habilita el módulo.	Control.	Control.
DV_IN	Señal de validez de datos.	Control.	Control.
IniMap	Señal de configuración del módulo MAPpi	Control.	Control.
Stnd	Selecciona el estándar que se está manejando.	Registro de configuración.	Registro de configuración.

Ext_LLR	Selecciona el valor a entregar ya sea extrínseco o LLR.	Control.	Control.
MAP1_2	Selecciona la posición del MAPpi.	0.	1.
K	Selecciona el tamaño del bloque a decodificar.	Registro de configuración.	Registro de configuración.
Ys	Valor sistemático.	Mux1.	Mux2.
Yp1	Valor de paridad uno	Mux1.	Mux2.
Yp2p	Valor de paridad dos.	Mux1.	Mux2.
Luk	Entrada de la información a-priori.	Muxluk.	Muxluk.
Salidas		**Destino**	**Destino**
LLR_OUT[10:0]	Bus de datos con los valores del LLR o extrínsecos de acuerdo a la señal de entrada Ext_LLR.	Mappi2.	Muxluk, Salida.
DV_OUT	Señal que indica que los valores proporcionados en el bus de datos LLR_OUT son validos para el próximo bloque.	Control, Mappi2.	Control, Salida.
DV_Oint	Señal que indica que los valores proporcionados en el bus de datos LLR_OUT son validos, durante las iteraciones	Control	Control
Ready	Indica que el módulo MAPpi ha sido configurado y que está listo para recibir un bloque de datos.	Control	Control

Tabla 4.5.Descripción de las señales de entrada y salida del bloque de MAPpi

Al inicio de este capítulo se demostró que es posible eliminar el interleaver de entrada del valor sistemático para el segundo decodificador MAP. Además se sugirió un cambio en el cálculo de los valores extrínsecos, en el cual podemos concluir que es posible obtener la información extrínseca sólo con el LLR y la información a-priori $L_e(u_k)$, de esta forma se permite eliminar una memoria encargada de aliviar la latencia generada por el decodificador MAP. Tal como se puede mostrar en la figura 4.9 en donde sólo se utiliza una memoria FIFO que almacena el $L_e(u_k)$

Este bloque tiene la capacidad de calcular el LLR o la información extrínseca, dependiendo de su posición o bien del número de iteraciones seleccionadas. Esta configuración es controlada por el control del turbo decodificador.

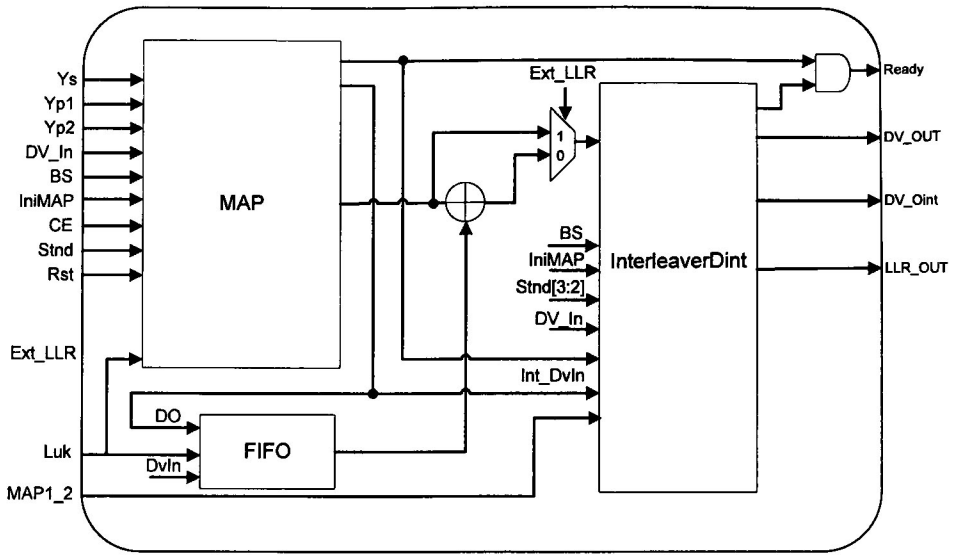


Figura 4.9. Descomposición modular del MAP_PI.

Los módulos que conformar este bloque se encuentran descritos en la tabla 4.6, esta descripción será profundizada posteriormente.

Módulo	Descripción
MAP	Decodificador MAP, éste encuentra el LLR a partir de los datos de entrada, y la información a-priori.
FIFO	Memoria FIFO, utilizada para compensar la latencia que tiene el MAP y así poder calcular la información extrínseca.
IntDint	Bloque entrelazador, su propósito es entrelazar la información generada por el MAP de tal manera que pueda ser usada en el próximo MAP.

Tabla 4.6. Descripción de componentes.

4.3.1.2.1 MAP

El decodificador MAP fue diseñado e implementado con el uso de ventanas deslizantes, que optimiza el uso de memoria, de acuerdo a lo especificado en el requerimiento RF-TDEC-6, es reconfigurable de acuerdo con los estándares descritos en el requerimiento RF-TDEC-1, fue diseñado bajo el algoritmo LOG-MAP mencionado en el requerimiento RF-TDEC-5.

El decodificador MAP tiene como valores de configuración el TCR y BS mostrados en la figura 4.10, las cuatro primeras señales son de datos, mientras que el resto son señales de control. El ancho de las palabras usadas por el decodificador MAP son tomadas

del trabajo previo [19], en donde se optimizó el desempeño del turbo decodificador en función del ancho de las palabras de datos de entrada y de los registros de procesamiento, estos últimos se analizarán más adelante.

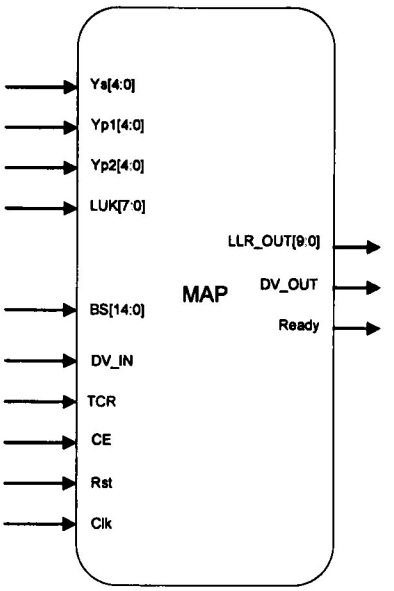


Figura 4.10. Diagrama de señales de entrada y de salida del bloque MAP.

La descripción y los puntos de interconexión se muestran en la tabla 4.7, recordemos que el turbo decodificador esta compuesto por dos bloques MAPpi por lo tanto dos decodificadores MAP, sin embargo en términos generales las conexiones de estos módulos son iguales, salvo que en diferente MAPpi.

Nombre	Descripción	Conexión
Entradas		**Origen**
Ys	Valor sistemático.	Exterior Mappi
Yp1	Valor de paridad uno.	Exterior Mappi
Yp2	Valor de paridad dos.*	Exterior Mappi
Luk	Valor de información a-priori, usada en el modo de dispositivos concatenados.	Exterior Mappi
BS	Selector de tamaño del bloque a decodificar.	Exterior Mappi.
DV_IN	Señal de validez de datos.	Control.
TCR	Selector del tipo de tasa de codificación.	Exterior Mappi
CE	Señal de habilitación del módulo.	Control.
Rst	Señal de Reset.	Exterior.

Clk	Señal de reloj. Se considera el flanco de subida.	Exterior.
Salidas		**Destino**
LLR_OUT	Valores del LLR calculados por el módulo.	Sumador, interleaver.
DV_OUT	Señal de validez de datos de salida.	Interleaver, exterior Mappi.
Ready	Señal de indicación de módulo listo.	Exterior Mappi.

Tabla 4.7. Descripción de las señales de entrada y salida del bloque de MAP

El funcionamiento del decodificador MAP depende de cinco procesos básicos, el cálculo de las métricas de rama, las recursiones FW, BWc, BWd y el LLR, sin embargo este grupo de procesos tiene que realizarse en un orden específico que permita el mayor ahorro de recursos.

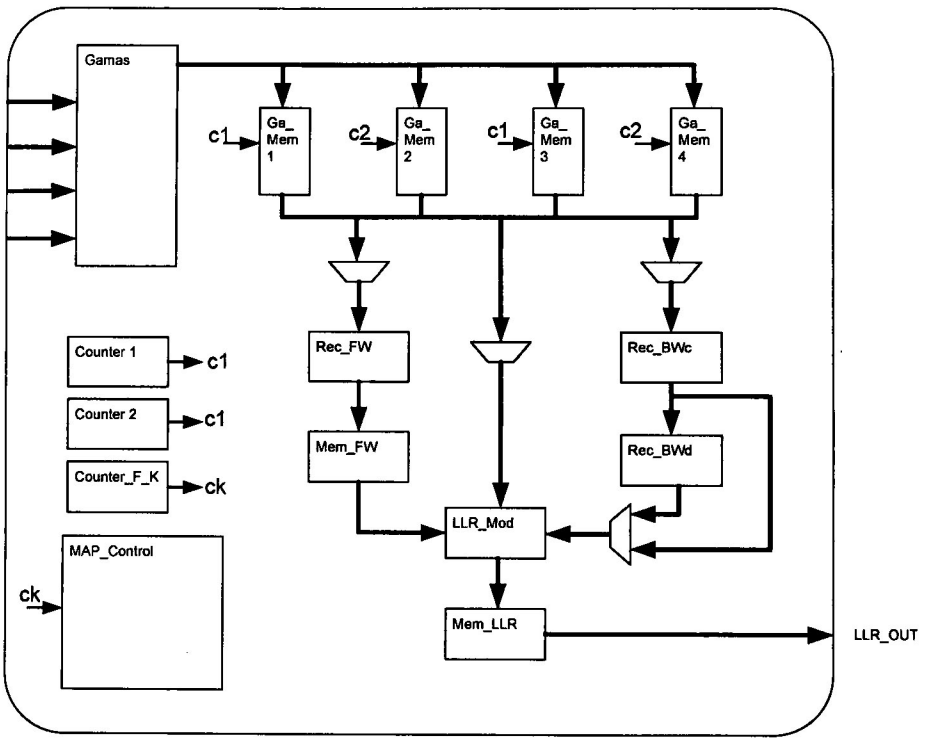


Figura 4.11. Descomposición modular del MAP

Gamas

Las métricas de rama es el bloque inicial del decodificador MAP, tal como lo muestra la ecuación 2.55, se observa que las métricas de rama sólo dependen de la información proveniente del exterior del MAP, valor sistemático, de paridad y la información a-priori.

Estas entradas de datos se pueden apreciar en la figura 4.12 en la cual se ilustran las entradas y salidas que conforman este bloque.

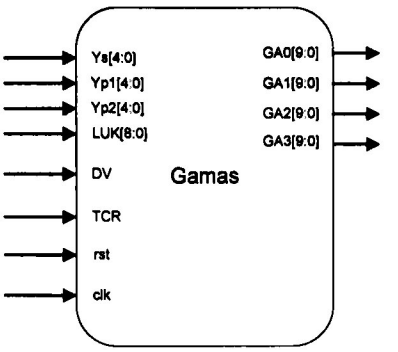


Figura 4.12.Diagrama de señales de entrada y de salida del bloque Gamas

En la tabla 4.8 se describen todas las señales de entrada y salidas que componen a este bloque, además de eso se indican los puntos de interconexión con el resto de los elementos del bloque MAP.

Nombre	Descripción	Conexión
Entradas		**Origen**
Ys	Valor sistemático.	Exterior MAP.
Yp1	Valor de paridad uno.	Exterior MAP.
Yp2	Valor de paridad dos.[2]	Exterior MAP.
Luk	Valor de información a-priori, usada en el modo de dispositivos concatenados.	Exterior MAP.
DV	Señal de validez de datos.	Exterior MAP.
TCR	Selector del tipo de tasa de codificación.	Exterior MAP.
Rst	Señal de Reset.	Exterior.
Clk	Señal de reloj. Se considera el flanco de subida.	Exterior.
Salidas		**Destino**
GA0	Valor de la gama cero.	Ga_Mem.
GA1	Valor de la gama uno.	Ga_Mem.
GA2	Valor de la gama dos.	Ga_Mem.
GA3	Valor de la gama tres.	Ga_Mem.

Tabla 4.8. Descripción de las señales de entrada y de salida del bloque Gamas.

[2] Sólo para las tasas de codificación de 1/4 y 1/5 del estándar CDMA2000[7]

La entrada TCR permite seleccionar la tasa de codificación con la que se realizará el cálculo de las métricas de rama, ya sea para el caso $^1/_2$. $^1/_3$ o bien para $^1/_4$ y $^1/_5$ descritos en [8].

Tal como se muestra en la ecuación 2.55 y más simplificado en la ecuación 4.1, el cálculo de las métricas de rama se realiza mediante una serie de sumas y restas indicadas en el grupo de ecuaciones 4.4. Esta acción se realiza mediante una estructura general la cual se muestra en la figura 4.13, ésta cambia dependiendo de la etiqueta que se esté calculando.

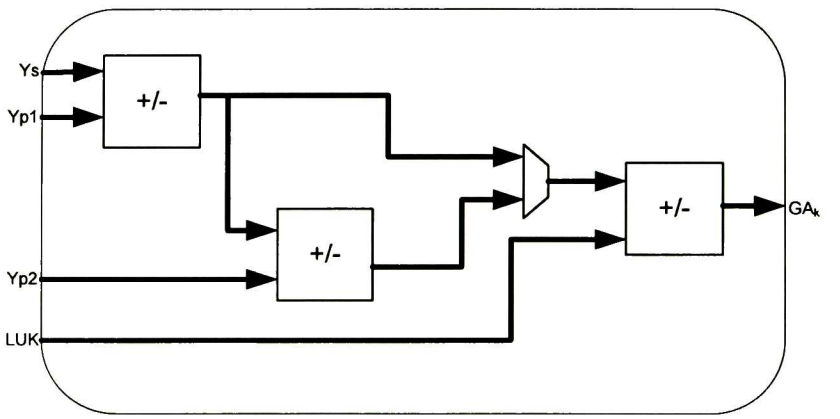


Figura 4.13. Estructura general para el cálculo de las métricas de rama.

Memoria de Gamas

Tal como se mencionó en el capítulo dos, la implementación de las ventanas deslizantes consiste en dividir la trama en secciones, por medio de memorias que almacenen las métricas de rama, a éstas se le conoce como ventanas, las métricas de rama contenidas en las memorias posteriormente serán usadas para realizar los cálculos de las recursiones y el LLR. Para este diseño las ventanas o memorias cuentan con 64 localidades.

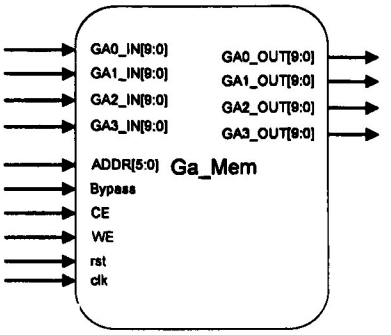


Figura 4.14. Diagrama de señales de entrada y de salida del bloque memoria de gamas

La configuración de entradas y salidas de las memorias gama se muestran en la figura 4.14, y se describen en la tabla 4.9. Las entradas de datos son las salidas de datos generado por el bloque gamas, con sus correspondientes salidas, mientas que el resto de las señales son de control.

Nombre	Descripción	Conexión
Entradas		**Origen**
GA0_IN	Valor de la gama cero.	Gamas.
GA1_IN	Valor de la gama uno.	Gamas.
GA2_IN	Valor de la gama dos.	Gamas.
GA3_IN	Valor de la gama tres.	Gamas.
ADDR	Dirección de lectura y escritura de la memoria.	Contador1, contador 2.
Bypass	Selector de paso, para bloques menores al tamaño de las ventanas.	MAP_Control.
CE	Señal de habilitación del bloque de memoria.	MAP_Control.
WE	Señal de habilitación de escritura.	MAP_Control.
Rst	Señal de Reset.	Exterior.
Clk	Señal de reloj. Se considera el flanco de subida.	Exterior.
Salidas		**Destino**
GA0_OUT	Valor de la métrica de rama cero leída.	MuxBW, MuxFW, MuxLLR.
GA1_OUT	Valor de la métrica de rama uno leída.	MuxBW, MuxFW, MuxLLR.
GA2_OUT	Valor de la métrica de rama dos leída.	MuxBW, MuxFW, MuxLLR.
GA3_OUT	Valor de la métrica de rama tres leída.	MuxBW, MuxFW, MuxLLR.

Tabla 4.9. Descripción de las señales de entrada y de salida del bloque de memoria de gamas.

La primera ventana tiene la opción de Bypass o de paso, que permite entregar a la salida los datos que tiene a la entrada, esta función se utiliza sólo para tramas menores al tamaño de memoria, con lo cual es posible disminuir la latencia para este tamaño de trama.

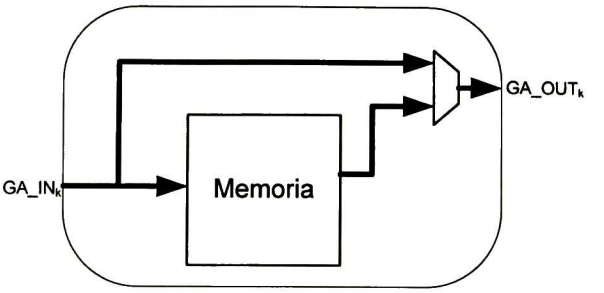


Figura 4.15. Estructura de la memoria de gamas.

Recursión Forward

De acuerdo a la ecuación 2.24 en la cual se describe el cálculo de la recursión forward, que se obtiene por medio de las métricas de rama y el valor obtenido en la recursión anterior. Esto se puede ver representado en la figura 4.16 en donde se puede observar que las entradas de datos para este módulo sólo esta compuesta por las métricas de rama, el resto de las entradas son de elementos de control.

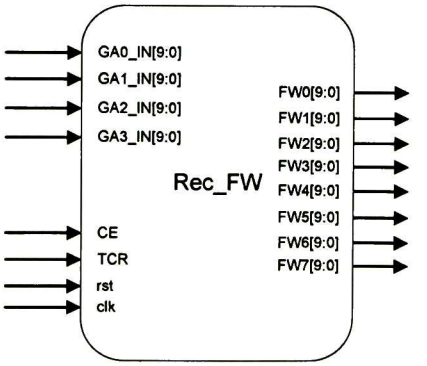


Figura 4.16. Diagrama de señales de entrada y de salida del bloque de recursión Forward.

Las entradas de las métricas de rama de este bloque provienen de un multiplexor tal como se aprecia en la tabla 4.10, con el fin de poder seleccionar la ventana a la que se le realizara la recursión.

Nombre	Descripción	Conexión
Entradas		**Origen**
GA0_IN	Valor de la gama cero.	MuxFW.
GA1_IN	Valor de la gama uno.	MuxFW.
GA2_IN	Valor de la gama dos.	MuxFW.

GA3_IN	Valor de la gama tres.	MuxFW.
CE	Señal de habilitación del bloque.	MAP_Control.
TCR	Selector del tipo de tasa de codificación.	MAP_Control.
Rst	Señal de Reset.	Exterior.
Clk	Señal de reloj. Se considera el flanco de subida.	Exterior.
Salidas		**Destino**
FW0	Resultado cero de la recursión forward.	FW_Mem.
FW1	Resultado uno de la recursión forward.	FW_Mem.
FW2	Resultado dos de la recursión forward.	FW_Mem.
FW3	Resultado tres de la recursión forward.	FW_Mem.
FW4	Resultado cuatro de la recursión forward.	FW_Mem.
FW5	Resultado cinco de la recursión forward.	FW_Mem.
FW6	Resultado seis de la recursión forward.	FW_Mem.
FW7	Resultado siete de la recursión forward.	FW_Mem.

Tabla 4.10. Descripción de las señales de entrada y de salida del bloque de memoria de gamas.

También es posible observar que las salidas de esta recursión van dirigidas hacia una memoria encargada de almacenar el resultado de este módulo, hasta que sean requeridas para el cálculo del LLR.

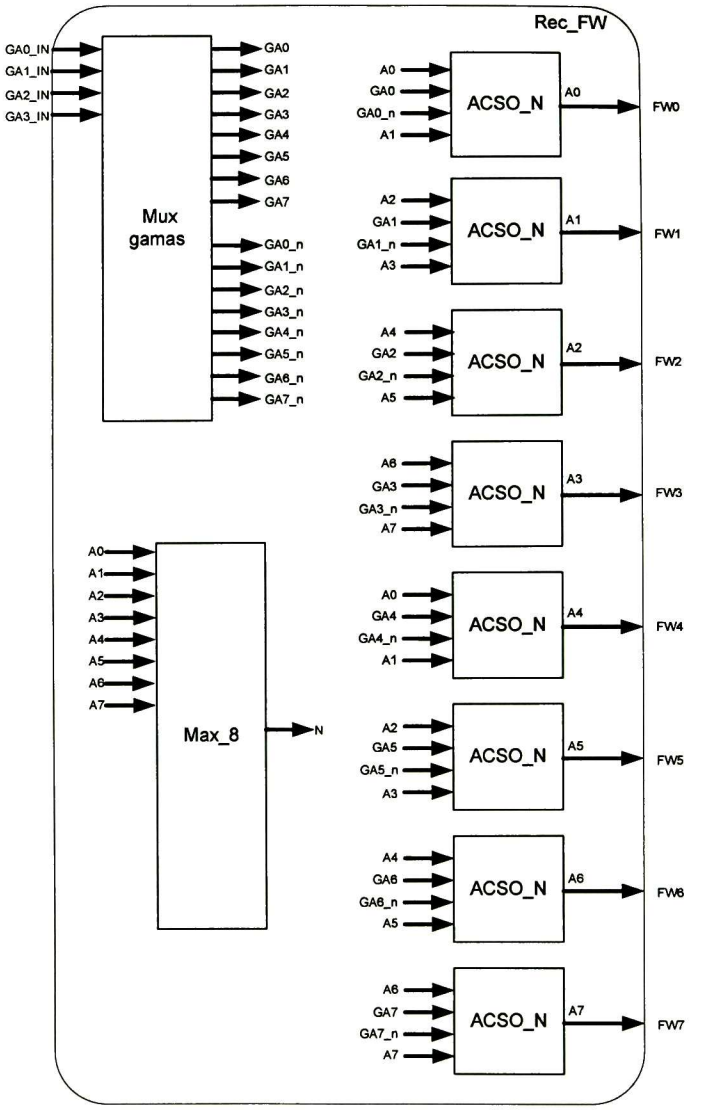


Figura 4.17. Estructura de la recursión Forward.

El diseño multiéstandar obliga a requerir un bloque de multiplexores, que permita seleccionar la tasa de codificación con la que se operará. Este bloque es nombrado como Mux Gamas, se muestra en la figura 4.17, sus entradas son las métricas de rama entregadas por las memorias, mientras que sus salidas son las métricas de rama adaptadas según el estándar seleccionado.

De acuerdo con lo mencionado en el capítulo dos el algoritmo MAP puede ser simplificado por el algoritmo Log-MAP. Tal como se muestra en la ecuación 2.57, en la cual

se define el operador MAX*, que es parte fundamental para el cálculo de las recursiones. En la figura 4.18, se muestra la estructura fundamental del bloque ACSO (Add Compare Select Offset) [20], que tiene como función sumar, comparar, seleccionar el máximo y además de agregar el offset de compensación. El MAX* está conformado por el bloque Max y el bloque de offset LUT.

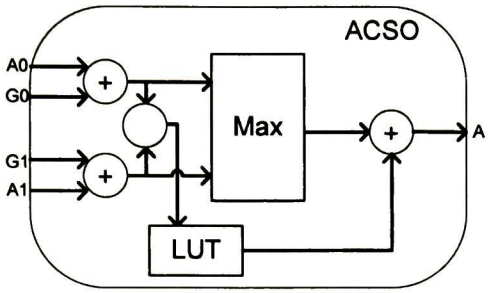


Figura 4.18. Estructura del bloque ACSO.

La forma en que se calcula el valor máximo, se muestra en la figura 4.19, esta estructura está formada por un multiplexor y una resta que tiene como función comparar las dos entradas y mediante el signo controlar la selección.

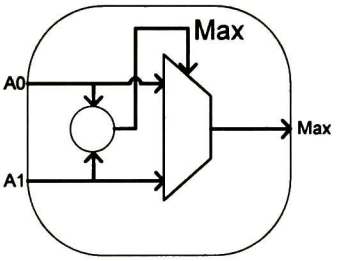


Figura 4.19. Estructura del bloque MAX

Un problema fundamental en el cálculo de las recursiones, es que siempre se calcula el máximo de la suma de las métricas de rama, con el valor obtenido en la recursión anterior $A(s)_{k-1}$. Esto da como resultado la tendencia de obtener valores cada vez más grandes, lo que se traduce en un desbordamiento de los registros contenedores de dicha información. Para aliviar este problema se utiliza un método de normalización, el cual tiene como fundamento, que no importa las magnitud de los estados, sólo la diferencia que existe entre ellos [21], esto se puede calcular por medio de la ecuación 4.7, en donde $\hat{A}_k(s)$ es el cálculo de la recursión normalizado, $A_k(s)$ es el cálculo de la recursión sin normalizar y $\max_{s'} A_k(s')$ es la magnitud máxima de los estados para la

recursión anterior. Este método permite evitar el desbordamiento de los registros sin perder información.

$$\hat{A}_k(s) = A_k(s) - \max_{s'} A_k(s') \tag{4.7}$$

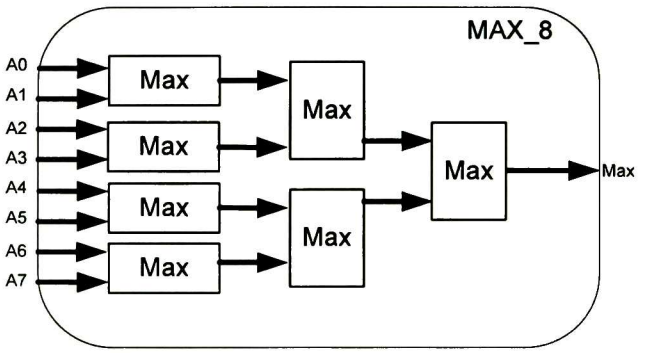


Figura 4.20. Estructura del bloque normalizador.

Para implementar la normalización es necesario encontrar el estado s' con el valor más grande, esto se realiza por medio del bloque llamado MAX_8, mostrado en la figura 4.20, el resultado de este módulo alimenta al bloque ACSO_N, mostrado en la figura 4.21, el cual realiza la resta y la almacena en el registro.

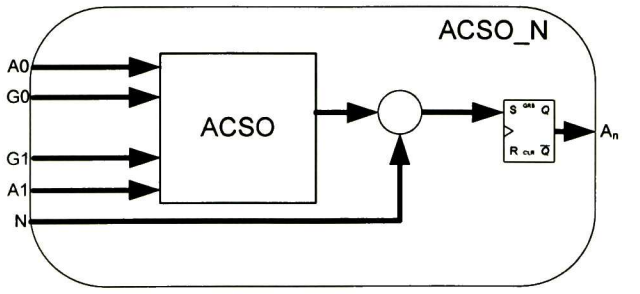


Figura 4.21. Estructura del bloque ACSO normalizado.

Memoria de la recursión Forward

Una vez que se realizó la recursión forward es necesario almacenar estos datos ya que no serán usados hasta el cálculo del LLR, para esto se utiliza esta memoria, que tiene 64 localidades al igual que las memorias usadas para almacenar las métricas de rama.

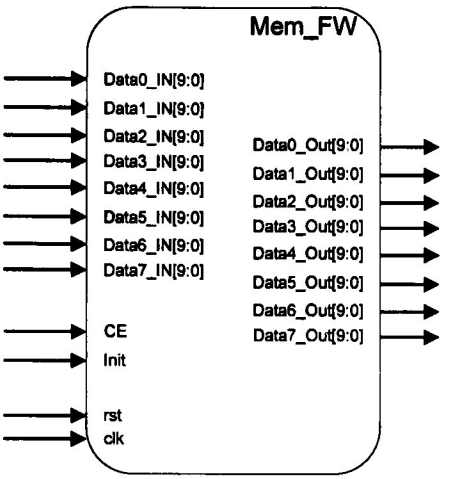


Figura 4.22.Diagrama de señales de entrada y de salida del bloque memoria forward

Como se muestra en la figura 4.22 esta memoria sólo tiene como datos de entrada las salidas de la recursión forward, además de dos señales de control, la cuales indican al módulo cuando iniciar el guardado de los datos o bien cuando reiniciar los contadores de la memoria. Esta memoria una vez que es activada, grabará los datos en orden ascendente y los leerá en forma descendente. Cada nuevo dato de entrada sustituirá al dato de salida permitiendo la optimización en el uso de la memoria. La descripción de entradas y salidas de ese módulo se encuentran descritas en la tabla 4.11.

Nombre	Descripción	Conexión
Entradas		**Origen**
Data0_IN	Datos de entrada a la memoria posición cero.	Rec_ FW.
Data1_IN	Datos de entrada a la memoria posición uno.	Rec_ FW.
Data2_IN	Datos de entrada a la memoria posición dos.	Rec_ FW.
Data3_IN	Datos de entrada a la memoria posición tres.	Rec_ FW.
Data4_IN	Datos de entrada a la memoria posición cuatro.	Rec_ FW.
Data5_IN	Datos de entrada a la memoria posición cinco.	Rec_ FW.
Data6_IN	Datos de entrada a la memoria posición seis.	Rec_ FW.
Data7_IN	Datos de entrada a la memoria posición siete.	Rec_ FW.

CE	Señal de habilitación del bloque.	MAP_Control.
Init	Señal de inicialización de la dirección de escritura.	MAP_Control.
Rst	Señal de Reset.	Exterior.
Clk	Señal de reloj. Se considera el flanco de subida.	Exterior.
Salidas		**Destino**
Data0_OUT	Datos de salida de la memoria posición cero.	LLR_Mod.
Data1_OUT	Datos de salida de la memoria posición uno.	LLR_Mod.
Data2_OUT	Datos de salida de la memoria posición dos.	LLR_Mod.
Data3_OUT	Datos de salida de la memoria posición tres.	LLR_Mod.
Data4_OUT	Datos de salida de la memoria posición cuatro.	LLR_Mod.
Data5_OUT	Datos de salida de la memoria posición cinco.	LLR_Mod.
Data6_OUT	Datos de salida de la memoria posición seis.	LLR_Mod.
Data7_OUT	Datos de salida de la memoria posición siete.	LLR_Mod.

Tabla 4.11. Descripción de las señales de entrada y de salida del bloque memoria forward.

Recursión Backward de convergencia

El propósito central de este bloque es encontrar el estado inicial para la recursión backward de decodificación, esto es necesario debido a la implementación de las ventanas deslizantes. Para esta recursión no se tiene certeza de cual es el estado inicial para cada ventana, esta situación no se presenta cuando se trata de una ventana única ya que se sabe que el estado inicial es cero debido a la terminación hecha por los bits de cola.

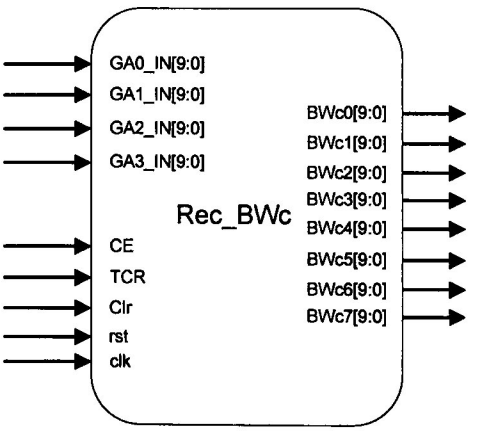


Figura 4.23.Diagrama de señales de entrada y de salida del bloque recursión backward

En la ecuación 2.26 se realiza el cálculo de la recursión backward, se puede apreciar que el cálculo de los valores actuales $B(s)_{k-1}$ dependen de la recursión siguiente

$B(s)_k$ y de las métricas de rama, por eso se le nombra hacia atrás. En la figura 4.23 se puede apreciar las entradas y las salidas que componen a este módulo.

Al igual que la recursión forward, ésta sólo tiene como datos de entrada el valor de las métricas de rama, en cambio para las entradas de control se tiene una nueva entrada llamada Clr, que tiene como propósito inicializar el contenido de los registros, para cada nueva ventana a procesar, el resto de las señales se describen en la tabla 4.12.

Nombre	Descripción	Conexión
Entradas		**Origen**
GA0_IN	Valor de la gama cero.	MuxBW.
GA1_IN	Valor de la gama uno.	MuxBW.
GA2_IN	Valor de la gama dos.	MuxBW.
GA3_IN	Valor de la gama tres.	MuxBW.
CE	Señal de habilitación del bloque.	MAP_Control.
TCR	Selector del tipo de tasa de codificación.	MAP_Control.
Rst	Señal de Reset.	Exterior.
Clk	Señal de reloj. Se considera el flanco de subida.	Exterior.
Salidas		**Destino**
BWc0	Resultado cero de la recursión Backward.	Rec_BWd, MuxBWd.
BWc 1	Resultado uno de la recursión Backward.	Rec_BWd, MuxBWd.
BWc 2	Resultado dos de la recursión Backward.	Rec_BWd, MuxBWd.
BWc 3	Resultado tres de la recursión Backward.	Rec_BWd, MuxBWd.
BWc 4	Resultado cuatro de la recursión Backward.	Rec_BWd, MuxBWd.
BWc 5	Resultado cinco de la recursión Backward.	Rec_BWd, MuxBWd.
BWc 6	Resultado seis de la recursión Backward.	Rec_BWd, MuxBWd.
BWc 7	Resultado siete de la recursión Backward.	Rec_BWd, MuxBWd.

Tabla 4.12. Descripción de las señales de entrada y de salida del bloque recursión backward de convergencia.

La recursión backward de convergencia al igual que la recursión forward requiere ser normalizada, ya que el método con el que se calculan es muy similar. Esto se puede observar en la figura 4.24, en donde se ilustra la estructura de esta recursión, los únicos cambios en la implementación, es el orden de interconexión entre las métricas de rama y los registros. Por otra parte las métricas de rama leídas de la memoria gama, entran en orden inverso, esto es controlado por los contadores que alimentan el bus de direcciones de las memorias de gama.

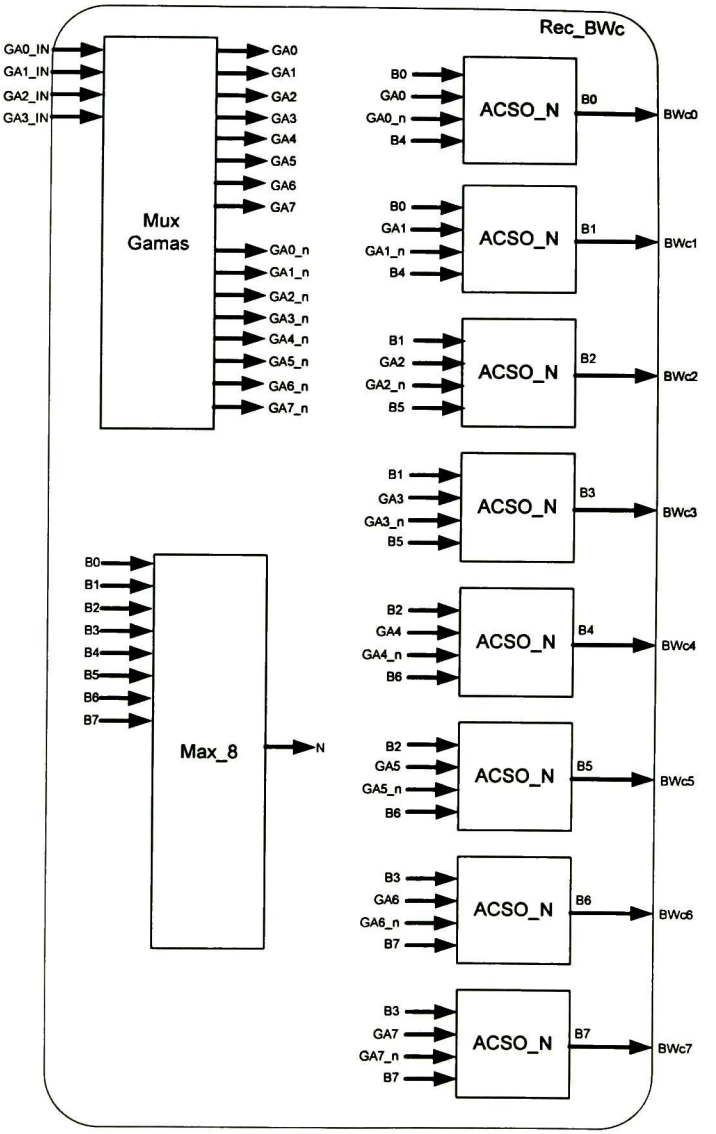


Figura 4.24. Estructura de la recursión Backward de convergencia.

Recursión Backward de decodificación

Del mismo modo que la recursión backward de convergencia, la recursión backward de decodificación está estructurada de acuerdo a la figura 4.24, sin embargo ésta tiene una serie de entradas de datos extras, tal como se muestra en la figura 4.25. El propósito de estas entradas es cargar a este bloque, con el estado inicial generado por la recursión

BWc, este proceso de inicialización es controlado por la entrada de control Clr, que tiene como función tomar los valores de las entradas y almacenarlos en los registros de datos.

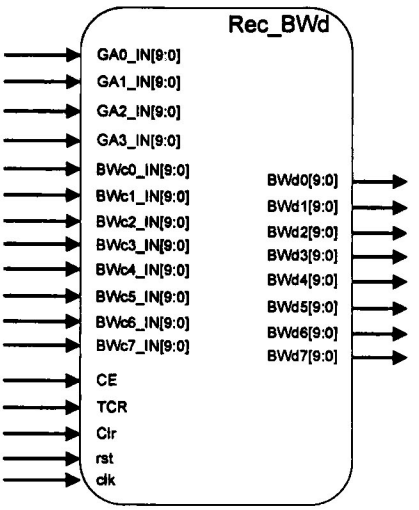


Figura 4.25. Diagrama de señales de entrada y de salida del bloque recursión backward de decodificación.

En la tabla 4.13 se describen e indican las conexiones para este módulo, de igual modo que la recursión BWc la entrada de las gamas entra en modo descendente.

Nombre	Descripción	Conexión
Entradas		**Origen**
GA0_IN	Valor de la gama cero.	MuxBW.
GA1_IN	Valor de la gama uno.	MuxBW.
GA2_IN	Valor de la gama dos.	MuxBW.
GA3_IN	Valor de la gama tres.	MuxBW.
BWc0_IN	Valor cero de la recursión Backward.	Rec_BWd.
BWc 1_IN	Valor uno de la recursión Backward.	Rec_BWd.
BWc 2_IN	Valor dos de la recursión Backward.	Rec_BWd.
BWc 3_IN	Valor tres de la recursión Backward.	Rec_BWd.
BWc 4_IN	Valor cuatro de la recursión Backward.	Rec_BWd.
BWc 5_IN	Valor cinco de la recursión Backward.	Rec_BWd.
BWc 6_IN	Valor seis de la recursión Backward.	Rec_BWd.
BWc 7_IN	Valor siete de la recursión Backward.	Rec_BWd.
CE	Señal de habilitación del bloque.	MAP_Control.
TCR	Selector del tipo de tasa de codificación.	MAP_Control.
CLR	Señal de inicialización de estados.	MAP_Control.
Rst	Señal de Reset.	
Clk	Señal de reloj. Se considera el flanco de	

	subida.	
Salidas		**Destino**
BWd0	Resultado cero de la recursión Backward.	MuxBWd.
BW d1	Resultado uno de la recursión Backward.	MuxBWd.
BW d2	Resultado dos de la recursión Backward.	MuxBWd.
BW d3	Resultado tres de la recursión Backward.	MuxBWd.
BW d4	Resultado cuatro de la recursión Backward.	MuxBWd.
BW d5	Resultado cinco de la recursión Backward.	MuxBWd.
BW d6	Resultado seis de la recursión Backward.	MuxBWd.
BW d7	Resultado siete de la recursión Backward.	MuxBWd.

Tabla 4.13. Descripción de las señales de entrada y de salida del bloque recursión backward de decodificación.

LLR

El bloque del LLR es el último bloque de proceso, toma los datos generados por los módulos descritos anteriormente y los procesa para generar el LLR, tal como se puede observar en la figura 4.26, la única señal de control que tiene este módulo es la selección de la tasa de codificación.

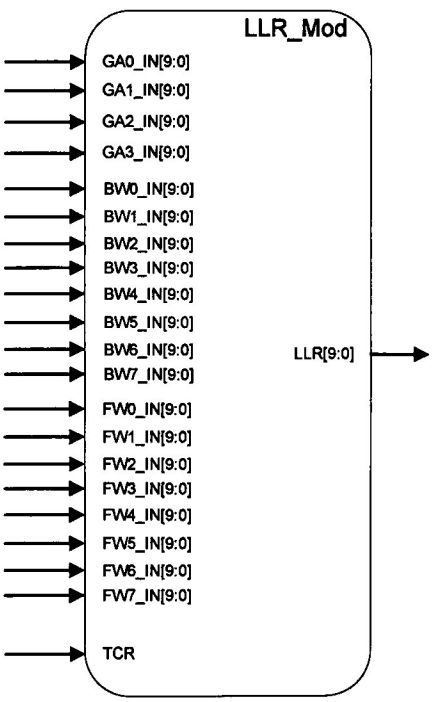


Figura 4.26. Diagrama de señales de entrada y de salida del bloque LLR

La entrada de las gamas, al igual que los datos provenientes de la recursión backward de decodificación entran en orden descendente, ésta es otra de las razones por lo que fue necesario almacenar los datos generados por la recursión forward. En la tabla 4.14 se describe e ilustra las conexiones de este módulo, se puede observar que la entrada de las recursión BW viene de un multiplexor, el cual tiene como función seleccionar al módulo backward de convergencia sólo en el caso que se esté procesando la ultima ventana de la trama.

Nombre	Descripción	Conexión
Entradas		**Origen**
GA0_IN	Valor de la gama cero.	MuxLLR.
GA1_IN	Valor de la gama uno.	MuxLLR.
GA2_IN	Valor de la gama dos.	MuxLLR.
GA3_IN	Valor de la gama tres.	MuxLLR.
BW0_IN	Valor cero de la recursión Backward.	MuxBWd.
BW 1_IN	Valor uno de la recursión Backward.	MuxBWd.
BW 2_IN	Valor dos de la recursión Backward.	MuxBWd.
BW 3_IN	Valor tres de la recursión Backward.	MuxBWd.
BW 4_IN	Valor cuatro de la recursión Backward.	MuxBWd.
BW 5_IN	Valor cinco de la recursión Backward.	MuxBWd.
BW 6_IN	Valor seis de la recursión Backward.	MuxBWd.
BW 7_IN	Valor siete de la recursión Backward.	MuxBWd.
FW0	Valor cero de la recursión forward.	FW_Mem.
FW1_IN	Valor uno de la recursión forward.	FW_Mem.
FW2_IN	Valor dos de la recursión forward.	FW_Mem.
FW3_IN	Valor tres de la recursión forward.	FW_Mem.
FW4_IN	Valor cuatro de la recursión forward.	FW_Mem.
FW5_IN	Valor cinco de la recursión forward.	FW_Mem.
FW6_IN	Valor seis de la recursión forward.	FW_Mem.
FW7_IN	Valor siete de la recursión forward.	FW_Mem.
TCR	Selector del tipo de tasa de codificación.	Exterior.
Salidas		**Destino**
LLR	Resultado del bloque LLR	LLR_Mem.

Tabla 4.14. Descripción de las señales de entrada y de salida del bloque LLR.

En la figura 4.27 se muestra la estructura del bloque LLR, como se puede observar éste está compuesto únicamente por operadores MAX*. Del mismo modo que en las recursiones y el cálculo de las métricas de rama, éste cuenta con un bloque de multiplexores encargado de seleccionar la tasa de codificación asignada por el estándar seleccionado.

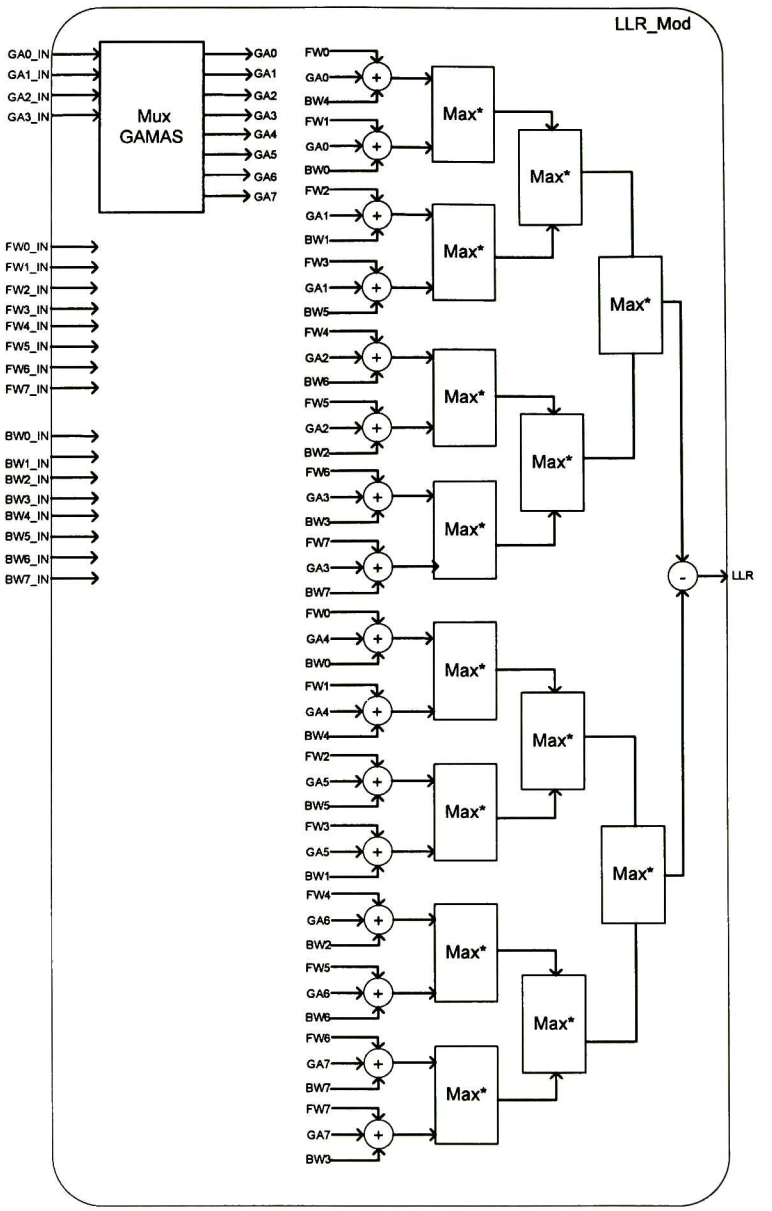


Figura 4.27. Estructura del cálculo del LLR.

Memoria para los daros LLR

Es el bloque final del decodificador MAP, tiene como propósito ordenar la secuencia entregada por el módulo LLR, debido a que este proceso genera los datos en orden inverso, por tanto esta memoria graba los valores del LLR en forma ascendente y los lee

en forma descendente, al igual que la memoria FW la cual escribe la dirección que lee y mantiene su propio contador interno, por lo que es necesario reiniciar el contador al terminar de procesar una trama.

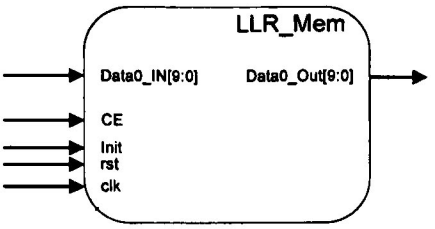


Figura 4.28. Diagrama de señales de entrada y de salida del bloque memoria LLR.

En la figura 4.28 se muestra la disposición de las entradas y las salidas del módulo de memoria del LLR, mismas que se describen en la tabla 4.15.

Nombre	Descripción	Conexión
Entradas		**Origen**
Data0_IN	Datos de entrada a la memoria.	LLR_Mod
CE	Señal de habilitación del bloque.	MAP_Control.
Init	Señal de inicialización de la dirección de escritura.	MAP_Control.
Rst	Señal de Reset.	Exterior.
Clk	Señal de reloj. Se considera el flanco de subida.	Exterior
Salidas		**Destino**
Data0_OUT	Datos de salida de la memoria posición.	Salida del MAP

Tabla 4.15. Descripción de las señales de entrada y de salida del bloque de memoria del LLR.

Unidad de control

Como se puede observar las entradas de los módulos anteriormente descritos, están compuestos por señales de datos y señales de control, estás últimas tienen como propósito manejar y gestionar el funcionamiento de cada módulo. La unidad de control tiene como fin manejar todas las señales de habilitación que componen al decodificador MAP. En la figura 4.29 se muestra el diagrama de entradas y salidas del bloque de control del decodificador MAP, también se pude observar que cuenta con algunas señales de entrada, éstas aportan los estímulos que generan una respuesta en las salidas del módulo.

Algunos de los elementos de control se encuentran fuera de este módulo, tal es el caso de los contadores de las memorias gamas, que tienen como función alimentar el bus de direcciones de las memorias, estos contadores son de 6 bits, y cuentan en forma

ascendente y descendente en forma alternada. Se cuenta con dos contadores sincronizados de tal forma que ambos corren en forma contraria, en otras palabras si uno cuenta en forma ascendente, el otro cuenta en forma descendente. El primer contador alimenta al memoria 1 y 3, mientras que el segundo a la 2 y 4, ya que como se puede observar en la figura 2.13 las memorias llevan la misma dirección en los pares indicados.

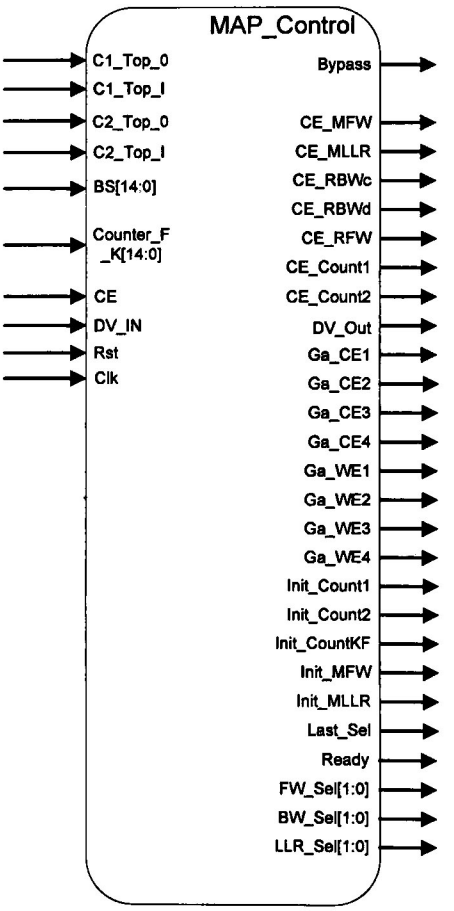


Figura 4.29. Diagrama de señales de entrada y de salida del control del MAP

Los contadores de gamas también tiene como función indicar al control cuando terminan las ventanas, a través de las señales Cx_TOP_x que se describen la tabla 4.16, junto con el resto de las señales que conforman el control.

Nombre	Descripción
Entradas	
C1_TOP_0	Señal indicadora del tope inferior del contador 1.
C1_TOP_I	Señal indicadora del tope superior del contador 1.

C2_TOP_0	Señal indicadora del tope inferior del contador 2.
C2_TOP_I	Señal indicadora del tope superior del contador 2.
BS	Indicador de tamaño del bloque a decodificar.
Counter_F_K	Contador del bloque de salida.
DV_IN	Señal de validez de datos.
CE	Señal de habilitación del bloque.
Rst	Señal de Reset.
Clk	Señal de reloj. Se considera el flanco de subida.
Salidas	
Bypass	Señal que controla el paso directo de los valores de gama.
CE_MFW	Señal de habilitación para la memoria que almacena la recursión FW.
CE_MLLR	Señal de habilitación para la memoria que almacena el LLR.
CE_RBWc	Señal de habilitación para la recursión BWc.
CE_RBWd	Señal de habilitación para la recursión BWd.
CE_RFW	Señal de habilitación para la recursión FW.
CE_Count1	Señal de habilitación para el contador 1.
CE_Count2	Señal de habilitación para el contador 2.
DV_Out	Señal de indicación de validez de datos de salida.
GA_CE1	Señal de habilitación para la memoria de gamas 1.
GA_CE2	Señal de habilitación para la memoria de gamas 2.
GA_CE3	Señal de habilitación para la memoria de gamas 3.
GA_CE4	Señal de habilitación para la memoria de gamas 4.
GA_WE1	Señal de habilitación de escritura para la memoria de gamas 1.
GA_WE2	Señal de habilitación de escritura para la memoria de gamas 2.
GA_WE3	Señal de habilitación de escritura para la memoria de gamas 3.
GA_WE4	Señal de habilitación de escritura para la memoria de gamas 4.
Init_Count1	Señal de de re inicialización para el contador 1.
Init_Count2	Señal de de re inicialización para el contador 2.
Init_CountKF	Señal de de re inicialización para el contador del bloque de salida.
Init_MFW	Señal de de re inicialización para los contadores de la memoria FW.
Init_MLLR	Señal de de re inicialización para los contadores de la memoria LLR.
Last_Sel	Selector de la venta final.
FW_Sel	Selector de ventana de la cual se realizará la recursión FW.
BW_Sel	Selector de ventana de la cual se realizará la recursión BW.
LLR_Sel	Selector de ventana de la cual se realizará la recursión LLR.
Ready	Señal de indicación de módulo listo.

Tabla 4.16 Descripción de las señales de entrada y de salida del bloque del control del MAP

La señal de Counter_F_K, es la salida de otro contador, que inicia su conteo por medio de la señal DV_Out, tiene como propósito indicar al control cuando se terminó de entregar la trama, una vez que esto sucede reinicia todos los módulos y espera por la siguiente trama a procesar. En la figura 4.30 se muestra la máquina de estados encargada de controlar el decodificador MAP, las secuencias de salida son una serie de pasos que realiza el control, de tal modo que se realice el proceso final para cada trama. Como se puede observar para ventanas mayores a 253 los pasos se repiten tantas veces como sea necesario para cubrir todo el bloque a decodificar, por lo tanto cuando el dato valido desaparece, el sistema de control procede a iniciar una secuencia de terminación de

trama dependiendo de la venta en donde ocurrió el fin, de este modo no importa el tamaño de la trama sólo la ventana en donde ocurra el final.

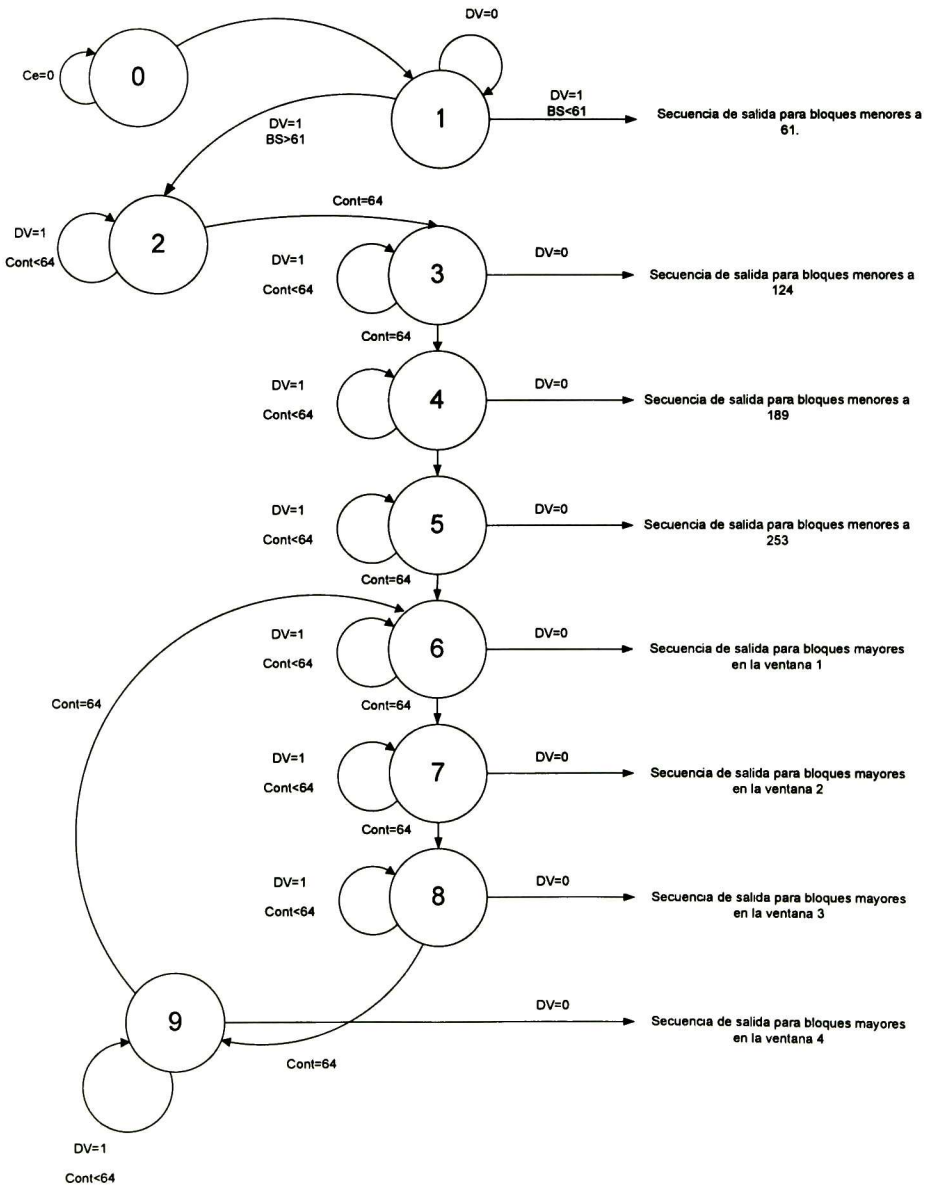


Figura 4.30. Diagrama de estados del control MAP

4.3.1.2.2 Interleaver/DeInterleaver

Retomando el análisis de la figura 4.9, el siguiente bloque a describir es el Interleaver y Deinterleaver, que tiene como función de-correlacionar las entradas de cada uno de los decodificadores, dicho de otra forma, cambia el orden en el que entran los datos al decodificador. De tal modo que si un decodificador no tiene la capacidad de corregir un símbolo, es posible que el segundo si pueda. Sin embargo es necesario que el interleaver utilizado por el codificador sea el mismo que el usado durante la decodificación, de lo contrario la información que toma cada decodificador no coincidirá. Cada estándar cuenta con su propio método de entrelazado por lo que es necesario tener un interleaver para cada estándar. Lo que implica que cada uno de estos interleavers tenga sus propias memorias de datos, esto representa una gran cantidad de memoria. La implementación de este módulo permite compartir las memorias de datos para ambos estándares, en la figura 4.31 se muestras el diagrama de las señales de entrada y de salida.

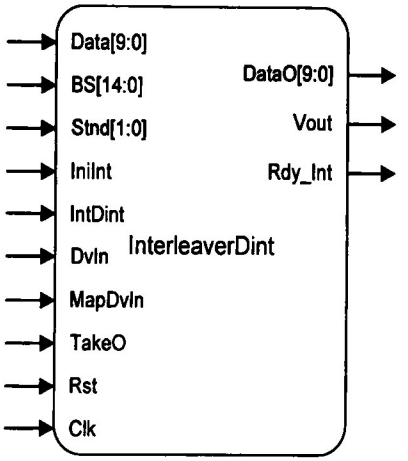


Figura 4.31. Diagrama de señales de entrada y de salida del bloque Interleaver/Deinterleaver.

Este módulo toma los datos a través de la entrada Data, y los entrega entrelazados o de-entrelazados por la salida DataO, dependiendo de la configuración dada por medio de las entradas IniInt o bien IniDint. En la tabla 4.17 se describen el resto de las señales de entrada y salida usadas por este bloque.

Nombre	Descripción	Conexión
Entradas		**Origen**
Data	Datos de entrada.	Mux_LLR
BS	Indicador del tamaño de bloque.	Exterior MAPpi
Stnd	Selecciona el estándar que se está manejando.	Exterior MAPpi
IniInt	Señal de configuración del Interleaver	Exterior MAPpi

IntDInt	Señal que selecciona el modo de operación del módulo, cero para Interleaver y uno para deinterleaver.	Exterior MAPpi
DVIn	Señal de validez de datos.	MAP
TakeO	Señal que selecciona el momento para comenzar a arrojar los datos.	MAP
Salidas		**Destino**
DataO	Bus de datos de salida.	Exterior MAPpi
Vout	Señal de validez de datos.	Exterior MAPpi
Rdy_Int	Señal de indicación de módulo listo.	Exterior MAPpi

Tabla 4.17. Descripción de las señales de entrada y de salida del bloque Interleaver/Deinterleaver.

Con el motivo de disminuir el uso de memoria, se propone que los estándares soportados compartan las memorias de datos y cada estándar maneje su propio generador de direcciones. Tal como se muestra en la figura 4.32 donde se puede apreciar la estructura del interleaver. La razón por la cual se utilizan dos memorias de datos es con el fin de que este módulo sea capaz de entregar una trama entrelazada al mismo tiempo que recibe la siguiente trama, esto es necesario para cumplir con la requisición **FUN-TDEC-06**, descrita en el capítulo 3.

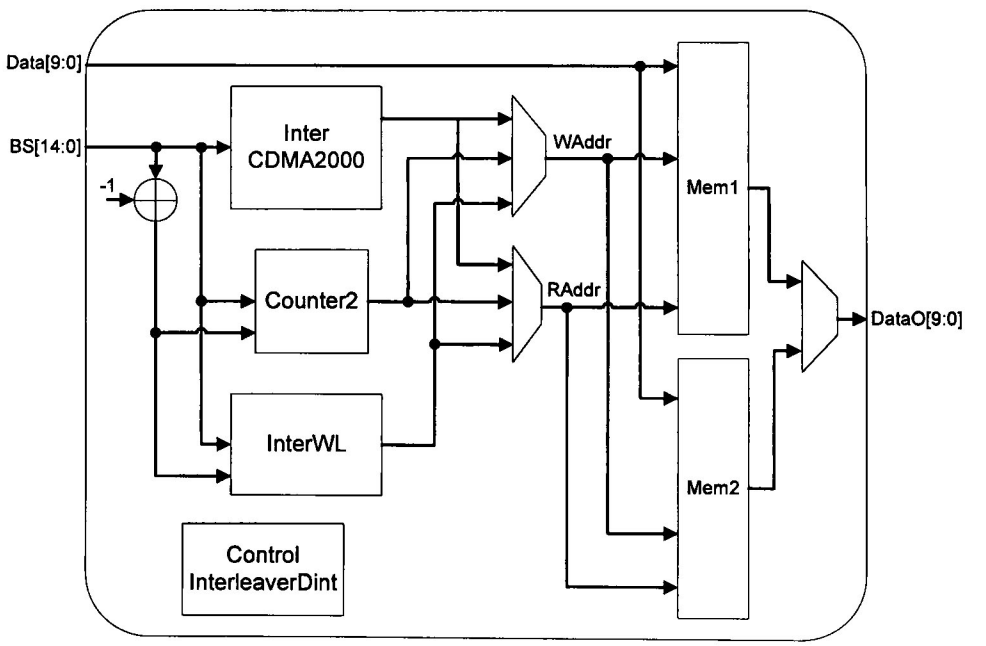


Figura 4.32. Estructura del Interleaver/Deinterleaver.

Este módulo está compuesto por dos generadores de direcciones, uno para el estándar CDMA2000 especificado en [8] y otro para el WCDMA además de un contador, estos módulos alimentan el bus de direcciones de las memorias, los multiplexores son los encargados de seleccionar el estándar que se está manejando así como la función.

En la figura 3.33 se puede observar las entradas y salidas del generador de direcciones para el estándar WCDMA, el cual se encuentra descrito en [22], en la tabla 4.18 se describen cada una de las señales que describen a este módulo.

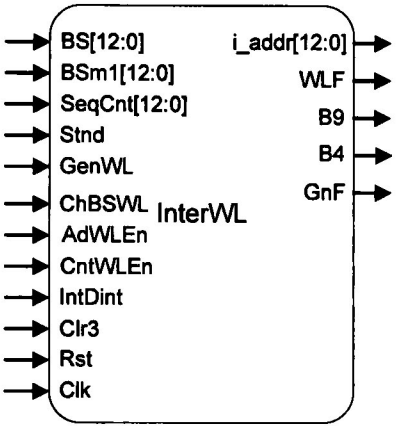


Figura 4.33. Diagrama de señales de entrada y de salida del generador de direcciones WCDMA.

Nombre	Descripción	Conexión
Entrada		**Origen**
BS	Indicador del tamaño de bloque.	Exterior InterleaverDint
BSm1	Indicador del tamaño de bloque menos uno.	Adder
Stnd	Selecciona el estándar que se está manejando.	Exterior InterleaverDint
GenWL	Señal para indicar que se comienzan a generar las direcciones.	Control InterleaverDint
ChBSWL	Señal de configuración.	Control InterleaverDint
AdWLEn **CntWLEn**	Señales permite la obtención de las direcciones.	Control InterleaverDint
IntDint	Señal que selecciona el modo de operación del módulo, cero para Interleaver y uno para deinterleaver.	Exterior InterleaverDint
Clr3	Señal de re inicialización.	Control InterleaverDint
Salidas		**Destino**
i_addr	Direcciones generadas.	Mux Raddr Mux Waddr
WLF	Señal de validez de datos	Control InterleaverDint

B4	Indica que el bit 5 tiene valor de uno.	Control InterleaverDint
B9	Señal de indicación de módulo listo.	Control InterleaverDint
GNF	Indica que ya han sido generadas todas las direcciones.	Control InterleaverDint

Tabla 4.18. Descripción de las señales de entrada y de salida del generador de direcciones WCDMA

4.3.2 Control General

Este módulo tiene como fin manejar las señales de habilitación de cada uno de los bloques que conforman el turbo decodificador, tal como se puede apreciar en la figura 4.34 las señales de entrada del lado izquierdo son las encargadas de manejar el control, mientras que del lado derecho se tienen las señales de habilitación o selección que manejan el resto del turbo decodificador.

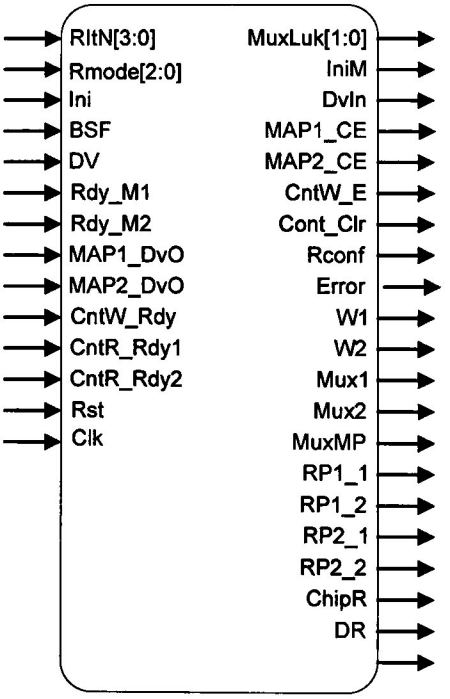


Figura 4.34. Diagrama de señales de entrada y de salida del control general.

De acuerdo con la especificación de los requerimientos expuestos en el capítulo 3, el turbo decodificador tiene la capacidad de ser reconfigurable, esta capacidad es manejada por el control general, por medio de algunas de sus entradas toma los

parámetros de configuración y genera el control pertinente. En la tabla 4.19 se encuentran descritas las señales de entrada y salida que conforman a este módulo.

Nombre	Descripción
Entradas	
RItN	Selecciona el número de iteraciones.
RMode	Selecciona el modo de operación
Ini	Señal para la configuración del turbo decodificador.
BSF	Señal que selecciona el inicio de del bloque.
DV	Señal de validez de datos.
Rdy_M1, Rdy_M2	Señal que indica que los dos MAPpi ya están configurados.
MAP1_DvO **MAP2_DvO**	Señal que indica que los datos de salida de los MAPpi son validos. Se mantiene durante Nturbo + 3 ciclos de reloj.
MAP2_DvOInt	Señal que indica que los datos de salida del MAPpi2 son validos. Se mantiene durante Nturbo ciclos de reloj.
CntR_Rdy1 **CntR_Rdy2** **CntW_Rdy**	Señal que indica que el contador de lectura escritura han llegado a la cuenta de Nturbo + 3.
Salidas	
MuxLuk	Maneja la procedencia de la información a-priori para el primer MAPpi.
IniM	Señal de configuración de los MAPpi.
DvIn	Señal de validez para el módulo MAPpi1.
MAP1_CE **MAP2_CE**	Señal de habilitación de los MAPpi.
CntW_E	Señal de habilitación del contador de escritura.
Cnt_Clr	Señal de re inicialización de los contadores.
Rconf	Señal habilitadora del registro de configuración.
Error	Señal que indica si la señal DV no se mantiene durante el la entrada del bloque de datos.
W1 **W2**	Señal para la escritura de memorias.
Mux1, Mux2 **MuxMP**	Señal para el manejo de los multiplexores.
RP1_1, RP1_2 **RP2_1, RP2_2**	Señales para el manejo de los contadores de lectura de las memorias.
ChipR	Señal que indica que el módulo ha sido configurado y que está listo para recibir un bloque de datos.
DR	Indica que los valores proporcionados en las salidas DO, LLRext, YsO, Yp1O, Yp2O, Yp1pO y Yp2pO son válidos.
MAP_LLR	Señal que indica al MAPpi2 si debe entregar el LLR o la información extrinseca.

Tabla 4.19. Descripción de las señales de entrada y de salida del control general.

El control general está compuesto por 4 módulos principales, que se muestran en la figura 4.35 en donde se ilustra la estructura del control, la razón de la separación se debe al gran número de estados necesarios para el control, todos los componentes y funciones del turbo decodificador. El bloque LLRCtrl es puramente combinacional y tiene como función controlar la salida del segundo MAPpi ya sea extrínseco o LLR, el módulo

Maq0 tiene como tareas la configuración del turbo decodificador, controla la operación del MAPpi1, además es el encargado de grabar las memorias de datos, para el caso de Maq1 se encarga del control del bloque MAPpi2, por lo que también se encarga de la lectura de las memorias de datos, por último la máquina de estados Maq2 tiene como función el manejo de la retroalimentación de la información a-priori, y para el caso de dispositivos concatenados, es el encargado de la trasmisión de la información al siguiente módulo.

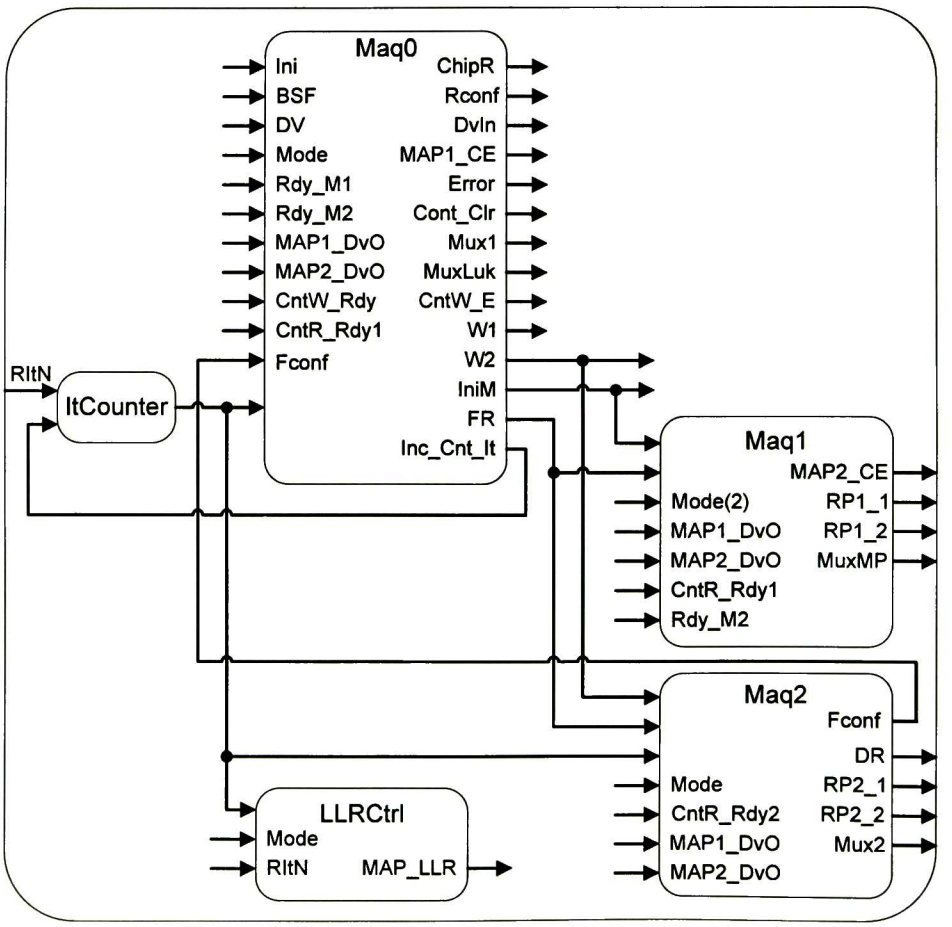


Figura 4.35. Estructura del control general del turbo decodificador.

En la figura 4.36 se muestra el diagrama de estados con el que está compuesto el primer bloque de control Maq0, el cual está compuesto por cuatro estados:

S0 Estado inicial en espera de la señal de inicialización para almacenar los datos de configuración al registro.

S1 Estado para la habilitación de la configuración de los bloques MAPpi.

S2 Estado de espera de la configuración del bloque MAPpi.

S3 Es el estado de espera de datos válidos, este estado es el que genera la señal ChipR, la cual indica que el dispositivo está listo para recibir un bloque de datos o para cambiar la configuración del dispositivo. Para cambiar la configuración del turbo decodificador las entradas Fconf e Ini deben ser igual a uno y pasa al estado S1, cargando los datos de configuración al registro durante la transición. Por último este estado es el punto de partida para los modos de operación, de tal forma que si DV y BSF son igual a uno se inicia el proceso de decodificación y dependiendo de la configuración Mode se selecciona a cualquiera de los 3 modos de operación: decodificación sencilla, dobles y de dispositivos concatenados.

Esta máquina es la encargada de manejar las otras dos máquinas de estados, tal como se muestra en la figura 4.35, en donde se pueden observar algunas dependencias de las máquinas Maq0 y Maq1, con algunas de las salidas de esta máquina de estados.

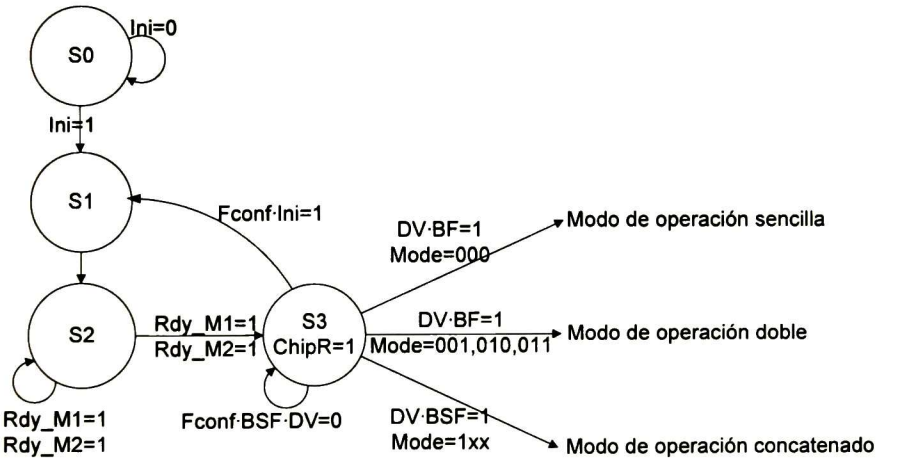


Figura 4.36. Diagrama de estados para el control Maq0

La máquina de estados Maq1 mostrada en la figura 4.37, es controlada por la primera máquina, y es capaz de operar en cualquier modo, al igual que Maq2 mostrada en la figura 4.38 depende de Maq0, también es capaz de trabajar con cualquier modo de operación, Se encarga de habilitar la señal DR, la cual indica la validez de los datos en el

puerto DO, además en el modo de operación de dispositivos concatenados ésta sirve para indicar al siguiente Turbo decodificador que hay un dato valido.

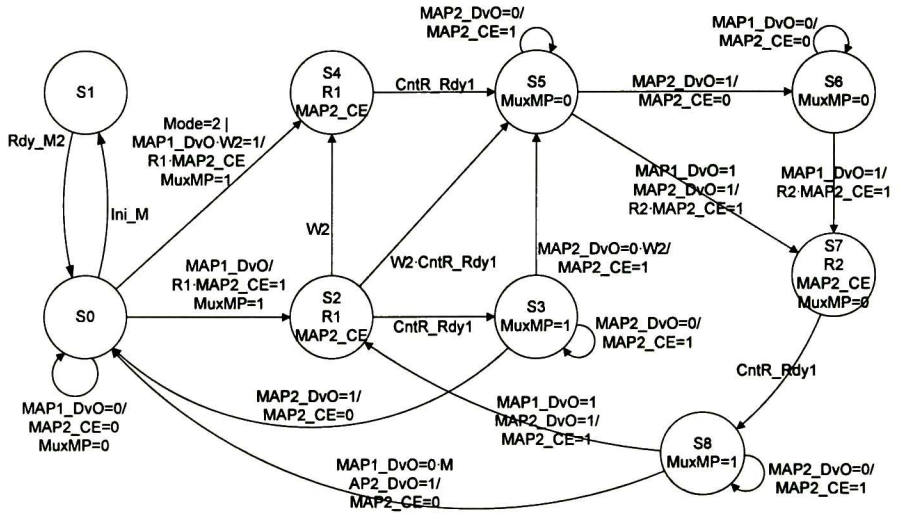


Figura 4.37. Diagrama de estados para el control Maq1

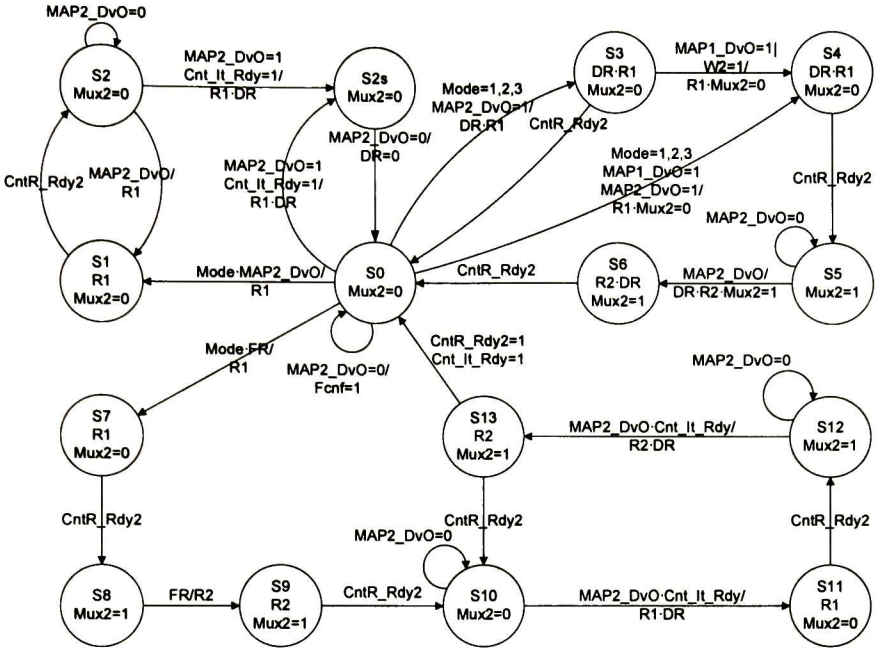


Figura 4.38. Diagrama de estados para el control Maq2

Capítulo V

Pruebas y Resultados.

La verificación consiste en comprobar que el diseño satisface su especificación y entrega la funcionalidad esperada [23], esto se realiza con una serie de camas de pruebas que tienen como objetivo comprobar cada una de las funcionalidades especificadas en el capítulo 3. Esto se realiza mediante la inspección del código desarrollado, ya sea en forma manual o con ayuda de las herramientas de desarrollo como ModelSim y Quartus que advierten sobre posibles problemas o incongruencias encontradas en el código, otra forma de verificación consiste probar la funcionalidad del diseño, ya sea por medio de simulaciones o bien pruebas físicas.

5.1 Verificación funcional

Esta verificación tiene como objetivo comprobar que los requerimientos funcionales sean realizados de manera correcta. Esto se realiza por medio de camas de pruebas implementadas en ModelSim, en algunos de los casos de prueba se realiza mediante el uso de un FPGA [21,20]. En la figura 5.1 se muestra la organización de las camas de prueba usadas para la verificación del turbo decodificador.

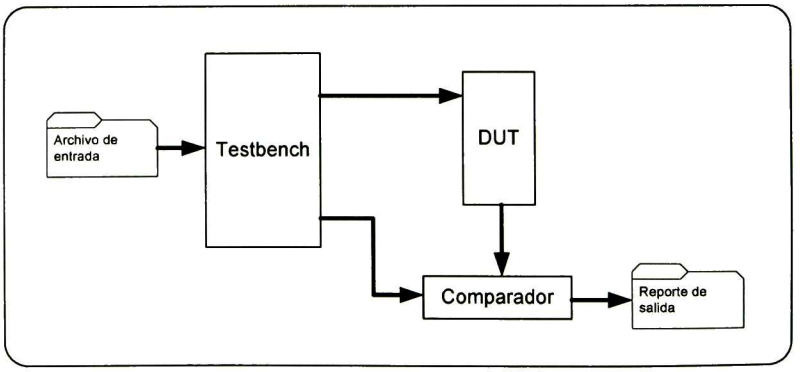


Figura 5.1. Organización del testbench.

Los archivos de entrada se generaron por medio de un programa realizado en C, el cual emula el funcionamiento del turbo decodificador implementado, al igual que el diseño este emplea el uso de ventanas deslizantes para optimizar el uso de memoria, también fue diseñado para operar con los diferentes esquemas propuestos.

5.1.1 Casos de prueba

En las siguientes tablas se muestra una descripción de los casos de prueba utilizados para la verificación funcional del turbo decodificador, esta descripción indica el tipo de prueba que se realizará, una descripción de la prueba, las entradas del DUT que estimula la prueba, una especificación del contenido de las entradas, la información esperada a la salida del dispositivo bajo prueba y los requerimientos que son cubiertos con dicha prueba.

Algunos de los casos de pruebas se realizaron en forma física, descargando el diseño en un FPGA, utilizando la estructura mostrada en la figura 5.2, en la cual se utilizo la ayuda de un procesador NIOS II para realzar la interfaz con la computadora [26]. Los datos son escritos en la memoria por el NIOS II, una vez que este proceso termina el control lee los datos de la memoria y se los entrega al turbo descodificador, posteriormente escribe los resultados del decodificador en la memoria, final mente el NIOS II lee estos datos y los entrega a la computado de regreso.

TC-TDEC-01	**Configuración de parámetros.**
Tipo de prueba	**Prueba funcional.**
Descripción	**Introducir los parámetros de configuración en las entradas correspondientes, mandar el pulso de inicialización y esperar por la señal de dispositivo listo.**
Entradas requeridas	**Mode.- Modo de operación,** **BS.- Tamaño de la trama,** **Stnd.- Estándar,** **ItN.- Número de iteraciones,** **Ini.- Señal de inicialización.**

	Rst.- Reset.
Especificación de entrada	**1.- Rst = 1;** **2.- Mode = 0, BS = 256, Stnd = 0, ItN = 5;** **3.- Ini = 1;** **4.- Ini = 0;**
Especificación de salida	**En las salidas del turbo decodificador se debe esperar la señal ChipR = 1, que indica que el módulo esta listo para ser usado.**
Requerimientos cubiertos	**RF-TDEC-01, RF-TDEC-02,RF-TDEC-03, RF-TDEC-04, RF-TDEC-10.**
Comentarios	**Parámetros de configuración:** • **Estándar** • **Tamaño de bloque** • **Número de iteraciones** • **Modo de operación** **Inicialización(Carga de parámetros)**

Tabla 5.1. Descripción del Caso de prueba TC-TDEC-01

TC-TDEC-02	**Disponibilidad del módulo**
Tipo de prueba	**Prueba funcional.**
Descripción	**Introducir un bloque de datos, verificar que la señal ChipR sea cero durante el proceso, y uno cuando se concluya la decodificación.**
Entradas requeridas	**DV, BSF, Señales de datos.**
Especificación de entrada	**1.- Dv = 1, BSF = 1, Datos ;** **2.- Dv = 1, BSF = 0, Datos;** **3.- Dv = 0, BSF = 0;**
Especificación	**En las salidas del turbo decodificador se debe esperar la señal ChipR = 1, que indica que el módulo ya termino de decodificar el**

de salida	**bloque, y esta listo para recibir otro.**
Requerimientos cubiertos	**RF-TDEC-01, RF-TDEC-02,RF-TDEC-03, RF-TDEC-04, RF-TDEC-8, RF-TDEC-10.**
Comentarios	**El turbo decodificador fue configurado previamente.**

Tabla 5.2. Descripción del Caso de prueba TC-TDEC-02

TC-TDEC-03	**Decodificación Sencilla**
Tipo de prueba	**Prueba funcional.**
Descripción	**Se generan bloques de datos en un programa de software, que posteriormente alimentaran al turbo decodificador implementado en una tarjeta de desarrollo, la cual con ayuda de un microprocesador embebido en el FPGA, cuenta los errores y calcula el BER.**
Entradas requeridas	**DV, BSF, Señales de datos.**
Especificación de entrada	**Figura 5.2**
Especificación de salida	**El BER obtenido por el turbo decodificador.**
Requerimientos cubiertos	**RF-TDEC-01, RF-TDEC-02,RF-TDEC-03, RF-TDEC-04, RF-TDEC-05, RF-TDEC-06, RF-TDEC-08, RF-TDEC-10.**
Comentarios	

Tabla 5.3. Descripción del Caso de prueba TC-TDEC-03

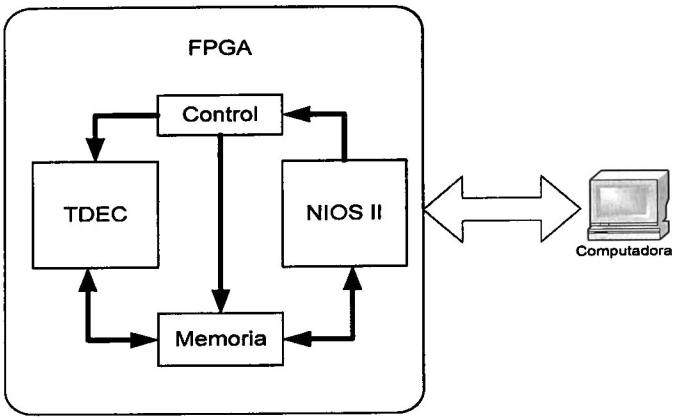


Figura 5.2. Diagrama de implementación para pruebas físicas.

TC-TDEC-04	Decodificación doble
Tipo de prueba	Prueba funcional.
Descripción	Se comprueba que el turbo decodificador tenga la capacidad de procesar dos bloques en forma simultanea,
Entradas requeridas	Señales de datos.
Especificación de entrada	Señales de datos obtenido mediante el generador de códigos.
Especificación de salida	Se comparan los resultados obtenidos con los generados por la decodificación sencilla.
Requerimientos cubiertos	RF-TDEC-01, RF-TDEC-02, RF-TDEC-03, RF-TDEC-04, RF-TDEC-05, RF-TDEC-06, RF-TDEC-07, RF-TDEC-08, RF-TDEC-10.
Comentarios	El turbo decodificador fue previamente configurado con los parámetros adecuados.

Tabla 5.4. Descripción del Caso de prueba TC-TDEC-04.

TC-TDEC-05	Decodificación concatenada
Tipo de prueba	Prueba funcional.

Descripción	**Se comprueba que el turbo decodificador tenga la capacidad de procesar un bloque de datos y realizará únicamente una iteración pasando el resultado a otros turbo decodificadores concatenados los cuales realizan las iteraciones posteriores.**
Entradas requeridas	**Señales de datos**
Especificación de entrada	**Señales de datos obtenido mediante el generador de códigos.**
Especificación de salida	**Se comparan los resultados obtenidos con los generados por la decodificación sencilla.**
Requerimientos cubiertos	**RF-TDEC-01, RF-TDEC-02, RF-TDEC-03, RF-TDEC-04, RF-TDEC-05, RF-TDEC-06, RF-TDEC-07, RF-TDEC-08, RF-TDEC-10.**
Comentarios	**El turbo decodificador fue previamente configurado con los parámetros adecuados.**

Tabla 5.5. Descripción del Caso de prueba TC-TDEC-05.

TC-TDEC-06	**Detección de errores**
Tipo de prueba	**Prueba funcional.**
Descripción	**Se introduce un bloque de diferente tamaño al configurado, y se espera que el turbo decodificador detecte e indique la incongruencia a través de la señal de error.**
Entradas requeridas	**Señales de datos**
Especificación de entrada	**1.- Dv = 1, BSF = 1, Datos ;** **2.- Dv = 1, BSF = 0, Datos;** **3.- Dv = 0, BSF = 0;**
Especificación de salida	**Error = 1**
Requerimientos cubiertos	**RF-TDEC-01, RF-TDEC-02, RF-TDEC-03, RF-TDEC-04, RF-TDEC-05, RF-TDEC-06, RF-TDEC-09.**
Comentarios	

Tabla 5.6. Descripción del Caso de prueba TC-TDEC-06.

5.2 Resultados

En esta sección se muestran los resultados obtenido por el trabajo realizado, se comprobarán los casos de prueba, se realizará la presentación de los resultados de síntesis por medio de la herramientas dispuestas en el paquete Quartus II, por último se analiza el desempeño del turbo decodificador respecto a otros productos existentes en el mercado. Estas pruebas de desempeño requieren la decodificación de grandes cantidades de bits, sin embargo la velocidad de proceso en medios simulados como ModelSim deja mucho que desear, por esta razón estas pruebas fueron realizadas por una implementación en hardware que nos permiten procesar un bloque en cuestión de segundos en vez de minutos.

5.2.1 Casos de prueba

Cada uno de los casos de pruebas se verificó mediante el uso de una cama de prueba, tal como se muestra en la tabla 5.7, en donde se muestra el nombre de las diferentes camas de pruebas utilizadas para cada caso. En algunos casos se utilizó un segundo método de prueba, como en el caso de TC-TDEC-03, en el cual la prueba también se realizó por medio de la cama de prueba, posteriormente se analizó su desempeño con una serie de pruebas en hardware. Estas pruebas consisten en controlar el turbo decodificador por medio de un microprocesador embebido en el FPGA (NIOS II), el cual mantendrá una comunicación con una computadora que proporciona los datos codificados, este esquema se ilustra en la figura 5.2.

Número de TC	Archivo	Verificado	Resultado
TC-TDEC-01	TCW01.vhd	Si	Aprobado
TC-TDEC-02	TCW02.vhd	Si	Aprobado
TC-TDEC-03	TCWNiosII.c TCW03.vhd	Si	Aprobado
TC-TDEC-04	TCW04.vhd	Si	Aprobado
TC-TDEC-05	TCW05.vhd	Si	Aprobado
TC-TDEC-06	TCW06.vhd	Si	Aprobado

Tabla 5.7. Resultados del proceso de verificación.

A manera de comprobación de que todos los requerimientos son cubiertos por el proceso de verificación, se muestra la tabla 5.8, en la cual se entrelazan los requerimientos con respecto a los casos prueba.

	TC-TDEC-01	TC-TDEC-02	TC-TDEC-03	TC-TDEC-04	TC-TDEC-05	TC-TDEC-06
RF-TDEC-1	●	●	●	●	●	●
RF-TDEC-2	●	●	●	●	●	●
RF-TDEC-3	●	●	●	●	●	●
RF-TDEC-4	●	●	●	●	●	●
RF-TDEC-5			●	●	●	●
RF-TDEC-6			●	●	●	●
RF-TDEC-1	●	●	●	●	●	●
RF-TDEC-2	●	●	●	●	●	●
RF-TDEC-3	●	●	●	●	●	●
RF-TDEC-4	●	●	●	●	●	●
RF-TDEC-5			●	●	●	●
RF-TDEC-6			●	●	●	●

Tabla 5.8. Matriz de pruebas contra requerimientos.

5.2.2 Resultado de síntesis

El proceso de síntesis es una parte de del flujo de compilación, éste es el proceso mediante el cual se convierte el código de descripción de hardware como Verilog, VHDL, en una forma optimizada de elementos lógicos LEs (Logic Elements) o bien módulos lógicos adaptativos ALM (Adaptive Logic Module), así como otros bloques dedicados, este proceso también genera la base de datos que contiene todos los archivos incluidos en el proyecto [27].

Este proceso se realizó mediante la herramienta Quartus II 9.0, para el FPGA EP2S180F1020C3 de Altera que se describe en la tabla 5.9 [24].

Nombre	ALUTs	I/O	Memoria	DSP
EP2S180F1020C3	143,520	743	9,383,040	96

Tabla 5.9. Características del Stratix II.

Este dispositivo tiene la capacidad suficiente para contener el turbo decodificador y el procesador NIOS II utilizado para realizar las pruebas, además tiene una memoria de más de 9 Mb, suficientes para implementar todas las funciones descritas en el capítulo tres, esta capacidad permite realizar todos los tamaños de trama, requeridos por el estándar CDMA2000 [8]. Los resultados de la síntesis se exponen en la tabla 5.10, estos resultados no incluyen el procesador NIOS II utilizado para las pruebas, ni el interleaver para el estándar WCDMA expuesto en [22], por problema de sincronización durante las pruebas de HW, este bloque fue remplazado por una tabla de consulta (Lookup table).

Descripción	Resultado	Porcentaje
Elementos lógicos	12,310	9%
Registros utilizados	1,983	1%
Pines utilizados	104	14%
Bloques de memoria	1,425,207	15%
Bloques DSP (Multiplicadores)	8	1%
Frecuencia Máxima	35.34MHz	-
Latencia Máxima (CLk)	(2L+516)*Nit	

Tabla 5.10. Resultados de síntesis.

Otro factor importante a analizar en el desempeño del turbo decodificador es la latencia, este valor lo podemos definir como el número de ciclos de reloj que le toma al dispositivo entregar el primer dato de la trama decodificada. Éste se puede encontrar por medio de la ecuación mostrada en tabla 5.10, como se puede observar este valor depende del tamaño de la trama **L** y del número de iteraciones **Nit**, en la tabla 5.11, se muestra el número de ciclos de reloj que le toma al turbo decodificador entregar el resultado para algunos tamaños de trama y hasta 6 iteraciones.

BS/Nit	it 1	it 2	it 3	it 4	it 5	it 6
64	644	1288	1932	2576	3220	3864
128	772	1544	2316	3088	3860	4632
256	1028	2056	3084	4112	5140	6168
512	1540	3080	4620	6160	7700	9240
1024	2564	5128	7692	10256	12820	15384
2048	4612	9224	13836	18448	23060	27672
4096	8708	17416	26124	34832	43540	52248
5114	10744	21488	32232	42976	53720	64464

Tabla 5.11. Latencia del turbo decodificador.

Una vez que se conoce la latencia, es necesario calcular el throughput, que es el volumen de información que fluye en un sistema, éste mide la cantidad de bits que puede decodificar el módulo en un segundo, estos valores son mostrados en la tabla 5.12, para el modo de operación sencillo, los cuales fueron calculados para una frecuencia de reloj de 34Mhz, en la tabla 5.13 se muestra el throughput ahora para el modo de operación decodificación doble, en el cual se duplica la cantidad de bits decodificados, esto se debe que esta vez se están decodificando dos tramas sin embargo la latencia es la misma.

BS/Nit	it 1	it 2	it 3	it 4	it 5	it 6
64	3.478	1.739	1.159	0.87	0.696	0.58

128	5.803	2.902	1.934	1.451	1.161	0.967
256	8.716	4.358	2.905	2.179	1.743	1.453
512	11.64	5.818	3.879	2.909	2.327	1.939
1024	13.98	6.989	4.659	3.495	2.796	2.33
2048	15.54	7.771	5.181	3.886	3.108	2.59
4096	16.46	8.232	5.488	4.116	3.293	2.744
5114	16.66	8.33	5.553	4.165	3.332	2.777

Tabla 5.12. Throughput del turbo decodificador modo sencilla (Mbits/s).

BS/Nit	it 1	it 2	it 3	it 4	it 5	it 6
64	6.328	3.314	2.244	1.697	1.364	1.141
128	9.956	5.359	3.666	2.786	2.247	1.882
256	13.96	7.751	5.365	4.103	3.321	2.79
512	17.47	9.978	6.984	5.372	4.364	3.675
1024	19.98	11.65	8.224	6.355	5.178	4.369
2048	21.53	12.72	9.025	6.995	5.71	4.824
4096	22.39	13.33	9.488	7.365	6.019	5.089
5114	22.57	13.46	9.585	7.444	6.085	5.145

Tabla 5.13. Throughput del turbo decodificador modo doble (Mbits/s).

El último modo de operación, en la tabla 5.14 para dispositivos concatenados ya no depende del número de iteraciones, ya que el proceso se paraleliza en cada iteración, además cada uno de los dispositivos procesa dos tramas por vez, esto se puede observar en la figura 4.5, en la cual podemos ver el flujo de las tramas en este modo.

BS	Para cualquier Nit
64	6.957
128	11.61
256	17.43
512	23.27
1024	27.96
2048	31.08
4096	32.93
5114	33.32

Tabla 5.14. Throughput del turbo decodificador modo concatenado (Mbits/s).

Se puede apreciar que el diseño cubre los requerimientos del estándar, para el modo de operación de dos bloques simultáneos, con tramas mayores a 64 símbolos y 3 iteraciones. En el caso de dispositivos concatenados, siempre se cumple sin importar el número de iteraciones ni el tamaño del bloque.

5.2.3 Análisis de desempeño

En este bloque se presentarán los resultados de desempeño obtenidos por el turbo decodificador, se divide en dos partes, la primera compara el desempeño del esquema propuesto, y el segundo evalúa la implementación física, con las simulaciones en software, además realiza una comparación de desempeño con otros fabricantes. Estas pruebas se realizan mediante graficas del SNR contra BER.

5.2.3.1 Esquema propuesto

En la figura 4.1 mostrada en el capítulo anterior se muestra el esquema del turbo decodificador en donde se propone eliminar el interleaver usado para entrelazar los datos sistemáticos para segundo MAP. En la figura 5.3 se muestra una grafica en la cual se realiza la comparación del esquema original esq 1, en contra del esquema propuesto esq2, se puede observar que existe una degradación del orden de 1e-7 para un SNR de 1 db.

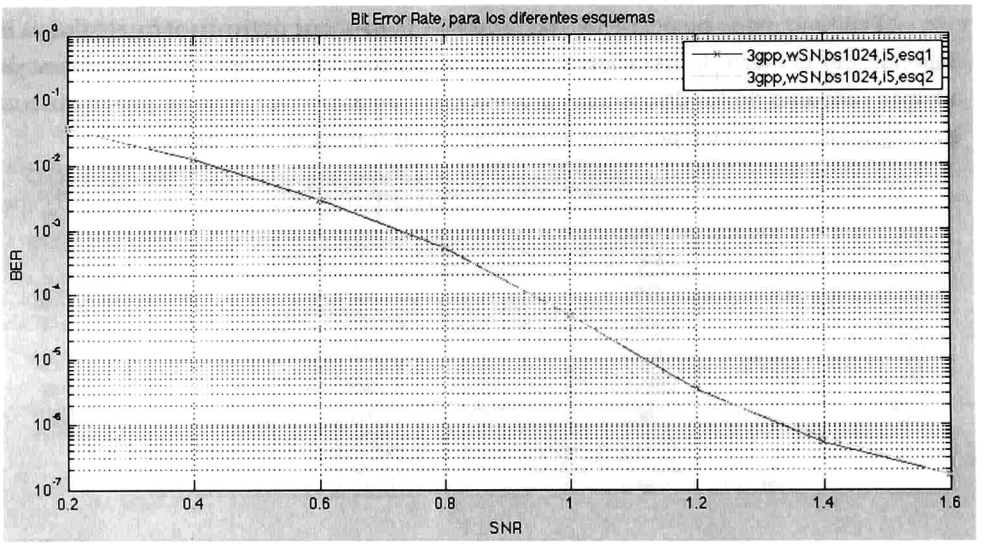


Figura 5.3. Comparación para diferentes esquemas.

5.2.3.2 Implementación en el FPGA

Esta sección se analizará el desempeño del turbo decodificador, mediante pruebas de desempeño realizadas mediante una tarjeta de desarrollo [24]. En la figura 5.4 se realiza una comparación entre lo obtenido en la simulación y lo que se implemento en la tarjeta

de desarrollo, la diferencia que presentan las curvas se debe a que las simulaciones fueron realizadas con punto flotante, es posible apreciar que la diferencia en el desempeño en base al BER de 1e-3 para 1db.

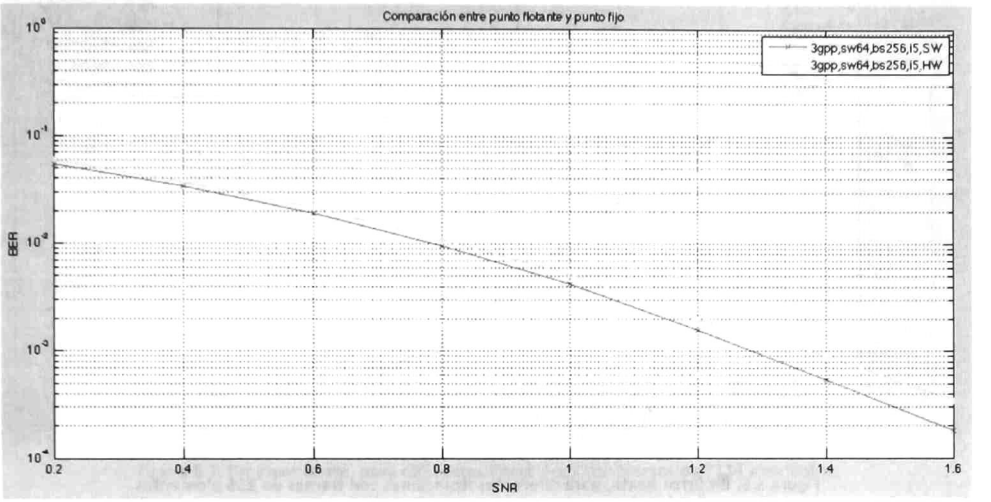


Figura 5.4. Comparación entre simulación SW e implementación HW.

En la figura 5.5 se muestra el BER para una trama de 256 símbolos y para diferentes iteraciones, se puede observar como disminuye el número de errores entre más iteraciones se realicen, también se puede observar que después de un determinado número de iteraciones, no existe una diferencia muy notoria, con lo que se comprueba lo mencionado en la literatura [12].

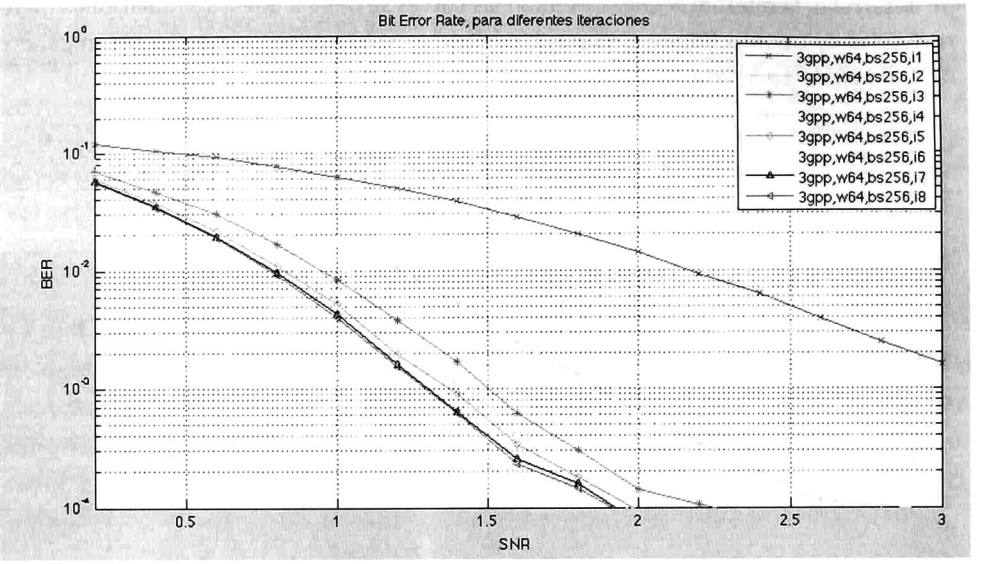


Figura 5.5. Bit Error Reate, para diferentes Iteraciones con tramas de 256 elementos.

La figura 5.6 se presenta el desempeño del turbo decodificador para un tamaño de trama de 1024 símbolos, en este caso la pendiente de las curvas es más pronunciado, esto se debe a que existe una mayor separación de elementos por parte del interleaver.

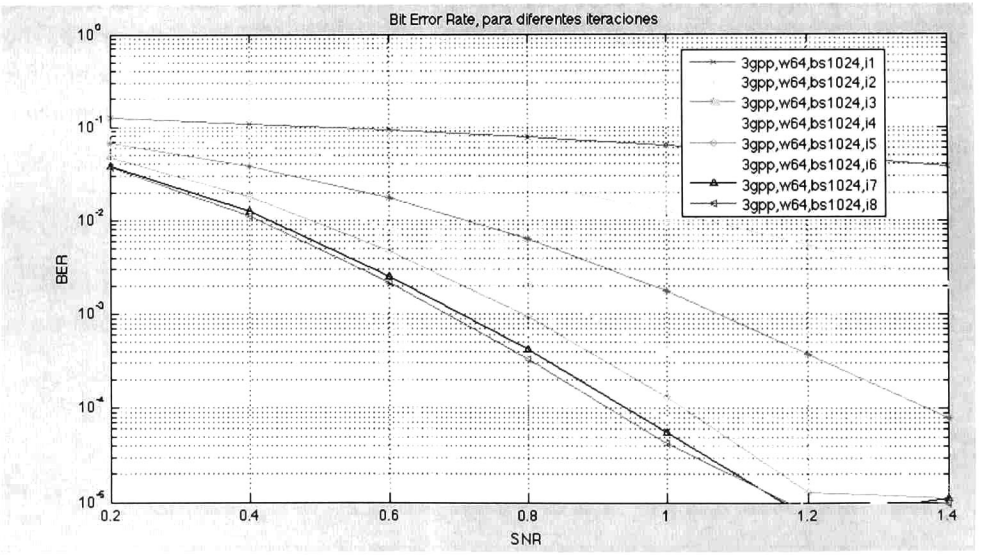


Figura 5.6. Bit Error Reate, para diferentes Iteraciones con tramas de 1024 elementos.

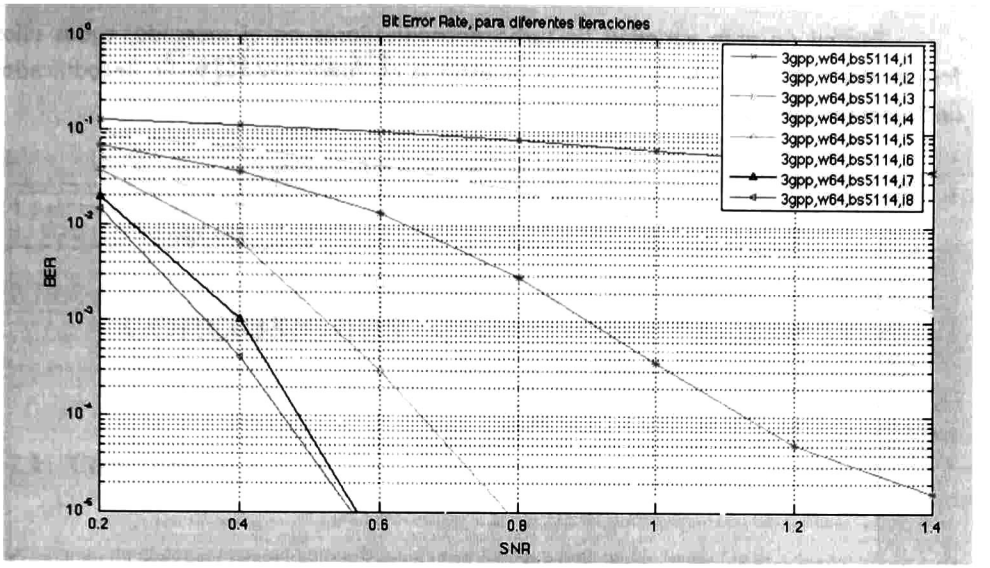


Figura 5.7. Bit Error Reate, para diferentes Iteraciones con tramas de 5114 elementos.

Por último se muestra la figura 5.7 que muestra el desempeño para el máximo tamaño de trama 5114. En las gráficas anteriormente mostradas se puede observar una gran diferencia en el desempeño que depende del tamaño de la trama. En la figura 5.8 se muestra una comparación para los tamaños de 256, 1024 y 5114. Se aprecia que entre más grande es la trama el desempeño es mayor.

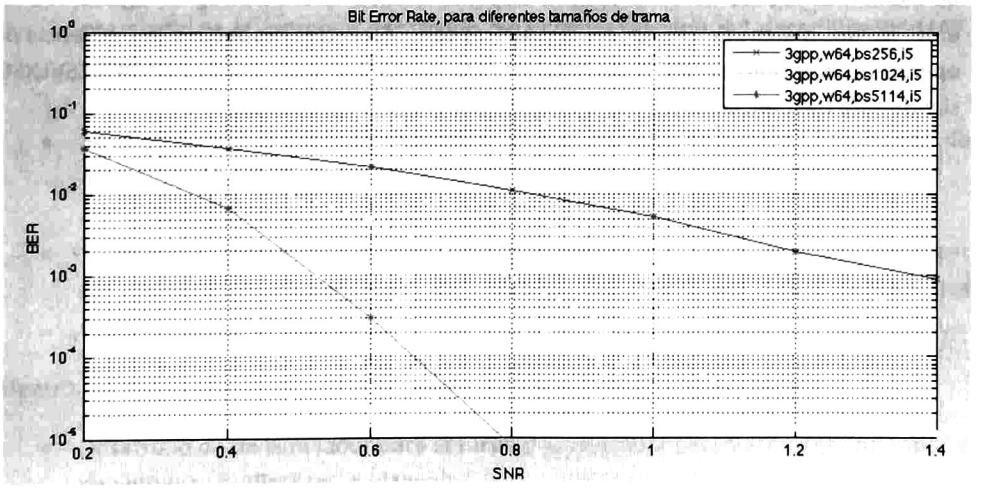


Figura 5.8. Bit Error Reate, para diferentes tamañas de trama, a cinco iteraciones.

Existen un gran variedad de turbo decodificadores en el mercado, todos ellos forman parte de IP. En la figura 5.9 se muestra la comparación del turbo decodificador implementado y el desarrollado por Xilinx [28].

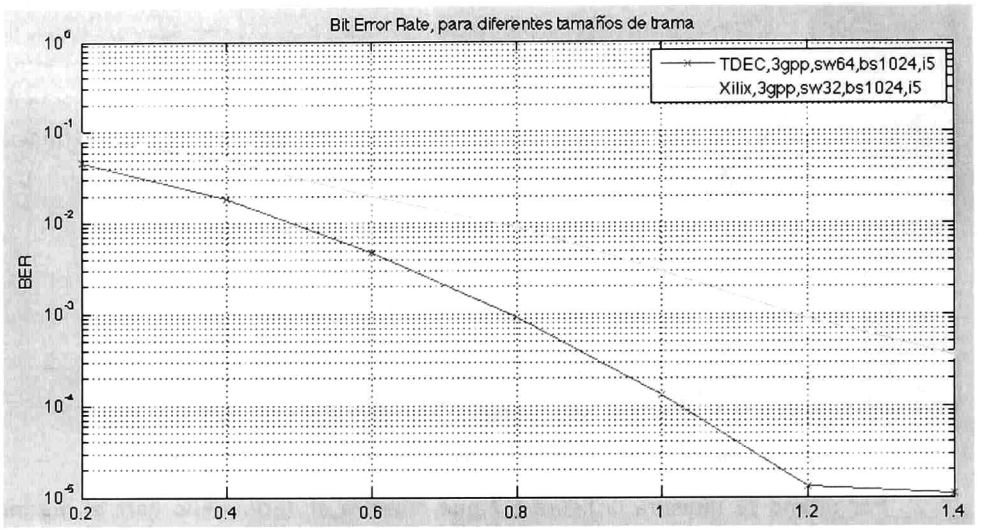


Figura 5.9. Comparación de desempeño con productos existentes.

Se puede apreciar en la figura 5.9 que el desempeño del turbo decodificador propuesto es superior al que presenta Xilinx [28]. La comparación respecto a la cantidad de elementos lógicos no se encuentra disponible, dado que los arreglos lógicos básicos entre las dos marcas de FPGAs (Xilinx y Altera) son diferentes. Respecto a la frecuencia de operación, el diseño de Xilinx permite alcanzar hasta 310 MHz para el FPGA XC5VLX50, siendo esta frecuencia mayor a la manejada por el diseño propuesto.

Capítulo VI

Conclusiones

6.1 Conclusiones

- Se presentó los conceptos fundamentan la turbo decodificación, desde las estructuras de los decodificadores MAP, el cual permitió tener un mejor entendimiento sobre el funcionamiento del algoritmo.

- Se presentaron cambios en la estructura del decodificador, que permiten eliminar un interleaver, lo cual se traduce en una gran reducción de área.

- Se propone un cambio en el cálculo de la información extrínseca, el cual permite prescindir de la memoria encargada de aliviar la latencia del decodificador MAP, con esto una vez más se reduce el área requerida para el diseño.

- Se implementó modos de operación, el cual permite aumentar la velocidad de decodificación.

- Se diseñó e implemento los métodos de verificación, que permitieron comprobar el correcto funcionamiento del diseño, así garantizar que cumple los requerimientos propuestos.

Dentro del desarrollo de esta tesis se puede remarcar los siguientes trabajos:

- Desarrollo de un simulador para el turbo decodificador parametrizable con soporte de ventanas deslizantes, en leguaje C.
- Diseño e implementación en Hardware de un turbo decodificador para el estándar 3GPP por medio del algoritmo Log-MAP, el cual nos permite disminuir su complejidad.

- Se hizo uso de ventanas deslizantes, las cuales ayudan a la optimización de memoria.
- Se agregaron funciones como:
  - Decodificación simple.
  - Decodificación doble.
  - Decodificación concatenada.
  - Soporte multi-estándar.
- Se realizó una optimización a la arquitectura la cual nos permitió prescindir de un interleaver y por ende, de una sección importante de memoria así como de los elementos lógicos relacionados con este dispositivo.
- Se realizó el proceso de verificación, con el cual se comprobó que satisface los requerimientos.
- Se desarrollaron herramientas de verificación que permitieron realizar el proceso de validación con mayor eficiencia.
- Se sintetizó y se verificó su funcionamiento con el uso de una tarjeta de desarrollo, con lo que se comprobó en forma física el funcionamiento del turbo decodificador propuesto.

6.2 Trabajo Futuro

Las mejoras que podrían realizarse son:

Integrar un bloque Interleaver/Deinterleaver al esquema del turbo decodificador, ya que este trabajo se hizo uso de una tabla de búsqueda.
Realizar un análisis de la estructura del FPGA durante su desarrollo.

- Realizar pruebas con otros fabricantes de FPGA (Xilinx)
- Optimización del área en la arquitectura.
- Implementar rutas de pipeline para aumentar la frecuencia de operación que permitan alcanzar el Throughput establecido por el estándar para todos los tamaños de trama y número de iteraciones.
- Realizar el estudio necesario para obtener la estimación de consumo de potencia.
  Llevar la arquitectura a un dispositivo ASIC.

Referencias

[1] Shannon, C. E. (1948). A Mathematical Theory of Communication. The Bell System Technical Journal , 379-423,623-656.

[2] Clark George C., J. J. (1981). Error Correction Coding for Digital Communications. New York: Plenum Publishers.

[3] Lin S. Castello D. Jr. (1983). Error control Coding Fundamentals and Applications. USA: Franklin f. Kuo.

[4] Berrou Claude, A. G. (1993). Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes. IEEE , 1064-1070.

[5] Castiñeira Moreira Jorge, P. G. (2006). Essentials of Error Control Coding. England: John Wiley & Sons, Ltd.

[6] Viterbi Andrew J., Omura Jim K. (1979). Principies of Digital Communications and Coding. USA: McGraw-Hill.

[7] Technical Specification Group Radio Access Network. (2002). 3GPP ts 25.212. Valbonne.

[8] Layer Standard for cdma2000 Spread Spectrum Systems. (2010). 3GPP2 C.S0002-E.

[9] Yllescas. C.L (2010). Análisis e implementación de un turbodecodificador para el estándar CDMA2000. Guadalajara jalisco: Centro de Investigación y de Estudios Avanzados del IPN Unidad Guadalajara.

[10] Branka Vucetic, J. Y. (2004). Turbo Codes. USA: Kluwer Academic.

[11] Hokfelt J. Edfors O. (2001) On the theory and Performance of trellis termination methos for turbo codes. IEEE Journal on selected areas in communications.

[12] Hanzo L, T. h. (2002). Turbo Coding, Turbo Equalisation and Space-Time Coding for Transmission over Wireless Channels.

[13] Bahl L. R., C. J. (1974). Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate. IEEE Transactions on infromation Theory , 284-287.

[14] Soleyman M. R. (2002). Turbo coding for satellite and wireless communications. USA: Klower Academic Publishers.

[15] Raouafi F., Dingninou A., Berrou C. (1977). Saving memory in turbo decoders using the Max-Log_MAP algorithm. IEEE.

[16] Schurgers Curt, C. F. (2001). Memory Optimization of MAP Turbo Decoder Algorithms. IEEE Transactions on very Large Scale , 305-312.

[17] Wiley, M. (2002). W-CDMA Mobile Communications System|. England: Keiji Tachikawa.

[18] Lee, W. F. (2000). VHDL Coding and Logic Synthesis whit Synopsys. San Diego, CA: Academic Press.

[19] Morales, C. A. (2009). Implementación de un turbodecodificador para sistemas de comunicación. Guadalajara: Centro de Investigación y de Estudios Avanzados del IPN Unidad Guadalajara.

[20] Boutillon Emmanuel, G. W. (2003). VLSI Architectures for the MAP Algorithm. IEEE TRANSACTIONS ON COMMUNICATIONS , 175-185.

[21] Wu Yufei, W. B. (2001). Data Width Requirements in SISO Decoding With Módulo Normalization. IEEE TRANSACTIONS ON COMMUNICATIONS , 1861-1868.

[22] Borrayo, S. H. (2010). Diseño e Implementación de un Interleaver/Deinterleaver Reconfigurable para los estándares 3GPP-WCDMA y 3GPP-LTE. Guadalajara: Centro de Investigación y Estudios Avanzados del I.P.N. Unidad Guadalajara.

[23] Sommerville, I. (2006). Ingeniería del Software. España: Pearson.

[24] Altera Corporation. (2005). Stratix II EP2S180 DSP Development Board, Reference Manual. San Jose CA.: Altera Corporation.

[25] Altera. (s.f.). Alteraforum. Recuperado el 15 de 10 de 2010, de http://www.alteraforum.com/

[26] Altera Corporation. (2008). Introduction to the Altera Nios II Soft Processor. San Jose CA: Altera Corporation.

[27] Altera Corporation. (2010). Handbook Version 10.0 Volume 1. San Jose CA.: Altera Corporation.

[28] Xilinx Logic Core. (2009). 3GPP Turbo Decoder v4.0. USA: Xilinx Inc.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.

UNIDAD GUADALAJARA

"2010, Año de la Patria, Bicentenario del Inicio de la Independencia y Centenario del Inicio de la Revolución"

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Diseño e implementación Digital del Turbodecodificador para el estándar de comunicaciones celulares 3GPP

del (la) C.

Joaquin Adrian ESPINO OROZCO

el día 04 de Febrero de 2011.

Dr. Deni Librado Torres Román
Investigador CINVESTAV 3B
CINVESTAV Unidad Guadalajara

Dr. Ramón Parra Michel
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dra. Susana Ortega Cisneros
Investigador CINVESTAV
CINVESTAV