BC-603
Don.-2011

xx(178892.1)

TK165.G8
E67
2010

CENTRO DE INVESTIGACIÓN Y
DE ESTUDIOS AVANZADOS DEL
INSTITUTO POLITÉCNICO
NACIONAL
COORDINACIÓN GENERAL DE
SERVICIOS BIBLIOGRÁFICOS

Centro de Investigación y Estudios Avanzados del
I.P.N. Unidad Guadalajara

Diseño e Implementación de un Interleaver/Deinterleaver Reconfigurable para los estándares 3GPP-WCDMA y 3GPP-LTE

Tesis que presenta:

Héctor Borrayo Sandoval

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
IPN
ADQUISICION
DE LIBROS

Guadalajara, Jalisco, agosto de 2010.

CLASIF.: TK165.G8 B67 2010
ADQUIS.: BC-603
FECHA: 14- Julio- 2011
PROCED.. Don.- 2011
$
ID. 173871-1001

Diseño e Implementación de un Interleaver/Deinterleaver Reconfigurable para los estándares 3GPP-WCDMA y 3GPP-LTE

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
Héctor Borrayo Sandoval
Ingeniero en Comunicaciones y Electrónica
Escuela Superior de Ingeniería Mecánica y Eléctrica del I.P.N. 2001-2005

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

Becario de CONACYT expediente No. 227604

CINVESTAV del I.P.N. Unidad Guadalajara, agosto de 2010

Agradecimientos:

A mis padres y hermano por su apoyo y comprensión.

A mis compañeros y amigos por su amistad.

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

Con la finalidad de desarrollar sistemas de comunicaciones más eficientes y seguros se han venido desarrollando diversas técnicas de codificación de canal, que consisten básicamente en agregar bits de redundancia a la información original y de esta manera protegerlos. Dentro de estas técnicas se encuentran los Turbo Códigos.

Los Turbo Códigos han sido tema importante de investigación durante los últimos 15 años, esto ha sido debido a su buen desempeño en la corrección de errores. Uno de los componentes clave para lograr este buen desempeño en los Turbo códigos es el par conformado por el Interleaver y el Deinterleaver, frecuentemente vistos o diseñados como coprocesadores capaces de tratar con los requerimientos de longitud variable de los datos encontrados en los estándares de comunicación más recientes.

Un Interleaver es un dispositivo que se encarga de reordenar uno a uno una secuencia de datos o bits en un orden pseudo aleatorio definido regularmente por formulas del estándar correspondiente. Asociado a un Interleaver existe el Deinterleaver, cuya función es restablecer el orden original de la secuencia de datos o bits.

En este trabajo se presenta una arquitectura de un Interleaver/Deinterleaver (I/D) reconfigurable para los estándares "3GPP-WCDMA" y "3GPP-LTE" El diseño de este I/D toma ventaja de la idea del cálculo de la operación módulo de forma iterativa. La arquitectura que aquí se presenta basa el diseño de algunos de sus bloques en trabajos de investigación previos; Sin embargo, se incluyen mejoras y bloques adicionales que permiten que nuestro I/D no solo sea capaz de generar todas las direcciones de entrelazado y de-entrelazado establecidas por los estándares 3GPP-WCDMA y 3GPP-LTE, sino que además se pueda presentar el diseño del I/D como un coprocesador stand-alone.

Abstract

In order to obtain more reliable and efficient communications systems, new channel coding techniques have been continuously developed. Channel coding consists basically in adding redundancy bits to the original information. One of these channel coding techniques are the Turbo codes.

During the last 15 years Turbo codes have become an important topic in communications researches taking an increasing importance in channel coding due to its good performance in error correction.

One key component in Turbo codes is the Interleaver/Deinterleaver (I/D) pair of blocks, often designed as a reconfigurable coprocessor able to deal with requirements of large data length variability found in the newest communications standards.

An Interleaver is a device that rearranges the order of a sequence of symbols or bits in a one to one pseudo random order defined by some equations usually given in its correspondent standard. Associated with any Interleaver exist a Deinterleaver, which is the device that restores the data or bit sequence into its original order.

In this work we introduce a reconfigurable I/D architecture for the turbo coder and turbo decoder used in the 3GPP-WCDMA and 3GPP-LTE standards. This architecture is designed under the idea of iterative module computation and some modules are based in previous works.

By enhancing some modules we obtained an architecture capable to handle all the data-length configurations and not only generate interleaving addresses required by the standards, but also with the capability of dealing with the flow of data stream through the I/D. The presented design operates as a stand-alone-system.

Tabla de Contenido

Tabla de contenido

Tabla de contenido

Índice de Figuras

Índice de Figuras

Índice de Tablas

Lista de acrónimos

Acrónimo	Significado
AGU	Del inglés *Address Generator Unit,* unidad generadora de direcciones.
ASIC	Del inglés *Application Specifica Integrated Circuit,* circuito integrado de aplicación especifica.
AWGN	Del inglés *Additive White Gaussian Noise,* Ruido aditivo blanco gaussiano.
BEC	Del inglés *Backward Error Correction,* Corrección de errores hacia atrás.
BER	Del inglés *Bit Error Rate,* Tasa de bits erróneos.
DUT	Del inglés *Devise Under Test,* dispositivo bajo pruebas.
FEC	Del inglés *Forward Error Correction,* Corrección de errores hacia adelante.
FIFO	Del inglés *First Input First Output,* primero en entrar, primero en salir.
FPGA	Del inglés *Field Programable Gates Array,* arreglo de compuertas programables.
GSM	Del inglés *Global System Mobile,* sistema global para la comunicaciones móviles.
HDL	Del inglés *Hardware Description Language,* lenguaje de descripción de hardware.
IP	Del inglés *Intelectual Property,* propiedad intelectual.
ITU	Del inglés *International Telecommunications Union,* union internacional de telecomunicaciones.
I/D	Del inglés *Interleaver/Deinterleaver,* entrelazador/deentrelazador.
LTE	Del inglés *Long Term Evolution,* que se refiere a un Nuevo estándar de la norma 3GPP.
LUT	Del inglés *Look Up Table,* Tabla de busqueda.

MIMO	Del inglés *Multiple Inputs Multiple Outputs,* multiples entradas – múltiples salidas.
MMA	Del inglés *Modular Multiplication Algorithm,* algoritmo de multiplicación modular.
NCBPS	Del inglés *Number of Coded Bits per Subcarrier,* número de bits codificados por subportadora.
OFDMA	Del inglés *Orthogonal Frecuency Division Multiple Access,* acceso múltiple por división de frecuencias ortogonales.
RAM	Del inglés *Random Access Memory,* memoria de acceso aleatorio.
ROM	Del inglés *Read Only Memory,* memoria de solo lectura.
RSC	Del inglés *Recursive Systematic Convolutional,* Convolutivo sistemático recursivo.
SC-FDMA	Del inglés *Single Carrier Frecuency Division Multiple Access,* acceso múltiple por división de frecuencia de una sola portadora.
SISO	Del inglés *Soft Input -Soft Output,* entrada suave- salida suave.
SNR	Del inglés *Signal to Noise Ratio,* Relación señal a ruido.
TB	Del inglés *Test Bench,* cama de pruebas.
TDMA	Del inglés *Time Division Multiple Access,* acceso múltiple por división de tiempo.
W-CDMA	Del inglés *Wideband Code Division Multiple Access,* Acceso múltiple por división de código de banda ancha.
WiMAX	Del inglés *Worldwide Interoperability for Microwave Access,* Interoperabilidad mundial para acceso por microondas.

Capítulo 1

Introducción

1.1 Motivación

En la actualidad existen una gran cantidad de sistemas de comunicaciones tanto *alámbricos*[1] como inalámbricos, el desarrollo científico y tecnológico que hoy existe ha permitido lograr grandes avances en los equipos y técnicas usadas para el mejoramiento de las comunicaciones. Particularmente las comunicaciones inalámbricas han venido teniendo gran auge y han surgido nuevos y mejores estándares de comunicaciones considerados de tercera y cuarta generación como WiMAX (que viene del inglés *Worldwide Interoperability for Microwave Access*), W-CDMA (que viene del inglés *Wideband Code Division Multiple Access*), LTE (que viene del inglés *Long Term Evolution*), entre otros. Muchos de estos nuevos estándares utilizan los llamados "Turbo Códigos" [1] para la codificación de canal debido a su buen desempeño el cual está estrechamente relacionado con la decodificación en forma iterativa y el uso del Interleaver.

En el país existen pocos trabajos de desarrollo de implementación en hardware sobre los Turbo Códigos, los cuales tienen un extenso campo de investigación y aplicaciones. Aunque existen fabricantes que cuentan con módulos usados en Turbo Códigos como por ejemplo, Altera con su *"Turbo Encoder/Decoder MegaCore Function"*, implementado para el estándar europeo W-CDMA (3GPP TS 25.212) y Xilinx con *"LogicCORE"*, un turbo codificador y decodificador para el estándar americano CDMA2000 (TIA/EIA 2002.2) [2], estos son considerados como propiedad intelectual (IP, del inglés *Intelectual Property*) del fabricante y no se pueden utilizar libremente.

Otra motivación para el trabajo aquí presentado es que a la par de la proliferación de nuevos y diversos estándares de comunicación ha surgido la necesidad de diseñar dispositivos multi-estándar sin que esto implique un aumento excesivo de hardware. En la actualidad ya existen en los mercados dispositivos de comunicación inalámbricos capaces de utilizar más de un estándar de comunicación permitiendo una mayor interoperabilidad con otros equipos y otras redes de comunicación, sin embargo aún falta mucho por hacer en este campo.

[1] Se utilizará el término alámbrico a lo largo del documento para referirse a alambrados.

1.2 Antecedentes

Los sistemas de comunicaciones han existido desde hace miles de años de distintas maneras. La forma más básica de un sistema de comunicaciones, y cuyos componentes podemos encontrar en cualquier sistema de comunicaciones de los más sencillos a los más complejos, es aquella que cuenta con un emisor, un receptor y un medio para la propagación del mensaje. La información que se transmite entre el emisor y el receptor debe adaptarse al medio de transmisión, esto implica la necesidad de disponer de un soporte adecuado a través del cual pueda viajar la información.

Actualmente existen diversos tipos de sistemas de comunicación:

- Internet
- Sistemas de comunicación satelital
- Redes alámbricas e inalámbricas
- Redes de telefonía fija
- Redes de telefonía celular

Estos sistemas de comunicación utilizan básicamente dos tipos de medios de propagación, lo que permite hablar de 2 tipos de sistemas de comunicación.

1. **Comunicación alámbrica**: el soporte usado aquí son cables y la información se transmite en forma de impulsos eléctricos.
2. **Comunicación inalámbrica**: en este caso el soporte material por el que viaja la información es el propio espacio, y concretamente en la atmosfera terrestre, el aire. La información se transmite en forma de ondas de radio.

En la actualidad los sistemas de comunicación inalámbrica han tomado especial relevancia y han crecido de manera vertiginosa ocupando cada vez una mayor parte del espectro radioeléctrico utilizable. Las pérdidas de información debido a interferencias o degradación de las señales que llevan dicha información se hacen más evidentes cuando la comunicación es de tipo inalámbrico. Además de las interferencias externas y la degradación de la señal que lleva la información, también existe el "ruido" causado por los componentes electrónicos de los sistemas de comunicación como los amplificadores o las resistencias.

Actualmente es imposible eliminar por completo el ruido en los sistemas de comunicación, debido principalmente a la imperfección de los componentes, sin embargo se busca tener una relación señal a ruido (SNR, del inglés Signal to Noise Ratio) suficientemente alta para que el efecto del ruido en los sistemas de comunicación permita un desempeño suficiente para operar satisfactoriamente. Para lograr tener sistemas de comunicación seguros, con mayores velocidades de transmisión, eficiente y con mejor aprovechamiento del ancho de banda se han venido desarrollando diversas técnicas que nos permitan proteger la información que se pretende

transmitir, logrando así tener sistemas de comunicación con mayor inmunidad al ruido. Una de estas técnicas son las usadas para la codificación de canal.

En 1948 Claude E. Shannon sentó las bases teóricas para la capacidad máxima (*C*) de un canal de comunicaciones limitado en banda (*B*) con ruido aditivo blanco gaussiano (AWGN, del inglés *Additive White Gaussian Noise*) [3] que responde a la ecuación (1.1).

$$C = B\log_2(1+SNR)bps \tag{1.1}$$

A partir de que Shannon estableció este límite teórico muchas investigaciones fueron hechas con la finalidad de desarrollar técnicas capaces de acercarse lo más posible al límite teórico de la capacidad de canal.

Dentro de este marco fueron desarrollados algunos esquemas de codificación de canal cuyo propósito es lograr transmisiones de datos cercanas a la capacidad de canal teórica establecida por Shannon, con probabilidades de error tan bajas como sea posible. Con el uso de la codificación de canal se ha logrado mayores velocidades de transmisión con tasas de bits erróneos (BER, del inglés *Bit Error Rate*) bajas y utilizando cada vez menor potencia.

En la

Figura 1.1 se muestra un modelo de un sistema de comunicaciones digital donde se muestra en que parte del sistema se sitúa la codificación de canal.

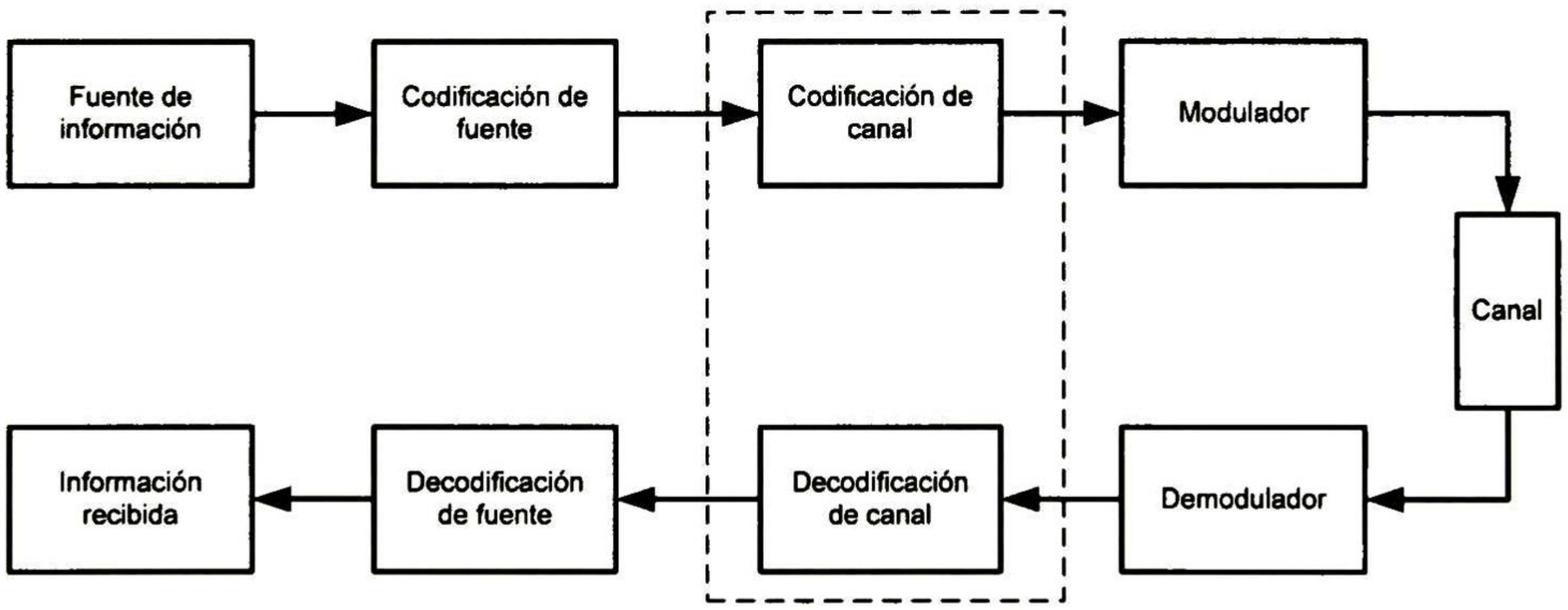

Figura 1.1. Modelo de un sistema de comunicaciones digital.

La codificación de canal consiste básicamente en agregar bits de redundancia de manera inteligente a la información original que se pretende transmitir, estos bits de redundancia son aprovechados por el receptor para minimizar los errores causados por el canal, aumentando de esta manera la confiabilidad de las comunicaciones.

Dentro de las técnicas de codificación de canal se encuentran las de corrección de errores hacia adelante (FEC, del inglés *Forward Error Correction*) y las de corrección de errores hacia atrás (BEC, del inglés *Backward Error Correction*).

El mecanismo BEC se usa cuando el receptor detecta un error en los datos recibidos y solicita nuevamente al transmisor que le envíe los datos otra vez. Este tipo de corrección es eficiente cuando los errores son poco frecuentes y el ancho de banda es limitado. Dentro de este grupo se encuentran los algoritmos que utilizan bits de paridad y los que utilizan la redundancia cíclica.

El mecanismo FEC es aquel que permite una corrección de errores en el receptor sin necesidad de retransmisión de la información original. Esta técnica es usada en sistemas de tiempo real donde no se puede esperar a la retransmisión para mostrar los datos; Dentro de las técnicas FEC se usan los tipos de codificación de bloque, convolutivos y concatenados, y es a estos últimos que pertenecen los llamados Turbo Códigos.

Los Turbo Códigos fueron introducidos por Claude Berrou y Alain Glavieux en 1993 [4] y desde entonces se han venido consolidando como uno de los descubrimientos más importantes en comunicaciones para la codificación de canal, el gran éxito de los Turbo Códigos se debe a que con ellos se han obtenido resultados muy cercanos al límite teórico para la capacidad de canal establecido por Shannon.

El buen desempeño mostrado por los Turbo Códigos está relacionado principalmente con el uso de un algoritmo iterativo en la decodificación y con el uso de interleavers.

En la Figura 1.2 se muestra el esquema general de un Turbo Codificador, asi como los bloques que lo componen que son dos codificadores convolutivos sistemáticos recursivos (RSC, del inglés *Recursive Systematic Convolutional*) en paralelo, separados por un Interleaver y un bloque que multiplexa y "perfora" las secuencias codificadas para formar la salida serial del Turbo Codificador.

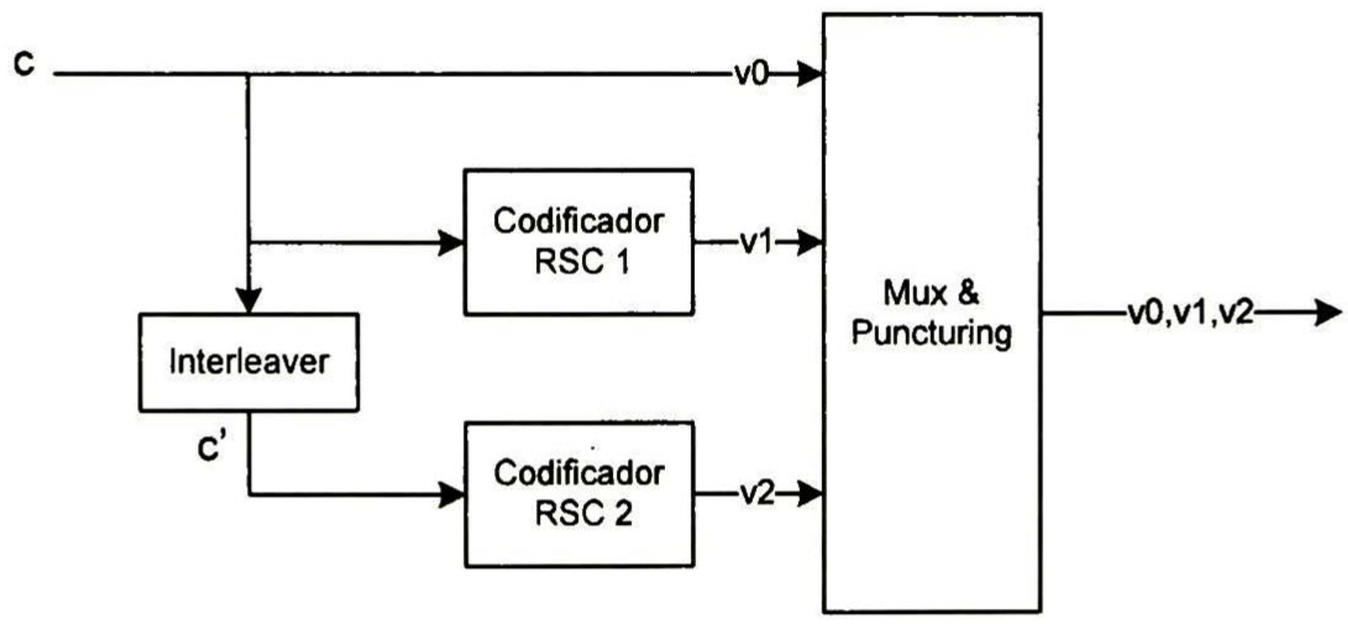

Figura 1.2. Esquema general de un Turbo Codificador, tomado de [2].

En la Figura 1.3 se muestra el esquema general de un Turbo Decodificador y los bloque que lo componen que son un par de decodificadores SISO (*Soft Input Soft Output*), dos instancias de un Interleaver y dos instancias de un Deinterleaver.

El presente trabajo de tesis se centra en el diseño y desarrollo del Interleaver/Deinterleaver (I/D) necesario en los Turbo Codificadores y Turbo Decodificadores para los estándares 3GPP-W CDMA y 3GPP-LTE. El Interleaver es un dispositivo que se encarga de reordenar uno a uno una secuencia de datos o bits en un orden pseudo aleatorio definido por alguna ecuación o forma dada por el estándar al que pertenezca. El dispositivo encargado de hacer el proceso inverso es llamado Deinterleaver, y se encarga de reordenar la secuencia de datos o bits previamente entrelazada a su orden original.

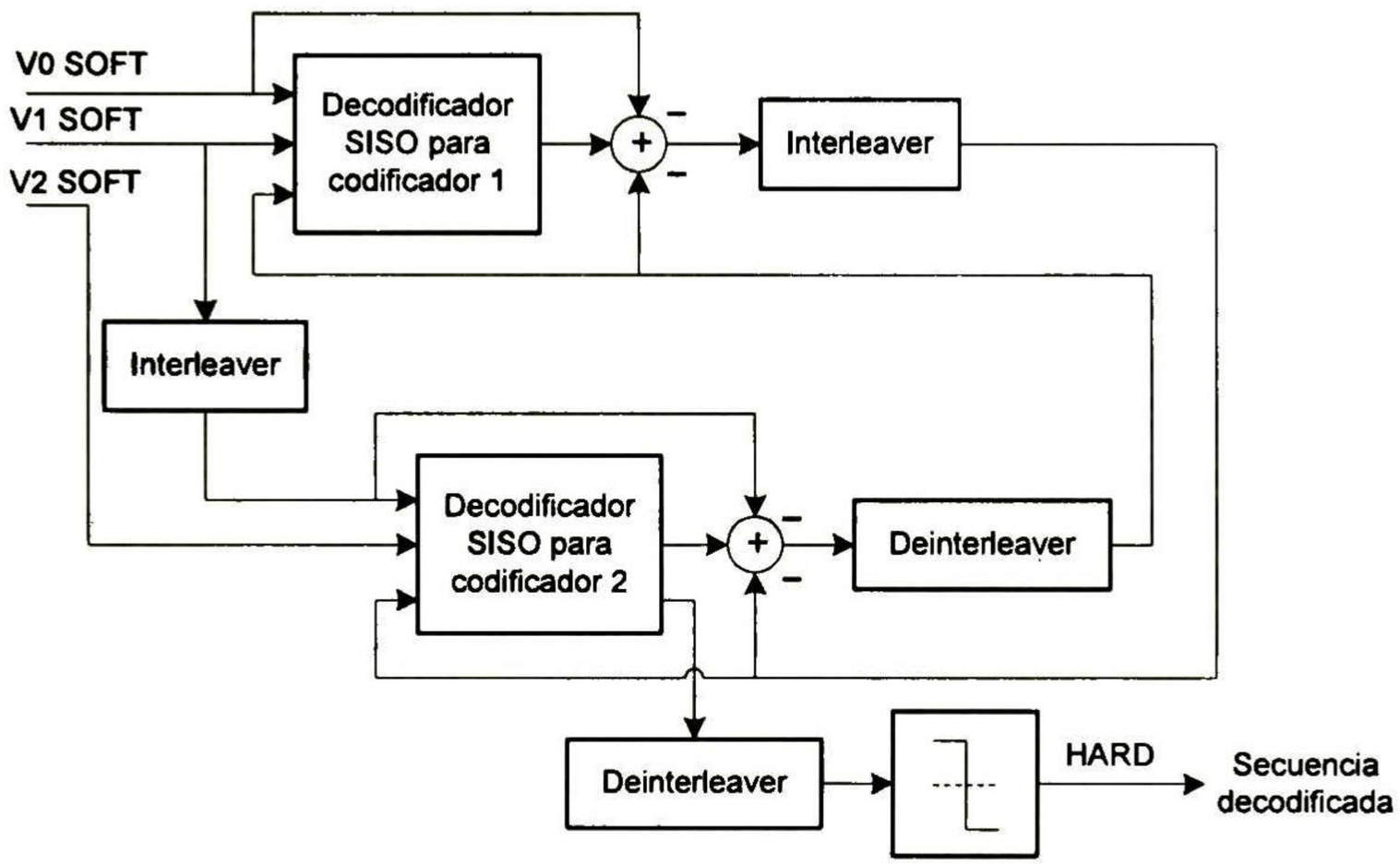

Figura 1.3. Esquema general de un Turbo Decodificador, tomado de [2].

1.3 Estado del arte

Debido a que en gran medida el éxito de los Turbo Códigos se debe al uso de interleavers apropiados, varias investigaciones en distintas partes del mundo se han venido enfocando en el diseño de estos bloques.

En la actualidad existen varios tipos de interleavers que dependiendo del estándar que los rige es la forma en que trabajan y sus características de tamaño, potencia y velocidad. La creciente necesidad de contar con dispositivos multi-estándar ha obligado a los investigadores a pensar también en interleavers multi-estándar, algunas de las implementaciones más conocidas de este tipo son:

- Interleavers basados en "Look Up Tables" (LUT).
- Interleavers con matrices especiales.
- Interleavers basados en mini procesadores.
- Interleavers reconfigurables.

Cada uno de estas aproximaciones tiene sus ventajas y desventajas, mismas que se tratarán con detenimiento posteriormente en el presente trabajo.

Existen buenos trabajos de investigaciones recientes sobre interleavers para uno más estándares que podemos utilizar como referencias, algunos de estos trabajos son los presentados en [5],[6] y [7].

1.4 Planteamiento del problema

Actualmente los Turbo Códigos se utilizan para la codificación de canal en varios estándares por separado, sin embargo cada vez es más necesario contar con turbo codificadores y turbo decodificadores multi-estándar; para lograr este objetivo es necesario que algunos de los bloques que componen dichos turbo codificadores/decodificadores sean también multi-estándar, uno de estos bloques es el I/D y es en este en el que el presente trabajo está enfocado.

Existen en el mercado gran cantidad de interleavers que funcionan bien para el estándar que fueron diseñados, también existen algunos interleavers multi-estándar, sin embargo estos últimos presentan algunos problemas, como que utilizan gran cantidad de hardware, consumen mucha potencia, no manejan flujo de datos o simplemente que no los hay para los estándares deseados.

En este trabajo en particular se trata el diseño de un I/D reconfigurable para los estándares 3GPP-WCDMA y 3GPP-LTE, que además sea capaz de manejar el flujo de datos. Actualmente sabemos de la existencia de algunas arquitecturas sobre interleavers reconfigurables para dichos estándares, sin embargo dichos diseños no manejan el flujo de datos y no son encontrados en la literatura como coprocesador.

1.5 Objetivos

Los principales objetivos del presente trabajo de tesis son:

- Obtener una arquitectura de I/D reconfigurable para los estándares 3GPP-WCDMA y 3GPP-LTE capaz de ser utilizada como un coprocesador.
- Incorporar el manejo de flujo de datos a nuestro diseño.
- Validar el diseño del I/D.
- Hacer un análisis de consumo de área y potencia.

- Realizar la implementación en FPGA del I/D.

Al final del presente trabajo se pretende tener el I/D como un bloque funcional capaz de ser utilizado en turbo codificadores y turbo decodificadores de los estándares 3GPP-W CDMA y 3GPP-LTE.

1.6 Estructura del trabajo

El presente trabajo de tesis está dividido en seis capítulos, el nombre de cada capítulo así como una breve descripción de cada uno de ellos se muestra a continuación.

En el capítulo 1 se presenta un breve antecedente teórico y se sitúa en contexto la importancia del I/D dentro de los sistemas de comunicaciones. En este capítulo también se mencionan los objetivos de nuestra tesis, así como la motivación de nuestro trabajo, antecedentes y justificación del mismo.

En el capítulo 2 se hace una detallada explicación de lo que es un Interleaver y un Deinterleaver, las aplicaciones que se les da y los tipos de interleavers más comunes que existen. Luego de explicar el concepto del Interleaver se analizan los estándares 3GPP-WCDMA y 3GPP-LTE, que son la base para hacer el diseño de nuestro I/D.

En el capítulo 3 se presentan algunas arquitecturas de trabajos previos de interleavers similares, se mencionan las técnicas en que se basan y se muestran las posibles ventajas y desventajas en comparación con nuestro diseño. Después de haber presentado dichas arquitecturas de distintos Interleavers, se presenta nuestro diseño, mostrando cada uno de sus bloques con su respectiva arquitectura y una explicación de la misma.

En el capítulo 4 se hace una descripción del diseño de nuestro I/D, se presenta cada módulo del diseño final como un módulo funcional y se establece una relación entre las entradas y las salidas de cada bloque.

En el capítulo 5 se muestran los resultados obtenidos. Se hacen algunas comparaciones entre los resultados arrojados por nuestro diseño contra valores obtenidos con herramientas como MATLAB y ModelSim para definir si los resultados de nuestro diseño eran los esperados. Se presentan los resultados de la síntesis del diseño en HDL así como algunas tablas comparativas del desempeño de nuestro diseño contra algunos otros diseños similares.

En el capítulo 6 se muestran las principales conclusiones a las que se llega a partir de los resultados que se obtienen a del diseño final; Además se menciona el uso que se le podrá dar al módulo obtenido y finalmente se enumeran las posibles mejoras que se le pueden hacer como parte de un trabajo futuro.

Capítulo 1. Introducción

Capítulo 2

Interleaver

2.1 El entrelazado y el Interleaver

El entrelazado es una simple pero potente técnica que puede ser utilizada para ayudar a los códigos de corrección de errores aleatorios a trabajar también en corrección de errores en ráfaga.

Los errores aleatorios ocurren en un canal cuando bits individuales del mensaje a transmitir se corrompen por el ruido, estos pueden ser caracterizados como bits erróneos aislados del mensaje. Los errores aleatorios pueden ser causados por el ruido térmico en los canales de comunicación.

Los errores en ráfaga pueden ser causados por el desvanecimiento en el canal de comunicación o por defectos mecánicos en el sistema de almacenamiento. Este tipo de errores afectan a una serie de bits adyacentes de una señal y pueden llegar a ser difíciles de corregir para algunos códigos, sin embargo los códigos de bloque pueden tratar los errores en ráfaga de forma efectiva. La habilidad de un código de bloque para corregir una ráfaga de errores depende del número de errores en la señal.

El entrelazado es realizado por los llamados Interleavers, los cuales son dispositivos que reordenan uno a uno el orden de una secuencia de símbolos o bits de una manera determinista no continua. Asociado con cualquier Interleaver siempre existe un Deinterleaver, el cual es un dispositivo que se encarga de restablecer la secuencia intercalada a su orden original [1]. Una vez que un bloque de datos ha sido procesado por algún Interleaver, a estos datos procesados se les conoce como entrelazados. En la Figura 2.1 se muestra un ejemplo del efecto de un Interleaver y un Deinterleaver sobre un bloque de datos, donde en A) se muestran los datos en su orden original antes de pasar por el Interleaver, luego en B) se muestran los datos entrelazados después de pasar por el Interleaver y finalmente en C) se muestran los datos con su orden original restablecido luego de pasar por el Deinterleaver.

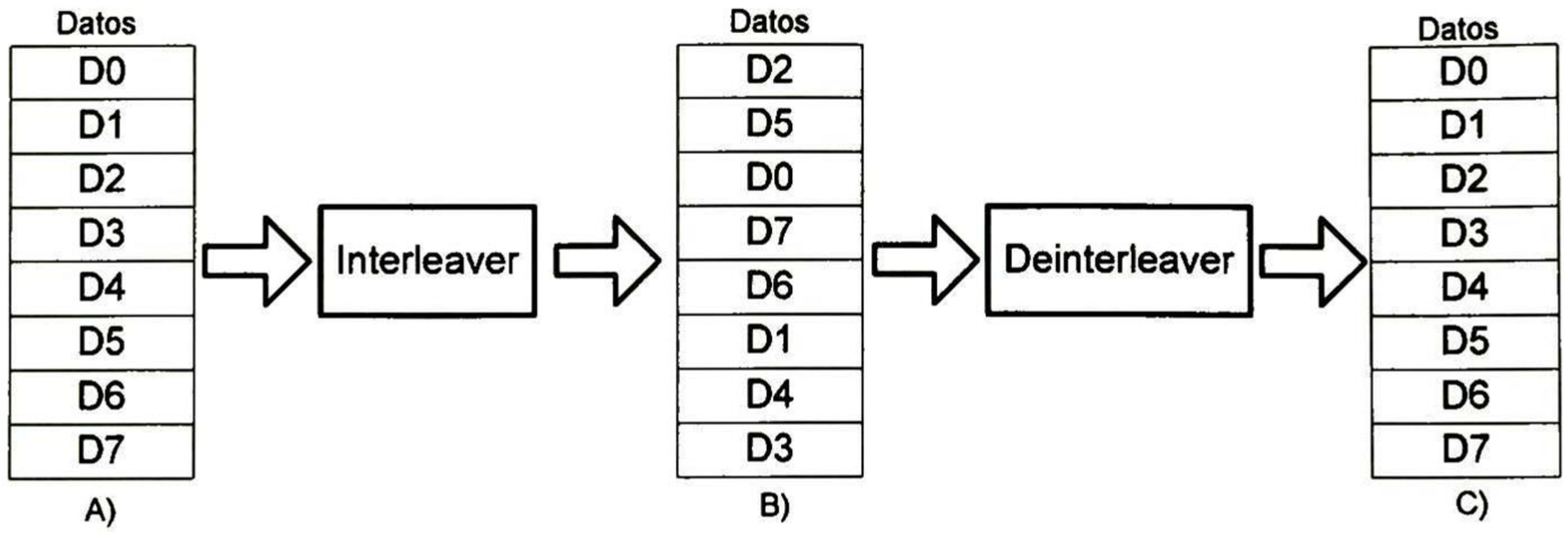

Figura 2.1. Efecto del Interleaver/Deinterleaver sobre un bloque de datos.

De los diferentes grupos de interleavers existentes se pueden identificar dos principales grupos, los de bloque y los convolutivos.

2.1.1 Interleaver de bloque

Un Interleaver de bloque acepta un grupo de símbolos y los reordena sin repetir u omitir alguno de los símbolos del grupo. El número de símbolos por cada grupo es fijo para cada Interleaver.

Un Interleaver de bloque opera sobre un solo bloque a la vez y no existe entrelazado entre bloques [8]. Un tipo común de estos interleavers son los que forman una matriz de *i* filas y *n* columnas, donde se escriben los datos fila por fila y se leen columna por columna. Estos interleavers son conocidos como interleavers de bloque (*n,i*).

El de-entrelazado es simplemente el proceso inverso, se escriben los datos en la matriz columna por columna y se leen renglón por renglón.

2.1.2 Interleaver convolutivo

El modo de trabajo que describe a un Interleaver convolutivo consiste en introducir y sacar datos de forma conmutada y sincronizada de un grupo de registros de corrimiento, cada uno con un retardo fijo. Basándonos en la Figura 2.2 podríamos considerar cada fila como una serie de registros FIFO (del inglés, *First In First Out*). En estos interleavers cada nuevo símbolo del vector de entrada alimentaria a un registro de corrimiento, desplazando al símbolo que tenía más tiempo en la fila de registros hacia el vector de salida. En este tipo de interleavers los registros son limpiados en su estado inicial, por lo que la secuencia de salida además de tener los bits procesados, incluirá algunos ceros, los cuales son fácilmente detectables gracias al tamaño fijo de los registros y la sincronización de lectura y escritura. También se dice que este tipo de interleavers tienen memoria ya que los datos que van saliendo pueden haber estado almacenados por varios ciclos.

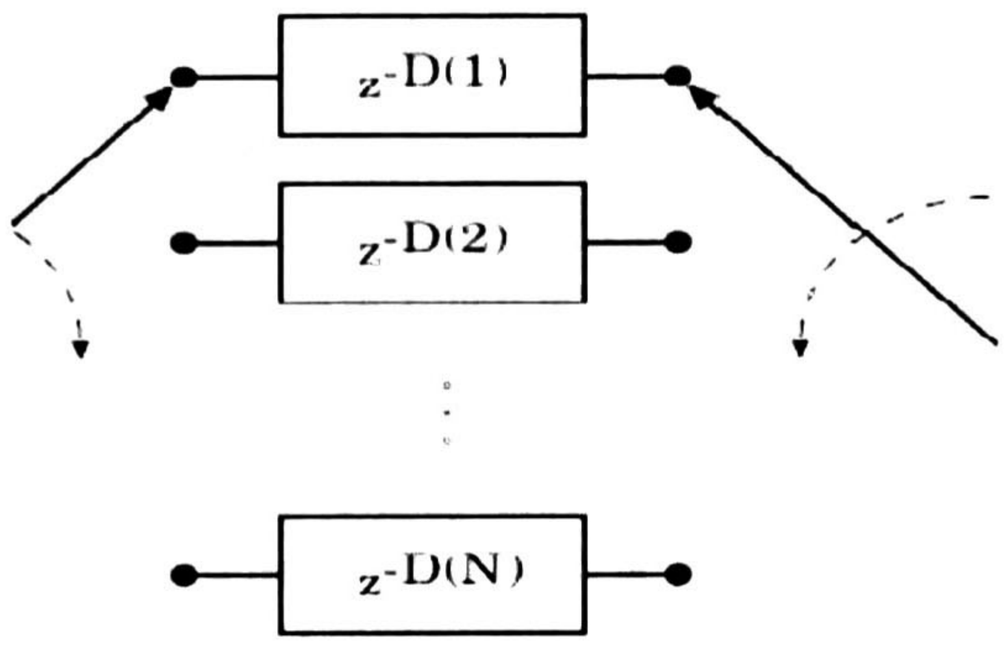

Figura 2.2. Diagrama general de un Interleaver Convolutivo, tomado de [9].

En la Figura 2.2 se muestra el diagrama general de un Interleaver convolutivo, donde se representan un grupo de registros de corrimiento así como sus respectivos retardos D(1), D(2),...D(N), donde el K-ésimo registro retiene K símbolos.

2.2 El Interleaver y su aplicación

La razón por la que los interleavers son tan útiles en la corrección de errores, es que son capaces de dispersar una secuencia de símbolos o bits, logrando que los errores de ráfaga de varios bits se conviertan en errores aleatorios de un bit. El orden o patrón en que esta dispersión se hace puede variar dependiendo de la aplicación que se le dará, o del estándar que lo rige. Muchos de los interleavers existentes tienen una trayectoria de entrelazado muy sencilla como por ejemplo escribir los símbolos en alguna matriz rectangular renglón por renglón y leerlos columna por columna, sin embargo hay otros con trayectorias de entrelazado no tan sencillas cuyo orden depende de formulas complejas.

Aunque el principal uso que se les da a los interleavers es para corrección de errores, son utilizados en varias áreas como en computación, criptografía y comunicaciones.

En los sistemas de comunicaciones el entrelazado se aplica en la parte de codificación de canal, que es la encargada de la corrección de errores. El bloque "Interleaver" puede situarse en el lado del transmisor entre el codificador de canal y el transmisor. Para recuperar la información original, del lado del receptor se usa un Deinterleaver situado entre el receptor y el decodificador, tal y como se muestra en la Figura 2.3.

Para mostrar más claramente lo mencionado anteriormente podemos ver el ejemplo de la Figura 2.3, tomado de [6] donde en a) tenemos la información original en bits que queremos

transmitir (B0-B15), después en b) esta información se codifica mediante un código de repetición donde cada bit es representado por tres bits iguales, después de esto podemos entrelazar los bits obteniendo la secuencia mostrada en c), donde cada uno de los bits que representan la información original (B0-B15) están lo suficientemente separados, esta nueva secuencia de bits entrelazados está lista para ser transmitida. Al transmitir la información codificada y entrelazada por un canal donde el ruido se presenta en forma de ráfagas de duración corta (cada ráfaga afecta máximo a 3 bits consecutivos en este ejemplo), esta se verá afectada por dicho ruido en secciones como se muestra en d). En el lado del receptor tendremos la secuencia de datos transmitida afectada por el ruido, posteriormente al pasar esta secuencia a través de un Deinterleaver obtendremos la secuencia original de los bits pero con los efectos del ruido e), al decodificar esta secuencia obtenemos los bits originales en f) con probabilidades de error muy bajas.

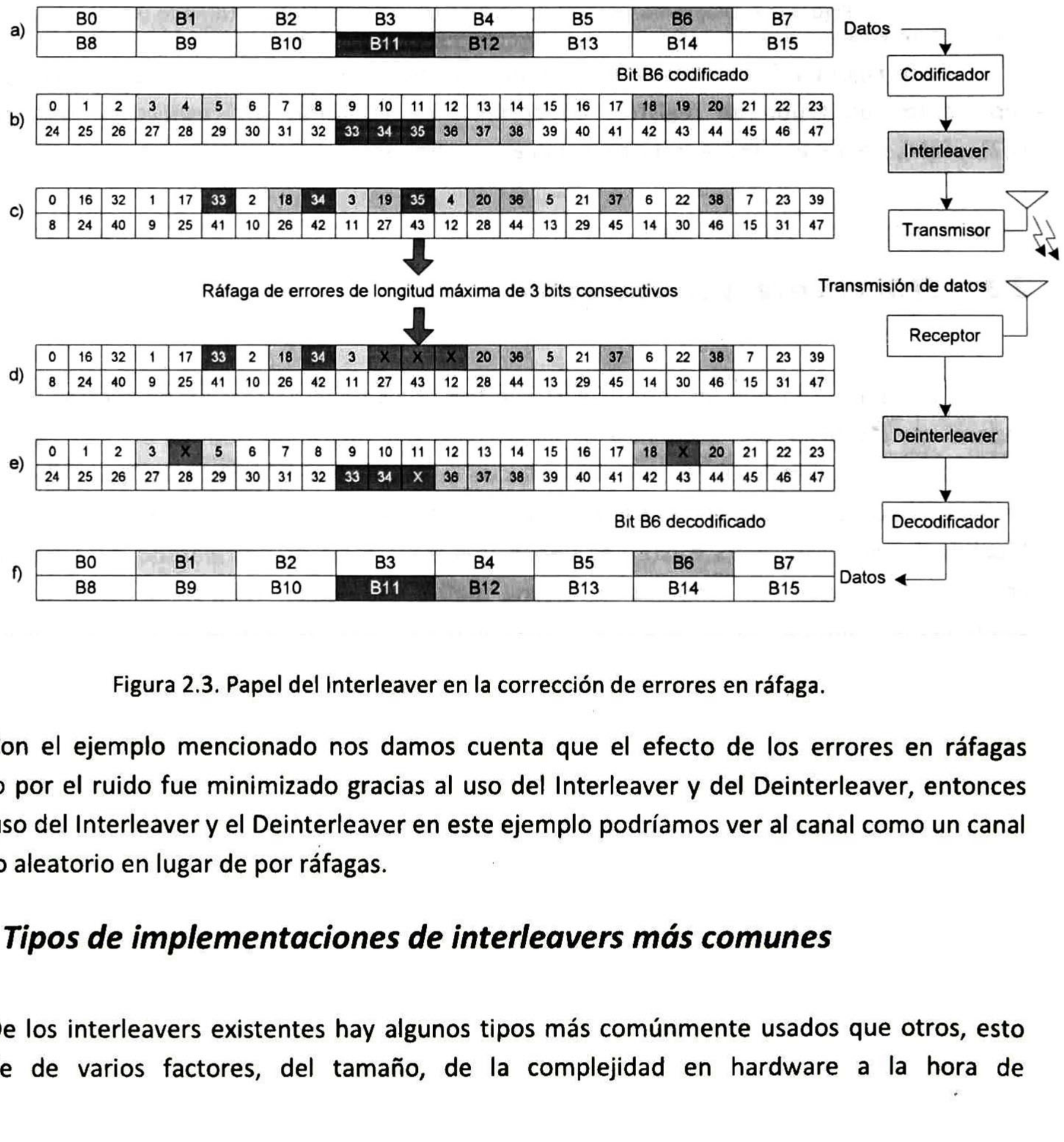

Figura 2.3. Papel del Interleaver en la corrección de errores en ráfaga.

Con el ejemplo mencionado nos damos cuenta que el efecto de los errores en ráfagas causado por el ruido fue minimizado gracias al uso del Interleaver y del Deinterleaver, entonces con el uso del Interleaver y el Deinterleaver en este ejemplo podríamos ver al canal como un canal de ruido aleatorio en lugar de por ráfagas.

2.3 Tipos de implementaciones de interleavers más comunes

De los interleavers existentes hay algunos tipos más comúnmente usados que otros, esto depende de varios factores, del tamaño, de la complejidad en hardware a la hora de

implementación, del estándar en que se usará, de los requerimientos de velocidad y consumo de potencia, etc.

Entre los interleavers más comúnmente implementados podemos mencionar:

- Implementaciones con ROM o LUT.
- Implementaciones con Matrices Especiales.
- Implementaciones con Barras de Interconexiones.
- Implementaciones Aritméticas.

2.3.1 Implementación de Interleaver con Look Up Tables (LUT)

Los interleavers implementados con LUTs son los más sencillos de implementar. Las direcciones de escritura para el entrelazado son guardadas en ROMs, estas direcciones son elegidas de acuerdo al estándar para el que se requiera el proceso de entrelazado. En este tipo de implementación las direcciones se leen de la ROM o LUT en forma secuencial por un contador, la dirección leída de la ROM será la localidad de memoria donde se escribirán los datos de forma entrelazada.

En la Figura 2.4 se muestra de forma muy general la arquitectura de un Interleaver de bloque basado en LUT para la trayectoria de entrelazado y una memoria para almacenar los datos entrelazados [8]. Mediante esta arquitectura podemos hacer entrelazado y de-entrelazado según se requiera.

Para realizar un entrelazado de datos, estos se escriben de de acuerdo a la trayectoria de entrelazado que se encuentra en la ROM y se leen de manera secuencial desde la memoria RAM.

Si queremos llevar a cabo un de-entrelazado tendremos que escribir los datos en la memoria de manera secuencial y posteriormente leer de la memoria de salida según la secuencia de entrelazada almacenada en la ROM.

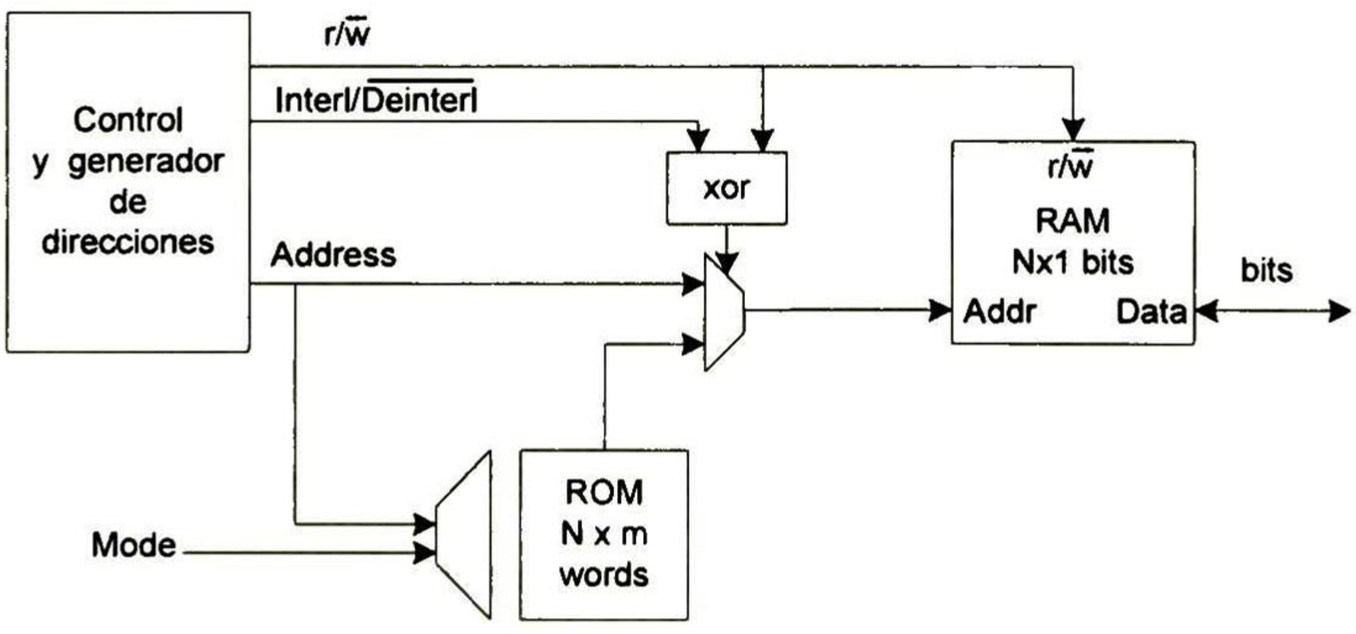

Figura 2.4. Arquitectura básica de un Interleaver basado en Look Up Table

Este tipo de interleavers son muy útiles cuando la cantidad de datos a entrelazar es constante y el patrón de entrelazado permanece fijo.

La principal ventaja de estos interleavers es que su implementación es de lo más sencilla y es muy general siempre que el tamaño de bloque a entrelazar permanezca fijo.

Las principales desventajas son que no se puede escribir y leer al mismo tiempo, sino que es necesario esperar a terminar la escritura para empezar a leer, y además en caso de ser necesario operar con varios estándares sería obligatorio almacenar todas las trayectorias en LUTs ocupando grandes cantidades de memoria y haciéndolo de impráctico debido al tamaño. Por ejemplo, si quisiéramos utilizar este tipo de Interleaver en el estándar 3GPP-LTE tan solo para un tamaño de bloque K=6144, requeriríamos de una memoria ROM de casi 80Kbits y aún quedarían otras 187 trayectorias de entrelazado por cubrir para este estándar, por lo que sería totalmente impráctico utilizar este tipo de Interleaver.

2.3.2 Implementación de interleavers mediante matrices especiales

Este tipo de implementación viene de la idea de que un Interleaver puede ser visto como una matriz especial donde los símbolos son escritos en la matriz renglón a renglón y leídos columna a columna. El ejemplo de la Figura 2.5 muestra el orden de escritura y lectura de los datos para este tipo de interleavers.

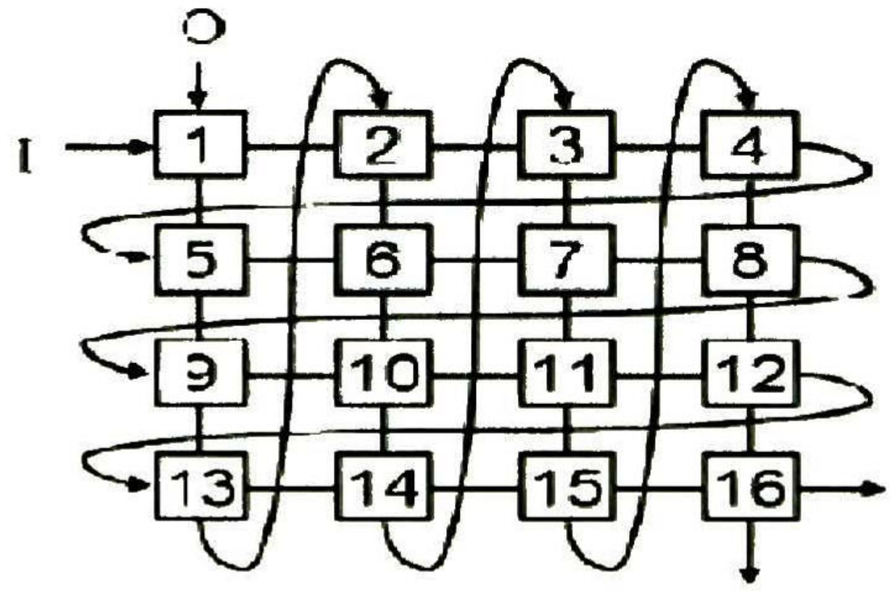

Figura 2.5. Ejemplo de orden de escritura y lectura de entrelazado en una matriz especial.

La función de permutación clásica para este tipo de interleavers está dada por la ecuación (1.2), donde "*m*" es el número de columnas de la matriz y "*n*" es el número de renglones de la misma.

$$i = m \times (k \bmod n) + floor(k/n), \quad k = 0,1,\ldots,m \times (n-1) \qquad (1.2)$$

Donde "*i*" es el índice del orden de lectura de la matriz iniciando desde cero de izquierda a derecha y de arriba hacia abajo.

Este tipo de Interleaver se podría ver como un tipo de buffer normal y llevar a cabo el entrelazado sin costo extra por instrucciones y control. El número de columnas y filas deben poder ser configurables, y soportar múltiples permutaciones entre filas y renglones. Este tipo de interleavers no son tan generales como los de LUT pero el costo por hacerlos soportar estándares adicionales puede llegar a ser menor que el costo de incrementar el tamaño de ROMs en los interleavers tradicionales.

En la Figura 2.6 se muestra la idea principal de este tipo de implementación. Esta arquitectura está basada en un bloque de memoria de una matriz especial donde los símbolos se escriben en forma de renglón y se leen en forma de columnas.

Una fila completa puede ser escrita en un ciclo de reloj y una columna completa puede ser leída también en un ciclo de reloj. Las permutaciones entre renglones se llevan a cabo antes de que los bits sean almacenados en memoria solo con reordenar los bits en el bus de entrada. De la misma manera la permutación entre columnas se lleva a cabo mediante el reordenamiento de bits en el bus de salida de datos, después de que los datos han sido leídos. Si el número de esquemas de permutaciones es pequeño, entonces los bloques de permutación podrán desarrollarse mediante un pequeño grupo de multiplexores controlados por las mismas señales de control.

La operación de de-entrelazado es casi idéntica a la de entrelazado, en este tipo de Interleaver el de-entrelazado es visto como otro esquema de entrelazado donde el numero de fila es visto como numero de columna y las permutaciones entre filas es visto como permutaciones entre columnas y viceversa.

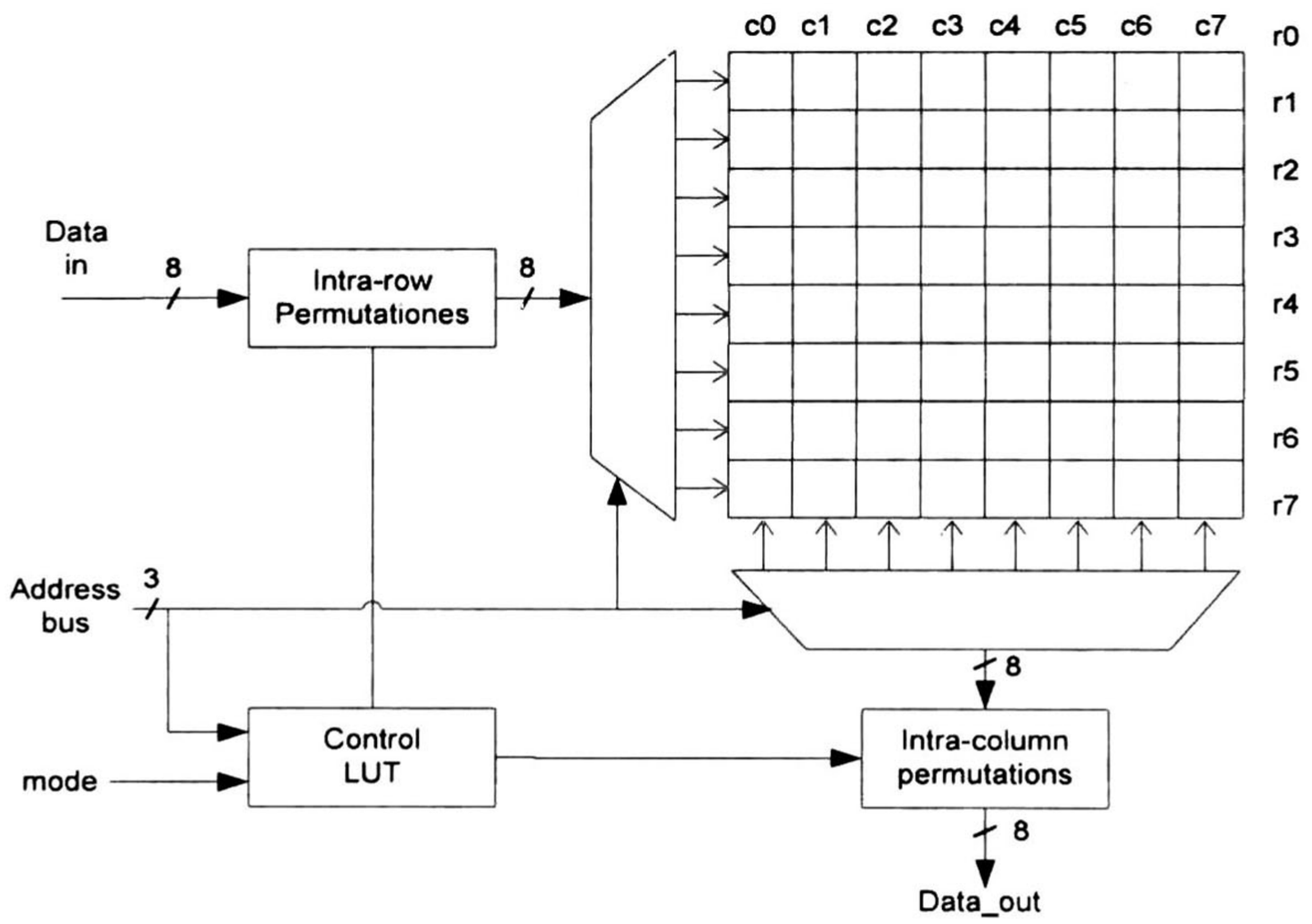

Figura 2.6. Arquitectura de un Interleaver basado en matriz especial, tomado de [8].

La matriz es dimensionada de acuerdo con el mayor número de columnas y renglones que se tiene que soportar. Si se utiliza un esquema de entrelazado con tamaño de bloque menor, las columnas y renglones extra pueden ser deshabilitados para ahorrar energía.

Este Interleaver se puede configurar a través de 3 señales de control especificando el número de filas, columnas y el esquema de permutaciones a utilizar. Podría requerirse un bloque de control para decidir el tipo de permutaciones de acuerdo al modo con que se esté trabajando, este se puede implementar con una pequeña LUT.

2.3.3 Implementación de Interleaver usando barra de interconexiones

Utilizando una barra de interconexiones de X entradas con X salidas, podemos interconectar cada entrada con cada salida dependiendo de la configuración que elijamos. Este comportamiento puede ser visto como un Interleaver.

La barra de interconexiones utiliza parámetros de configuración para seleccionar las trayectorias que se harán, esto con la ayuda de una Unidad Generadora de Control (UGC). Las señales de la UGC dependen del número de entradas. En una barra de interconexiones cada entrada se conecta a cada salida a través de un switch llamado punto de interconexión. Cada punto de interconexión es controlado por la UGC, utilizando este tipo de implementación podemos hacer una permutación de X entradas a X salidas en un solo ciclo de reloj.

Una barra de interconexiones de X entradas y X salidas puede lograrse utilizando X^2 switches por canal, un ejemplo de barra de interconexiones se muestra en la Figura 2.7 donde se muestra en a) una barra de interconexiones genérica programable de X conexiones de entrada y X conexiones de salida, en b) se muestra una barra de interconexiones de 6 entradas y 6 salidas con puntos de conexión ya programados.

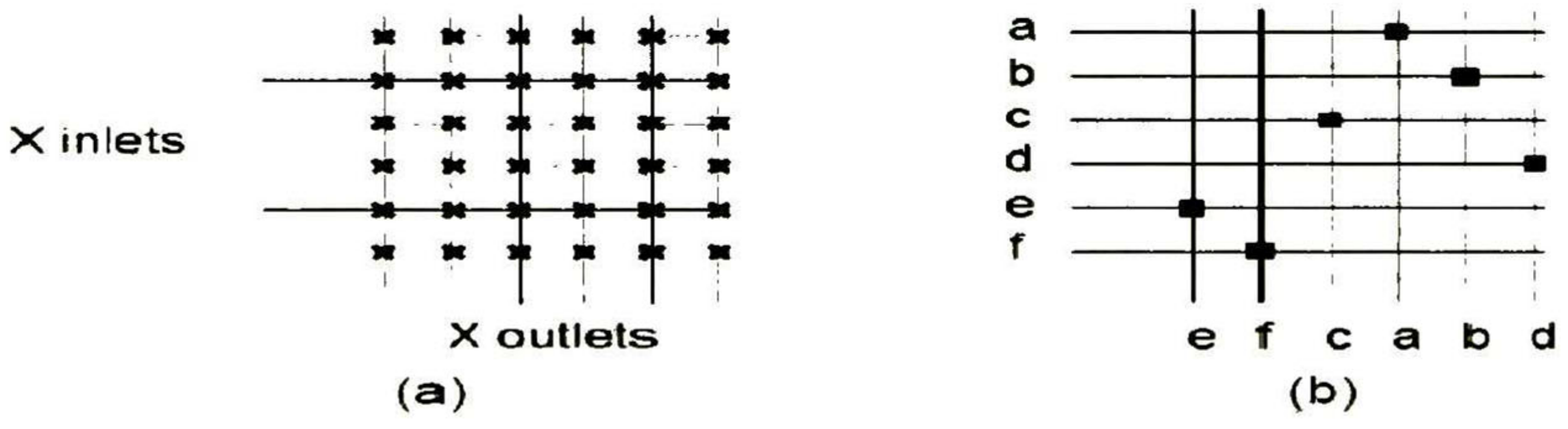

Figura 2.7. Ejemplo de una barra de interconexiones.

En la actualidad existen diversas técnicas para diseñar barras de interconexión de gran tamaño que nos ayudan a minimizar la carga capacitiva o el número de switches utilizados, algunas de estas técnicas se pueden encontrar en [6].

2.3.4 Interleaver aritmético

La implementación aritmética obtiene su nombre de la manera en que la memoria obtiene la secuencia de direcciones para almacenar los datos de forma entrelazada, las direcciones de escritura tienen la información del orden de permutación y son generadas a través de una Unidad Generadora de direcciones (AGU, del inglés *Address Generator Unit*).

En la Figura 2.8 se muestra la arquitectura básica en la que se basa este tipo de Interleaver, donde se muestran los bloques principales que son: la AGU, el contador (*Counter*) y la memoria RAM (*Memory*).

La AGU cuenta con unidades de multiplicación, suma y división para poder realizar apropiadamente las operaciones necesarias para generar las direcciones de escritura.

Por ejemplo, si nos basáramos en la ecuación (1.3) para entrelazado del estándar 802.11ª [10], donde "*k*" denota el índice del bit antes de la primera permutación, "*i*" denota el índice del bit después de la primera permutación pero antes de la segunda y NCBPS es el número de bits codificados por sub portadora, necesitaríamos un contador para "*k*", 3 divisores por 16, un sumador y un multiplicador.

$$i = (NCBPS/16)(k \bmod 16) + floor(k/16), \qquad k = 0,1,...,NCBPS-1 \tag{1.3}$$

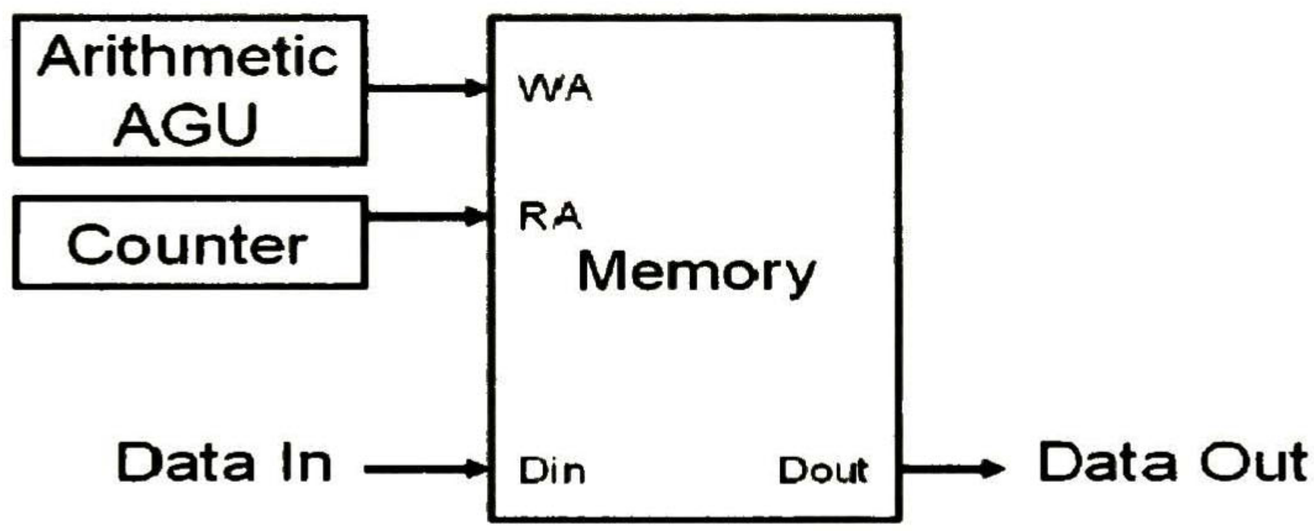

Figura 2.8. Arquitectura de un Interleaver aritmético básico, tomado de [6].

En la Figura 2.9 se muestra el diagrama a bloques de la implementación de la ecuación (1.3).

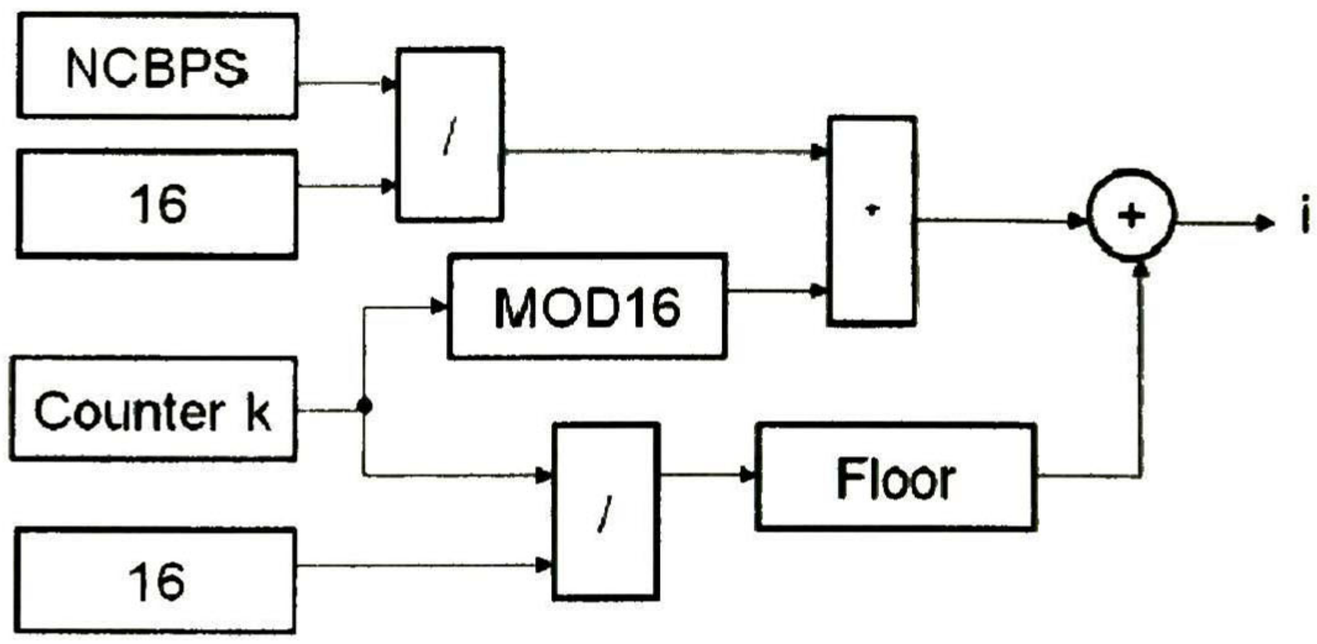

Figura 2.9. Diagrama a bloques que representa la ecuación (1.3).

Para implementar la ecuación (1.4), donde *NCBPS* es el número de bits codificados por subportadora y "*S*= max (*NCBPS*/2,1)", necesitaríamos 3 divisores por "*S*", por "i" y por *NCBPS*, 3 sumadores y un multiplicador.

$$j = S \times floor(i/S) + (i + NCBPS - floor(16 \times i / NCBPS)) \bmod S \qquad (1.4)$$

i=0,1,...,*NCBPS*-1.

En la Figura 2.10 se muestra el diagrama a bloques para la implementación de dicha ecuación.

Entonces una AGU para el Interleaver del estándar 802.11a se lograría uniendo las arquitecturas de la Figura 2.9 y de la Figura 2.10, de donde obtendríamos algo similar a lo que se muestra en la Figura 2.11.

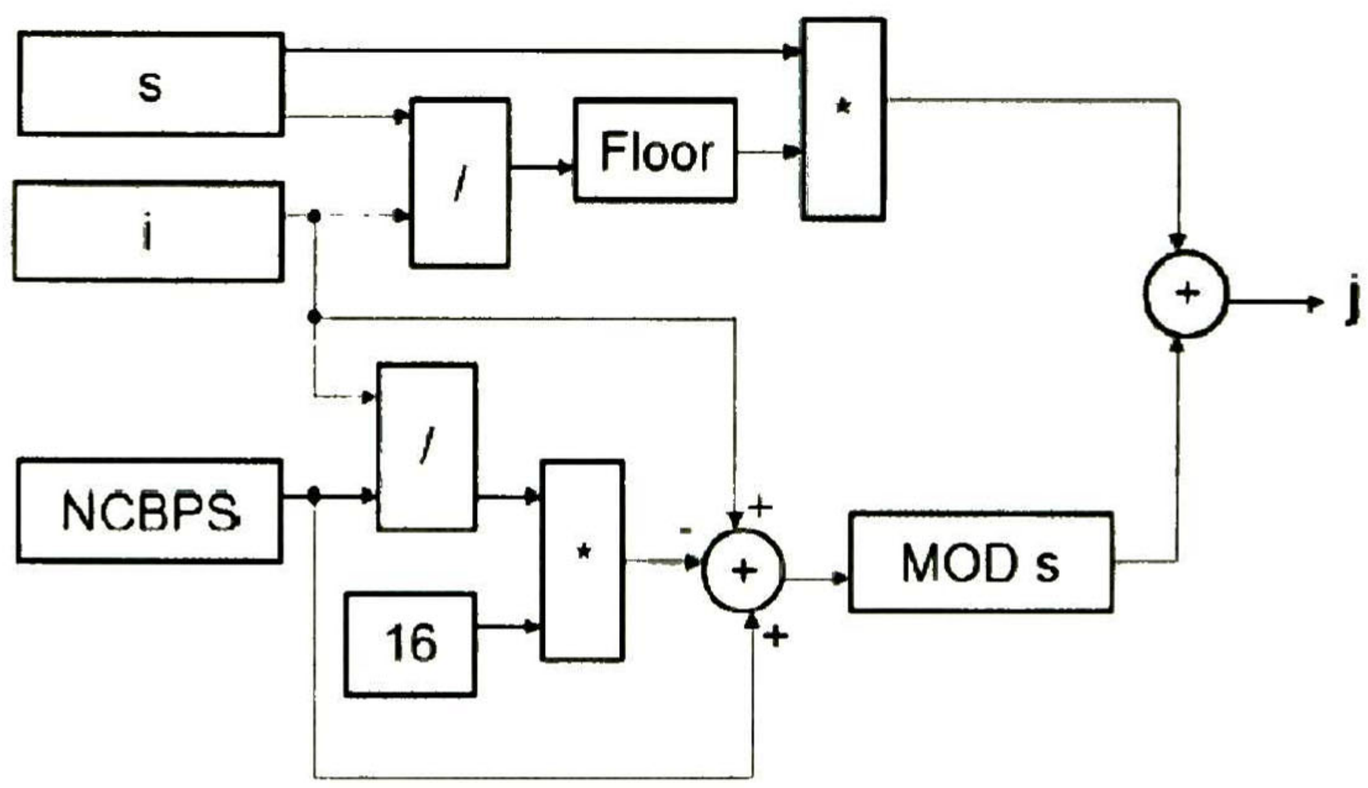

Figura 2.10.Diagrama a bloques que representa la ecuación (1.4).

Esta arquitectura podría ser implementada utilizando un microprocesador o con el uso directo de divisores, multiplicadores y sumadores.

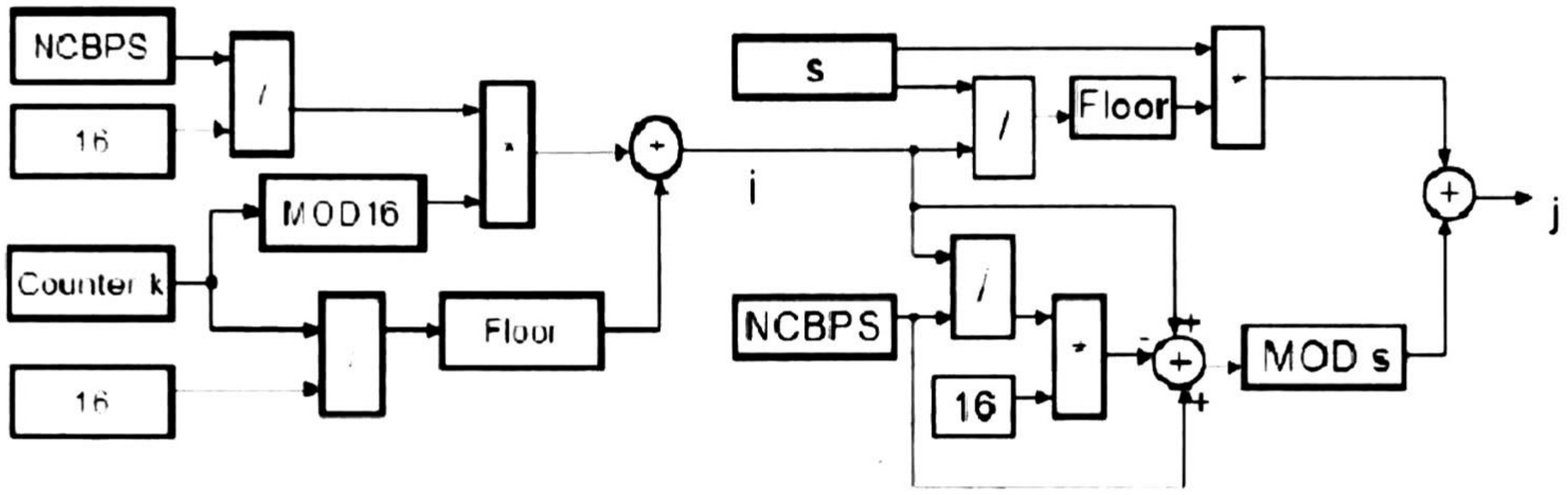

Figura 2.11. Diagrama a bloques de la UAG para el estándar 802.11a.

2.4 Estándares 3GPP-WCDMA y 3GPP-LTE

Como ya mencionamos previamente, el tipo de interleavers que se utiliza en cada aplicación depende de las necesidades en general. La trayectoria de entrelazado generada por los interleavers usados en los estándares 3GPP-WCDMA [11] y 3GPP-LTE [12] está definida por permutaciones definidas en dichos estándares. A continuación se presentan las ecuaciones en las que se basan las permutaciones de estos interleavers de bloque, junto con una descripción de los parámetros involucrados.

2.4.1 3GPP – W CDMA

W- CDMA, o en español "Acceso Múltiple por División de Código de Banda Ancha", es una tecnología usada en comunicaciones móviles inalámbricas de tercera generación que aumenta las tasas de transmisión de datos de los sistemas GSM utilizando la interfaz aérea CDMA en lugar de TDMA (del inglés, *Time Division Multiple Access*). W-CDMA soporta tasas de transmisión que van de 144 hasta 512 Kbps para áreas de cobertura amplias y éstas pueden llegar hasta los 2Mbps para mayor cobertura en áreas locales. W-CDMA es considerado como la evolución de los sistemas GSM, por lo que existe una interoperabilidad con plataformas de comunicación anteriores.

Esta tecnología propuesta por el grupo 3GPP dio lugar a un estándar [11], en el cual podemos encontrar las especificaciones sobre las permutaciones que debe realizar el Interleaver que se utiliza en dicho estándar.

El Interleaver interno usado por los Turbo Códigos en este estándar consiste de un grupo de bits alimentando una matriz con relleno, permutaciones entre los renglones (Inter-Row) de la matriz, permutaciones entre los elementos de cada renglón (Intra-Row) de la matriz, y sacar los

bits de la matriz rectangular removiendo los bits inválidos. Los bits que alimentan al Interleaver interno del turbo codificador son denotados como $x_1,x_2,x_3,...,x_k$, donde K es el número de bits y toma su valor de entre $40 \leq K \leq 5114$.

Los símbolos utilizados en este algoritmo son:

- K — Número de bits que entran al Interleaver interno del turbo codificador
- R — Número de renglones de la matriz rectangular
- C — Número de columnas de la matriz rectangular
- p — Número primo
- v — Raíz primitiva
- i — Índice del número de fila de la matriz rectangular
- j — Índice del número de columna de la matriz rectangular
- q_i — Vector de mínimos enteros primos
- r_i — Vector de enteros primos permutados
- k — Índice de la secuencia de bits
- $(s(j))_{j \in \{0,1,...,p-2\}}$ — Secuencia base para la permutación "intra-row"
- $(T(i))_{i \in \{0,1,...,R-1\}}$ — Trayectoria de permutación entre renglones
- $(U_i(j))_{j \in \{0,1,...,C-1\}}$ — Trayectoria de permutación dentro del i-esimo renglón

2.4.1.1 Introducción de los bits de entrada a la matriz rectangular con relleno

La secuencia de bits de entrada $x_1,x_2,x_3,...,x_k$ se escribe en la matriz rectangular del Interleaver interno del turbo codificador de la siguiente manera:

1.- Determinar el número de renglones de la matriz rectangular, R, tal que:

$$R = \begin{cases} 5, & if\,(40 \leq K \leq 159) \\ 10, & if\,((160 \leq K \leq 200)\ or\ (481 \leq K \leq 530)) \\ 20, & if\,(K = any\ other\ value) \end{cases}$$

Los renglones de la matriz rectangular están enumerados 0,1,..., R-1 de arriba hacia abajo.

2.- Determine el número primo, p, que será usado en la "intra-permutación " y el numero de columnas de la matriz rectangular, C, tal que:

if $(481 \le K \le 530)$ *then*

$p = 53$, *and* $C = p$

else

Encuentre el número primo mínimo "p" de la Tabla 2.1 tal que:

$K \le R \times (p+1)$,

Y determine C, tal que:

$$C = \begin{cases} p-1, & if\,(K \le R \times (p-1)) \\ p, & if\,(R \times (p-1) < K \le (R \times p)) \\ p+1, & if\,((R \times p) < K) \end{cases}$$

end if

Las columnas de la matriz rectangular están enumeradas 0,1,..., C-1 de izquierda a derecha

Tabla 2.1. Lista de números primos "p" y su raíz primitiva asociada "v" para el estándar 3GPP-WCDMA.

p	v	P	v	P	v	p	v	p	v
7	3	47	5	101	2	157	5	223	3
11	2	53	2	103	5	163	2	227	2
13	2	59	2	107	2	167	5	229	6
17	3	61	2	109	6	173	2	233	3
19	2	67	2	113	3	179	2	239	7
23	5	71	7	127	3	181	2	241	7
29	2	73	5	131	2	191	19	251	6
31	3	79	3	137	3	193	5	257	3
37	2	83	2	139	2	197	2		
41	6	89	3	149	2	199	3		
43	3	97	5	151	6	211	2		

3.- Escribir la secuencia de bits de entrada $x_1, x_2, x_3, \ldots, x_k$ dentro de la matriz rectangular R x C renglón por renglón empezando con el bit y1 en la columna 0 y renglón 0:

$$\begin{bmatrix} y_1 & y_2 & y_3 & \cdots y_C \\ y_{(C+1)} & y_{(C+2)} & y_{(C+3)} & \cdots y_{2C} \\ \vdots & \vdots & \vdots & \vdots \\ y_{((R-1)C+1)} & y_{((R-1)C+2)} & y_{((R-1)C+3)} & \cdots y_{(R \times C)} \end{bmatrix}$$

Donde $y_k=x_k$ para k=1,2,...,k, y si R x C > k , los bits extra son rellenados tal que y_k= 0 ó 1 para k=K+1,K+2,...,R x C. Estos bits extra de relleno son retirados de la matriz rectangular después de hacer las permutaciones entre renglones y dentro de cada renglón.

2.4.1.2 Permutaciones intra-row e inter-row

Después de llenar la matriz rectangular R x C con los bits de entrada, lo siguiente es hacer las permutaciones intra-row e inter-row paso a paso según se muestra en los puntos 1- 6:

1.- Seleccionar una raíz primitiva "v" de la Tabla 2.1 y que se encuentra a la derecha del número primo "p".

2.- Construir la secuencia base $(s(j))_{j \in \{0,1,...,p-2\}}$ para la permutación "intra-row" tal que:

$$s(j) = (v \times s(j-1)) \bmod p, \quad j = 1,2,...,(p-2), \quad and \; s(0) = 1$$

3.- Asignar q_0=1 para que sea el primer número primo de la secuencia $(q_i)_{i \in \{0,1,...,R-1\}}$, y determine los siguientes enteros primos de la secuencia q_i, tal que:

$$m.c.d.(q_i, p-1) = 1, \quad q_i > 6, \quad and \quad q_i > q_{(i-1)}$$

Para cada i=1,2,...,R-1. Donde *m.c.d.* es el máximo común divisor.

4.- Permutar la secuencia $(q_i)_{i \in \{0,1,...,R-1\}}$ para formar la secuencia $(r_i)_{i \in \{0,1,...,R-1\}}$ tal que:

$$r_{T(i)} = q_i, \quad i = 0,1,...,R-1$$

Donde $(T(i))_{i \in \{0,1,...,R-1\}}$ puede ser una de las cuatro la secuencia de permutación "inter-row" que se muestran en la **Tabla 2.2**, dependiendo del número de bits de entrada K.

Tabla 2.2. Trayectorias de permutación "Inter-row" en función del tamaño de bloque K.

Número de bits de entrada K	No. de filas R	Trayectorias de permutación "inter-row" <T(0),T(1),...T(R-1)>
(40≤ K ≤159)	5	<4,3,2,1,0>
(160≤ K ≤200)ó(481≤ K ≤530)	10	<9,8,7,6,5,4,3,2,1,0>
(2281≤ K ≤2480)ó(3161≤ K ≤3210)	20	<19,9,14,4,0,2,5,7,12,18,16,13,17,15,3,1,6,11,8,10>
K= Cualquier otro valor	20	<19,9,14,4,0,2,5,7,12,18,10,8,13,17,3,1,16,6,15,11>

5.- Realizar la i-ésima (i=0,1,...,R-1) permutación "intra-row" tal que:

$$if\,(C = p) \quad then$$
$$U_{i(j)} = S((j \times r_i) \bmod (p-1)), \quad j = 0,1,...,(p-2), \quad and \quad U_{i(p-1)} = 0,$$
$$end\ if$$

Donde $U_{i(j)}$ es la posición{on original del j-ésimo bit permutado de la i-ésima fila.

$$if\,(C = p+1) \quad then$$
$$U_{i(j)} = S((j \times r_i) \bmod (p-1)), \quad j = 0,1,...,(p-2), \quad and \quad U_{i(p-1)} = 0, and\ U_{i(p)} = p$$
$$Donde\ U_{i(j)}\ es\ la\ posición\ original\ del\ j-esimo\ bit\ permutado\ de\ la\ i-esima\ fila$$
$$if\,(K = R \times C) \quad then$$
$$Intercambiar \quad U_{R-1(p)} \quad con \quad U_{R-1(0)}$$
$$end\ if$$
$$end\ if$$

$$if\,(C = p-1) \quad then$$
$$U_{i(j)} = S((j \times r_i) \bmod (p-1))-1, \quad j = 0,1,...,(p-2),$$
$$end\ if$$

Donde Ui(j) es la posición original del j-ésimo bit permutado de la i-ésima fila

6.- Realizar las permutaciones "Inter-row" en la matriz rectangular basado en la trayectoria $(T(i))_{i \in \{0,1,...,R-1\}}$, donde T(i) es la posición de fila original de la i-ésima fila permutada.

2.4.1.3 Extracción con depuración de bits de salida de la matriz rectangular

Después de aplicar las permutaciones "intra-row" e "inter-row" a la matriz rectangular, los bits permutados se denotan por y'_k:

$$\begin{bmatrix} y'_1 & y'_{(R+1)} & y'_{(2R+1)} & \cdots y'_{((C-1)R+1)} \\ y'_2 & y'_{(R+2)} & y'_{(2R+2)} & \cdots y'_{((C-1)R+2)} \\ \vdots & \vdots & \vdots & \vdots \\ y'_R & y'_{2R} & y'_{3R} & \cdots y'_{(R \times C)} \end{bmatrix}$$

La salida del Interleaver interno del turbo codificador es la secuencia de bits leída columna por columna de la matriz rectangular permutada, donde el primer bit leído es y'_1 de la fila 0 y la columna 0, y el último bit leído es el y'_{CR} en la fila R y columna C-1. La salida es depurada borrando los bits que fueron rellenados al alimentar la matriz rectangular antes de las permutaciones.

El número de bits que salen del Interleaver interno del turbo codificador es "K", y el número de bits inválidos eliminados es (R x C) – K.

2.4.1.4 Ejemplo del efecto del Interleaver interno del estándar 3GPP-W CDMA sobre un bloque de tamaño K=40.

Para dejar más claro el efecto del Interleaver interno de los Turbo Codificadores para el estándar 3GPP-W CDMA, realizaremos el proceso de entrelazado según el algoritmo antes descrito. Para fines prácticos se utilizará un tamaño de bloque K=40, que es el mínimo permitido por el estándar. El vector que entrada que se procesará es *Kin={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32, 33,34,35,36,37,38,39}*

Formación de la matriz rectangular

Elegimos el numero de renglones $R = \begin{cases} 5, & if\,(40 \le K \le 159) \\ 10, & if\,((160 \le K \le 200)\ or\ (481 \le K \le 530)) \\ 20, & if\,(K = any\ other\ value) \end{cases}$

Donde dado que K=40, **entonces R=5**.

Buscamos el mínimo número primo "p" de la tabla Tabla 2.1 tal que:

$$K \le R \times (p+1), \quad \rightarrow \quad 40 \le 5 \times (7+1)$$

Obteniendo **p=7.**

Determinar el número de columnas, C, tal que :

$$C = \begin{cases} p-1, & if\,(K \le R \times (p-1)) \\ p, & if\,(R \times (p-1) < K \le (R \times p)) \\ p+1, & if\,((R \times p) < K) \end{cases} \rightarrow \quad C = \begin{cases} 6, & if\,(40 \le (5 \times 6)) \\ 7, & if\,((5 \times 6) < 40 \le (5 \times 7)) \\ 8, & if\,((5 \times 7) < 40) \end{cases}$$

Por lo que **C=8.**

Luego se escriben los bits de entrada (de 0 a K-1) en la matriz rectangular de RxC de izquierda a derecha y de arriba hacia abajo

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*
8	*9*	*19*	*11*	*12*	*13*	*14*	*15*
16	*17*	*18*	*19*	*20*	*21*	*22*	*23*
24	*25*	*26*	*27*	*28*	*29*	*30*	*31*
32	*33*	*34*	*35*	*36*	*37*	*38*	*39*

Permutaciones Intra-Row e Inter-Row

Se elije la raíz primitiva "v" relacionada con "p" en la Tabla 2.1. Donde **v=3,** ya que p=7.

Se forma la secuencia base, S(j), para las permutaciones Intra-Row tal que:

S(j)=(3 x S(j-1)) mod 7, j=1,2,3,4,5 y S(0)=1.

S(1)=(3xS(0))mod 7=(3x1)mod 7=3, S(2)=(3xS(1))mod 7=(3x3)mod 7=2,

S(3)=(3xS(2))mod 7=(3x2)mod 7=6, S(4)=(3xS(3))mod 7=(3x6)mod 7=4,

S(5)=(3xS(4))mod 7=(3x4)mod 7=5,

Por lo que **S(j)={1,3,2,6,4,5}.**

Se determina la secuencia de enteros primos q_i con i=0,1,2,3,4, tal que:

q_0=1 y g.c.d.(q_i, 6) =1, q_i>6, y q_i> $q_{(i-1)}$. Donde g.c.d.= Máximo común divisor

q_0=1, q_1=7, q_2=11, q_3=13, q_4=17.

Permutar la secuencia q_i={1,7,11,13,17} para formar la secuencia r_i tal que:

$$r_{T(i)} = q_i, \quad i = 0,1,2,3,4$$

Donde T(i) es la trayectoria de la permutación Inter-Row definida en la tabla Tabla 2.2

Y para R=5, T={4,3,2,1,0}, por lo que calculamos r

$$r_{T(0)} = r_4 = q_0 = 1, \quad r_{T(1)} = r_3 = q_1 = 7, \quad r_{T(2)} = r_2 = q_2 = 11,$$

$$r_{T(3)} = r_1 = q_3 = 13, \quad r_{T(4)} = r_0 = q_4 = 17$$

Entonces **r= {17, 13, 11, 7, 1}.**

Luego se lleva a cabo cada una de las permutaciones Intra-Row para cada uno de los renglones de i=0, 1, 2, 3, 4. Basándonos en que como C=p+1 entonces:

$$U_{i(j)} = S((j \times r_i) \bmod(6)), \quad j = 0,1,...,5, \quad y \quad U_{i(6)} = 0, \; and \; U_{i(7)} = 7$$

Para el primer renglón, i=0:

$$U_{0(0)} = S((0 \times 17) \bmod(6)) = S(0) = 1, \quad U_{0(1)} = S((1 \times 17) \bmod(6)) = S(5) = 5$$
$$U_{0(2)} = S((2 \times 17) \bmod(6)) = S(4) = 4, \quad U_{0(3)} = S((3 \times 17) \bmod(6)) = S(3) = 6$$
$$U_{0(4)} = S((4 \times 17) \bmod(6)) = S(2) = 2, \quad U_{0(5)} = S((5 \times 17) \bmod(6)) = S(1) = 3$$

$$U_{0(j)} = \{1,5,4,6,2,3,0,7\}, \quad j = 0,1,...7$$

Para el renglón 2, i=1:

$$U_{1(0)} = S((0\times 13)\bmod(6)) = S(0) = 1, \quad U_{1(1)} = S((1\times 13)\bmod(6)) = S(1) = 3$$
$$U_{1(2)} = S((2\times 13)\bmod(6)) = S(2) = 2, \quad U_{1(3)} = S((3\times 13)\bmod(6)) = S(3) = 6$$
$$U_{1(4)} = S((4\times 13)\bmod(6)) = S(4) = 4, \quad U_{1(5)} = S((5\times 13)\bmod(6)) = S(5) = 5$$
$$U_{1(j)} = \{1,3,2,6,4,5,0,7\}, \quad j = 0,1,\ldots 7$$

Para el renglón 3, i=2:

$$U_{2(0)} = S((0\times 11)\bmod(6)) = S(0) = 1, \quad U_{2(1)} = S((1\times 11)\bmod(6)) = S(5) = 5$$
$$U_{2(2)} = S((2\times 11)\bmod(6)) = S(4) = 4, \quad U_{2(3)} = S((3\times 11)\bmod(6)) = S(3) = 6$$
$$U_{2(4)} = S((4\times 11)\bmod(6)) = S(2) = 2, \quad U_{2(5)} = S((5\times 11)\bmod(6)) = S(1) = 3$$
$$U_{2(j)} = \{1,5,4,6,2,3,0,7\}, \quad j = 0,1,\ldots 7$$

Para el renglón 4, i=3:

$$U_{3(0)} = S((0\times 7)\bmod(6)) = S(0) = 1, \quad U_{3(1)} = S((1\times 7)\bmod(6)) = S(1) = 3$$
$$U_{3(2)} = S((2\times 7)\bmod(6)) = S(2) = 2, \quad U_{3(3)} = S((3\times 7)\bmod(6)) = S(3) = 6$$
$$U_{3(4)} = S((4\times 7)\bmod(6)) = S(4) = 4, \quad U_{3(5)} = S((5\times 7)\bmod(6)) = S(5) = 5$$
$$U_{3(j)} = \{1,3,2,6,4,5,0,7\}, \quad j = 0,1,\ldots 7$$

Para el renglón 5, i=4:

$$U_{4(0)} = S((0\times 1)\bmod(6)) = S(0) = 1, \quad U_{4(1)} = S((1\times 1)\bmod(6)) = S(1) = 3$$
$$U_{4(2)} = S((2\times 1)\bmod(6)) = S(2) = 2, \quad U_{4(3)} = S((3\times 1)\bmod(6)) = S(3) = 6$$
$$U_{4(4)} = S((4\times 1)\bmod(6)) = S(4) = 4, \quad U_{4(5)} = S((5\times 1)\bmod(6)) = S(5) = 5$$

Y finalmente se hace el intercambio $U_{4,(7)}$ con $U_{4,(0)}$, obteniendo:

$$U_{4(j)} = \{7,3,2,6,4,5,0,1\}, \quad j = 0,1,\ldots 7$$

Y la matriz con las permutaciones Intra-Row:

$$U_{i,j} = \begin{Bmatrix} 1 & 5 & 4 & 6 & 2 & 3 & 0 & 7 \\ 1 & 3 & 2 & 6 & 4 & 5 & 0 & 7 \\ 1 & 5 & 4 & 6 & 2 & 3 & 0 & 7 \\ 1 & 3 & 2 & 6 & 4 & 5 & 0 & 7 \\ 7 & 3 & 2 & 6 & 4 & 5 & 0 & 1 \end{Bmatrix}$$

Aplicando las permutaciones a cada renglón de la matriz original (izquierda), en la siguiente tabla tenemos como resultado los valores permutados (derecha).

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*		*1*	*5*	*4*	*6*	*2*	*3*	*0*	*7*
8	*9*	*10*	*11*	*12*	*13*	*14*	*15*		*9*	*11*	*10*	*14*	*12*	*13*	*8*	*15*
16	*17*	*18*	*19*	*20*	*21*	*22*	*23*		*17*	*21*	*20*	*22*	*18*	*19*	*16*	*23*
24	*25*	*26*	*27*	*28*	*29*	*30*	*31*		*25*	*27*	*26*	*30*	*28*	*29*	*24*	*31*
32	*33*	*34*	*35*	*36*	*37*	*38*	*39*		*39*	*35*	*34*	*38*	*36*	*37*	*32*	*33*

Luego se realizan las permutaciones Inter-Row, basados en el vector T(i)={4,3,2,1,0}, con lo que obtenemos la matriz:

39	*35*	*34*	*38*	*36*	*37*	*32*	*33*
25	*27*	*26*	*30*	*28*	*29*	*24*	*31*
17	*21*	*20*	*22*	*18*	*19*	*16*	*23*
9	*11*	*10*	*14*	*12*	*13*	*8*	*15*
1	*5*	*4*	*6*	*2*	*3*	*0*	*7*

Lectura de los valores de salida

Para obtener los valores de salida leemos los valores de la matriz anterior columna por columna, de arriba hacia abajo y de izquierda a derecha, obteniendo: Kout=*{39,25,17,9,1,35,27,21,11,5,34,26,20,10,4,38,30,22,14,6,36,28,18,12,2,37,29,19,13,3,32,24,16,8,0,33,31,23,15,7}*.

2.4.2 3GPP – LTE

LTE es un nuevo estándar de la norma 3GPP, que para muchos es una evolución en los sistemas 3G y para otros es un nuevo concepto de arquitectura evolutiva, 4G. LTE será la clave para el despegue de Internet móvil, servicios de transmisión de datos a más de 300 metros y video de alta definición. La novedad de LTE es la interfaz radioeléctrica basada en acceso múltiple por división de frecuencias ortogonales (OFDMA, del inglés *Orthogonal Frecuency Division Multiple Access*) para el enlace descendente y acceso múltiple por división de frecuencia, de una sola portadora (SC-FDMA, del inglés *Single Carrier – Frecuency Division Multiple Access*) para el enlace ascendente.

LTE cuenta con tasas de enlaces pico de más de 300 Mbits/s para enlaces de bajada para tecnologías MIMO de 4x4 antenas, y de más de 80 Mbits/s para enlaces de subida.

2.4.2.1 Algoritmo del Interleaver interno del Turbo Codificador para el estándar 3GPP-LTE

Los bits entrantes al Interleaver interno del turbo codificador están denotados por $c_0,c_1,\ldots,c_{k-1}$, donde K es el número de bits del bloque entrante y puede tomar cualquiera de los valores mostrado en la Tabla 2.3.

Los bits de salida del Interleaver interno del turbo codificador están denotados por $c'_0,c'_1,...,c'_{k-1}$. Donde la ecuación (1.5) es la relación entre los bits de entrada y los bits de salida.

$$c'_i = c_{\Pi(i)}, \qquad i = 0,1,...,(K-1) \tag{1.5}$$

Para encontrar la relación de los índices de los bits de salida "i" y los índices de los bits de entrada "π(i) " se utiliza la formula de la ecuación (1.6).

$$\Pi(i) = (f_1 \times i + f_2 \times i^2) \bmod K \tag{1.6}$$

Los parámetros f_1 y f_2 dependen del tamaño de bloque K y podemos encontrarlos en la Tabla 2.3.

2.4.2.2 Ejemplo del efecto del Interleaver interno del estándar 3GPP-LTE sobre un bloque de tamaño K=40.

A continuación se muestra un ejemplo para dejar más claro el efecto que el Interleaver interno de este estándar tiene sobre un bloque de entrada de tamaño K=40.

Supongamos que nuestros datos de entrada son *C={0 ,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16, 17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39}*

Entonces, para obtener los datos de salida tenemos que:

$$C'_i = C_{\Pi(i)}, \quad \Pi(i) = (3 \times i + 10 \times i^2) \bmod 40, \quad i = 0,1,...,(39)$$

$$\Pi(i) = 0,13,6,19,12,25,18,31,24,37,30,3,36,9,2,15,8,21,14,27,$$
$$20,33,26,39,32,5,38,11,4,17,10,23,16,29,22,35,28,1,34,7$$

Aplicando este indexado a los datos de entrada, obtenemos los valores de salida C'_i={0,13,6,19,12,25,18,31,24,37,30,3,36,9,2,15,8,21,14,27,20,33,26,39,32,5,38,11,4,17,10,23, 16,29,22,35,28,1,34,7}.

Tabla 2.3. Parámetros f1 y f2 correspondientes a cada tamaño de bloque K, para el estándar 3GPP-LTE.

i	K_i	f_1	f_2	*i*	K_i	f_1	f_2	*i*	K_i	f_1	f_2	*i*	K_i	f_1	f_2
1	40	3	10	48	416	25	52	95	1120	67	140	142	3200	111	240
2	48	7	12	49	424	51	106	96	1152	35	72	143	3264	443	204
3	56	19	42	50	432	47	72	97	1184	19	74	144	3328	51	104
4	64	7	16	51	440	91	110	98	1216	39	76	145	3392	51	212
5	72	7	18	52	448	29	168	99	1248	19	78	146	3456	451	192
6	80	11	20	53	456	29	114	100	1280	199	240	147	3520	257	220
7	88	5	22	54	464	247	58	101	1312	21	82	148	3584	57	336
8	96	11	24	55	472	29	118	102	1344	211	252	149	3648	313	228
9	104	7	26	56	480	89	180	103	1376	21	86	150	3712	271	232
10	112	41	84	57	488	91	122	104	1408	43	88	151	3776	179	236
11	120	103	90	58	496	157	62	105	1440	149	60	152	3840	331	120
12	128	15	32	59	504	55	84	106	1472	45	92	153	3904	363	244
13	136	9	34	60	512	31	64	107	1504	49	846	154	3968	375	248
14	144	17	108	61	528	17	66	108	1536	71	48	155	4032	127	168
15	152	9	38	62	544	35	68	109	1568	13	28	156	4096	31	64
16	160	21	120	63	560	227	420	110	1600	17	80	157	4160	33	130
17	168	101	84	64	576	65	96	111	1632	25	102	158	4224	43	264
18	176	21	44	65	592	19	74	112	1664	183	104	159	4288	33	134
19	184	57	46	66	608	37	76	113	1696	55	954	160	4352	477	408
20	192	23	48	67	624	41	234	114	1728	127	96	161	4416	35	138
21	200	13	50	68	640	39	80	115	1760	27	110	162	4480	233	280
22	208	27	52	69	656	185	82	116	1792	29	112	163	4544	357	142
23	216	11	36	70	672	43	252	117	1824	29	114	164	4608	337	480
24	224	27	56	71	688	21	86	118	1856	57	116	165	4672	37	146
25	232	85	58	72	704	155	44	119	1888	45	354	166	4736	71	444
26	240	29	60	73	720	79	120	120	1920	31	120	167	4800	71	120
27	248	33	62	74	736	139	92	121	1952	59	610	168	4864	37	152
28	256	15	32	75	752	23	94	122	1984	185	124	169	4928	39	462
29	264	17	198	76	768	217	48	123	2016	113	420	170	4992	127	234
30	272	33	68	77	784	25	98	124	2048	31	64	171	5056	39	158
31	280	103	210	78	800	17	80	125	2112	17	66	172	5120	39	80
32	288	19	36	79	816	127	102	126	2176	171	136	173	5184	31	96
33	296	19	74	80	832	25	52	127	2240	209	420	174	5248	113	902
34	304	37	76	81	848	239	106	128	2304	253	216	175	5312	41	166
35	312	19	78	82	864	17	48	129	2368	367	444	176	5376	251	336
36	320	21	120	83	880	137	110	130	2432	265	456	177	5440	43	170
37	328	21	82	84	896	215	112	131	2496	181	468	178	5504	21	86
38	336	115	84	85	912	29	114	132	2560	39	80	179	5568	43	174
39	344	193	86	86	928	15	58	133	2624	27	164	180	5632	45	176
40	352	21	44	87	944	147	118	134	2688	127	504	181	5696	45	178
41	360	133	90	88	960	29	60	135	2752	143	172	182	5760	161	120
42	368	81	46	89	976	59	122	136	2816	43	88	183	5824	89	182
43	376	45	94	90	992	65	124	137	2880	29	300	184	5888	323	184
44	384	23	48	91	1008	55	84	138	2944	45	92	185	5952	47	186
45	392	243	98	92	1024	31	64	139	3008	157	188	186	6016	23	94
46	400	151	40	93	1056	17	66	140	3072	47	96	187	6080	47	190
47	408	155	102	94	1088	171	204	141	3136	13	28	188	6144	263	480

Capítulo 2. El Interleaver

Capítulo 3

Arquitecturas de Interleavers

La arquitectura que se presenta en este trabajo de tesis, es una arquitectura funcional cuyo diseño está basado en varios trabajos previos relacionados, así como en el aprovechamiento de avances en reducciones a nivel algorítmico, y simplificaciones computacionales para el cálculo de la operación módulo (esta última mostrada en el apéndice C).

A lo largo de este capítulo se presentan:

- Algunos trabajos previos sobre interleavers.
- Nuestra arquitectura para el estándar 3GPP-WCDMA
- Nuestra arquitectura final para ambos estándares 3GPP-WCDMA y 3GPP-LTE

3.1 Hardware de un interleavers de baja complejidad para Turbo Códigos [14].

Zhongfeng Wang y Qingwei Li, presentan una arquitectura de un Interleaver usado en el turbo codificador del estándar 3GPP- W CDMA, en el cual para diferentes tamaños de bloque "K" se requieren distintas trayectorias de entrelazado, por lo que si se utilizara una aproximación basada en ROMs requeriríamos mas de 100M bits de almacenaje para todas las trayectorias de entrelazado, lo cual es inaceptable viéndolo desde un punto de vista de costo en hardware. En este diseño se intenta hacer mejoras o aprovechar mejoras a nivel algorítmico, arquitectónico y optimizaciones a nivel de circuitería VLSI. De esta forma se pueden eliminar algunos multiplicadores, divisores y hardware que realiza operaciones de módulo teniendo como resultado la reducción en latencia computacional y consumo de potencia.

La arquitectura de [14], primeramente se encarga de calcular todos los parámetros necesarios de este estándar (C,R,P,v,U,Q,S).

El número de renglones, R, es encontrado mediante lógica contenida en el bloque "Computing Core" de la Figura 3.5, donde se hacen las comparaciones correspondientes:

$$R=\begin{cases}5, & if\,(40\le K\le 159)\\ 10, & if\,((160\le K\le 200)\ or\ (481\le K\le 530))\\ 20, & if\,(K=any\ other\ value)\end{cases}$$

Para encontrar los parámetros *P* y *v*, buscamos valores de *P* que satisfagan la ecuación *P*R≥K-R*. Donde *P* y *v* están almacenados en una ROM por lo que se leen los valores de *P* hasta encontrar el que satisfaga la ecuación, encontrando también *v* dado que se almacenan en pares. Las comparaciones necesarias se hacen dentro del bloque "Computing Core".

Para calcular el arreglo "*S*" usado en las permutaciones, se basan en la ecuación (3.1).

$$S[K]=(v\times S[K-1])\bmod P,\quad K=1,2,...,P-1 \tag{3.1}$$

Donde sabiendo que *v*=2,3,5,6,7 y 19 se decide calcular *S[k]* de forma gradual obteniendo primero *S*[*k*-1]. Entonces asumiendo X<P y Y<P se tiene que:

$$(X+Y)\bmod P=\begin{cases}X+Y, & if\,(X+Y)<P\\ X+Y-P, & en\ cualquier\ otro\ caso\end{cases}$$

Y dado que:

$$S[k-1]<P,\qquad (S[k-1]*2\ mod\ P)=S[k-1]*2\ \ ó\ \ S[k-1]*2-P$$

Dependiendo de si *S[k-1]*2 <P* o no. Entonces haciendo *X= S[k-1]*2 mod P* y *Y= S[k-1]*, se puede calcular *S[k-1]*3 mod P* usando la ecuación (3.1). En la Figura 3.1 se muestra la arquitectura usada para calcular S[*k*] a partir de S[*k*-1] para cualquier valor de *v*.

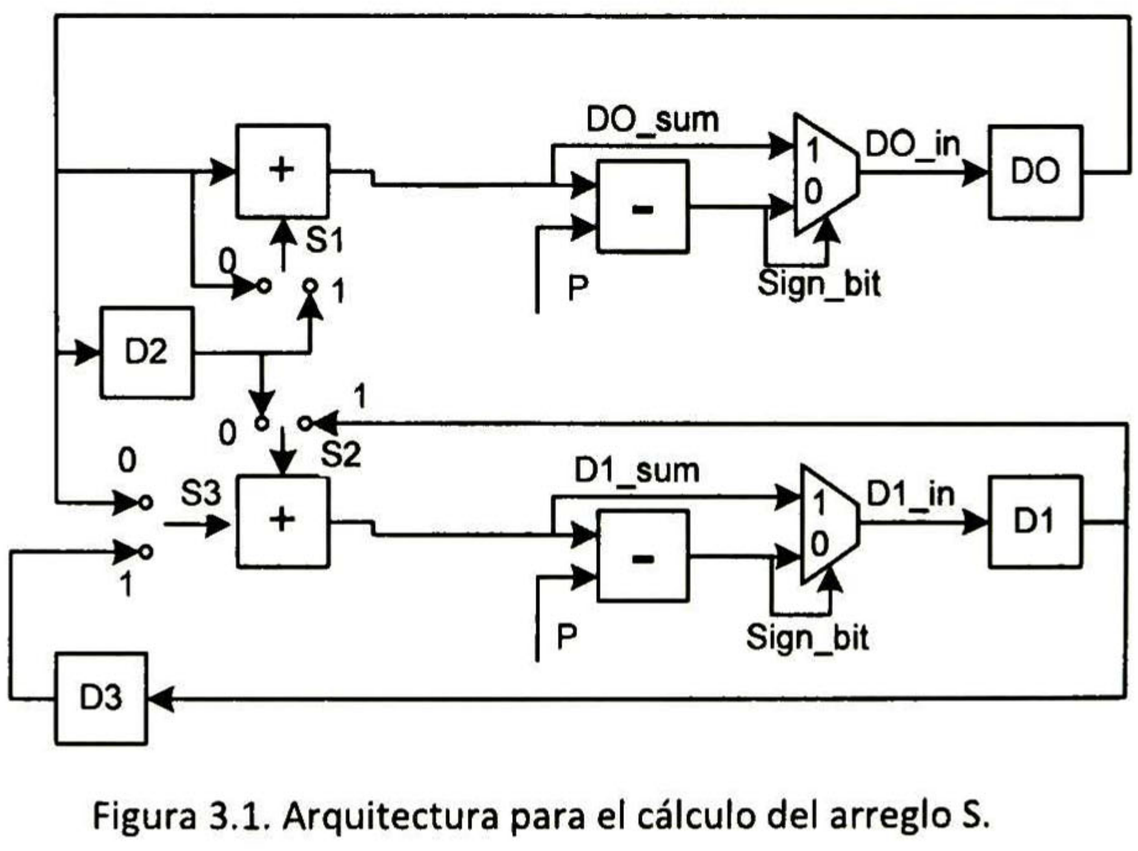

Figura 3.1. Arquitectura para el cálculo del arreglo S.

En la figura Figura 3.1 D0,D1,D2 y D3 son registros donde se van almacenando los resultados parciales de las operaciones de módulo. Esta porción de arquitectura así como sus señales de control S1, S2 y S3 están contenidas también dentro del bloque "Computing Core"

Para encontrar el vector Q, primeramente se pre calculan todos los posibles valores y se guardan en una ROM, entonces la tarea es encontrar los índices o direcciones de memoria que se deben brincar en cada caso. En [14] no se menciona de forma clara como se hace esto, pero esta tarea también la realiza el "Computing Core". En la Figura 3.2 se muestra el diagrama de estados que rige el comportamiento de la arquitectura del presente diseño.

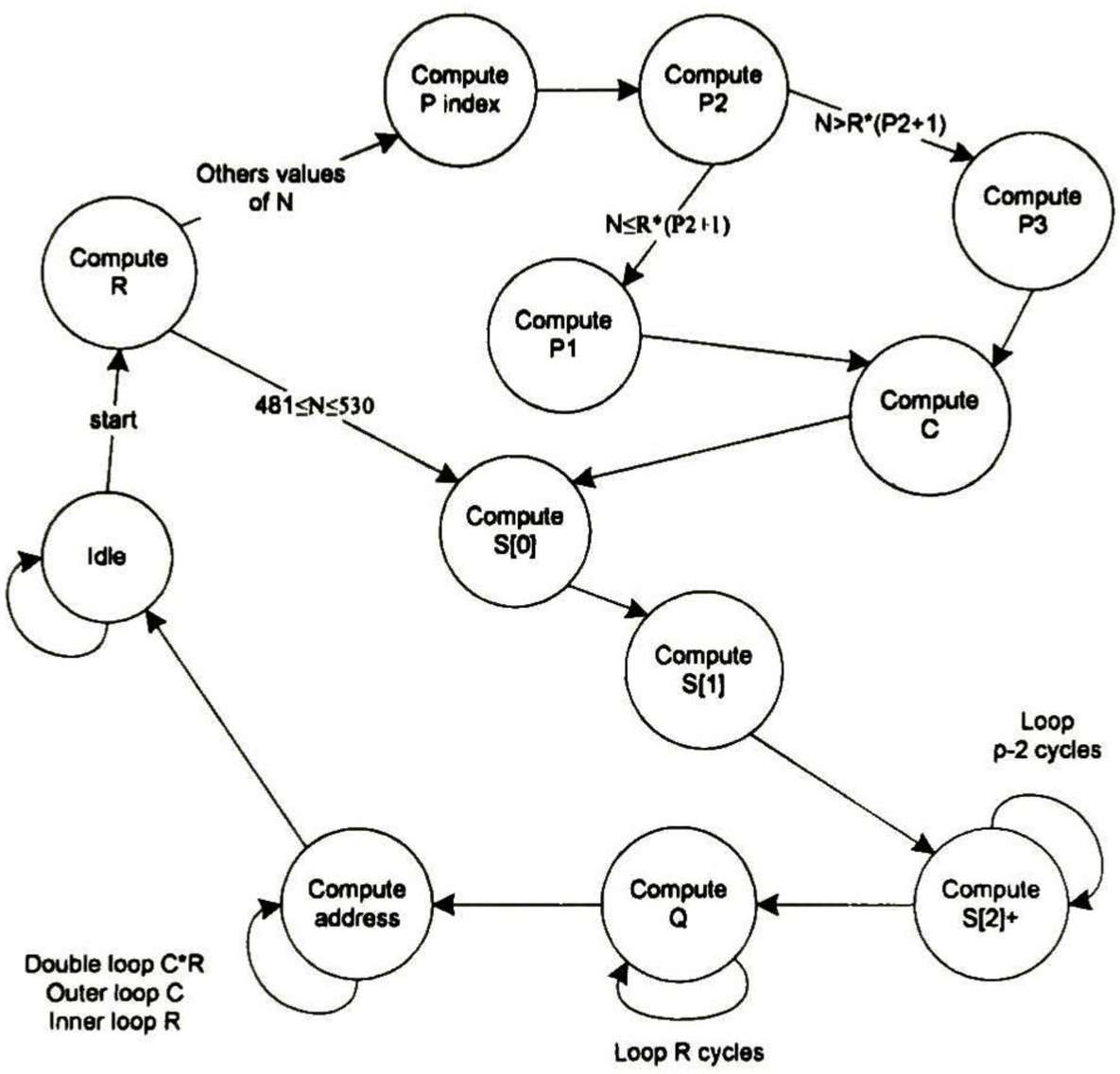

Figura 3.2. Diagrama de estados para el generador de direcciones de entrelazado.

Suponiendo que los símbolos de entrada del Interleaver son $A_0, A_1, \ldots, A_{K-1}$, después de insertar los bits de relleno y de escribirlos por filas, tenemos:

$$\begin{bmatrix} X_{0,0} & X_{0,1} & \cdots & X_{0,C-1} \\ X_{1,0} & X_{1,1} & \cdots & X_{1,C-1} \\ \vdots & \vdots & & \vdots \\ X_{R-1,0} & X_{R-1,1} & \cdots & X_{R-1,C-1} \end{bmatrix}$$

Donde $X_{i,j} = A_{i*C+j}$, y si $i*C+j \geq K$, entonces $X_{i,j}$ es un bit de relleno.

Para la permutación "intra-row", se calcula el parámetro $U_{i,j}$, el cual representa la posición de bit original del *j*-esimo bit permutado de la *i*-esima fila, según la ecuación (3.2).

$$U_{i,j} = S((j \times r_i) \bmod (P-1)), \quad j = 0,1,\ldots,p-2 \tag{3.2}$$

Donde el arreglo "*r*" se define como $rT(i)=Qi$, $i=0,1,\ldots,R-1$, y $T(i)_{i\in\{0,1,\ldots,R-1\}}$ es la trayectoria de permutación "inter-row" . Después de la permutación "intra-row", la secuencia se convierte en:

$$\begin{bmatrix} Y_{0,0} & Y_{0,1} & \cdots & Y_{0,C-1} \\ Y_{1,0} & Y_{1,1} & \cdots & Y_{1,C-1} \\ \vdots & \vdots & & \vdots \\ Y_{R-1,0} & Y_{R-1,1} & \cdots & Y_{R-1,C-1} \end{bmatrix}$$

Donde $Y_{i,j}=X_{i,Ui,j}$.

La permutación "Inter-row" permuta las filas de acuerdo a *T(i)*, donde *T(i)* es la posición original de la j-esima fila permutada. Después de la permutación "inter-row" la secuencia se convierte en:

$$\begin{bmatrix} Z_{0,0} & Z_{0,1} & \cdots & Z_{0,C-1} \\ Z_{1,0} & Z_{1,1} & \cdots & Z_{1,C-1} \\ \vdots & \vdots & & \vdots \\ Z_{R-1,0} & Z_{R-1,1} & \cdots & Z_{R-1,C-1} \end{bmatrix}$$

Donde $Z_{i,j}=Y_{T(i),j}=X_{T(i),UT(i)j}=A_{T(i)*C+UT(i)j}$

Entonces tenemos que las permutaciones intra-row e inter-row pueden llevarse acabo como se muestra en el siguiente pseudo código:

for$(i=0,\ i=R-1)$
  for$(j=0,\ j=C-1)$
    calcular $U_{i,j}$
    $Z_{i,j} = Y_{T(i),j} = X_{T(i),U_{T(i),j}}$
  end
end

Después de hacer las permutaciones, la secuencia de salida, *Z*, puede ser leída en el orden $Z_{0,0}, Z_{1,0}, \ldots, Z_{R-1,0}, Z_{0,1}, Z_{1,1}, \ldots, Z_{R-1,1}, \ldots, Z_{0,C-1}, \ldots, Z_{R-1,C-1}$.

Lo que se hace en este diseño es cambiar a "*i*" como índice interno y a "*j*" como el índice externo, y calcular dirección de lectura intercalada en lugar de intercalar datos de acuerdo a:

$$
\begin{aligned}
&for(j=0,\ j=C-1)\\
&\quad for(i=0,\ i=R-1)\\
&\quad\quad calcular \quad U_{i,j}\\
&\quad\quad Addr=T(i)\times C+U_{T(i),j}\\
&\quad\quad if\ Addr\le K-1,\ Saco\ Addr\\
&\quad end\\
&end
\end{aligned}
$$

Donde según la ecuación (3.3) para poder calcular $U_{T(i),j}$ es necesario encontrar *SIdx* que es el índice de "S"

$$U_{T(i),j}=S((j\times r_{T(i)})\bmod(P-1))=S((j\times Q_i)\bmod(P-1)) \tag{3.3}$$

Tenemos que *SIdx=(j*Qi) mod (P-1)*, lo cual según [14] puede ser calculado mediante el hardware de la Figura 3.3, sin embargo no queda claro en que algoritmo se basa esta arquitectura y tampoco es evidente como puede calcularse *SIdx* a partir de dicha arquitectura.

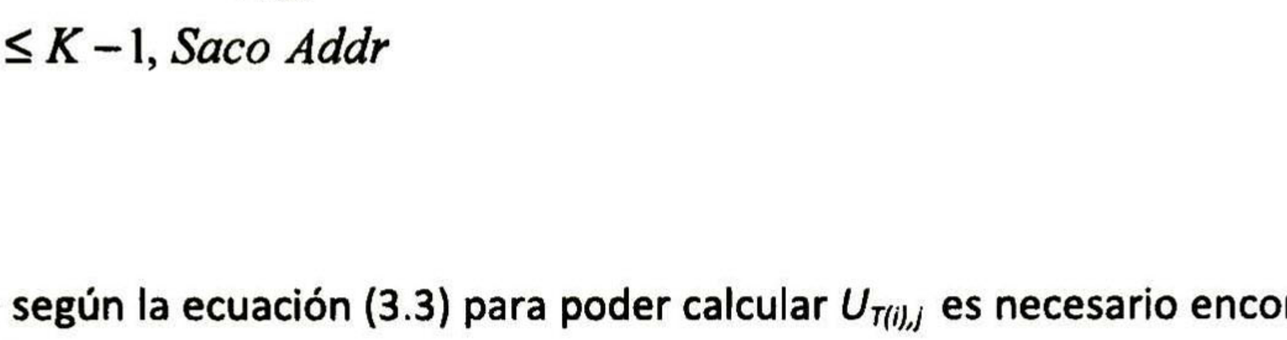

Figura 3.3. Arquitectura para el cálculo del índice del arreglo S.

Para evitar la multiplicación "j*Q_i" se calcula SIdx(i,j) de forma recursiva a partir de SIdx(i,j-1) según la ecuación (3.4).

$$SIdx_{i,j}=(j\times Q_i)\bmod(P-1)=(SIdx_{i,j-1}+Q_i)\bmod(P-1) \tag{3.4}$$

Donde Q tiene que ser calculado previamente y a la vez aplicarle "mod (P-1)" , lo cual se lleva a cabo mediante el hardware mostrado en la Figura 3.4.

Finalmente la arquitectura completa de este Interleaver es la que se muestra en la Figura 3.5, donde se muestran los principales bloques y como se interconectan entre si.

Como podemos ver, esta arquitectura basa su funcionamiento principalmente en el uso de memorias ROM, donde se guardan la mayoría de los parámetros necesarios para los cálculos de las direcciones de entrelazado, y en desarrollar operaciones de forma iterativa.

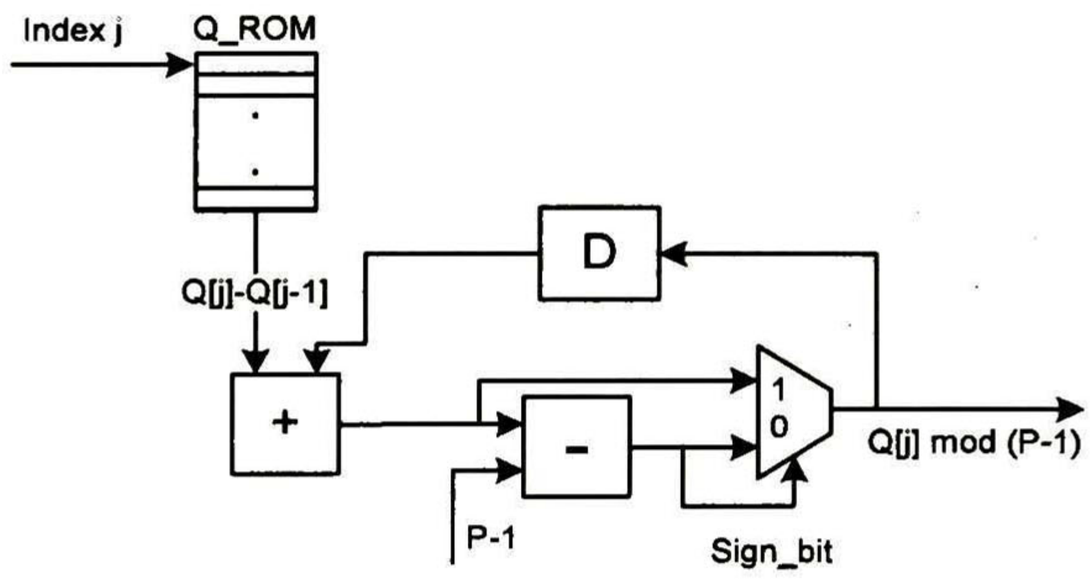

Figura 3.4. Calculo de Q[i] mod (P-1).

Esta arquitectura tiene el acierto de hacer las operaciones en forma iterativa con lo que se ahorra en hardware, el uso de ROMs para el almacenamiento de algunos parámetros en lugar de calcularlos también ayuda a ahorrar en hardware.

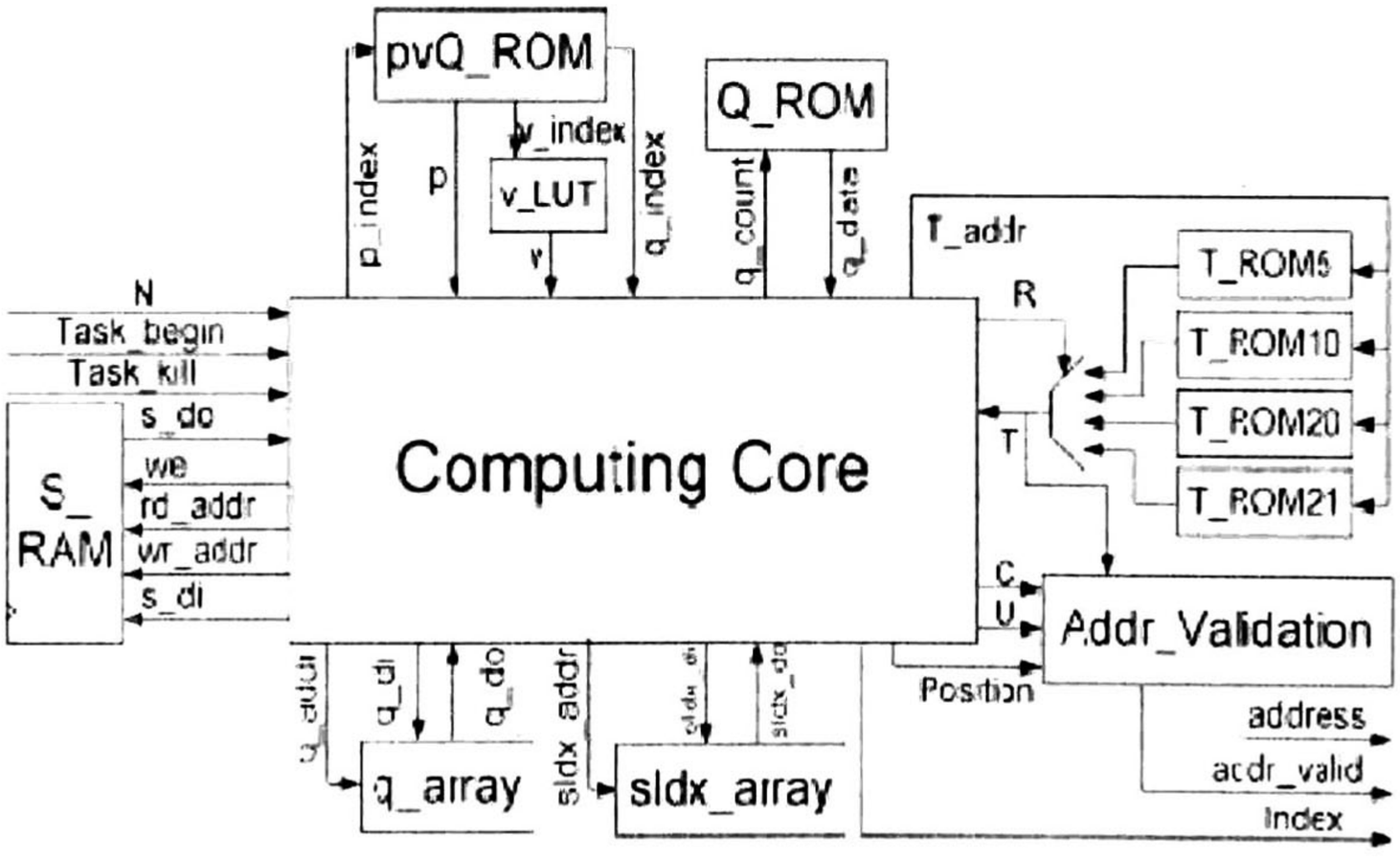

Figura 3.5. Arquitectura final del generador de direcciones de entrelazado, tomado de [14].

Las desventajas de esta arquitectura son que, no se explica como el módulo "Computing Core" realiza ciertas operaciones, otro problema con esta arquitectura es que no explica cómo trata las excepciones que surgen en el cálculo de las direcciones de entrelazado para este estándar, donde una vez calculadas las direcciones de forma normal, se tienen que hacer cambio de índices solo en ciertas direcciones.

3.2 *Un Interleaver eficiente en hardware para turbo decodificación en el estándar 3G [15]*

Resumiendo el algoritmo para el entrelazado (según el estándar 3GPP-WCDMA) en los siguientes pasos, tenemos:

1. Se introducen los datos a la matriz rectangular RxC fila por fila rellenando con cero cuando se presente el caso K< R*C.
2. Se lleva a cabo la permutación "intra-row" de la matriz rectangular, basándonos en la secuencia base recursivamente construida "S"
3. Se lleva a cabo la permutación "Inter-row" de la matriz rectangular, basándonos en la bien definida trayectoria de permutación "T" y en la secuencia de mínimos números primos permutada "r" (obtenida de T y una secuencia de mínimos numero primos ordenados "q").
4. La salida de datos, se lee columna por columna de la matriz rectangular, depurando los bits inválidos.

La innovación en esta aproximación es combinar los pasos del 2 al 4 en una única función que permite el cálculo de direcciones en "tiempo-real". Se separa el proceso de entrelazado en una fase de preparación y en otra de cálculo de direcciones en "tiempo-real" La fase de preparación, obtiene los parámetros necesarios para el cálculo de las direcciones en "tiempo-real", utilizando un tiempo casi constante para cualquier longitud de las secuencias de datos. En este proceso se determinan R,C,p,s,T y r, tal y como lo requiere el algoritmo. Al final del proceso de preparación el registro DONE le avisa al generador de direcciones en tiempo real que las operaciones han sido completadas y los registros necesarios han sido preparados. Entonces, el generador de direcciones en tiempo real utiliza las secuencias preparadas para calcular las direcciones "al vuelo" Las dos unidades se comunican a través de simples protocolos de control de registros. El decodificador del procesador interactúa con la etapa de generación de direcciones en "tiempo-real" a través de las señales de control "Interleave" ó "Deinterleave" Durante las iteraciones de decodificación, la fase de preparación se puede apagar para ahorrar energía. La j-esima dirección generalizada de entrelazado es calculada usando la ecuación (3.5), donde los operadores mostrados tienen el significado usual. En la Figura 3.6 se muestra la arquitectura general del Interleaver de esta aproximación.

$$Interleaved\_Address[j] = C \times T[j\%R] + S[((j/R) \times r[T[j\%R]])\%(p-1)] \qquad (3.5)$$

El mecanismo de control para la generación de direcciones está dado por el siguiente pseudo código:

```
Initialize   counter
While(counter < K)
   Compute   Interleaved _ Address[j]
   if (Interleaved _ Address[j] < K)
         Increment counter
   Increment j
```

Donde, la latencia y requerimientos computacionales de la fase de cálculo de direcciones en tiempo-real están en función de la longitud de cada secuencia de entrada.

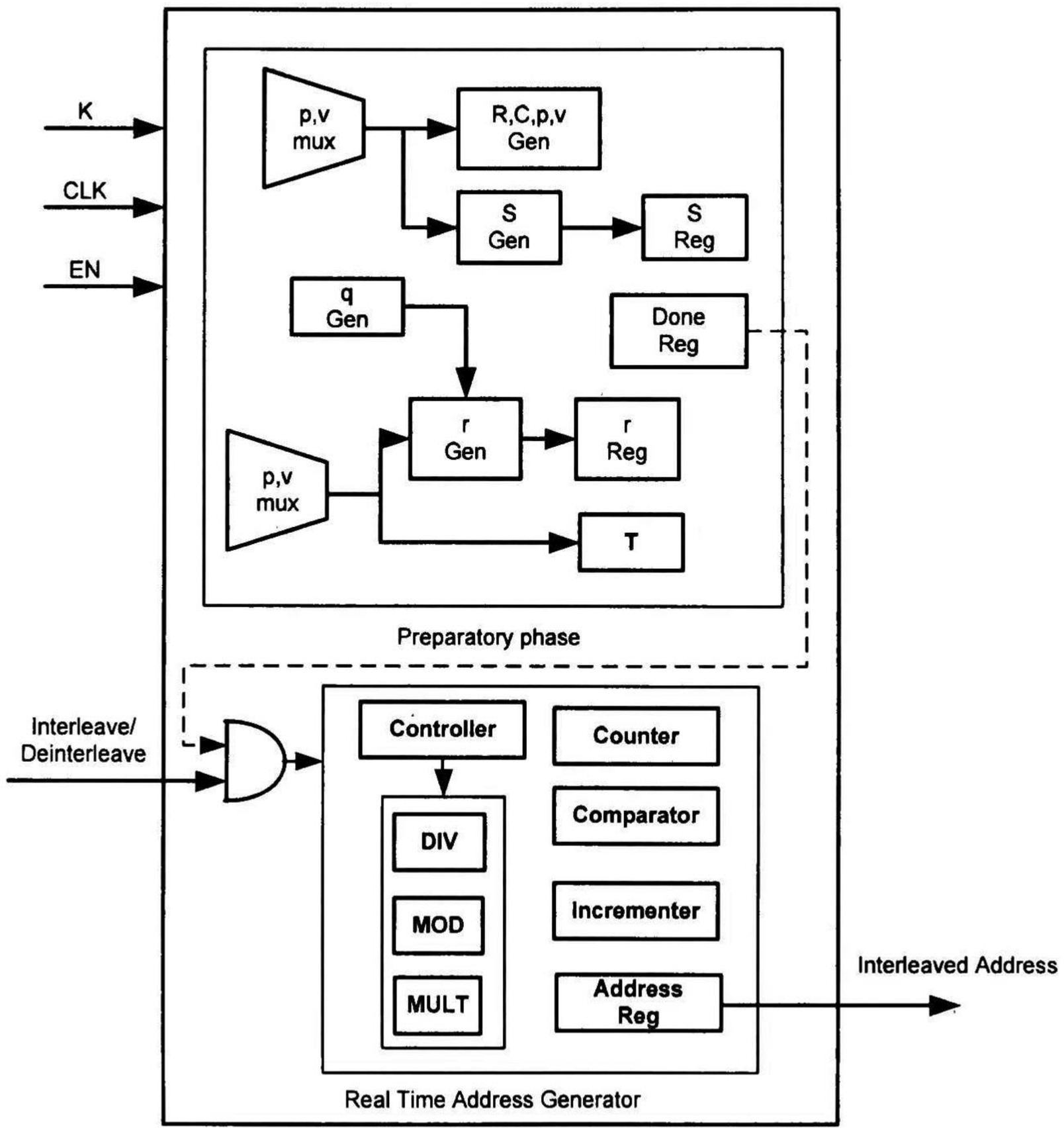

Figura 3.6. Hardware de la arquitectura del Interleaver de dos estados, tomado de [15].

Este diseño solo menciona el procedimiento de forma muy general de cómo se debe de hacer el proceso para generar las direcciones de entrelazado, cuenta con módulos específicos para hacer cada tarea por lo que baja su eficiencia en hardware. Otro problema con este diseño es que

no muestra ningún bloque dedicado a lidiar con las excepciones que el estándar presenta en la generación de direcciones de entrelazado.

3.3 Interleaver con hardware configurable de bajo costo para turbo decodificación en 3G [16].

Después de analizar el algoritmo del entrelazado, se llega a la conclusión de que las siguientes operaciones se tienen que hacer con la finalidad de cumplir con los requerimientos.

- Funciones de módulo
- Permutaciones (intra-row e inter-row)
- Multiplicaciones
- Uso de memorias
- Encontrar los mínimos números primos
- Calculo del máximo común divisor
- Manejo de excepciones

Algunas de estas complejas funciones pueden ser implementadas con el apoyo de memorias ROM, mientras las restantes pueden ser simplificadas para reducir el uso de hardware.

Esta arquitectura trabaja en dos modalidades para facilitar el proceso y aprovechar el hardware al máximo, en modo "pre-computation" que es donde se encuentran los parámetros básicos necesarios para el cálculo de las direcciones de entrelazado y el modo "run-time" que es donde se calculan las direcciones finales de entrelazado.

3.3.1 Pre-computation

La pre-computación en el hardware facilita el rápido cambio en el tamaño del bloque del Interleaver en funcionamiento. Debido a que diferentes tamaños de bloques tienen diferentes trayectorias de entrelazado, cada vez que se cambia el tamaño del bloque una nueva pre-computación es requerida. El diagrama de flujo para la fase de pre-computo se muestra en la Figura 3.7 a). El único parámetro pasado al Interleaver es el tamaño del bloque y el resto de los parámetros son calculados por el mismo hardware. Los parámetros que se calculan en la fase de pre-computo son: numero de filas, "R", numero de columnas, "C", secuencia de mínimos primos, "q(i)", trayectoria de permutación "inter-row", "T(j)", números primos "p" y su entero asociado "v" Adicionalmente la trayectoria de permutación "intra-row" S(j), también es calculada y guardada en una RAM en la etapa de pre-computo. Para calcular el número de filas y la trayectoria de permutación "intra-row", se desarrollan funciones lógicas para substituir las "Look Up Tables", estas funciones lógicas consumen menos hardware comparado con la implementación con "Look Up Tables"

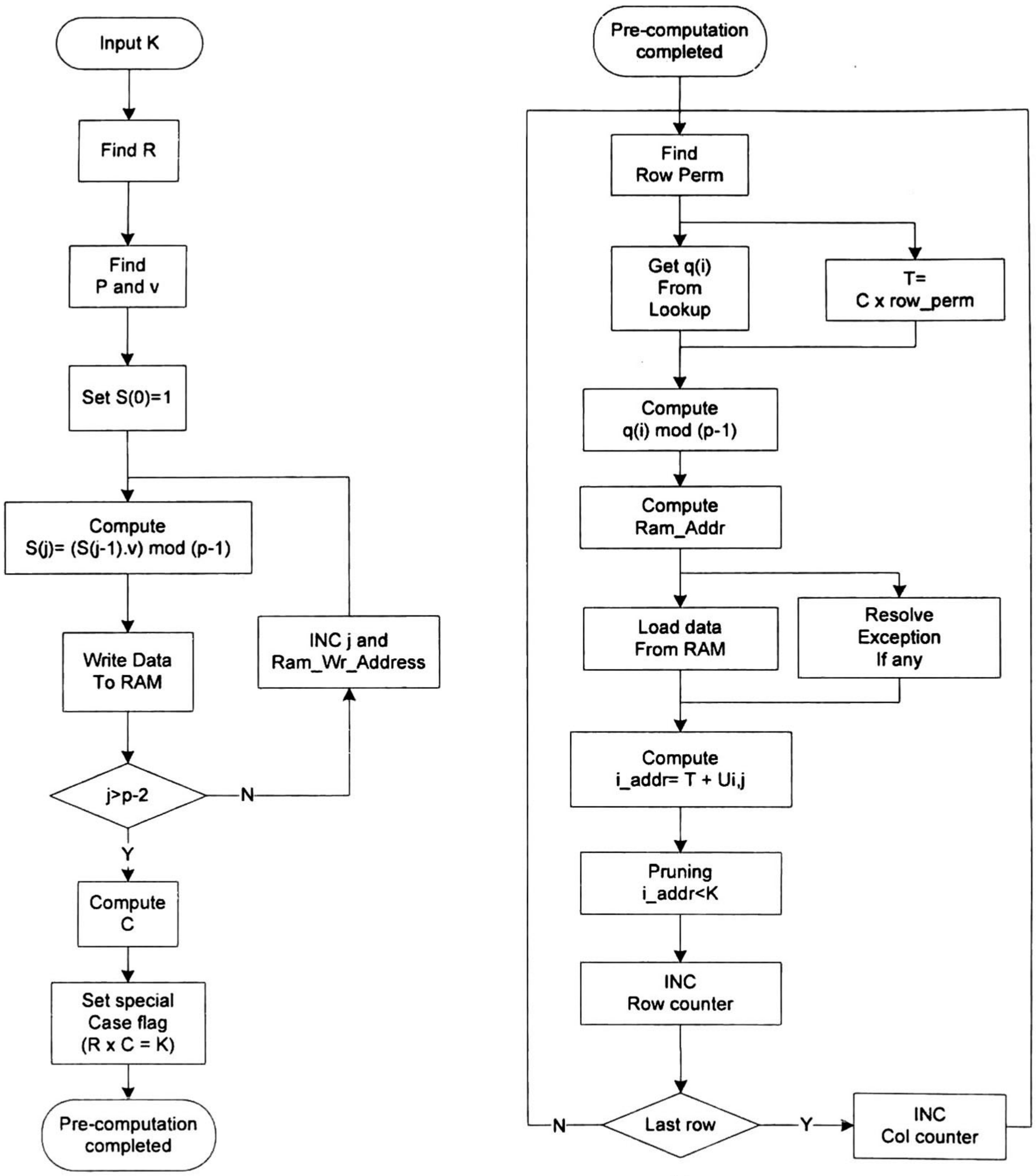

Figura 3.7. Diagramas de flujo para la fase de Pre-cálculo (a) y Run Time (b), sacado de [16].

Para encontrar la secuencia de mínimos números primos (q), este trabajo se enfoca en encontrar "q mod (p-1)" en lugar de encontrar "q" Esto nos trae el beneficio de encontrar la dirección de la RAM en forma recursiva y evitar el cálculo de la operación módulo.

Al encontrar "q mod (p-1)" mediante simulaciones las siguientes observaciones fueron hechas:

- Aproximadamente la mitad de las trayectorias q(i) son las mismas
- Otras pueden tener 1 diferencia en la secuencia
- Solo una trayectoria tiene 2 diferencias

En este diseño se manipulan todos los valores de "q" para tener q(i)<2(p-1) para todas las posibles secuencias de "q". Utilizando las observaciones mencionadas, se construyen secuencias de "q" y se dividen en subgrupos para diferentes valores de "p", esto ayuda a reducir el tamaño total en requerimientos de ROMs para esos valores. Después de obtener el valor de "p" se calcula, con ayuda de la trayectoria de permutación "inter-row", la dirección en ROM del subgrupo apropiado de "q", todo esto se hace en el modo "run-time".

El módulo mostrado en la Figura 3.8 (a) es parte del hardware necesario para obtener "q mod (p-1)" según el algoritmo mostrado en el apéndice C, este hardware es multiplexado de tal manera que pueda ser utilizado también en la fase de pre-computo para calcular la secuencia de permutación "intra-row" S(j).

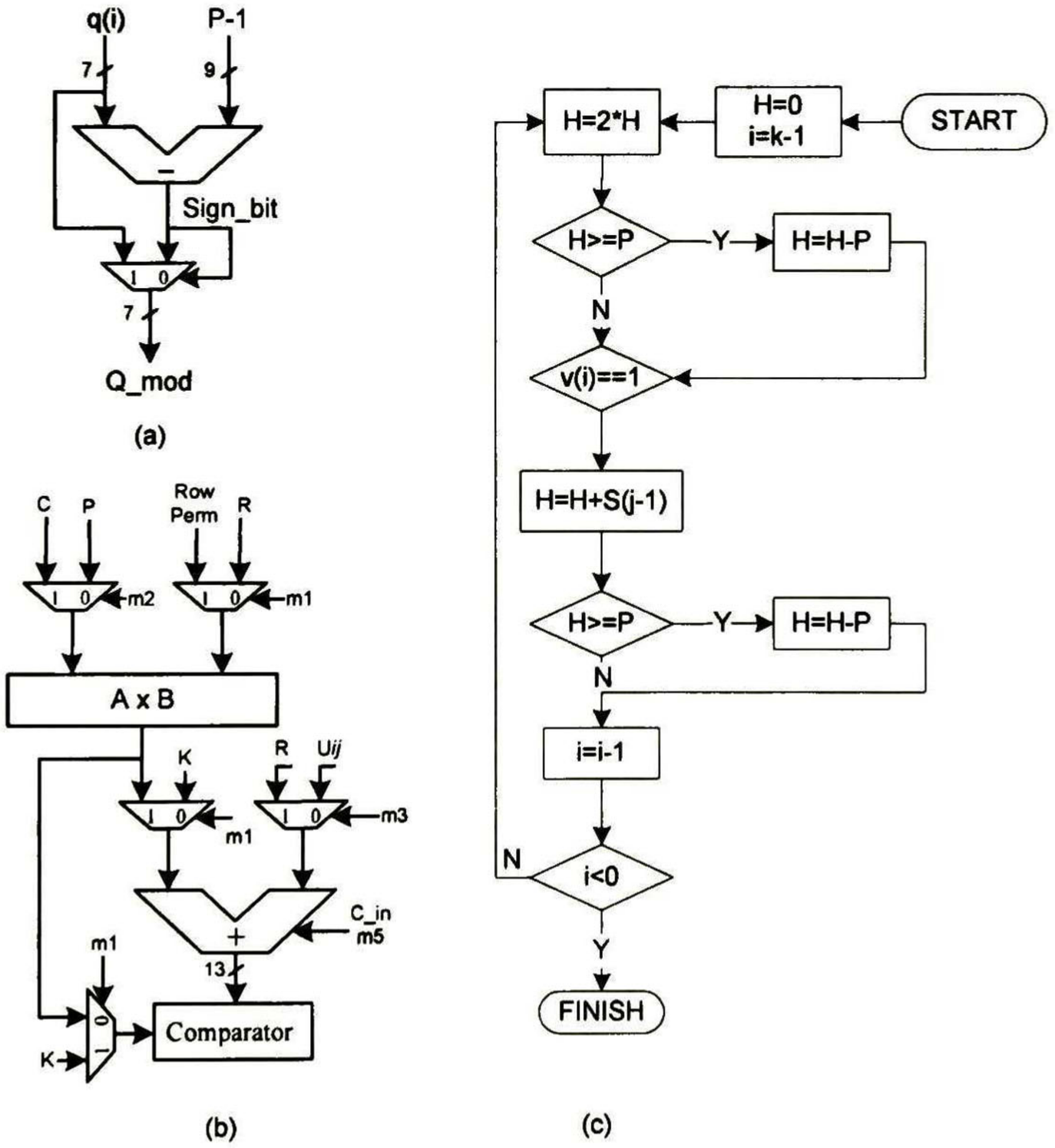

Figura 3.8. (a) Cálculo de "q(i) mod (p-1)", (b) Hardware multiplexable para la suma, multiplicación y comparación, (c) Diagrama de flujo para el algoritmo de multiplicación y módulo.

Los parámetros "p" y "v" se almacenan de forma combinada en una "Look Up Table". La "Look Up Table" se direcciona a través de un contador generado por el controlador. Con cada incremento en el contador se hace la comparación (P*R ≥K-R) para encontrar el valor adecuado de "p" y "v". Una vez que "p" es encontrado, "C" solo puede tener tres valores, p-1,p ó p+1, por lo que "C" es encontrado en máximo tres ciclos de reloj después de encontrar "p". En caso de que C=p+1, la condición R*C=K es comparada y una bandera apropiada para el caso especial es generada.

Para el cálculo de "p" y "C", la multiplicación, adición y comparación son requeridas. El bloque mostrado en la Figura 3.8(b) es capaz de realizar estas operaciones y es multiplexado para trabajar en el modo pre-computo y en modo "run-time".

El cálculo de la trayectoria de permutación "intra-row" también requiere el cálculo de la operación módulo. La operación módulo se calcula de forma iterativa utilizando el algoritmo de multiplicación y módulo mostrado en [módulo iterativo]. La operación de módulo requerida aquí es [S(j-1)*v mod p], donde dado que "v" se puede representar con 5 bits, un máximo de 5 iteraciones son necesarias para calcular una operación de multiplicación con módulo. El algoritmo para calcular las operaciones de multiplicación y módulo se muestra en la Figura 3.8(c).

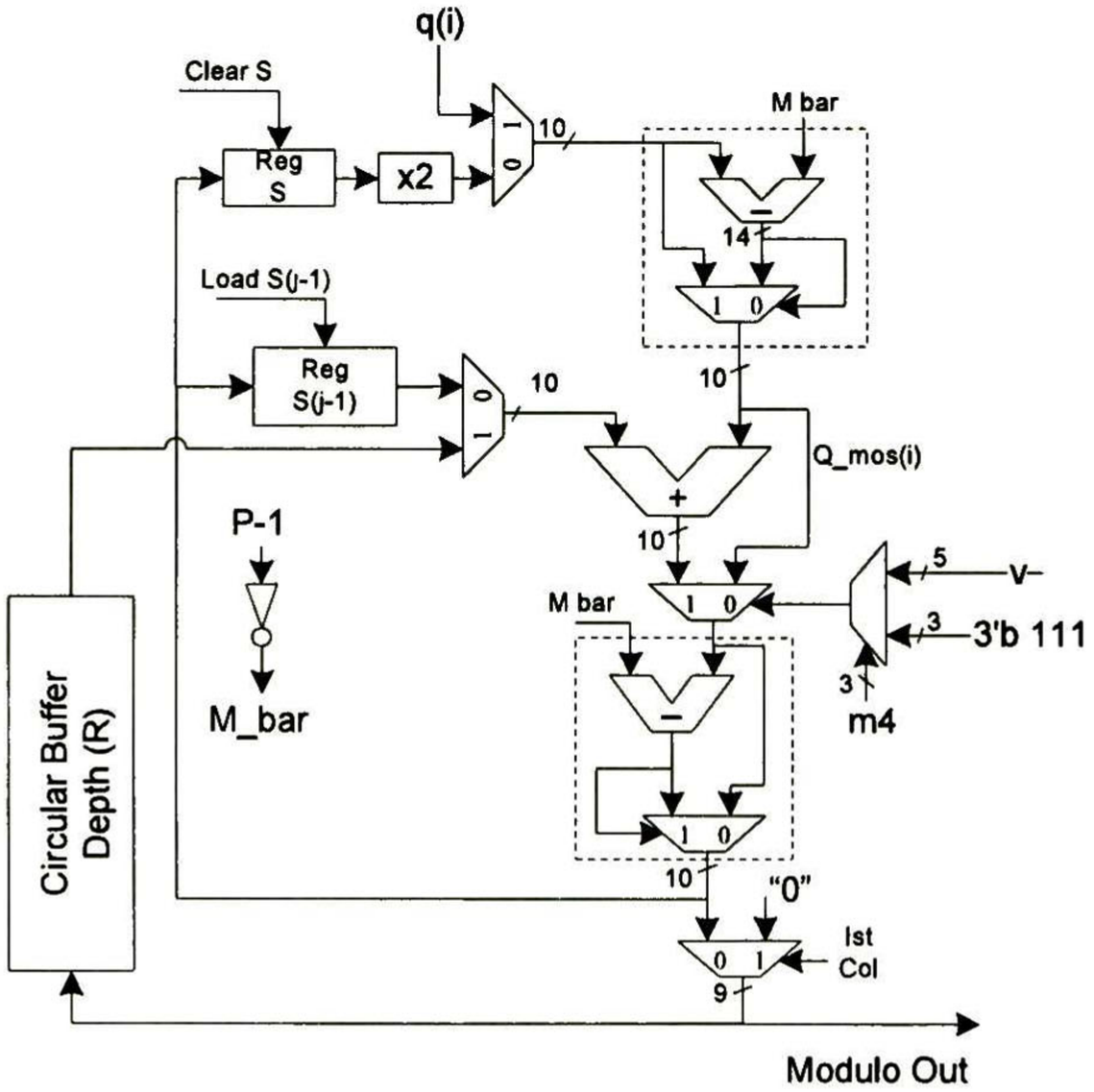

Figura 3.9. Hardware para calcular S(j) y direcciones para la lectura de S(j) RAM.

El hardware para el cálculo de S(j) se muestra en la Figura 3.9. El multiplexado de hardware se hace de tal forma que el primer sumador también es usado para calcular "q mod (p-

1)" como se muestra en la Figura 3.8(a), mientras los otros 2 sumadores se usan en el modo "run-time" para calcular la dirección de la RAM de forma recursiva. Durante la fase de pre-computo la dirección de escritura de la RAM es generada por el controlador. La señal "Modulo Out" es utilizada para entregar datos a la RAM durante la fase de pre-computo, y durante la fase "run-time" entrega direcciones de lectura para la RAM. Después de calcular los valores de S(j), estos son almacenados en una RAM de 256*6 bits. El uso de esta RAM depende del parámetro "p" y se llena hasta (p-2) localidades empezando por cero.

El controlador se encarga de configurar el hardware mostrado en la Figura 3.8(b) para la multiplicación, suma y comparación, así como también el hardware necesario para la operación módulo mostrado en la Figura 3.8(c).

El diagrama de estados del controlador se muestra en la Figura 3.10, cuando el controlador está en modo "run-time", una nueva pre-computación puede ser iniciada cambiando el tamaño del bloque "K" y mandando un pulso llamado "Changue_block_size".

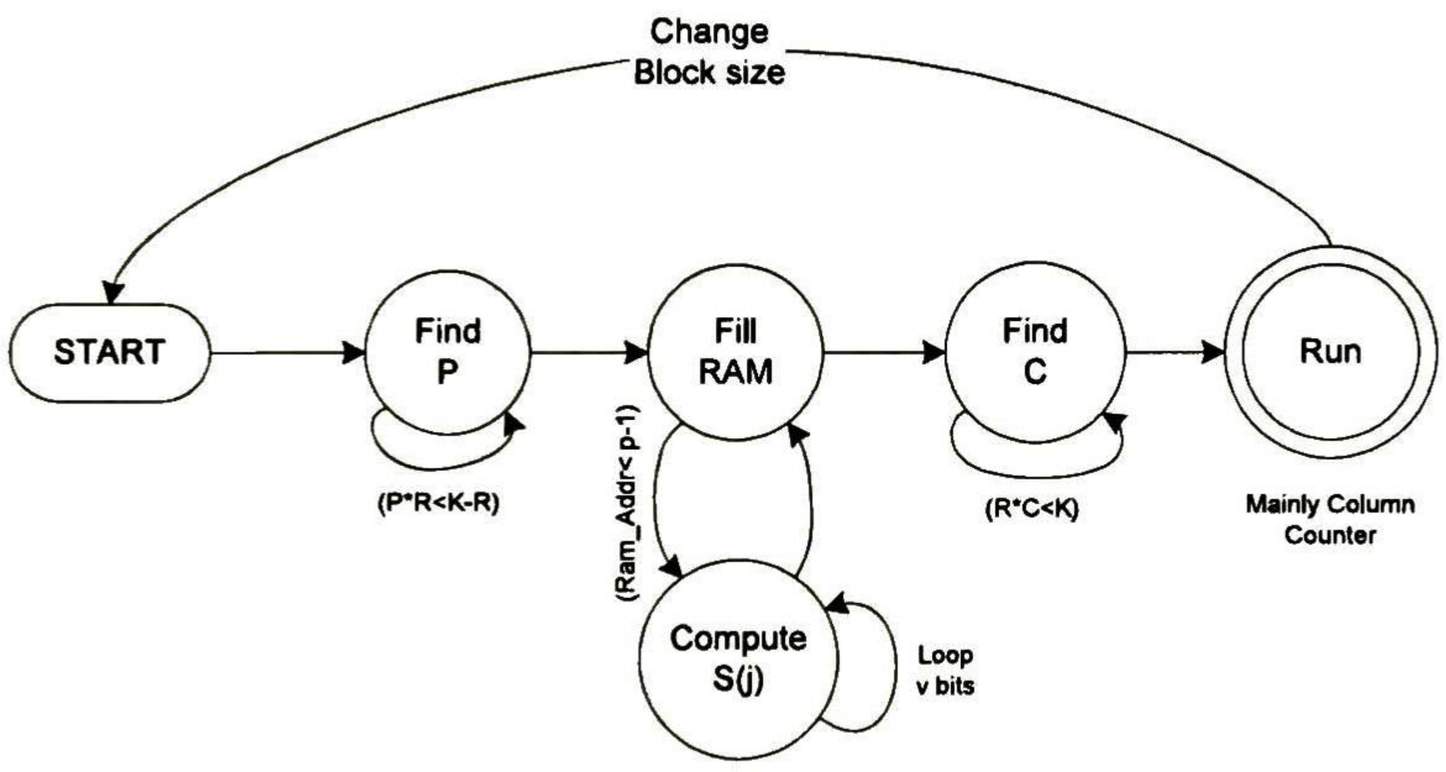

Figura 3.10. Controlador para el hardware del Interleaver 3G.

3.3.2 Run-Time

Después de completar la fase de "pre-computation", el controlador pasa al estado "RUN" y el hardware es configurado para realizar los cálculos de este modo. Las direcciones de la RAM son obtenidas usando el hardware mostrado en la Figura 3.9.

El diagrama de flujo para el Interleaver 3G se muestra en la Figura 3.7(b) y el hardware completo en la Figura 3.11. Un buffer circular es incluido para soportar el cálculo recursivo de las direcciones de la RAM.

Las direcciones de la RAM se calculan mediante la la ecuación recursiva (3.6).

$$Ram\_Adr(i,j) = [Ram\_Adr(i,j-1) + Q\_\mathrm{mod}(i)] \bmod (p-1) \tag{3.6}$$

Después del cálculo de la dirección de la RAM, la dirección final de entrelazado es calculada mediante la ecuación (3.7) que está basada en multiplicación y suma.

$$i\_addr = (C \times Row\_Perm) + U_{(i,j)} \tag{3.7}$$

El hardware utilizado en la fase de pre-computo, se utiliza también en el cálculo de la dirección de entrelazado final en modo "run-time". Cuando el tamaño de bloque no es exactamente R*C algunas direcciones son etiquetadas como invalidas.

También existen algunas excepciones en el algoritmo que se listan a continuación:

$$\begin{aligned} &if\,(C = p) \quad : \quad U_{i,(p-1)} = 0; \\ &if\,(C = p+1): \quad U_{i,(p-1)} = 0;\ U_{i,p} = p;\ and \\ &\qquad\qquad if\,(K = R \times C)\ then \\ &\qquad\qquad exchange\ U(R-1,0)\ with\ U(R-1,p) \end{aligned}$$

Existen algunas banderas relacionadas a la primera y última fila y columna, usando estas banderas junto con un par de multiplexores se puede obtener un bloque que trate con excepciones.

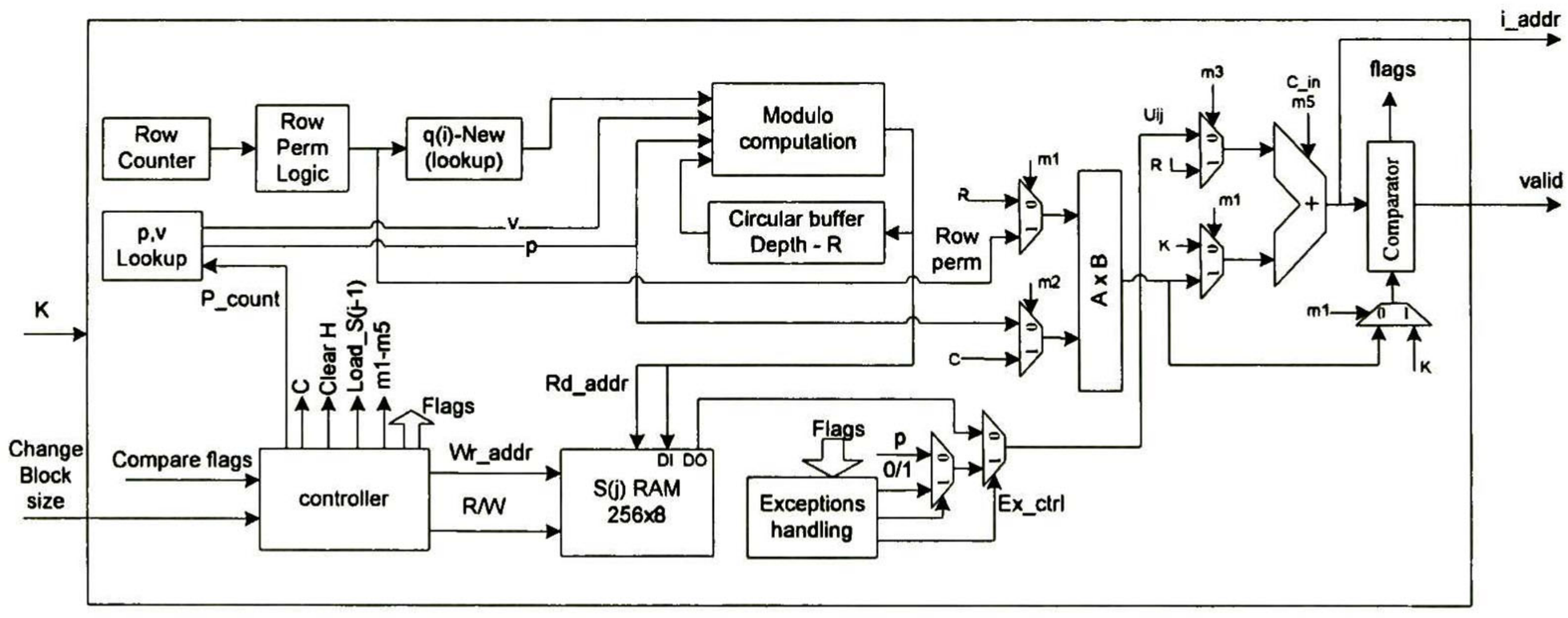

Figura 3.11. Hardware completo para el Interleaver 3G.

3.4 Interleaver/Deinterleaver Reconfigurable propuesto para los estándares 3GPP-WCDMA y 3GPP-LTE

Las arquitecturas anteriormente mencionadas tienen algunas deficiencias, sin embargo cuentan con el acierto de aprovechar algunos avances existentes basadas en reducciones

algorítmicas, métodos computacionales, multiplexaje de hardware y realización de cálculos de forma iterativa.

Nuestro diseño retoma algunas de estas ideas y utiliza algunos bloques propuestos por arquitecturas ya existentes para mejorarlos en algunos aspectos importantes que serán mencionados detalladamente más adelante.

Una gran diferencia de nuestra arquitectura con las arquitecturas existentes conocidas, es el hecho de que a diferencia de las otras arquitecturas, la nuestra es capaz de manejar el flujo de datos a través de todo el proceso de entrelazado y de-entrelazado tanto para el estándar 3GPP-W CDMA como para el 3GPP-LTE. Otra diferencia de nuestro diseño con la mayoría de los existentes para el estándar 3GPP-W CDMA es el manejo de las excepciones que el estándar presenta.

En este diseño presentamos una arquitectura reconfigurable para un Interleaver/Deinterleaver que podrá ser utilizado en los turbo códigos por los estándares 3GPP-W CDMA y 3GPP-LTE. Este Interleaver está diseñado bajo la idea del cálculo de la operación módulo de forma iterativa presentado en el apéndice C. Otra característica a favor de la presente arquitectura es que gran parte del hardware utilizado para generar las direcciones de entrelazado para el estándar 3GPP-W CDMA también es utilizado para el estándar 3GPP-LTE.

El Interleaver presentado aquí debe ser capaz de soportar turbo codificación de datos a tasas de entre 384 kbps y 2Mbpds para el caso del estándar 3GPP-W CDMA, y tasas de entre 50Mbps y de poco más de 300Mbps para el estándar 3GPP-LTE. El principal objetivo de este trabajo es obtener una arquitectura capaz de manejar los más de 5000 diferentes tamaños de bloques de ambos estándares y aun así mantener una baja complejidad y ahorrar tanto hardware como nos sea posible.

Para tener una arquitectura capaz de trabajar con ambos estándares (3GPP-W CDMA y 3GPP-LTE) partimos de obtener la arquitectura para el algoritmo 3GPP-W CDMA la cual es más compleja y de la cual posteriormente se podrá rehusar parte del hardware para el estándar 3GPP-LTE.

3.4.1 Arquitectura para el estándar 3GPP-W CDMA

Basándonos en el algoritmo 3GPP-W CDMA partimos de la idea de la formación de una matriz rectangular (R x C), la cual se rellena con los datos de entrada (renglón a renglón) y posteriormente se realizan permutaciones entre filas y dentro de ellas para después leer los datos ya procesados. Analizando este algoritmo nos damos cuenta de que es necesario llevar acabo diferente tipo de operaciones a fin de obtener las trayectorias de entrelazado necesarias; algunas de esas operaciones son las siguientes:

- Comparaciones
- Adiciones

- Multiplicaciones
- Permutaciones
- Operaciones módulo
- Calculo del M.C.D (máximo común divisor)

De la misma forma que lo hicieron algunas otras arquitecturas y para facilitar el proceso de entrelazado, separamos el algoritmo en dos modos.

El primer modo, llamado modo de "pre-calculo", tiene que ser llevado a cabo en cada ocasión que un cambio en el tamaño del bloque de datos de entrada, K, se presente. Cuando el tamaño del bloque de datos de entrada permanece fijo, la etapa de "pre-calculo" solo se necesita una sola vez, incluso si es necesario el entrelazado de varios bloques. En este modo se calculan los parámetros necesarios para obtener las direcciones finales de entrelazado, estos parámetros son:

El número de renglones (R), número de columnas (C), número entero primo (p), la raíz primitiva (v), la secuencia base para las permutaciones dentro de cada renglón S(j) (llamadas "intra-row" en el estándar), la secuencia de mínimos enteros primos (q_i) y la secuencia de enteros primos permutados (r_i).

El segundo modo, es el llamado modo de "calculo", en este modo se calcula la secuencia de permutación "inter-row", U_{ij}, y la dirección de entrelazado *i_addr*, que que utilizamos para leer o escribir en una RAM de datos dependiendo de si estamos entrelazando o de-entrelazando datos. Los cálculos que en este modo se hacen, son necesarios cada vez que se requiera procesar (entrelazar o de-entrelazar) un nuevo bloque de datos, incluso si el tamaño de este permanece constante.

En lugar de generar las direcciones de entrelazado para cada bloque de tamaño igual, estas se podrían guardar en alguna RAM y después solamente leerlas, sin embargo la aproximación presentada en este trabajo a base de estos dos modos y el cálculo de direcciones en el modo de "cálculo" es lo que permite reducir el tamaño de la implementación del Interleaver en comparación con otras aproximaciones basadas en "Look Up Tables", donde las direcciones de entrelazado son guardadas en memorias.

A continuación se hace una descripción más detallada de cada uno de los dos modos en que opera el Interleaver para este estándar y junto con cada descripción se presenta su arquitectura relacionada.

3.4.1.1 Modo de "Pre-cálculo"

En este modo primeramente se calcula el número de renglones "R" mediante el uso de algunas funciones lógicas sencillas dependientes solo del tamaño del bloque K. Los parámetros "p" y "v" se almacenan en pares en una LUT, tal y como se aprecia en el estándar, de esta manera

podemos después leer esta LUT a través de un contador generado por el controlador basados en la ecuación (3.8).

$$K \leq R \times (p+1) \rightarrow (K-R) \leq (R \times p) \tag{3.8}$$

Para realizar las operaciones necesarias en la ecuación (3.8), utilizamos la arquitectura presentada en [16] y mostrada en la Figura 3.12.

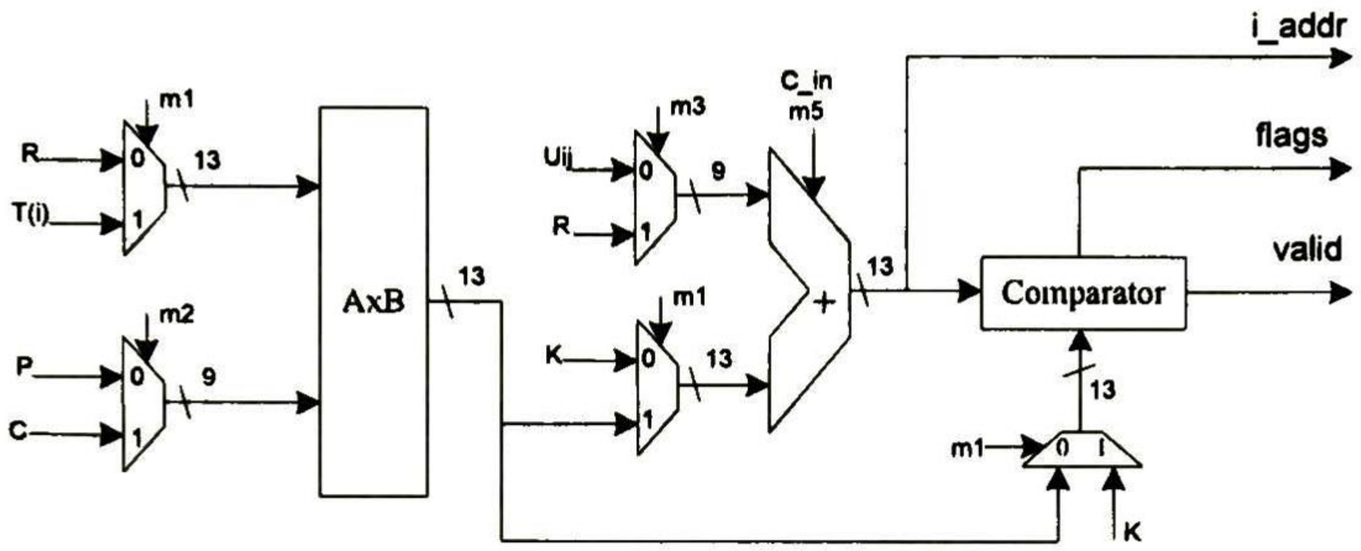

Figura 3.12. Arquitectura del hardware para multiplicaciones, sumas y comparaciones.

La arquitectura de la Figura 3.12. Es capaz de realizar multiplicación, suma y comparaciones, este bloque es controlado y configurado a través de las banderas m1-m5 generadas por el bloque del controlador, el cual se muestra en la Figura 3.18 de la arquitectura completa.

Luego de que "p" y "v" son obtenidos, el siguiente paso es obtener el numero de columnas (C). Del estándar podemos saber que "C" solo puede tomar uno de tres diferentes valores, que son p-1,p ó p+1, y que además debe ser el mínimo valor que cumpla con la condición (R x C)≥K. Entonces para obtener el valor adecuado de "C", el controlador genera C=p-1 y configura la arquitectura mostrada en la Figura 3.12 para hacer las comparaciones necesarias, si no cumple con dichas condiciones incrementa en 1 el valor de "C" y nuevamente vuelve a comparar, si aun sigue sin ser el valor adecuado, entonces de forma automática se hace la asignación C=p+1.

De acuerdo al algoritmo, es necesario calcular la secuencia de enteros primos "q_i", sin embargo al generar todas las posibles secuencias q_i mediante simulaciones en Matlab, nos percatamos de que aproximadamente la mitad de los diferentes tamaños de bloque (K), tienen secuencias q_i relacionadas que son idénticas, y el resto de los tamaños de bloque tienen secuencias q_i relacionadas con solo una elemento diferente a otras secuencias. Solo una secuencia q_i cuenta con dos elementos diferentes a las otras secuencias.

Basándonos en las anteriores observaciones hechas sobre las secuencias q_i fue que decidimos almacenar todos los posibles valores de qi en una memoria ROM llamada "q_iLUT", y posteriormente dividirlos en subgrupos que serán leídos según el tamaño de bloque con que se esté trabajando y el valor del parámetro "p"

Para calcular la secuencia base para la permutación dentro de cada renglón "S(j)" , nos basamos en la ecuación (3.9).

$$S(j) = (v \times S(j-1)) \bmod p \tag{3.9}$$

De donde podemos notar que es necesario llevar a cabo operaciones de módulo; afortunadamente, en el articulo presentado en el apéndice C se muestra un algoritmo para calcular la operación módulo para expresiones de la forma *"AxB mod M"* en forma simplificada.

Este algoritmo iterativo se basa principalmente en adiciones, corrimientos, comparaciones y extracciones de bits.

Utilizando este algoritmo simplificado y basándonos en el diagrama de flujo mostrado en el apéndice C, podemos obtener su diagrama de flujo adaptado a nuestras necesidades, asi como el hardware que implementa dicho algoritmo, tal y como se presenta en la Figura 3.13.

Con la ayuda de esta arquitectura podemos calcular y escribir en RAM cada elemento de la secuencia S(j) en máximo cuatro ciclos de reloj.

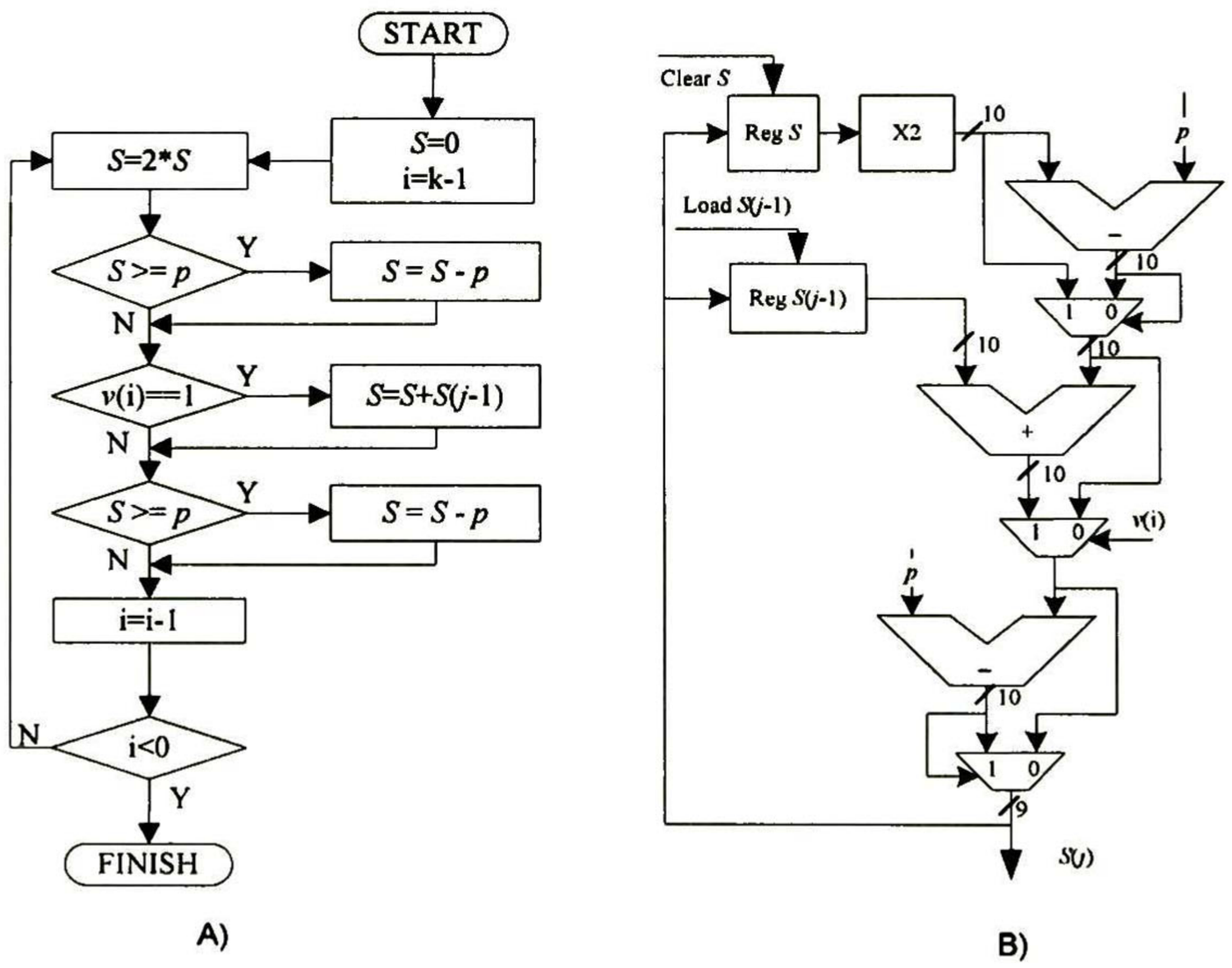

Figura 3.13. Diagrama de flujo A) y arquitectura B) para el cálculo de (v x S(j-1)) mod p.

Analizando el algoritmo presentado en el apéndice C, podemos notar que al aplicar dicho algoritmo a la expresión del la ecuación (3.9), el número de iteraciones necesarias para obtener cada resultado parcial S(j), dependerá del valor de "v". Según el estándar, "v" solo puede tomar seis diferentes valores, 2,3,5,6,7 y 19; Relacionando esto con el algoritmo del apéndice C, tenemos que el número de iteraciones para obtener cada S(j) será de 1 iteración cuando v=2 y v=3, de 2

iteraciones cuando v=5, v=6 y v=7 y solo necesitaremos 4 iteraciones cuando v=19. En este caso cada iteración se lleva a cabo en un ciclo de reloj.

3.4.1.2 Modo de cálculo

En el modo de "calculo" es donde se llevan a cabo las permutaciones entre filas T(i) y entre los elementos de cada una de ellas U_{ij}. El orden de las permutaciones entre filas, T(i), están almacenadas en una "Look Up Table", mientras que el orden de permutación entre los elementos de cada fila están almacenados en una memoria llamada "S(j)RAM". Mediante la obtención de estos parámetros y el uso de la ecuación (3.10), las direcciones de entrelazado pueden finalmente ser generadas.

$$i\_addr((i \times C) + j) = (C \times T(i)) + U_{i,j} \tag{3.10}$$

Donde i=0,1,...,R-1; j=0,1,...,C-1.

Entonces para encontrar las direcciones de entrelazado, según (3.10) tenemos que llevar a cabo algunas multiplicaciones y sumas. Primero multiplicamos C x T(i), lo que nos da por resultado una dirección que nos apunta al primer elemento de cada renglón, R, de la matriz rectangular, en seguida hacemos la suma (C x T(i))+ $U_{i,j}$ lo que nos da como resultado un desplazamiento en cada renglón de la matriz rectangular. Estas sumas y multiplicaciones se llevan a cabo mediante el uso de la arquitectura mostrada en la Figura 3.12, y que previamente se usó en la etapa de "pre-calculo". Un ejemplo ilustrativo de lo que se acaba de mencionar se presenta en la Figura 3.14.

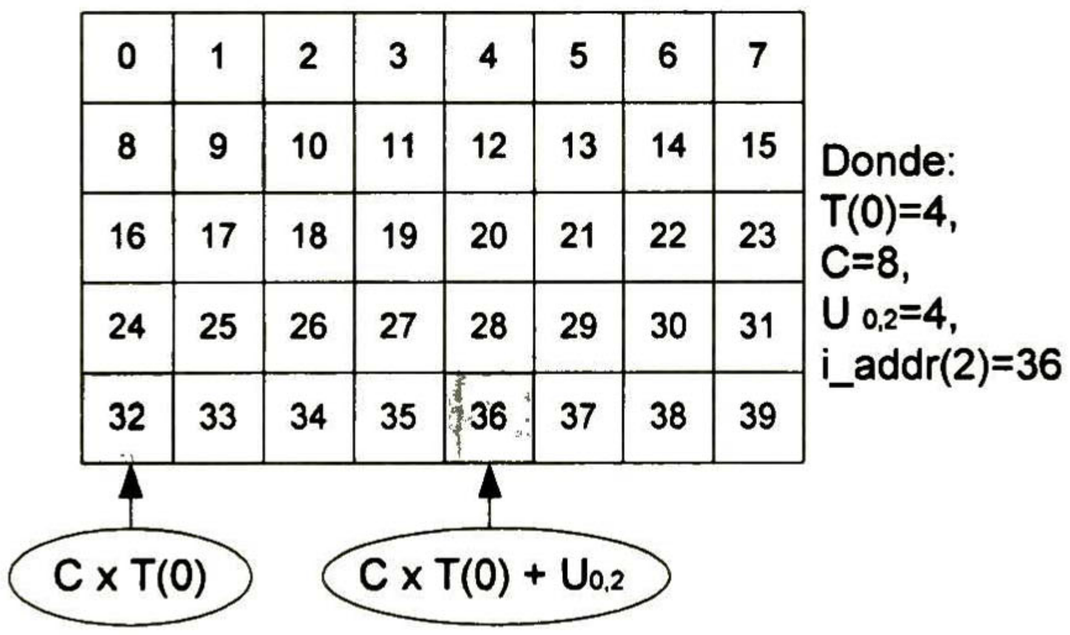

Figura 3.14. Ejemplo del cálculo de i_addr.

Como podemos notar, para obtener las direcciones de entrelazado debemos conocer previamente U_{ij}. Para calcular U_{ij} en modo de "cálculo" debemos realizar la operación "(j x r_i) mod (p-1)" la cual tiene la misma forma que la expresión "(v x S(j-1))mod p" calculada en el modo de "pre-calculo" Debido a las similitudes mencionadas y a que ambas operaciones se realizan en diferentes tiempos es que podemos reutilizar el hardware mostrado en la Figura 3.13 para realizar ambas operaciones con solo hacer algunas modificaciones.

En la sección 3.3 se hicieron algunas modificaciones de donde se obtuvo la arquitectura de la Figura 3.9 donde podemos notar que un buffer circular, usado para las operaciones iterativas, y un multiplexor para habilitar dicho buffer en el modo de cálculo fueron añadidos. En la arquitectura mostrada en la Figura 3.9 también fue agregado un multiplexor a través del cual q_i es alimentado durante el modo de cálculo. La arquitectura de la Figura 3.9 es capaz de realizar las operaciones de módulo basándose en substracciones, lo cual se debe en gran medida al bloque encerrado por líneas punteadas en la misma figura, pero aunque dicha arquitectura funciona adecuadamente en la etapa de pre-calculo, en la etapa de cálculo falla en alrededor del 12% de los casos, esto ocurre cuando q(i)>2P-1. En nuestra arquitectura resolvimos este problema modificando el bloque encerrado de la Figura 3.9, pasando del bloque mostrado en la Figura 3.15(a) al bloque de 3.15(b), el cual puede realizar todas las restas necesarias para la obtención de la operación módulo necesarias para generar las direcciones de entrelazado requeridas por el estándar.

Entonces substituyendo 3.15(a) con 3.15 (b) en la Figura 3.9, obtenemos la arquitectura mostrada en la Figura 3.16 que trabaja adecuadamente en ambos modos, pre-calculo y cálculo.

Mediante la arquitectura de la Figura 3.16 podemos finalmente obtener la dirección de la "S(j)RAM" (para leer U_{ij}) según la ecuación (3.11).

$$Ram\_Adr(i,j) = [Ram\_Adr(i,j-1) + Q\_\mathrm{mod}(i)] \mod (p-1) \qquad (3.11)$$

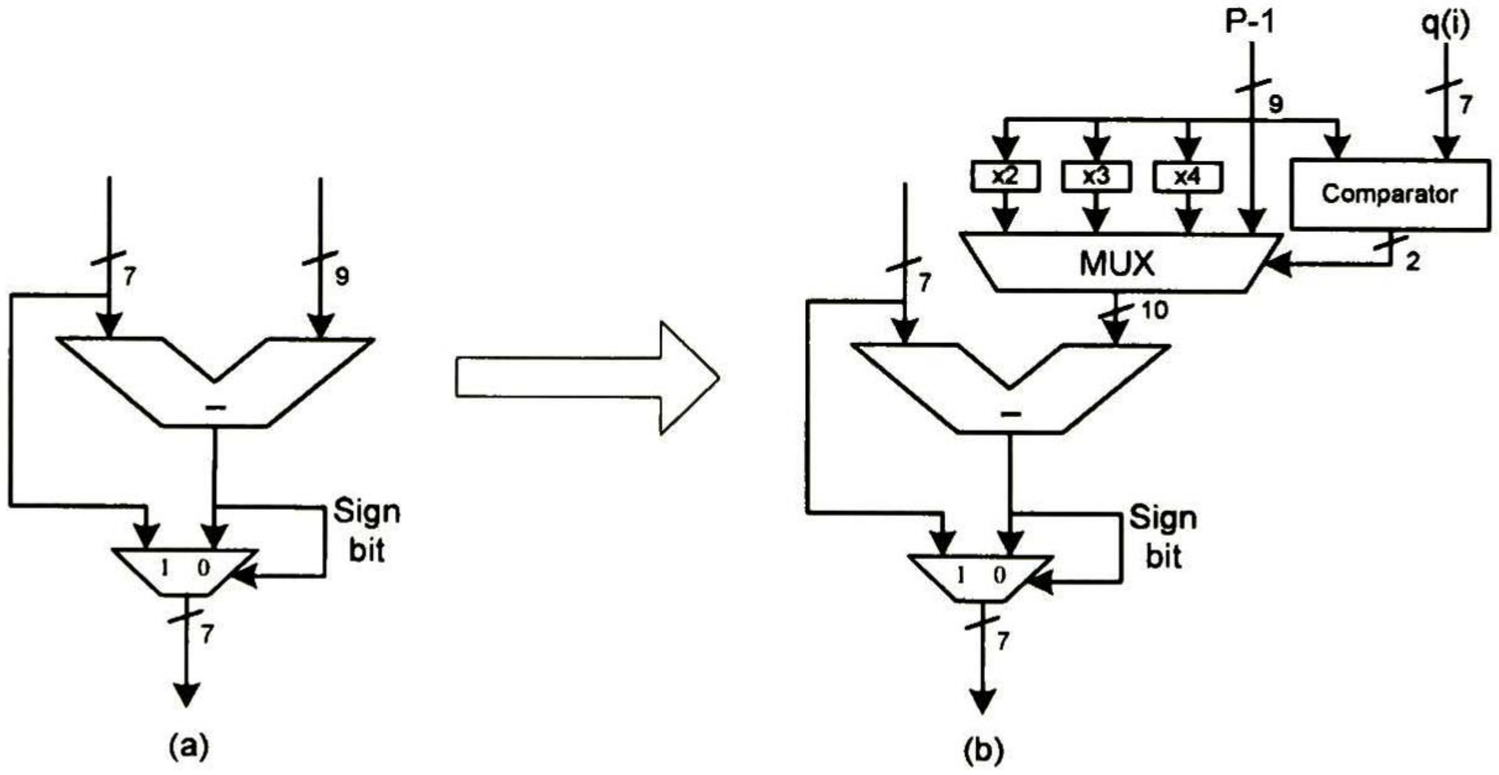

Figura 3.15. (a) Arquitectura del hardware usado en [16] para obtener Q_mod(i)=q(i)mod(p-1), (b) Arquitectura modificada.

Analizando varias arquitecturas de interleavers para este estándar nos dimos cuenta de que ninguna de estas se ocupaba del manejo de datos en el entrelazado, sino solo de generar direcciones. Normalmente esto no sería ningún problema únicamente generar las direcciones de entrelazado y de manera externa utilizarlas de forma continua para el entrelazado de datos, pero

para el estándar 3GPP-W CDMA existen algunas excepciones mientras se generan las direcciones de entrelazado que no nos permiten, de forma normal, entregar las direcciones de entrelazado de forma continua. Para resolver este problema, en el presente diseño agregamos algunos bloques para el manejo de datos siendo los más importantes una RAM de datos (dataRAM) y bloque para el manejo de excepciones (Exc Handling).

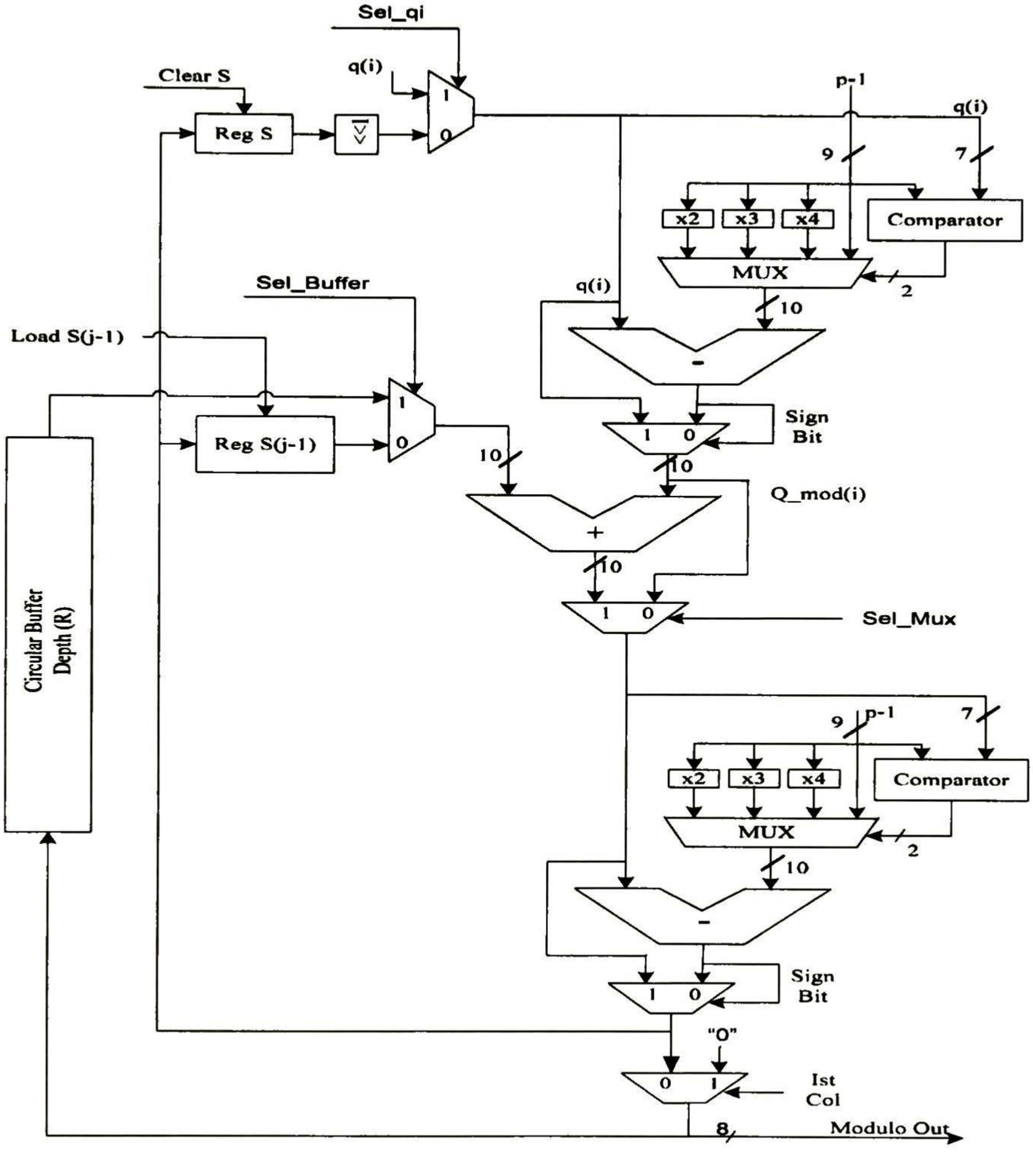

Figura 3.16. Arquitectura obtenida a partir de la Figura 3.9 para obtener el cálculo correcto de S(j) y las direcciones de lectura para la S(j)RAM.

3.4.1.3 Arquitectura propuesta y comparación con otras arquitecturas

En el algoritmo para la generación de direcciones de entrelazado del estándar 3GPP-W CDMA existen algunas excepciones, las cuales son tratadas por nuestra arquitectura con el bloque de excepciones mostrado en la arquitectura de la Figura 3.18.

Existen 3 tipos de excepciones de las que nos debemos de ocupar; la primera ocurre cuando el tamaño de bloque de datos K< R*C, por lo que se generarán algunas direcciones mayores a K que serán etiquetadas como invalidas. Por ejemplo, cuando K=41 la matriz rectangular tendrá un tamaño de 5x10 y por tanto habrá 9 direcciones invalidas. Debido a lo anterior no podremos generar una dirección valida por ciclo de reloj (ver Tabla 3.1).

Tabla 3.1. Promedio de generación de direcciones de entrelazado para diferentes tamaños de bloques.

Tamaño del bloque (K)	Ciclos de reloj totales en modo de calculo	Promedio de direcciones válidas por ciclo
40	40	1.00
41	50	0.82
2041	2060	0.99
4241	4440	0.95
4840	4840	1.00
5114	5129	0.99

El segundo tipo de excepción ocurre cuando C=p+1, en este caso nos salimos de la secuencia de generación de direcciones y hacemos la asignación $U_{i,(p-1)}=0$ y $U_{i,p}=p$.

El tercer tipo de excepción ocurre cuando las condiciones C=p+1 y K=RxC se cumplen; en este caso además de hacer las asignaciones $U_{i,(p-1)}=0$ y $U_{i,p}=p$, también intercambiamos $U_{(R-1,0)}$ con $U_{(R-1,p)}$.

Otro bloque muy importante en nuestra arquitectura, es el bloque "Modulo Computation" que además de encargarse de llevar a cabo las operaciones de módulo en ambos modos, también escribe la secuencia S(j) en la memoria "S(j)RAM" Conectado a este bloque existe el bloque "Buffer Circular" que se utiliza en el modo de cálculo para soportar operaciones recursivas.

Con la finalidad de sincronizar cada bloque de la arquitectura el bloque del controlador genera las señales de control necesarias, el diagrama de la máquina de estados de este bloque se muestra en la Figura 3.17.

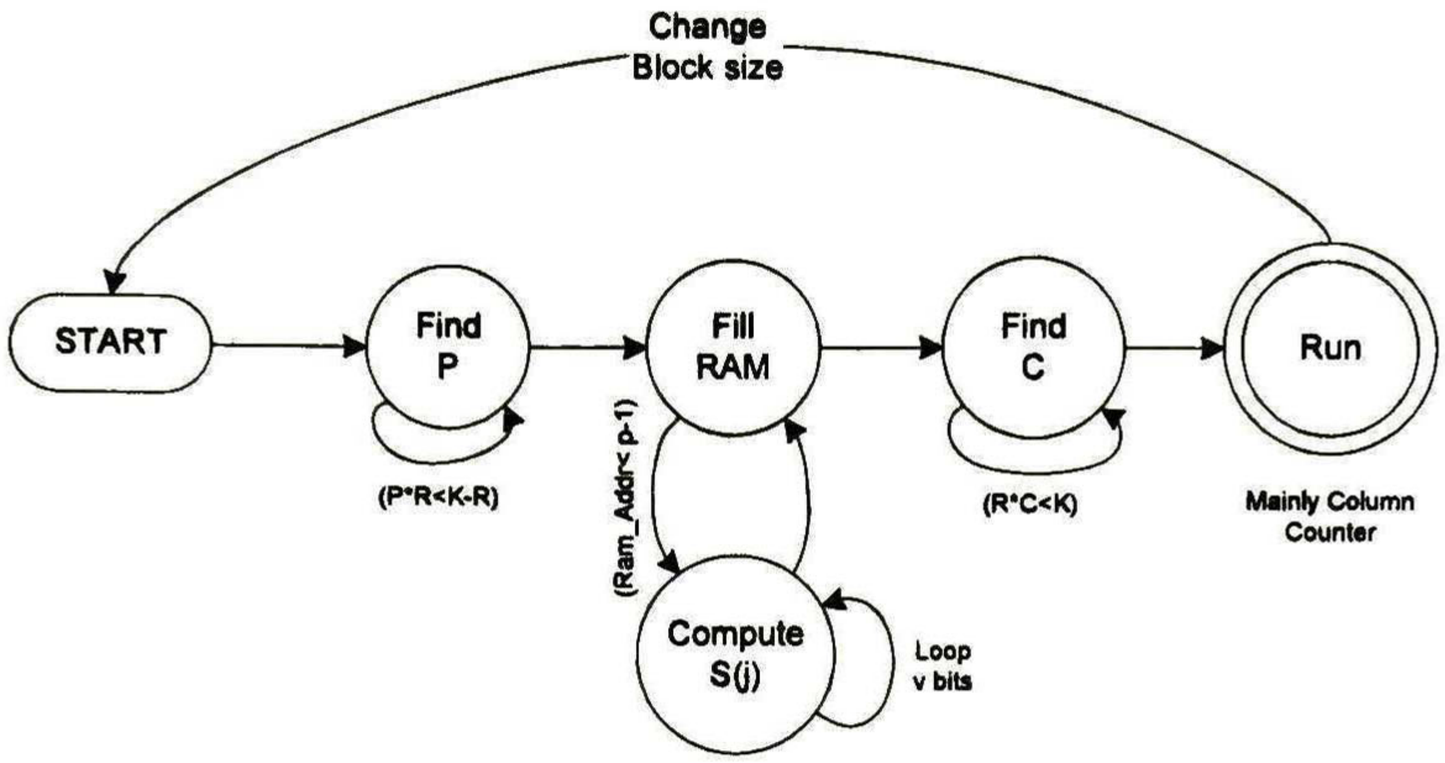

Figura 3.17. Maquina de estados del controlador para el cálculo de las direcciones de entrelazado.

Mientras la mayoría de los interleavers solo generan direcciones de entrelazado y una bandera indicando si la dirección es válida o no, nuestra arquitectura permite calcular las direcciones de entrelazado además de ocuparse de entrelazado de datos. Por ejemplo, si recibimos un bloque de datos de entrada de cualquier valor de entre 40 y 5114, nuestro Interleaver toma este bloque y lo reordena de acuerdo a su trayectoria de entrelazado según el estándar. A la salida de nuestro bloque Interleaver tendremos una secuencia de datos ya procesada, incluso cuando alguna excepción se presente. A fin de poder hacer el manejo de datos nos auxiliamos de una RAM de datos que incrementa el tamaño de nuestro diseño, esto podría verse como una desventaja, pero en realidad el almacenamiento de datos es necesaria siempre que el entrelazado se lleva a cabo en Turbo Códigos.

Normalmente los interleavers usados con los turbo códigos trabajan varias veces con el mismo tamaño de bloque, K, antes de ser necesario un cambio del mismo. Cuando una operación de entrelazado es requerida el modo de "pre-calculo" es desarrollado, donde S(j) es calculado y guardado en RAM. En seguida el modo de "calculo" inicia y las direcciones de entrelazado se calculan continuamente mientras el valor de K no cambie. Solamente cuando K cambia el modo de "pre-calculo" es llevado a cabo otra vez.

Como se puede notar en la Figura 3.18. Que muestra la arquitectura final para este estándar, una memoria RAM extra fue añadida. La llamada "I/O RAM" es una memoria utilizada para manejo de datos cuando existe alguna excepción.

Cuando nuestra arquitectura se utiliza como "Interleaver" la memoria I/O RAM tiene la función de retener los datos de entrada antes de pasarlos a la memoria de datos "dataRAM", de esta forma si una dirección de escritura (i_addr) inválida es generada, el dato correspondiente se mantiene en la I/O RAM esperando una dirección válida de entrelazado, mientras los datos que

siguen llegando se hacen esperar en esta I/O RAM. Los datos de salida en modo Interleaver, se leen de forma consecutiva con la ayuda de un contador.

Se generarán direcciones inválidas de lectura, y por tanto se leerá basura. La memoria "I/O RAM" retiene los datos antes de sacarlos e ignora esta basura y espera nuevos datos válidos. Esta RAM puede verse como un buffer de salida que empieza a entregar los datos hasta que se llena.

Un retardo máximo de 199 ciclos de reloj puede ser generado debido a direcciones inválidas, y ese es precisamente el número de localidades de la I/O RAM.

Para poder utilizar las direcciones generadas "i_addr" para lectura o escritura de la "dataRAM" según el modo de funcionamiento, se utilizan algunos multiplexores que se muestran en la Figura 3.18.

3.4.2 Arquitectura para el estándar 3GPP-W CDMA y 3GPP-LTE

El esquema de turbo códigos que se adapta tanto para el estándar 3GPP-W CDMA como para el 3GPP-LTE es el de códigos concatenados en paralelo con dos codificadores de ocho estados cada uno. Los interleavers usados en cada estándar tienen diferente estructura, por lo que el principal reto es fusionar ambas estructuras en una sola arquitectura funcional y que sea de bajo costo en hardware.

Para lograr lo previamente mencionado partimos de la arquitectura que ya tenemos para el Interleaver del estándar 3GPP-W CDMA y mediante algunas modificaciones y rehusó de hardware llegaremos a una arquitectura final capaz de trabajar para ambos estándares.

Basándonos en el estándar 3GPP-LTE para turbo codificación, tenemos que la trayectoria de permutación para el Interleaver interno está dada por la ecuación (3.12).

$$\prod(X) = (f_1 \times X + f_2 \times X^2) \bmod K \qquad (3.12)$$

Donde x=0,1,2,...,(K-1), y K es el tamaño de bloque del Interleaver. La permutación polinomial mencionada en la expresión (3.12) representa un Interleaver deterministico para distintos tamaños de bloques y ciertos valores de f1 y f2. Los valores de f1 y f2 para cada tamaño de bloque se mencionan en el estándar 3GPP-LTE [12]. Los interleavers de permutaciones polinomiales cuadráticas se presentan en [17]. En [17] también se menciona que este tipo de interleavers no son tan buenos como los Interleavers pseudo aleatorios, pero son capaces de lograr comportamientos promedio de los interleavers aleatorios. Además estos interleavers tienen la ventaja de presentar representaciones relativamente compactas.

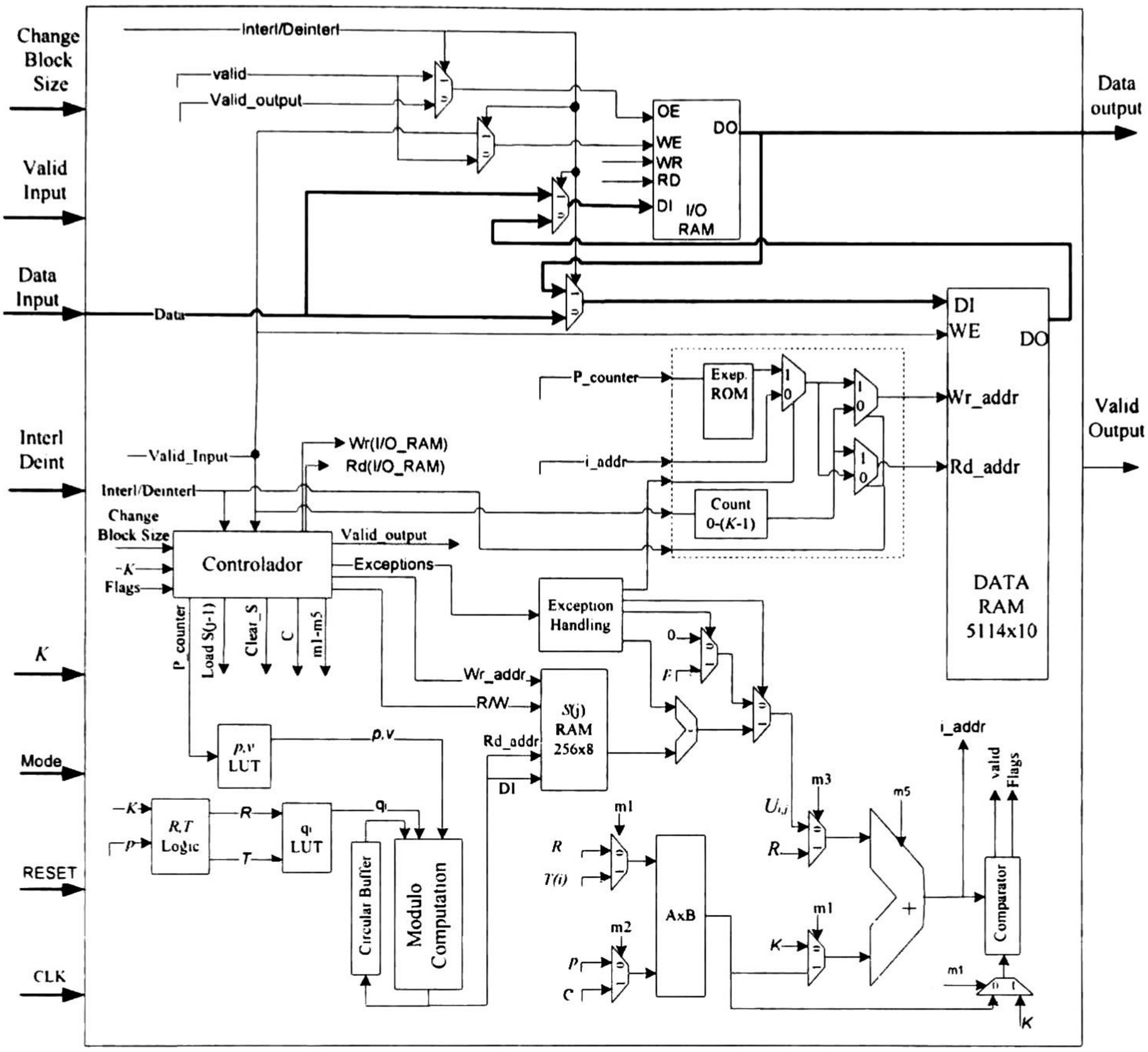

Figura 3.18. Arquitectura completa para el Interleaver/Deinterleaver para el estándar 3GPP-W CDMA.

Si tratamos de implementar directamente la permutación polinomial (3.12), nos daremos cuenta de que sería ineficiente en hardware debido a un par de multiplicaciones y a la compleja operación módulo. Además de estos problemas existe un detalle de aumento exponencial con $(f_1*x + f_2* x^2)$ al incrementar "x". Para poder resolver estos problemas se utilizan algunas simplificaciones a nivel algorítmico hechas en [17]. Con estas simplificaciones podemos reescribir la permutación polinomial (3.12) de tal forma que se pueda resolver de forma recursiva.

$$\begin{aligned} f(x+1) &= [f_1 \times (x+1) + f_2 \times (x+1)^2] \bmod K \\ f(x+1) &= [f(x) + (f_1 + 2f_2 x] \bmod K \\ f(x+1) &= [f(x) + g(x)] \bmod k \end{aligned} \tag{3.13}$$

Donde g(x) puede ser calculada nuevamente de forma recursiva utilizando las siguientes simplificaciones.

$$\begin{aligned} g(x+1) &= [f_1 + f_2 + 2f_2 \times (x+1)] \bmod K \\ g(x+1) &= [g(x) + 2f_2] \bmod K \end{aligned} \tag{3.14}$$

Con las anteriores simplificaciones algebraicas se puede hacer el cálculo de las permutaciones polinomiales cuadráticas mucho más eficientes en hardware.

Desde el punto de vista del hardware, se necesitan tener cargados los valores de g(0) y f2 para poder iniciar los cálculos. Utilizando los valores de f1 y f2 proporcionados en tablas del estándar podemos calcular g(0) para cualquier tamaño de bloque. Los términos g(0) y f2 son almacenados en LUTs la cual es direccionadas de acuerdo al tamaño del bloque. Utilizando las simplificaciones mencionadas en (3.13) y (3.14), los términos f(x+1) y g(x+1) pueden ser calculadas recursivamente mediante el uso de sumadores y comparadores.

Para el estándar 3GPP-LTE solo necesitamos una etapa de modo de cálculo y las direcciones se generan una por cada ciclo de reloj. Obteniendo el diagrama de flujo para el cálculo de las direcciones de entrelazado (i_addr) fácilmente podemos deducir su hardware relacionado, en la Figura 3.19 se muestran ambos, el diagrama de flujo de este algoritmo (a) y su arquitectura relacionada (b).

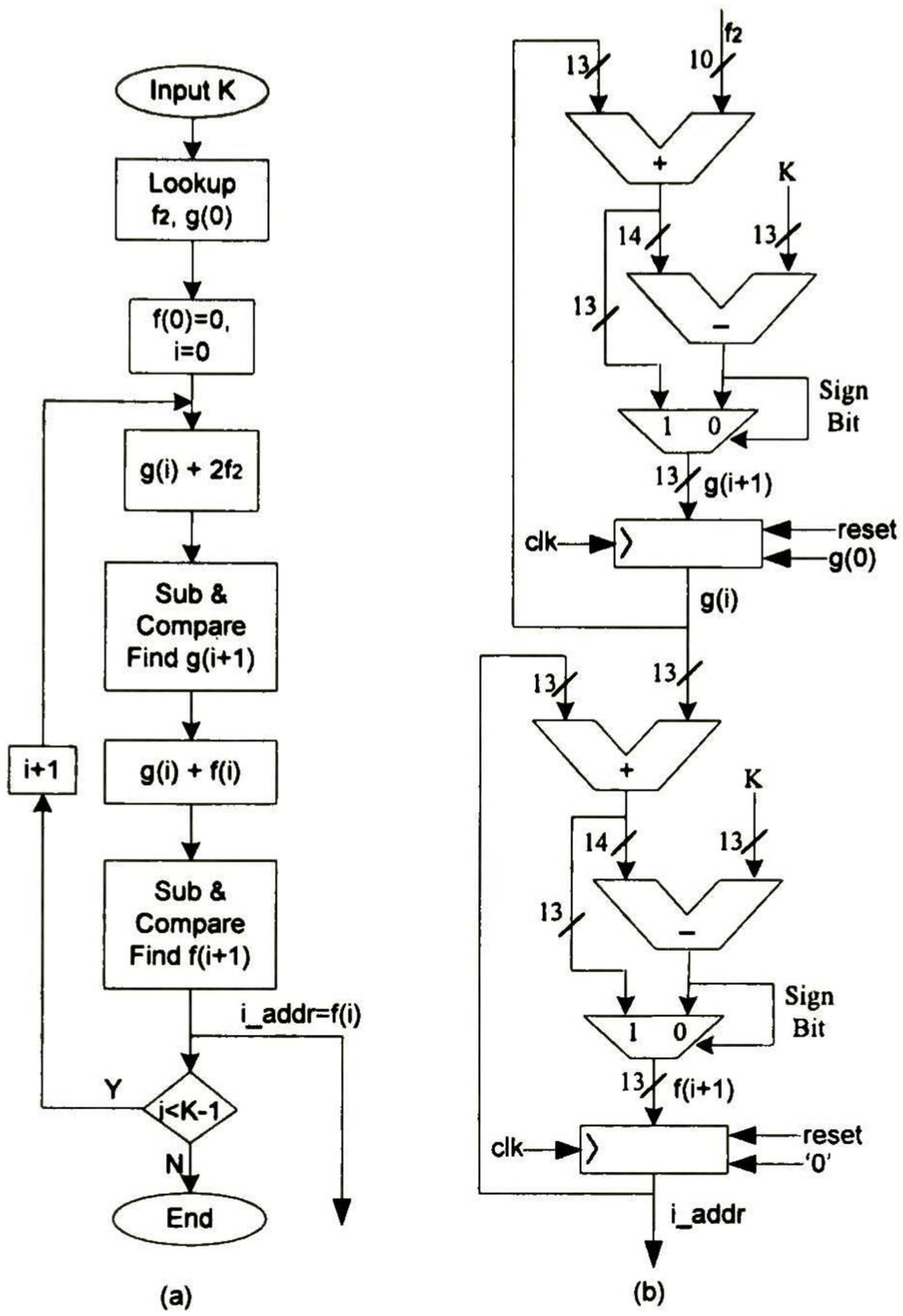

Figura 3.19. (a) Diagrama de flujo para el cálculo de "i_addr" para el estándar 3GPP-LTE, y (b) su arquitectura correspondiente.

Al analizar la arquitectura anterior para el cálculo de "i_addr" del estándar 3GPP-LTE nos damos cuenta que es una arquitectura mucho más sencilla que la necesaria para el entrelazado del estándar 3GPP-W CDMA, por lo que haciendo algunas modificaciones sencillas a las arquitecturas mostradas en las Figura 3.12 y Figura 3.16 podemos obtener la arquitectura mostrada en la Figura 3.20 donde se muestra una arquitectura capaz de generar las direcciones de entrelazado para ambos estándares 3GPP-W CDMA y 3GPP-LTE. En esta arquitectura los bloques sombreados son los utilizados para el estándar 3GPP-LTE.

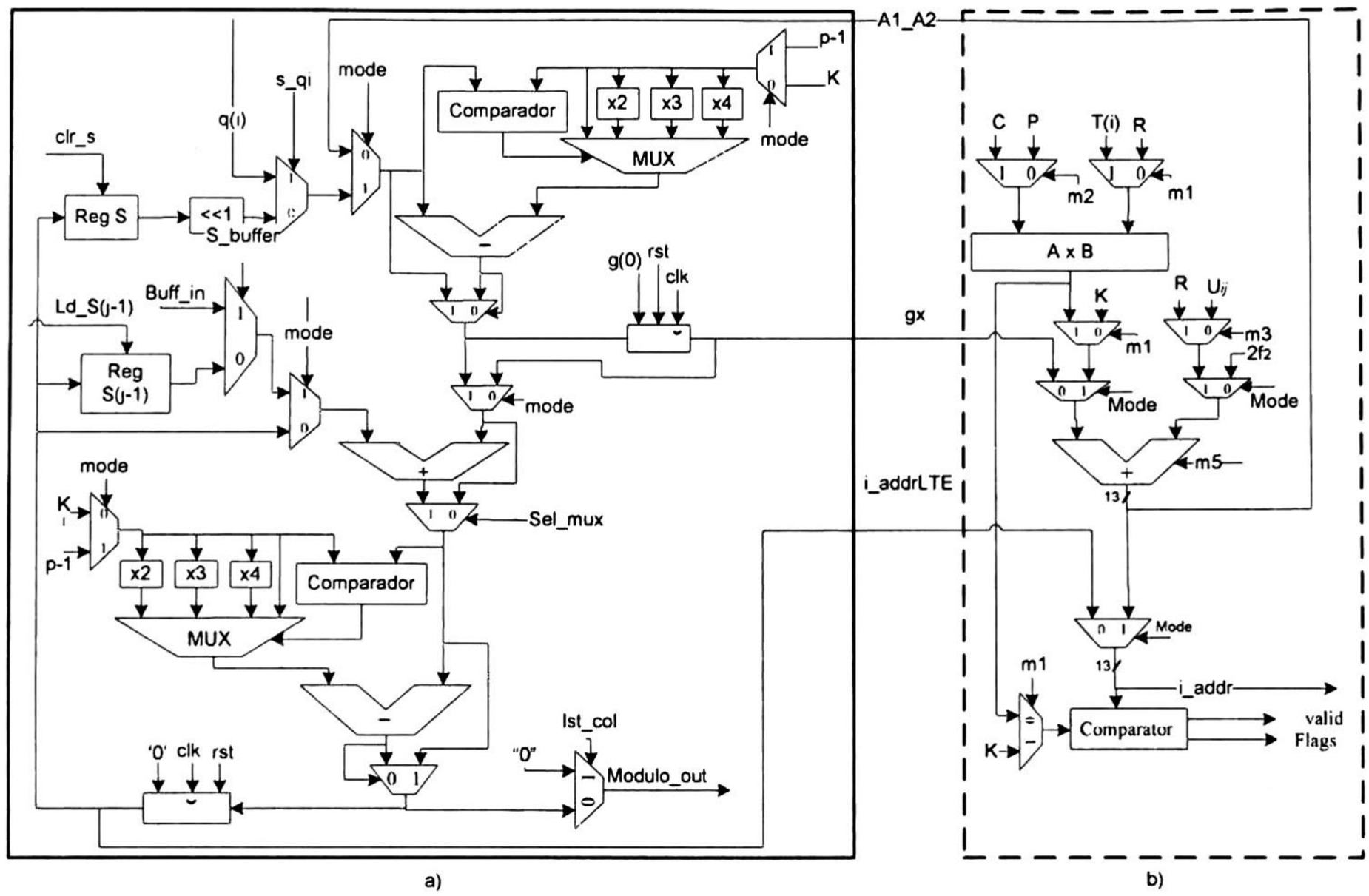

Figura 3.20. Interconexión de los bloques (a) "Modulo Computation" y (b) "Multi-operation" utilizados para generar las direcciones de entrelazado del estándar 3GPP-LTE.

Para lograr generar las direcciones de entrelazado de ambos estándares es necesario incluir algunas señales de control extra para el multiplexado de bloques, la forma en que el entrelazado o de-entrelazado de datos se hace sigue siendo similar, excepto porque para el estándar 3GPP-LTE donde aplicamos una especie de bypass al bloque "I/O RAM". En la Figura 3.21 se muestra el diagrama de estados para el Interleaver dual.

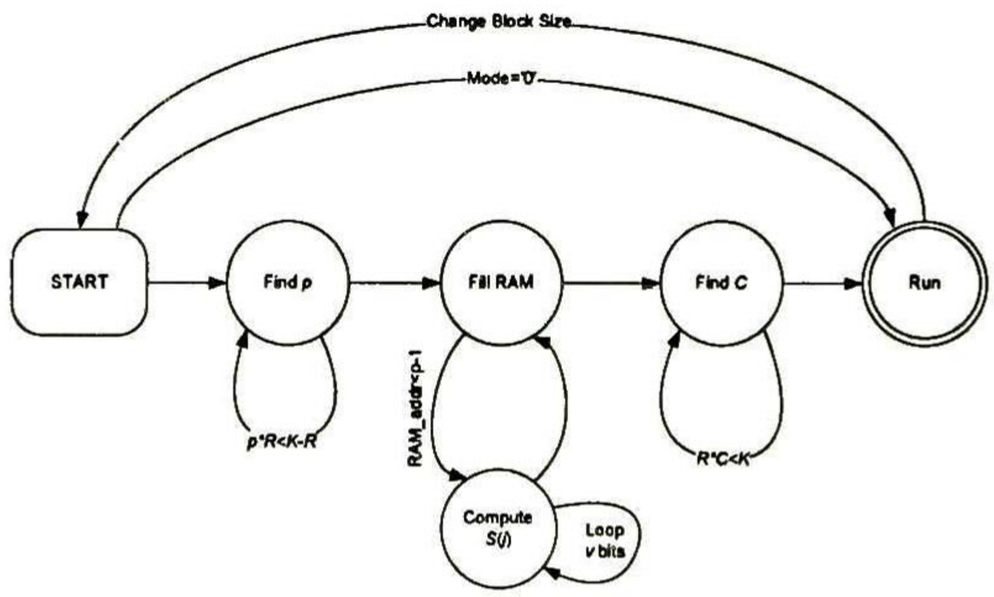

Figura 3.21. Diagrama de estados para el Interleaver/Deinterleaver para los estándares 3GPP-W CDMA y 3GPP-LTE.

Finalmente la arquitectura que incluye los bloques necesarios para hacer el entrelazado y de-entrelazado de datos para ambos estándares, se muestra en la Figura 3.22.

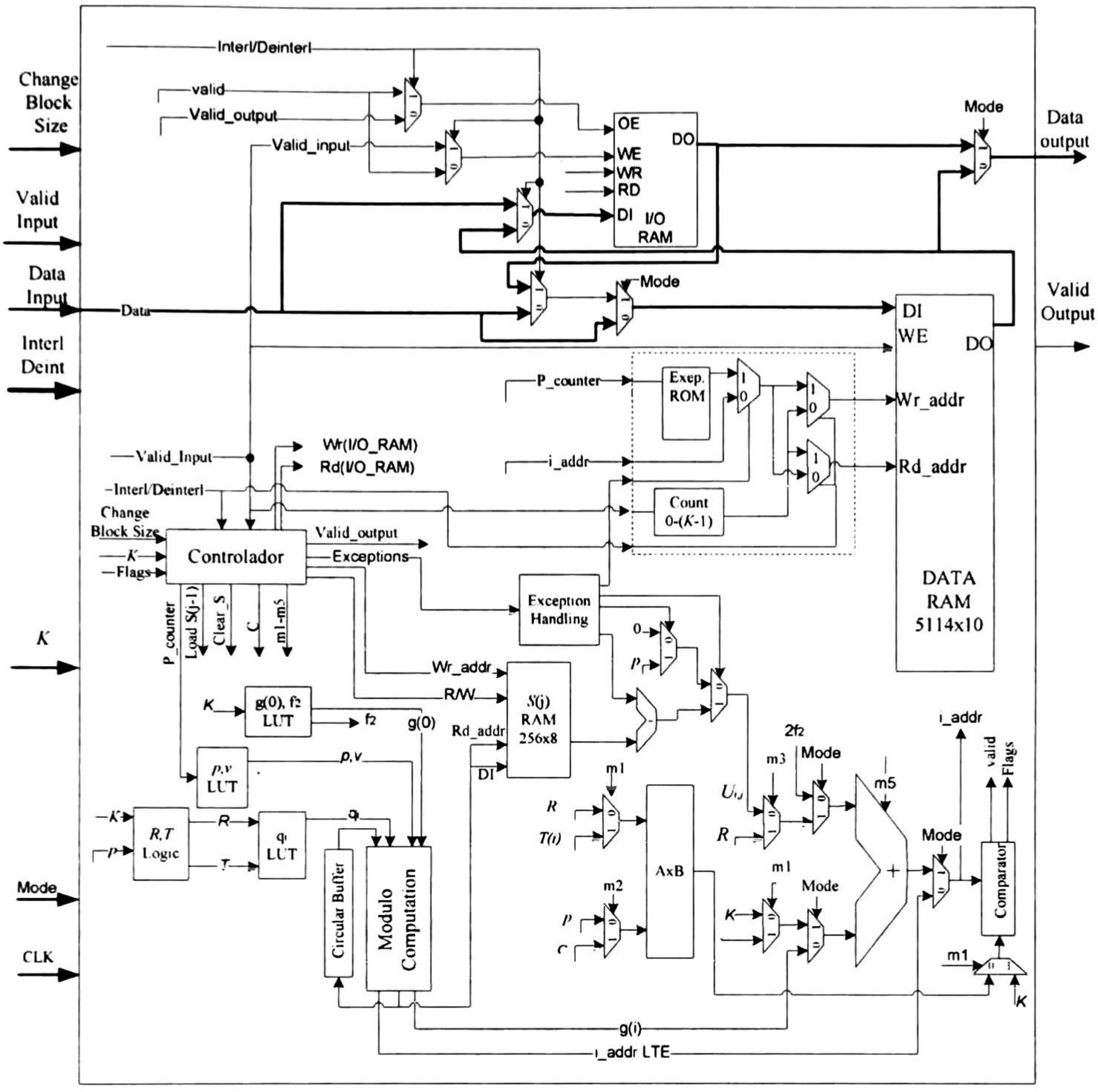

Figura 3.22. Arquitectura del Interleaver/Deinterleaver Reconfigurable para los estándares 3GPP-w CDMA y 3GPP-LTE.

Capítulo 3. Arquitecturas de distintos Interleavers

Capítulo 4

Descripción del diseño

En este capítulo se hace un análisis de nuestro diseño, donde iremos descomponiendo nuestra arquitectura principal en los bloques funcionales que la componen, haciendo una descripción sobre el papel que juega cada uno de estos bloques en el proceso de entrelazado o de-entrelazado de datos.

Nuestro bloque del I/D tiene la función de tomar un bloque de datos de tamaño K (K está definido por los estándares 3GPP-W CDMA y 3GPP-LTE), procesarlo y entregar a la salida un bloque de datos del mismo tamaño pero reordenado según el estándar que se eligió.

A continuación se hace una descripción más detallada de la forma de utilizar correctamente el I/D.

1. Al utilizar por primera vez el I/D es necesario fijar algunos parámetros a fin de garantizar un correcto funcionamiento. Los siguientes parámetros, sin importar el orden, deben ser fijados:
    - Se debe elegir el estándar con el que se trabajará. Con *mode="1"* se elije el estándar 3GPP-W CDMA, y con mode="0" se elije el estándar 3GPP-LTE.
    - Se debe seleccionar un tamaño de bloque de entrada K, a través de la señal de 13 bits con el mismo nombre que entra al I/D. K deberá ser acorde con el estándar seleccionado.
    - A través de la señal "I1_D0" se debe indicar al I/D si se requiere un entrelazado (I1_D0="1") o de-entrelazado de datos (I1_D0="0").
2. Después de tener fijos los parámetros antes mencionados, se aplica un RESET al I/D dejándolo listo para trabajar correctamente.
3. Una vez que se le aplicó el RESET al I/D, el tiempo necesario para poder recibir y procesar un bloque de datos de entrada dependerá del estándar elegido previamente.
    - Cuando el I/D está trabajando con el estándar 3GPP-LTE, pasarán 3 ciclos de reloj después del RESET para que el I/D pueda recibir y comenzar a procesar datos de entrada, siempre que se mantengan fijos los parámetros del punto 1.

- Cuando el I/D está trabajando con el estándar 3GPP-W CDMA, después de un RESET se requiere esperar un máximo de 800 ciclos de reloj para calcular los parámetros iníciales necesarios en este estándar. Después de calcular estos parámetros el I/D estará listo para recibir y procesar datos de entrada, siempre que se mantengan fijos los parámetros del punto 1.

4. Luego de pasar los puntos del 1 al 3 el I/D entra a una etapa de **funcionamiento normal**. En esta etapa el I/D iniciará el procesado de datos al momento en que estos sean introducidos a través de la señal de 10 bits "Data_Input" siempre que se le indique que estos datos son validos mediante la puesta en alto de señal "Dt_valid". El bloque de entrada deberá estar formado de K datos válidos consecutivos introducidos uno por cada ciclo de reloj y durante K ciclos para garantizar el funcionamiento correcto del I/D.

   En esta etapa, una vez que el primer dato valido entra al I/D, pasarán RxC ciclos de reloj antes de sacar los datos ya procesados. Una vez que comienzan a salir los datos procesados por la señal "Data_output", estos saldrán uno por cada ciclo de reloj y durante K ciclos. La señal de "SalnKvalid" será puesta en alto todo el tiempo que el I/D entregue los datos procesados.

5. Cuando el I/D está en funcionamiento normal y se requiere cambiar alguno o todos de los parámetros del punto 1, estos se pueden hacer pero solo serán tomados en cuenta después de avisarle al I/D mediante la puesta en alto de la señal "Change_block_size", con lo que el I/D vuelve al punto 3.

En la Figura 4.1 se muestra un ejemplo con formas de onda de las señales de entrada donde se ilustra los pasos antes mencionados sobre el uso del I/D.

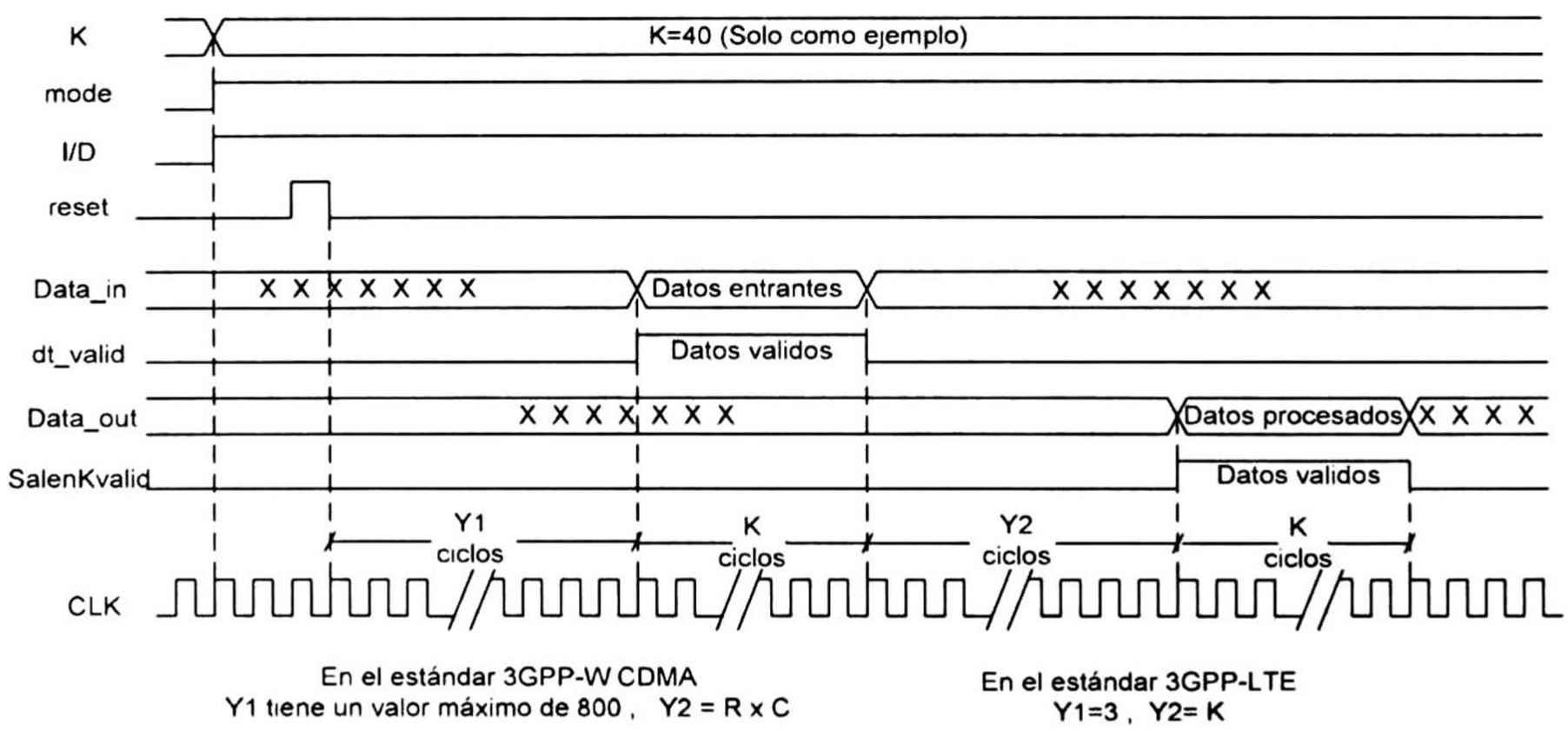

Figura 4.1. Ejemplo con diagrama de ondas para el uso del bloque "I/D".

Viendo nuestro diseño como un bloque funcional que pueda ser utilizado por un turbo codificador o turbo decodificador, tendríamos el bloque de la Figura 4.2. Las flechas que representan las señales en cada bloque son más gruesas cuando representan un bus que cuando representan señales de un solo bit, el ancho del bus está indicado entre corchetes.

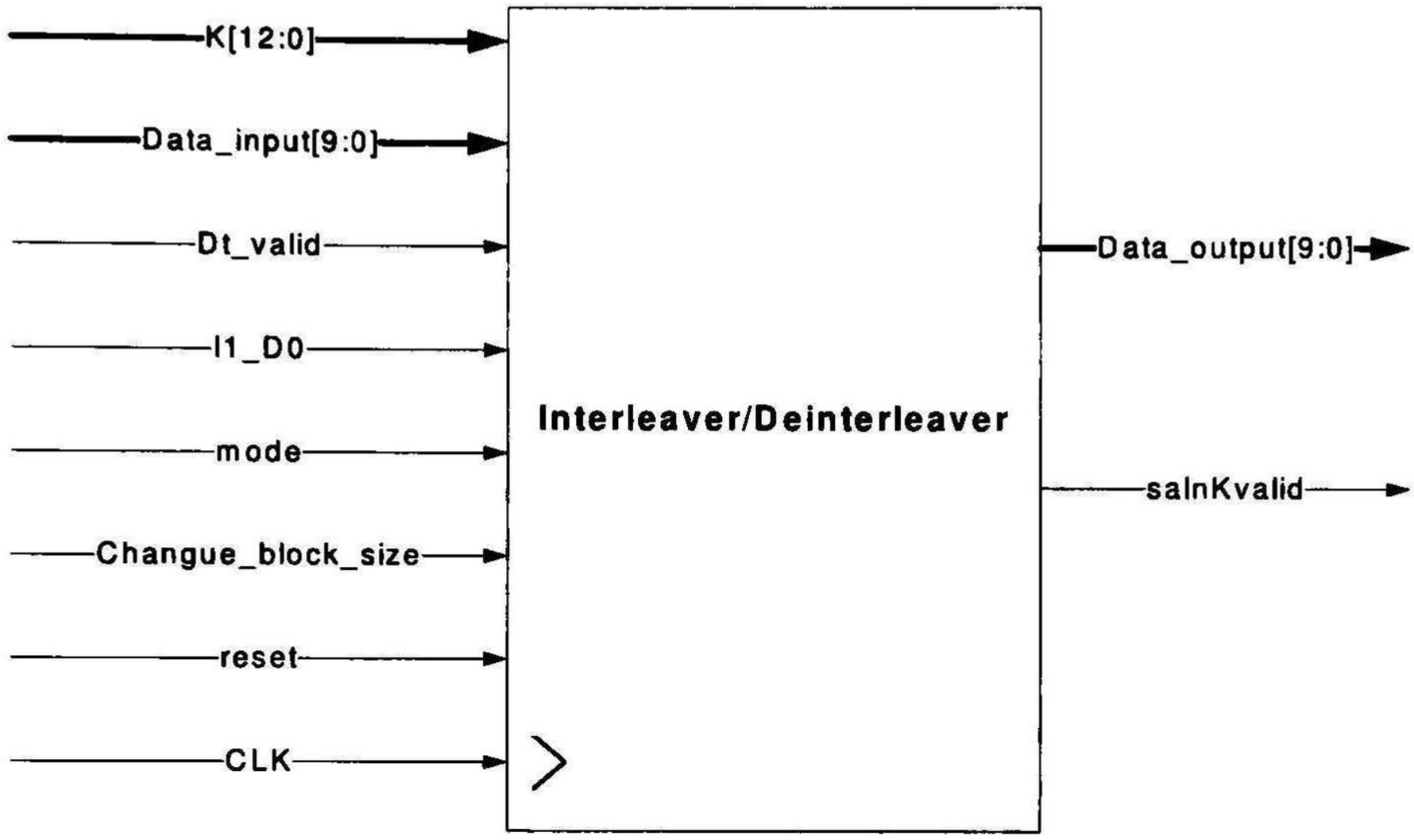

Figura 4.2. Bloque que representa al Interleaver/Deinterleaver donde se incluyen las señales de entrada-salida.

Donde una descripción de las señales de entrada y salida de la Figura 4.2 así como su origen y destino según sea el caso se muestra en la Tabla 4.1.

Tabla 4.1. Descripción de las señales de entrada-salida del bloque "I/D".

Entrada	Descripción	Procedencia
K	Señal de 13 bits que indica el tamaño de los bloques de entradas al I/D. Los valores que K puede tomar son $40 \leq K \leq 5114$ para el estándar 3GPP-W CDMA y algunos valores entre 40 y 6144 para el estándar 3GPP-LTE según lo indicado por la tabla 2.3.	Señal externa
Data_input	Señal de 10 bits por donde se introducen los datos de entrada al bloque I/D, uno por cada ciclo de reloj y	Señal externa

	durante K ciclos. Los datos solo son leídos durante la etapa de procesamiento.	
Dt_valid	Señal que indica al I/D en qué momento iniciar el procesamiento de los datos de entrada. Solo los datos que entran cuando esta señal está en alto son procesados. Dt_valid debe estar en alto K ciclos consecutivos para cada bloque.	Señal externa
I1_D0	Esta señal le indica al I/D el tipo de procesamiento que se aplicará a los datos de entrada. Cuando "I1_D0=1" los datos de entrada serán entrelazados y cuando "I1_D0=0" los datos de entrada serán de-entrelazados.	Señal externa
Mode	Esta señal le indica al I/D de acuerdo a que estándar deberá procesar los datos. Cuando "Mode=1" las trayectorias de entrelazado o de-entrelazado serán dictadas por el estándar 3GPP-W CDMA, y cuando "Mode=0" será el estándar 3GPP-LTE el que rija dichas trayectorias.	Señal externa
Change_Block_Size	Cuando esta señal se pone en alto, le indica al I/D que un cambio en el tamaño de bloque, K, ha ocurrido, reiniciando de esta manera la máquina de estados y el cálculo de nuevos parámetros iníciales. La función de esta señal solo es necesaria con el estándar 3GPP-W CDMA por lo que en "Mode=0" está deshabilitada.	Señal externa
Reset	Señal asíncrona que reinicia la máquina de estados usada para el estándar 3GPP-W CDMA y limpia los registros usados en el estándar 3GPP-LTE. Esta señal se activa en alto y al ser puesta en bajo nuevamente inicia el cálculo de parámetros para el valor K vigente.	Señal externa
CLK	Reloj que rige directamente a todos los bloques síncronos internos. El CLK máximo soportado por el I/D es de 30Mhz.	Señal externa
Salida	**Descripción**	**Destino**
Data_output	Señal de 10 bits a través de la cual salen los datos del I/D ya procesados. Una vez que estos datos empiezan a salir, saldrán uno por cada ciclo de reloj y durante K ciclos, que	Al exterior, a un MAP ó un RSC

	es lo necesario para sacar un bloque completo de datos.	
salenKvalid	Cuando esta señal está en alto nos indica que los datos a la salida del I/D son datos ya procesados correctamente. Esta señal dura en alto K ciclos para cada bloque.	Al exterior, a un MAP ó un RSC

Luego partiendo de la idea de que para lograr un entrelazado de datos lo principal es tener un generador de direcciones de entrelazado y un bloque de manejo y almacenamiento de los datos, podemos dividir nuestro diseño como se muestra en la Figura 4.3, donde una descripción de los bloques "Interleaver address generator" y "Data Handling" se da a continuación.

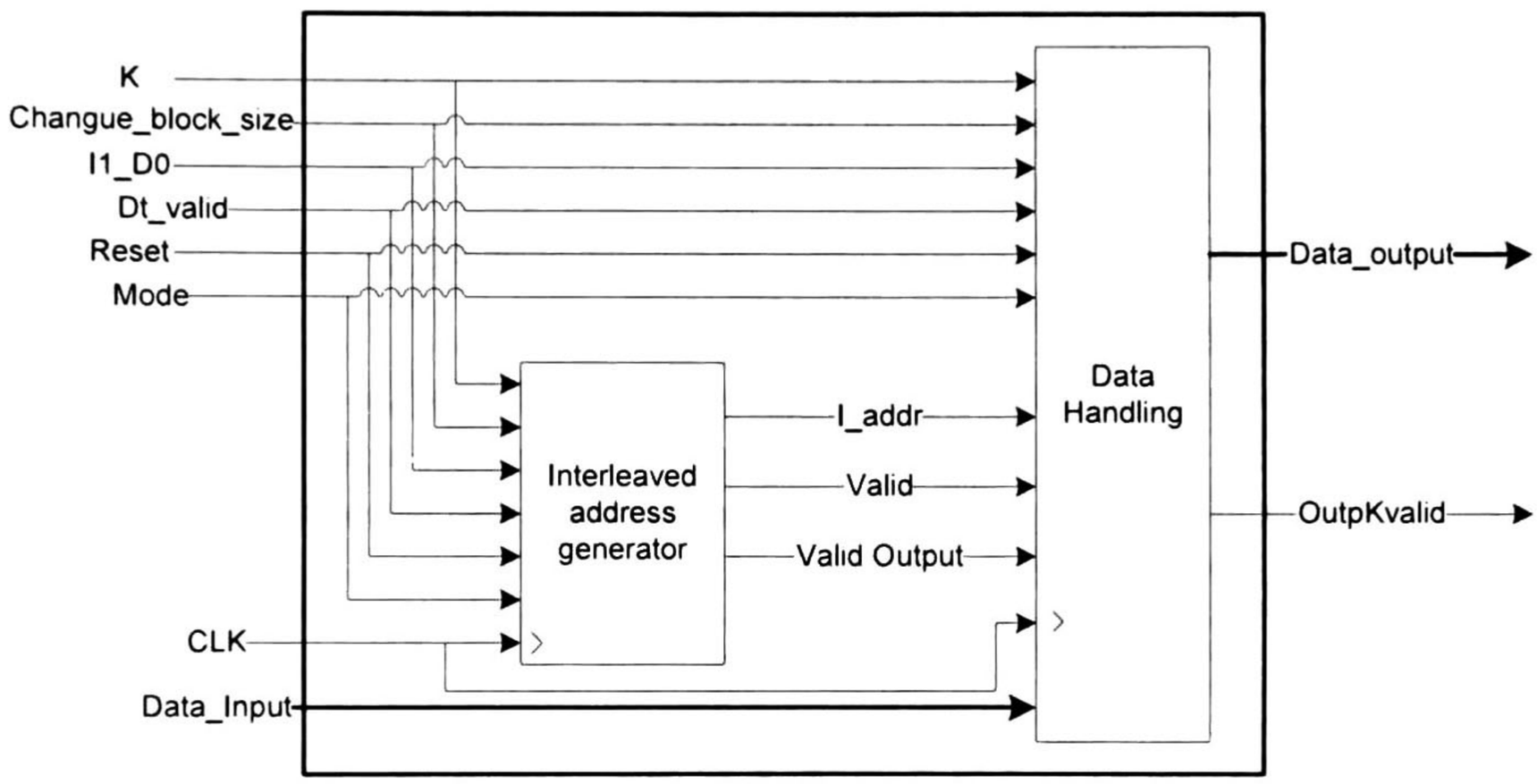

Figura 4.3. Bloques funcionales de mayor jerarquía del I/D.

4.1 Bloque "Interleaved address generator"

El bloque generador de direcciones entrelazadas funciona como la mayoría de los Interleavers existentes, es decir, solamente genera las direcciones entrelazadas que serán utilizadas posteriormente ya sea para el entrelazado o para el de-entrelazado de datos por otro bloque o sistema. Este bloque recibe las señales de entrada y en función del estándar elegido, el modo de operación, el tamaño de bloque y la recepción de los datos de entrada, generará las direcciones de entrelazado correspondientes que serán aprovechadas por el bloque "Data Handling".

Observando la Figura 4.3 nos damos cuenta de que todas las señales de entrada de este bloque provienen directamente del exterior y ya han sido descritas en la tabla Tabla 4.1, por otro lado las señales de salida de este bloque se describen en la Tabla 4.2.

Tabla 4.2. Descripción de las señales de salida del bloque "Interleaved Address Generator".

Salida	Descripción	Destino
I_addr	Esta señal de 13 bits entrega las direcciones de entrelazado que serán aprovechadas para el entrelazado o de-entrelazado de datos. Durante el proceso de generación de direcciones en el estándar 3GPP-W CDMA se generan algunas direcciones inválidas, las cuales son debidamente etiquetadas.	bloque "Data Handling"
Valid	Esta señal, puede ser vista como una bandera que al ser puesta en alto durante el proceso de generación de direcciones "i_addr", indica que dichas direcciones son validas.	bloque "Data Handling"
Valid_output	Esta señal solamente permanece en alto durante el proceso de generación de direcciones de entrelazado.	bloque "Data Handling"

En la Figura 4.4 se muestran los bloques y señales principales que conforman al generador de direcciones de entrelazado y como se conecta cada bloque con los demás.

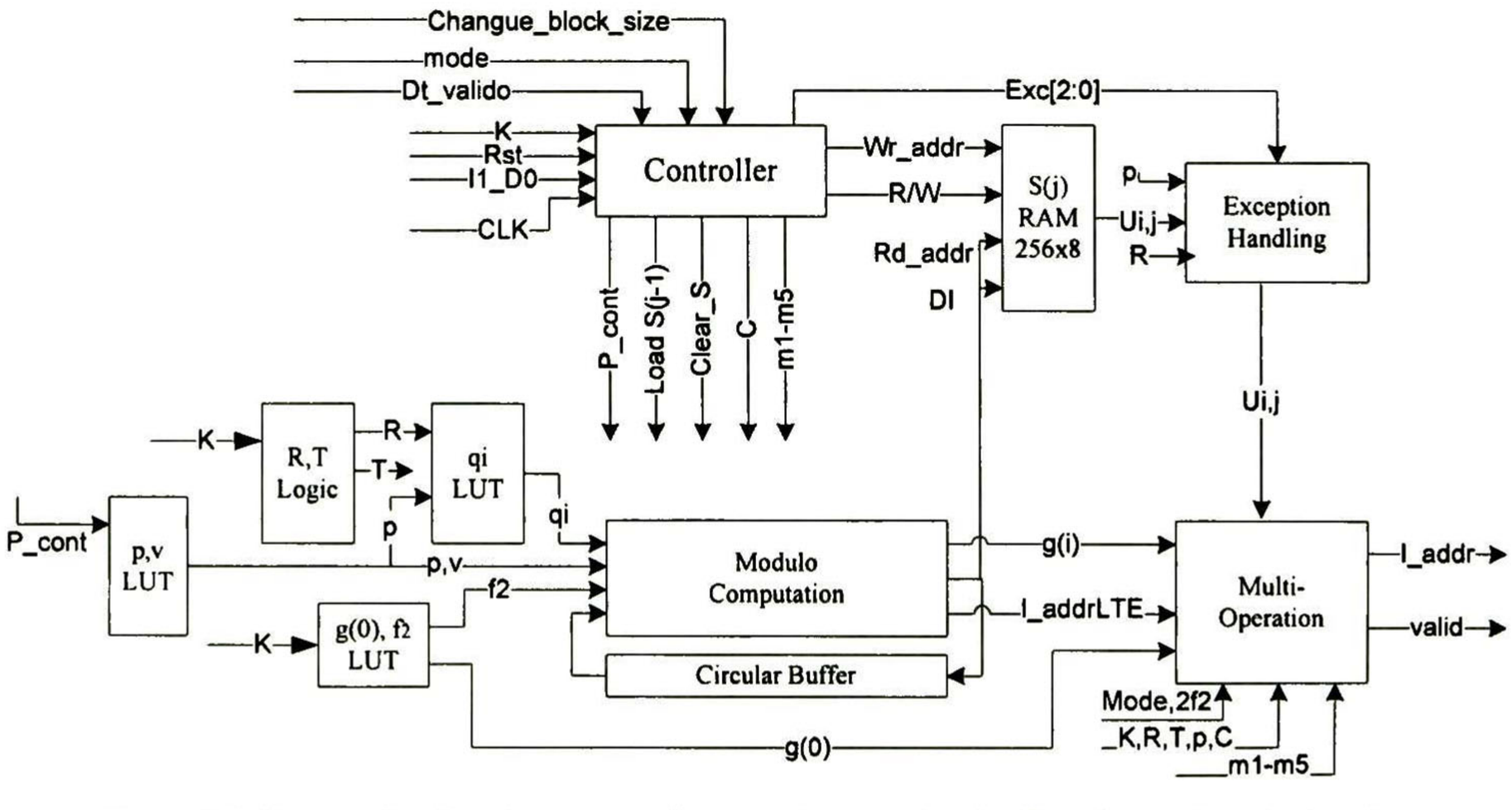

Figura 4.4. Bloques funcionales que conforman el generador de direcciones de entrelazado.

A continuación se hace una descripción de los bloques y las señales de la Figura 4.4.

4.1.1 Bloque "Controller"

Como ya se mencionó en el capítulo 3, varios bloques de la arquitectura del I/D funcionan en diferentes etapas del proceso, por lo que es necesario una maquina de estados para controlarlos mediante la generación de las banderas adecuadas. El bloque del controlador basa su funcionamiento en la máquina de estados mostrada en la Figura 3.21.

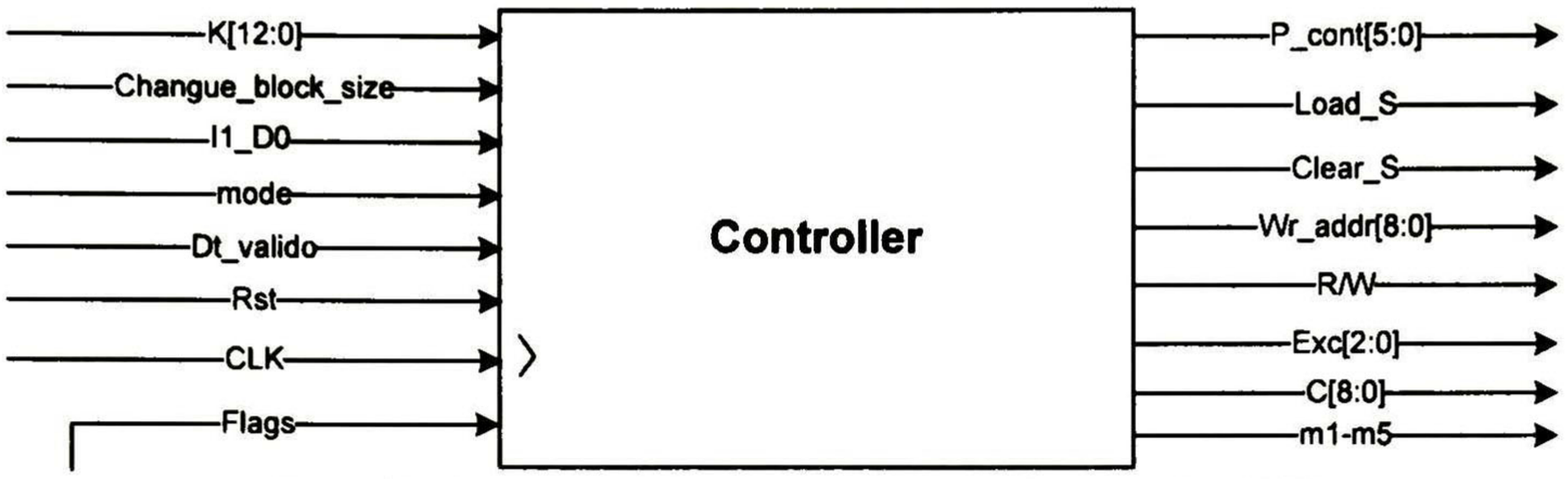

Figura 4.5. Diagrama del bloque que representa al controlador del I/D.

En la Figura 4.5 se muestra el bloque del controlador con sus entradas y salidas. Las banderas que salen del controlador comienzan a generarse después de un cambio del tamaño de bloque o después de un reset y ayudan a sincronizar al resto de la arquitectura.

En la Tabla 4.3 se hace una descripción de cada una de las señales del controlador que no hayan sido descritas previamente.

Tabla 4.3. Descripción de las señales de entrada-salida del bloque "Controller".

Entrada	Descripción	Origen
Flags	Esta es una señal de retroalimentación sobre el funcionamiento del bloque "Multi-Operations". Por medio de esta bandera se le informa al controlador si los cálculos efectuados en otros bloques y en cada estado son correctos o no, cuando no los son, el controlador sigue intentándolo hasta que esta bandera se pone en alto y entonces pasa al siguiente estado.	Bloque "Multi-Operations"
Salida	**Descripción**	**Destino**
P_count	Esta señal de 6 bits es un contador que sirve para direccionar la memoria ROM donde están almacenados en pares los valores de "p" y "v". En cada ciclo de reloj esta señal lee una nueva localidad de la "p,v	Bloque "p,v LUT"

	ROM" hasta encontrar los valores correctos y pasar al siguiente estado. "P_count" solo se utiliza con el estándar 3GPP-W CDM A.	
Load_S	Esta señal sirve para la sincronización en el cálculo iterativo y retroalimentación de la secuencia "S(j)" por parte del bloque "Modulo Computation" en la etapa de pre-calculo. De la ecuación **¡Error! No se encuentra el origen de la referencia.** se observa la necesidad de conocer valores anteriores S(j-1) para el cálculo de S(j). El controlador genera esta señal de acuerdo al algoritmo mostrado en la sección 3.1. Solo se utiliza con el estándar 3GPP-W CDMA.	bloque "Modulo Computation"
Clear_S	Esta señal sirve para limpiar los registros de retroalimentación durante el cálculo de "S(j)" en la etapa de pre-calculo. Esta señal se pone en alto en función del valor de "v". Para v=2 y v=3 "Clear_s" se pone en alto cada ciclo de reloj, para v=5, v=6 y v=7 cada 2 ciclos y para v=19 cada 4 ciclos de reloj. Solo se utiliza con el estándar 3GPP-W CDMA.	bloque "Modulo Computation"
Wr_addr	Esta señal de 9 bits entrega las direcciones de escritura de la "S(j)RAM". Estas direcciones solo son generadas durante el pre-calculo para almacenar el vector S(j), y van de 0 a p-2. El incremento de estas direcciones solo ocurre cuando "Clear_S" está en alto. Solo se utiliza con el estándar 3GPP-W CDMA.	Bloque "S(j)RAM"
R/W	Sirve para habilitar la lectura o escritura en la "S(j)RAM". En modo pre-calculo, cuando la secuencia S(j) está siendo calculada, R/W=1 sirve para escribir dicha secuencia S(j) en la "S(j)RAM". En el modo de cálculo, R/W=0, nos sirve para leer la secuencia S(j) de la memoria "S(j)RAM" y posteriormente obtener la secuencia $U_{i,j}$. Solo se utiliza con el estándar 3GPP-W CDMA.	Bloque "S(j)RAM"
Exc	Sirve para indicarle al bloque"Exceptions Handling" que existe alguna excepción y el tipo de dicha excepción. Solo se utiliza con el estándar 3GPP-W CDMA.	Bloque "Exceptions Handling"
C	Señal de 9 bits que indica el numero de columnas de la matriz virtual creada para el estándar 3GPP-W CDMA	Bloque "Multi Operations"
m1-m5	Cada una de estas señales de 1 bit sirven para configurar la mayoría de la arquitectura del I/D en cada etapa del proceso de cálculo de direcciones	Bloque "Multi Operations"

4.1.2 Bloque "p,v LUT"

El bloque "p,v LUT" es simplemente lo que se le conoce como una "Look Up Table" o una memoria ROM, donde están almacenados todos los posibles valores de "p" y "v" en pares, tal y como lo muestra la Tabla 2.1. La razón por la que se decidió hacerlo de esta manera, es porque el tamaño de la memoria utilizada es relativamente pequeño (de 52 localidades de 14 bits cada una).

Esta memoria se diseño de tal forma que cada que se le pase una dirección de lectura, un valor de "p" y uno de "v" se mostrarán a la salida sin esperar ningún habilitador. Los valores de "p" y "v" son sacados por un mismo bus, donde los 9 bits más altos representan el valor de "p" y los 5 bits más bajos representan el valor de "v". Este bloque solamente se utiliza cuando se está trabajando bajo el estándar 3GPP-W CDMA.

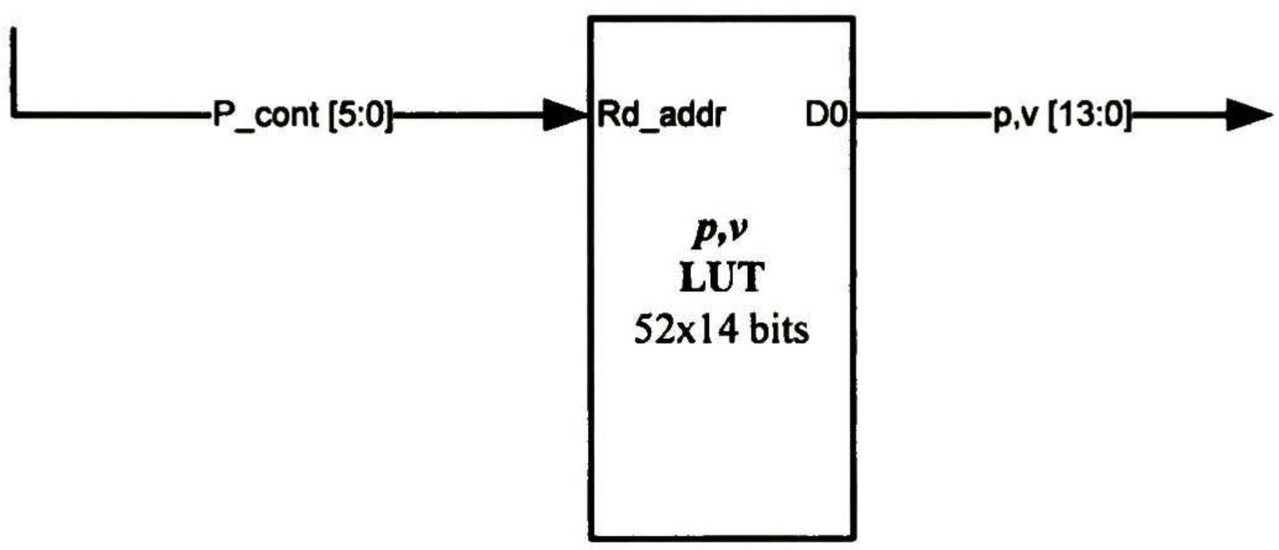

Figura 4.6. Diagrama que representa al bloque "p,v LUT"

En la Figura 4.6 podemos ver el bloque de donde obtenemos p y v, las señales de entrada y salida con que este bloque cuenta son las mostradas en la Tabla 4.4.

Tabla 4.4. Descripción de las señales de entrada-salida del bloque "p,v LUT".

Entrada	Descripción	Origen
P_count	Esta señal de 6 bits se encarga de proporcionar a la LUT la dirección de donde se leerán p y v. Esta señal está limitada de tal forma que nunca tendrá un valor menor a 0 ó mayor a 51	Bloque "Controller"
Salida	**Descripción**	**Destino**
pv	Esta señal de 14 bits entrega los valores leídos de p y v de forma concatenada donde los 9 bits más altos representan al valor de p y los 5 más bajos al valor de v.	Bloque "Multi Operations" y Bloque "Modulo Computation"

4.1.3 Bloque "R,T Logic"

Este bloque síncrono es utilizado por la arquitectura del I/D para encontrar el numero de renglones, "R", y la trayectoria de permutación de los mismos, "T", ambos parámetros son función solo de K. Para encontrar el número de renglones, R, este bloque utiliza lógica sencilla como comparaciones y asignaciones. La trayectoria de permutación de renglones, "T", se elige entre un grupo de 4 vectores previamente almacenados en este bloque, el vector adecuado dependerá del valor de K.

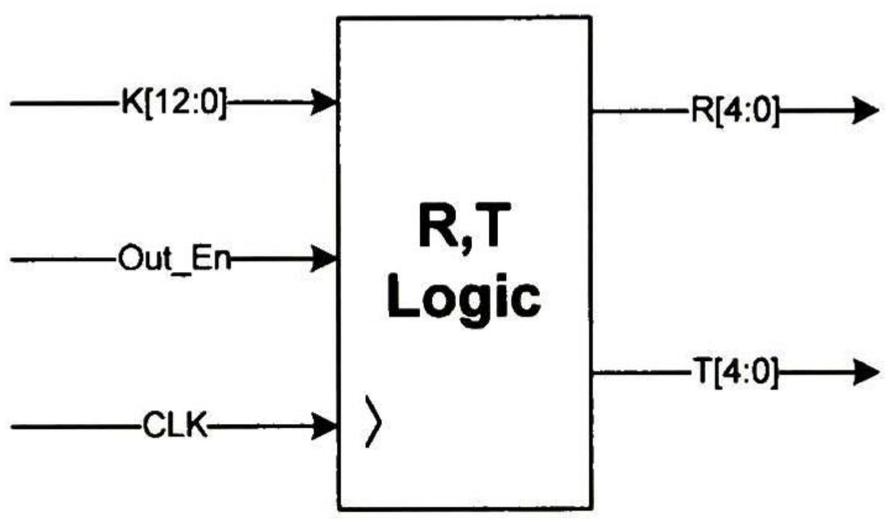

Figura 4.7. Diagrama que representa al bloque "R,T Logic"

En la Tabla 4.5 se hace una descripción de las señales de entrada y salida de este bloque mostrado en la Figura 4.7.

Tabla 4.5. Descripción de las señales de entrada-salida del bloque "R,T Logic"

Entrada	Descripción	Origen
K	Señal de 13 bits que indica el tamaño del bloque, y es en función a esta señal que se elije el valor adecuado de R y T.	Exterior
Out_En	Esta señal es utilizada para sincronizar la salida del vector T, con el resto de los procesos que lo utilizan. Cuando Out_En está en alto, en cada ciclo de reloj saldrá un elemento del vector T. Esta señal solo permanece en alto durante la etapa de cálculo en el estándar 3GPP-W CDMA	Bloque "Controller"
CLK	Reloj general del sistema.	Exterior
Salida	**Descripción**	**Destino**
R	Esta señal de 5 bits contiene el valor del número de renglones y solo puede tener los valores de 5, 10 ó 20. Esta señal solo es	Bloque "Multi Operations", bloque

	utilizada con el estándar 3GPP-W CDMA.	"Circular Buffer" y bloque "qi LUT"
T	A través de esta señal de 5 bits se entregan cada uno de los valores del vector de permutaciones entre renglones T(i). Los elementos de este vector son leídos de forma periódica durante el proceso de cálculo.	Bloque "Multi Operations"

4.1.4 Bloque "qi LUT"

De este bloque síncrono podemos obtener la secuencia auxiliar "qi" que posteriormente se utiliza para obtener la permutación entre los elementos de cada renglón. En el estándar se menciona como calcular este vector, sin embargo después de calcular todas las posibles trayectorias de "qi" nos damos cuenta de que aproximadamente la mitad de estas trayectorias es la misma y el resto difieren en solo uno o máximo dos elementos del vector. Debido a lo anterior fue que decidimos guardar los vectores con las posibles trayectorias "qi" en una ROM dentro del bloque de donde se lee el vector indicado de acuerdo a los valores de p y R. La memoria en que están guardados los elementos del vector "qi" tiene un tamaño de 22 localidades de 7 bits cada una. En este bloque se utiliza lógica extra para decidir cuál es el vector que se deberá leer.

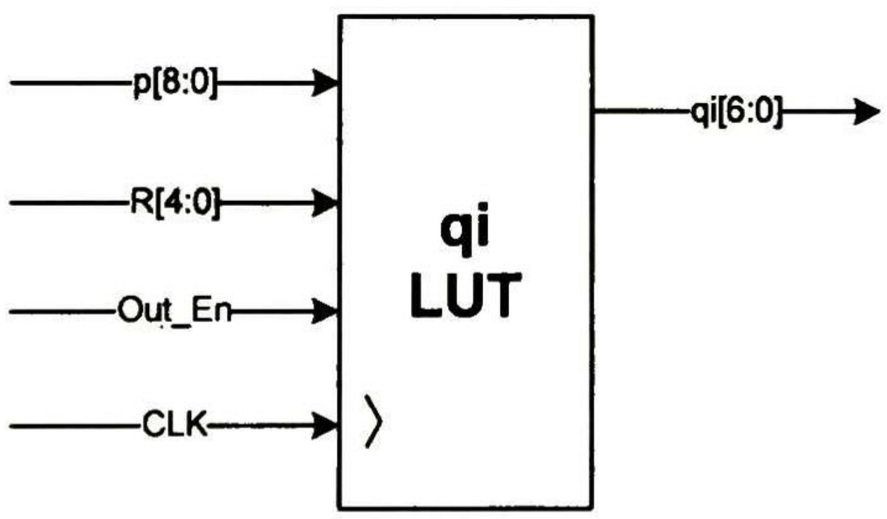

Figura 4.8. Diagrama que representa al bloque "qi LUT"

En la Figura 4.8 se muestra el diagrama del bloque "qi LUT" y en la Tabla 4.6 se hace una descripción de las señales relacionadas con este bloque.

Tabla 4.6. Descripción de las señales de entrada-salida del bloque "qi LUT"

Entrada	Descripción	Origen
p	Esta señal de 9 bits contiene al parámetro "p", el cual es	Bloque "p,v LUT"

	necesario para elegir un vector apropiado "qi"	
R	Esta señal de 5 bits contiene al parámetro "R", el cual, al igual que "p" es necesario para la elección del vector "qi"	Bloque "R,T Logic"
Out_En	Señal que sirve para habilitar la salida de los elementos del vector "qi" Con esta señal se sincronizan las operaciones que utilizan los elementos del vector "qi". Esta señal solo es puesta en alto durante la etapa de cálculo de direcciones en el estándar 3GPP-W CDMA.	Bloque "Controller"
CLK	Reloj general del sistema	Exterior
Salida	**Descripción**	**Destino**
qi	Esta señal de 7 bits es por donde son entregados los elementos del vector "qi" al bloque correspondiente. Los tiempos en que estos valores son entregados dependen de la máquina de estados y es el controlador el que habilita la salida. El vector "qi" puede tener un máximo de 20 valores y cada uno de ellos será leído en más de una ocasión	Bloque "Modulo Computation"

4.1.5 Bloque "g(0),f2 LUT"

Este es un bloque exclusivamente utilizado para el cálculo de direcciones del estándar 3GPP-LTE, dentro de este bloque hay una memoria ROM donde son guardados todos los valores de los parámetros "g(0)" y "f2" que según el estándar son los necesarios para el cálculo de las direcciones. La elección del valor correcto de "g(0)" y "f2" depende del valor de K, para hacer esta relación se realizan una serie de comparaciones dentro del bloque. La LUT utilizada dentro de este bloque tiene un tamaño de 189 localidades de 20 bits cada una, esto es porque en cada localidad de la memoria están almacenados en pares "g(0)" y "f2", tal y como en la Tabla 2.3.

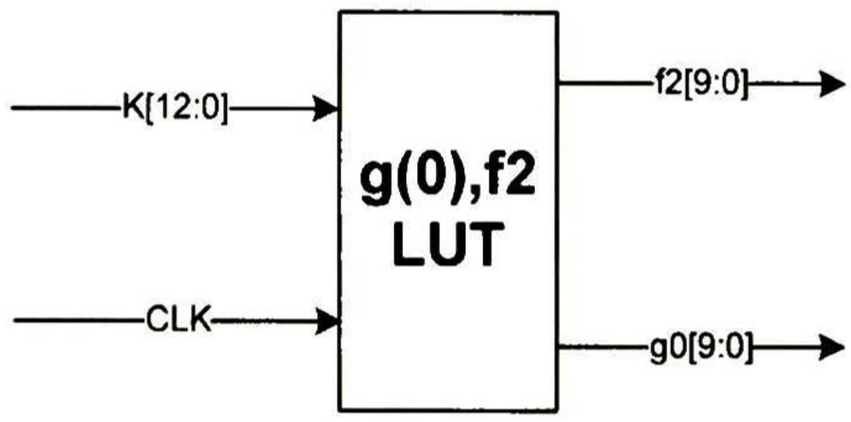

Figura 4.9. Diagrama que representa al bloque "g(0),f2 LUT"

En la Figura 4.9 se muestra el bloque "g(0),f2 LUT" . Las señales relacionadas con este bloque se describen en la Tabla 4.7.

Tabla 4.7. Descripción de las señales de entrada-salida del bloque "g0,f2 LUT".

Entrada	Descripción	Origen
K	Señal de 13 bits que indica el tamaño del bloque, y es en función a esta señal que se elije el valor adecuado de f2 y g0.	Exterior
CLK	Reloj general del sistema	Exterior
Salida	**Descripción**	**Destino**
f2	Esta señal de 10 bits es por donde se entrega el parámetro f2 al bloque correspondiente para hacer los cálculos de las direcciones de entrelazado. Una vez que este valor es encontrado permanece fijo hasta que el valor de K vuelva a cambiar. Esta señal solo se utiliza con el estándar 3GPP-LTE	Bloque "Multi Operations"
g0	Esta señal de 10 bits es utilizada para entregar el parámetro g(0) al bloque correspondiente para hacer los cálculos de las direcciones de entrelazado, al igual que f2, esta señal permanece fija durante el tiempo que K permanezca fija y solo se utiliza con el estándar 3GPP-LTE	Bloque "Modulo Computation"

4.1.6 Bloque "Multi-Operation"

Este es uno de los bloques más importante de la arquitectura, y es el encargado de hacer las sumas, multiplicaciones y comparaciones necesarias en cada etapa del proceso de generación de direcciones. Este módulo es puramente combinacional y está construido básicamente con multiplexores, sumadores y multiplicadores. El correcto funcionamiento de este bloque está estrechamente relacionado con su comunicación con el controlador a través de las banderas m1-m5 y la bandera de retroalimentación "Flags"

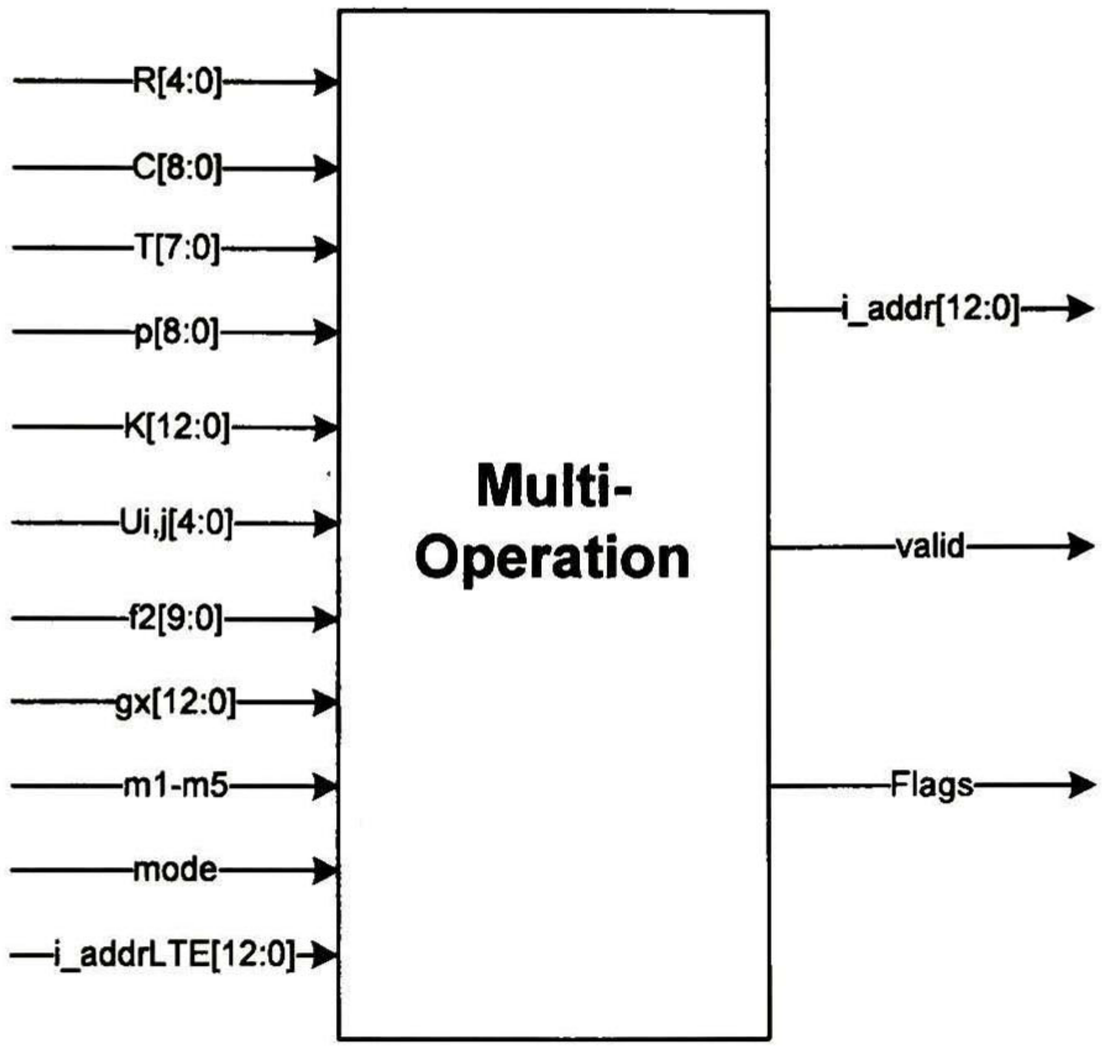

Figura 4.10. Diagrama que representa al bloque "Multi-operation"

A este bloque, mostrado en la Figura 4.10, le entran la mayoría de los parámetros involucrados en el proceso de generación de direcciones de entrelazado, y de acuerdo al estado en que el proceso se encuentre es la operación que este bloque realizará. Las señales de entrada y salida así como una descripción de las mismas se muestran en la Tabla 4.8 .

Tabla 4.8. Descripción de las señales de entrada-salida del bloque "Multi Operations".

Entrada	Descripción	Origen
R	Señal que contiene al parámetro "R" para ser utilizado en operaciones del proceso de generación de direcciones.	Bloque "R,T Logic"
C	Señal que contiene al parámetro "C" para ser utilizado en operaciones del proceso de generación de direcciones.	Bloque "Controller"
T	Señal que contiene a los elementos del vector "T" para ser utilizados en el proceso de generación de direcciones.	Bloque "R,T Logic"
p	Señal que contiene al parámetro "p" para realizar las operaciones requeridas por el sistema.	Bloque "p,v LUT"

K	Señal que contiene el tamaño del bloque de entrada.	Exterior
Uij	Señal que contiene los desplazamientos en los elementos de cada renglón. Esta señal es utilizada durante la etapa de cálculo con el estándar 3GPP-LTE	Bloque "Exceptions Handling"
f2	Señal que contiene al parámetro "f2" para ser utilizado en el cálculo de direcciones en el estándar 3GPP-LTE.	Bloque "g0,f2 LUT"
gx	Señal que sirve como retroalimentación en el proceso iterativo del cálculo de direcciones en el estándar 3GPP-LTE. Esta señal inicialmente contiene al parámetro "g(0)".	Bloque "Modulo Computation"
m1-m5	Estas señales son utilizadas para configurar la arquitectura del bloque "Multi Operations" según se requiera en cada estado del proceso de generación de direcciones de entrelazado.	Bloque "Controller"
mode	Esta señal ayuda en la conjuración de la arquitectura del bloque en función del estándar con que se esté trabajando, 3GPP-LTE ó 3GPP-W CDMA.	Exterior
I_addrLTE	Esta señal de 13 bis contiene las direcciones de entrelazado generadas para el estándar 3GPP-LTE.	Bloque "Modulo Computation"
Salida	**Descripción**	**Destino**
I_addr	Esta señal de 13 bits es por donde las direcciones de entrelazado generadas son entregadas, estas direcciones son sacadas una por ciclo de reloj gracias al controlador y su manejo de los parámetros $U_{i,j}$ y T. Las direcciones sacadas a través de esta señal no necesariamente son todas validas.	Bloque "Muxes"
valid	Esta señal sirve para indicar si las direcciones entrelazadas que salen por "i_addr" son validas o no, lo cual depende del tamaño del bloque K y de las excepciones del estándar. En el estándar 3GPP-LTE todas las direcciones generadas durante el proceso de generación de direcciones de entrelazado son validas, sin embargo en el estándar 3GPP-WCDMA	Bloque "In/Out RAM"

	no sucede igual.	
Flags	Esta señal sirve para indicarle al controlador el resultado de alguna comparación hecha por este bloque. En cada estado que alguna comparación es hecha, el controlador está en espera de una respuesta a través de esta señal.	Bloque "Controller"

En la Figura 4.11 se muestra la arquitectura interna del bloque "Multi Operations"

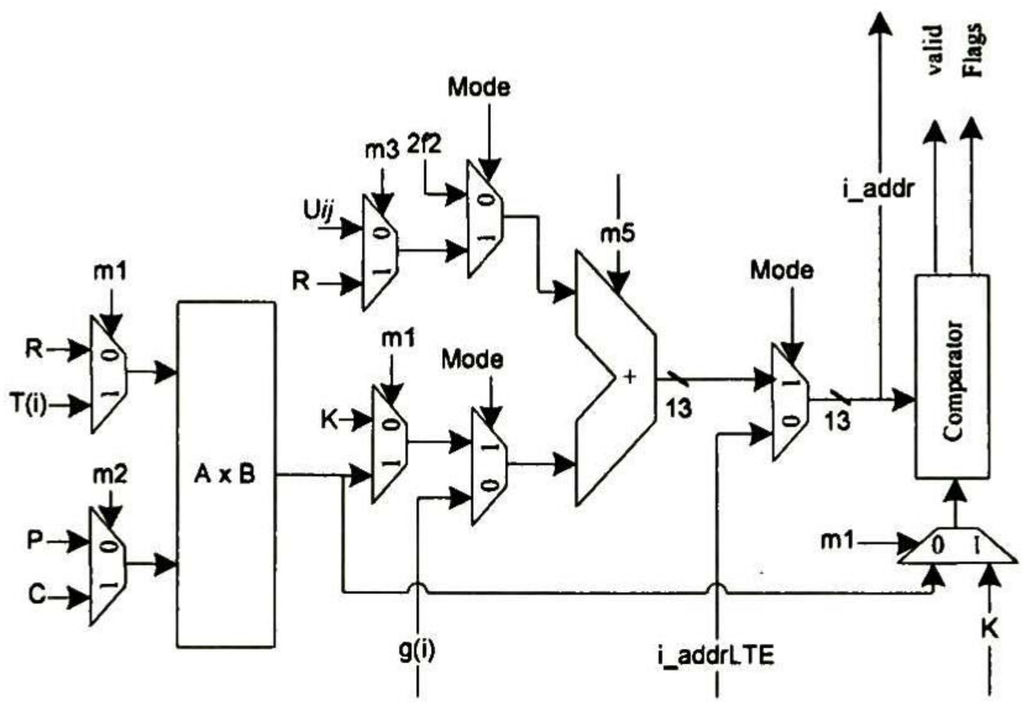

Figura 4.11. Arquitectura interna del bloque "Multi-operation"

4.1.7 Bloque "Exceptions Handling"

Este bloque es el encargado del manejo de las excepciones generadas cuando se está trabajando bajo el estándar 3GPP-W CDMA. Para lograr generar las direcciones de entrelazado del estándar 3GPP-W CDMA uno de los parámetros más importantes es el que contiene los desplazamientos dentro de cada renglón $U_{i,j}$, sin embargo cuando ocurre una excepción algunos de estos desplazamientos cambian y en lugar de ser un elemento extraído de la S(j)RAM, como la mayoría de los valores de $U_{i,j}$, es un valor constante definido por el estándar y contenido por el bloque de manejo de excepciones. Este bloque entra en acción cuando alguna de las excepciones mencionadas en la sección 2.6.1 se presenta, en cualquiera de esos casos el controlador le informa a este bloque acerca del tipo de excepción que está ocurriendo y entonces el bloque de manejo de excepciones actúa.

Figura 4.12. Diagrama que representa al bloque "Exceptions Handling".

En la Figura 4.12 se muestra el bloque de "Exceptions Handling" con sus señales de entrada y salida. En la Tabla 4.9 se hace una breve descripción de las señales del bloque "Exceptions Handling".

Tabla 4.9. Descripción de las señales de entrada-salida del bloque "Exceptions Handling".

Entrada	**Descripción**	**Origen**
Exc	Esta señal de 3 bits proveniente es por donde se le indica al bloque de manejo de excepciones que tipo de excepción está ocurriendo, y por tanto que medidas deberá tomar. En base a esta señal es que se controlan varios selectores de multiplexores en los tiempos adecuados.	Bloque "Controller"
R	Esta señal contiene al parámetro "R", que ayuda a llevar un conteo de las filas en la sincronización de generación de excepciones.	Bloque "R,T Logic"
p	Señal que contiene al parámetro "p"	Bloque "p,v Lut"
Uij_in	Señal que contiene los desplazamientos en cada renglón antes de considerar las excepciones.	Bloque "S(j)RAM"
CLK	Reloj general del sistema.	Exterior
Salida	**Descripción**	**Destino**
Uij	Señal que contiene los desplazamientos para cada elemento de cada renglón durante el cálculo de direcciones de entrelazado para el estándar 3GPP-W CDMA.	Bloque "Multi Operations"

En la Figura 4.13 se muestran las partes principales en la arquitectura del bloque de excepciones, el cual cuenta con un bloque secuencial (Exc_type) que recibe las banderas que contienen el tipo de excepciones y en base a ellas controla la parte combinacional que consta de multiplexores y un sumador.

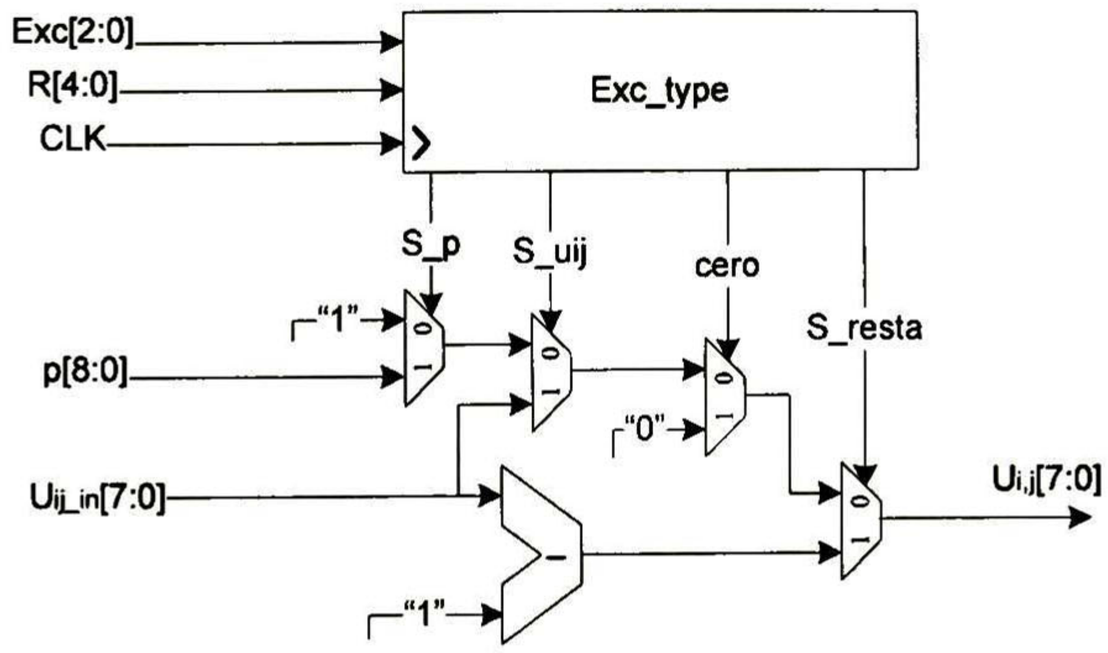

Figura 4.13. Arquitectura interna del bloque "Exceptions Handling"

4.1.8 Bloque "S(j)RAM"

Este bloque es, como su nombre lo dice, simplemente una memoria RAM donde se guardan los valores de la secuencia base para las permutaciones dentro de cada renglón, S(j). Esta memoria se utiliza solo en la generación de direcciones del estándar 3GPP-W CDMA y es llenada con los valores de S(j) durante la etapa de pre-calculo; Luego de ser llenada en la etapa de pre-calculo, es leída en la etapa de "calculo" en un orden calculado tal que se obtienen los desplazamientos de cada renglón.

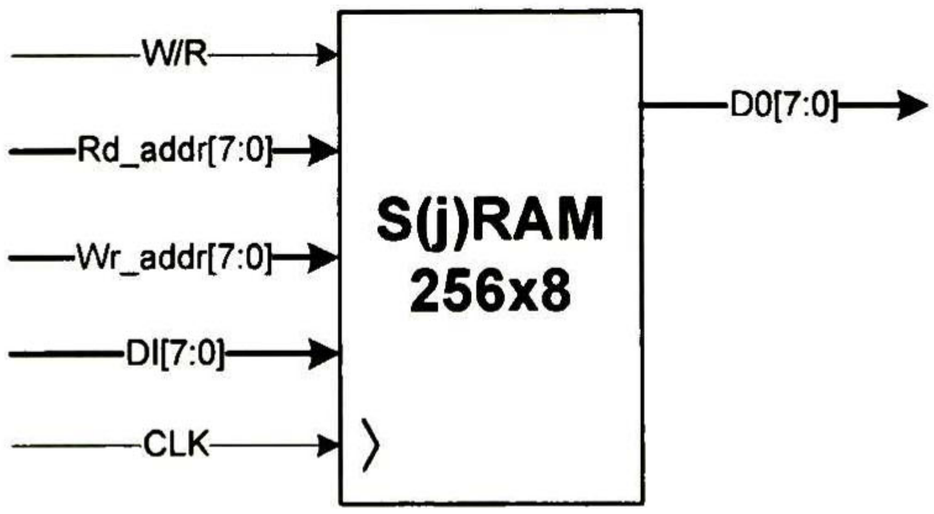

Figura 4.14. Diagrama que representa al bloque "S(j)RAM" .

En la Figura 4.14 se muestra el bloque "S(j)RAM" así como las señales de entrada y salida que utiliza, las cuales se describen en la Tabla 4.10.

Tabla 4.10. Descripción de las señales de entrada-salida del bloque "S(j)RAM".

Entrada	Descripción	Origen
W/R	Esta señal sirve para habilitar ya sea la escritura o la	Bloque "Controller"

	lectura en este bloque. La habilitación de la escritura en este bloque, "W/R=1", solo ocurre en la etapa de "pre-calculo". Durante la etapa de cálculo esta señal es puesta en bajo para habilitar la lectura de la "S(j)RAM".	
Rd_addr	Esta señal es generada durante la etapa de "calculo". El orden de estas direcciones esta dado por la ecuación **¡Error! No se encuentra el origen de la referencia.** y nos sirve para leer de la "S(j)RAM" los valores de los desplazamientos Uij, para el cálculo de las direcciones de entrelazado.	Bloque "Modulo Computation"
Wr_addr	Esta señal se utiliza durante la etapa de "pre-calculo". El cálculo de los valores de "S(j)" y "Wr_addr" son hechas a la par. Estas direcciones de escritura para este bloque, son calculadas de forma ascendente continua y van de 0 a un máximo de 255 dependiendo del valor del parámetro "p"	Bloque "Controller"
DI	Por esta señal es por donde entras los datos, "S(j)". Estos datos de entrada solo pueden ser escritos durante el proceso de "pre-calculo" y en los tiempos que el controlador lo permita.	Bloque "Modulo Computation"
CLK	Reloj general del sistema	Exterior
Salida	**Descripción**	**Destino**
DO	A través de esta señal es por donde los datos leídos, que representan los desplazamientos en los elementos de cada renglón son entregados. Estos datos contienen las permutaciones dentro de cada renglón antes de aplicarle las correcciones relacionadas a las excepciones	Bloque "Exceptions Handling"

4.1.9 Bloque "Modulo Computation"

Este bloque es, junto con el controlador, posiblemente el bloque más importante de todos; este bloque es utilizado para calcular y guardar en RAM (con la ayuda del controlador) la secuencia "S(j)" durante la etapa de "pre-calculo", después en la etapa de "calculo" este mismo bloque es configurado por el controlador para generar las direcciones de lectura de la "S(j)RAM" y de esta forma encontrar los valores $U_{i,j}$. Las operaciones anteriores son realizadas cuando se está

trabajando bajo el estándar 3GPP-W CDMA, sin embargo este bloque también juega un papel primordial en el cálculo de las direcciones de entrelazado para el estándar 3GPP-LTE.

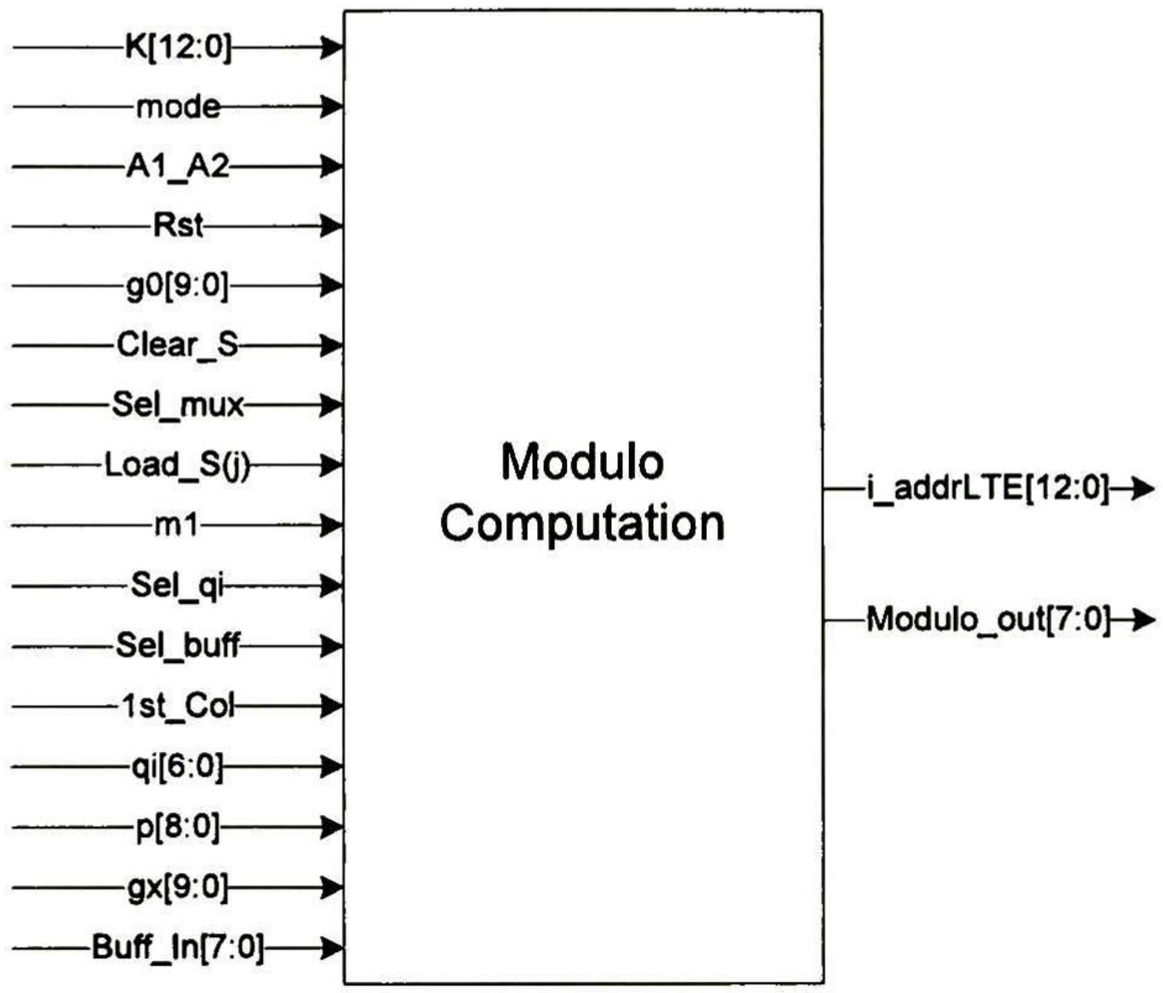

Figura 4.15. Diagrama que representa al bloque "Modulo Computation"

En la Figura 4.15 se muestra el bloque "Modulo Computation" así como sus señales de entrada y salida. En la Tabla 4.10 se muestra una breve explicación de las señales de este bloque.

Tabla 4.11. Descripción de las señales de entrada-salida del bloque "Modulo Computation".

Entrada	Descripción	Origen
K	Señal que indica el tamaño del bloque a procesar	Exterior
mode	Sirve para configurar algunos bloques internos según el estándar en el que se esté trabajando. Cuando "mode=1", este bloque se configura para trabajar bajo el estándar 3GPP-W CDMA y cuando "mode=0" este bloque será configurado según lo requiera el estándar 3GPP-LTE.	Exterior
A1_A2	Esta señal de 13 bits, se utiliza solamente cuando el estándar elegido sea 3GPP-LTE, y sirve para interconectar este bloque con el bloque "Multi-operation" (ver Figura 3.20).	Bloque "Multi Operations"
Rst	Cuando "Rst=1" sirve para limpiar todos los registros internos de este bloque, permitiendo así comenzar	Exterior

	nuevamente las operaciones de este bloque. Al ser "Rst" una señal asíncrona, esta tiene efecto en cualquier momento que sea puesta en alto.	
g0	Esta señal contiene al parámetro del mismo nombre "g0" y solo es utilizada con el estándar 3GPP-LTE.	Bloque "g0,f2 LUT"
Clear_S	Esta señal sirve para poner en cero el registro interno "RegS". "Clear_S" es puesta en alto de forma periódica durante el cálculo de la secuencia "S(j)" en la etapa de "pre-calculo".	Bloque "Controller"
Sel_mux	Esta señal sirve para controlar algunas de las sumas del proceso de cálculo de la trayectoria "S(j)". Esta señal también es puesta en alto de forma periódica por el "Controller" durante el cálculo de "S(j)" en la etapa de pre-calculo.	Bloque "Controller"
Load_S(j)	Esta señal ayuda en el cálculo de cada uno de los elementos de la secuencia "S(j)", cada que se requiere calcular un nuevo valor del vector "S(j)", esta señal es puesta en alto para retroalimentar y utilizar el valor anterior de dicho vector (S(j-1)).	Bloque "Controller"
m1	Esta señal sirve para elegir entre "p" en la etapa de cálculo y "p-1" en la de pre-calculo para la operación módulo.	Bloque "Controller"
Sel_qi	Sirve para controlar en que momentos entrará "qi" al bloque. Los valores de "qi" solo son utilizados durante la etapa de "calculo" del proceso, por lo que "Sel_qi" solo puede estar en alto durante esta etapa.	Bloque "Controller"
Sel_buff	Esta señal, cuando está en alto, sirve para habilitar la entrada proveniente del buffer circular lo que solo es necesario en las operaciones iterativas que se realizan en la etapa de "calculo".	Bloque "Controller"
Ist_Col	Esta señal sirve para controlar un multiplexor en la parte final del proceso de los cálculos que realiza este bloque. Como ya lo mencionamos, este bloque entrega los desplazamientos a lo largo de cada renglón. Cuando esta señal está en alto el bloque "Modulo Computation" entregará un desplazamiento de valor cero que será aplicado al primer elemento de cada renglón.	Bloque "Controller"

qi	Señal que por donde entran a este bloque los elementos del vector "qi".	Bloque "Controller"
p	Señal que contiene al parámetro del mismo nombre "p".	Bloque "p,v LUT"
gx	A través se introducen los valores del parámetro "gx" a este bloque. Este parámetro es utilizado para el cálculo de direcciones solamente cuando el estándar elegido es el 3GPP-LTE.	Bloque "Multi Operations"
Buff_in	A través de esta señal es por donde se realiza la retroalimentación de los valores usados durante las operaciones iterativas en la etapa de "calculo".	Bloque "Circular Buffer"
Salida	**Descripción**	**Destino**
I_addrLTE	Esta señal es utilizada solamente cuando se está trabajando bajo el estándar 3GPP-LTE, y sirve para interconectar a este bloque con el bloque "Multi-Operation", en la Figura 3.20 se puede ver como esta señal conecta a estos bloques.	Bloque "Multi Operations"
Modulo_out	Esta señal es por donde se entregan los resultados de las operaciones realizadas por este bloque.	Bloques "Circular Buffer" y "S(j)RAM".

En la Figura 4.16 se muestra la arquitectura interna del bloque "Modulo Computation", donde los bloques sombreados son los que permiten reconfigurar esta arquitectura para trabajar ya sea para el estándar 3GPP-W CDMA o 3GPP-LTE.

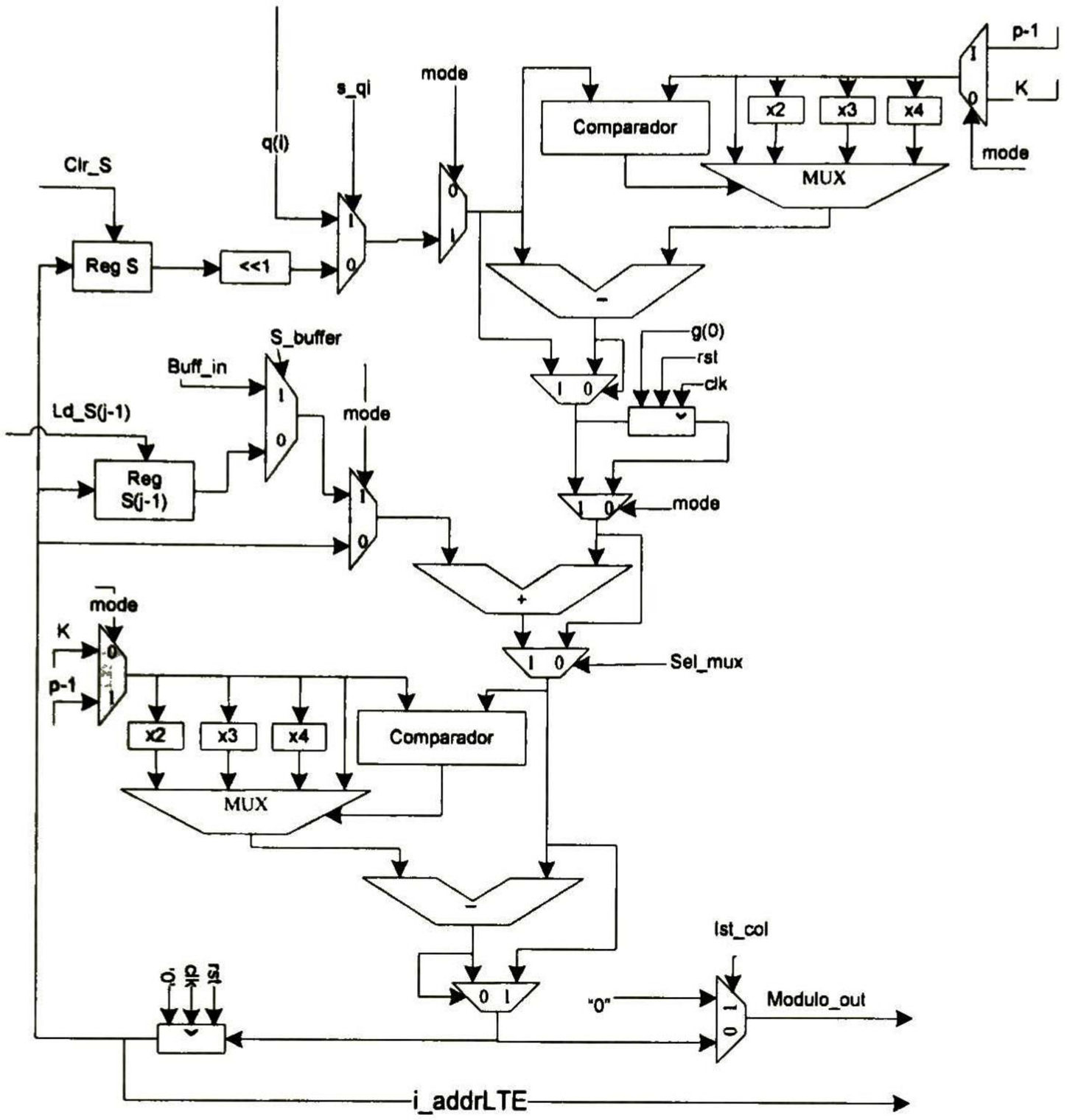

Figura 4.16. Arquitectura interna del bloque "Modulo Computation"

4.1.10 Bloque "Circular Buffer"

Este bloque, es en realidad una serie de registros de corrimiento donde el dato a la entrada es retenido durante 5,10 o 20 ciclos de reloj antes de ser retroalimentado al proceso iterativo para el cálculo de las direcciones de lectura del la "S(j)RAM". Este bloque es uno de los más sencillos de nuestra arquitectura, sin embargo es un bloque muy importante ya que permite que haya una reducción en hardware permitiendo el cálculo de la operación módulo de forma iterativa.

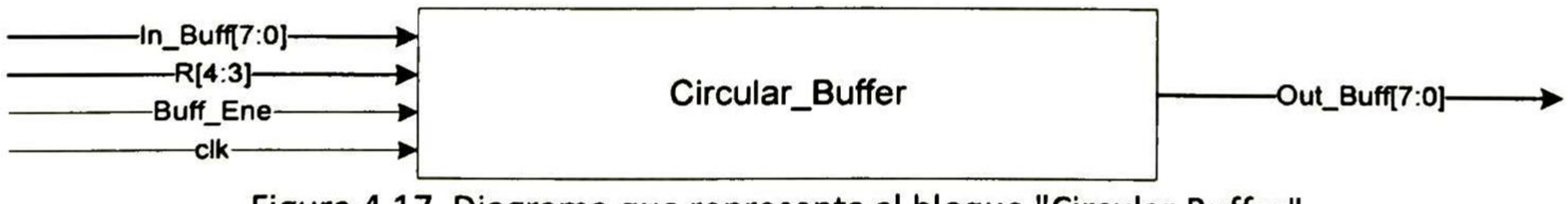

Figura 4.17. Diagrama que representa al bloque "Circular Buffer"

En la Figura 4.17 se muestran las entradas y salidas del bloque "Circular Buffer" las cuales se describen en la Tabla 4.12.

Tabla 4.12. Descripción de las señales de entrada-salida del bloque "Circular Buffer".

Entrada	Descripción	Origen
In_Buff	Señal por donde entran los valores utilizados en operaciones durante la etapa de "calculo". Por esta señal entra un dato por cada ciclo de reloj.	Bloque "Modulo Computation"
R	Por esta señal se introduce el parámetro "R" al buffer, esto ayuda al bloque a determinar el número de registros a través de los cuales pasarán los datos antes de salir del buffer.	Bloque "R,T Logic"
Buff_Ene	Esta señal de entrada sirve para habilitar el funcionamiento de los registros internos, de esta manera cuando el buffer no es necesario no consumirá energía extra.	Bloque "Controller"
clk	Reloj general del sistema.	Exterior
Salida	**Descripción**	**Destino**
Out_Buff	A través de esta señal salen los datos después de R ciclos de que fueron introducidos al buffer.	Bloque "Modulo Computation"

La arquitectura interna del bloque "Circular Buffer" se muestra en la Figura 4.18, de donde observamos que este buffer son simplemente un grupo de registros de corrimiento conectados en serie y un selector de la salida que se debe tomar.

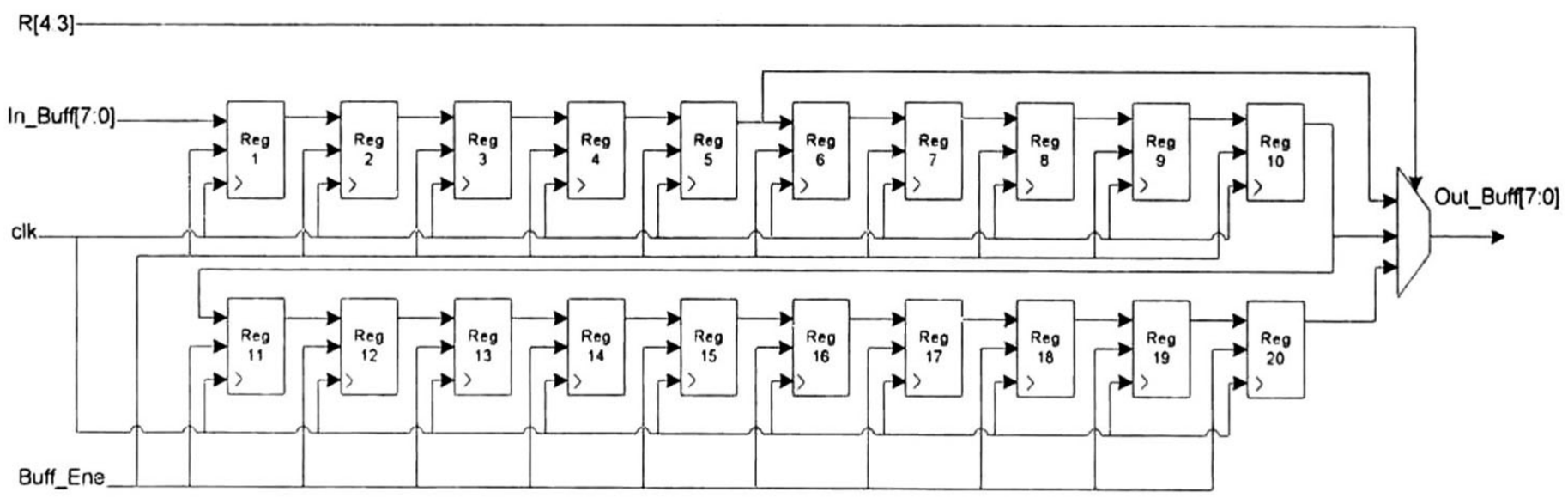

Figura 4.18. Arquitectura interna del bloque "Circular Buffer"

4.2 *Bloque "Data Handling"*

El bloque de manejo de datos, es la parte de la arquitectura del Interleaver, encargada de soportar el flujo de datos. Este bloque aprovecha las direcciones, generadas por el bloque "Interleaver Address Generator" para procesar los datos de entrada según el modo de funcionamiento. Este bloque consta básicamente de memorias RAM y multiplexores. En la Figura 4.19 se muestran los sub-bloques que forman este bloque de manejo de datos, así como la manera en que estos están interconectados.

Las señales que alimentan a cada uno de los sub-bloques mostrados en la Figura 4.19 pueden provenir tanto del exterior como del bloque "Interleaver Address Generator" o bien de algún otro sub-bloque del mismo bloque "Data Handling".

Las dos únicas salidas del Interleavers, mencionadas al principio de este capítulo, son las proporcionadas por el bloque "Data Handling", estas señales son las llamadas "Data_output" y "OutKvalid".

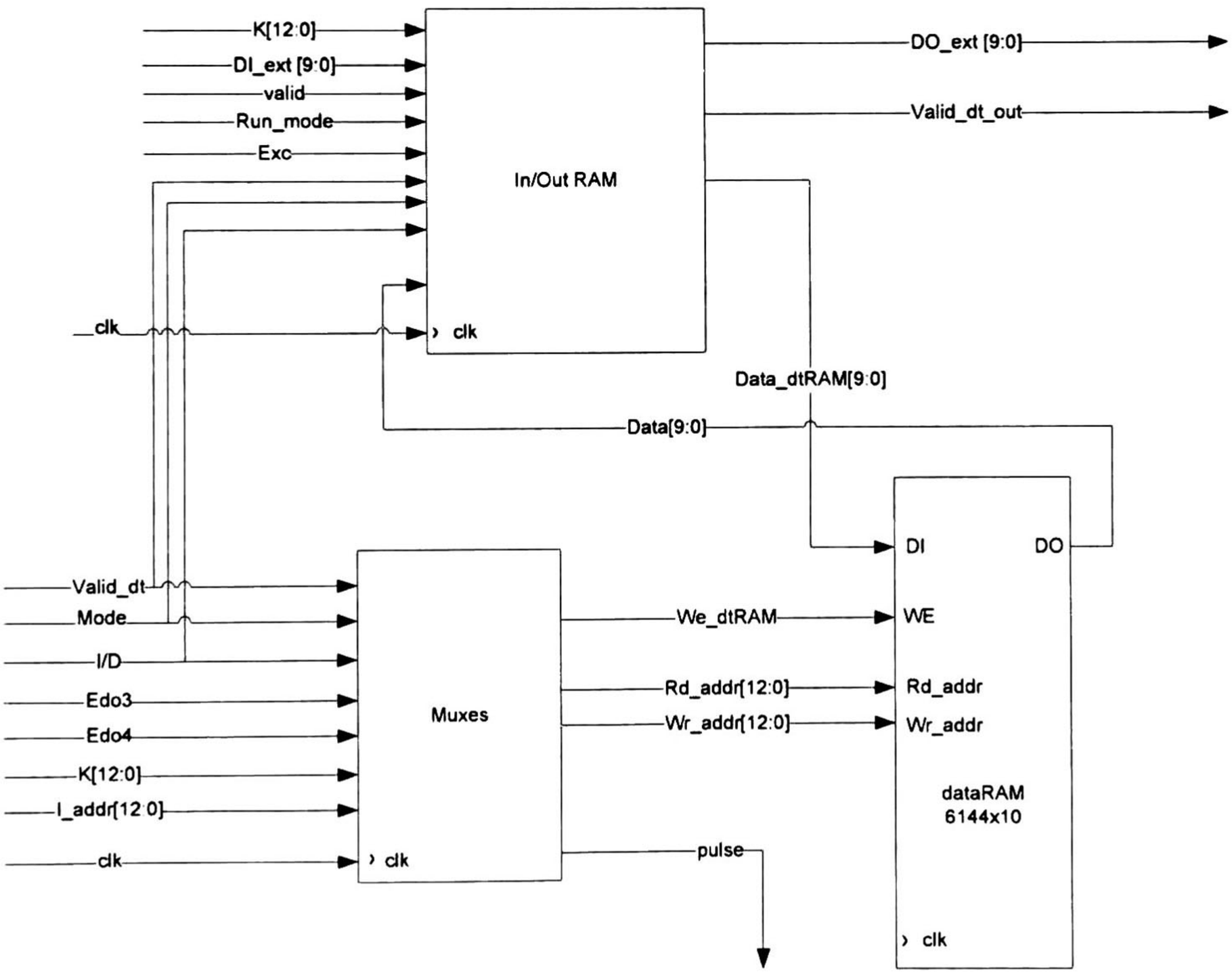

Figura 4.19. Arquitectura del bloque "Data Handling" .

A continuación se hace una descripción de cada uno de los sub-bloques que conforman al bloque "Data Handling".

4.2.1 Bloque "dataRAM"

El bloque "dataRAM" es simplemente una memoria RAM que cuenta con 6144 localidades, cada una de 10 bits. En esta memoria es donde se almacenas los "K" datos del bloque de entrada. Cuando el I/D está funcionando como Interleaver, los datos se deben escribir en esta memoria de forma entrelazada y deben ser leídos de forma secuencial. Cuando el I/D está funcionando como Deinterleaver, los datos se deben escribir de forma secuencial y se deben leer de forma entrelazada.

Idealmente se podría considerar que una vez que los datos son leídos de esta memoria de datos, dichos datos estarían procesados en su totalidad, sin embargo, debido a las excepciones en el estándar 3GPP-W CDMA esto no ocurre así. Para lidiar con las excepciones mencionadas previamente es que se utiliza el bloque "In/Out RAM" mostrado en la Figura 4.19 y que se explicara posteriormente.

En el bloque "dataRAM" tanto la escritura como la lectura se hace de forma síncrona en cada ciclo de reloj, solamente en el caso de la escritura se requiere activar un habilitador para poder escribir en la memoria.

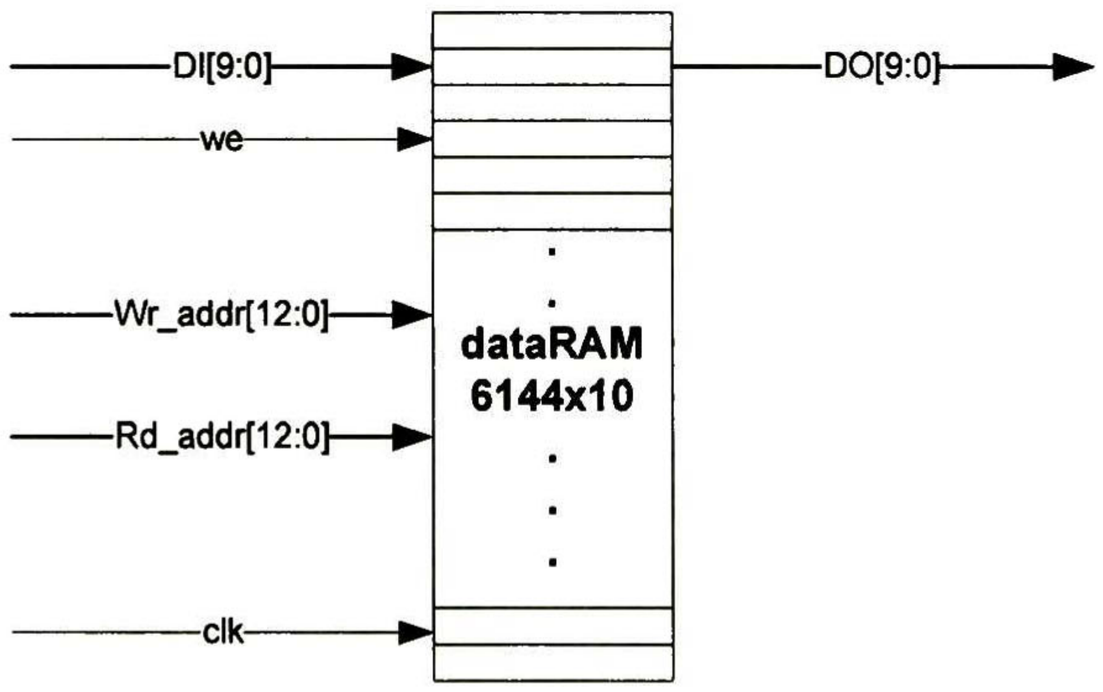

Figura 4.20. Diagrama que representa al bloque "dataRAM"

En la Figura 4.20 se muestra el diagrama que representa al bloque "dataRAM" así como las señales de entrada y salida del mismo. En la Tabla 4.13 se da una descripción de las señales relacionadas al bloque "dataRAM".

Tabla 4.13. Descripción de las señales de entrada-salida del bloque "dataRAM".

Entrada	Descripción	Origen

DI	A través de este puerto de 10 bits se introducen los datos que serán procesados. Los datos introducidos por este puerto son almacenados en la dirección indicada por la dirección "Wr_addr", siempre y cuando esté habilitada la escritura.	Bloque "In/Out RAM"
we	Esta señal cuando está en alto sirve para habilitar la escritura en la RAM.	Bloque "Muxes"
Wr_addr	Esta señal contiene las direcciones de escritura para la RAM de datos y proviene del.	Bloque "Muxes"
Rd_addr	Esta señal contiene las direcciones de lectura de la RAM de datos.	Bloque "Muxes"
clk	Reloj general del sistema	Exterior
Salida	**Descripción**	**Destino**
DO	A través de este puerto se entregan los datos leídos de la RAM de datos.	Bloque "In/Out RAM"

4.2.2 Bloque "Muxes"

El bloque "Muxes" tiene la tarea de entregar las direcciones de escritura y de lectura al bloque "dataRAM" Dentro de este bloque se generan las direcciones secuenciales, y también recibe y direcciona las direcciones generadas por el bloque "Interleaver Address Generator". Cuando el I/D funciona como Interleaver, las direcciones entrelazadas son entregadas como direcciones de escritura para el bloque "dataRAM" y las direcciones secuenciales son utilizadas como direcciones de lectura. Cuando el I/D funciona como Deinterleaver, las direcciones secuenciales son utilizadas como direcciones de escritura para la "dataRAM" y las direcciones de entrelazado son utilizadas como direcciones de lectura.

Con la finalidad de entregar las direcciones de escritura y lectura a la memoria de datos en el momento y forma adecuados, existen varias señales de control que alimenta al bloque "Muxes"

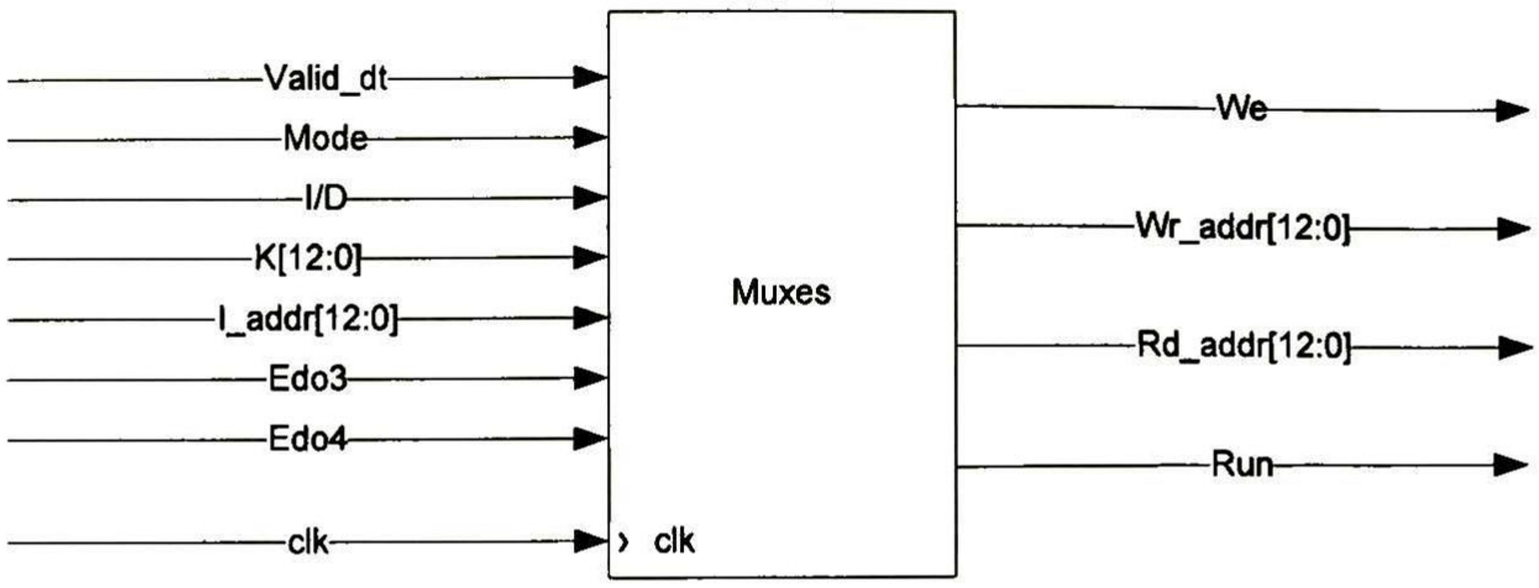

Figura 4.21. Diagrama que representa el bloque "Muxes".

En el diagrama de la Figura 4.21 se muestran las señales de entrada y salida relacionadas con este bloque. En la Tabla 4.14 se hace una descripción de las señales relacionadas al bloque "Muxes"

En la Figura 4.22 se muestra la arquitectura interna del bloque "muxes", en esta arquitectura están los sub-bloques "Counter K" y "Reg".

Tabla 4.14. Descripción de las señales de entrada-salida del bloque "Muxes"

Entrada	Descripción	Origen
Valid_dt	Señal que indica que los datos entrantes al I/D en general son válidos.	Exterior
mode	Señal que indica el estándar en el que el I/D trabajará. En este bloque esta señal también ayuda para decidir en qué momento pedir el inicio de la generación de direcciones de entrelazado.	Exterior
I/D	Esta señal indica si el I/D trabajará como Interleaver o Deinterleaver. Esta señal también sirve para direccionar las direcciones de entrelazado como de escritura o lectura, según corresponda.	Exterior
K	Señal que indica el tamaño del bloque. En este bloque esta señal sirve para limitar el contador que va de 0 a K-1 y que posteriormente servirá como dirección de lectura o escritura.	Exterior
I_addr	Esta señal contiene las direcciones entrelazadas, las cuales pueden pertenecer ya sea al estándar 3GPP-W CDMA o 3GPP-LTE.	Bloque "Multi Operations"
Edo3 Edo4	Estas señales en conjunto con I/D, sirven para indicarle al sub bloque"Counter K" en qué momento deberá empezar a generar	Bloque "Controller"

	direcciones consecutivas.	
clk	Reloj general del sistema.	Exterior
Salida	**Descripción**	**Destino**
We	Esta señal sirve para habilitar la escritura de la memoria de datos. La señal "We" depende únicamente de la señal de entrada "Valid_dt".	Bloque "dataRAM"
Wr_addr	Esta señal entrega las direcciones de escritura de la RAM de datos.	Bloque "dataRAM"
Rd_addr	Esta señal entrega las direcciones de lectura de la RAM de datos.	Bloque "dataRAM"
Run	Esta señal sirve para indicar en qué momento empezar a generar las direcciones de entrelazado cuando se está trabajando en el estándar 3GPP-W CDMA. Esta señal depende directamente de la señal de entrada "Valid_dt".	Bloque "Controller"

El sub-bloque "Counter K" tiene la tarea de generar direcciones consecutivas que van desde 0 hasta K-1, estas direcciones pueden ser utilizadas para escribir o leer en el bloque "dataRAM" según se requiera.

Cuando el modo de operación del I/D es 3GPP-W CDMA la generación de las direcciones secuenciales depende de las señales I/D, Edo3, Valid_dt. Cuando la arquitectura está configurada como Interleaver, el conteo secuencial que será usado para lectura, comienza al terminar el estado 3 (Edo3). Cuando la arquitectura está configurada como Deinterleaver, el conteo secuencial empieza cuando la señal "Valid_dt" se pone en alto.

Cuando el modo de operación es el del estándar 3GPP-LTE, el comienzo del conteo secuencial depende de las señales I/D y Valid_dt. Asi pues, para el proceso de entrelazado, el conteo secuencial comienza cuando la señal "Valid_dt" baja, y en el caso del de-entrelazado, el conteo secuencial comienza cuando "Valid_dt" se pone en alto.

El sub-bloque "Reg" son simplemente registros que generan las señales "We" y "Run" en los momentos adecuados auxiliándose de algunas señales.

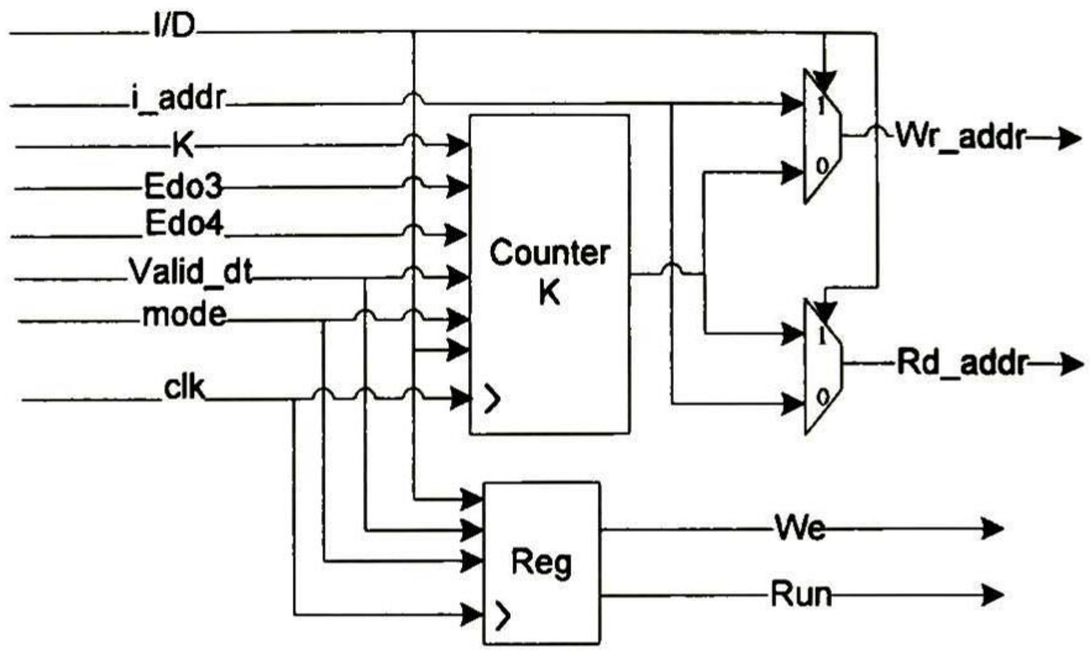

Figura 4.22. Arquitectura del bloque "Muxes".

4.2.3 Bloque "In/Out RAM"

El bloque "In/Out RAM" es un bloque ideado para tratar con las direcciones de entrelazado inválidas que se generan en el estándar 3GPP-W CDMA. El bloque "In/Out RAM" trabaja como una FIFO que retiene los datos relacionados con direcciones de entrelazado inválidas. Cuando se realiza un entrelazado de datos, estos son pre-procesados por este bloque antes de ir a al bloque "dataRAM", en seguida, los datos leídos de la "dataRAM" son pasados por un "bypass" interno del bloque "In/Out RAM" y luego estos datos son entregados al exterior ya procesados. Por otro lado cuando re desea realizar un de-entrelazado de datos, estos son pasados por el "bypass" interno del bloque "In/Out RAM" para luego ser entregados al bloque "dataRAM" de donde son leídos y nuevamente pasados por el bloque "In/Out RAM" donde se le aplicará la última parte del proceso de de-entrelazado antes de ser entregados al exterior.

Este bloque consta básicamente de una memoria RAM de 256 localidades de 10 bits cada una (la cual funciona como una FIFO), un bloque para controlar las direcciones de lectura y de escritura de esta RAM en función de las direcciones de entrelazado generadas y los modos de operación, y una serie de multiplexores para direccionar el flujo de datos a través de este bloque.

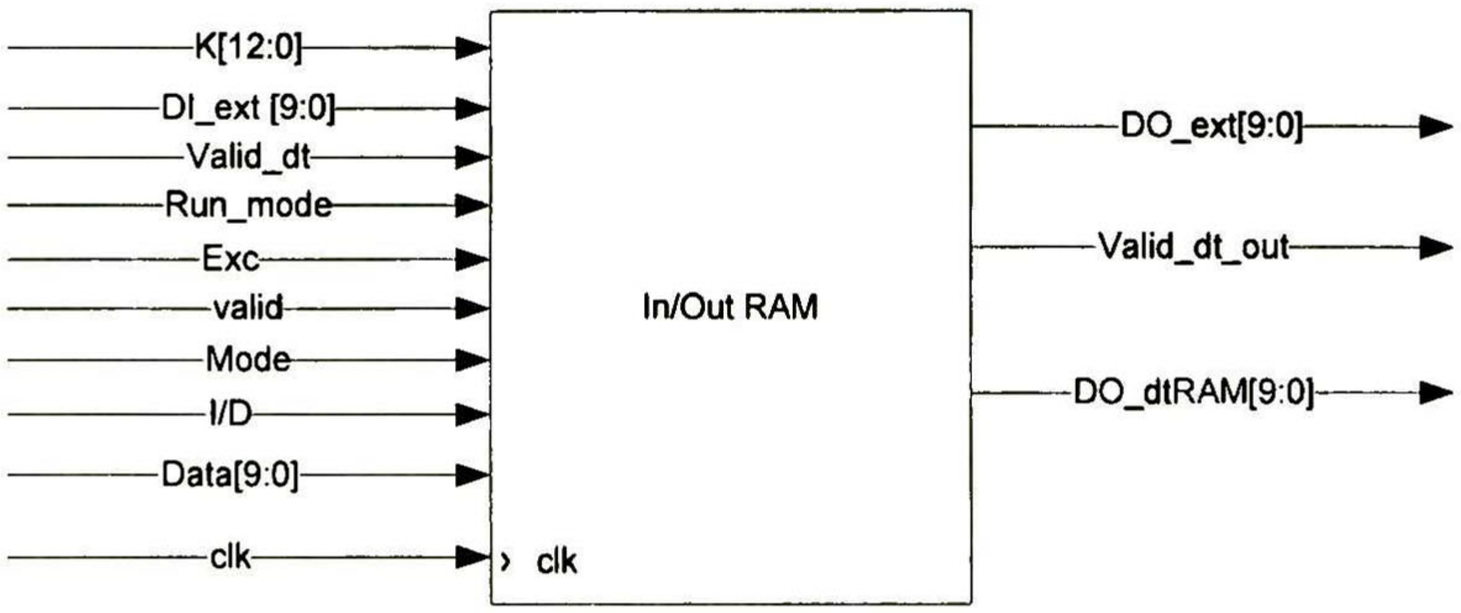

Figura 4.23. Diagrama que representa al bloque "In/Out RAM".

En la Figura 4.23 se muestra el bloque que representa al bloque "In/Out RAM", en este diagrama también se muestran las señales relacionadas con este bloque las cuales de describen en la Tabla 4.15.

Tabla 4.15. Descripción de las señales de entrada-salida al bloque "In/Out RAM".

Entrada	Descripción	Origen
K	Señal que indica el tamaño del bloque a ser procesado. Esta señal sirve para indicar cuantos datos de entrada deberán ser retenidos antes de pasar a la RAM de datos.	Exterior
DI_ext	Esta señal transporta cada uno de los K datos entrantes al I/D, a través de esta señal entra un dato por cada ciclo de reloj durante un periodo de K ciclos.	Exterior
Valid_dt	Esta señal cuando está en alto indica que los datos entrantes al I/D y al bloque "In/Out RAM" son validos y que se deberán procesar.	Exterior
Run_mode	Esta señal sirve para indicar al bloque en qué momento se están generando las direcciones de entrelazado.	Bloque "Controller"
Exc	Esta señal indica cuando existen excepciones y entonces el bloque "In/Out RAM" tomará las debidas correcciones.	Bloque "Controller"
valid	Cuando se generan direcciones de entrelazado invalidas, esta señal ayuda a retener los datos relacionados con dichas direcciones mientras se generan direcciones validas.	Bloque "Multi Operations"
Mode	Esta señal nos ayuda a controlas el multiplexor de bypass, de tal manera que cuando se está trabajando bajo el estándar 3GPP-LTE, el bloque "In/Out RAM" quedará deshabilitado.	Exterior
I/D	Esta señal indica si la "In/Out RAM" procesará los datos antes de entrar a la RAM de datos (I/D="1") o después de salir de la misma (I/D="0").	Exterior
Data	Por esta señal entran los datos provenientes de la "dataRAM" cuando el bloque "I/D" trabaja como Deinterleaver en el estándar 3GPP-W CDMA.	Bloque "dataRAM"
clk	Reloj general del sistema.	Exterior
Salida	**Descripción**	**Destino**
DO_ext	Señal que entrega el bloque de datos de tamaño "K" ya	Exterior

	procesado.	
Valid_dt_out	Señal que indica que los datos que salen a través de la señal "DO_ext" son validos.	Exterior
DO_dtRAM	Señal que entrega los datos de entrada después de ser pasados a través del bloque "In/Out RAM"	Bloque "dataRAM"

En la Figura 4.24 se muestra la arquitectura interna del bloque "In/Out RAM". En esta misma figura se puede ver el bloque "Add Gen" que no es otra cosa que un par de contadores que dependen de varias banderas para incrementarse o reiniciarse.

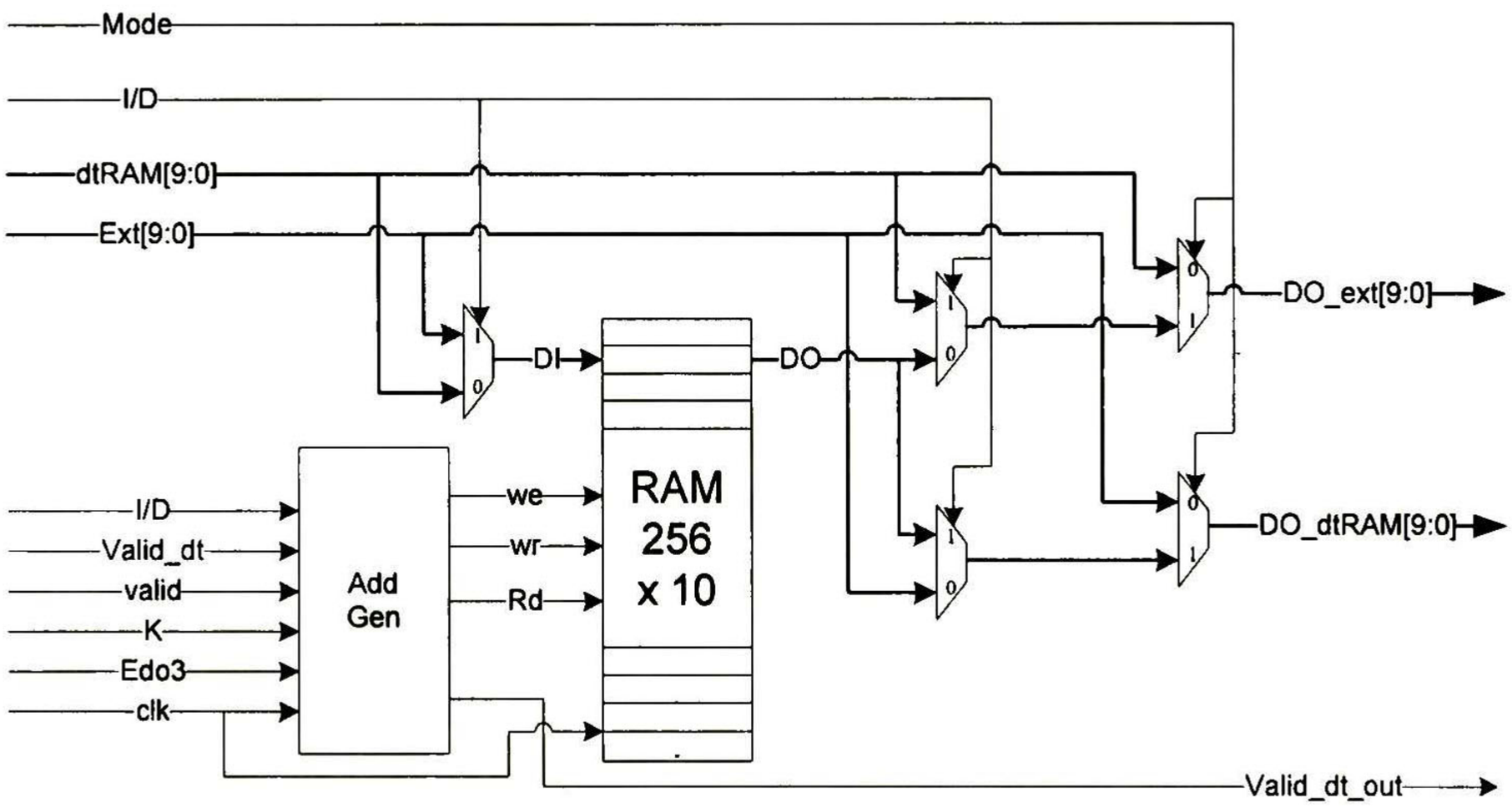

Figura 4.24. Arquitectura interna del bloque "In/Out RAM"

Capítulo 5

Pruebas y Resultados

En este capítulo se menciona el proceso de pruebas por el que pasó cada bloque de nuestro diseño antes de tener la arquitectura final y funcional. Durante el diseño de cada bloque se buscó aproximar el comportamiento de nuestro diseño del I/D lo más posible a lo que el estándar correspondiente dicta.

Para hacerle las pruebas necesarias a nuestro diseño, o a cualquier diseño en general, es necesario utilizar alguna metodología de pruebas, en este capítulo se presenta la metodología elegida para nuestros módulos a fin de comprobar la funcionalidad de los mismos, cabe mencionar que este plan de verificación fue elegido en función de las especificaciones de los estándares 3GPP-W CDMA y 3GPP-LTE.

En este capítulo también se muestran los resultados de la síntesis de cada uno de los bloques que conforman nuestro diseño y algunos resultados que muestran el desempeño del mismo. Para poder hacer todo lo mencionado nos auxiliamos en el uso de algunas herramientas como es el caso de MATLAB, Quartus II y ModelSim.

5.1 Metodología de pruebas: Verificación funcional.

La verificación funcional es el proceso que comprueba el correcto funcionamiento de un dispositivo, mediante la inserción de estímulos de entrada y el análisis de los datos salientes a dicho dispositivo. Este tipo de análisis es una evaluación de las diversas características que presenta el dispositivo y que se obtienen de los requerimientos de diseño.

Dentro de la verificación funcional se deben definir las pruebas que se le harán al "Device Under Test" (DUT), en función de sus características y funciones.

En el caso de nuestro diseño, introducimos bloques de datos de tamaños variables "K" para cada uno de los siguientes 4 casos:

1. Cuando nuestro diseño esté configurado como Interleaver bajo el estándar 3GPP-W CDMA.

2. Cuando nuestro diseño esté configurado como Deinterleaver bajo el estándar 3GPP-W CDMA.
3. Cuando nuestro diseño esté configurado como Interleaver bajo el estándar 3GPP-LTE.
4. Cuando nuestro diseño esté configurado como Deinterleaver bajo el estándar 3GPP-LTE.

En cada uno de los casos anteriores el tamaño del bloque depende del estándar en el que se esté trabajando y la elección del tamaño de los mismos se tomaron al azar, y con la ayuda de MATLAB se generaron los paquetes de datos sin procesar así como los ya procesados.

El archivo con los datos sin procesar se entrega a la cama de prueba que los hace pasar a través del DUT junto con otros estímulos previamente definidos, obteniendo de esta manera un nuevo archivo con los datos procesados por el DUT. El archivo de datos procesados generados por MATLAB (Golden Values) se compara, dentro de MATLAB, con el archivo de datos procesados generados por el DUT.

En la Figura 5.1 se muestra la arquitectura de la metodología de pruebas que utilizamos. Durante la etapa de pruebas nos apoyamos en las herramientas ModelSim y MATLAB.

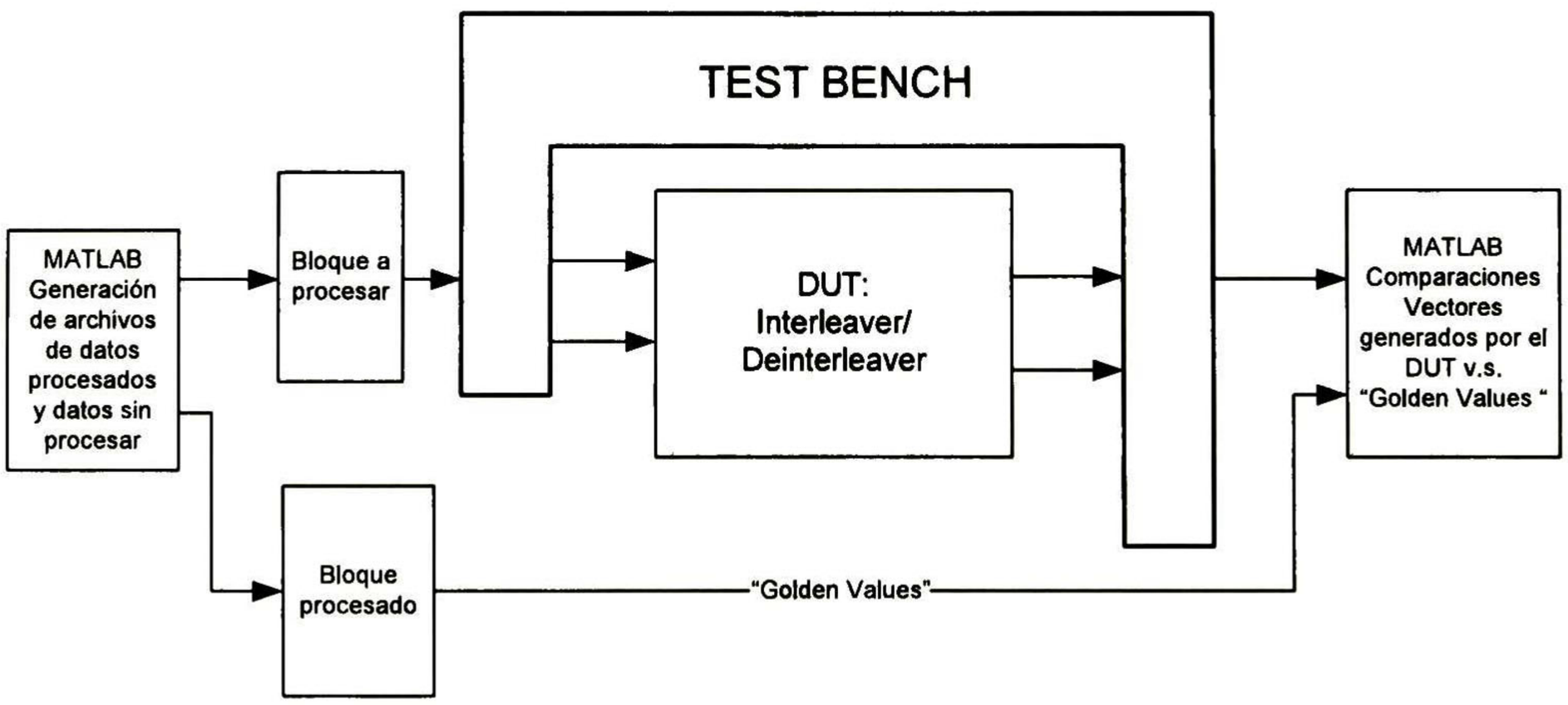

Figura 5.1. Diagrama a bloques que representa la metodología de pruebas utilizada con el DUT "I/D"

Finalmente, al hacer las comparaciones en MATLAB de los vectores correspondientes de datos procesados podemos validar la funcionalidad de nuestro diseño.

5.2 Generación de direcciones de entrelazado ("Golden Values")

Con la finalidad de tener valores de referencia confiables contra que comparar los resultados arrojados por nuestro diseño, se utilizó un programa en MATLAB que genera todos los parámetros necesarios para la generación de direcciones de entrelazados, asi como dichas

direcciones. Los valores generados por este programa son los que tomaremos como "Golden Values" que son valores correctos de referencia.

5.2.1 3GPP-W CDMA

Mediante el uso de MATLAB y las formulas de permutación del estándar podemos obtener cada uno de los parámetros necesarios para lograr generar las direcciones de entrelazado.

Una vez que hicimos un programa en MATLAB para generar las direcciones de entrelazado observamos el efecto que el entrelazado tiene en una secuencia de datos para distintos tamaños de bloques "K"

Tabla 5.1. Relación entre datos de entrada y direcciones entrelazadas de destino para un tamaño de bloque K=40 para el estándar 3GPP-W CDMA.

Dato Numero:	Dirección Destino	Dato Numero:	Dirección Destino	Dato Numero:	Dirección Destino	Dato Numero:	Dirección Destino
1	39	11	34	21	36	31	32
2	25	12	26	22	28	32	24
3	17	13	20	23	18	33	16
4	9	14	10	24	12	34	8
5	1	15	4	25	2	35	0
6	35	16	38	26	37	36	33
7	27	17	30	27	29	37	31
8	21	18	22	28	19	38	23
9	11	19	14	29	13	39	15
10	5	20	6	30	3	40	7

De la Tabla 5.1 podemos darnos cuenta como una secuencia de datos de entrada se reordena de manera entrelazada y sin repetir direcciones. Viendo lo anterior de forma grafica en la Figura 5.2 podemos observar de la manera en que se dispersa la secuencia de entrada de los K datos consecutivos.

Si utilizamos el mismo programa en MATLAB para entrelazar diferentes tamaños de bloques, para cada uno obtendremos un vector con la trayectoria de direcciones de entrelazado correpondiente.

En las Figura 5.3 y Figura 5.4 se muestran ejemplos de la dispersión que sufrirían bloques de datos de K=120 y K=4800 al utilizar las direcciones de entrelazado generadas en MATLAB para el estándar 3GPP-W CDMA.

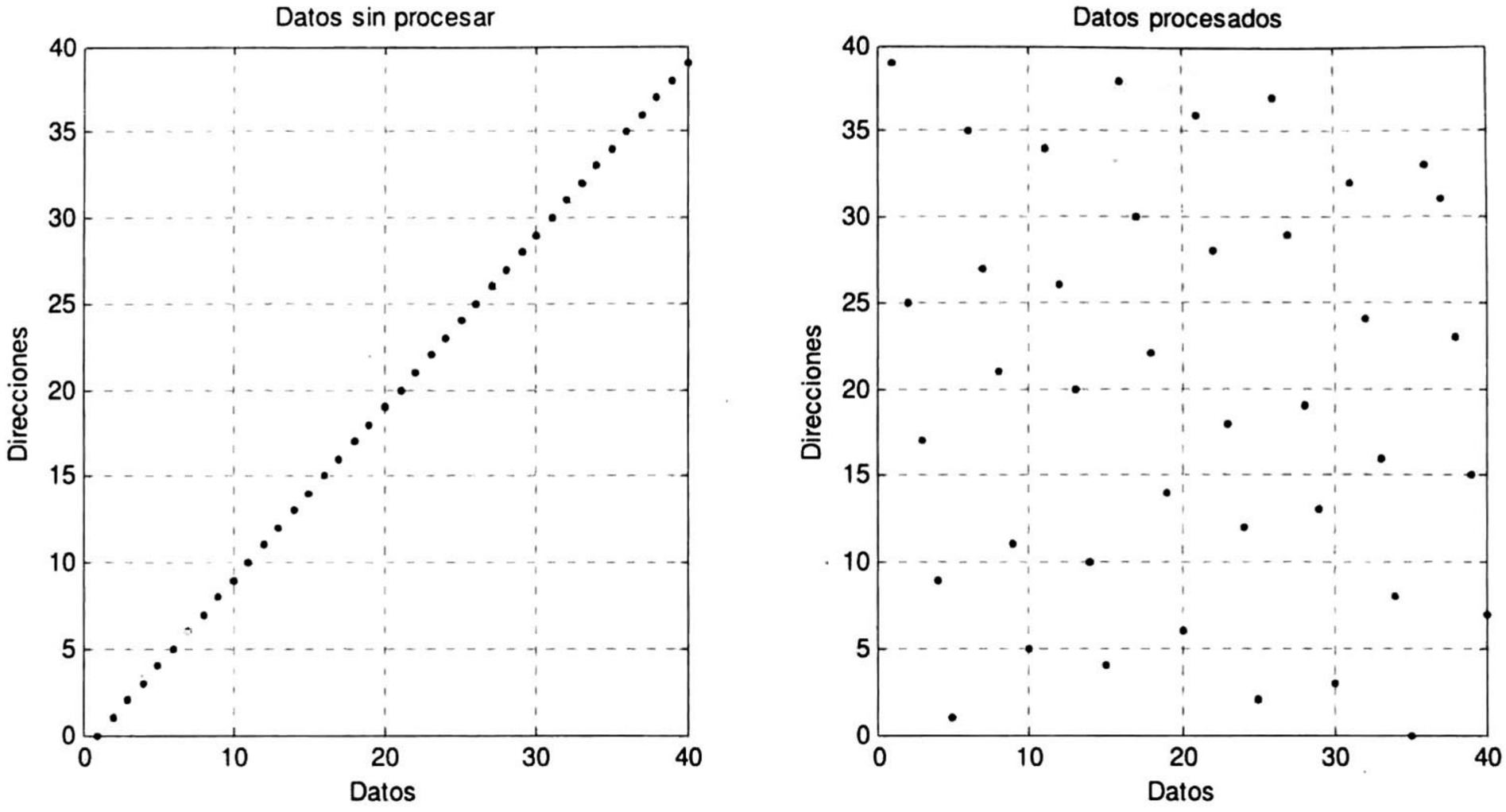

Figura 5.2. Comparación del orden de la secuencia de datos sin entrelazar (izquierda) contra el orden de datos ya entrelazados (derecha) para un tamaño de bloque K=40.

Figura 5.3. Comparación del orden de la secuencia de datos sin entrelazar (izquierda) contra el orden de datos ya entrelazado (derecha) para un tamaño de bloque K=120.

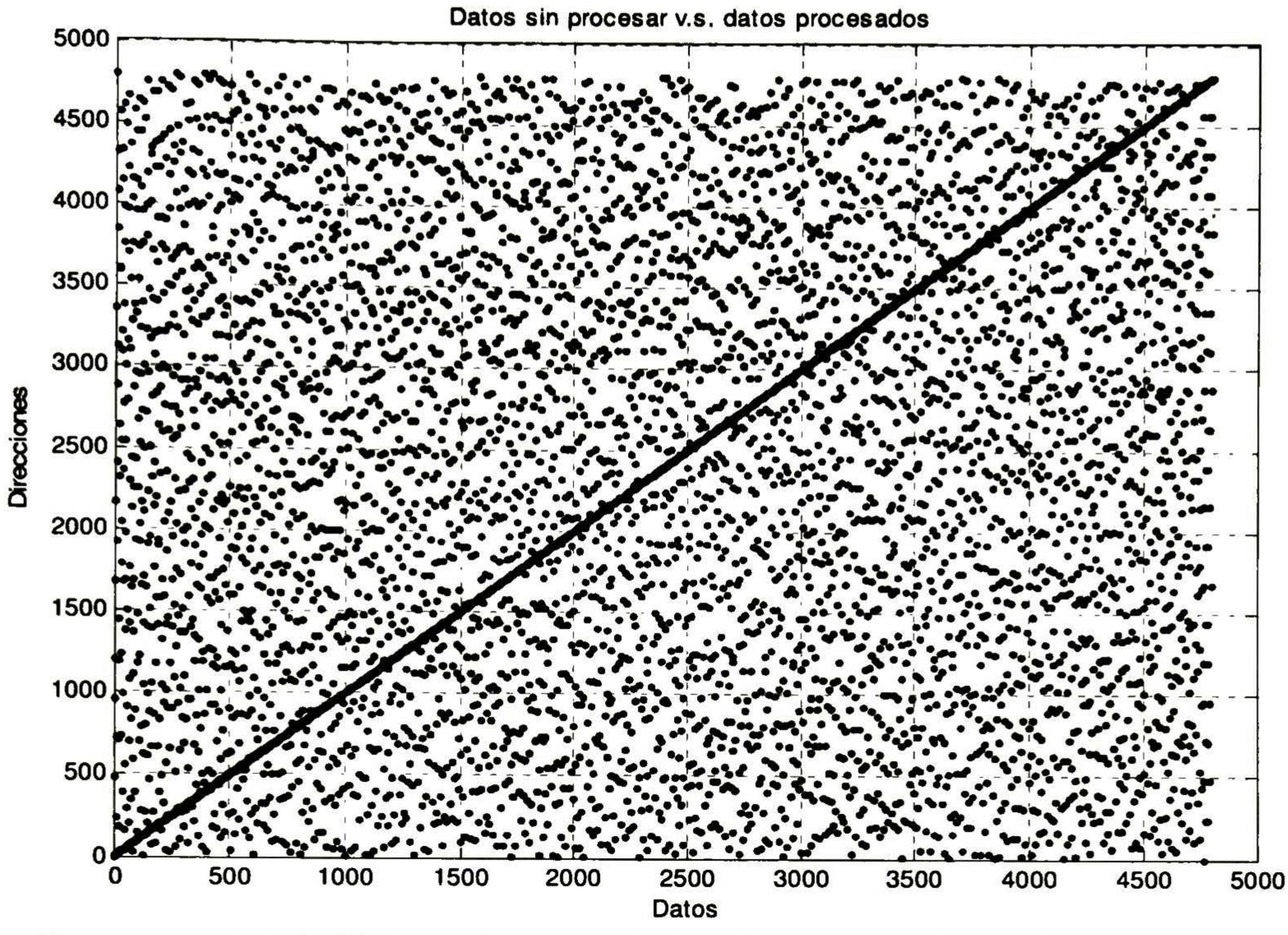

Figura 5.4. Comparación del orden de la secuencia de datos sin entrelazar contra el orden de datos ya entrelazados para un tamaño de bloque K=4800.

5.2.2 3GPP-LTE

Para el caso del estándar 3GPP-LTE la generación de direcciones mediante el uso de MATLAB se hace mucho más sencillo ya que la trayectoria de entrelazado se rige solo con una formula del tipo cuadrático.

Si hacemos un entrelazado de datos para un tamaño de bloque K=40 obtenemos lo que se muestra en la Tabla 5.2.

Tabla 5.2. Relación entre datos de entrada y direcciones entrelazadas de destino para un tamaño de bloque K=40 para el estándar 3GPP-LTE.

Dato numero:	Dirección destino	Dato numero:	Dirección destino	Dato numero:	Dirección destino	Dato numero:	Dirección destino
1	13	11	3	21	33	31	23
2	6	12	36	22	26	32	16
3	19	13	9	23	39	33	29

4	12	14	2	24	32	34	22
5	25	15	15	25	5	35	35
6	18	16	8	26	38	36	28
7	31	17	21	27	11	37	1
8	24	18	14	28	4	38	34
9	37	19	27	29	17	39	7
10	30	20	20	30	10	40	0

Para tener más claro el efecto del entrelazado de datos podemos observar las secuencias de la Figura 5.5 que representa lo expresado en la tabla 5.2.

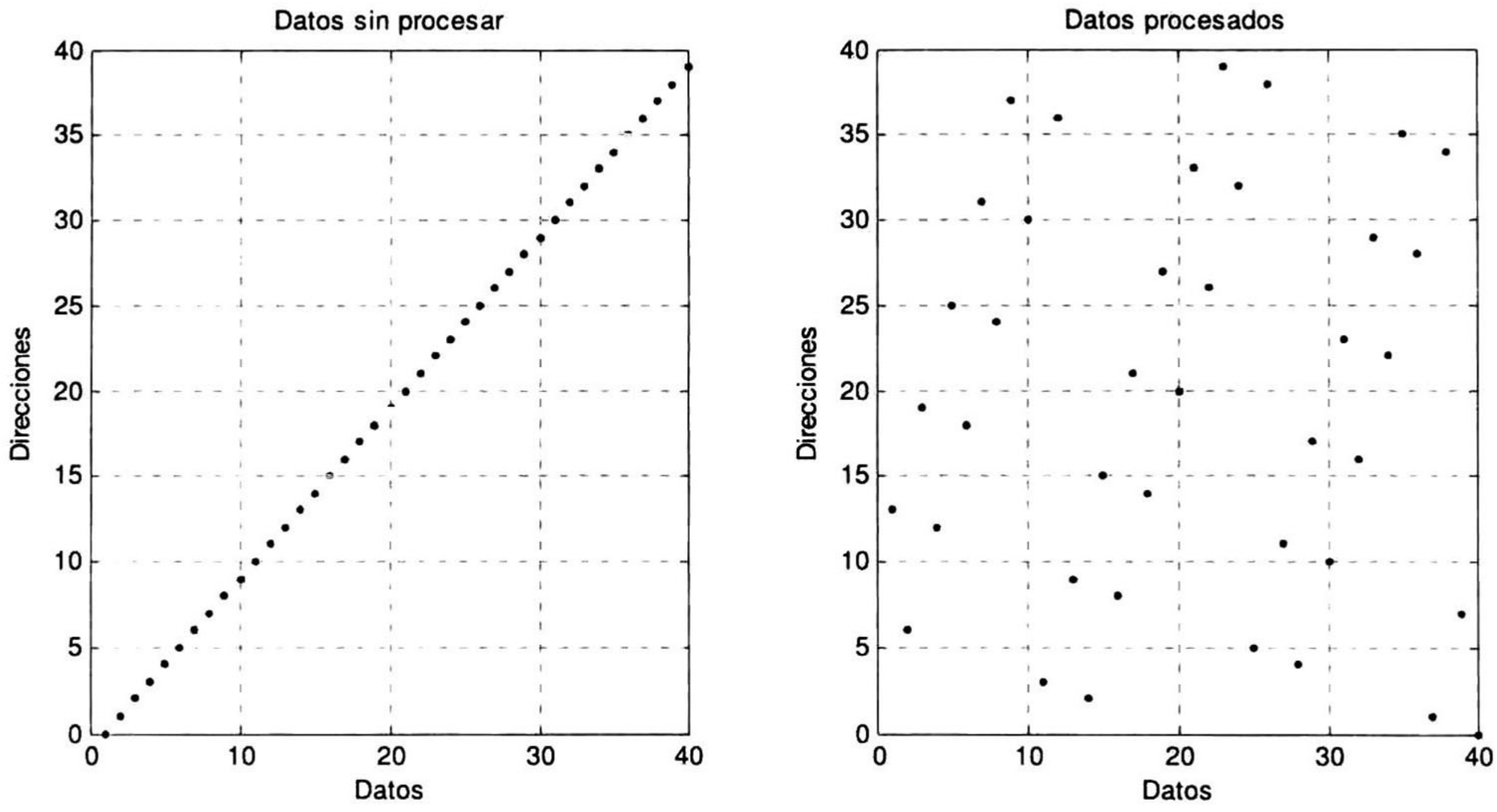

Figura 5.5. Comparación del orden de la secuencia de datos sin entrelazar (izquierda) contra el orden de datos ya entrelazados (derecha) para un tamaño de bloque K=40.

En la Figura 5.5 se observa cómo se dispersa la secuencia de datos, si comparamos con la Figura 5.2 que también corresponde para un tamaño de bloque K=40 pero para el estándar 3GPP-W CDMA podemos notar la diferencia en el patrón de dispersión.

Si aplicamos el proceso de entrelazado a bloque de tamaños K=120 y K=4800 bajo el estándar 3GPP-LTE obtendremos las secuencias mostradas en gráficamente en las Figura 5.6 y Figura 5.7 respectivamente.

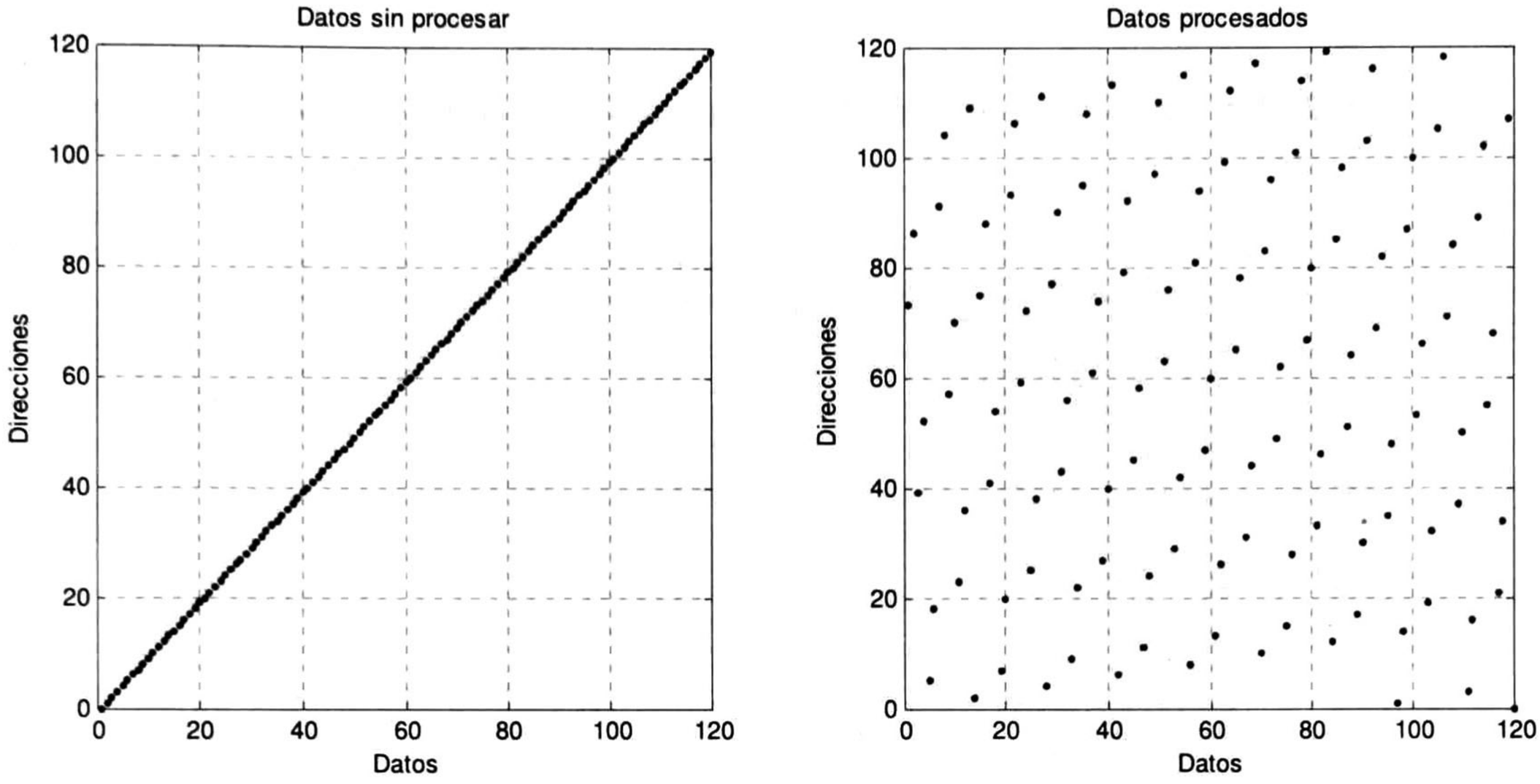

Figura 5.6. Comparación del orden de la secuencia de datos sin entrelazar (izquierda) contra el orden de datos ya entrelazados (derecha) para un tamaño de bloque K=120.

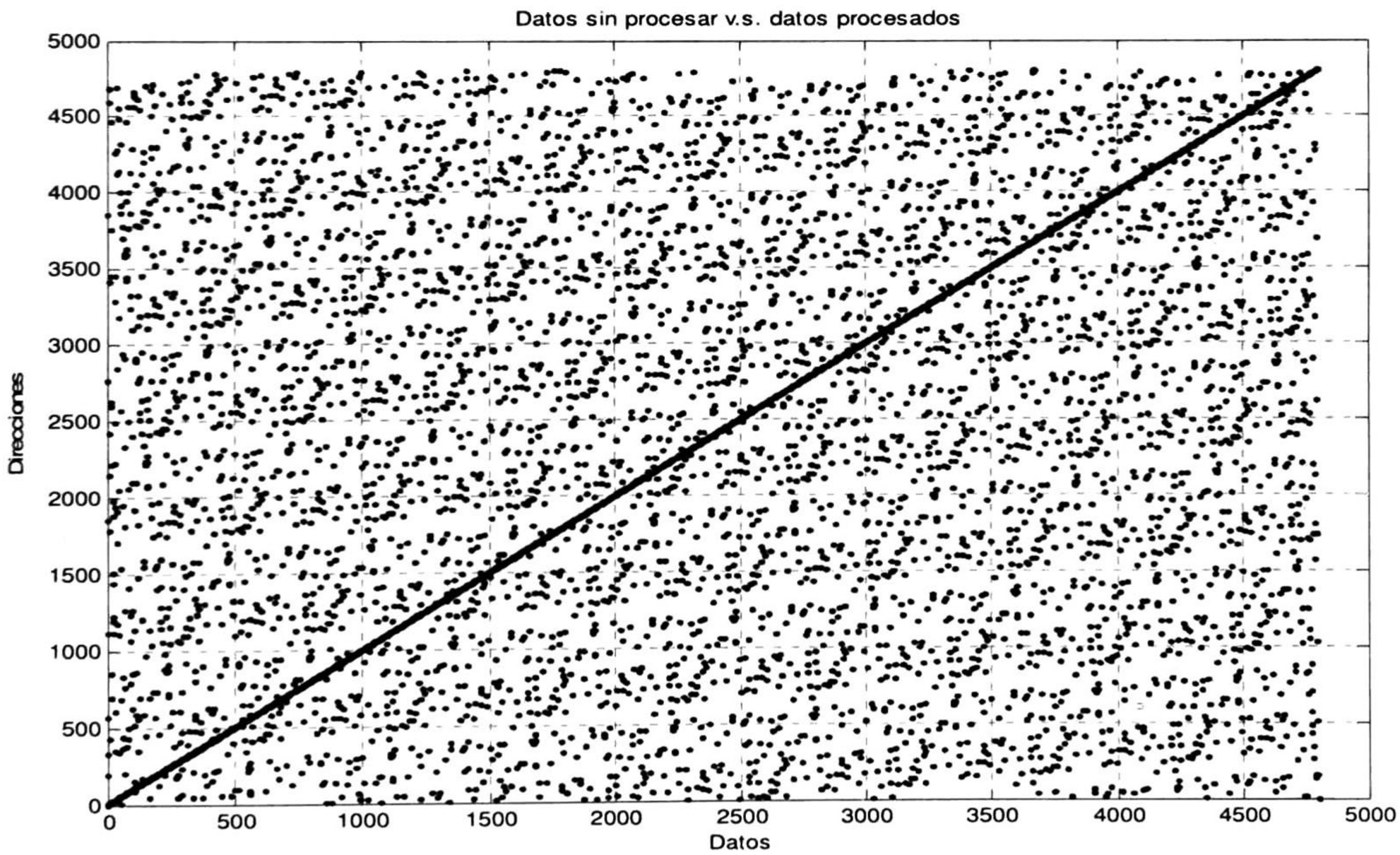

Figura 5.7. Comparación del orden de la secuencia de datos sin entrelazar contra el orden de datos ya entrelazados para un tamaño de bloque K=4800.

5.3 Verificación de parámetros generados por cada bloque

Con la finalidad de encontrar cualquier posible error en la generación de direcciones de entrelazado, fue que se optó por verificar a forma de monitoreo los resultados de cada uno de los bloques que conforman el I/D comparando los resultados de MATLAB con los valores generados en simulaciones en ModelSim de nuestro diseño en HDL para cada módulo. Aunque se hicieron varias pruebas para diferentes tamaños de bloque, los resultados que a continuación se presentan son para valores de K pequeños a fin de que puedan ser apreciados los resultados.

5.3.1 3GPP-W CDMA

Basándonos en un ejemplo para un tamaño de bloque K=48 para comparar los valores de los parámetros necesarios obtenemos los resultados mostrados a continuación, donde en cada figura los valores de interés se encuentran encerrados en óvalos o rectángulos.

5.3.1.1 Parámetro R:

Para obtener el parámetro "R" utilizamos la parte combinacional del bloque "R,T Logic" que solo depende del tamaño del bloque "K", por lo que al tener un valor fijo de K (en este caso K=48) tendremos un valor de R (en este caso R=5). En la Figura 5.8 podemos ver encerrado en un ovalo blanco el resultado del valor obtenido por simulación de nuestro bloque en HDL para el valor de R, de acuerdo a los resultados mostrados en MATLAB este resultado es correcto.

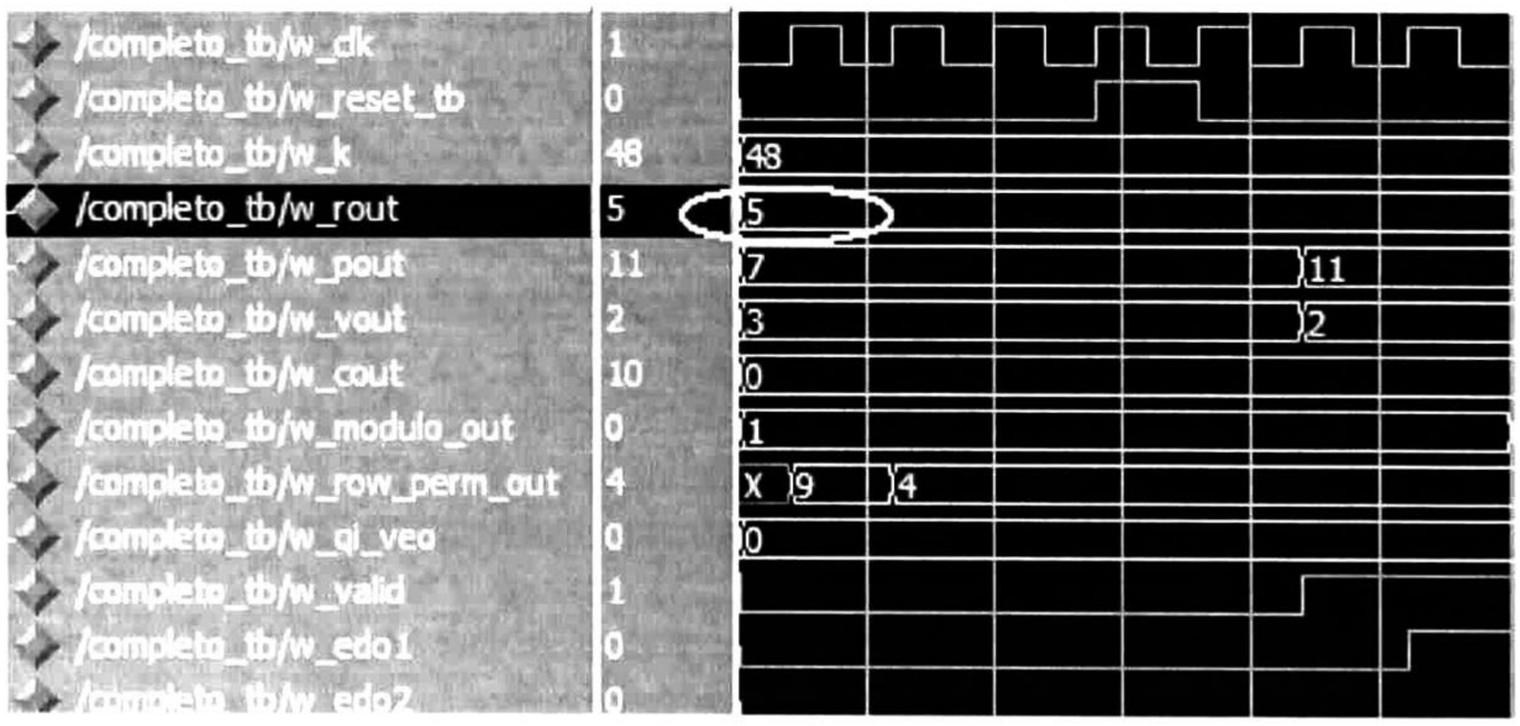

Figura 5.8. Simulación en ModelSim que muestra el valor obtenido de R=5 para K=48.

5.3.1.2 P,v LUT:

Para obtener los parámetros "p" y "v" de la LUT correspondiente, justo después del RESET, se inicia el contador que direcciona la LUT, este contador se incrementa en cada ciclo de

reloj direccionando nuevos valores de p y v, cuando se tienen los valores correctos de estos parámetros la bandea "valid" se pone en alto y el contador se queda fijo. Como se puede ver en la Figura 5.9 que muestra algunos resultados de la simulación en ModelSim de nuestro diseño, los valores obtenidos de p y v son 11 y 2 respectivamente, que según nuestras simulaciones en MATLAB son los resultados esperados.

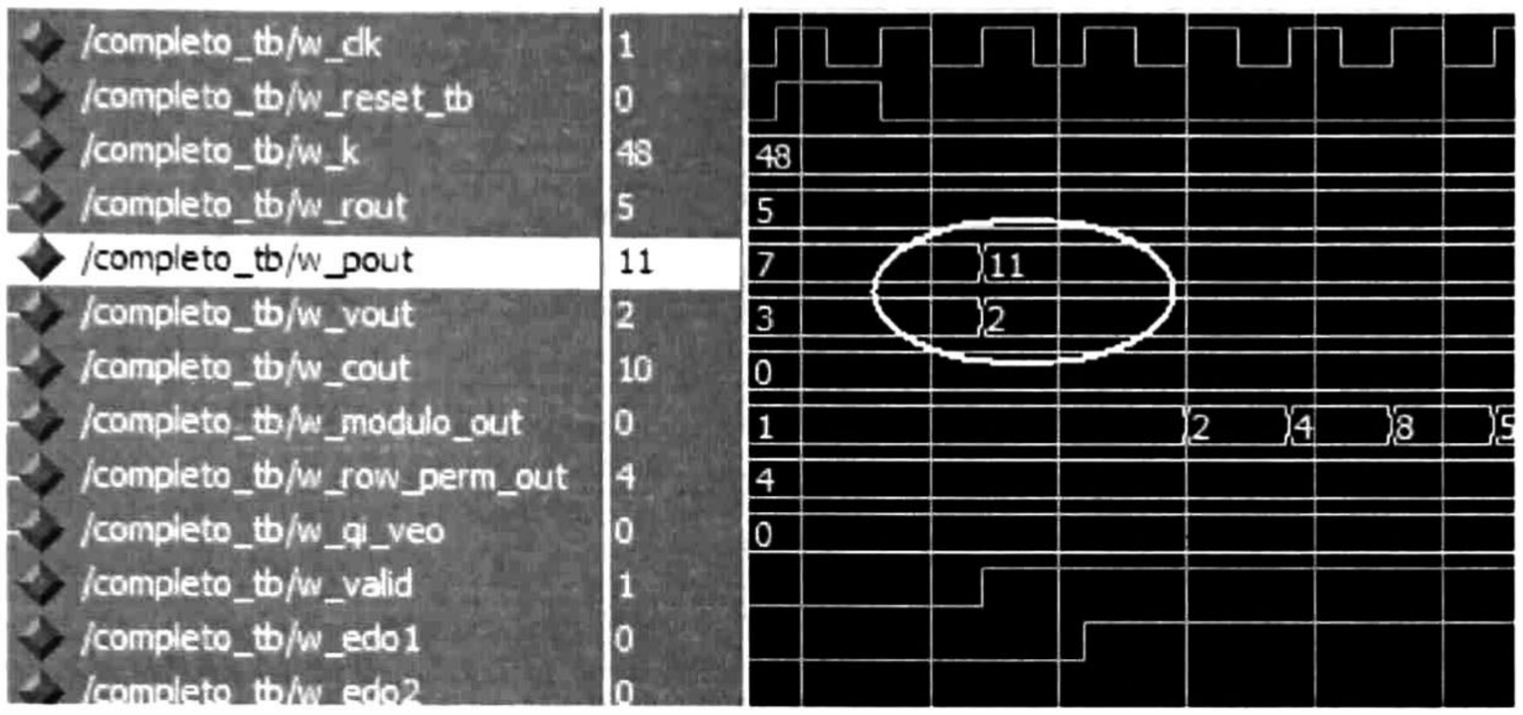

Figura 5.9. Simulación en ModelSim que muestra los valores obtenidos, p=11 y v=2 del bloque "p,v LUT" para K=48.

5.3.1.3 Secuencia S(j):

El bloque "Modulo Computation" es el encargado de calcular esta secuencia base de permutación y arranca en el estado 1, después de tener el valor de "p". Los resultados obtenidos para este vector en la simulación hecha en ModelSim se muestran en la Figura 5.10 Según lo obtenido en MATLAB el vector S(j)= [1, 4, 8, 5, 10, 9, 7, 3, 6], que es el mismo generado por nuestro bloque.

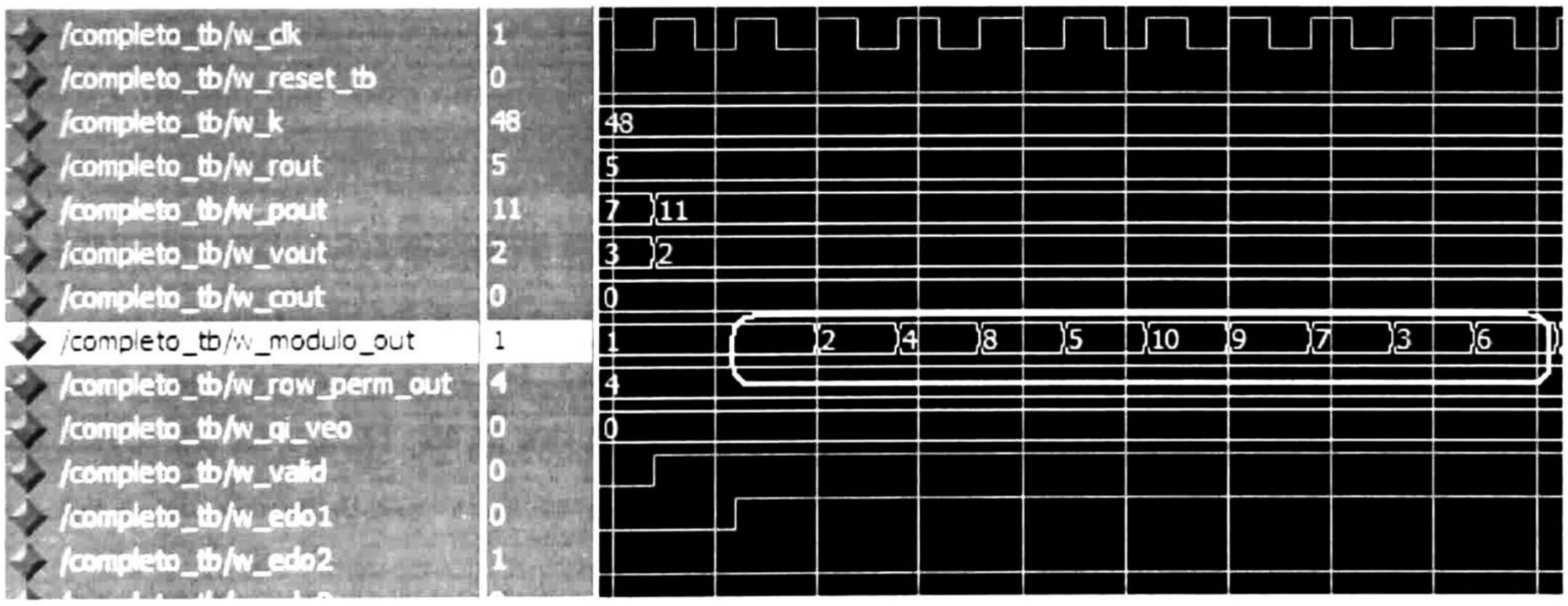

Figura 5.10. Simulación en ModelSim que muestra los valores obtenidos del vector S(j) en el estado 1 para K=48.

5.3.1.4 Parámetro C:

Para obtener el valor de C se requiere previamente tener los valores de "p" y "R". Durante el estado 2 se comienza a calcular el valor de C, y una vez que este es el correcto, la bandera "valid" se pone en alto. En la Figura 5.11 observamos el resultado de las simulaciones para K=48, donde C=10 que corresponde a lo obtenido en MATLAB.

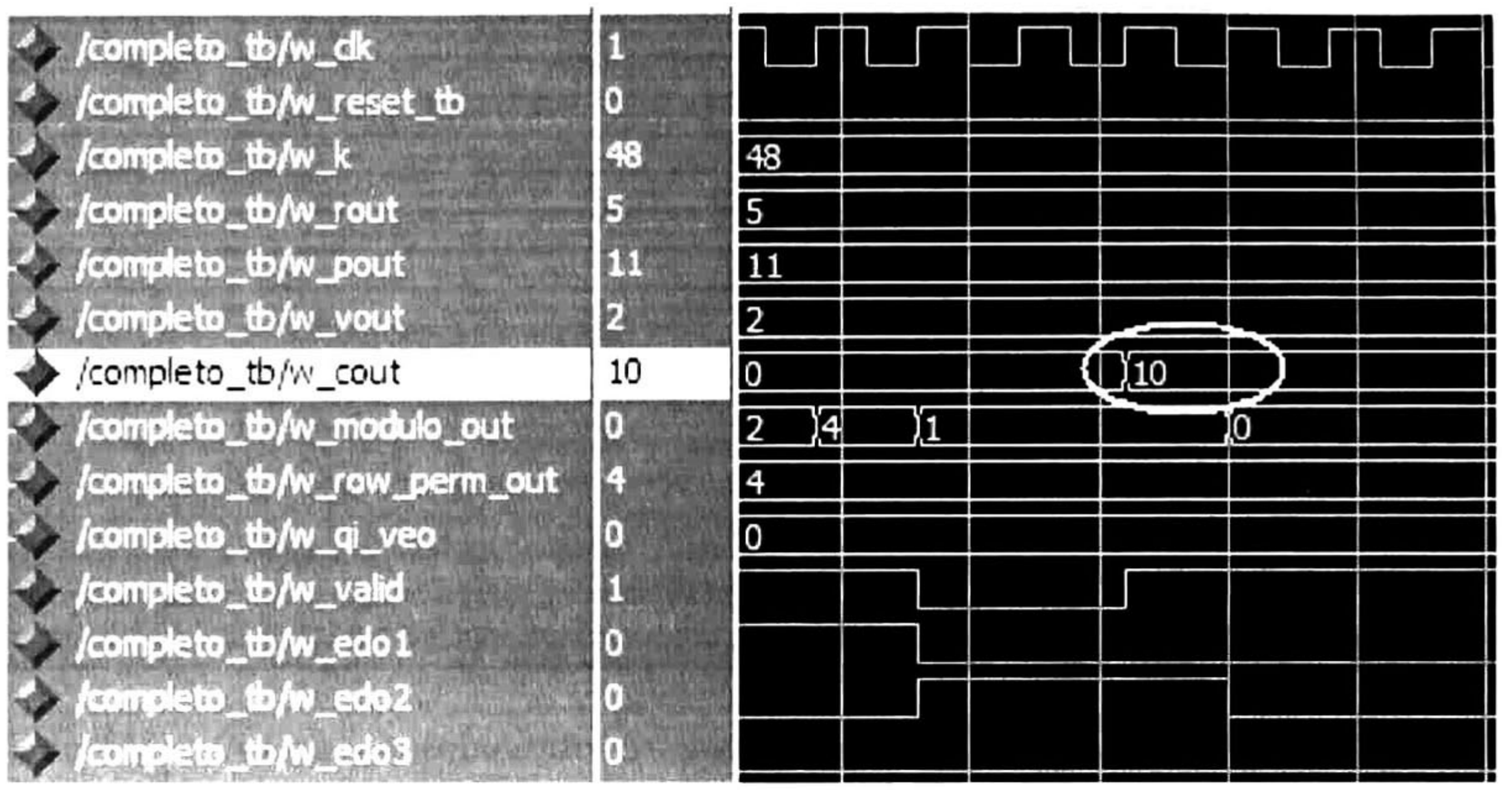

Figura 5.11. Simulación en ModelSim que muestra el valor obtenido de C=10 para un valor de K=48.

5.3.1.5 Direcciones de entrelazado "i_addr":

Luego que se tienen los parámetros necesarios para calcular las direcciones de entrelazado "i_addr" entramos al estado 3 (modo de cálculo), donde entran en acción los bloques "Modulo Computation", "S(j)RAM", "R,T Logic" y "Multi Operations" Como resultado de lo anterior obtenemos los parámetros necesarios para el cálculo de las direcciones de entrelazado, estos parámetros son "C", el vector de permutaciones entre renglones "T" y los desplazamientos para cada renglón provenientes de la "S(j)RAM" En la Figura 5.12 se muestran los valores de los parámetros mencionados y las direcciones entrelazadas obtenidas con estos parámetros a través del bloque "Multi Operation"

Según el resultado obtenido en MATLAB las direcciones de entrelazado para un tamaño de bloque K=48 para el estándar 3GPP-W CDMA son:

I_addr= [40, 30, 20, 10, 0, 41, 36, 21, 17, 6, 43, 34, 23, 18, 4,47, 31, 27, 15, 1,44, 32, 24, 13, 2, **49**, 39, 29, 19, 9, **48**, 33, 28, 12, 3, 46, 35, 26, 11, 5, 42, 38, 22, 14, 8, 45, 37, 25, 16, 7], donde las direcciones invalidas aparecen en negritas.

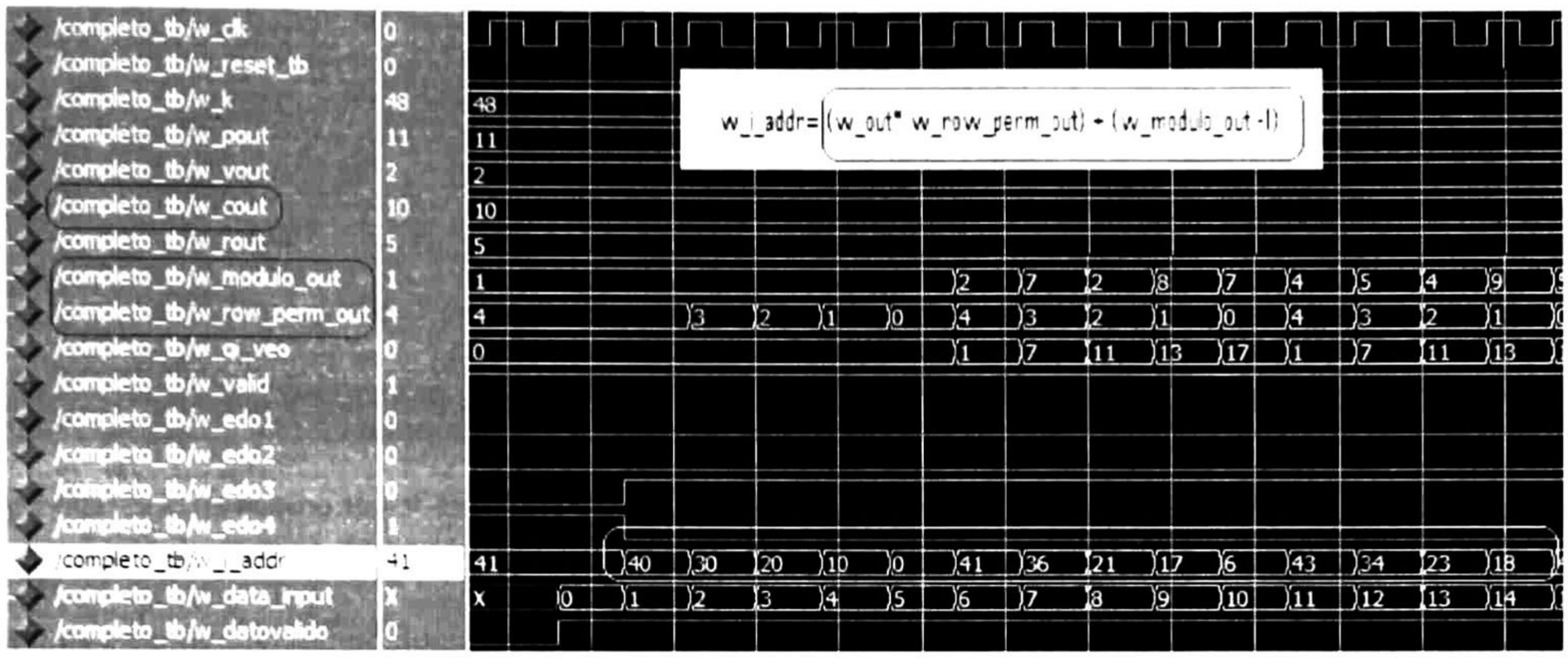

Figura 5.12. Simulación en ModelSim que muestra las direcciones de entrelazado generadas, asi como los parámetros necesarios para obtener dichas direcciones para un valor de K=48.

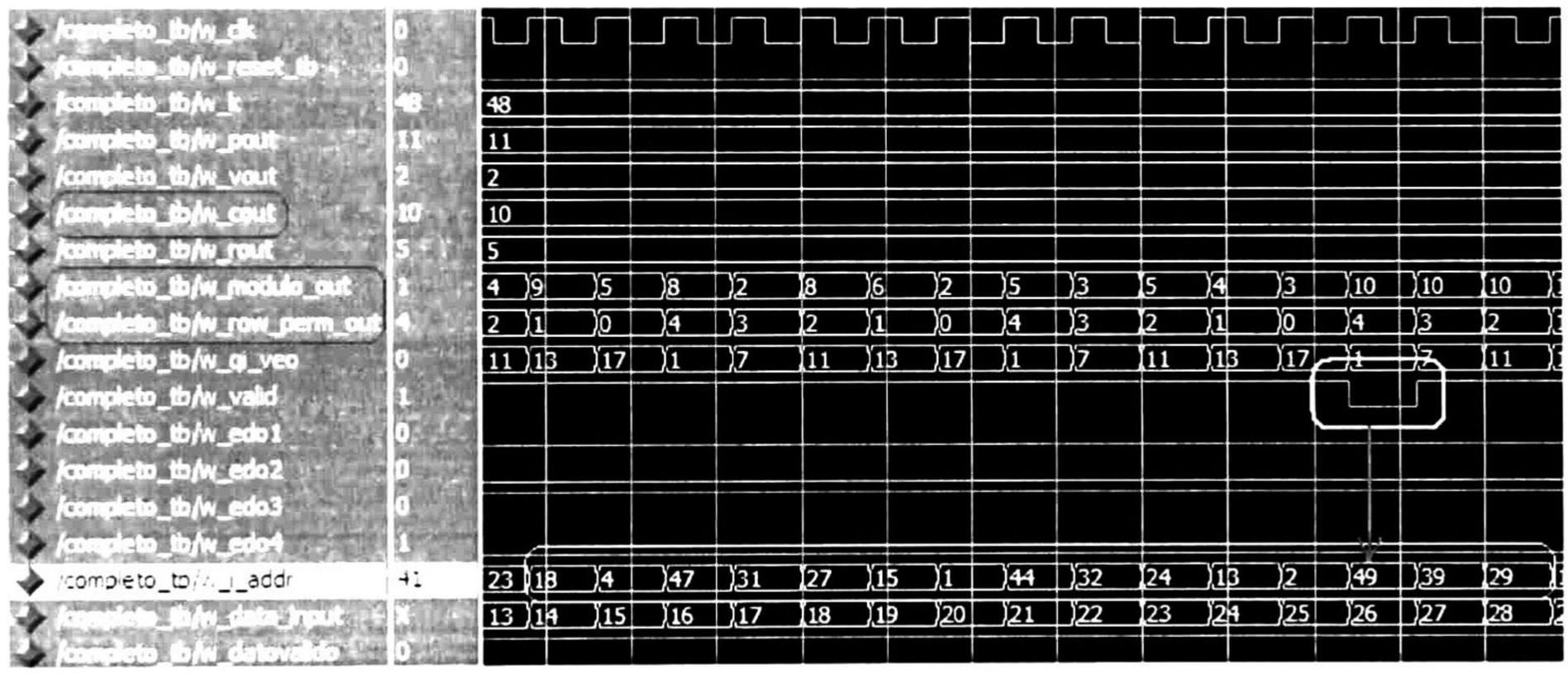

Figura 5.13. Simulación en ModelSim que muestra las direcciones de entrelazado generadas en el estado 3, así como la bandera "valid" que indica si son o no validas, todo para K=48.

Observando las figuras 5.11, 5.12 y 5.13 observamos las direcciones de entrelazado obtenidas en la simulación en ModelSim de nuestro diseño para K=48, donde las direcciones validas están indicadas por la bandera "valid" en alto.

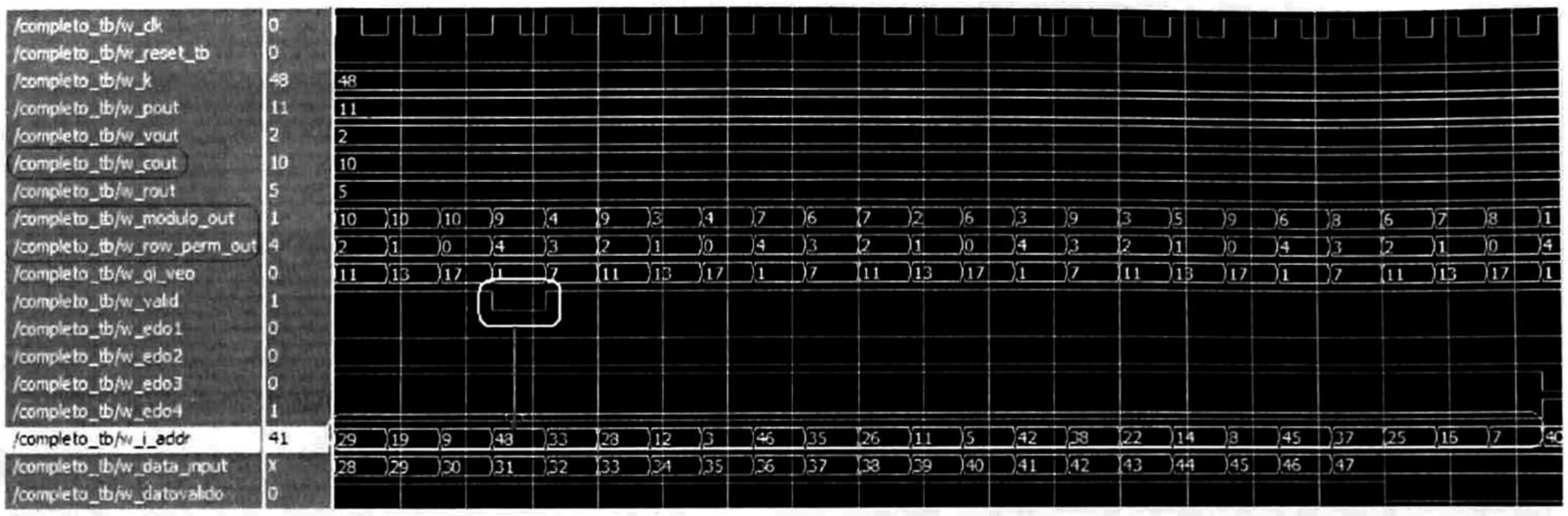

Figura 5.14. Simulación en ModelSim que muestra las direcciones de entrelazado generadas en el estado 3 y la bandera "valid" para K=48.

Al observar las formas de onda anteriores obtenidas de simulaciones en ModelSim y donde se muestran los parámetros generados por los diferentes bloques de nuestro diseño, pudimos comparar con los valores esperados según nuestro programa en MATLAB observando valores iguales, hasta llegar a la generación de las direcciones de entrelazado correctas para el caso de K=48 en el estándar 3GPP- W CDMA

5.3.2 3GPP-LTE

Si configuramos nuestro diseño bajo el estándar 3GPP-LTE y generamos las direcciones de entrelazado para un tamaño de bloque K=48, tendremos los resultados mostrados en las figuras 5.14, 5.15 y 5.16 después hacer las simulaciones correspondientes en ModelSim.

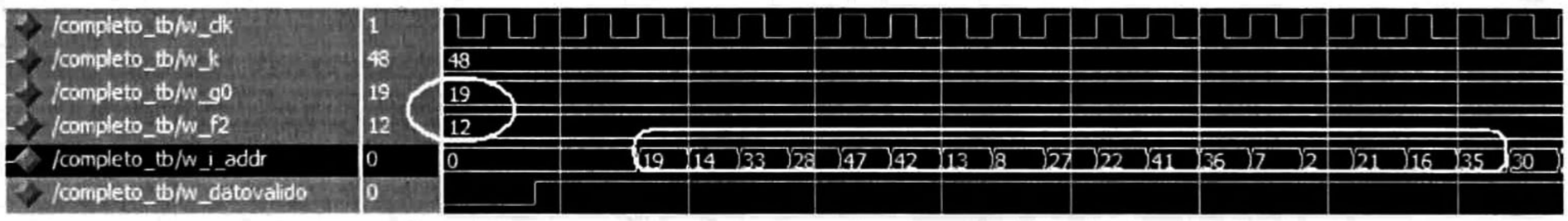

Figura 5.15. Simulación en ModelSim que muestra los parámetros "g0" y "f2" asi como las direcciones de entrelazado generadas por nuestro diseño para K=48 bajo el estándar 3GPP-LTE.

El cálculo de las direcciones de entrelazado para este estándar solamente depende de 2 parámetros, "g0" y "f2", ambos previamente calculados y guardados en una LUT dentro de nuestro diseño, para leer los valores adecuados de estos parámetros nos basamos solamente en el valor de K para direccionar la LUT. En la Figura 5.15 se muestran encerrados en un ovalo blanco los valores de "g0" y "f2" leídos de la LUT de nuestro diseño.

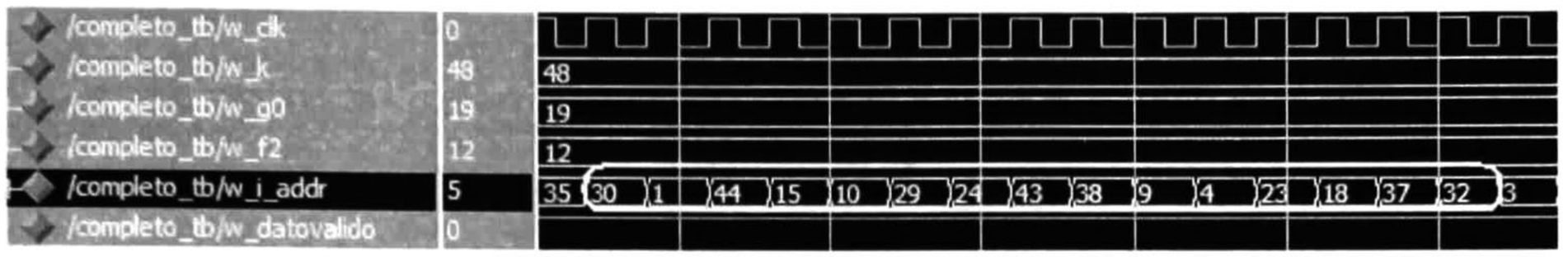

Figura 5.16. Simulación en ModelSim que muestra las direcciones de entrelazado generadas por nuestro diseño para K=48 bajo el estándar 3GPP-LTE.

En las figura 5.14, 5.15 y 5.16 se muestran las direcciones de entrelazado encerradas en rectángulos blancos, para este estándar todas las direcciones generadas son consideradas válidas y el momento en que comienzan a generarse depende de los datos de entrada.

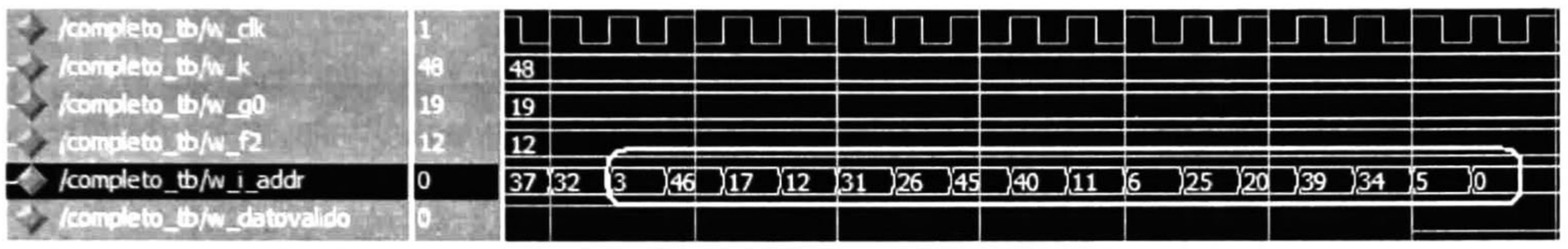

Figura 5.17. Simulación en ModelSim que muestra las direcciones de entrelazado generadas por nuestro diseño para K=48 bajo el estándar 3GPP-LTE.

Comparando los resultados obtenidos por simulación en ModelSim con los siguientes que son generados en MATLAB:

I_addr=[19, 14, 33, 28, 47, 42, 13, 8, 27, 22, 41, 36, 7, 2, 21, 16, 35, 30, 1, 44, 15, 10, 29, 24, 43, 38, 9, 4, 23, 18, 37, 32, 3, 46, 17, 12, 31, 26, 45, 40, 11, 6, 25, 20, 39, 34, 5, 0], nos damos cuenta que estos son los esperados.

5.4 Manejo de datos

Luego de tener la direcciones de entrelazado el bloque de manejo de datos se encarga de aprovechar estas direcciones para procesar los datos según se requiera. En el caso del estándar 3GPP-LTE la parte de manejo de datos funciona de forma muy sencilla, cada que se genera una dirección de entrelazado en ese momento dicha dirección se usa para leer o escribir en la memoria de datos. Para el estándar 3GPP-W CDMA los datos pasan por el bloque "In/Out RAM" antes de llegar a la memoria de datos, teniendo el efecto mostrado en la Figura 5.18, donde en las dos últimas filas se muestra, los datos que serán escritos en la memoria de datos provenientes del bloque "In/Out RAM" (en la penúltima fila) y las direcciones en que estos datos serán escritos en la memoria de datos (última fila). De estas mismas filas podemos notar el efecto que el bloque "In/Out RAM " tiene en los datos de entrada. Cuando se genera una dirección inválida (como 49) el dato correspondiente (25 en este caso) se queda en espera hasta que se genere la siguiente dirección valida.

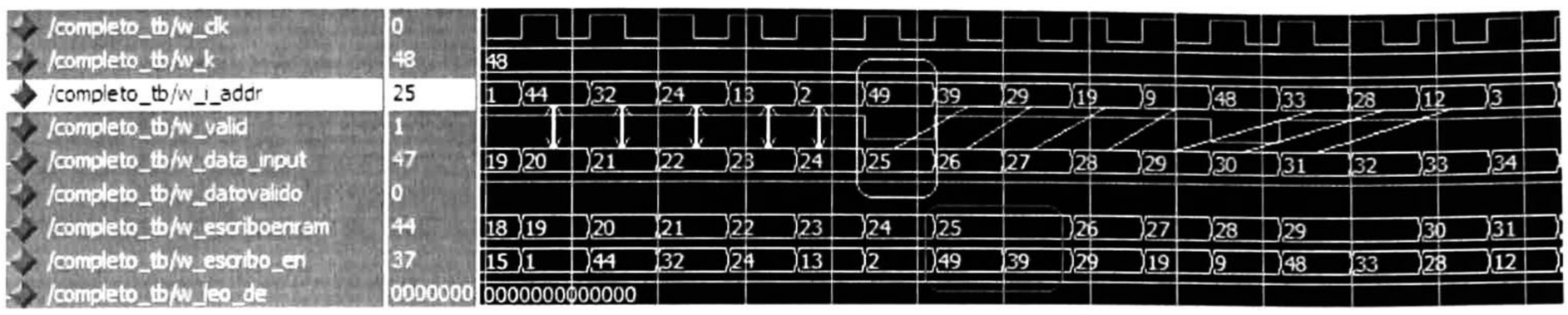

Figura 5.18. Simulación en ModelSim que muestra el efecto del bloque "In/Out RAM" en los datos de entrada cuando se generan direcciones inválidas para un entrelazado con K=48 en el estándar 3GPP-W CDMA.

5.5 Resultados

Después de hacer varias pruebas a nuestro diseño, en todas obtuvimos los resultados esperados según lo obtenido en MATLAB para ambos estándares (3GPP-W CDMA y 3GPP-LTE) y ambas funcionalidades (Interleaver/Deinterleaver).

Luego de sintetizar nuestro diseño completo para ser implementado en FPGA (Cyclone 2 de Altera) en la herramienta de Quartus II, versión 9.1 obtuvimos los siguientes resultados, en la Figura 5.19 se muestra el reporte de compilación de nuestro diseño generado por el Quartus II, el cual nos muestra el uso de los recursos disponibles del Cyclone 2 de nuestro diseño. Cabe mencionar que esta versión del diseño incluye algunos puertos de monitoreo, así como la utilización de registros como memorias internas del FPGA.

Quartus II Version	9.1 Build 304 01/25/2010 SP 1 SJ Web Edition
Revision Name	completo
Top-level Entity Name	completo
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	4,526 / 33,216 (14 %)
Total combinational functions	3,490 / 33,216 (11 %)
Dedicated logic registers	2,529 / 33,216 (8 %)
Total registers	2529
Total pins	173 / 475 (36 %)
Total virtual pins	0
Total memory bits	53,710 / 483,840 (11 %)
Embedded Multiplier 9-bit elements	3 / 70 (4 %)
Total PLLs	0 / 4 (0 %)

Figura 5.19. Reporte de compilación generado por Quartus II para nuestro diseño.

Según lo reportado por Quartus II, la frecuencia máxima de nuestro diseño en el peor de los casos para esta versión es de 28Mhz en este FPGA y que corresponde a un throughput de alrededor de los 280 Mbps, lo cual es mayor a lo requerido por el estándar 3GPP-W CDMA y suficiente para el 3GPP-LTE.

En la Figura 5.20 se muestra un diagrama del RTL generado por el mismo software (Quartus II para esta aproximación de nuestro diseño).

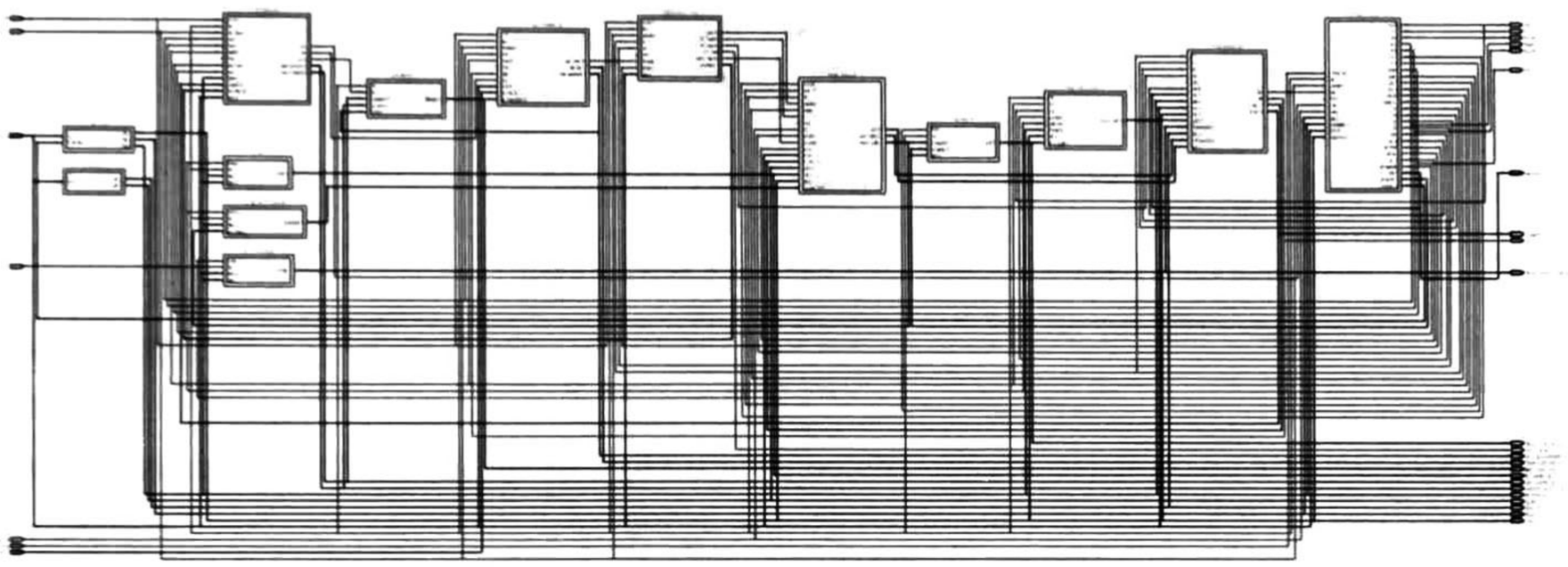

Figura 5.20. Diagrama RTL de nuestro diseño generado por Quartus II.

En la Figura 5.21 se muestra una tabla generada por Quartus II donde se muestra la distribución del uso de recursos de nuestro diseño para el FPGA.

Fitter Resource Usage Summary

	Resource	Usage
1	⊟ Total logic elements	4,526 / 33,216 (14 %)
2	-- Combinational with no register	1997
3	-- Register only	1036
4	-- Combinational with a register	1493
5		
27	Total block memory bits	53,710 / 483,840 (11 %)
28	Total block memory implementation bits	64,512 / 483,840 (13 %)

Figura 5.21. Distribución de uso de recursos generados por Quartus II.

5.5.1 Diseño considerando memorias externas

Con la finalidad de reducir el tamaño de nuestro diseño y prepararlo para una implementación en ASIC, se le hicieron algunos cambios. Al extraer las memorias de mayor tamaño de nuestro diseño original, se obtuvieron los siguientes resultados. En la Figura 5.22 se muestra el reporte de compilación después de hacer los cambios mencionados, donde podemos ver las diferencias notables con lo mostrado en la Figura 5.19.

Quartus II Version	9.1 Build 304 01/25/2010 SP 1 SJ Web Edition
Revision Name	completo
Top-level Entity Name	completo
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	1,828 / 33,216 (6 %)
Total combinational functions	1,766 / 33,216 (5 %)
Dedicated logic registers	625 / 33,216 (2 %)
Total registers	625
Total pins	148 / 475 (31 %)
Total virtual pins	0
Total memory bits	2,560 / 483,840 (< 1 %)
Embedded Multiplier 9-bit elements	3 / 70 (4 %)
Total PLLs	0 / 4 (0 %)

Figura 5.22. Reporte de compilación generado por Quartus II para nuestro diseño luego de extraer las memorias del diseño original.

Las memorias que sacamos del diseño original fueron la memoria de datos (dataRAM) cuyo tamaño es de 61Kbits, la memoria "SJ RAM" de 2Kbits y la memoria "fg ROM" de 2.5Hbits.

En la Figura 5.23 se muestra el diagrama del RTL de nuestro diseño después de las modificaciones para prepararlo para generar un ASIC.

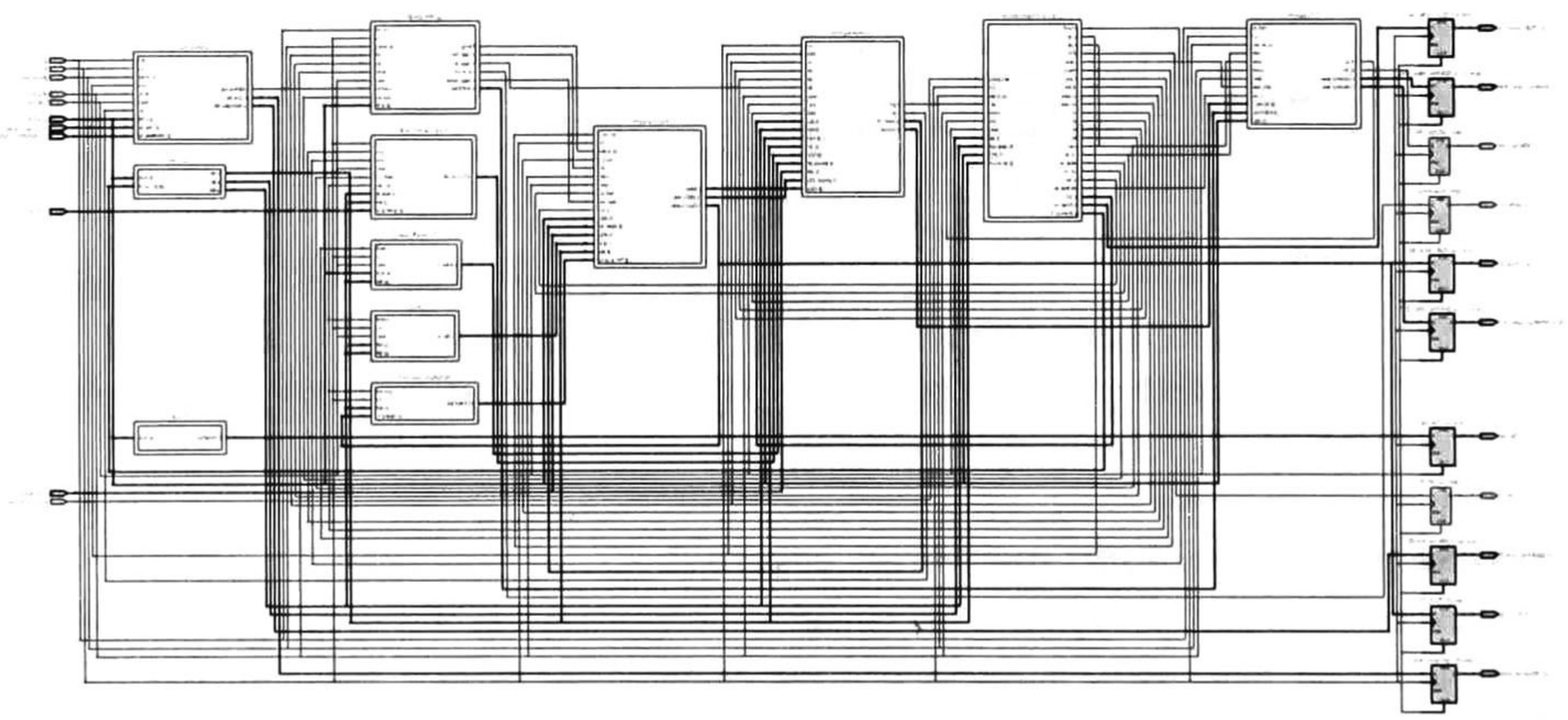

Figura 5.23. Diagrama RTL de nuestro diseño generado por Quartus II después de sacar las memorias de mayor tamaño.

En la Figura 5.24 se muestra la distribución del uso de recursos del FPGA (Cyclone 2 de Altera) luego de sacar las memorias y quitar algunos puertos de prueba, comparando esta figura con la Figura 5.21 podemos notar gran reducción en el uso de recursos.

Fitter Resource Usage Summary		
	Resource	Usage
1	⊟ Total logic elements	1,828 / 33,216 (6 %)
2	-- Combinational with no register	1203
3	-- Register only	62
4	-- Combinational with a register	563
5		
27	Total block memory bits	2,560 / 483,840 (< 1 %)
28	Total block memory implementation bits	4,608 / 483,840 (< 1 %)

Figura 5.24. Distribución de uso de recursos generado por Quartus II luego de sacar las memorias de nuestro diseño.

5.5.2 Latencia

La latencia en nuestro diseño en funcionamiento normal es variable y depende principalmente de dos factores, el tamaño del bloque "K" y del estándar en el que se esté operando.

Cuando nuestro Interleaver/Deinterleaver está configurado para trabajar en el estándar 3GPP-LTE la latencia será la misma que el tamaño del bloque "K". En el caso en que se esté trabajando bajo el estándar 3GPP-W CDMA la latencia dependerá no solo del tamaño del bloque sino también de las excepciones, entonces la latencia será la suma de el tamaño del bloque K y las direcciones inválidas generadas.

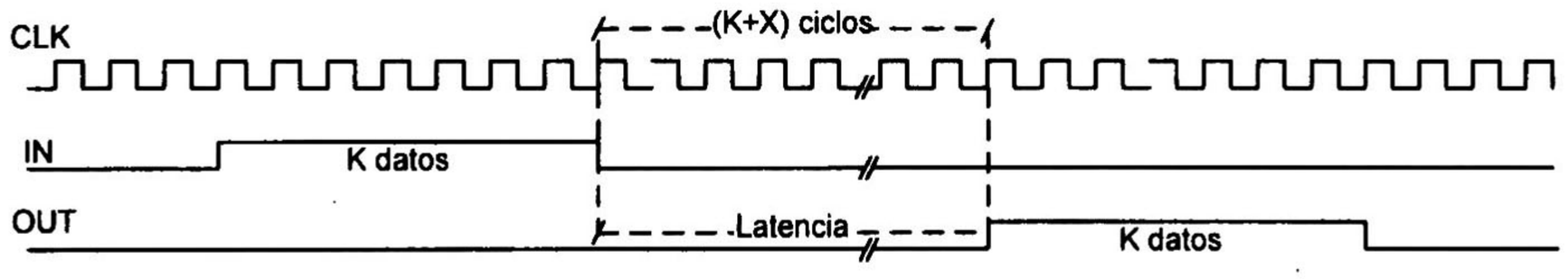

Figura 5.25. Forma de onda que ejemplifica la latencia de nuestro diseño.

En la Figura 5.25 se muestra la manera general de obtener la latencia de nuestro diseño, donde K es el tamaño del bloque y X representa el número de direcciones inválidas. Para el estándar 3GPP-LTE X=0 y para el estándar 3GPP-W CDMA $0 \leq X \leq 256$.

5.5.3 Potencia

En la actualidad la disminución en el consumo de potencia de los nuevos diseños se ha vuelto un tema de mucha importancia. En un sistema cada bloque del mismo tiene un

presupuesto de consumo de potencia al que se tiene que apegar, esto con la finalidad de lograr diseños más compactos y portables (en el caso que utilicen baterías).

Para conocer la potencia que nuestro I/D consume hicimos algunas mediciones con dos diferentes herramientas, una de Synopsis y otra llamada "Chip Estimator" Luego de mapear nuestro diseño a una tecnológica de 90 nm, la herramienta de Synopsis nos arrojó un consumo de 22mW para nuestro I/D donde el 97% del consumo esta potencia es "potencia dinámica" y el resto "potencia estática", por otro lado, la herramienta de "Chip Estimator" nos dio una estimación de 34mW para el mismo diseño, donde 90.3% de esta potencia corresponde a la potencia dinámica y el resto a la potencia estática.

En un diseño electrónico se le conoce como potencia activa o dinámica a aquella consumida durante las transiciones de estados lógicos de las distintas señales, así cuando una señal pasa de "1" a "0" o viceversa, se genera una corriente a través de los transistores involucrados, entonces cuanto más señales cambien de estado mayor potencia se consumirá. Para lograr una disminución en el consumo de esta potencia se podría buscar una disminución en el cambio de transiciones de cada nodo.

La potencia estática se consume incluso cuando no existe ninguna transición en las señales del diseño, este tipo de potencia se debe a las corrientes de fuga de los transistores, por lo que la potencia estática nos puede servir como un indicador aproximado del número de transistores de algún diseño.

Es difícil hacer una comparación justa en cuanto a consumo de potencia de nuestro diseño contra otros existentes, dado que el consumo de potencia depende de varios factores como la frecuencia de operación y la tecnología utilizada, sin embargo la Tabla 5.3 muestra algunos diseños de interleavers y su consumo de potencia.

Tabla 5.3. Consumo de potencia para distintos diseños de Interleavers.

Implementación	Estándares soportados	Tecnología utilizada	Consumo de potencia
I/D	3GPP-W CDMA 3GPP-LTE	90 nm	Aprox 28 mW
L. Horvath, et al.[19]	DVB bit e Interleaver de simbolo	0.6 µm	Aprox 300 mW
Rizwan Asghar[58]	3GPP-LTE 802.11n	65 nm	Aprox 11.7 mW
LUT[6]	802.11ª	------	Aprox 324 mW

5.5.4 Tamaño

Otro parámetro importante que se debe tomar en cuenta en el diseño digital, si es que se pretende mapear a un ASIC, es el tamaño en silicio que se requiere, el cual puede variar para un mismo diseño dependiendo de la tecnología a la que se mapea cada diseño. En nuestro caso y solo para fines informativos el mapeo fue hecho a una tecnología de 90 nm, para eso utilizamos la herramienta "Chip Estimator", con la que obtuvo que nuestro I/D utilizaría un área de 0.821mm^2.

Revisando algunos otros diseños de distintos Interleaver se obtuvieron los datos mostrados en la Tabla 5.4 de donde podemos notar que el área utilizada por nuestro Interleaver al ser mapeado a la tecnología de 90nm podría considerarse como un área aceptable.

Tabla 5.4. Uso de área de distintos diseños de interleavers.

Implementación	Estándares soportados	Tecnología utilizada	Área utilizada
I/D	3GPP-W CDMA 3GPP-LTE	90 nm	0.821 mm^2
Shin y Park [18]	WCDMA cdma2000	0.25 µm	2.678mm^2
Horvath et al [19]	DVB bit e Interleaver de símbolo	0.6 µm	69mm^2
Wu et al [21]	WiMAX,WLAN,802.11n	0.18µm	0.72mm^2
Chang [22]	DVB bit e Interleaver de símbolo	0.35µm	2.9mm^2

La mayoría de estos diseños utilizan memorias de distintos tamaños, pero estas no son incluidas en el tamaño total del diseño, ya que las consideran como módulos externos que se conectarán al Interleaver, en nuestro caso también consideramos algunas memorias como externas y de esta forma reducimos el tamaño en gran medida.

5.5.5 Frecuencia

Otro parámetro importante de nuestro I/D y de los interleavers en general es la frecuencia a la que pueden operar, nuestro diseño en el FPGA Cyclone 2 de Altera es capaz de operar a una frecuencia de 28MHz que es suficiente para el uso al que está destinado, sin embargo este parámetro depende y podría variar de acuerdo al dispositivo en el que será implementado, es decir que la frecuencia máxima de operación cambiara de un FPGA a otro, o bien si es mapeada a un ASIC esta frecuencia aumentará notablemente.

Si los requerimientos de frecuencia fueran aun mayores, se podrían incluir registros intermedios para romper las trayectorias que implican mayores retardos, aumentando con esto la frecuencia de operación, aunque quizá también la latencia del I/D.

Capítulo 5. Pruebas y Resultados

Capítulo 6

Conclusiones y trabajo futuro

6.1 Conclusiones

En el presente trabajo de tesis se abordó el tema de la importancia de contar con arquitecturas reconfigurables y del rehúso de hardware en el campo del diseño electrónico, todo esto para lograr cumplir con la exigencias de las tendencias tecnológicas actuales que incluyen la necesidad de módulos multi estándar y al mismo tiempo utilizar la menor cantidad de recursos posible.

Durante nuestro trabajo de tesis el módulo que desarrollamos desde el diseño hasta la implementación en FPGA, fue un Interleaver/Deinterleaver, el cual juega un papel primordial dentro de los Turbo Códigos para la corrección de errores. Nuestro diseño es reconfigurable y puede trabajar con cualquier tamaño de bloque de los estándares 3GPP-W CDMA y 3GPP-LTE.

Aunque en el mercado existen Interleavers y Deinterleavers casi para cualquier estándar, estos suelen ser propiedad intelectual (IP) del fabricante, y por tanto tener un costo muy elevado. El módulo que diseñamos en este trabajo de tesis podrá ser utilizado como parte de un sistema de comunicaciones mayor si así se requiere.

El diseño de nuestro Interleaver/Deinterleaver aprovecha investigaciones previas hechas en las áreas de algoritmos y computación, por lo que algunas de las operaciones necesarias para el cálculo de parámetros en la generación de direcciones de entrelazado en ambos estándares son hechas de forma iterativa, reduciendo de esta manera la complejidad del hardware requerido.

Otra característica que diferencia a nuestro Interleaver/Deinterleaver de otros diseños similares existentes, es que nuestro diseño no solo genera las direcciones de entrelazado (como la mayoría lo hacen) sino que, pese a las excepciones que el estándar 3GPP-W CDMA tiene, procesa los datos entrantes de acuerdo a dichas direcciones.

Luego de hacer nuestro diseño e implementación del mismo, donde algunos bloques se fueron mejorando según su comportamiento, se hicieron las pruebas que consideramos necesarias para validar nuestro Interleaver/Deinterleaver obteniendo los siguientes resultados, observaciones y/o conclusiones:

- Obtuvimos un Interleaver/Deinterleaver reconfigurable para los estándares 3GPP-W CDMA y 3GPP-LTE, que en la actualidad son 2 de los estándares más importantes en la

telefonía móvil. Con los resultados obtenidos en el presente trabajo de tesis, se publicó el artículo "Design and Implementation of a Configurable Interleaver/Deinterleaver for the 3GPP-W CDMA Standard" [23] donde se presenta la arquitectura de nuestro diseño.

- El rehúso de hardware mediante el multiplexado del mismo para usarse en distintas etapas o estándares, redujo los requerimientos de hardware y de consumo potencia.
- El bloque encargado de calcular la operación módulo mediante operaciones recursivas que reducen la complejidad de implementación en hardware fue uno de los principales aciertos en nuestro diseño.
- El uso de LUTs es conveniente cuando los parámetros que contienen no son demasiados y/o la complejidad en el cálculo de los mismos es elevada.
- Luego de hacer un análisis y síntesis de nuestro diseño para ser implementado en el FPGA "Cyclone 2" de Altera, este utiliza un 14% del total de los recursos del mismo.
- Nuestro diseño no solo genera las direcciones de entrelazado sino que entrelaza los paquetes de datos sin ser afectado por las excepciones que existan, para lograr esto utiliza una memoria de datos y una memoria que funciona como buffer de entrada y salida.
- Se hizo una comparación de nuestro diseño contra otros similares existentes, de donde se obtuvo que nuestro diseño es competitivo en uso de área y potencia además de manejar el flujo de datos.
- Se le hicieron algunas modificaciones a nuestro diseño original para obtener un nuevo diseño que sería la base para implementar nuestro Interleaver/Deinterleaver en ASIC. En este nuevo diseño las memorias de mayor tamaño fueron consideradas externamente.
- Luego de preparar este nuevo diseño para la síntesis e implementación en el FPGA "Cyclone 2", este utiliza solo el 6% de los recursos del mismo.
- Actualmente se está trabajando en un artículo donde se reportan los resultados obtenidos de desempeño y consumo de recursos de nuestro trabajo con estas últimas modificaciones.

6.2 Trabajo futuro

Nuestro trabajo de tesis es un bloque ya funcional, sin embargo es posible tomar este trabajo y hacerle varias mejoras como parte de un trabajo futuro.

- Una paralelización en los procesos que así lo permitan ayudarían a tener una arquitectura más compacta y eficiente.
- Se podría buscar hacer una optimización tanto en área como en consumo de potencia de los módulos que ya existen.
- Se podría investigar sobre nuevos algoritmos que se tradujeran en arquitecturas más eficientes en velocidad y/o consumo de hardware.
- Dado que el diseño de nuestro Interleaver/Deinterleaver ya incluye a uno de los estándares (3GPP-W CDMA) cuya trayectoria de entrelazado es de las más difíciles de

implementar, se podría aprovechar la arquitectura existente para incluir nuevos estándares.

- Un análisis de tiempos para identificar las trayectorias de retardos mayores ayudaría para intentar romper estas trayectorias introduciendo registros intermedios y de esta forma aumentando la velocidad de nuestra arquitectura.
- Finalmente, luego de hacer alguna o todas las mejoras mencionadas, se podría hacer una implementación del diseño resultante en ASIC.
- Las ideas utilizadas en el presente trabajo se podría utilizar como base para el desarrollo de una herramienta capaz de generar bloques genéricos utilizables en Interleavers de distintos estándares.

Capítulo 6. Conclusiones y trabajo futuro

Referencias

[1] Branka Vucetic, Jinhong Yuan. "Turbo Codes, Principles and Applications". Kluwer Academic Publishers, 4a. ed. 2004, pp. 314.

[2] Anabel Morales, Ramón Parra y Luis F. Gonzalez. "Implementación de un Turbo decodificador para un sistema de Comunicaciones", CINVESTAV 2009.

[3] Claude E. Shannon, "A Mathematical Theory of Communications", The Bell System Technical Journal, Vol. 27, pp. 379-423, 623-656, July, Octuber. 1948.

[4] Claude Berrou, Alain Glavieux and Punya Thitimajshima. "Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes", in Proc. ICC'93, pp. 1064-1070.

[5] R. Asghar and D. Liu, "Multimode Flex-Interleaver Core for Baseband Processor Plataform", in Hindway Journal of Computer Systems, Networks and Communications, oct. 2009.

[6] C. Sanchez-Ortiz, R. Parra-Michel and M. Guzman-Renteria, "Design and Implementation of a Multi-Standard Interleaver for 802.11a, 802.11n, 802.16e & dvb standards", CINVESTAV 2008.

[7] E.J.C. Rijshouwer and C.H. van Berkel. "A Programmable, Scalable-Througput Interleaver", ST Ericsson, DSP Inovation Center.

[8] E. Tell and D. Liu, "A Hardware Architecture for Multimode Block Interleaver", International Conference on Circuits and Systems for Communications (ICCSC), Moscow, Rusia, June 2004.

[9] G. D. Fortney, Jr.,"Burst-correcting codes for the classic bursty channel", IEEE Trans. Commun., vol. 19, no. 5, Oct. 1971, pp. 772-781.

[10] IEEE 802.11a Standard LAN/MAN Wireless LANS. http://grouper.ieee..org/groups/802/11/

[11]3GPP, "Technical specification group radio access networks; multiplexing and channel coding (FDD)," Technical Specification 25.212 V8.4.0, December 2008.

[12]3GPP-LTE, "Technical specification group radio access networks; E-UTRA; multiplexing and channel coding, release 8," Technical Specification 3GPP TS 36.212 v8.0.0,2007-2009.

[13]G. R. Blakley, "A Computer Algorithm for Calculating the Product A*B mod M," IEEE Trans. On Computers, vol. C-32, No. 5, pp. 497-500, May 1983.

[14]Z. Wang and Q. Li, "Very low-complexity hardware interleaver for turbo decoding," IEEE Trans. On Circuits and Sys. – II: vol. 54, no. 7, pp. 636-640, July 2007.

[15]P. Ampadu and K. Kornegay, "An Efficient Hardware Interleaver for 3G Turbo Decoding",in *Proceedings of IEEE Radio and Wireless Conference* (RAWCON 03), pp. 199-200, August 2003.

[16]R. Asghar and D. Liu, "Very Low Cost Configurable Hardware Interleaver for 3G Turbo Decoding", in *Proceedings of the 3rd International Conference on Information and Communication Technologies: From Theory to Applications* (ICTTA 08), pp. 1-5, Damascus, Syria , April 2008.

[17] O. Y. Takeshita and D. J. Costello, "New Deterministic Interleaver Design for Turbo Codes", in *Proceedings of 35th annual Allerton Conference on Communications Control and Computing.* September 2007.

[18]M. C. Shin and I. C. Park, "Processor-based turbo interleaver for multiple third-generation wireless standards," IEEE Communications Letters, vol. 7, no. 5, pp. 210-212, 2003.

[19]L. Horvath, et al, "A novel, high-speed, reconfigurable demapper-symbol Deinterleaver architecture for DVB-T," in *Proccedings of IEEE International Symposium on Circuits and Systems (ISCAS 99),* vol. 4, pp. 382-385, Orlando, Fla, USA, May-june 1999.

[20]Rizwan Asghar and Dake Liu, "Multimode Flex-Interleaver Core for baseband Processor Plataform ", in Journal of Computer Systems, Networks, and Communications. Vol 2010. October 2009.

[21]Y.-W. Wu, P. Ting and H.-P. Ma, "A high speed interleaver for emerging wireless communications," in *Proceedings of the International Conference on Wireless Networks, Communications and Mobile Computing,* vol. 2, pp. 1192-1197, Maui, Hawaii, USA, June 2005.

[22]Y.-N. Chang, "Design of an efficient memory based DVB-T channel decoder," in *Proceedings of the IEE International Symposium on Circuits and Systems (ISCAS 05),* vol. 5, pp. 5019-5022, Kaohsiung, Taiwan, May 2005.

[23] H. Borrayo-Sandoval, R. Parra-Michel, L. Fernando-Gonzalez, Fernando Printzen, "Design and Implementation of a Configurable Interleaver/Deinterleaver for the 3GPP-WCDMA Standard", in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs,* ReConFig 2009, Cancun, Mex, Dec. 2009.

Apéndice A. Funciones prototipo utilizadas en MATLAB.

En esta sección se presenta un listado con las funciones prototipo que utilizamos en MATLAB y que nos ayudan a relacionar de forma más sencilla nuestra arquitectura con los algoritmos correspondientes según los estándares. En las tablas A.1, A.2 y A.3 se hace una descripción de las funciones utilizadas en los tres principales niveles jerárquicos del proceso de entrelazado de datos de acuerdo a los estándares 3GPP-W CDMA y 3GPP-LTE.

Tabla A.1. Función encargada del entrelazado de datos de acuerdo a los estándares 3GPP-W CDMA y 3GPP-LTE.

Función	Dependencia de archivos	Descripción
interleaver(K,I/D,mode)	Data_in.txt gen_iaddr.m	Toma los K primeros datos (cada uno de 10 bits y en renglones diferentes) del archivo de texto "Data_in" ubicado en la carpeta "work" de MATLAB y los reordena de acuerdo a la trayectoria de entrelazado correspondiente. I/D=1 indica entrelazado de datos y el de-entrelazado está indicado por I/D=0. Con mode=1 se elije el estándar 3GPP-W CDMA y con mode=0 el estándar 3GPP-LTE.

Tabla A.2. Funciones que generan las direcciones de entrelazado

Función	Dependencia de archivos	Descripción
gen_iaddr(K,mode)	RT_logic.m pv_LUT.m findC.m findS.m findr.m intra_row.m inter_row.m read_addr.m addrLTE.m	Con esta función se generan las direcciones de entrelazado para cada tamaño de bloque K, de acuerdo al estándar que se le indique. Cuando "mode=1" las direcciones generadas son las indicadas por el estándar 3GPP-W CDMA y cuando "mode=0" las direcciones generadas son las indicadas por el estándar 3GPP-LTE. Las direcciones obtenidas con esta función son todas válidas.
addrLTE(K)		Genera las direcciones de entrelazado de acuerdo al estándar 3GPP-LTE.

Tabla A.3. Funciones para el cálculo de los parámetros iníciales para el estándar 3GPP-W CDMA.

Función	Dependencia de archivos	Descripción
RT_logic(K)		Con esta función se encuentra el número de renglones "R" y el vector con el orden de permutación entre renglones "T"
pv_LUT(K,R)	RT_logic.m	Encuentra los valores adecuados de "p" y "V"
findC(K,R,p,casoespecial)	RT_logic.m pv_LUT.m	Encuentra el número de columnas "C"
findS(p,V)	pv_LUT.m	Calcula el vector "S" que sirve como secuencia base para las permutaciones Inter-row.
findr(p,R,T)	pv_LUT.m RT_logic.m	Encuentra la secuencia auxiliar "r" en base al vector "T". Esta secuencia ayuda para la permutación "Inter-row"

Apéndice B. Funciones prototipo utilizadas en VHDL.

En este apéndice se muestran los módulos hechos en VHDL que componen nuestro diseño, así como la dependencia entre estos bloques. En la tabla B.1 se muestran cada uno de los bloques de nuestro diseño, así como la dependencia con otros bloques y una breve descripción de la función de cada uno de ellos.

Tabla B.1. Lista con los bloques en VHDL utilizados en el I/D.

Bloque funcional	Dependencia de archivos	Descripción
Interleaver.vhd	Gen_addr.vhd Data_handling.vhd	Módulo principal que recibe un bloque de datos de entrada, los reordena de acuerdo al estándar seleccionado y al tamaño del bloque y entrega el mismo bloque de datos pero ya procesados (reordenados).
Gen_addr.vhd	Controller.vhd pv_r.vhd Row_perm.vhd qi.vhd fg.vhd multi_operation.vhd Exc_handling.vhd sRAM.vhd ModuloSJ.vhd Circular_buffer.vhd dataRAM.vhd	Módulo que genera las direcciones de entrelazado según el estándar y tamaño de bloque elegido.
Data_handling.vhd	dataRAM.vhd muxes.vhd in_out_RAM	Módulo usado para direccionar y almacenar los datos a procesar.
Controller.vhd	pv_r.vhd Multi_operation.vhd	Módulo que sincroniza el funcionamiento de todo el Interleaver mediante la generación de banderas y control de procesos involucrados.
pv_r.vhd	Controller.vhd	Módulo encargado de encontrar los valores de los parámetros "p" y "v". Este modulo también calcula el valor de "R".
Row_perm.vhd	pv_r.vhd Controller.vhd	Módulo con tabla incluida encargado de elegir el vector "T(i)" adecuado según el tamaño del bloque de datos. Este vector contiene las permutaciones entre renglones.

Apéndice C. Algoritmo Computacional para el cálculo del producto A*B módulo M.

Cuando se hace el cálculo de A*B Módulo cierto entero positivo $M<2^H$, es un desperdicio construir el producto ordinario de enteros A*B, como en los métodos tradicionales, y entonces obtener el número entero positivo (residuo) R<M tal que A*B=M*Q + R, donde el entero no negativo Q es, como de costumbre, el cociente de la división de A*B y M. Pero, el cociente Q no nos interesa y el manejo del producto de enteros de 2H-bits es un desperdicio de esfuerzo cuando nuestro propósito es encontrar un residuo, R, de H-bits modulo otro entero, M, de H-bits.

Lo que a continuación se presenta es un algoritmo que no gasta recursos en la división de enteros y hace uso solamente de las palabras de H-bits. Este algoritmo es mucho más rápido que hacer una multiplicación de enteros y posteriormente una división de enteros. En este artículo se considera únicamente la aritmética hecha en un bit a la vez, es decir, cada resultado parcial dependerá del valor de cada bit individual de los operandos.

Algoritmo de multiplicación modular (MMA, del inglés *Modular Multiplication Algorithm*)

El algoritmo MMA [13], cuyo diagrama de flujo se muestra en la figura C.1, toma como entradas cuatro enteros no negativos, A, B, M y H.

A y B son enteros menores que M, el cual a su vez es menor que 2^H. MMA entrega el residuo A*B dividido por M. En otras palabras la acción de MMA está dada por la siguiente ecuación.

$$(A, B, M, H) \rightarrow (A * B) \bmod M$$

En este algoritmo tenemos que $1 \le M < 2^H$, $0 \le A < M$, $0 \le B = \sum B(T)*(2^T) < M$, y donde la suma en B se aplica sobre todos los enteros no negativos T<H, es decir B está representado como la suma de cada uno de los bits que lo componen y el peso que cada uno de ellos tienen de acuerdo a su posición.

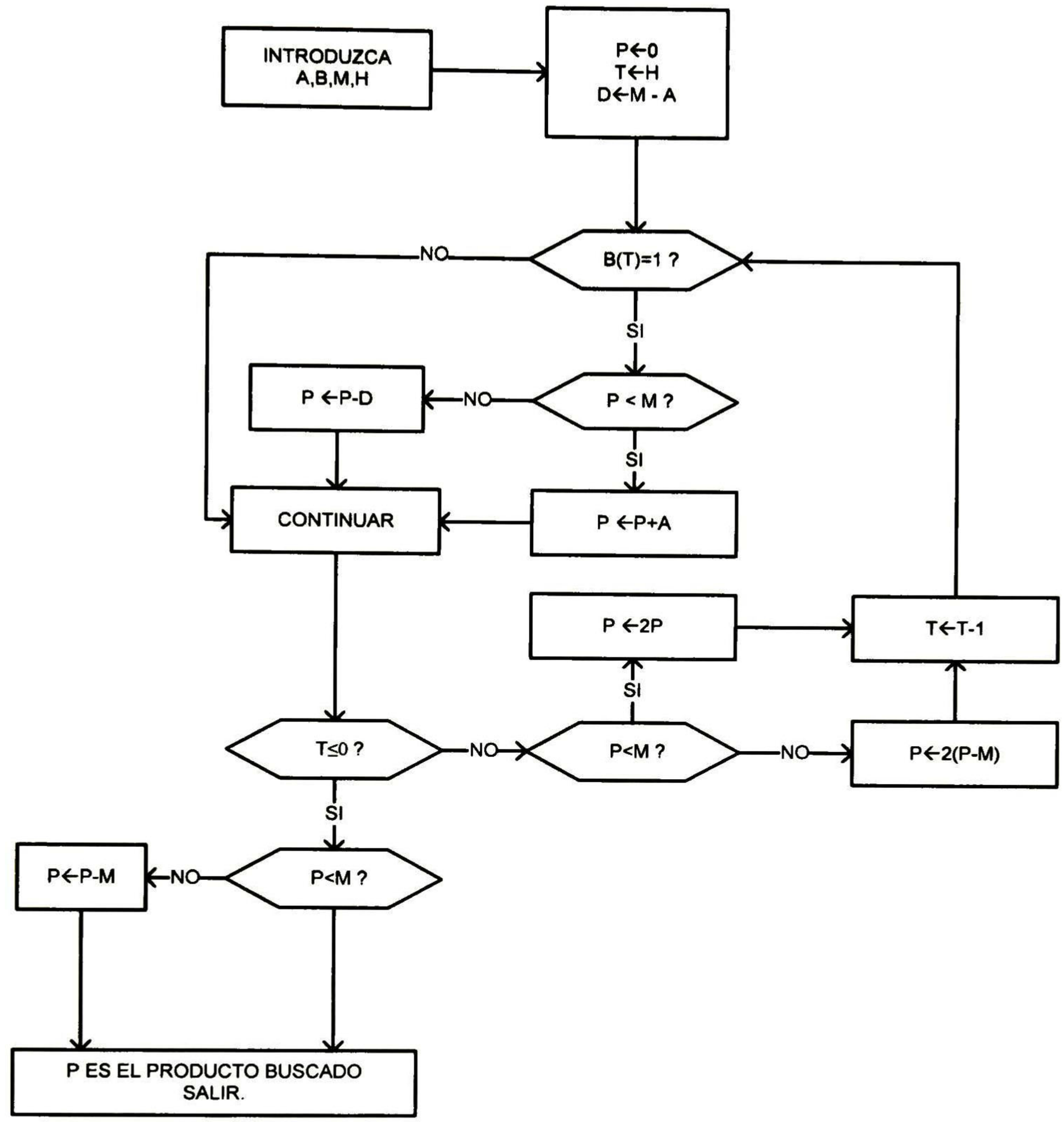

Figura C. 1. Diagrama de flujo del algoritmo de multiplicación modular.

Observando el diagrama de flujo tenemos que para obtener el resultado de la operación A*B Mod M, donde $M<2^H$ partimos de definir los valores de los enteros A, B, M y H, donde las operaciones iterativas que se realizaran dependerán de comparaciones del valor del resultado P y del valor de cada uno de los bits de B, partiendo por el bit más significativo y hasta llegar al bit menos significativo.

Para dejar más claro la forma en que el algoritmo MMA funciona realizaremos un sencillo ejemplo. Calcularemos *"(9x7) mod 11"* donde siguiendo el diagrama de flujo de la figura C.1 tenemos:

A=7, B=9=01001_b, M=11 y H=4

P=0, T=4 y D=4

B (4)=1?, No.

T≤0? , No

P<M?, Si. Entonces P=2P=0.

T=T-1=3

B(3)=1?, Si

P<M?, Si

P=P+A=7

T≤0?, No

P<M?, Si. Entonces P=2P=14.

T=T-1=2

B(2)=1?. No

T≤0? , No

P<M?, No. Entonces P=2(P-M)=6.

T=T-1=1

B(1)=1?, No

T≤0? , No

P<M?, Si. Entonces P=2P=12

T=T-1=0

B(0)=1?, Si

P<M?, No. Entonces P=P-D=8

T≤0?, Si

P<M?, Si. Entonces **P=8** es el valor buscado y concuerda con *"(9x7) mod 11"*.

El algoritmo MMA actúa como actuaria un humano y suma los múltiplos mayores de A, al resultado parcial P antes de sumar los múltiplos más pequeños de A. El numero P nunca será mayor que 2M en ninguna parte del proceso. Debido a lo anterior P solo requiere H+1 bits para ser representado.

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 de un Interleaver/Deinterleaver Reconfigurable para los estándares 3GPP-WCDMA y 3GPP-LTE

del (la) C.

Héctor BORRAYO SANDOVAL

el día 19 de Agosto de 2010.

Dr. Ramón Parra Michel
Investigador CINVESTAV 2C
CINVESTAV Unidad Guadalajara

Dr. José Raúl Loo Yau
Investigador CINVESTAV 2B
CINVESTAV

Dr. Alberto Alcocer Ochoa
Hardware Engineering
Intel de México

CINVESTAV - IPN
Biblioteca Central
SSIT0009797