CT-919-551
DON.2016

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

Diseño e implementación digital de un transmisor reconfigurable para sistemas OFDM

CINVESTAV
IPN
ADQUISICION
LIBROS

Tesis que presenta:

LEONARDO OROZCO GALVAN

para obtener el grado de:
Maestro en Ciencias

en la especialidad de:
Ingeniería Eléctrica

Director de Tesis
Dr. Ramón Parra Michel

CINVESTAV del IPN Unidad Guadalajara, Guadalajara, Jalisco, Agosto de 2015.

CLASIF.. CT00870
ADQUIS.. CT-919-551
FECHA: 25-04-2016
PROCED.. DON.-2016
$

Diseño e implementación digital de un transmisor reconfigurable para sistemas OFDM

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:

LEONARDO OROZCO GALVAN

Ingeniero en Comunicaciones y Electrónica

Instituto Politécnico Nacional 2008-2012

Becario de Conacyt, expediente no. 300943

Director de Tesis

Dr. Ramón Parra Michel

CINVESTAV del IPN Unidad Guadalajara, Agosto de 2015.

Agradecimientos

A Conacyt por el apoyo económico brindado. A mis padres, hermanos y familia en general. A los doctores Ramón Parra y Eduardo Romero, por su asesoría y apoyo en la elaboración de esta tesis. A los miembros de Tele13, Carlos Hernán, Francisco, Gerardo, Chaides, Pedro y Orlando. A los compañeros de tele, en especial a Abisaí Ramírez, Fernado Peña y J. L. Pizano. A los doctores que compartieron su conocimiento conmigo durante la maestría. Finalmente, a mis amigos Roberto Leyva, Jonathan R. Domínguez, *Moy* Morales y Sergio *Rodas*.

Resumen

Los sistemas de comunicación tales como 5G requerirán de alta interoperabilidad entre estándares, obligando a los diseños actuales a considerar la inclusión de múltiples estándares en un sólo dispositivo. Por lo tanto, los bloques tienen que ser diseñados con una alta capacidad in términos de reconfigurabilidad. En esta tesis se presenta la arquitectura e implementación de un transmisor OFDM reconfigurable con tales capacidades. La arquitectura propuesta es capaz de generar tramas para el estándar LTE (Long Term Evolution) y con un mínimo de modificaciones también puede generar tramas para el estándar IEEE 802.11ac. Los parámetros de configuración pueden ser ajustados "sobre la marcha" por medio de registros de configuración, tales como: tamaño de la trama, longitud de prefijo cíclico, esquema de modulación de datos y valor de los pilotos. Adicionalmente, es posible operar en cinco anchos de banda requeridos por LTE, permitiendo su inclusión como diseño multi estándar y plataformas SDR (software design radio). Los resultados muestran un moderado consumo de la FPGA y un desempeño de SQNR (relación señal a ruido de cuantización) promedio de 50 dB.

Abstract

Future communication systems such as 5G will require high interoperability between standards, obligating current designs to contemplate the inclusion of multiple standards into a single device. Therefore, the main communication blocks have to be designing with high capability in terms of reconfigurability. This thesis presents the architecture and implementation of a reconfigurable OFDM transmitter with such capabilities. The proposed architecture is capable of generating frames for the Long Term Evolution (LTE) standard and with minimal modifications it can also generate frames for IEEE 802.11ac standard. The configuration parameters can be adjusted on-the-fly via configuration registers, such as: frame size, Cyclic Prefix size, data modulation type and pilot values. Additionally, it is possible to operate in the five bandwidths required for LTE, allowing its inclusion in multi standard design and software design radio platforms. Results shown a moderate FPGA consumption and good SQNR (signal-to-quantization-noise ratio) performance of 50 dB in average.

Índice general

Índice de tablas

Índice de figuras

Lista de acrónimos

OFDM Multiplexaje por División de Frecuencias Ortogonales

RB Resource Block

RE Resource Element

LTE Long Term Evolution

IDFT Transformada Discreta de Fourier Inversa

DFT Transformada Discreta de Fourier

FFT Transformada Rápida de Fourier

IFFT Transformada Rápida de Fourier Inversa

IMT-Advanced International Mobile Telecommunications - Advanced

ITU International Telecommunication Union

UMB Ultra Mobile Broadband

OFDMA Acceso Múltiple por División de Frecuencias Ortogonales

QAM Quadrature Amplitude Modulation

DQPSK Differential Quadrature Phase Shift Keying

OQPSK Offset Quadrature Phase Shift Keying

BPSK Binary Phase Shift Keying

DBPSK Differential Binary Phase Shift Keying

IP Propiedad Intelectual

CP Prefijo Cíclico

AGU Unidad Generadora de Direcciones

DR Data Register

PR Pilots Register

NCR Null Carrier Register

CPR Cyclic Prefix Register

FR Frame Register

SltR Slot Register

OB Out Buffer

TRC Transmission Rate controller

BF Butterfly operation

LBFS Last Butterfly Stage

SEBF Single External Butterfly

DEBF Double External Butterfly

ShBF Shared Butterfly

OFDMSyC OFDM Symbol Conformer

OFDMSltC OFDM Slot Conformer

FCCU Frame Conformer Control Unit

FPGA Field Programmable Gate Array

SQNR Signal-to-Quantization-Noise Ratio

MSB Bit más significativo

LSB Bit menos significativo

Capítulo 1

Introducción

1.1. Introducción

Actualmente, los sistemas de comunicación inalámbrica forman parte de la vida cotidiana de un alto porcentaje de la población mundial. Aunque a simple vista parecería que su principal aplicación ha sido para fines recreativos, indudablemente han sido muy importantes para el desarrollo de los países. Esto se debe a que las comunicaciones están presentes en todo tipo de actividades, desde las tareas financieras, económicas y administrativas hasta la investigación científica.

En particular, los sistemas móviles han tenido un auge masivo en las últimas décadas. Esto se debe, en cierta medida, a que sus costos se han ido reduciendo con el surgimiento de tecnologías más eficientes. Recíprocamente, la gran demanda ha hecho que los sistemas móviles requieran de tasas de transmisión cada vez más altas. Es por ello que la tarea de desarrollar estos sistemas se ha vuelto compleja y sumamente importante.

Un acontecimiento crucial para el desarrollo de los sistemas de comunicación ha sido el gran avance de los sistemas de cómputo. La convergencia entre ambas ramas le ha brindado a las comunicaciones, características sofisticadas que permiten reducir los tiempos de procesamiento. El Multiplexaje por División de Frecuencias Ortogonales (OFDM) es un ejemplo de esto, pues para realizar el proceso de modulación, utiliza el cálculo eficiente de la Transformada Discreta de Fourier Inversa (IDFT).

Los sistemas de comunicación de cuarta generación, tales como 4G Long Term Evolution (LTE), utilizan banda ancha y un manejo eficiente del espectro. Es por ello que 4G LTE ha surgido como la opción actual para satisfacer la demanda de altas tasas de transmisión.

1.1.1. Sistemas de comunicación 4G

En Octubre de 2010, la International Telecommunication Union (ITU) anunció que dos sistemas cumplieron con los requerimientos IMT-Advanced (International Mobile Telecommunications Advanced)[8]. Uno de estos sistemas fue LTE-Advanced, mientras que el otro fue una versión mejorada de WiMAX, conocida como WiMAX 2.0.

Originalmente, Qualcomm tenía intenciones de desarrollar un sucesor 4G para cdma2000 bajo el nombre *Ultra Mobile Broadband* (UMB). Sin embargo, ningún operador de red anunció planes de adoptar esta tecnología y el proyecto fue cancelado en 2008. En su lugar, la mayoría de los operadores cdma2000 decidieron cambiar a LTE. Esto originó una situación donde había dos opciones de comunicación móvil 4G: LTE y WiMAX. De estos dos, LTE tiene, por mucho, mayor soporte entre los operadores de red y fabricantes de equipos, al grado de que varios operadores WiMAX han optado por migrar sus redes a LTE. Debido a su soporte, LTE probablemente será la tecnología de comunicación móvil dominante para los siguientes años [6].

1.1.2. Acceso Múltiple por División de Frecuencias Ortogonales

La técnica utilizada para radio transmisión y recepción en LTE es conocida como Acceso Múltiple por División de Frecuencias Ortogonales (OFDMA). OFDMA es una versión modificada de la técnica conocida como Multiplexaje por división de frecuencias Ortogonales (OFDM), que realiza la misma función que cualquier otra técnica de acceso múltiple, permitiendo a la estación base comunicarse con distintos móviles al mismo tiempo. Además, es una manera poderosa de mejorar la eficiencia espectral del sistema y minimizar los problemas de fading e interferencia intersimbólica [6].

1.1.2.1. Subportadoras en OFDM

Un transmisor OFDM toma un bloque de símbolos (a_0, a_1,..., a_{N_c}) de un flujo de información y transmite cada símbolo en una radio frecuencia diferente, la cual es conocida como subportadora. El ancho de banda de cada subportadora individual es pequeño, por lo tanto tiene una tasa de símbolo relativamente baja. No obstante, colectivamente, las subportadoras ocupan el mismo ancho de banda que un sistema de portadora única tradicional. En OFDM, el espacio entre subportadoras Δf está dado por [6]:

$$\Delta f = \frac{1}{T} \tag{1.1}$$

donde T es la duración del símbolo OFDM. Qué espacio entre subportadoras usar, depende del tipo de entorno en el que el sistema se encuentra operando, incluyendo aspectos como la máxima selectividad en frecuencia del canal (tiempo de dispersión máximo esperado) y la

máxima tasa esperada de variaciones en el canal (dispersión Doppler máxima esperada). Una vez que el espacio entre subportadoras ha sido seleccionado, el número de subportadoras puede ser decidido basándose en el ancho de banda de transmisión global asumido [7]. En LTE, la duración del símbolo es de $T = 66{,}67$us, por lo que el espacio entre subportadoras es de 15 KHz [6] [7][25].

1.1.2.2. Transmisión en OFDM

La transmisión por medio de OFDM puede ser vista como un tipo de transmisión multiportadora. En notación de banda base compleja, una señal OFDM $x(t)$ básica, durante el intervalo de tiempo $mT \leq t < (m+1)T$, puede ser expresada como:

$$x(t) = \sum_{k=0}^{N_c-1} x_k(t) = \sum_{k=0}^{N_c-1} a_k^{(m)} e^{j2\pi k\Delta f t} \tag{1.2}$$

donde $x_k(t)$ es la k-ésima subportadora modulada con la frecuencia $f_k = k\Delta f$ y $a_k^{(m)}$ es el, generalmente complejo, símbolo modulado aplicado a la k-ésima subportadora durante el m-ésimo símbolo OFDM, el cual ocurre durante el intervalo $mT \leq t < (m+1)T$ [7].

La transmisión OFDM está basada en bloques, lo cual implica que, durante cada intervalo de símbolo OFDM, N_c símbolos modulados son transmitidos en paralelo. Los símbolos modulados pueden ser de cualquier alfabeto de modulación, tal como 4QAM, 16QAM o 64 QAM (*Quadrature Amplitude Modulation*) [7].

El término *Multiplexaje por división de frecuencias Ortogonales* se debe al hecho de que dos subportadoras OFDM moduladas x_{k1} y x_{k2} son mutuamente ortogonales en el intervalo de tiempo $mT \leq t < (m+1)T$, esto es:

$$\int_{mT}^{(m+1)T} x_{k1}(t)x_{k2}^*(t)dt = 0 \qquad \text{para} x_{k1} \neq x_{k2} \tag{1.3}$$

Así, una transmisión OFDM puede ser vista como una transmisión de un conjunto de funciones ortogonales $\varphi_k(t)$, donde [7]:

$$\varphi_k(t) = \begin{cases} e^{j2\pi k\Delta f t} & 0 \leq t < T \\ 0 & \text{otro caso} \end{cases} \tag{1.4}$$

1.1.2.3. Transmisor OFDM

La figura 1.1 muestra los componentes más importantes de un transmisor OFDM particular de $N_c = 8$ símbolos. El transmisor acepta un flujo de bits desde protocolos de capas superiores y los convierte en símbolos utilizando el esquema de modulación elegido. El convertidor serie/paralelo toma un bloque de N_c símbolos, y los dirige a N_c sub-flujos

paralelos. El transmisor mezcla cada sub-flujo con una subportadora, cuya frecuencia es un múltiplo entero de $\Delta f = 15$kHz. Para tener consistencia con las especificaciones de LTE, es conveniente poner a las portadoras en las frecuencias de -60,-45,...,45 KHz. Ahora, se tienen N_c ondas senoidales, cuyas amplitudes y fases representan a los N_c símbolos transmitidos. Sumando estas senoides y dividiéndolas por el factor de escala N_c, se puede generar una sola forma de onda en el dominio del tiempo, la cual es una representación de frecuencia intermedia de la señal que se quiere transmitir. La tarea restante es mezclar la forma de onda con radio-frecuencia para su transmisión. Por lo tanto, se puede interpretar al transmisor OFDM como un banco de moduladores de frecuencia intermedia, cada uno de los cuales es ajustado al *offset* de frecuencia de la correspondiente portadora [6]. En la figura 1.2 se muestra el caso general de un transmisor OFDM de N_c símbolos bajo esta interpretación.

1.1.2.4. Receptor OFDM

La figura 1.3 muestra los componentes más importantes de un receptor OFDM particular de $N_c = 8$ símbolos. El receptor acepta la señal entrante, la mezcla con una replica compleja conjugada de la radio frecuencia original y filtra el resultado. Después, dirige la señal a N_c trayectorias separadas y mezcla cada una con una réplica compleja conjugada de una de las subportadoras originales. La información que llega a cada subportadora está ahora en una frecuencia cero. Integrando este resultado para la duración de un símbolo, el receptor puede extraer la amplitud y fase del símbolo que llega a cada subportadora, al mismo tiempo que rechaza cualquier ruido e interferencia.

Debido al canal, cada supbortadora es modificada por una fase arbitraria. Para lidiar con esto, el transmisor OFDMA inserta símbolos de referencia (pilotos) en el flujo de datos con amplitud y fase definidas por especificaciones relevantes. De este modo, los pilotos son utilizados para propósitos de medición y estimación de canal, así como de sincronización [25]. En el proceso de estimación de canal, el receptor mide los símbolos de referencia entrantes, los compara con aquellos que las especificaciones definen y utiliza el resultado para remover el cambio de fase de la señal entrante.

En la figura 1.3 se ha asumido que cada subportadora es desviada en fase por el mismo factor, 30°. Sin embargo, en presencia de desvanecimiento en frecuencia, el cambio de fase es una función que varía lentamente en frecuencia, al igual que en tiempo. Para lidiar con esto, los símbolos de referencia en LTE son dispersados en los dominios de tiempo y frecuencia. El receptor puede entonces medir el cambio de fase como una función de frecuencia y puede aplicar una corrección de fase a cada suportadora individual. Después de la etapa de eliminación de fase, el receptor devuelve los símbolos a su orden original por medio de un convertidor paralelo/serie y recobra los bits transmitidos.

Por lo tanto, se puede interpretar al receptor OFDM como un banco de demoduladores de frecuencia intermedia, cada uno de los cuales es ajustado al offset de frecuencia de la correspondiente portadora [6]. En la figura 1.4 se muestra el caso general de un receptor OFDM de N_c símbolos bajo esta interpretación.

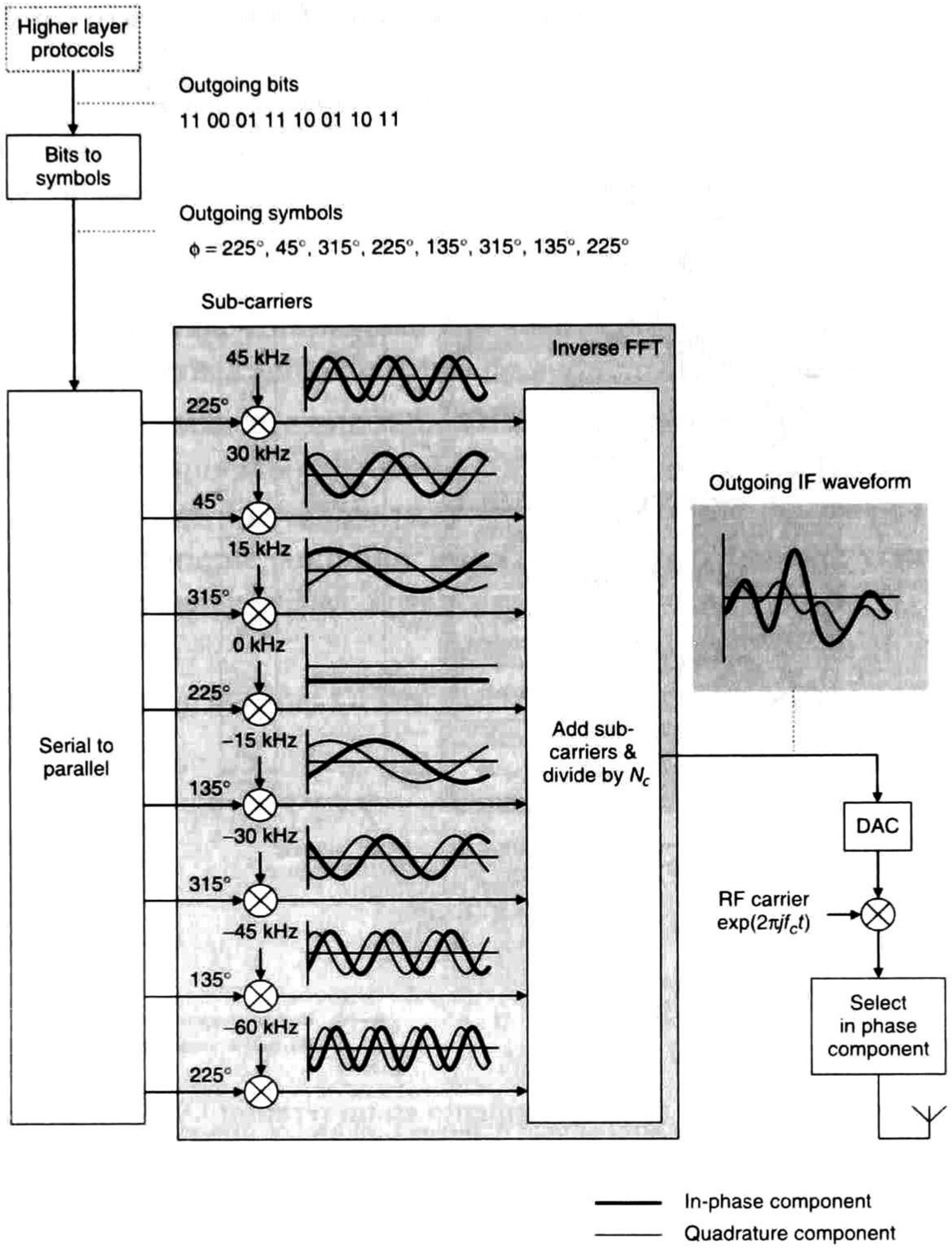


Figura 1.1: Fases de procesamiento en un transmisor OFDM [6]

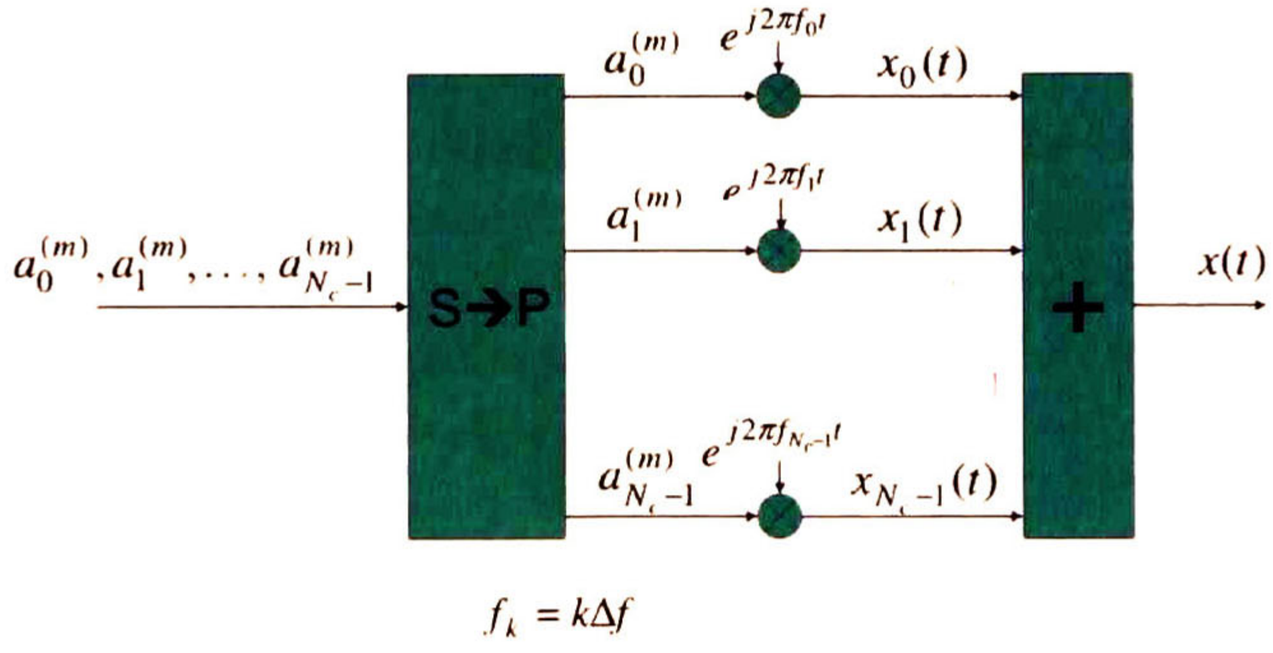


Figura 1.2: Transmisor OFDM interpretado como un banco de moduladores de frec. Intermedia [7]

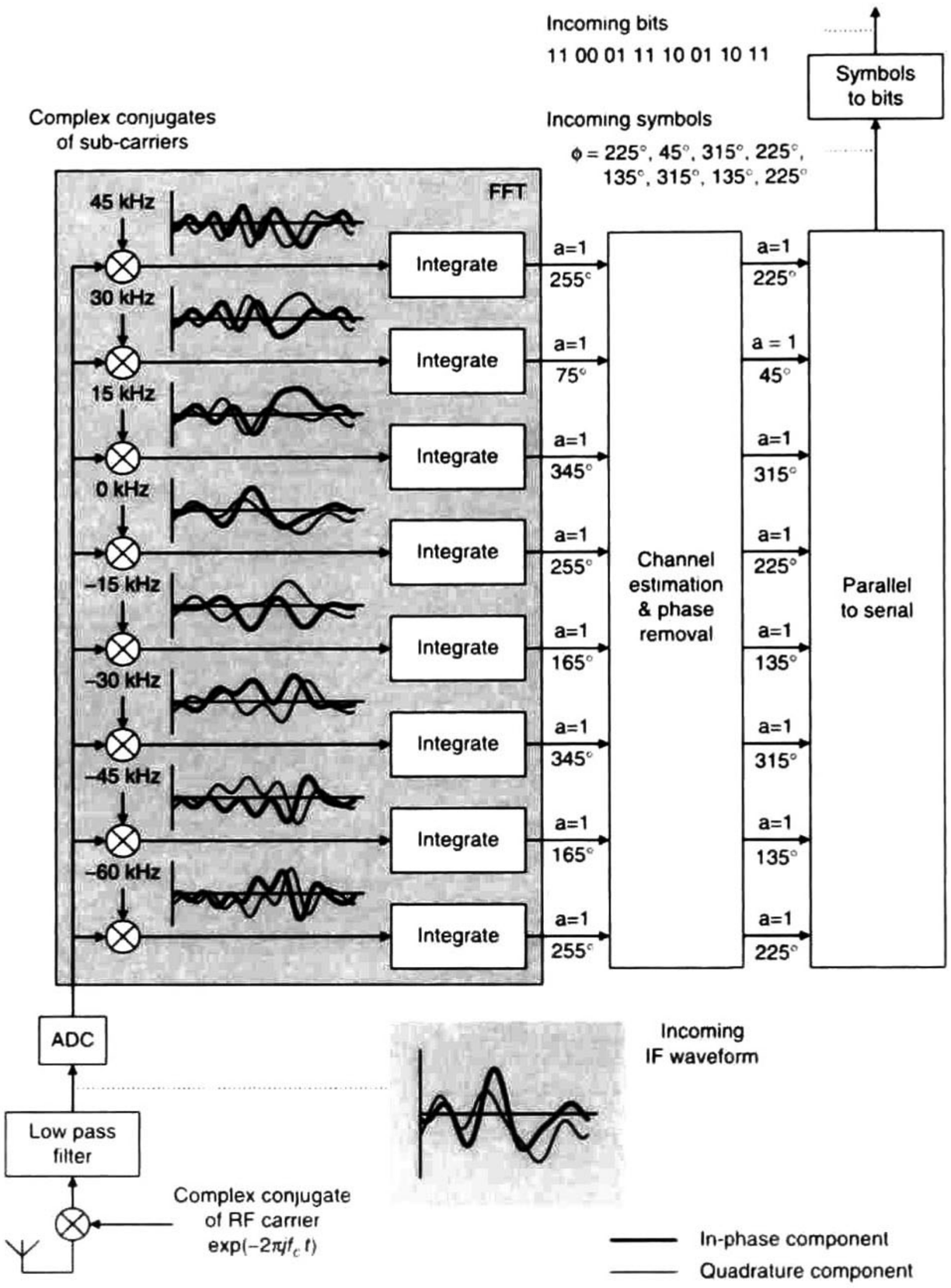


Figura 1.3: Fases de procesamiento en un receptor OFDM [6]

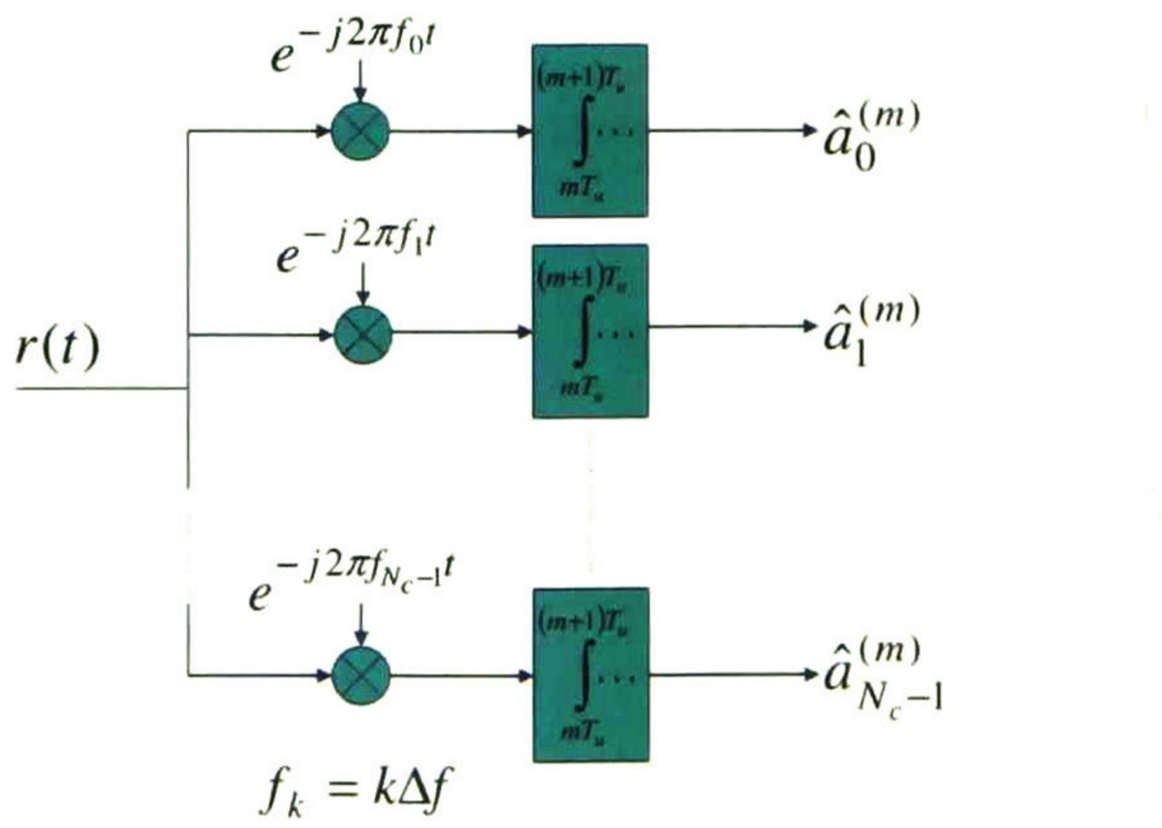


Figura 1.4: Receptor OFDM interpretado como un banco de demoduladores de frec. Intermedia [7]

1.1.2.5. Implementación OFDM utilizando procesamiento IFFT/FFT

Aunque un banco de moduladores/moduladores de frecuencia intermedia puede ser utilizado para ilustrar el principio básico de la modulación y demodulación OFDM respectivamente, no son las estructuras de modulación/demodulación más apropiadas para una implementación actual. De hecho, debido a su estructura específica y la selección de un espacio entre portadoras Δf igual a la tasa de símbolo por subportadora $\frac{1}{T}$, OFDM permite una implementación de baja complejidad por medio del algoritmo computacionalmente eficiente de la Transformada Rápida de Fourier computacionalmente eficiente [7].

Para confirmar esto, considere una señal OFDM discreta en tiempo (muestreada) donde se asume que las tasa de muestreo f_s es un múltiplo del espacio entre subportadoras Δf, esto es: $f_s = \frac{1}{T_s} = N \quad \Delta f$. El parámetro N debe ser escogido de manera que el teorema de muestro sea suficientemente cumplido. Dado que $N_c \quad \Delta f$ puede ser visto como el ancho de banda nominal de la señal OFDM, entonces N debe ser mayor que N_c por un margen suficiente [7].

Con estas suposiciones, la señal OFDM discreta puede ser expresada como:

$$x_n = x\left(nT_s\right) = \sum_{k=0}^{N_c-1} a_k e^{j2\pi kn/N} = \sum_{k=0}^{N-1} a_k^{'} e^{j2\pi kn/N} \tag{1.5}$$

Donde

$$a_k^{'} = \begin{cases} a_k & 0 \leq k < N_c \\ 0 & N_c \leq k < N \end{cases} \tag{1.6}$$

Así, la señal OFDM muestreada x_n es la Transformada Discreta de Fourier Inversa (IDFT) de longitud N del bloque de símbolos modulados (a_0, a_1, ..., a_{N_c-1}) extendido con ceros hasta la longitud N.

La modulación OFDM puede ser implementada por medio del procesamiento de la IDFT seguida de una conversión analógica-digital, como se muestra en la figura 1.5. En particular, seleccionando la IDFT de longitud N igual a 2^m para algún entero m, la modulación OFDM puede ser implementada por medio de procesamiento de la Transformada Rápida de Fourier Inversa de radix-2 [17][14][22]. La relación N/N_c típicamente es un número no entero y pude ser visto como un sobre muestreo de la señal OFDM en tiempo discreto. Por ejemplo, en LTE, el número de subportadoras N_c es de aproximadamente 600 en el caso de espectro de 10 MHz. La longitud de IFFT seleccionada puede ser de $N = 1024$. Esto corresponde con la tasa de muestreo $f_s = N \quad \Delta f = 15{,}36$ MHz, donde $\Delta f = 15$ KHz es el espacio entre subportadoras [7].

De manera similar, el procesamiento de la DFT/FFT puede ser utilizado para la demodulación OFDM, reemplazando el banco de demoduladores de frecuencia intermedia, como se muestra en la figura 1.6.

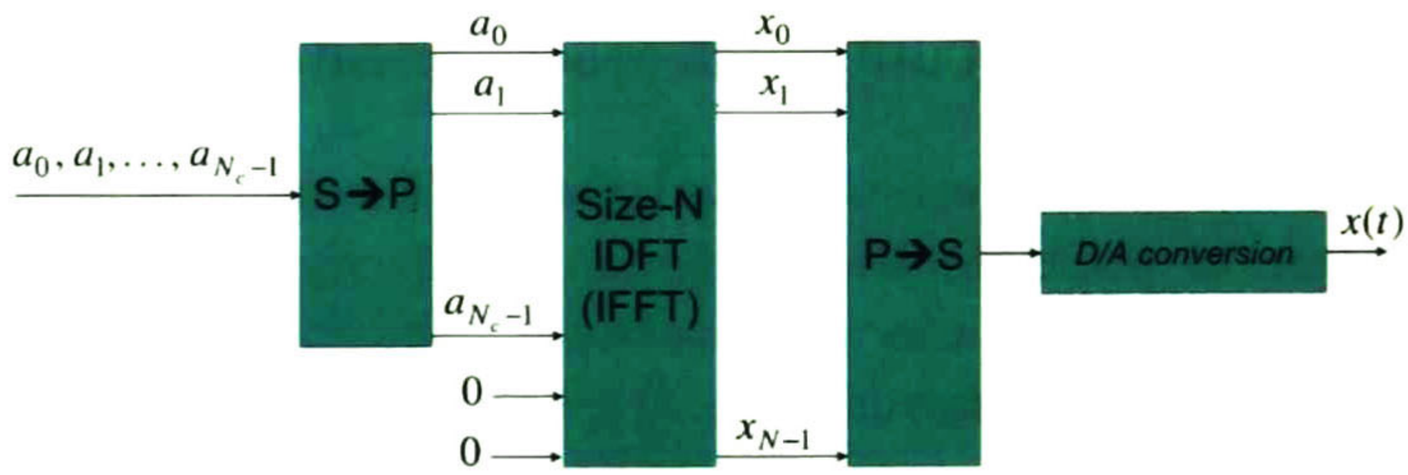


Figura 1.5: Modulación OFDM por medio del procesamiento de la IFFT [7]

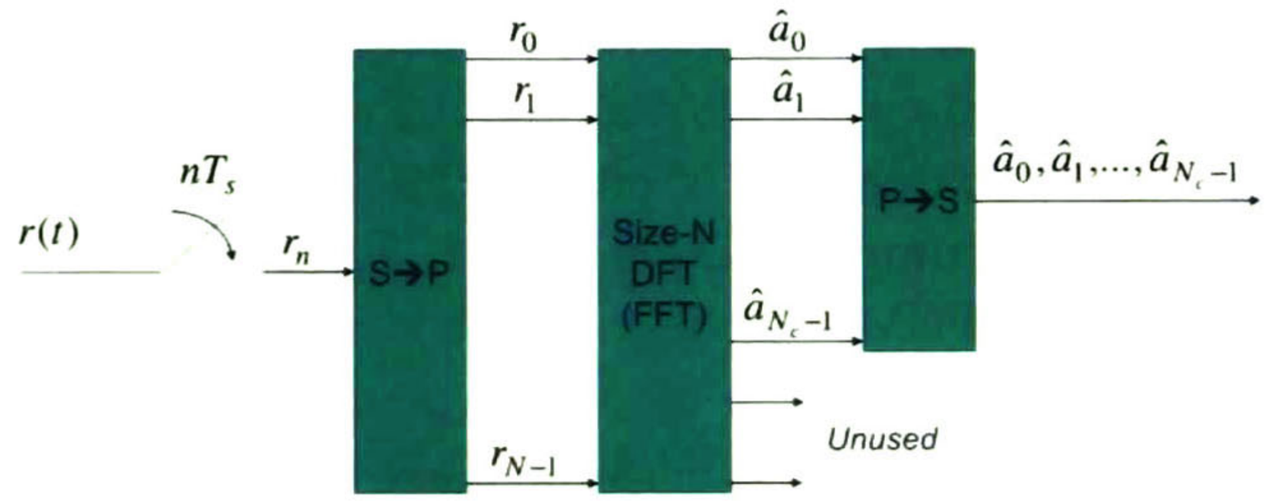


Figura 1.6: Demodulación OFDM por medio del procesamiento de la FFT [7]

1.1.2.6. Inserción de Prefijo Cíclico

Un prefijo cíclico es necesario para prevenir la interferencia con símbolos OFDM que fueron transmitidos previamente. La interferencia intersimbóloca puede ser vista como resultado directo de la propagación multitrayectoria. El prefijo cíclico (CP) ayuda a mantener la ortogonalidad entre las subportadoras en el receptor. Además, proporciona una extensión periódica de la señal OFDM a través de la cual, la operación de "convolución lineal", realizada a la señal transmitida por el canal, es aproximadamente una operación de "convolución circular". Imitar una convolución circular mediante prefijo cíclico es importante si se quiere que OFDM represente la señal modulada en el dominio de la frecuencia [25].

La validez de la ecualización, en el dominio de la frecuencia, realizada en el receptor es únicamente asegurada si la respuesta del canal puede ser vista como una convolución circular, cosa que la inserción de prefijo cíclico puede asegurar [7][25].

La inserción de prefijo cíclico implica que la última parte del símbolo OFDM es copiada e insertada al inicio del símbolo OFDM. Así, la inserción de prefijo cíclico incrementa la longitud del símbolo OFDM de T a $T + T_{CP}$, donde T_{CP} es la longitud del prefijo cíclico, con una correspondiente reducción de la tasa del símbolo OFDM como consecuencia.

En la práctica, la inserción de CP es realizada en la salida de tiempo discreta de la IFFT del transmisor. Entonces, la inserción de CP implica que las últimas N_{CP} muestras del bloque de salida de la IFFT de longitud N, sean copiadas e insertadas al inicio del bloque, incrementando la longitud del bloque de N a $N + N_{CP}$. Del lado del receptor, las muestras correspondientes son descartadas antes de la demodulación OFDM [7].

1.1.3. Parámetros definidos por el estándar LTE

1.1.3.1. Estructura de la trama LTE

La figura 1.7 muestra la trama de LTE, la cual tiene una duración de 10 ms. Las tramas son divididas en 10 subtramas de 1 ms. A su vez, las subtramas son divididas en dos ranuras de tiempo de 0.5 ms cada una. Cada ranura consta de 6 o 7 símbolos OFDM dependiendo de si se usa prefijo cíclico normal o extendido.

LTE utiliza un espacio tridimensional para administrar los recursos tiempo, frecuencia y espacio (antenas). La unidad mas pequeña es llamada Elemento Recurso (RE), el cual consiste de un periodo de duración de un símbolo OFDM por una sub-portadora. El área que consiste de 12 sub-portadoras consecutivas por una ranura de duración es llamada Bloque Recurso (RB), el cual consta de 12×7 REs para el caso de CP normal.

Existen tres tipos de información contenida en la subtrama de LTE, estas son datos de usuario, señales de referencia (pilotos) e información de control. La figura 1.8 muestra la posición de estas señales dentro de la subtrama correspondiente a una antena (modo unicast).

1.1.3.2. Planificación en el dominio de la frecuencia

LTE soporta diferentes anchos de banda, lo cuales se pueden alcanzar seleccionando diferentes longitudes de IFFT/FFT. LTE mantiene la duración del símbolo OFDM constante con un valor fijo de 66.7 μs. Esto hace posible el uso del mismo espacio entre subportadoras de 15 KHz para todos los anchos de banda. Aunque la longitud real de FFT utilizada en cada ancho de banda no es especificado por el estándar, una longitud de FFT de 2048 es usualmente asociada con 20 MHz. Las longitudes de FFT para otros anchos de banda son usualmente versiones escaladas de este valor, como se muestra en la tabla 1.1.

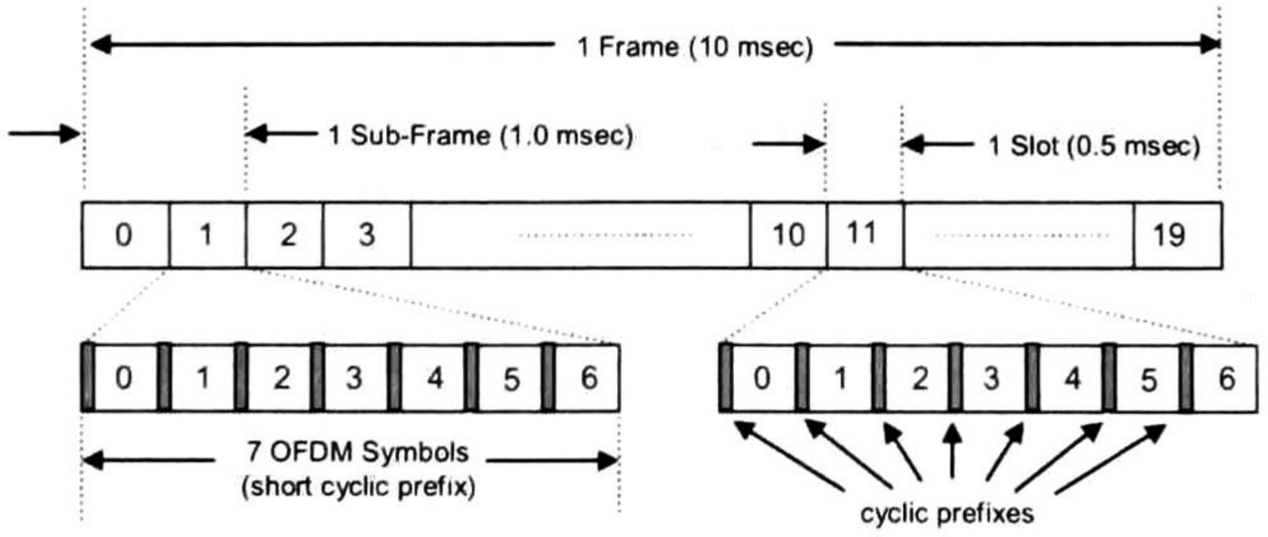


Figura 1.7: Trama *downlink* de LTE [3]

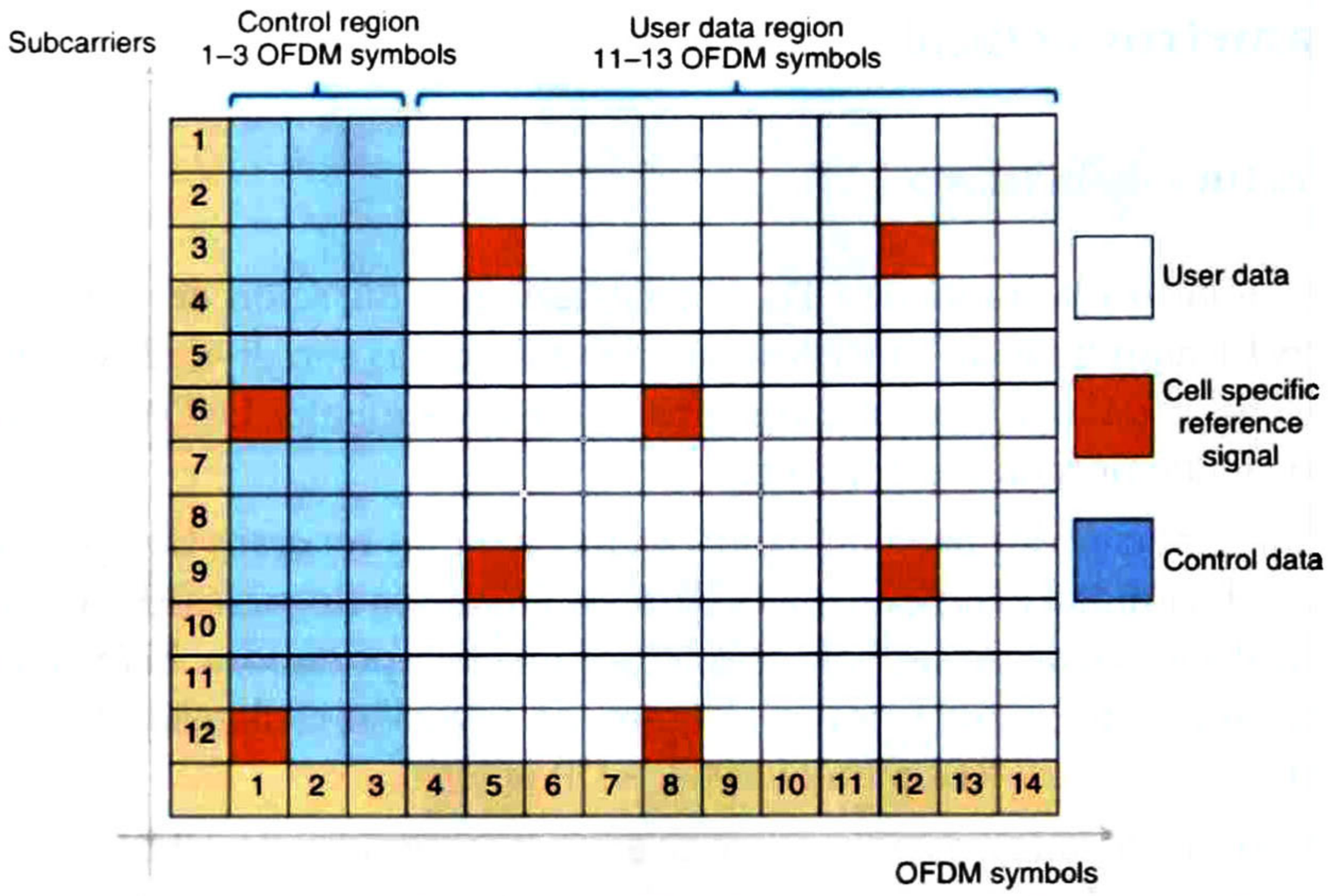


Figura 1.8: Subtrama *downlink* de LTE en modo *unicast* [25]

Tabla 1.1: Configuración de patrones para la generación de símbolos ODFM en LTE

Parámetros OFDM para transmisión *downlink*.
Duración de subtrama: 1 ms. Espacio entre subportadoras:15 KHz

Ancho de Banda [MHz]	1.25	2.5	5	10	15	20
Frecuencia de muestreo [MHz]	1.92	3.84	7.68	15.36	23.04	30.72
Longitud de FFT	128	256	512	1024	1536	2048
Número de RB	6	15	25	50	75	100
Símbolos OFDM por ranura	14/12	(CP normal /CP extendido)				
Longitud de Prefijo Cíclico [μs]	4.7/5.6	(CP normal /CP extendido)				

1.2. Estado del Arte

1.2.1. Transmisores reconfigurables

En [20] una arquitectura de transmisor configurable para para sistemas de comunicación que trabaja con entrenamiento implícito (IT) es implementado en una FPGA Xilinx Virtex-5 XC5VLX110T. Este transmisor soporta las modulaciones 4/16/64-QAM con entrenamiento superpuesto (ST) y entrenamiento superpuesto dependiente de datos (DDST), operando con un SQNR de 82 dB y una frecuencia de operación de 115 MHz. Sin embargo, la tasa de datos es fija y no considera filtro formador. En [2] un transmisor multi-estándar implementado en

una FPGA Altera Stratix V 5SGXMA7N1F45C1. Su arquitectura incorpora entrenamiento implícito y explícito en un transmisor reconfigurable para aplicaciones SDR (*Software Defined Radio*). Adicionalmente considera un filtro formador y es capaz de realizar diferentes tipos de modulación, tales como: 4/16/64 QAM, BPSK (*Binary Phase Shift Keying*), DBPSK (*Differential Binary Phase Shift Keying*), OQPSK (*Offset Quadrature Phase Shift Keying*) y DQPSK (*Differential Quadrature Phase Shift Keying*). No obstante, ambos casos, [20] y [2], sólo abordan sistemas de portadora única.

En [26] a una arquitectura reconfigurable para un modulador OFDM basado en FPGA es presentado. Dicho transmisor fue probado para los sistemas CMMB (*Chinese Mobile Multimedia Broadcasting*) y DAB (*Digital Audio Broadcasting*), pero no para aplicaciones LTE. Además, un *core IP* (*Intellectual Property*) fue utilizado para el procesamiento de la IFFT.

En [10] un modulador OFDM basado en FPGA para el estándar IEEE 802.11a es presentado. Se reporta que el procesador IFFT utilizado es un *core* Xilinx IP de 64 puntos, el cual es adecuado para el estándar IEEE 802.11a, pero, no se considera la necesidad de contar con un procesador FFT/IFFT de longitud variable para soportar otros estándares. En [21] se discute un sistema de comunicaciones inalámbrico basado en FPGA, el cual se enfoca en la implementacion de diversos bloques del sistema OFDM utilizando la herramienta DSPbuilder de Altera.

1.2.2. Procesadores FFT/IFFT de longitud variable

Al identificar que el procesador IFFT/FFT es el bloque principal de un modulador/demodulador OFDM, se investigaron las arquitecturas actuales para su implementación. Existen dos categorías de arquitecturas para procesadores FFT: las arquitecturas pipeline y las arquitecturas basadas en memoria [15][16][4][11] [12].

1.2.2.1. Arquitectura Multi-path Delay Commutator MDC

En la tabla 1.2 se muestran las principales características de la implementación de la figura 1.9 reportada por [24] en 2013. La arquitectura MDC requiere mucho más memoria que otras arquitecturas tales como la SDF (Single Delay Feedback) [13].

1.2.2.2. Arquitectura pipelined escalable

En la tabla 1.3 se muestran las principales características de la implementación de la figura 1.10 reportada por [19] en 2013. Esta implementación requiere de un reloj con frecuencia relativamente elevada, aunque utiliza sólo 2 mariposas.

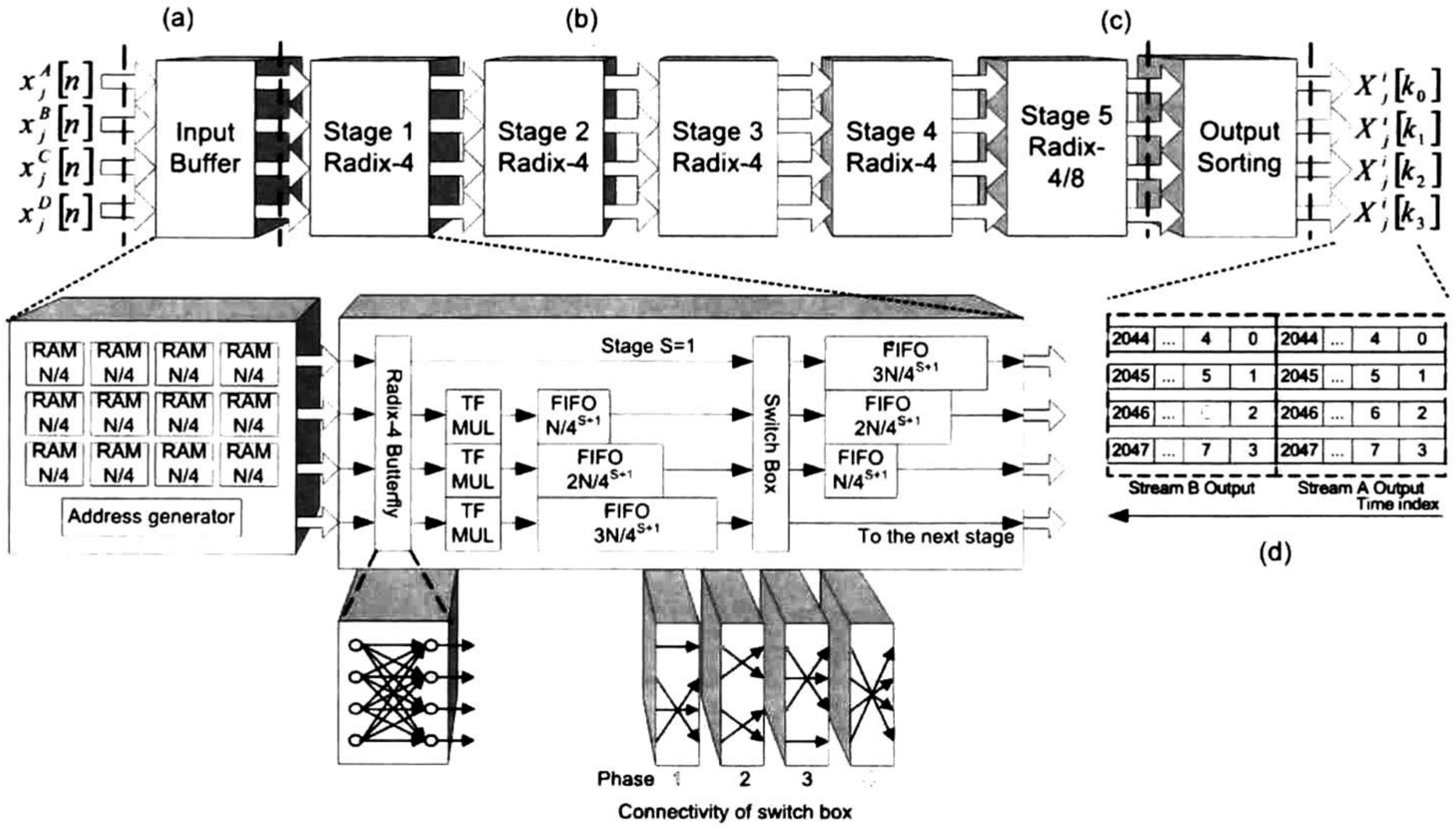


Figura 1.9: Procesador FFT. Arquitectura Multi-path Delay Commutator

Tabla 1.2: Datos relevantes de la arquitectura Multi-path Delay Commutator

Arquitectura	Multi-path Delay Commutator
Categoría	Pipeline
Longitudes de FFT [muestras]	2048,1024,512,256,128
Frecuencia Máxima [MHz]	250
Frecuencia utilizada [MHz]	40
Tecnología [nm]	90
Tiempo de ejecución [ciclos de reloj]	128 (FFT de 128 muestras) 512 (FFT de 512 muestras) 1024 (FFT de 1024 muestras) 2048 (FFT de 2048 muestras)
Longitud de palabra [bits]	8 (palabra de entrada) 12 (palabra de salida)
Número de multiplicadores complejos	12
Memoria Dual Port SRAM (palabras)	10224
Memoria Dual Port SRAM [KB]	23.56

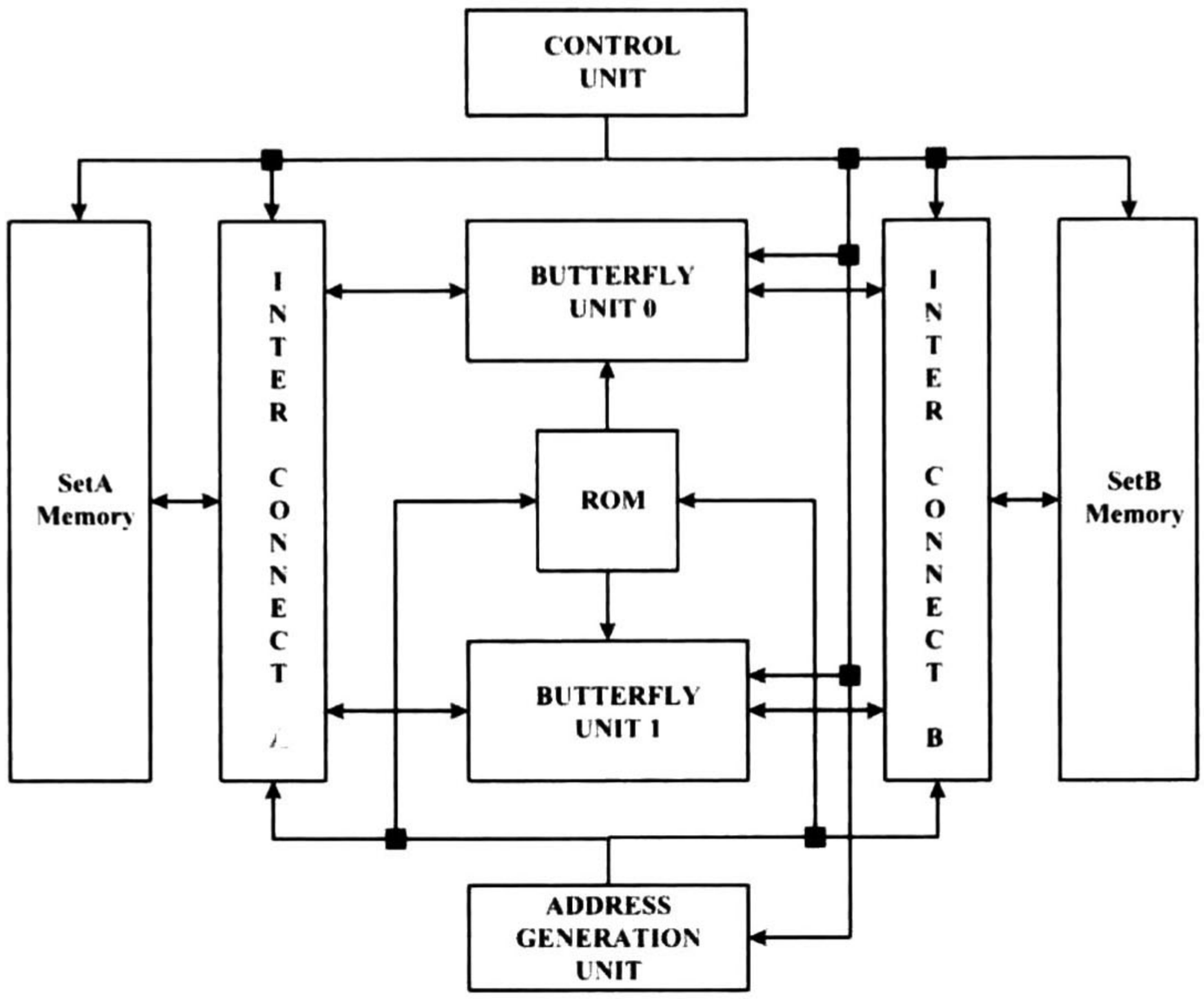


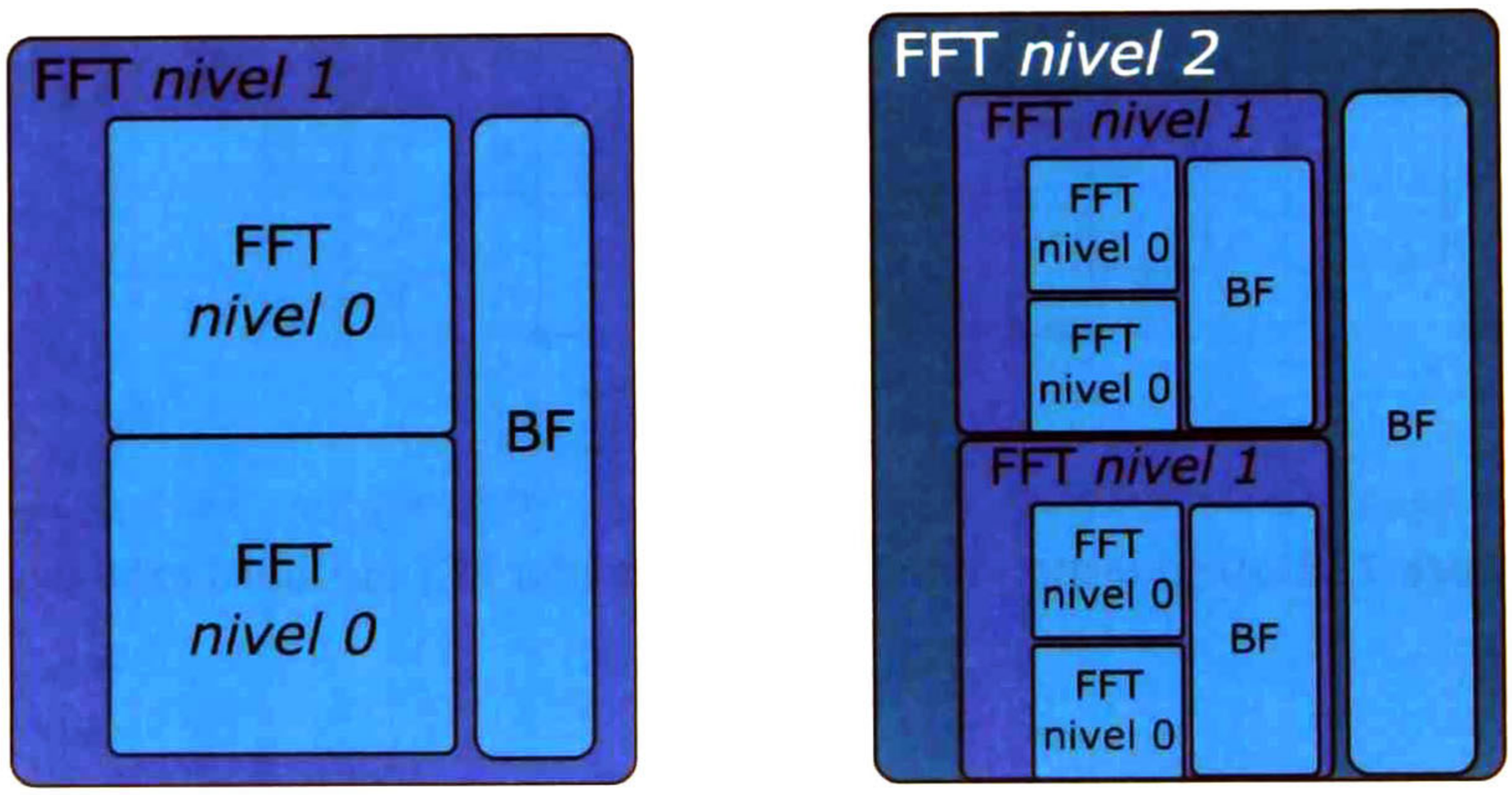
Figura 1.10: Arquitectura interna del procesador FFT pipelined escalable

Tabla 1.3: Datos relevantes de la arquitectura pipelined escalable

Arquitectura	Pipelined escalable
Categoría	Pipeline
Longitudes de FFT [muestras]	2048,1024,512,256,128,64,32,16
Frecuencia Máxima [MHz]	200
Frecuencia utilizada [MHz]	200
Tecnología	Altera Stratix V
Tiempo de ejecución [ciclos de reloj]	520 (FFT de 128 muestras) 1106 (FFT de 256 muestras) 2396 (FFT de 512 muestras) 5222 (FFT de 1024 muestras) 11376 (FFT de 2048 muestras)
Longitud de palabra [bits]	16 (palabra de entrada) 16 (palabra de salida)
Número de multiplicadores complejos	4
Memoria Dual Port SRAM (palabras)	4096
Memoria Dual Port SRAM [KB]	16

1.2.2.3. Procesador FFT: Arquitectura multinúcleo

La arquitectura multinúcleo de la figura 1.11a esta basada en el algoritmo Radix-2 de Diezmado en Tiempo. Esta arquitectura permite obtener un procesador FFT de nivel 1 de longitud $N/2$ combinando dos FFT´s de longitud $N/4$ mediante una última etapa de operaciones mariposa (**BF**). Si esta arquitectura se replica recursivamente, de modo que dos procesadores nivel 1 de longitud $N/2$ son utilizados para conformar uno de longitud N, se obtiene un procesador de nivel 2. Esto es mostrado en la figura 1.11b [17].



(a) (b)

Figura 1.11: FFT Radix-2: Arquitectura multinúcleo

1.3. Motivación

El tráfico de datos móviles se ha incrementado dramáticamente desde 2010. La figura 1.12 muestra mediciones de tráfico realizadas por Ericsson a nivel mundial, en petabytes por mes [18]. Además se anticipa que esta tendencia va a continuar. Como resultado de esta cuestión, las redes 2G y 3G comenzaron a suturarse al rededor de 2010, dando lugar a la necesidad de aumentar la capacidad de red.

Existen 3 maneras de incrementar la capacidad de un sistema de comunicación móvil. La primera, es utilizar células más pequeñas. La segunda consiste en incrementar el ancho de banda. Sin embargo, existe una cantidad finita de espectro disponible. La tercera técnica es mejorar la tecnología de comunicación existente [6]. Siendo esta última, la motivación para trabajar en tecnologías de cuarta generación como OFDM, el cual es empleado por uno de los estándares mayormente utilizados tales como 3GPP-4G LTE.

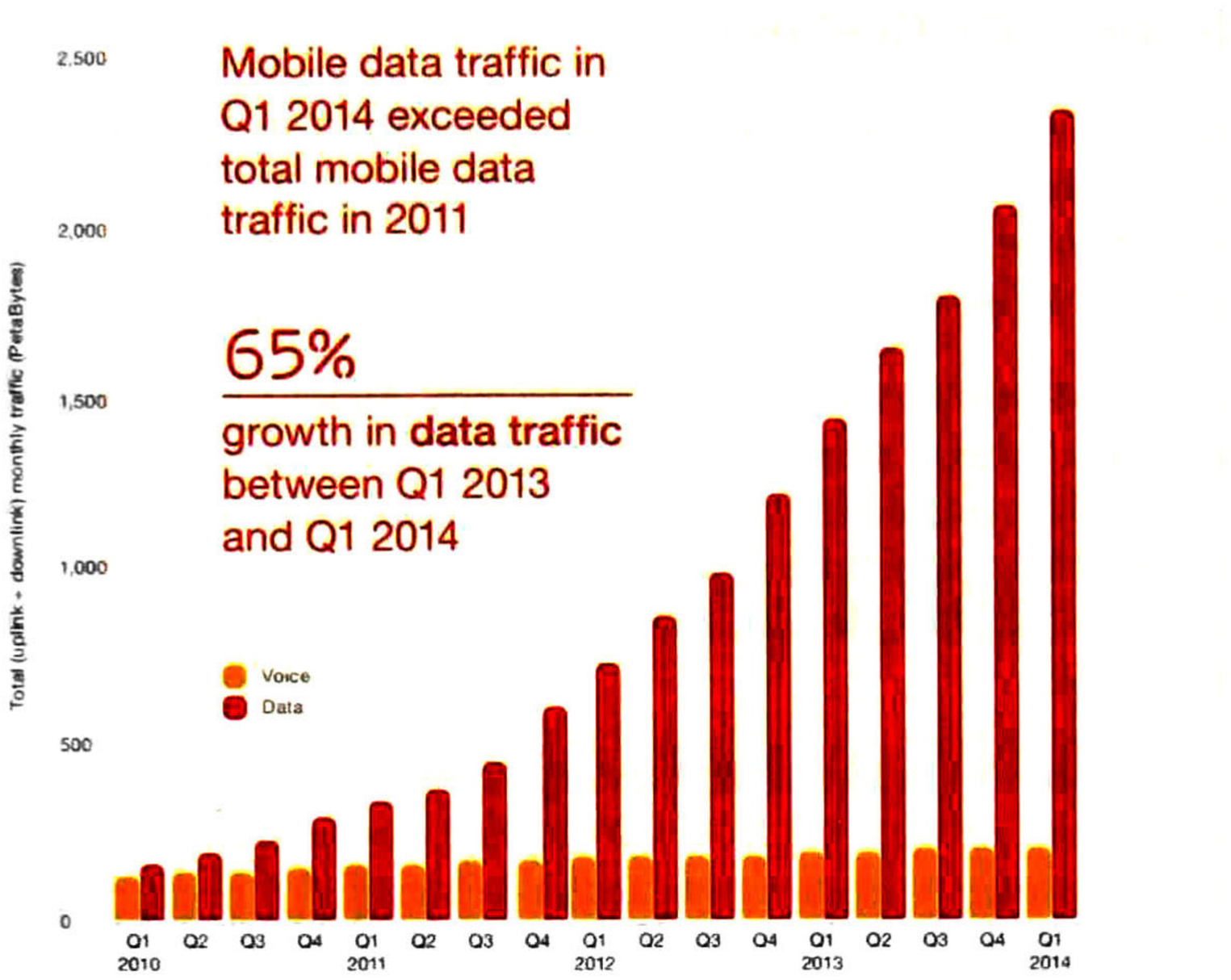


Figura 1.12: Mediciones de tráfico de voz y datos en redes de telecomunicación móvil en el periodo 2010 a 2014.[18]

1.4. Definición del Problema

Después de revisar la literatura, se identificó la necesidad de implementar, en hardware, arquitecturas de los módulos utilizados en los sistemas de comunicación móvil de cuarta generación. Esto se debe a que la mayoría de las propuestas reportadas en la sección 1.2, se menciona la utilización de bloques de propiedad intelectual de Xilinx o Altera. Es por ello que se plantea como problema implementar bloques propios. Particularmente, módulos necesarios para conformar un Transmisor OFDM reconfigurable. La reconfigurabilidad de este módulo debe permitirle ser capaz de soportar cambios dinámicos en el ancho de banda, en el tamaño de la trama y en la manera de generar los símbolos OFDM, garantizando que puede ser utilizado para la transmisión de antena única del estándar 4G LTE.

1.5. Objetivos

1.5.1. Objetivo General

Analizar y simular un Transmisor OFDM en punto fijo, mediante Simulink y Matlab e implementar en HDL los bloques particulares del Transmisor OFDM.

1.5.2. Objetivos Particulares

1. Modelar, diseñar e implementar un procesador (I)FFT de longitud variable en punto fijo. El procesador debe satisfacer las longitudes de 128, 256, 512, 1024 y 2048 muestras. El tiempo de procesamiento de este módulo debe ser capaz de soportar las tasas de transmisión exigidas por LTE.

2. Diseñar e implementar un generador de tramas. Aunque la implementación debe ser realizada para generar tramas del estándar LTE; el diseño debe ser reconfigurable, de modo que la trama de LTE sea sólo un caso particular. La reconfigurabilidad debe permitir modificar el tamaño de la trama, el número de portadoras nulas, la longitud del prefijo cíclico y las posiciones en las que datos y pilotos son distribuidos en un símbolo OFDM.

3. Implementar un transmisor OFDM que sea capaz de generar las tramas LTE para los anchos de banda de 1.25, 2.5, 5, 10 y 20 MHz. La señal transmitida debe satisfacer la tasa de transmisión asociada a cada ancho de banda.

1.6. Perspectiva de los capítulos

En el capítulo 1, se aborda la base teórica de OFDM y LTE, así como las arquitecturas actuales para implementar sus bloques principales. En el capítulo 2, se describe el modelo de oro y de referencia del Transmisor reconfigurable OFDM. En el capítulo 3, se explica la arquitectura del Transmisor OFDM y sus bloque más importantes, en particular, el procesador FFT/IFFT de longitud variable y sus distintas versiones realizadas. En el capítulo 4, se abordan los resultados de desempeño tales como tiempo de procesamiento, métricas de *Signal-to-Quantization-Noise Ratio* (SQNR) y cumplimiento de los requerimientos de diseño del transmisor OFDM y sus bloques principales. Finalmente, en el capítulo 5, se exponen las conclusiones y el trabajo a futuro de esta tesis.

Capítulo 2

Modelo de oro y referencia del Transmisor reconfigurable para sistemas OFDM

El modelo del transmisor mostrado en la figura 2.1 fue implementado en Simulink. Para su elaboración, se consideró la ranura de LTE para formar los símbolos OFDM conforme a sus especificaciones. Internamente, el modelo genera una secuencia aleatoria para representar los datos de usuario a ser transmitidos. Estos datos son procesados para obtener las muestras del símbolo OFDM y su respectivo prefijo cíclico, las cuales salen por los puertos **TxSignal** y **TxSignalFP**. La diferencia entre ambas radica en que la primera es representada como una cantidad compleja, mientras que la segunda en punto fijo. Además, los datos de usuario (sin procesar) son enviados a través de la salida **Txbits** para fines de análisis de rendimiento del sistema completo, contemplando al canal y al receptor de OFDM, los cuales no son discutidos en este trabajo. Para este modelo, los bloques configurados en punto fijo fueron los moduladores de QAM y el procesador IFFT. Los resultados de simulación muestran que con un tamaño de palabra de 16 bits, 15 bits de parte fraccionaria y 1 bit de signo, el SQNR promedio de la salida **TxSignalFP** respecto a **TxSignal**, se mantiene en un valor por encima de 50 dB.

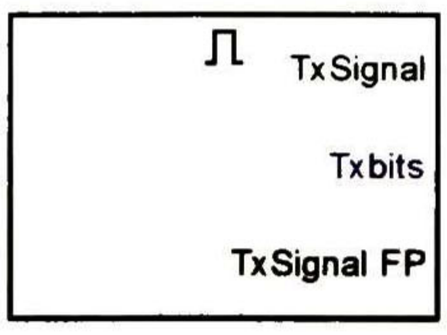


Figura 2.1: Diagrama del modelo del transmisor OFDM

Tabla 2.1: Parámetros configurables del modelo del Transmisor OFDM

Parámetros de Configuración	Descripción
Ancho de banda	La tabla 2.2 muestra los posibles valores del parámetro *BW*, el cual representa el ancho de banda ocupado por el símbolo OFDM.
Esquema de modulación	Los datos de usuario y de control son modulados mediante uno de los esquemas denotados por el parámetro *Mary*, cuyos posibles valores son mostrados en la tabla 2.2.
Tamaño de la trama	El tamaño de la trama es definido por dos parámetros. El primero es el número de ranuras dentro de la trama (N_{slots}); el segundo es el número de símbolos OFDM por ranura (N_{Sy}).
Número de portadoras nulas	El número de portadoras nulas (N_{nc}) determina la cantidad de ceros necesarios para extender la longitud del símbolo OFDM de N_c a N, como se mencionó en la sección 1.1.2.5. Este parámetro está definido por: $N_{nc} = N - N_c$ y $N_c = N_{RB} \cdot N_{c_{RB}}$. Donde N es la longitud de FFT asociada al ancho de banda elegido, N_c es el número de sub-portadoras útiles, es decir, las portadoras que son moduladas por datos y pilotos, N_{RB} es el número de RBs mostrado en la tabla 1.1, y $N_{c_{RB}}$ es el número de sub-portadoras contenidas en un RB, el cual tiene un valor de 12.
Pilotos	Se debe establecer N_p como el número de pilotos y definir el valor de estos
Longitud del Prefijo Cíclico	Este parámetro define la cantidad de muestras utilizadas para conformar el prefijo cíclico, el cual es un número entero definido en el rango $0 \leq CP\_length < N$.

2.1. Parámetros de Configuración

El transmisor OFDM debe su reconfigurabilidad a los parámetros descritos en la tabla 2.1. Aunque el modelo esta basado en sistemas LTE, es posible modificar estos parámetros para alcanzar distintas configuraciones alternas y de este modo, dar soporte a actualizaciones de LTE o en su defecto, migrar a otro estándar.

2.1.1. Configuración de los símbolos OFDM

En LTE se han identificado tres maneras distintas de formar un símbolo OFDM, es decir, tres patrones en los que datos y pilotos ocupan subportadoras específicas dentro de un símbolo. En la figura 1.8, el **patrón 1** está presente en los símbolos OFDM 1 y 8; el **patrón 2**, en los símbolos 5 y 12; y el **patrón 3**, en los 10 simbolos restantes. Los tres patrones incluyen datos. Sin embargo, la diferencia radica en los pilotos. El **patrón 1** incluye

Tabla 2.2: Parámetros configurables del modelo del Transmisor OFDM

Parámetro	Descripción	valores	Parámetro	Descripción	valores
Mary	Modulación	4-QAM	*BW*	Ancho de banda	1.25 MHz
		16-QAM			2.5 MHz
		64-QAM			5 MHz
					10 MHz
					20 MHz

pilotos separados por 5 subportadoras. El **patrón 2** es similar al **patrón 1**, sólo que la subportadora a partir de la cual se comienzan a insertar pilotos, es diferente. Es decir, en el **patrón 2**, las posiciones de los pilotos están desfasadas respecto al **patrón 1** por tres subportadoras. Finalmente, el **patrón 3** no incluye pilotos. Para generar los **patrones**, se definieron tres parámetros: *pilotDist*, referente al número de subportadoras que separan a un par de pilotos dentro del símbolo OFDM; *pilotOffset*, que permite modificar la posición inicial de los pilotos y finalmente *enaPilots*, que es una bandera que indica la presencia o no de pilotos dentro del símbolo OFDM. De este modo, modificando el valor de estos tres parámetros se puede cambiar el **patrón** y en consecuencia la ranura de tiempo. La tabla 2.3 muestra la configuración que deben tener los tres parámetros para formar símbolos OFDM bajo el estándar LTE. La formación de los símbolos OFDM es como sigue:

Antes de ser modulado mediante la IFFT, un símbolo OFDM es una secuencia de ceros, datos y pilotos colectivamente llamados símbolos[1] Cada símbolo de esta secuencia es mezclado con una subportadora. Dado que existen N subportadoras numeradas desde 1 hasta N, el símbolo OFDM, en otras palabras, la secuencia de símbolos, es de longitud N. Así, el número de subportadora puede interpretarse como un índice que ayuda a posicionar un símbolo dentro de la secuencia. Dentro del símbolo OFDM existen tres intervalos de subportadoras nulas, donde se insertan ceros. Dichos intervalos son:

- Las subportadoras 1 a $\frac{N_{nc}}{2}$.
- La subportadora $N/2$ (central).
- Las subportadoras $N - \frac{N_{nc}}{2} + 2$ a N.

Si **ena_pilots** no es acertada, se insertan únicamente datos en el resto de las subportadoras. En caso contrario, datos y pilotos son insertados de la siguiente manera: a medida que se hace un recorrido desde la subportadora 1 hasta la N, se realiza un conteo que inicia desde 1. Durante el conteo se van insertando datos en las subportadoras. Sin embargo, cuando el conteo llega hasta *pilotDist*, en lugar de insertar un dato, se inserta un piloto

[1]Cabe aclarar la distinción entre símbolos y símbolos OFDM. Una secuencia de N símbolos forma un símbolo OFDM.

y el conteo se reinicia en 1. Este conteo es repetido hasta terminar el recorrido de las N portadoras. Cabe aclarar que el intervalo de subportadoras nulas tiene prioridad, por lo tanto en estos intervalos siempre se insertan ceros sin importar si el conteo indica que se debe insertar un dato o un piloto. No obstante, es importante llevar el seguimiento de este conteo desde la subporatadora 1, para que al momento de llegar a un intervalo no destinado para subportadoras nulas, se sepa si se debe insertar un piloto o un dato. Además, se debe pausar el conteo durante la portadora $N/2$. Finalmente el parámetro *pilotOffset* se utiliza para alterar la suportadora a partir de la cual inicia el conteo. Para un valor de *pilotOffset* de n, el conteo inicia en la subportadora $n + 1$, para algún n entero.

2.2. Modelo del Transmisor OFDM

El modelo de oro y el modelo de referencia del transmisor OFDM son mostrado en la figura 2.2. Las partes que lo conforman son: un generador de números aleatorios, un mapeador de símbolos OFDM, un modulador de OFDM y un bloque para insertar CP. A continuación se presenta una explicación de estos módulos.

2.2.1. Generador de números aleatorios

Este bloque se encarga de generar números enteros de manera aleatoria, los cuales representan datos y se encuentran en el rango de 0 a $Mary - 1$. Un *buffer* es utilizado para almacenar bloques de N_c datos y dirigirlos a la etapa de mapeo de símbolos OFDM.

2.2.2. Mapeador de símbolos OFDM

El objetivo del mapeador es entregar la secuencia a la que se e debe aplicar la modulación OFDM. El mapeador consta de los bloques nombrados como **patrón1**, **patrón2** y **patrón3** así como los **switch multipuerto**. Estos bloques se encargan de generar la secuencia de ceros, pilotos y datos en el orden descrito en la sección 2.1.1. Los bloques que conforman al mapeador son presentados a continuación.

Tabla 2.3: Configuración de patrones para la generación de símbolos ODFM en LTE

	patrón 1	**patrón** 2	**patrón** 3
pilotOffset	0	3	0
pilotDist	6	6	0
enaPilots	1	1	0

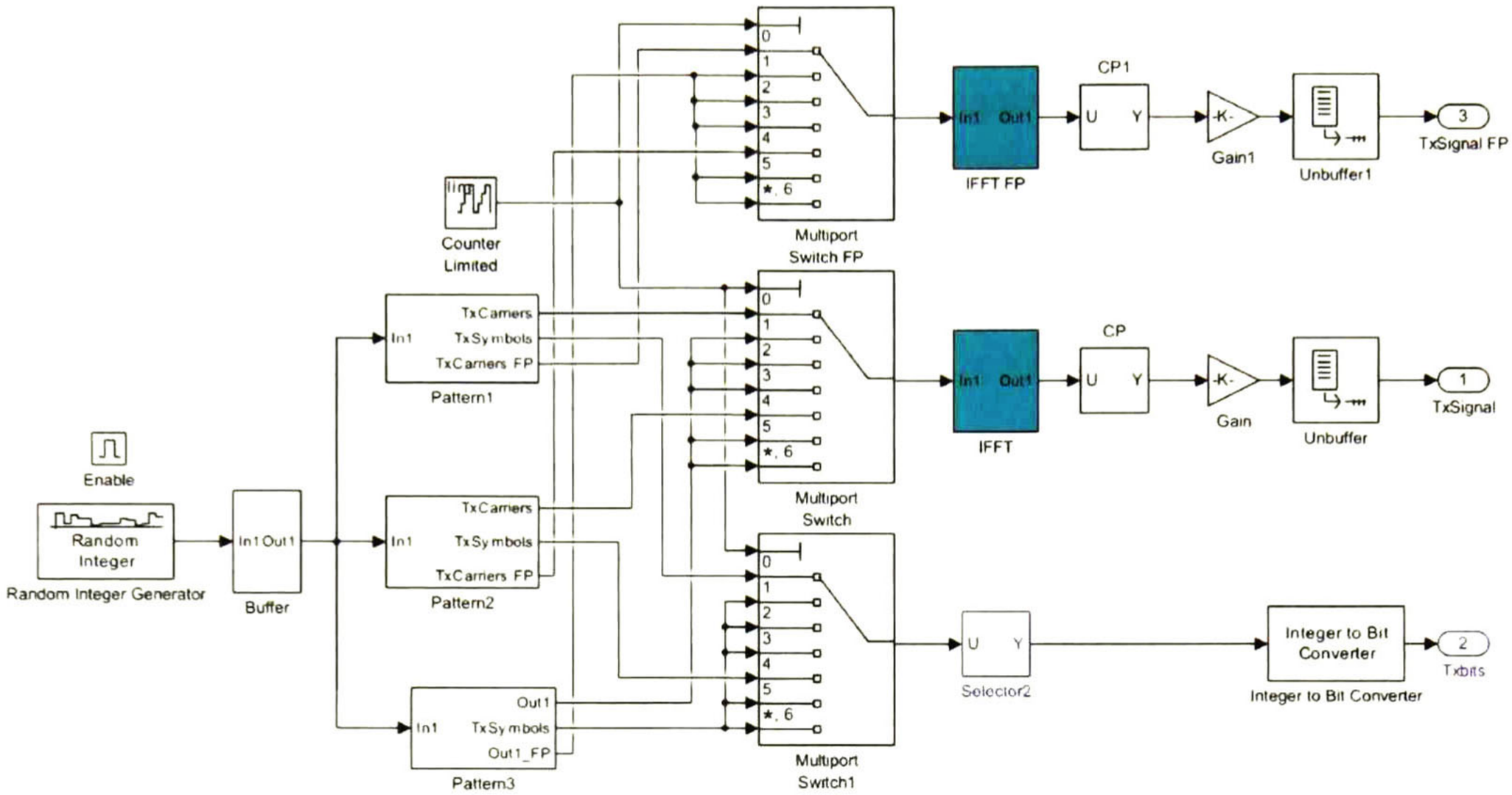


Figura 2.2: Modelo en Simulink del transmisor OFDM reconfigurable

2.2.2.1. Bloque patrón

Este bloque es mostrado en la figura 2.3 y se encarga de acomodar los ceros, datos y pilotos en la portadora correspondiente. Además de modular los datos mediante el esquema definido por *Mary*. Para **patrón1** y **patrón2**, el flujo de N_c datos es diezmado, eliminando los últimos N_p datos mediante el bloque **selector**. Los datos eliminados son reemplazados por pilotos y no son considerados para la transmisión. En el caso de **patrón3** no se realiza este proceso de eliminación porque en este patrón sólo se transmiten datos. Los datos que no fueron eliminados, son enviados a dos moduladores de QAM, uno en punto fijo y otro en representación de punto flotante. El orden de la constelación de la modulación es Gray y los factores de normalización son mostrados en la tabla 2.4. Por su parte, los pilotos insertados son modulados en BPSK y normalizados con el factor $\frac{1}{\sqrt{5}}$. La configuración de punto fijo para las componentes **I** y **Q** del modulador fue de 16 bits de tamaño de palabra, 15 bits para la parte fraccionaria y un bit para signo. Posteriormente, los datos modulados son enviados al **formador de símbolos** de la figura 2.4, el cual concatena N_{nc} ceros, $N_c - N_p$ datos y N_p pilotos para el caso de **patrón1** y **patrón2**. Sin embargo, para **patrón3** sólo se concatenan N_{nc} ceros y $N - N_{nc}$ datos. Después de la concatenación, ceros, datos y pilotos son reacomodados en la subportadora correspondiente, según la descripción de la sección 2.1.1.

2.2.2.2. Switch multipuerto

Este módulo se encarga de elegir una secuencia de las tres posibles que provienen de los bloques **patrón**. La elección de la secuencia corresponde con la figura 1.7, de modo que para el símbolo OFDM 1, se toma la secuencia de **patrón1**; para el símbolo OFDM 5, se toma la secuencia de **patrón2**; y para el resto de los símbolos se toma la secuencia de **patrón3**. La secuencia elegida es enviada al modulador OFDM.

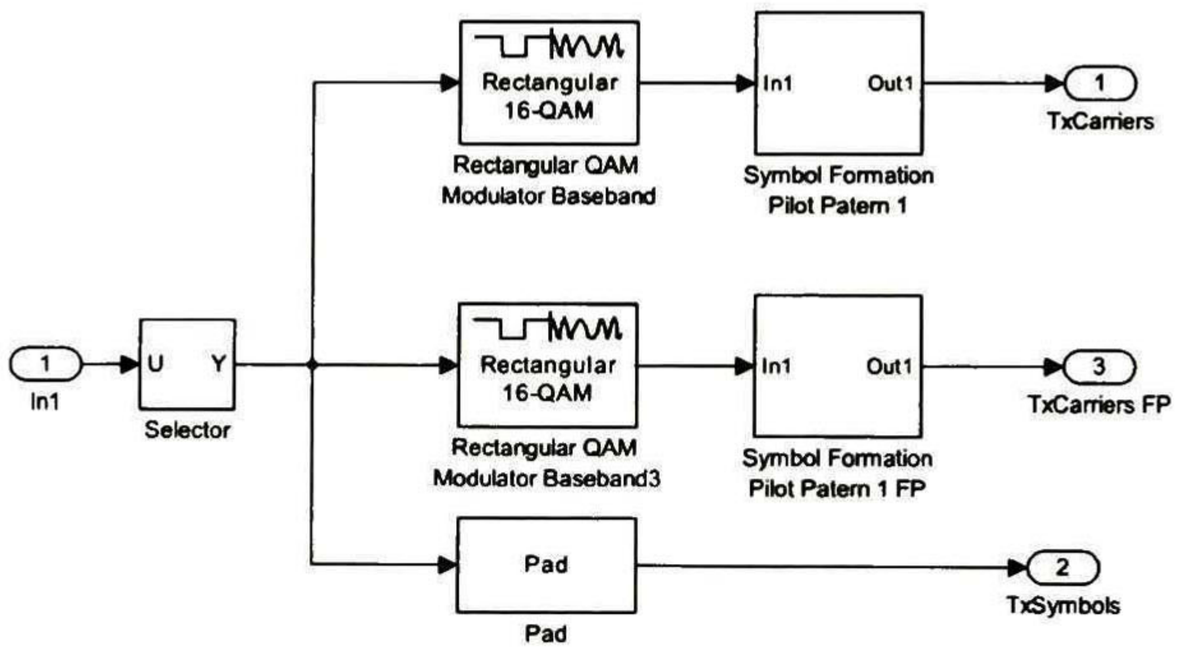


Figura 2.3: Bloque formador de patrón OFDM

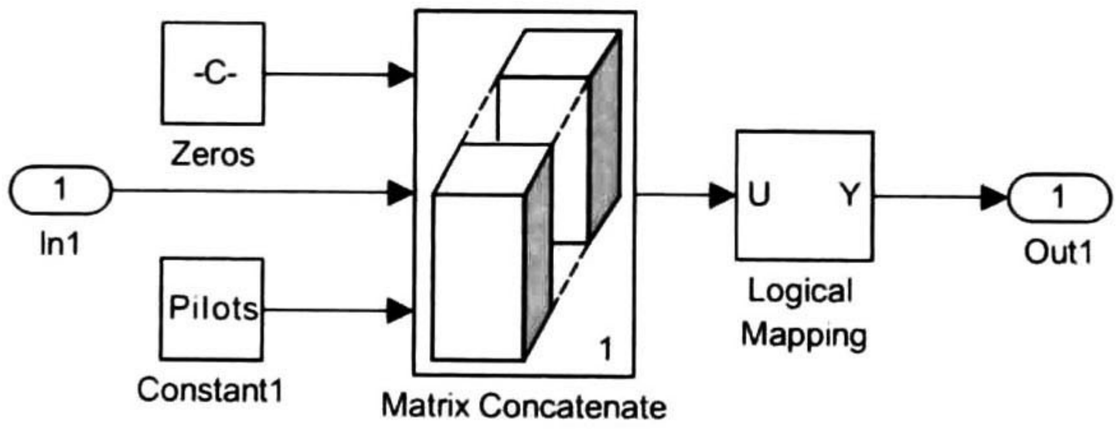


Figura 2.4: Bloque **Symbol Formation**

Tabla 2.4: Factores de normalización para el modulador QAM.

Esquema de modulación	Factor de normalización
4-QAM	$1/\sqrt{2}$
16-QAM	$1/\sqrt{10}$
64-QAM	$1/\sqrt{42}$

2.2.3. Modulador de OFDM

Este bloque realiza la modulación de OFDM mediante la IFFT. Para ello, la secuencia de longitud N que proviene del switch multipuerto pasa por un bloque que intercambia las muestras 1 a $N/2$ con las muestras $N/2+1$ a N. La secuencia intercambiada es procesada por un bloque IFFT que normaliza la salida con el factor $1/N$, obteniéndose como resultado un símbolo OFDM de longitud N. La configuración de punto fijo para la parte real e imaginaria del procesador IFFT fue de 16 bits de tamaño de palabra, 15 bits para la parte fraccionaria y un bit para signo.

2.2.4. Bloque de inserción de CP

Este bloque toma al símbolo OFDM y copia las últimas N_{CP} muestras para concatenarlas al principio del símbolo OFDM, extendiendo su longitud de N a $N + N_{CP}$.

2.2.5. *Buffer* de salida

Este módulo toma el símbolo OFDM con su respectivo CP y comienza a transmitir cada muestra que los compone a la frecuencia de muestreo establecida en la tabla 1.1.

Conclusiones del capítulo

En conclusión, el modelo del transmisor OFDM espera recibir una secuencia de N_c datos pertenecientes a algún alfabeto de modulación tal como 4, 16 o 64 QAM. Como respuesta, el modelo generar un símbolo OFDM de longitud N y su respectivo prefijo cíclico de longitud N_{CP}, que en conjunto son enviados a una salida serial que transmite las $N + N_{CP}$ muestras complejas.

Capítulo 3

Arquitectura del transmisor reconfigurable para sistemas OFDM

3.1. Transmisor OFDM reconfigurable

3.1.1. Descripción de caja negra

El módulo **`transmisor OFDM reconfigurable`** es mostrado en la figura 3.1. Las entradas y salidas de este módulo son presentadas en las tablas 3.1, 3.2, 3.3 y 3.4.

La configuración de este módulo se realiza haciendo uso de las señales **InData**, **sel_mem**, **BW** y **wr**. El objetivo es escribir los registros de configuración mencionados en la tabla 3.4, la cual muestra los posibles valores de la señal **sel_mem**.

En el caso de los registros de configuración **`NCR`** y **`CPR`**, se debe establecer el valor del parámetro que se desea escribir a través del puerto **InData** y seleccionar el ancho de banda **BW** al que estará asociado. La escritura se hace efectiva activando la señal **wr**. Este proceso debe ser repetido para el resto de los anchos de banda.

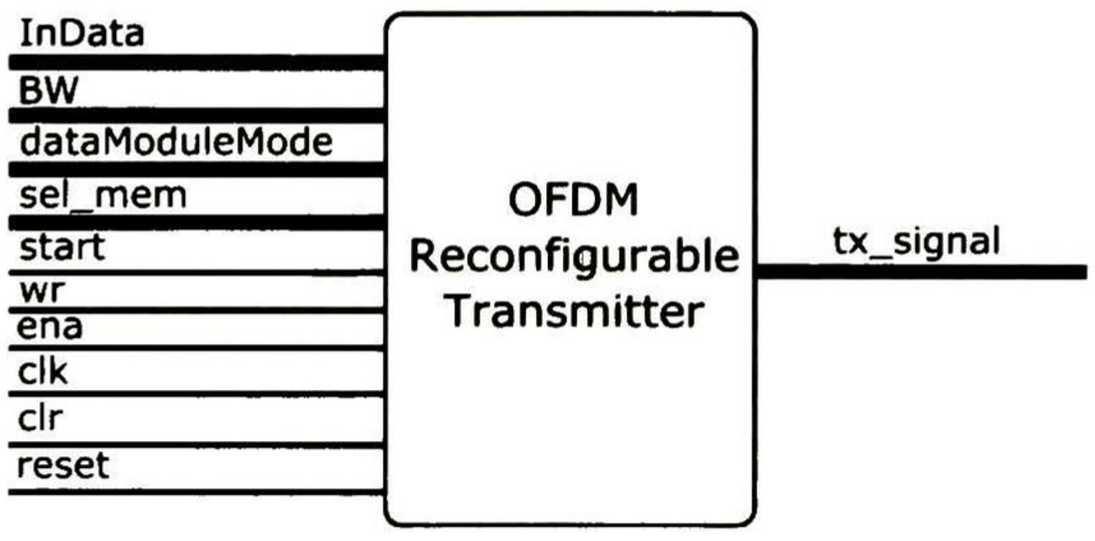


Figura 3.1: `Transmisor OFDM reconfigurable`: Diagrama de entradas y salidas

Para escribir el registro **FR** se necesitan dos pulsos en alto de la señal **wr**. En cada pulso se debe establecer el valor de la señal **InData**, durante el primer pulso, se debe establecer el valor del *número de ranuras de tiempo* (N_{slots}) presentes en la trama; mientras que durante el segundo, se debe establecer el valor del *número de símbolos OFDM* (N_{Sy}) presentes en cada ranura de tiempo.

Para escribir el registro **SlotR** se necesitan N_{Sy} pulsos de la señal **wr**. En cada pulso, la entrada **InData** debe tener el valor del **patrón** para la generación de símbolo OFDM, el cual es una palabra formada a partir de los 11 bits del parámetro *pilotOffset*, los 11 bits de *pilotDist* y un bit de *enaPilots*. La figura 3.2 muestra los posibles valores de este **patrón** en el estándar LTE.

El registro **PR** esta dividido en bancos de memoria de 8 localidades. Cada banco está asociado asociado a un ancho de banda. Por lo tanto, se pueden realizar hasta 8 escrituras por cada ancho de banda. Para cargar un banco, se selecciona un ancho de banda y se define una palabra de 32 bits en el puerto **InData** cada vez que se active la señal **wr**. La primer palabra ingresada contendrá a los pilotos 1 a 32, la segunda palabra contendrá a los pilotos 33 a 64 y así sucesivamente. El MSB de cada palabra es el primer piloto de cada conjunto.

El registro **DR** se escribe definiendo una palabra en el puerto **InData** y activando la señal **wr**. Esta palabra conforma un conjunto de datos. El número de bits requeridos para un dato depende del tipo de modulación. Por lo tanto, la cantidad de datos en la palabra esta definida en función de **dataModuleMode**, la tabla 3.5 muestra esta relación. Los bits más significativos (MSB) de cada palabra son considerados como el primer dato de cada conjunto. En consecuencia, en la modulación 64-QAM, los dos bits menos significativos (LSB) son descartados. Es decir, no son considerados como parte del conjunto de datos. La primer palabra ingresada se considera como el primer conjunto de datos; la segunda, como el segundo conjunto de datos; y así sucesivamente.

Después de haber configurado al **transmisor OFDM** y haber ingresado al menos la cantidad de datos necesarios para formar una ranura de tiempo, la señal **start** debe ser activada. Después de un tiempo, la señal a transmitir será definida en el puerto **tx_signal**. El ancho de banda **BW** configurado, define la tasa de transmisión. Los datos para conformar el resto de la trama pueden ser ingresados después de la activación de la señal **start**.

3.1.2. Descripción de caja blanca

En la figura 3.3 se muestra el diagrama de módulos funcionales que componen al **transmisor OFDM reconfigurable**. Estos módulos son listados en la tabla 3.6.

Tras activar la señal **start**, el conformador de trama **FC** comienza a generar las muestras y direcciones para el **procesador IFFT**, a través de los puertos **sample_FFT** y **addr_FFT**. Éstas direcciones son codificadas por el módulo **IFFTshift** y son entregadas al **procesador IFFT**. Simultáneamente, **FC** activa la señal **ena_mem_fft**, la cual es utilizada para habilitar la escritura en el **procesador IFFT**. Cuando **FC** concluye con ésta escritura, deja de generar

Tabla 3.1: Entradas y salidas del módulo **`Transmisor OFDM reconfigurable`**

Nombre	Descripción	bits
	Entradas	
InData	Bus de datos de entrada	32
BW	Código para selección de ancho de banda	3
dataModuleMode	Código para selección del tipo de modulación de datos	2
sel_mem	Código para selección del banco de memoria	3
wr	Señal de habilitación de escritura	1
start	Señal de inicio, activa en alto	1
clk	Señal de reloj	1
ena	Señal de habilitación global del módulo, activa en alto	1
clr	Señal de reinicio síncrono, activa en alto	1
clr	Señal de reinicio asíncrono, activa en bajo	1
	Salidas	
tx_signal	Bus para la señal transmitida	32

Tabla 3.2: Valores de selección de la señal **BW**

Código **BW**	Ancho de banda [MHz]
3'b000	1.25
3'b001	2.5
3'b010	5
3'b011	10
3'b100	20

Tabla 3.3: Valores de selección de la señal **dataModuleMode**

Código **dataModuleMode**	Tipo de modulación
2'b00	4-QAM
2'b01	4-QAM
2'b10	16-QAM
2'b11	64-QAM

Tabla 3.4: Valores de selección de la señal **sel_mem**

Código **sel_mem**	Registro de configuración	Sigla
3'b000	Datos	**DR**
3'b001	Pilotos	**PR**
3'b010	Portadoras nulas	**NCR**
3'b011	Longitud de prefijo cíclico	**CPR**
3'b100	Trama	**FR**
3'b101	Ranura	**SltR**

Tabla 3.5: Relación de **dataModuleMode** y el número de datos en la palabra **InData**

Código **dataModuleMode**	Número de datos en la palabra **InData**	Número de bits requeridos para un dato
2'b00	16	2
2'b01	16	2
2'b10	8	4
2'b11	5	6

	InData										
				Patrón							
N° de Patrón	**bit 31**	**...**	**bit 23**	**bit 22**	**...**	**bit 12**	**bit 11**	**...**	**bit 1**	**bit 0**
				pilotOffset			*pilotDist*			*enaPilots*	
1	no utilizados			0			6			1	
2	no utilizados			3			6			1	
3	no utilizados			0			0			0	

Figura 3.2: Registro de configuración de ranura: valores de la señal **InData** cuando se requiere ingresar un **patrón para la generación de símbolo OFDM** en el estándar LTE

Tabla 3.6: Módulos funcionales del **`transmisor OFDM reconfigurable`**

Módulos funcionales	Abreviatura
Conformador de trama	**FC**
Decodificador IFFT shift	**IFFTshift**
Procesador FFT/IFFT DEBF de nivel 4	**procesador IFFT**
Módulo de transferencia con prefijo cíclico	**CP AGU**
Lector con control de tasa de transmisión	**TRC**
Buffer de salida	**OB**

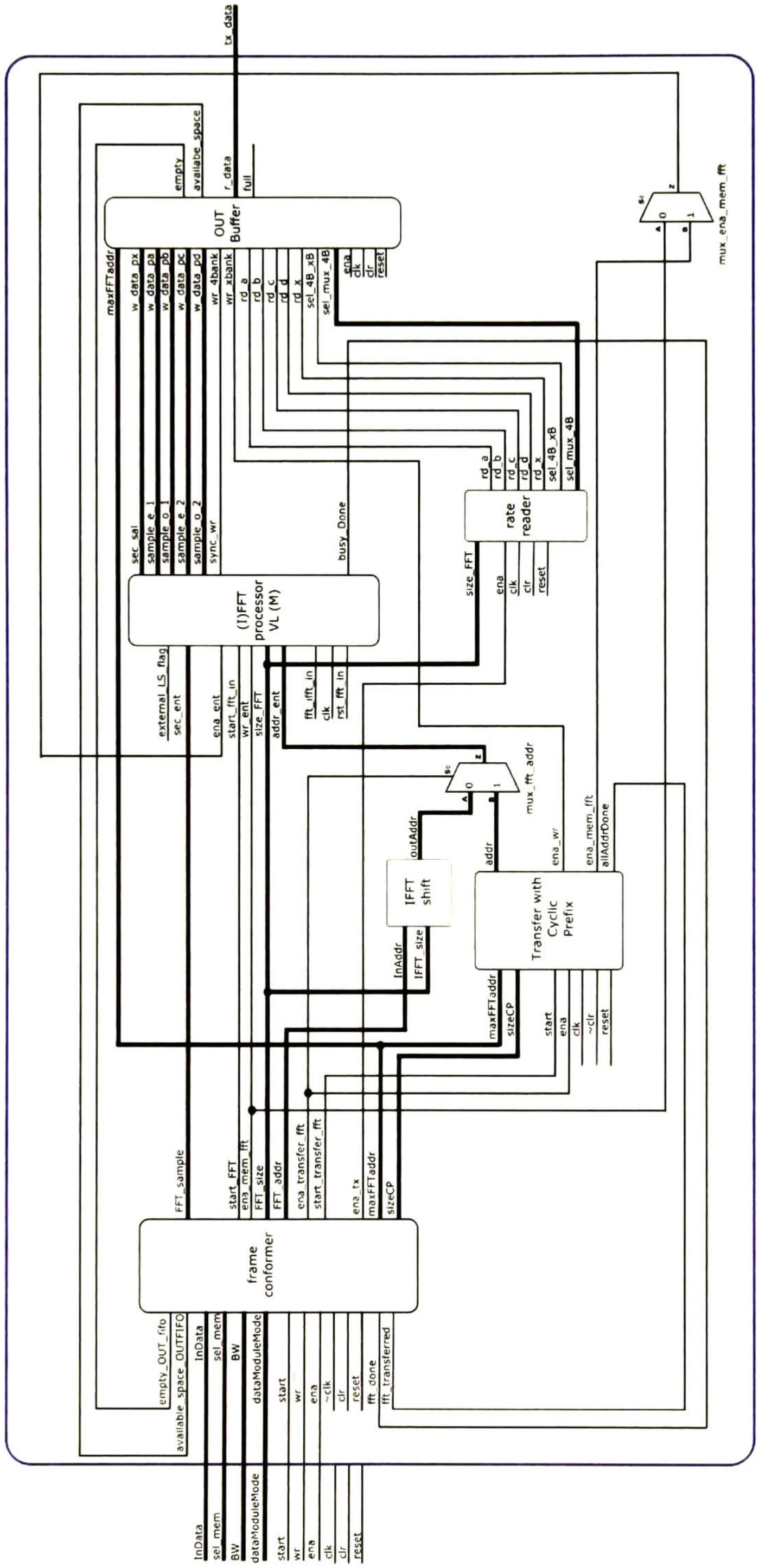


Figura 3.3: `Transmisor OFDM reconfigurable:` Diagrama de módulos funcionales

muestras y direcciones y desactiva la señal **ena_mem_fft**. Posteriormente, activa la señal **start_fft** para iniciar el cálculo de la operación IFFT.

Para el caso de la longitud de IFFT de 128 muestras, la transferencia del resultado al *buffer* de salida **OB** se realiza después de que el **procesador IFFT** termina el cálculo. Sin embargo, para el resto de las longitudes, el **procesador IFFT** entrega el resultado de manera síncrona conforme va realizando el cálculo. Si esto es así, las muestras resultantes son entregadas de cuatro en cuatro, a través de los puertos **sample_e_1**, **sample_o_1**, **sample_e_2** y **sample_o_2**. Simultáneamente, el **procesador IFFT** activa la señal **sync_wr** para que **OB** escriba las 4 muestras. De este modo, cuando el **procesador IFFT** activa la señal **busy_Done**, el cálculo ha concluido y en consecuencia el símbolo OFDM han sido almacenadas en **OB**.

Cuando **FC** recibe la señal **busy_Done**, activa las señales **ena_transfer** y **start_transfer**. Estas señales activan al módulo **CP AGU**, el cual se encarga de coordinar la transferencia de algunas de las muestras resultantes de la IFFT hacia **OB**. Si la longitud de IFFT es de 128 muestras, se realiza la transferencia de las muestras de CP y de las 128 muestras que conforman al símbolo OFDM. Sin embargo, para el resto de las longitudes, sólo se transfieren las muestras de CP, puesto que el resto de las muestras ya se transfirió durante el cálculo de la IFFT. La transferencia es realizada mediante las señales **addr**, **ena_mem_fft** y **ena_wr**.

Las señales **addr** y **ena_mem_fft** permiten leer una muestra del **procesador IFFT**. Por su parte, la señal **ena_wr** habilita la escritura de ésta muestra en **OB**, a través del puerto **w_data_px**. Lo anterior quiere decir que, si la IFFT es de 128 muestras, el símbolo OFDM y su CP se envía al puerto **w_data_px**. Sin embargo, para el resto de longitudes de IFFT, el símbolo OFDM se distribuye en los puertos **w_data_pa**, **w_data_pb**, **w_data_pc** y **w_data_pd**; y el CP se envía al puerto **w_data_px**.

Cuando la transferencia concluye, la señal **allAddrDone** es activada. Si **FC** recibe esta señal, activa al módulo **TRC** mediante la señal **ena_tx**. Dicho módulo controla la lectura de **OB** con el objetivo de alcanzar la tasa de transmisión asociada al ancho de banda. Si la IFFT es de 128 muestras, **TRC** sólo realiza lecturas mediante la señal **rd_x** hasta leer el CP y el símbolo OFDM. Pero, para el resto de las longitudes, **TRC** conmuta la lectura de **OB**. Primero activa la señal **rd_x** hasta leer todas las muestras de CP. Posteriormente, conmuta la activación de las señales **rd_a** y **rd_b** hasta leer la primera mitad del símbolo OFDM. Finalmente, conmuta la activación de las señales **rd_c** y **rd_d** hasta leer la segunda mitad del símbolo OFDM. Las señales **sel_4B_xB** y **sel_mux_4B** son utilizadas para seleccionar el valor de la señal **r_data** en **OB**.

Mientras la transmisión se lleva a cabo, **OB** lleva cuenta del espacio disponible que tiene. Si determina que aún soporta al menos dos símbolos OFDM, activa la señal **available_space**.

Otro evento que ocurre cuando la transferencia de las muestras de CP concluye, es que el **FC** inicia la escritura de nuevas muestras en el **procesador IFFT**. Cuando este proceso de escritura termina, **FC** revisa el estado de la señal **available_space**. Si esta señal está activa, entonces se inicia el cálculo de la IFFT. En caso contrario, no se activa el cálculo y se espera

hasta que la señal **available_space** sea activada. De este modo se evita un overflow en **OB**.

El proceso anterior es repetido para el resto de la trama. Cuando ya no hay más muestras que generar para el **procesador IFFT**, **FC** sólo mantiene activo a **TRC** para transmitir los símbolos OFDM que aún están en **OB**. Cuando éste indica que esta vació mediante la señal **empty**, significa que todos los símbolos OFDM han sido transmitidos. En consecuencia, **FC** desactiva la señal **ena_tx**, poniendo fin a la transmisión de la trama.

3.2. Procesador FFT/IFFT de Longitud Variable

El procesador FFT/IFFT de longitud variable (VL) esta basado en el algoritmo FFT radix-2 de Diezmado en Tiempo (DIT). Sus bases teóricas son explicadas a continuación.

La Transformada Discreta de Fourier de N puntos de la secuencia de entrada $x(n)$ se define como:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad k = 0, 1, ..., N-1 \tag{3.1}$$

donde $W_N = e^{-j\frac{2\pi}{N}}$ son los factores rotatorios, n and k representan los indices de tiempo discreto y frecuencia normalizada. Similarmente, la Transformada Discreta de Fourier Inversa (IDFT) es expresada como:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-nk}, \quad n = 0, 1, ..., N-1 \tag{3.2}$$

Desde el punto de vista de diezmado, hay dos tipos básicos del algoritmo FFT: diezmado en tiempo (DIT) diezmado en frecuencia (DIF). No hay deferencia en la complejidad computacional entre ellos y el número de muestras de la secuencia de entrada tiene que ser una potencia de dos. El algoritmo FFT DIT radix-2 divide la secuencia original en dos secuencia de $N/2$ puntos $f_1(n)$ y $f_2(n)$, que corresponden a las muestras de índices pares e impares de $x(n)$, respectivamente. Así, la DFT de N puntos puede ser calculada como:

$$\begin{aligned} X(k) &= F_1(k) + W_N^k F_2(k), \ k = 0, 1, ..., N/2-1 \\ X(k+N/2) &= F_1(k) - W_N^k F_2(k), \ k = 0, 1, ..., N/2-1 \end{aligned} \tag{3.3}$$

donde $F_1(k)$ y $F_2(k)$ son las DFTs de $N/2$ puntos de $f_1(n)$ y $f_2(n)$, respectivamente. Las subsecuencias $F_1(k)$ y $F_2(k)$ resueltas recursivamente aplicando la ecuación 3.3. De este modo se obtiene la operación elemental conocida como DIT mariposa (butterfly (BF)). Su representación es mostrada en la Figura 3.4.

En el algoritmo FFT DIT radix-2, $\log_2(N)$ etapas de mariposas son necesarias para calcular la FFT de N puntos. Durante cada etapa, $N/2$ mariposas deben ser calculadas. Por

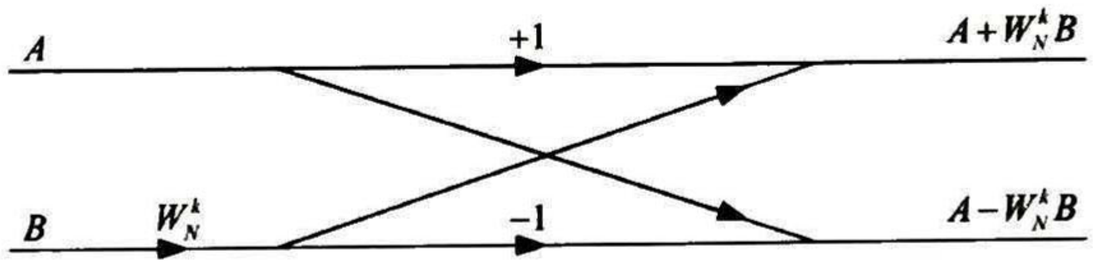


Figura 3.4: Butterfly DIT radix-2

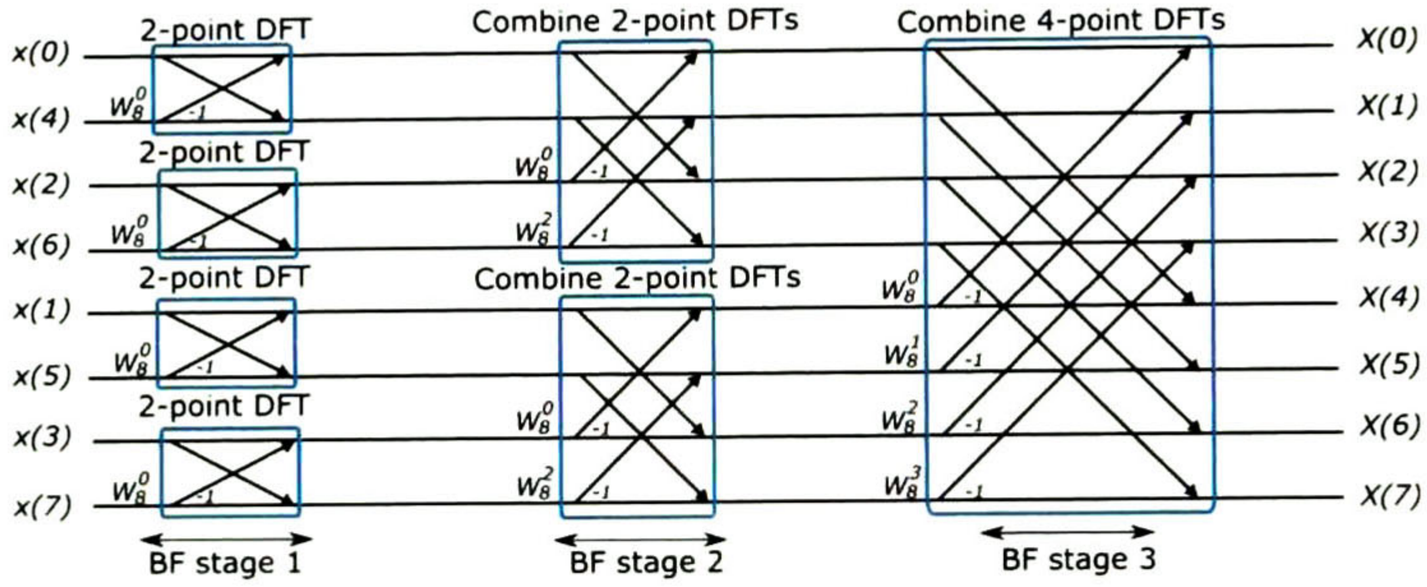


Figura 3.5: Ejemplo de FFT DIT radix-2 de 8 puntos[17].

ejemplo, en el flujograma de la FFT de 8 puntos mostrado en la figura 3.5, hay 3 etapas de mariposas y 4 mariposas son calculadas durante cada etapa.

La arquitectura *multinúcleo* puede interpretarse a partir del ejemplo de la figura 3.5, donde un procesador FFT de nivel 1 puede ser conformado combinando dos FFTs de $N/4$ puntos para obtener una FFT de $N/2$ puntos. El proceso de combinación consiste en aplicar la última etapa de mariposas (LBFS) correspondiente. Si la arquitectura es aplicada nuevamente, de modo que, dos procesadores de $N/2$ puntos de nivel 1 son combinados para obtener una FFT de N puntos, entonces un procesador FFT de nivel 2 es obtenido. Ensamblando Lv niveles, se puede obtener un procesador FFT con $Lv + 1$ longitudes disponibles. La longitud de FFT para el nivel l es determinado como:

$$FFT\_length(l) = \frac{N}{2^{Lv-l}} \tag{3.4}$$

donde N es una potencia de 2 y representa la máxima longitud de FFT, Lv es la cantidad total de niveles y l es el nivel actual, el cual es un entero que está en el rango de 0 a Lv.

Se realizaron dos implementaciones del procesador FFT/IFFT de longitud variable. Ambas están basadas en la arquitectura multinúcleo. Sin embargo, la principal diferencia entre ambas, radica en la cantidad de mariposas externas **EBF** que son utilizadas para ejecutar la última etapa de mariposas **LBFS**. La primera implementación es un procesador FFT/IFFT con mariposa externa única **SEBF**. Por su parte, la segunda implementación es un procesador con doble mariposa externa **DEBF**.

Las muestras son representadas mediante 32 bits, 16 para la parte real y 16 para la parte

imaginaria. La configuración de punto fijo para ambas partes es de un bit para el signo y 15 bits para la parte fraccional.

3.3. Procesador FFT/IFFT SEBF de nivel Lv

3.3.1. Descripción de caja negra

El procesador FFT/IFFT SEBF de nivel Lv **proc_fft_with_SEBF_**Lv de la figura 3.6, es un módulo reconfigurable en términos de longitud variable. Puede calcular la Transformada Rápida de Fourier directa e inversa para las longitudes mostradas en la tabla 3.8. Las señales de entrada y salida de este módulo se muestran en la tabla 3.7. En el apéndice A, se presenta un programa desarrollado para obtener los archivos de diseño de este módulo. Estableciendo la longitud máxima de FFT/IFFT (N) y el número de niveles (Lv), el programa es capaz de construir de manera automatizada al procesador FFT/IFFT SEBF de nivel Lv.

Después de reiniciar al módulo con la señal **rst_fft_in**, se debe corroborar que la señal **busy_done** esté en **1**, indicando que es posible comenzar a ingresar los datos de entrada. El tipo de operación deseada, así como la longitud de la FFT/IFFT se configuran mediante las entradas **ifft_fft_in** y **size_FFT**, respectivamente.

El ingreso de muestras para la FFT/IFFT se realiza activando las señales **wr_ent** y **ena_ent**. Dirección y dato deben ser actualizados en los flancos de bajada del reloj para que durante el flanco de subida, el valor de ambos ya esté definido en lo puertos **sec_ent** e **in_addr**, respectivamente. La direcciones, asociadas a las muestras, van desde 0 hasta la longitud de FFT/IFFT deseada. El procesador considera como muestra $x(0)$ a aquella que es almacenada en la dirección 0; como $x(\frac{N}{2}-1)$, a la almacenada en la dirección $\frac{N}{2}-1$; como $x(-\frac{N}{2})$, a la almacenada en la dirección $\frac{N}{2}$; y como $x(-1)$, a la almacenada en la dirección $N-1$. Al terminar la escritura, las señales **wr_ent** y **ena_ent** deben regresar a su valor habitual de **0**. En estas condiciones, el módulo esta listo para recibir un pulso en alto de la señal **start_fft_in** durante un ciclo de reloj, con lo que la señal **busy_done** cambiará a **0**.

Durante un cierto tiempo, el módulo se mantendrá calculando la operación. Al finalizar,

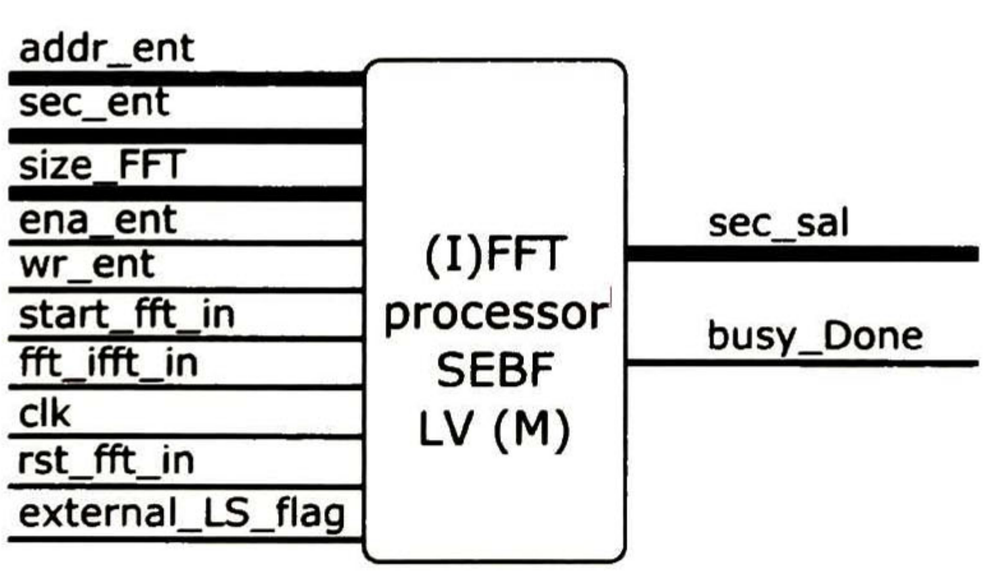


Figura 3.6: Procesador FFT/IFFT SEBF de nivel Lv: Diagrama de entradas y salidas

Tabla 3.7: Entradas y salidas del procesador FFT/IFFT SEBF de nivel Lv.

Nombre	Descripción	bits
	Entradas	
external_LS_flag	Bandera activa en alto para indicar que los cálculos serán mediante Last Stage (LS) Butterfly o Dragonfly.	1
size_FFT	Código para selección de longitud de FFT	3
start_fft_in	Señal de inicio activa en alto para comenzar el cálculo	1
rst_fft_in	Señal de reinicio asíncrono activa en bajo	1
ifft_fft_in	Selector de operación. **0**: FFT; **1**: IFFT	1
clk	Señal de reloj	1
sec_ent	Puerto de datos de entrada 16 MSB: Parte real 16 LSB: Parte imaginaria	32
in_addr	Puerto de direcciones	11
wr_ent	Señal de habilitación de escritura, activa en alto	1
ena_ent	Señal de habilitación de memoria, activa en alto	1
	Salidas	
busy_done	**1**: módulo listo **0** módulo ocupado	1
sec_sal	Puerto de datos de salida 16 MSB: Parte real 16 LSB: Parte imaginaria	32

la señal **busy_done** cambiará a **1**, indicando que es posible leer los datos de salida o comenzar a escribir nuevos datos sin leer el resultado previo. Los datos de salida se obtienen proporcionando una dirección en el puerto **in_addr** y activando **ena_ent** y desactivando **wr_ent**. Esto habilita el acceso a la memoria bajo la operación de lectura. Las direcciones de memoria van desde 0 hasta la longitud deseada. Las direcciones deben actualizarse en los flancos de bajada del reloj para garantizar que el valor adecuado de las muestras ocurra en los flancos de subida.

3.3.2. Descripción de caja blanca

El procesador FFT/IFFT SEBF de nivel Lv de la figura 3.7 esta basado en la arquitectura multinúcleo de la sección 3.2. En esta implementación se estableció una longitud máxima (N) de 2048 muestras y 4 niveles (Lv) de profundidad. Aplicando la ecuación (3.4), se determinan las 5 longitudes de FFT/IFFT. Estas longitudes son: N, $N/2$, $N/4$, $N/8$ y $N/16$.

En la tabla 3.9 se presentan los módulos funcionales que componen al procesador

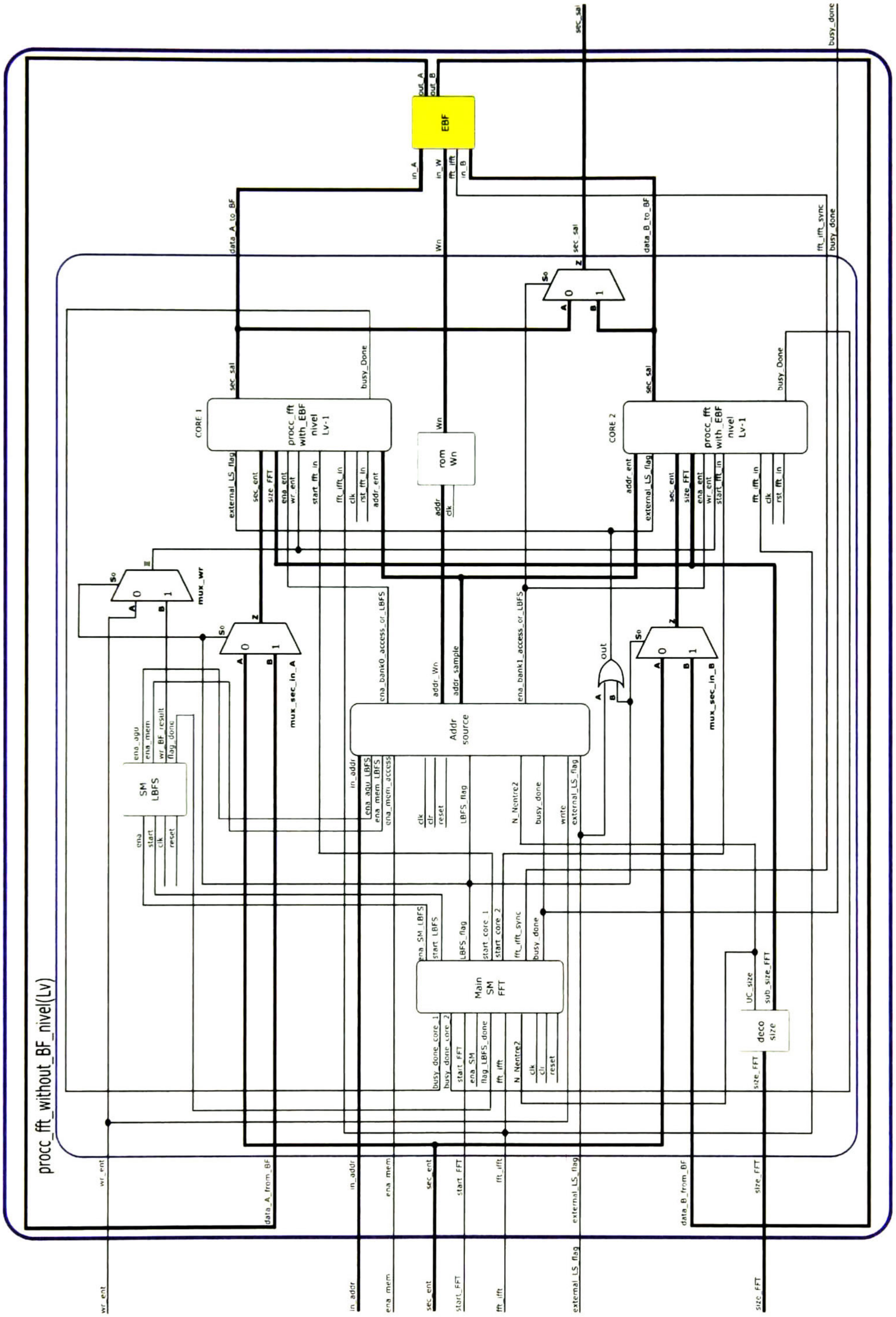


Figura 3.7: Procesador FFT/IFFT SEBF de nivel Lv: Diagrama de módulos funcionales

Tabla 3.8: Valores de la señal de entrada **FFT_size**

Valor **FFT_size**	Longitud de FFT
3'b000	2048
3'b001	1024
3'b010	512
3'b011	256
3'b100	128

Tabla 3.9: procesador FFT/IFFT SEBF de nivel Lv: Lista de módulos funcionales

Módulos primarios	Módulos internos	Descripción
proc_fft_without_BF_Lv		Procesador FFT/IFFT sin SEBF de nivel Lv
	Main_SM_FFT	Unidad de control
	SM_LBFS	Control de la última etapa de mariposas
	deco_size_Lv	Decodificador de longitud de FFT/IFFT nivel Lv
	Addr_source	Fuente de direcciones
	ROM_Wn	Memoria de factores rotatorios
	proc_fft_with_SEBF_$Lv-1$	Procesador FFT/IFFT SEBF nivel $Lv-1$
EBF		Mariposa Externa

FFT/IFFT SEBF de nivel Lv. Se puede apreciar que está conformado por dos módulos principales: un procesador FFT/IFFT sin SEBF de nivel Lv y una mariposa externa. El procesador FFT/IFFT sin SEBF de nivel Lv contiene dos procesadores FFT/IFFT SEBF de nivel $Lv-1$, los cuales son llamados **core_1** y **core_2**. Estos procesadores contienen un banco de memoria cada uno. Cuando **core_1** y **core_2** se combinan mediante la mariposa externa **EBF**, logran conformar la FFT/IFFT de nivel Lv. A su vez, los procesadores FFT/IFFT SEBF de nivel $Lv-1$ replican esta arquitectura recursivamente hasta llegar al procesador FFT/IFFT de nivel 0. Las tablas 3.10 y 3.11 muestran las señales de entrada y salida de la mariposa externa y del procesador FFT/IFFT sin SEBF de nivel Lv.

Después de utilizar la señal de **rst_fft_in**, se espera que las muestras sean ingresadas al módulo. Sin embargo, se debe establecer el valor de la señal **size_FFT** antes de comenzar el ingreso de las muestras. La razón es porque esta señal define la manera en la que el módulo **addr_source** genera las direcciones para los procesadores FFT/IFFT SEBF de nivel $Lv-1$. El módulo **deco_size_**Lv recibe la señal **size_FFT** y genera un nuevo tamaño de FFT llamado **sub_size_FFT** para estos procesadores. Además, genera una bandera llamada **UC_size**. Cuando esta señal está en **1** indica que únicamente se requiere de **core_1**. Sin embargo, en **0** indica que se requiere de la combinación de **core_1** y **core_2**.

Cuando se desea ingresar una muestra, se debe establecer la dirección **in_addr** y activar en alto las señales **ena_mem** y **wr_ent**. Estas señales son recibidas por el módulo **addr_source**, el cual hace uso de la señal **UC_size** para generar la subdirección

Tabla 3.10: Entradas y salidas de la mariposa externa **EBF**.

Nombre	Descripción	bits
	Entradas	
in_A	Muestra A	32
in_B	Muestra B	32
in_W	Factor rotatorio	32
fft_ifft	Señal de selección de operación, **0**, FFT; **1**, IFFT	1
	Salidas	
out_A	Resultado A	32
out_B	Resultado B	32

addr_sample y habilitar el accesos a uno de los dos bancos mediante las señales **ena_bank0** y **ena_bank1**. El control **Main_SM_FFT** mantiene la señal **LBFS_flag** en **0** durante el ingreso de las muestras para darle al usuario el control de la escritura. La señal **LBFS_flag** es utilizada como selector para los multiplexores **mux_wr**, **mux_sec_in_A** y **mux_sec_in_B**. En consecuencia, la señal **sec_ent** del nivel Lv se conecta con sus homólogas del nivel $Lv - 1$. Además, la señal de escritura **wr_ent**, que proviene desde fuera del módulo, es la que se conecta con la entrada **wr_ent** de los procesadores de nivel $Lv - 1$.

Después de ingresar todas las muestras, se espera que se active la señal **start**. Cuando **Main_SM_FFT** recibe esta señal, inicia el procesamiento de uno o ambos procesadores mediante las señales **start_core_1** y **start_core_2**. El número de procesadores que deben trabajar esta determinado por la señal **UC_size**, la cual es recibida por **Main_SM_FFT** a través de la entrada **N_Nentre2**.

Cuando los procesadores FFT/IFFT SEBF de nivel $Lv - 1$ reciben la señal **start**, declaran su salida **busy_Done** en **0** para indicar que se encuentran ocupados realizando el cálculo. **Main_SM_FFT** recibe las señales **busy_Done** a través de las entradas **busy_done_core_1** y **busy_done_core_2**. Si **N_Nentre2** se encuentra en **1**, la señal **busy_done_core_2** se vuelve irrelevante y se pone especial atención en la señal **busy_done_core_1**. Cuando **busy_done_core_1** regresa al valor de **1**, la señal **busy_Done** del nivel Lv se activa en **1** para notificar que la FFT/IFFT solicitada ha sido calculada.

Por el contrario, si **N_Nentre2** esta en **0**, se espera a que ambas señales, **busy_done_core_1** y **busy_done_core_2**, regresen a su valor en **1**. Posteriormente, **Main_SM_FFT** activa las señales **ena_SM_LBFS**, **start_LBFS** y **LBFS_flag** para iniciar la última etapa de mariposas. Como consecuencia, el selector de los multiplexores **mux_wr**, **mux_sec_in_A** y **mux_sec_in_B** cambia. Por lo tanto, el control de la escritura lo tendrá el módulo **SM_LBFS** y las entradas **sec_ent** de **core_1** y **core_2** se conectarán a las entradas **data_A_from_BF** y **data_B_from_BF** respectivamente. **Main_SM_FFT** usa la señal **LBFS_flag** para entregar el control de **addr_source** al módulo **SM_LBFS**, el cual dicta los accesos a memoria a través de la señal **ena_mem_LBFS** y activa la generación de direcciones para la última etapa de ma-

Tabla 3.11: Entradas y salidas del procesador FFT/IFFT sin SEBF de nivel Lv

Nombre	Descripción	bits
	Entradas	
data_A_from_BF	Puerto de entrada del resultado A de la mariposa	32
data_B_from_BF	Puerto de entrada del resultado B de la mariposa	32
external_LS_flag	Bandera activa en alto para indicar que los cálculos serán mediante LBFS externa	1
size_FFT	Código para selección de longitud de FFT	3
start_fft_in	Señal de inicio activa en alto para comenzar el cálculo	1
rst_fft_in	Señal de reinicio asíncrono activa en bajo	1
ifft_fft_in	Selector de operación. **0**: FFT; **1**: IFFT	1
clk	Señal de reloj	1
sec_ent	Puerto de datos de entrada Parte real: 16 MSB Parte imaginaria: 16 LSB	32
in_addr	Puerto de direcciones	11
wr_ent	Señal de habilitación de escritura, activa en alto	1
ena_ent	Señal de habilitación de memoria, activa en alto	1
	Salidas	
data_A_to_BF	Muestra A para la mariposa	32
data_B_to_BF	Muestra B para la mariposa	32
Wn	Factor rotatorio	32
busy_done	**1**: módulo listo **0** módulo ocupado	1
sec_sal	Puerto de datos de salida Parte real: 16 MSB Parte imaginaria: 16 LSB	32

riposas mediante la señal **ena_agu_LBFS**. Con estas direcciones se obtienen dos muestras de **core_1** y **core_2** y un factor rotatorio de la memoria **ROM_Wn**. Esta terna de operandos es recibida por la mariposa externa **EBF** para proporcionar las señales **out_A** y **out_B**.

El módulo **procc_fft_without_SEBF_**Lv recibe los datos de salida de la mariposa a través de los puertos **data_A_from_BF** y **data_B_from_BF**. El objetivo es almacenarlos en la misma dirección del procesador FFT/IFFT SEBF dónde surgieron. Cuando la última etapa de mariposas termina, **SM_LBFS** activa la señal **flag_done** para que **Main_SM_FFT** regrese a su estado inicial y establezca la señal **busy_done** del nivel Lv en **1**. Además, las señales **ena_SM_LBFS**, **start_LBFS** y **LBFS_flag** regresan a **0**, de modo que el control de acceso a los bancos regresa al usuario.

La operación de lectura se realiza estableciendo una dirección en el puerto **in_addr** y manteniendo la señal **ena_mem** en **1** y **wr_ent** en **0**. Estas señales son recibidas por **addr_source** para generar la dirección **addr_sample** y las señales para habilitar el acceso a uno de los dos bancos **ena_bank0** y **ena_bank1**.

En las siguientes secciones se hará una descripción de los módulos de la tabla 3.9. Sin embargo, la mariposa y el procesador FFT utilizado para conformar el nivel cero de esta arquitectura ya fueron reportados en [1].

3.3.2.1. Decodificador de longitud FFT/IFFT de nivel Lv

En la tabla 3.12 se muestran las entradas y salidas del módulo **deco_size_**Lv. Este módulo utiliza el puerto **size_FFT** para recibir la longitud de FFT/IFFT solicitada en el nivel Lv. Las señales **sub_size_FFT** y **UC_size** son generadas con el objetivo de satisfacer dicha longitud. La señal **sub_size_FFT** define la longitud para los dos procesadores FFT/IFFT SEBF de nivel $Lv - 1$. Por otro lado, **UC_size** le indica a la unidad de control, **Main_SM_FFT**, el número de procesadores FFT/IFFT SEBF de nivel $Lv - 1$ que deben ser empleados. Si en el nivel Lv se requiere una FFT/IFFT de longitud N o $N/2$, la señal **sub_size_FFT** solicitará una longitud de $N/2$ en ambos casos. Sin embargo, para el primero, la señal **UC_size** valdrá **0**; mientras que para el segundo, valdrá **1**.

3.3.2.2. Control de la última etapa de mariposas

El módulo de control de la última etapa de mariposas **SM_LBFS** es mostrado en la figura 3.8. Sus entradas y salidas son presentadas en la tabla 3.13.

Tras recibir la señal **start**, este módulo activa la señal **ena_agu**. Esto produce que las direcciones, necesarias para la última etapa de mariposas, comiencen a ser generadas por el módulo **addr_source**. Estas direcciones son utilizadas para leer y escribir muestras en los procesadores FFT/IFFT SEBF.

Durante cada dirección, se realiza una operación de lectura, seguida de una operación de escritura. Con la lectura se obtienen las muestras necesarias para la mariposa y con la

Tabla 3.12: Entradas y salidas del módulo **deco_size_**Lv.

Nombre	Descripción	bits
	Entradas	
size_FFT	Longitud de FFT/IFFT en el nivel Lv.	$ceil[(log_2(Lv+1)]$
	Salidas	
sub_size_FFT	Longitud de FFT/IFFT en el nivel $Lv-1$	$ceil[(log_2(Lv)]$
UC_size	Número de procesadores FFT/IFFT de nivel $Lv-1$ que deben ser activados. **1**: Un procesador **0**: Dos procesadores	1

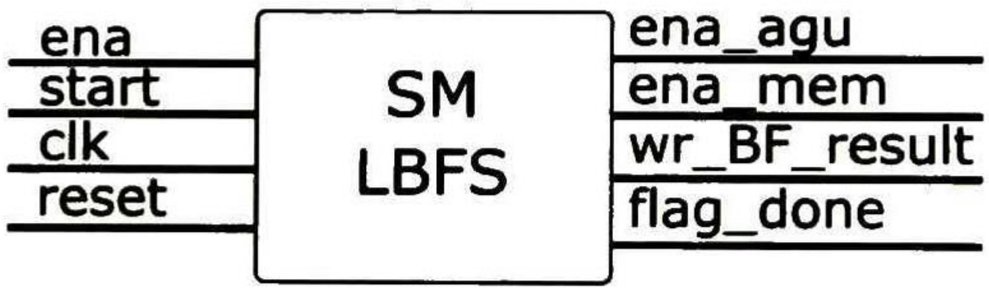


Figura 3.8: Control de la última etapa de mariposas: Diagrama de entradas y salidas

escritura se almacena el resultado de la mariposa. Este módulo es el encargado de coordinar las operaciones de lectura y escritura mediante las señales **ena_mem** y **wr_BF_result**.

Para ejecutar una operación de lectura en los procesadores, la señal **ena_mem** debe mantenerse en **1** y **wr_BF_result** en **0**, mientras que para la operación de escritura, ambas señales deben mantenerse en **1**.

Además, éste módulo lleva cuenta de las mariposas de la última etapa. Cuando la última mariposa ha sido escrita, **SM_LBFS** activa la señal **flag_done**.

3.3.2.3. Fuente de direcciones

El módulo Fuente de direcciones es mostrado en la figura 3.9. Sus señales de entrada y salida están mostradas en la tabla 3.14. Este módulo está encargado de proporcionar las direcciones **addr_sample** y las señales de habilitación **ena_bank0** y **ena_bank1** que permiten el acceso a la memoria de los procesadores FFT/IFFT SEBF de nivel $Lv-1$. Este acceso puede generarse bajo diferentes condiciones.

Si la señal **flag_LBFS** esta activa en alto, **addr_source** realizará multiplexaje para que la señal de entrada **ena_mem_LBFS** salga por los puertos **ena_bank0** y **ena_bank1**. Por su parte, las direcciones que proporciona el módulo interno **AGU_LBFS**, serán seleccionada para estar presentes en los puertos **addr_sample** y **addr_Wn**. Para generar una FFT/IFFT de longitud N, estas direcciones van desde 0 hasta $\frac{N}{2}-1$, incrementándose en 1 después de cada mariposa ejecutada. Cuando **flag_LBFS** no esta activa, las habilitaciones estarán

Tabla 3.13: Entradas y salidas del módulo de control de la última etapa de mariposas

Nombre	Descripción	bits
	Entradas	
ena	Señal de habilitación activa en alto	1
start	Señal de inicio activa en alto	1
clk	Señal de reloj	1
reset	Señal de reinicio asíncrono activa en bajo	1
	Salidas	
ena_AGU	Señal activa en alto para la habilitación de la unidad generadora de direcciones de la última etapa de mariposas	1
ena_mem	Habilitación de acceso a memoria activa en alto	1
wr_BF_result	Habilitación de escritura activa en alto	1
flag_done	Bandera activa en alto para indicar que se ha completado la última etapa de mariposas	1

determinadas por la señal **external_LS_flag**.

Si **external_LS_flag** está en alto, el acceso será para el procesamiento de mariposas externas con el propósito de formar una FFT de nivel $Lv+1$. En estas condiciones, primero se accederá a todo el primer banco y hasta terminar con él, se comenzará el segundo. Además, las direcciones para los módulos de nivel $Lv - 1$ serán generadas a partir de **in_addr** mediante el módulo interno **rd_addr_deco**. Este módulo forma la dirección **addr_sample** tomando todos los bits de la dirección **in_addr**, excepto el MSB. Cuando **external_LS_flag** está en bajo, el acceso dependerá de las señales **write** y **ena_mem_access**.

Si **ena_mem_access** esta en **1** y **write** está en **0**, se realiza una lectura en la dirección y banco que el módulo interno **rd_addr_deco** proporciona, pero, si ambas están en **1**, se realiza una escritura en la dirección y banco que el módulo **wr_addr_deco** proporciona.

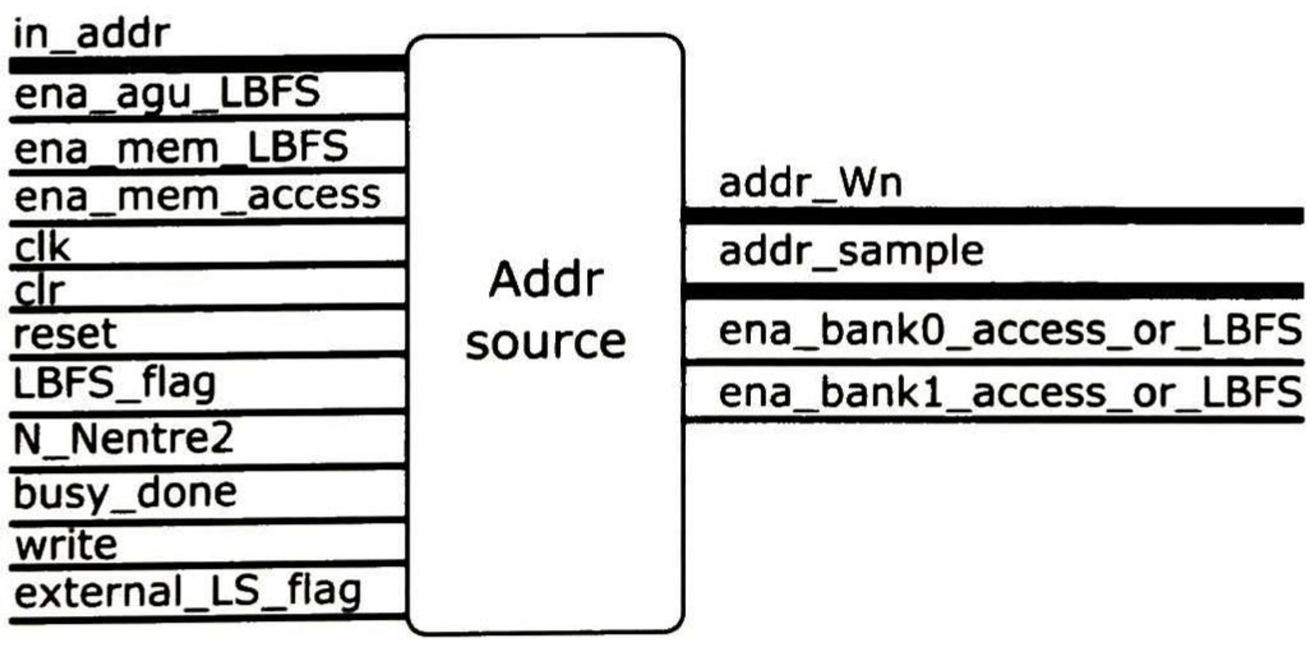


Figura 3.9: Fuente de direcciones: Diagrama de entradas y salidas

Tabla 3.14: Entradas y salidas del módulo Fuente de direcciones

Nombre	Descripción	bits
	Entradas	
in_addr	Señal de direcciones	$ceil(log_2(N))$
ena_agu_LBFS	Señal de habilitación de la unidad generadora de direcciones para la última etapa de mariposas. Activa en alto	1
ena_mem_LBFS	Señal de habilitación de memoria por acceso de la última etapa de mariposas	1
ena_mem_access	Señal de habilitación de memoria por acceso de usuario	1
clk	Señal de reloj	1
clr	Señal de reinicio síncrono activa en alto	1
reset	Señal de reinicio asíncrono activa en bajo	1
LBFS_flag	Bandera activa en alto para indicar que la última etapa de mariposas esta en proceso	1
N_Nentre2	Número de procesadores FFT/IFFT SEBF de nivel $Lv - 1$ activos.	1
	1: Un procesador	1
	0: Dos procesadores	1
write	Señal de habilitación de escritura, controlada por el usuario y activa en alto	1
external_LS_flag	Señal activa en alto para indicar que el procesador de nivel Lv será utilizado como base de una FFT de nivel superior $(Lv + 1)$	1
	Salidas	
ena_bank_0	Señal de habilitación del primer banco de memoria. Es activa en alto debido al acceso de usuario o por acceso de la última etapa de mariposas	1
ena_bank_1	Señal de habilitación del segundo banco de memoria. Es activa en alto debido al acceso de usuario o por acceso de la última etapa de mariposas	1
addr_Wn	Señal de direcciones para la memoria **ROM_Wn**	$ceil(log_2(N)) - 1$
addr_sample	Señal de direcciones para los procesadores FFT/IFFT SEBF	$ceil(log_2(N)) - 1$

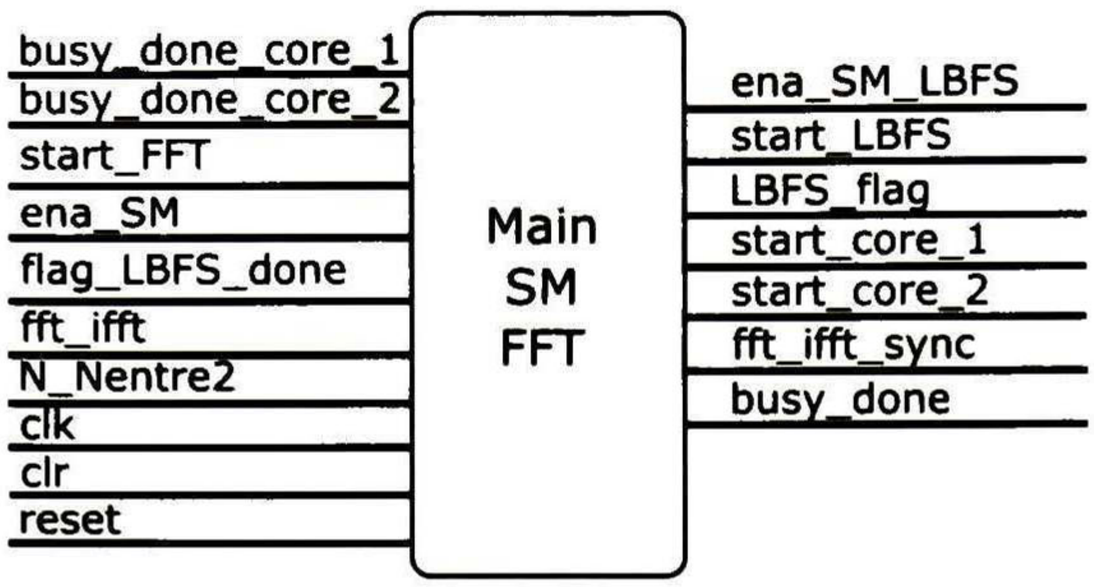


Figura 3.10: Unidad de control FFT: Diagrama de entradas y salidas

3.3.2.4. Unidad de control FFT

La unidad de control FFT **Main_SM_FFT** de la figura 3.10, genera las señales de habilitación e inicio de los módulos que conforman al procesador FFT/IFFT sin SEBF de nivel Lv. En la tabla 3.15 se muestran las entradas y salidas de este modulo. La figura 3.11 muestra su diagrama de estados, cuyas combinaciones de entradas (x) y salidas (y) son presentadas en la tabla 3.15.

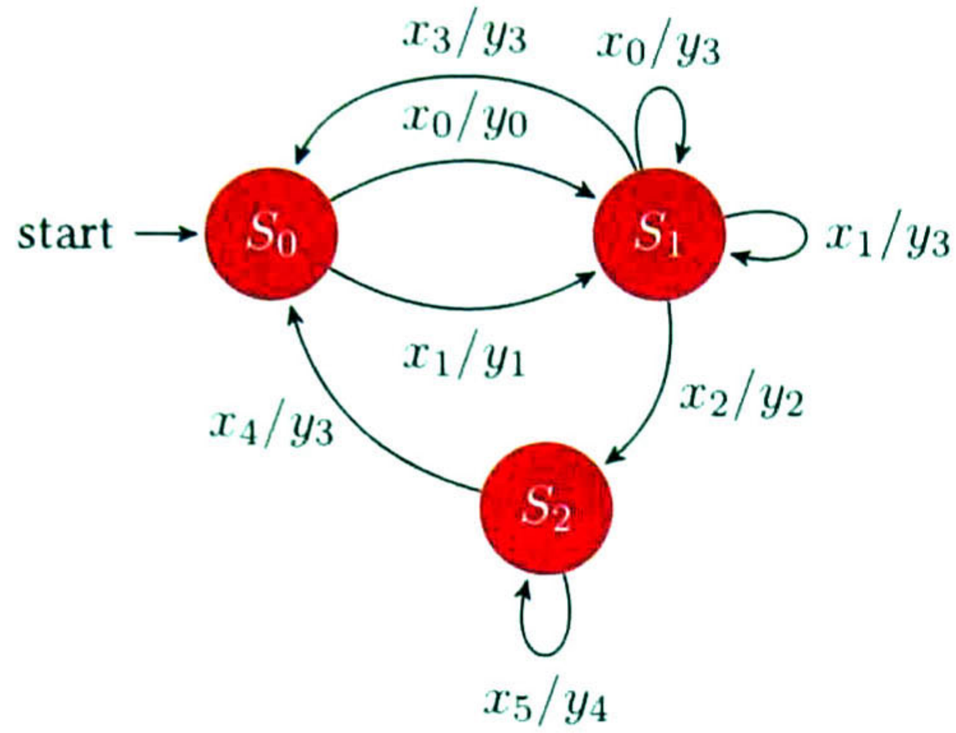
Al permanecer en el estado inicial (IDLE) y recibir la señal de inicio **start**, la unidad de control pasa al estado de ejecución de proceso (EXE PROCESS), declara la señal **busy_done** del nivel Lv como **0** y revisa el valor de **N_Nentre2**.

Si **N_Nentre2** esta en alto, activa unicamente la señal **start_core_1**. Con esto, **core_1** comenzará la ejecución de la FFT/IFFT y la unidad de control permanecerá en este estado hasta que la señal **busy_done_core_1** regrese al valor de **1**. Cuando esto sucede, la unidad de control activa la señal **busy_done** del nivel Lv y regresa al estado IDLE. Esto marca el fin del procesamiento de la FFT/IFFT solicitada.

Si **N_Nentre2** esta en bajo, la unidad de control pasa al estado (EXE PROCESS) y activa las señales **start_core_1** y **start_core_2**. Durante un tiempo permanecerá en este estado hasta recibir las señales **busy_done_core_1** y **busy_done_core_2**. Cuando esto sucede, las FFT/IFFT´s de mitad de longitud están listas y es momento de combinarlas mediante la última etapa de mariposas. Por ello, la unidad de control pasa el estado LBFS y activa las señales **start_LBFS**, **ena_SM_LBFS** y **LBFS_flag**. Cuando la unidad de control recibe la señal **flag_LBFS_done**, activa la señal **busy_done** del nivel Lv y regresa a su estado inicial.

Tabla 3.15: Entradas y salidas de la unidad de control FFT **Main_SM_FFT**.

Nombre	Descripción	bits
	Entradas	
busy_done_core_1	Señal de estado de **core_1** **1**: listo **0**: ocupado	1
busy_done_core_1	Señal de estado de **core_2** **1**: listo **0**: ocupado	1
start_FFT	Señal de inicio activa en alto	1
ena_SM	Señal de habilitación activa en alto	1
flag_LBFS_done	Señal activa en alto para indicar que la última etapa de mariposas ha concluido	1
fft_ifft	Señal de selección de operación **0**: FFT **1**: IFFT	1
clk	Señal de reloj	1
clr	Señal de reinicio síncrono	1
reset	Señal de reinicio síncrono	1
	Salidas	
ena_SM_LBFS	Señal de habilitación para el módulo **SM_LBFS**	1
start_LBFS	Señal de inicio para el módulo **SM_LBFS**	1
LBFS_flag	Bandera activa en alto para indicar que la última etapa de mariposas esta en proceso	1
start_core_1	Señal de inicio para **core_1**	1
start_core_2	Señal de inicio para **core_2**	1
busy_done	Señal de estado del procesador FFT/IFFT SEBF de nivel Lv **1**: listo **0**: ocupado	1



(a)

S_n	Estado
S_0	IDLE
S_1	EXE PROCESS
S_2	LBFS

(b)

Figura 3.11: Diagrama de estados de la unidad de control FFT **Main_SM_FFT**

Tabla 3.16: Entradas y salidas asociadas al diagrama de estados de la figura 3.11

Entrada	**N_Nentre2** **N_Nentre2**	**busy_done** **core_1**	**busy_done** **core_2**	**LBFS_done**	
x_0	1	0	0	0	
x_1	0	0	0	0	
x_2	0	1	1	0	
x_3	1	1	0	0	
x_4	0	1	1	1	
x_5	0	1	1	0	
Salida	**start** **core_1**	**start** **core_2**	**ena_LBFS**	**start_LBFS**	**LBFS_flag**
y_0	1	0	0	0	0
y_1	1	1	0	0	0
y_2	0	0	1	1	1
y_3	0	0	0	0	0
y_4	0	0	1	0	1

3.4. Procesador FFT/IFFT DEBF de nivel Lv

3.4.1. Descripción de caja negra

El procesador FFT/IFFT DEBF de nivel Lv es el módulo interno de la figura 3.12(b). Sin embargo, debido a que en el nivel Lv no se utilizan los puertos **addr_ent_o**, **sec_ent_o** y **ena_ent_o**, el procesador fue encapsulado en el **procesador FFT/IFFT** Lv. Este módulo es mostrado en la figura 3.12(a). El procesador FFT/IFFT nivel Lv tiene los mismos puertos que el procesador FFT/IFFT SEBF. Por lo tanto, sus entradas y salidas se presentan en dos tablas complementarias, 3.7 y 3.17.

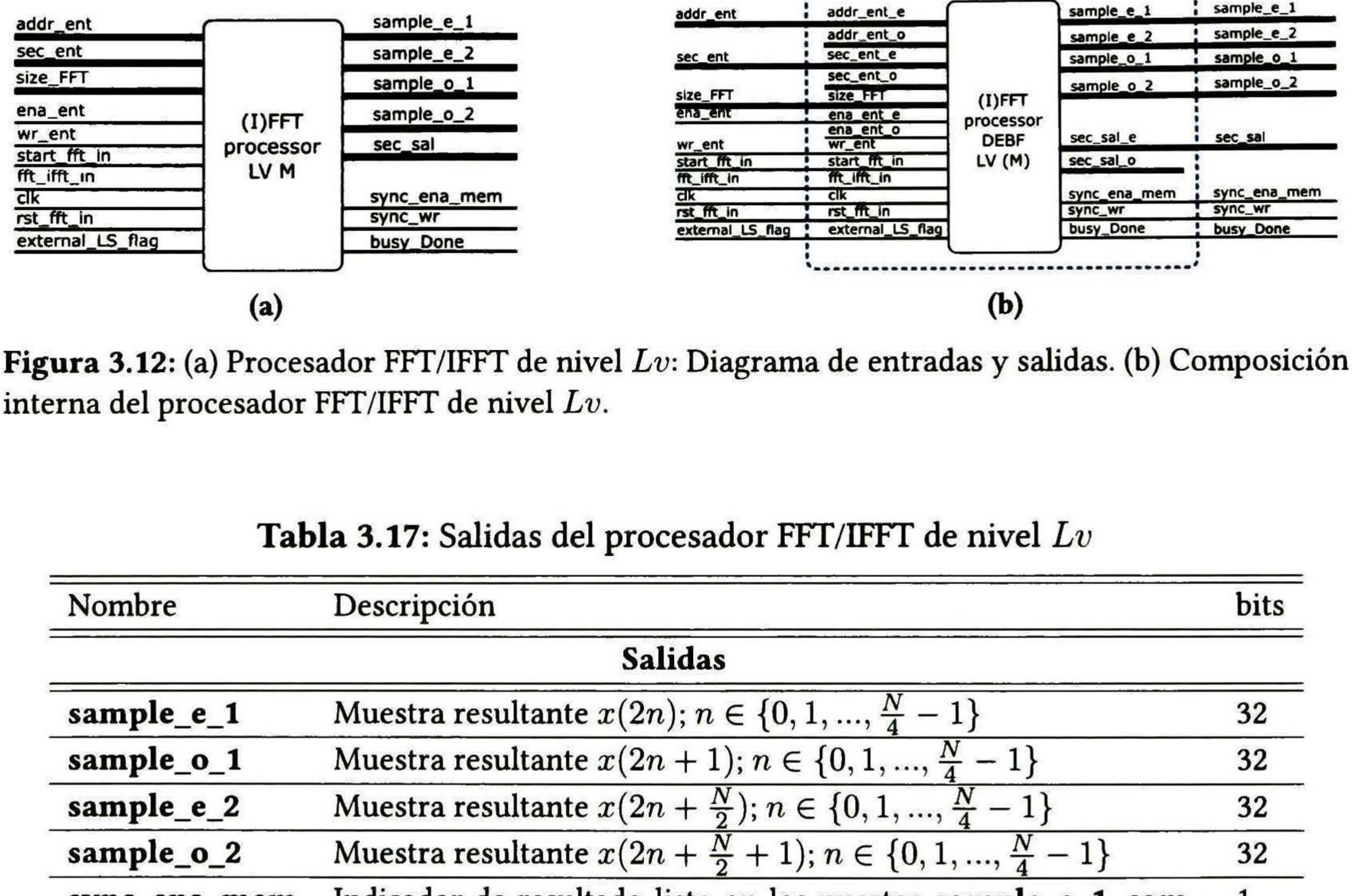


Figura 3.12: (a) Procesador FFT/IFFT de nivel Lv: Diagrama de entradas y salidas. (b) Composición interna del procesador FFT/IFFT de nivel Lv.

Tabla 3.17: Salidas del procesador FFT/IFFT de nivel Lv

Nombre	Descripción	bits
	Salidas	
sample_e_1	Muestra resultante $x(2n); n \in \{0, 1, ..., \frac{N}{4} - 1\}$	32
sample_o_1	Muestra resultante $x(2n+1); n \in \{0, 1, ..., \frac{N}{4} - 1\}$	32
sample_e_2	Muestra resultante $x(2n+\frac{N}{2}); n \in \{0, 1, ..., \frac{N}{4} - 1\}$	32
sample_o_2	Muestra resultante $x(2n+\frac{N}{2}+1); n \in \{0, 1, ..., \frac{N}{4} - 1\}$	32
sync_ena_mem	Indicador de resultado listo en los puertos **sample_e_1**, **sample_o_1**, **sample_e_2** y **sample_o_2**. Activo en alto	1
sync_wr	Indicador de resultado actualizado en los puertos **sample_e_1**, **sample_o_1**, **sample_e_2** y **sample_o_2**. Activo en alto	1

En realidad, el procesador FFT/IFFT DEBF es una extensión del SEBF. En este sentido, los puertos **addr_ent**, **sec_ent** y **ena_ent** del procesador SEBF, son los puertos **addr_ent_e**, **sec_ent_e** y **ena_ent_e** en el procesador DEBF. El hecho de que en el procesador DEBF

se modificara el nombre de estos puertos, se debió a que se agregó un duplicado de cada puerto. Por lo tanto, para distinguir cada par, los puertos **addr_ent**, **sec_ent** y **ena_ent** se renombraron como **addr_ent_e**, **sec_ent_e** y **ena_ent_e** y en conjunto, son conocidos como puertos **e**. Por otro lado, los puertos **addr_ent_o**, **sec_ent_o** y **ena_ent_o** son los puertos que fueron agregados y son conocidos como puertos **o**. Por este motivo no hay necesidad de agregar una tabla para describir los puertos **e** y **o**, pues su funcionamiento es idéntico al mostrado en la tabla3.7.

La manera de ingresar las muestras de entrada en el procesador FFT/IFFT de nivel Lv es idéntico al del procesador SEBF. Sin embargo, el procesador FFT/IFFT DEBF es capaz de escribir o leer dos muestras simultáneamente a través de los puertos **e** y **o**. Aunque éstos puertos pueden utilizarse para ingresar muestras en cualquier dirección; se decidió utilizar a los **e** para las muestra pares y a los **o** para las muestras impares. Sin embargo, en el nivel Lv, los puertos **e** son los únicos utilizados para ingresar todas las muestras, sin hacer distinción entre muestras pares o impares. Esto puede apreciarse en la figura 3.12(b).

La descripción funcional de los procesadores FFT/IFFT de nivel Lv y DEBF es idéntica a la descripción del SEBF presentada en la sección 3.3.1. Sin embargo los procesadores FFT/IFFT de nivel Lv y DEBF cuentan con una característica adicional que permite entregar las muestras resultantes durante el procesamiento. Después de un tiempo de haber iniciado el cálculo mediante la señal **start**, el procesador DEBF activará la señal **sync_ena_mem**. Cuando esto sucede, cuatro muestras resultantes son entregadas en los puertos **sample_e_1**, **sample_o_1**, **sample_e_2** y **sample_o_2**. En cada cuarteto de muestras la señal **sync_wr** conmuta entre **0** y **1**. Por lo tanto, cuando **sync_wr** vale **1**, indica que un nuevo cuarteto de muestras se ha definido. Este mecanismo para obtener las muestras resultantes aplica para todas las longitudes de FFT/IFFT, excepto para la menor. Para esta longitud, el resultado debe ser obtenido mediante el mecanismo descrito en 3.3.1.

Cuando todas las muestras resultantes han sido entregadas, **sync_ena_mem** y **sync_wr** regresan a **0**. Además la señal **busy_Done** se activa para indicar que el cálculo ha finalizado.

3.4.2. Descripción de caja blanca

Para implementar el procesdor FFT/IFFT DEBF de nivel Lv, primero fue necesario implementar el nivel 1. Esto se debe a que el nivel 1 utiliza dos procesadores de nivel 0, los cuales son de longitud estática y por lo tanto, no ejecutan una LBFS interna.

La estructura del procesador DEBF de nivel 1 y nivel Lv, es igual a la estructura del procesador SEBF, de hecho, la descripción funcional es idéntica a la descrita en la sección 3.3.2. Por ello, en esta sección no se explica el funcionamiento completo del procesador FFT/IFFT DEBF, sino que sólo se aborda la última etapa de mariposas, pues esto es en lo que difieren los procesadore SEBF y DEBF.

En la figura 3.13 se puede observar que del procesador sin DEBF de nivel 1 se toman 4 muestras para ejecutar la LBFS mediante dos mariposas. En la figura 3.14 se puede

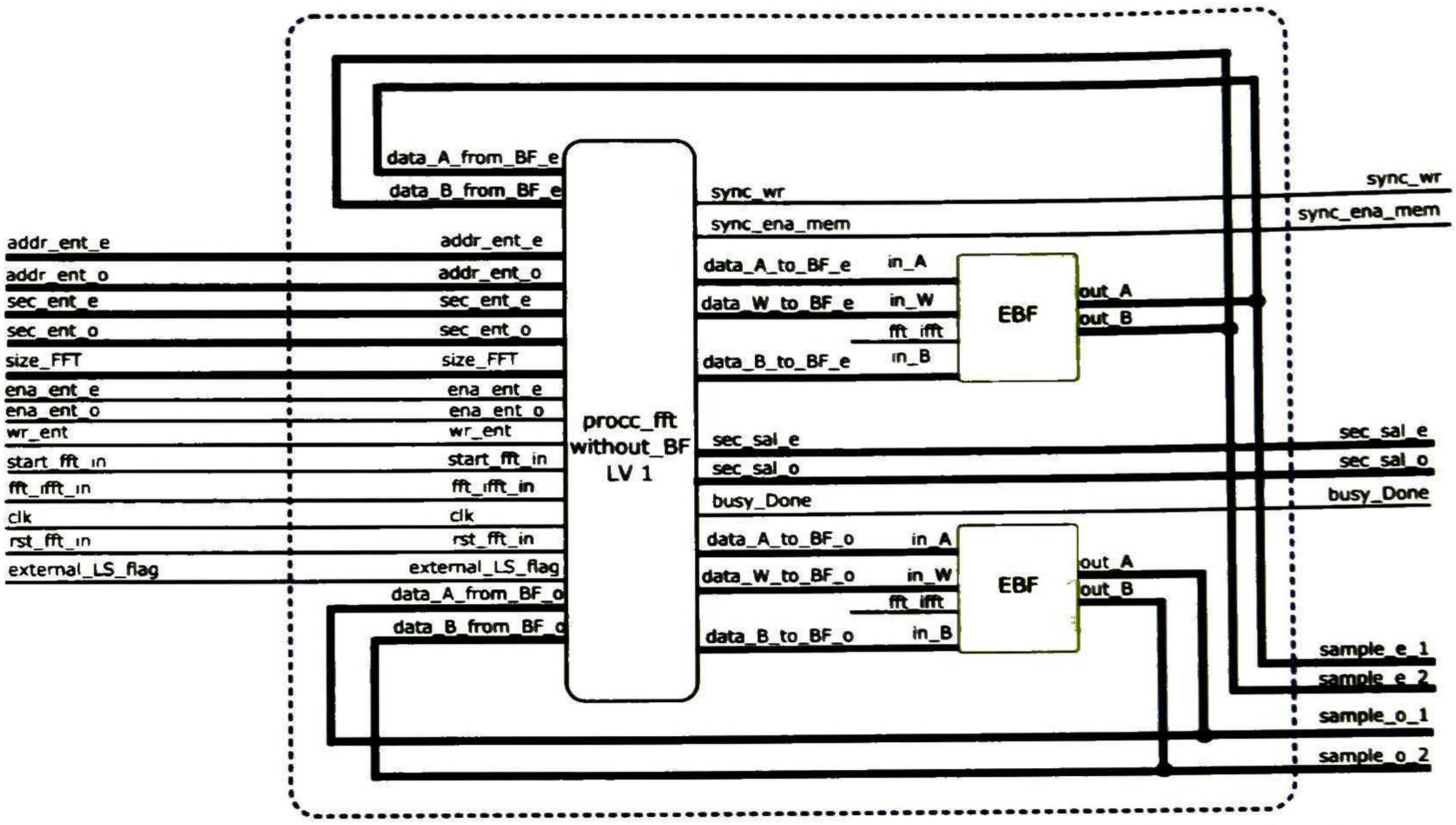


Figura 3.13: Procesador FFT/IFFT DEBF de nivel 1: Diagrama de módulos funcionales

apreciar que esto es posible porque cada procesador de nivel 0 cuenta con los puertos **e** y **o**. En consecuencia, en cada procesador de nivel 0 se pueden leer y escribir dos muestras simultáneamente. El resultado de ambas mariposas es enviado a los puertos **sample_e_1**, **sample_e_2**, **sample_o_1** y **sample_o_2**.

Cuando la unidad de control del nivel 1 requiere ejecutar la LBFS, activa las señales **flag_LBFS**, **start_LBFS** y **ena_LBFS**.

La señal **flag_LBFS** es utilizada como señal de selección en los multiplexores **mux_sec_in_A_e**, **mux_sec_in_A_o**, **mux_sec_in_B_e** y **mux_sec_in_B_o**. Esto produce que los datos que provienen de las mariposas sean recibidos por los procesadores de nivel 0. El **core 1** recibe a **data_A_from_BF_e** en el puerto **sec_ent_e** y a **data_A_from_BF_o** en el puerto **sec_ent_o**. Similarmente, **core 2** recibe a **data_B_from_BF_e** en el puerto **sec_ent_e** y a **data_B_from_BF_o** en el puerto **sec_ent_o**.

Las señales **start_LBFS** y **ena_LBFS** activan a **SM_LBFS**, el cual habilita la generación de direcciones para la última etapa de mariposas al activar la señal **ena_agu**. Además coordina la lectura y escritura de muestras en **core 1** y **core 2**, mediante las señales **ena_mem** y **wr_BF_result**.

Cuando el módulo **fuente de direcciones** recibe la señal **ena_agu**, comienza a generar las direcciones **addr_sample_e** y **addr_sample_o** para obtener 4 muestras de los procesadores de nivel 0. Además, genera las direcciones **add_Wn_e** y **add_Wn_o** para obtener dos factores rotatorios de las memorias **ROM_Wn**.

Con la dirección **addr_sample_e** se obtiene una muestra par en cada procesador de

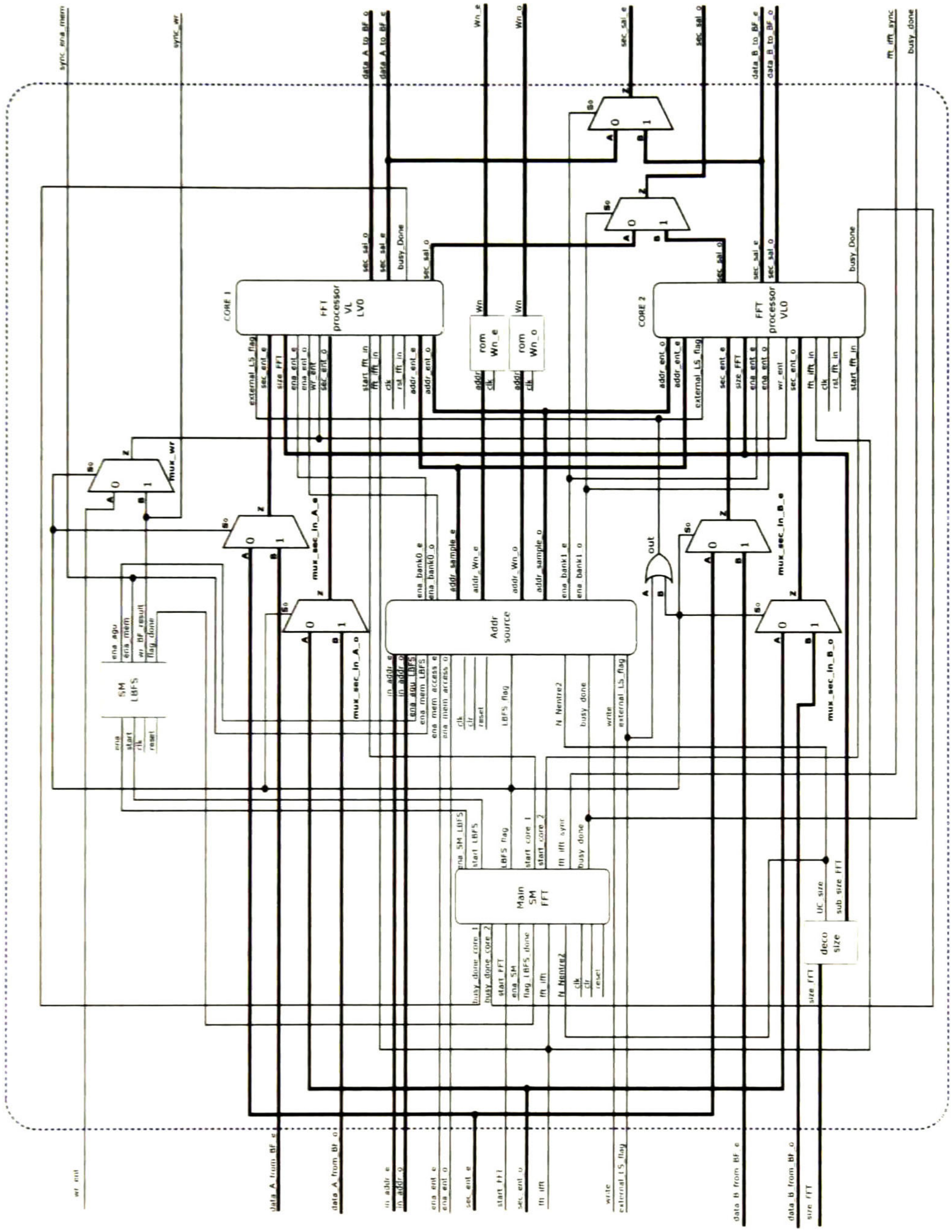


Figura 3.14: Procesador FFT/IFFT sin DEBF de nivel 1: Diagrama de módulos funcionales

nivel 0. Estas muestras son obtenidas en el puerto **sec_sal_e**. Del mismo modo, con la dirección **addr_sample_o** se obtiene una muestra impar en cada procesador de nivel 0. Estas muestras son obtenidas en el puerto **sec_sal_o**. La muestra par obtenida de **core 1** se envía a través del puerto **data_A_to_BF_e** y la obtenida de **core 2** se envía a través del puerto **data_B_to_BF_e**. Homólogamente, la muestra impar obtenida de **core 1** se envía a través del puerto **data_A_to_BF_o** y la obtenida de **core 2** se envía a través del puerto **data_B_to_BF_o**. Con las 4 muestras y los dos factores rotatorios, las dos mariposas se ejecutan y las 4 muestras resultantes son recibidas en los puertos **data_A_from_BF_e**, **data_A_from_BF_o**, **data_B_from_BF_e** y **data_B_from_BF_o**. Los procesadores almacenan estas 4 muestras en la misma dirección dónde se originaron. Posteriormente, el módulo **fuente de direcciones** genera nuevas direcciones para obtener las siguientes 4 muestras y los 2 factores rotatorios. El proceso es repetido hasta terminar con todas las mariposas de la última etapa.

Otra modificación realizada es que para poder obtener dos muestras de salida en el nivel 1, se agregó un multiplexor para la secuencia de salida impar **sec_sal_o**. Así, una muestra se obtiene por el puerto **sec_sal_e** y la otra por **sec_sal_o**.

En el nivel 1, las señales **wr_BF_result** y **ena_mem** son enviadas a través de los puertos **sync_wr** y **sync_ena_mem**. Estas señales serán activadas cuando la LBFS del nivel 1 se ejecute.

Debido a que los procesadores de nivel 0 no ejecutan una LBFS interna, la FFT/IFFT de menor longitud no puede proporcionar el resultado haciendo uso de las señales **sync_ena_mem** y **sync_wr**. Esto es lo que hace diferente a la composición interna del nivel 1 respecto a los niveles superiores.

La implementación del procesador DEBF de nivel Lv es la expansión del nivel 1. En la figura 3.15 se puede apreciar que el procesador DEBF de nivel Lv procura proporcionar el resultado que se da durante la LBFS en el nivel requerido. Es decir, puede proporcionar el resultado de la LBFS del nivel Lv o el de los niveles inferiores. Por esta razón es que aparece el módulo **mux_sync_samples**, el cual tiene por objetivo multiplexar las 4 muestras resultantes de las mariposas externas con las 4 muestras que provienen desde las mariposas que se encuentran en el interior del procesador FFT/IFFT sin EBF. La señal de selección para este multiplexor es llamada **ext_int**. En la figura 3.16 se aprecia que esta señal proviene de **deco size**. Similar a como sucede con las muestras resultantes de la LBFS, las señales **sync_wr** y **sync_ena_mem** también son multiplexadas para elegir entre las señales **sync_wr** y **sync_ena_mem** del nivel Lv o las que provienen del interior de **core_1**. Es por ello que en el procesador sin DEBF de nivel Lv aparecen los multiplexores **mux_sync_wr** y **mux_sync_ena_mem**.

A continuación se presentan los módulos que fueron modificados en el procesador SEBF para obtener al procesador DEBF.

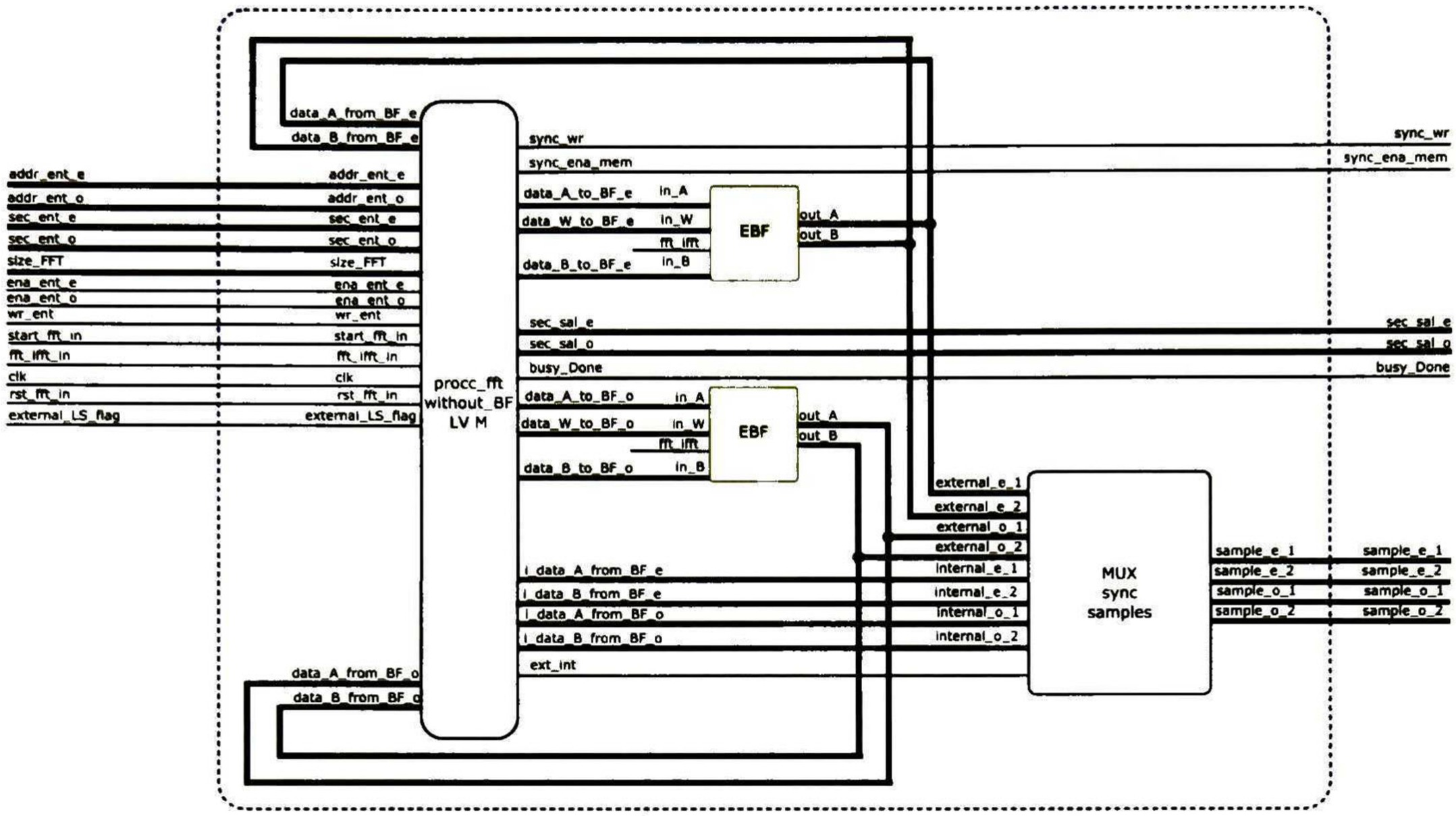


Figura 3.15: Procesador FFT/IFFT DEBF nivel Lv: Diagrama de módulos funcionales

3.4.2.1. Control de la última etapa de mariposas

La unidad de control de la última etapa de mariposas **SM LBFS**, usada en el procesador DEBF, funciona igual que la utilizada en el procesador SEBF de la sección 3.3.2.2. La diferencia es que en la versión DEBF, el tiempo requerido para activar la señal **flag_done** debe reducirse a la mitad en comparación con la versión usada en SEBF. Esto se debe a que la segunda mariposas externa produce que la LBFS sea ejecutada al doble de velocidad.

3.4.2.2. Fuente de direcciones

Al igual que el procesador DEBF y SEBF, el módulo **addr source** usado en DEBF, es una extensión del usado en SEBF. En este sentido, los puertos **addr_ent**, **sec_ent** y **ena_ent** del la versión SEBF, son los puertos **addr_ent_e**, **sec_ent_e** y **ena_ent_e** en la versión DEBF. El hecho de que en la versión DEBF se modificara el nombre de estos puertos, se debió a que se agregó un duplicado de cada puerto. Por lo tanto, para distinguir cada par, los puertos **addr_ent**, **sec_ent** y **ena_ent** se renombraron como **addr_ent_e**, **sec_ent_e** y **ena_ent_e**. Por otro lado, los puertos **addr_ent_o**, **sec_ent_o** y **ena_ent_o** fueron los puertos agregados. Por este motivo no hay necesidad de agregar una tabla para describir estos puertos, pues su funcionamiento es idéntico al mostrado en la tabla3.14. Lo mismo aplica para los puertos de salida **ena_bank0_e**, **ena_bank1_e**, **ena_bank0_o** y **ena_bank1_0**.

Los puertos **addr_sample_e** y **addr_sample_o** funcionan igual a como fue descrito en

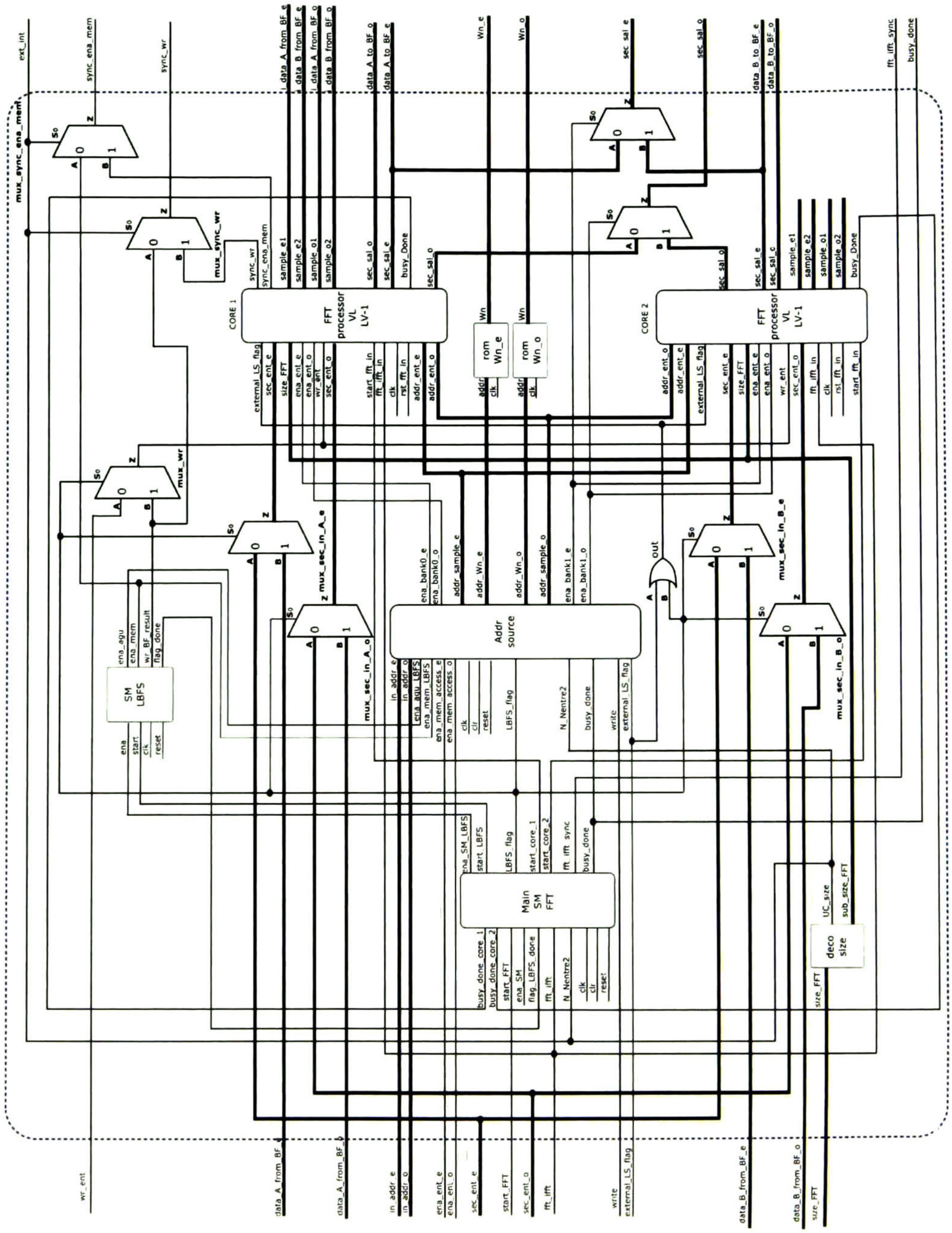


Figura 3.16: Procesador FFT/IFFT sin DEBF nivel Lv: Diagrama de módulos funcionales

la sección 3.3.2.3. No obstante, durante la LBFS existe una diferencia entre ambos puertos. Para generar una FFT/IFFT de longitud N, las direcciones del puerto **addr_sample_e**, van desde 0 hasta $\frac{N}{2} - 2$. Por su parte, las direcciones del puerto **addr_sample_o** van desde 1 hasta $\frac{N}{2} - 1$. Otra diferencia es que éstas direcciones se incrementan en 2 después de cada mariposa ejecutada. Por lo tanto, se duplica la velocidad con la que la LBFS es ejecutada.

3.4.3. Versiones modificadas del procesador FFT/IFFT

En esta sección se presentan otras versiones implementadas del procesador FFT/IFFT de longitud variable. Debido a que no fueron utilizadas en el transmisor OFDM, sólo se les hace mención.

3.4.3.1. Procesador FFT/IFFT SEBF de 5 niveles

Esta versión es similar a versión SEBF de 4 niveles, sin embargo, en esta versión se dispone de 6 longitudes de FFT/IFFT. Las longitudes de FFT/IFFT son 2048, 1024, 512, 256, 128 y 64. En esta versión se buscó dividir la FFT/IFFT de 128 muestras en 2 que se ejecutaran en paralelo y con ello, disminuir el tiempo de procesamiento. Sin embargo, el número de mariposas se duplicó en relación a la versión de 4 niveles.

3.4.3.2. Procesador FFT/IFFT SEBF de 4 niveles con BF compartida (ShBF)

Este procesador es en esencia el procesador FFT/IFFT SEBF de 4 niveles de la sección 3.3.2, pero con los niveles 0 y 1 modificados. En los procesadores de nivel 0 se extrajo la mariposa externa. De este modo, en el nivel 1, una mariposa es utilizada como recurso compartido por los procesadores de nivel 0 modificados. Esto permite disminuir el número de mariposa a la mitad en comparación con la versión SEBF de la sección 3.3.2.

3.5. Conformador de trama

La principal tarea de este módulo es generar secuencias de ceros, datos y pilotos colectivamente llamados símbolos. Como se explicó en la sección 3.5.2.2, esta secuencia de símbolos es de longitud N y es modulada por el procesador IFFT para obtener un símbolo OFDM. También se mencionó que el número de subportadora fue utilizado como índice para posicionar un símbolo dentro de la secuencia. En el caso de la arquitectura, el conformador de trama (**FC**) genera una dirección para cada símbolo, cuya función es la misma que la del número de subportadora.

3.5.1. Descripción de caja negra

El módulo **FC** es mostrado en la figura 3.17. Está encargado de coordinar al procesador FFT/IFFT, habilitar la transferencia del resultado de la IFFT al *buffer* de salida (**OB**) y activar la señal **ena_tx** para realizar la transmisión. Sus entradas y salidas son mostradas en la tabla 3.18. Los valores de las señales **BW**, **dataModuleMode** y **sel_mem**, ya han sido reportados en las tablas 3.2, 3.3 y 3.4.

La configuración del conformador de trama consiste en manipular las señales **InData**, **sel_mem** y **wr** para cargar los registros de configuración mencionados en la tabla 3.4. La manera de realizar la carga de estos registros es idéntica a la descrita en la sección 3.1.1. Con los datos y pilotos cargados en la etapa de configuración, se espera que la señal **start** sea activada. Cuando esto sucede, el ancho de banda y el tipo de modulación de datos ya deben estar definidos en los puertos **BW** y **dataModuleMode**.

Una vez que la señal **start** se activa, comienza la generación de símbolos y direcciones para el procesador FFT/IFFT. Además, el tamaño de FFT/IFFT y el número máximo de direcciones para el procesador FFT/IFFT son definidos en los puertos **FFT_size** y **max_FFTaddr**. Cuando **FC** comienza a generar los símbolos y direcciones en los puertos **FFT_sample** y **FFT_addr**, también activa la señal **ena_mem_fft** para habilitar la escritura en el procesador FFT/IFFT. Después de que **FC** ha escrito todas los símbolos en el procesador FFT/IFFT, la señal **ena_mem_fft** es desactivada. Posteriormente, revisa el valor de la señal **available_space OUTFIFO**. Si esta señal está en **1**, entonces activa la señal **start_fft**. En caso contrario, espera hasta que el valor de **available_space_OUTFIFO** sea **1**.

Después de un tiempo de haber activado el cálculo de la IFFT, la señal **fft_done** será activada, lo cual marcará el inicio de la transferencia de datos del procesador FFT/IFFT al *buffer* de salida. Dicha transferencia es habilitada activando las señales **ena_transfer_fft** y **start_transfer_fft**, ésta última sólo se activa durante un ciclo de reloj. Cuando la transferencia se ha completado, la señal **fft_transferred** se activa. Si esto sucede, se

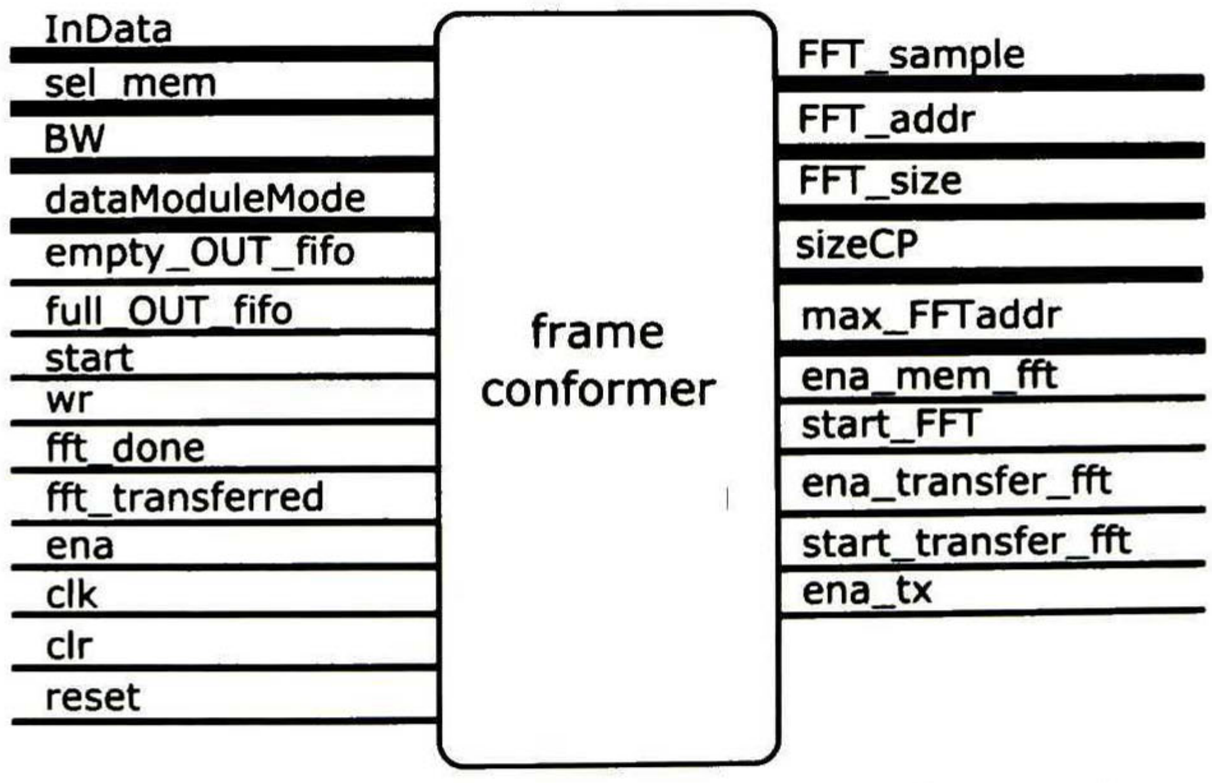


Figura 3.17: Conformador de trama: Diagrama de entradas y salidas

desactiva **ena_transfer_fft**. Una vez que el **OB** indica que no esta vacío, mediante la señal **empty_OUT_fifo**, **FC** activa la señal **ena_tx**. Esto produce que el símbolo OFDM, almacenado en el *buffer* de salida, comience a ser transmitido. El proceso descrito es realizado nuevamente para los siguientes símbolos OFDM. Sin embargo, la señal **ena_tx** no se desactiva hasta que toda la trama haya sido transmitida.

3.5.2. Descripción de caja blanca

El conformador de trama es mostrado en la figura 3.18. Los módulos que lo componen son mostrados en la tabla 3.19.

Después de haber configurado el módulo y haber recibido la señal **start**, la generación de trama comienza. La primera acción de la unidad de control de **FC** es establece el valor de las señales **sync_BW** y **sync_dataModuleMode**.

La señal **sync_BW** es utilizada por el decodificador de ancho de banda para generar las señales **size_FFT**, **addr_NnullCarriers**, **initial_addr_REG_pilots** y **max_addr_FFT**. Por su parte, la dirección **addr_NnullCarriers** es utilizada para leer una palabra del registro **NCR**. Esta palabra es recibida por el conformador de símbolos OFDM (**OFDMSyC**) a través del puerto **NnullCarriers**. Con ella **OFDMSyC** determina la cantidad de ceros que deberá insertar en el símbolo OFDM. Además, **addr_NnullCarriers** también es utilizada para obtener una palabra del registro **CPR**. La palabra obtenida es enviada al puerto **size_CP**.

La siguiente acción realizada por **FCCU** es la carga de la primer palabra de datos en el modulador. Para esto, **FCCU** fija el valor de la señal **sel_mux_InitProcess** en **0** y activa las señales **ena_ini_LOAD_data_MODUL** y **ena_ini_LOAD_MODUL**. Esto hace que las señales **enaData** y **load**, del modulador BPSK-QAM, se activen. Como consecuencia, el modulador carga la palabra que proviene de **DR** a través de la entrada **InData**.

Posteriormente, **FCCU** utiliza la dirección **addr_RegConfig** para obtener las primeras dos localidades del registro **FR**. **FCCU** recibe estas localidades a través del puerto **Word_RegConfig**. Esto le permite saber el número de ranuras de tiempo (N_{slots}) configuradas y el número de símbolos OFDM (N_{Sy}) dentro de una ranura. Después de esto, **FCCU** cambia el valor de la señal **sel_mux_InitProcess** a **1**. Esto hace que las señales **enaData** y **load**, del modulador BPSK-QAM, queden controladas por **OFDMSyC**. Además, activa las señales **ena_OFDMslotConf** y **start_OFDMslotConf**. Cuando **OFDMSlotC** recibe estas señales, inicia su procesamiento utilizando la señal **addr_REG_slotConfig** para leer una palabra del registro **SltR**. Dicha palabra es recibida a través del puerto **slotConfigWord**. **OFDMSlotC** decodifica esta palabra para generar las señales **pilotDist**, **pilotOffset** y **enaPilots**. Simultáneamente, activa las señales **ena_OFDMsymbConf** y **start_OFDMsymbConf**.

Cuando **OFDMSyC** recibe estas señales, comienza a coordinar las señales para generar los símbolos que el procesador FFT/IFFT requiere. Además, este modulo se encarga de generar una dirección para cada símbolo.

Tabla 3.18: Entradas y salidas del módulo conformador de trama

Nombre	Descripción	bits
	Entradas	
InData	Bus de datos de entrada	32
sel_mem	Selector de memoria	3
BW	Selector de ancho de banda	3
dataModuleMode	Selector del tipo de modulación de datos	2
empty_OUT_fifo	Bandera de *buffer* de salida vacío, activa en alto	1
available_space_OUTFIFO	Bandera del *buffer* de salida. En **1** indica que hay espacio disponible para almacenar un símbolo OFDM	1
start	Señal de inicio	1
wr	Señal de habilitación de escritura	1
fft_done	**0**: procesador FFT/IFFT ocupado; **1**: procesador FFT/IFFT listo	1
fft_transferred	**0**: FFT/IFFT transferida; **1**: FFT/IFFT no transferida	1
ena	Señal de habilitación	1
clk	Señal de reloj	1
clr	Señal de reinicio síncrono	1
reset	Señal de reinicio asíncrono	1
	Salidas	
FFT_sample	símbolo para el procesador FFT/IFFT	32
FFT_addr	Dirección para el procesador FFT/IFFT	11
size_CP	Longitud del prefijo cíclico	11
FFT_size	Longitud de FFT/IFFT	3
max_FFTaddr	Número máximo de direcciones FFT/IFFT	11
ena_mem_fft	Señal de habilitación de acceso a memoria del procesador FFT/IFFT	1
start_FFT	Señal de inicio del procesador FFT/IFFT	1
ena_transfer_fft	Señal de habilitación de transferencia de datos entre el procesador FFT/IFFT y el *buffer* de salida	1
ena_transfer_fft	Señal de inicio de transferencia de datos entre el procesador FFT/IFFT y el *buffer* de salida	1
ena_tx	Señal de habilitación de transmisión de símbolos OFDM	1

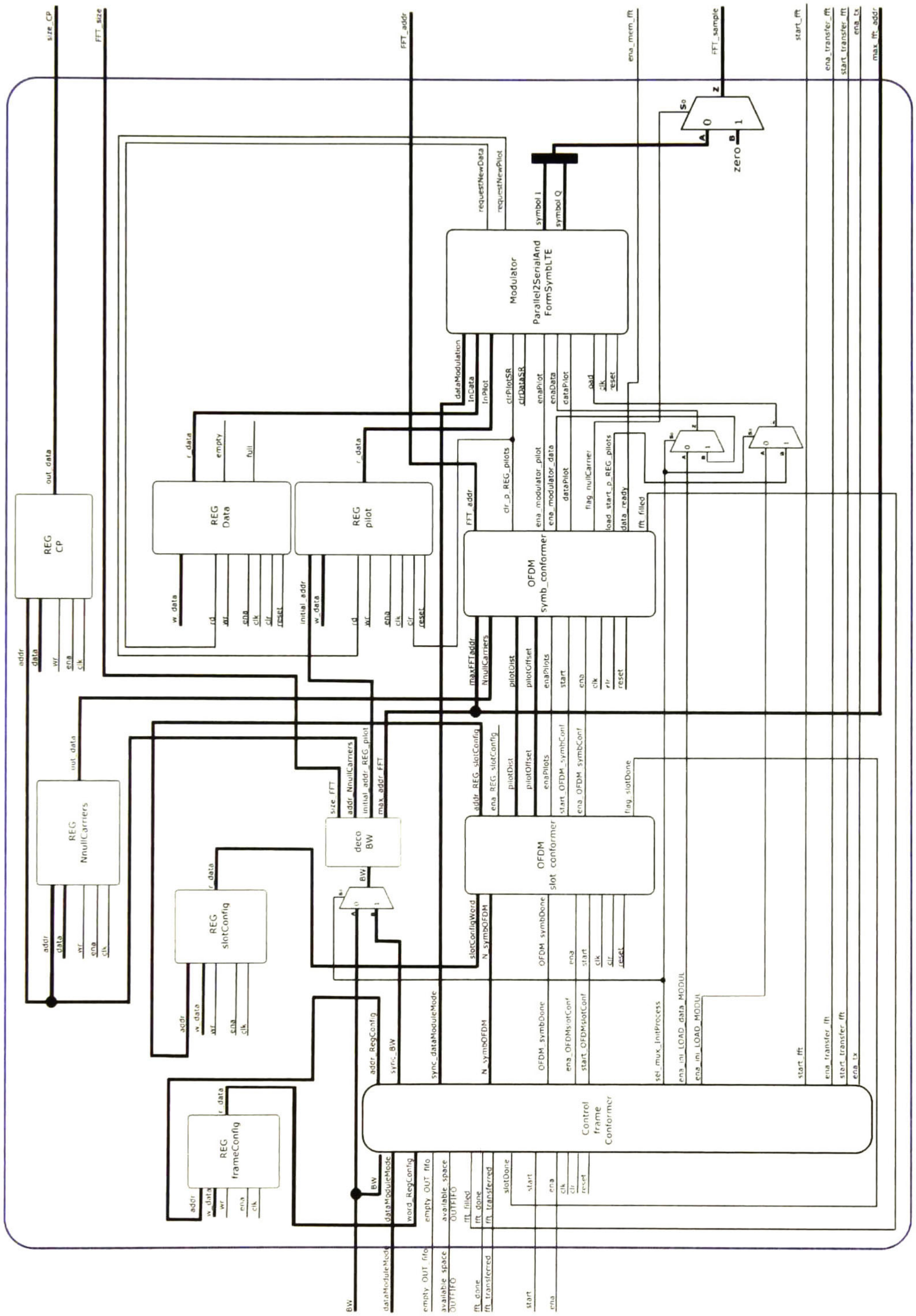


Figura 3.18: Conformador de trama (**FC**): Diagrama de módulos funcionales

Tabla 3.19: Conformador de trama: Lista de módulos funcionales

Módulos	Descripción	Abreviatura
REG_CP	Registro de configuración de prefijo cíclico	**CPR**
REG_frameConfig	Registro de configuración de trama	**FR**
REG_slotConfig	Registro de configuración de ranura	**SltR**
REG_NnullCarrier	Registro de portadoras nulas	**NCR**
REG_pilots	Registro de pilotos	**PR**
DATA_FIFO	FIFO de datos	**DR**
OFDM_slotConformer	Conformador de ranura OFDM	**OFDMSltC**
OFDM_symb_conf	Conformador de símbolo OFDM	**OFDMSyC**
parallel2SerialAndSymbForm	Modulador BPSK-QAM	
deco_BW	Decodificador de ancho de banda	
decoAddrIFFTshift	Decodificador de direcciones IFFT	
MUC_frameConformer	Unidad de control del conformador de trama	**FCCU**

Existen tres tipos de señales que son consideradas símbolos para el procesador FFT/IFFT: portadoras nulas, datos y pilotos. El orden en el que estos tres aparecen en el símbolo OFDM está determinado por las señales **NnullCarriers**, *pilotDist*, *pilotOffset* y *enaPilots*.

Si **OFDMSyC** recibe la señal *enaPilots* en **0**, no es necesario realizar la carga inicial de pilotos en el modulador BPSK-QAM. Sin embargo, si viene en **1**, hay que realizarla. Con este fin, **OFDMSyC** activa la señal **clr_p_REG_pilots** para limpiar el registro interno de pilotos del modulador. La señal **clr_p_REG_pilots** también provoca que el registro de pilotos se posicione en la dirección **initial_addr_REG_pilot**. Esto permite obtener la primer palabra de pilotos asociados al ancho de banda seleccionado. Posteriormente, **OFDMSyC** activa las señales **ena_modulator_pilot** y **load_start_p_REG_pilots**. Esto hace que las señales **enaPilot** y **load**, del modulador BPSK-QAM, se activen. En consecuencia, el modulador carga la palabra que proviene del registro de pilotos a través de la entrada **InPilot**.

Para este punto, el modulador ha sido inicializado con una palabra de datos y otra de pilotos. Por lo tanto, esta listo para comenzar la modulación y proporcionar los símbolos para el procesador FFT/IFFT.

La generación de símbolos y direcciones es controlada a través de cinco fases ejecutadas por **OFDMSyC**. Estas cinco fases son descritas en la sección 3.5.2.7. A lo largo de estas fases, **OFDMSyC** activa la señal **data_ready** y establece el valor de las señales **flag_nullCarrier**, **ena_modulator_data** y **ena_modulator_pilot**. Al finalizar las cinco fases, se tendrá al procesador FFT/IFFT cargado con las símbolos para formar al símbolo OFDM.

Las señales **ena_modulator_data** y **ena_modulator_pilot** activan la modulación de datos y pilotos. Cuando el modulador BPSK-QAM termina con la palabra actual de datos o pilotos, activa las señales **requestNewData** o **requestNewPilot**, según sea el caso. Estas señales son conectadas a la entrada **rd** de los registros **DR** y **PR** respectivamente.

En consecuencia, el respectivo registro prepara la siguiente palabra y el modulador BPSK-QAM la carga automáticamente. El modulador utiliza los puertos **symbol_I** y **symbol_Q** para enviar las señales moduladas al multiplexor de muestras. El multiplexor de muestras utiliza la señal **flag_nullCarrier** para determinar si la salida será la señal modulada o a una constante cero.

Cuando el **OFDMSyC** termina de cargar al procesador FFT/IFFT, activa la bandera **FFT_filled**. La unidad de control recibe esta señal y posteriormente activa la señal **start_fft**, siempre y cuando **available_space_OUTFIFO** este en **1**. Esto inicia el procesamiento del módulo FFT/IFFT. Cuando el cálculo termina, el procesador FFT/IFFT activa la señal **fft_done**. Al recibir esta señal, la unidad de control activa la señal **ena_transfer** para habilitar la de transferencia de símbolos del procesador FFT/IFFT al *buffer* de salida (**OB**). Otra señal que es activada es **ena_tx**, la cual habilita la transmisión del símbolo OFDM.

La señal **fft_transferred** se activa al finalizar la transferencia de símbolos. Cuando la **FCCU** recibe esta señal, activa la señal **OFDM_symbolDone** y desactiva la señal **ena_transfer**. No obstante, la señal **ena_tx** no se desactiva hasta transmitir el último símbolo OFDM de la trama.

La señal **OFDM_symbolDone** es utilizada por **OFDMSlotC** como bandera para iniciar la generación del siguiente símbolo OFDM. Para esto, la dirección **addr_REG_slotConfig** se actualiza para obtener la siguiente palabra del registro **SltR**. El resto del proceso de generación de símbolo OFDM es idéntico al recién descrito.

Cuando todos los símbolos OFDM dentro de una ranura han sido formados, **OFDMSlotC** activa la señal **flag_slot_done**. Esta señal es recibida por **FCCU**. Como resultado, **FCCU** activa nuevamente a **OFDMSlotC** para generar al siguiente conjunto de símbolos OFDM que conforman la ranura. Cuando todas las ranuras han sido realizadas, la **FCCU** desactiva las señales **ena_OFDMslotConf** y **start_OFDMslotConf**. Esto provoca que la generación de símbolos OFDM cese. El resto consiste en mantener activa la señal **ena_tx** hasta que **OB** informe que esta vacío mediante la activación de la señal **empty_OUT_FIFO**.

3.5.2.1. Registro de configuración de trama

El registro **FR** tiene un tamaño de palabra de 23 bits. La primera localidad debe almacenar el número de ranuras dentro de la trama (N_{Slots}); mientras que la segunda debe almacenar el número de símbolos OFDM dentro de una ranura (N_{sy}).

3.5.2.2. Registro de configuración de ranura

El registro **SltR** de la figura 3.19 tiene un tamaño de palabra de 23 bits. Habrán N_{sy} localidades, cada una asociada a un símbolo OFDM. Las localidades deben almacenar los parámetros *pilotOffset*, *pilotDist* y *enaPilots*.

Dirección	bit 22	...	bit 12	bit 11	...	bit 1	bit 0	Símbolo OFDM
reg 0	*pilotOffset*			*pilotDist*			*enaPilots*	0
reg 1	*pilotOffset*			*pilotDist*			*enaPilots*	1
⋮	⋮			⋮			⋮	⋮
reg $N_{sy}-2$	*pilotOffset*			*pilotDist*			*enaPilots*	$N_{sy}-2$
reg $N_{sy}-1$	*pilotOffset*			*pilotDist*			*enaPilots*	$N_{sy}-1$

Figura 3.19: Registro de configuración de ranura

3.5.2.3. Registro número de portadoras nulas

Del mismo modo en el que cada ancho de banda determina un tamaño de FFT/IFFT, también determina un número de portadoras nulas. Por lo tanto, este registro necesita tantas localidades como anchos de banda **BW** existan. El parámetro obtenido de este registro es utilizado por el conformador de símbolo OFDM para determinar la cantidad de ceros que se insertarán en la FFT/IFFT. Se decidió que la primera localidad contenga el número de portadoras nulas asociado al primer ancho de banda; la segunda localidad, el número de portadoras nulas asociado al segundo ancho de banda; y así sucesivamente.

3.5.2.4. Registro de Prefijo Cíclico

Este registro necesita tantas localidades como anchos de banda **BW** existan. El parámetro obtenido de este registro es utilizado para determinar la cantidad de símbolos leídos del procesador FFT/IFFT como prefijo cíclico. Se decidió que la primera localidad contenga al CP asociado al primer ancho de banda; la segunda localidad, al CP asociado al segundo ancho de banda; y así sucesivamente.

3.5.2.5. Registro de pilotos

El tamaño de palabra de este registro es de 32 bits. Se estableció que cada ancho de banda tenga asociadas 8 localidades de este registro. En consecuencia, habrán 40 localidades, dónde las primeras 8 deberán almacenar los pilotos usados para el primer ancho de banda y así sucesivamente. Debido a que se utilizan los mismos pilotos para los diversos símbolos OFDM, es necesario que cada vez que se inicie la formación de un nuevo símbolo OFDM, se active en alto la señal **clr**. Esto lleva al apuntador al inicio de la región de memoria correspondiente al ancho de banda. Dicho inicio de región es indicado por la dirección presente en la entrada **initial_addr**.

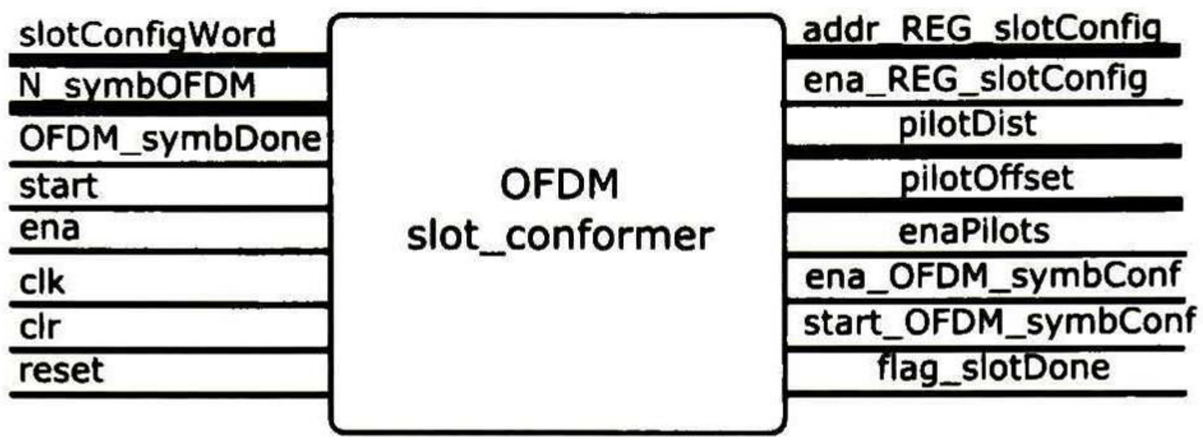


Figura 3.20: Conformador de ranura OFDM

3.5.2.6. Conformador de ranura OFDM

El módulo **`OFDMSlotC`** es mostrado en la figura 3.20. Sus entradas y salidas son mostradas en la tabla 3.20. Este módulo es capaz de coordinar la generación de símbolos ODFM bajo diferentes configuraciones. Si se configura el número de símbolos OFDM, así como la distancia y offset entre los pilotos acorde a los valores de la figura 3.2, se obtiene la ranura particular de LTE de la figura 1.8. Para lograr la reconfigurabilidad, **`OFDMSlotC`** se apoya en el registro **`SltR`**.

Al estar habilitado y recibir la señal **start**, el módulo **`OFDMSlotC`** lee una palabra del registro **`SltR`** a través del puerto **slotConfigWord**. Esta palabra es utilizada para generar las señales *pilotOffset*, *pilotDist* y *enaPilots*. Después, el conformador de ranura activa las señales **ena_OFDM_symbConf** y **start_OFDM_symbConf**. Posteriormente, realiza una pausa para dar tiempo a la generación del símbolo OFDM.

Cuando la bandera **OFDM_symbDone** se activa, la dirección **addr_REG_slotConfig** se incrementa para leer la siguiente palabra y repetir el proceso de generación de símbolos OFDM. Una vez que todos los símbolos se han generado, la señal **flag_slotDone** se activa para notificar que una ranura se ha completado.

3.5.2.7. Conformador de símbolo OFDM

El módulo **`OFDMSyC`** es mostrado en la figura 3.21. Sus entradas y salidas son mostradas en la tabla 3.21. Después de la activación de la señal **start**, el módulo **`OFDMSyC`** realiza dos tareas principales: la primera, es generar la secuencia de direcciones necesarias para el

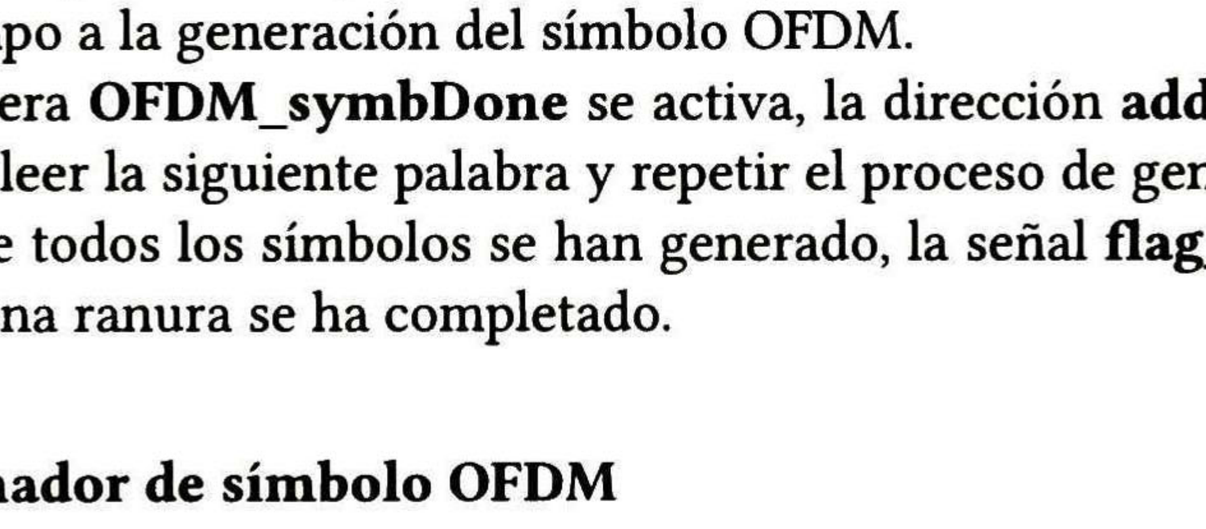


Figura 3.21: Conformador de símbolo OFDM

Tabla 3.20: Entradas y salidas del conformador de ranura OFDM

Nombre	Descripción	bits
	Entradas	
slotConfigWord	Palabra que contiene la información de configuración de los símbolos OFDM	23
NsymbOFDM	Número de símbolos OFDM presentes en una ranura de tiempo	3
OFDM_symbDone	Bandera de estado de conformación de símbolo OFDM **1**: Símbolo realizado **0**: Símbolo no realizado	1
ena	Señal de habilitación, activa en alto	1
start	Señal de inicio, activa en alto	1
clk	Señal de reloj	1
clr	Señal de reinicio síncrono, activa en alto	1
reset	Señal de reinicio síncrono, activa en bajo	1
	Salidas	
addr_slotConfig	Dirección para el registro de configuración de ranura	3
ena_slotConfig	Señal de habilitación del registro de configuración de ranura	1
pilotDist	Distancia entre pilotos	11
pilotOffset	Offset entre pilotos	11
enaPilots	Bandera de inclusión de pilotos **1**: Símbolo OFDM con pilotos **0**: Símbolo OFDM sin pilotos	1
start_OFDMsymbConf	Señal de inicio para el conformador de símbolos OFDM, activa en alto	1
ena_OFDMsymbConf	Señal de habilitación para el conformador de símbolos OFDM, activa en alto	1
flag_slotDone	Señal de estado de conformación de ranura **1**: Ranura completada **0**: Ranura no completada	1

ingreso de entradas en el procesador FFT/IFFT a través del puerto **FFT_addr**; la segunda, es generar las señales requeridas para que los datos y pilotos sean modulados y proporcionados en el instante justo en el que su dirección correspondiente esta presente. La asociación dirección-símbolo esta determinada por las señales **NnullCarriers**, *pilotOffset*, *pilotDist* y *enaPilots*. Las primeras y últimas direcciones, al igual que la dirección correspondiente a la componente de corriente directa tendrán asociada un símbolo con valor cero. El resto de direcciones tendrán asociados símbolos modulados, ya sean datos y pilotos o sólo datos. El orden en el que estos símbolos son generadas se describe a través de las siguientes cinco fases:

Fase 1: Generación de portadoras nulas. **OFDMSyC** mantiene desactivadas a las señales **ena_modulator_data** y **ena_modulator_pilot** y fija el valor de la señal **flag_nullCarrier** en **1** y. La señal **flag_nullCarrier** es utilizada como selector para el multiplexor de muestras, el cual mantendrá un cero a la salida. De este modo se almacenarán símbolos con valor cero en las primera direcciones del procesador FFT/IFFT.

Fase 2: Generación de datos y pilotos modulados. **OFDMSyC** fija el valor de la señal **flag_nullCarrier** en **0**. Si la señal *enaPilots* está en **0**, establece el valor de **ena_modulator_data** en **1** y **ena_modulator_data** en **0**. Esto produce que sólo datos sean incluidos en el símbolo OFDM. Si la señal *enaPilots* está en **1**, **OFDMSyC** comienza a conmutar la activación de las señales **ena_modulator_data** y **ena_modulator_pilot**. Esto produce que datos y pilotos sean incluidos en el símbolo OFDM. El multiplexor de muestras mantendrá las señales moduladas a la salida.

Fase 3: Generación de la componente de DC. Durante esta fase se incluye una portadora nula en el centro del símbolo OFDM. Este símbolo corresponde con la componente de corriente directa. Por ello, **OFDMSyC** fija el valor de la señal **flag_nullCarrier** en **1** y se desactivan las señales **ena_modulator_data** y **ena_modulator_pilot**.

Fase 4: Generación de datos y pilotos modulados. Idéntica a la fase 2.

Fase 5: Generación de portadoras nulas. Idéntica a la fase 1.

Al finalizar las cinco fases, **OFDMSyC** activa la señal **fft_filled**. Cabe resaltar que las señales **ena_modulator_data** y **ena_modulator_pilot** se generan un ciclo de reloj antes de realmente requerir los símbolos modulados. Esto se debe a que el modulador QAM-BPSK demora un ciclo de reloj en proporcionar la modulación. Sin embargo, la señal **dataPilot** se activa en el instante preciso en el que se desea que el modulador proporcione un piloto, en caso contrario, se entrega un dato.

Por otro lado, la señal **flag_nullCarrier** tiene por objetivo indicar si el símbolo que debe ingresar al procesador FFT/IFFT es un cero o un símbolo modulado. La cantidad de ceros que serán incluidos esta determinado por la entrada **NnullCarriers**. Las señales **clr_p_REG_pilots** y **load_start_p_REG_pilots** se activan al inicio de un símbolo OFDM, siempre y cuando éste requiere la inclusión de pilotos. La primera señal sirve para obtener la palabra inicial del registro **PR**; la segunda sirve para volver a cargar al modulador de pilotos con la palabra recién obtenida.

Tabla 3.21: Entradas y salidas del conformador de símbolos OFDM

Nombre	Descripción	bits
	Entradas	
maxFFTaddr	Máxima direcciones para el procesador FFT/IFFT	11
NnullCarriers	Número de portadoras nulas incluidas en la formación de los símbolos OFDM	11
pilotDist	Distancia entre pilotos	11
pilotOffset	Offset entre pilotos	11
enaPilots	Bandera de inclusión de pilotos **1**: Símbolo OFDM con pilotos **0**: Símbolo OFDM sin pilotos	1
ena	Señal de habilitación, activa en alto	1
start	Señal de inicio, activa en alto	1
clk	Señal de reloj	1
clr	Señal de reinicio síncrono, activa en alto	1
reset	Señal de reinicio síncrono, activa en bajo	1
	Salidas	
FFT_addr	Dirección para el procesador FFT/IFFT	11
clr_p_REG_pilots	Señal de reinicio síncrono para el registro de configuración de pilotos y para el modulador de pilotos	1
ena_modulator_pilot	Señal de habilitación de modulación de pilotos	1
ena_modulator_data	Señal de habilitación de modulación de datos	1
dataPilot	Señal de selección de símbolo **1**: Piloto modulado **0**: Dato modulado	1
flag_nullCarrier	Señal de selección de símbolo **1**: Portadora nula **0**: símbolo modulada (dato o piloto)	1
load_start_p_REG_pilots	Señal de habilitación utilizada para cargar una palabra en el modulador, activa en alto	1
data_ready	Señal activa en alto para indicar que los datos y direcciones son válidos	1
fft_filled	Señal activa en alto para indicar que todo los datos y direcciones para un símbolo OFDM han sido generados	1

3.5.2.8. Modulador BPSK-QAM

El modulador **parallel2SerialAndSymbFormLTE** es mostrado en la figura 3.22. Sus entradas y salidas son mostradas en la tabla 3.22. Este módulo esta integrado por dos moduladores, uno dedicado a la modulación de pilotos y otro a datos. Los puertos **InData** e **InPilots** permiten cargar palabras de 32 bits en los moduladores al activar la señal **load** y alguna de las señales **enaPilot** o **enaData**. Mantener en alto alguna de las señales **enaPilot** o **enaData**, después de la carga inicial, hará que un registro de corrimiento comience a desplazar los bits de la palabra recién ingresada. A su vez, los bits desplazados son decodificados en una dirección que se utiliza para leer una memoria y obtener de ella un símbolo modulada. La señal **dataPilot** permite elegir el símbolo modulado que estará presente en los puertos **symbol I** y **symbol Q**: en bajo, se tomarán las símbolos del modulador de datos; en alto, las del modulador de pilotos.

Debido a que las señales **enaPilot** y **enaData** actúan sobre registros de corrimiento, se necesita que estas se activen un ciclo de reloj antes de requerir el símbolo modulado, y de manera similar, se deben desactivar un ciclo de reloj antes de requerir una pausa en el modulador. Debido a que la señal **dataPilot** actúa sobre un multiplexor, su activación en alto debe ocurrir en el momento preciso en el que se desea un piloto modulado.

Tres símbolos antes de que la palabra presente en el registro de corrimiento se agote, se activa una de las señales **requestNewData** o **requestNewPilot** según sea el caso, esto sirve para solicitar la siguiente palabra a la memoria correspondiente. Así mismo, dos símbolos antes de que la palabra presente en el registro de corrimiento se agote, se activa una señal interna llamada **loadNewData** para que durante la aparición del siguiente y último símbolo, se realice la carga automática de la nueva palabra en el registro de corrimiento.

Dado que los pilotos son modulados en BPSK, el modulador de la figura 3.22 no cuenta con una señal de selección de modulación para pilotos, sin embargo, para los datos se cuenta con la señal **dataModulation**, que determina la cantidad de bits desplazados por el registro de corrimiento en cada símbolo requerido, además, interviene en la decodificación de la dirección usada por la memoria de símbolos moduladas. La tabla 3.23 muestra los tipos de modulación que se pueden aplicar a los datos.

3.5.2.9. Decodificador de ancho de banda

Este módulo recibe el código de ancho de banda **BW** y genera los parámetros: **addrForREG_NnullCarriers**, **initial_addr_Pilot_REG** y **maxAddrFFT** acorde a la tabla 3.24. Se decidió que esta decodificación se realizara en un módulo externo al control de **FC** porque si en un futuro se deseara cambiar las longitudes de FFT, sería posible modificando sólo este bloque y no la máquina de control.

Tabla 3.22: Entradas y salidas del modulador BPSK-QAM

Nombre	Descripción	bits
	Entradas	
dataModulation	Selector del tipo de modulación de datos	3
InData	Palabra de datos	32
InPilots	Palabra de pilotos	32
clrPilotSR	Señal de reinicio síncrono del registro de pilotos, activa en alto	1
clrDataSR	Señal de reinicio síncrono del registro de datos, activa en alto	1
enaPilot	Señal de habilitación de modulador de pilotos	1
enaData	Señal de habilitación de modulador de datos	1
dataPilot	Señal selección de símbolo modulado **1**: Piloto modulado **0**: Dato modulado	1
load	Señal de habilitación de carga. Activa en alto, permite cargar una palabra en el modulador. Si **enaData** está en **1**, el modulador de datos carga la palabra presente en la entrada **InData**. Si **enaPilot** está en **1**, el modulador de pilotos carga la palabra presente en la entrada **InPilot**.	1
clk	Señal de reloj	1
reset	Señal de reinicio síncrono, activa en bajo	1
	Salidas	
request_NewData	Señal activa en alto para solicitar la siguiente palabra de datos	1
request_NewPilot	Señal activa en alto para solicitar la siguiente palabra de pilotos	1
symbol_I	Componente I de la señal modulada	16
symbol_Q	Componente Q de la señal modulada	16

Tabla 3.23: Modulaciones permitidas por el módulo **`parallel2SerialAndSymbFormLTE`**

dataModulation	Modulación
2'b00	BPSK
2'b01	4-QAM
2'b10	16-QAM
2'b11	64-QAM

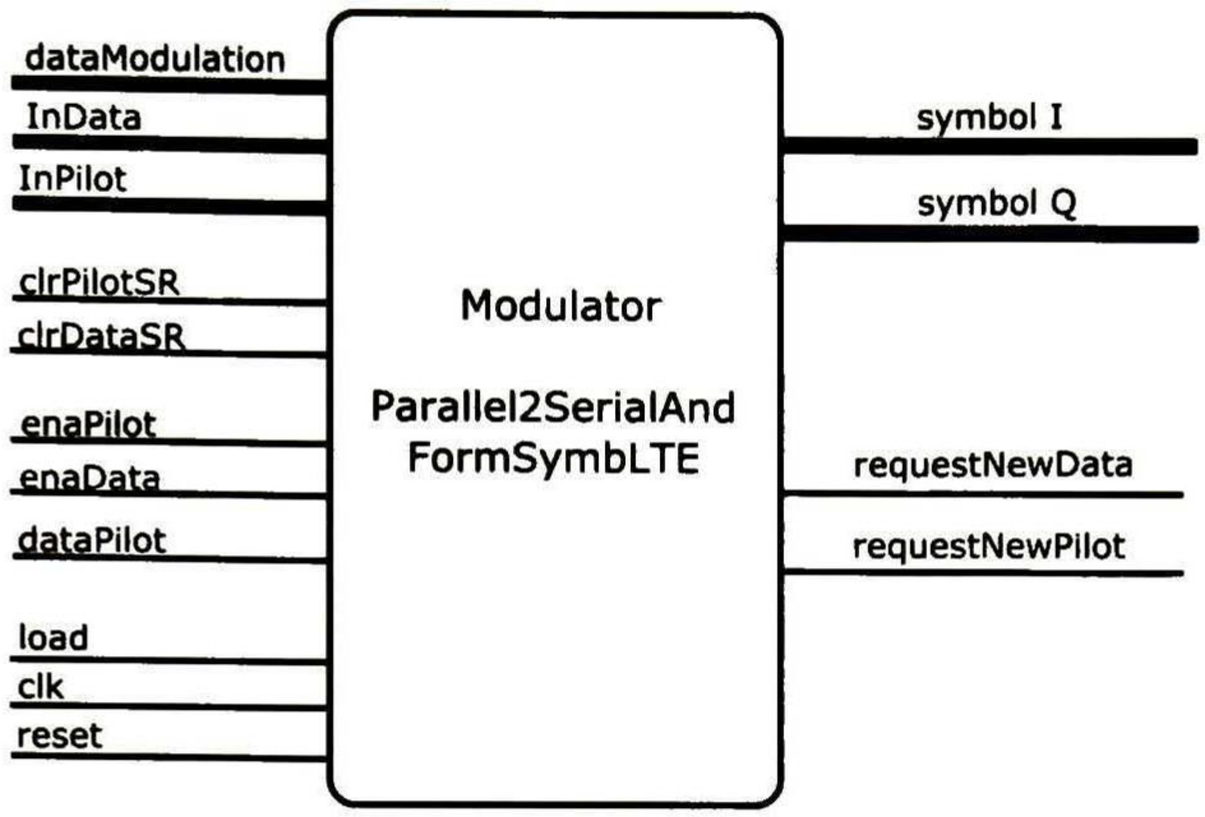


Figura 3.22: Modulador BPSK-QAM

Tabla 3.24: Parámetros decodificados a partir del ancho de banda **BW**

BW	**addrForREG** **NnullCarriers**	**initial_addr** **Pilot_REG**	**maxAddr** **FFT**	**FFT_size** code
3'b000	0	0	127	3'b100
3'b001	1	8	255	3'b011
3'b010	2	16	511	3'b010
3'b011	3	24	1023	3'b001
3'b100	4	32	2047	3'b000

Tabla 3.25: Entradas y salidas del decodificador de ancho de banda

Nombre	Descripción	bits
	Entradas	
BW	Código de ancho de banda	3
	Salidas	
size_FFT	Código de longitud de FFT	3
addr_NnullCarriers	Dirección para el registro número de portadoras nulas	3
initial_addr_REG_pilot	Dirección inicial para el registro de pilotos	6
max_addr_FFT	Máxima dirección para el procesador FFT/IFFT	11

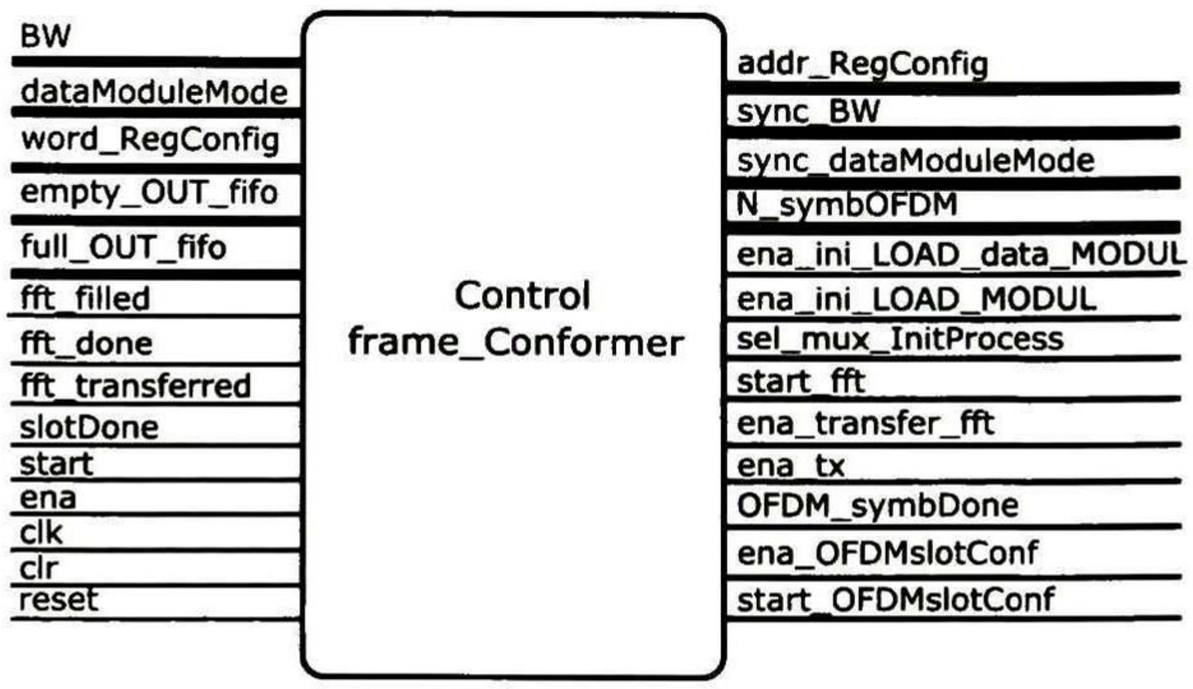


Figura 3.23: Unidad de control del conformador de trama

3.5.2.10. Unidad de control del conformador de trama

La Unidad de control del conformador de trama **FCCU** es mostrada en la figura 3.23. Sus entradas y salidas son mostradas en las tablas 3.26 y 3.27.

Durante las primeras etapas de procesamiento, **FCCU** mantiene en **0** a la señal **sel_mux_InitProcess**. Tras recibir la señal **start**, **FCCU** establece el valor de las señales síncronas **sync_BW** y **sync_dataModuleMode**. Estas señales conservan el último valor que las entradas **BW** y **dataModuleMode** hayan tenido hasta antes de la activación de la señal **start**. Pese a que las entradas **BW** y **dataModuleMode** cambien, sus homólogas síncronas no cambiarán hasta que la trama haya sido generada. Posteriormente, **FCCU** utiliza la dirección **addr_RegConfig** para leer las primeras dos palabras del registro **FR**. Las palabras son recibidas a través del puerto **word_RegConfig**. La primer palabra permite saber el parámetro N_{slots} y la segunda N_{Sy}. La siguiente acción de **FCCU** es inicializar al modulador con una palabra de datos. Esto se realiza mediante las señales **ena_ini_LOAD_data_MODUL** y **ena_ini_LOAD_MODUL**. Después de la inicialización, el proceso de generación de trama inicia. Durante esta etapa, el valor de la señal **sel_mux_InitProcess** cambia a **1** y se mantiene así hasta terminar de generar la trama.

El proceso de generación de trama inicia cuando **FCCU** activa las señales de habilitación e inicio **ena_OFDMslotConf** y **start_OFDMslotConf**. Después de un tiempo, la señal **fft_filled** será activada. Cuando esto sucede, **FCCU** revisa el estado de la señal **available_space_OUTFIFO**. Si **available_space_OUTFIFO** está en **1**, **FCCU** activa la señal **start_fft** y pasa a un estado de espera. De lo contrario no activa la señal **start_fft** hasta que **available_space_OUTFIFO** este en **1**. Además, la generación de símbolos OFDM es pausada automáticamente pese a que la señal **ena_OFDMslotConf** siga activa. Cuando la entrada **fft_done** es activada, el estado de espera termina. Como consecuencia, la unidad de control activa las señales **ena_transfer_fft** y **ena_tx**. Cuando la transferencia termina, la entrada **fft_transferred** es activada. Esto le indica a **FCCU** que debe deshabilitar la señal **ena_transfer_fft**. Sin embargo, la señal **ena_tx** no se desactiva hasta terminar de transmitir la trama completa.

Tabla 3.26: Entradas de la Unidad de control del conformador de trama

Nombre	Descripción	bits
	Entradas	
BW	Selector de ancho de banda	3
dataModulation	Selector del tipo de modulación de datos	2
word_RegConfig	Palabra de configuración de trama	23
empty_OUT_fifo	Bandera de estado del *buffer* de salida, **1** indica que está vacío	1
available_space OUTFIFO	Bandera de estado del *buffer* de salida, **1** indica que hay espacio disponible para almacenar un símbolo OFDM	1
fft_filled	Bandera de estado de escritura en el procesador FFT/IFFT **1**: Escritura de símbolos completa **0**: Escritura de símbolos no completa	1
fft_done	Bandera de estado del procesador FFT/IFFT **1**: Resultado listo **0**: Resultado no listo	1
fft_transferred	Bandera de estado de transferencia de símbolos desde el procesador FFT/IFFT al *buffer* de salida **1**: Transferencia completada **0**: Transferencia no completada	1
slotDone	Bandera de estado de ranura **1**: Ranura completada **0**: Ranura no completada	1
ena	Señal de habilitación, activa en alto	1
start	Señal de inicio, activa en alto	1
clk	Señal de reloj	1
reset	Señal de reinicio síncrono, activa en bajo	1
clr	Señal de reinicio asíncrono, activa en alto	1

Tabla 3.27: Salidas de la Unidad de control del conformador de trama

Nombre	Descripción	bits
	Salidas	
addr_RegConfig	Dirección para el registro de configuración de trama	3
sync_BW	Selector de ancho de banda Señal síncrona que toma su valor de la señal **BW** durante la activación de la señal **start**. Este valor es mantenido constante hasta el fin de la trama	3
sync_dataModulation	Selector del tipo de modulación de datos Señal síncrona que toma su valor de la señal **dataModulation** durante la activación de la señal **start**. Este valor es mantenido constante hasta el fin de la trama	2
N_symbOFDM	Número de símbolos OFDM presentes en una ranura de tiempo	3
OFDM_symbDone	Bandera de estado del símbolo OFDM **1**: Símbolo realizado y enviado al *buffer* de salida **0**: Símbolo no realizado	1
ena_OFDMslotConf	Señal del habilitación del conformador de ranura OFDM, activa en alto	1
start_OFDMslotConf	Señal del inicio del conformador de ranura OFDM, activa en alto	1
sel_mux_InitProcess	Señal de selección para multiplexor	1
ena_ini_LOAD_data_modul	Señal de habilitación del modulador de datos. Es utilizada con el propósito de realizar la carga inicial del modulador, activa en alto	1
ena_ini_LOAD_modul	Señal de habilitación del carga general para el modulador BPSK-QAM. Es utilizada con el propósito de realizar la carga inicial del modulador, activa en alto	1
start_fft	Señal de inicio para el procesador FFT/IFFT, activa en alto	1
ena_transfer_fft	Señal de habilitación de transferencia de símbolos del procesador FFT/IFFT al *buffer* de salida, activa en alto	1
ena_tx	Señal de habilitación de transmisión, activa en alto	1

Otro efecto que la señal **fft_transferred** produce cuando es activada, es la activación de la señal **OFDM_symbDone**. Esta señal reanuda la generación de símbolos OFDM. El proceso de generación de símbolos OFDM recién descrito es replicado para generar los N_{Sy} símbolos OFDM configurados. Cuando la ranura es completada, **FCCU** recibe la señal **slotDone** para llevar un conteo de ranuras realizadas. El proceso de generación de ranura es repetido hasta completar las N_{Slots} ranuras configuradas. Cuando esto sucede, la señal **ena_OFDMslotConf** es desactivada. Finalmente, **FCCU** queda en espera de que la señal **empty_OUT_fifo** vuelva a **1** para desactivar la señal **ena_tx** y regresar a su estado inicial.

3.6. Decodificador de direcciones IFFT

El decodificador de la figura 3.24 realiza el equivalente a la operación ifftshift de Matlab. Es decir, permite almacenar la primera mitad de símbolos en las segunda mitad de direcciones de la FFT/IFFT, y viceversa, almacenar los símbolos finales en la primera mitad de direcciones [23].

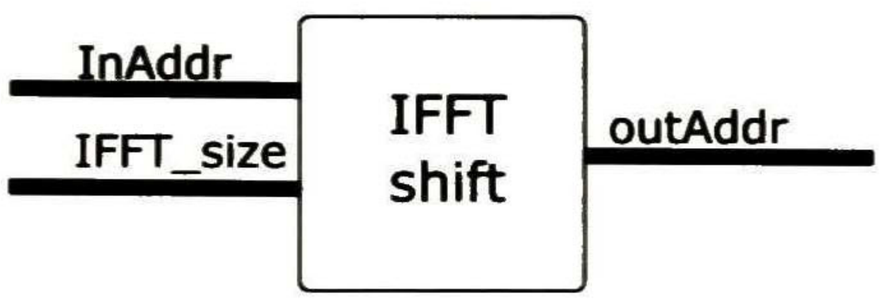


Figura 3.24: Decodificador de direcciones IFFT

Tabla 3.28: Entradas y salidas del decodificador de direcciones IFFT

Nombre	Descripción	bits
	Entradas	
InAddr	Dirección para el procesador FFT/IFFT	11
IFFT_size	Código de longitud de IFFT	3
	Salidas	
OUTAddr	Dirección redefinida para el procesador FFT/IFFT	11

Capítulo 4

Metodología de Verificación y Resultados

4.1. Metodología de verificación

La metodología empleada para verificar la funcionalidad de los módulos implementados es mostrada en la figura 4.1. En la primera parte, se utiliza Matlab para generar las señales de entrada utilizadas en las camas de prueba y en el modelo; en la segunda, el módulo es sometido a una cama de prueba dónde los resultados de interés son exportados a un archivo de salida; en la tercera, la simulación del modelo es ejecutada para obtener su respuesta y exportarla a un archivo de salida; finalmente, ambos archivos son utilizados para comparar ambas respuestas. El objetivo es verificar que ante el mismo estímulo de entrada, el módulo y su modelo proporcionen la misma respuesta.

4.2. Requerimientos de diseño

En la tabla 4.1 se muestran los requerimientos de diseño del transmisor OFDM. Estos requerimientos finalmente definen el carácter de los planes de pruebas propuestos para verificar los módulos. En las siguientes secciones de este capítulo, se describen los resultados que satisfacen estos requerimientos.

Figura 4.1: Metodología de verificación de módulos

Tabla 4.1: Requerimientos de diseño del transmisor OFDM

Núm. de req.	Descripción
R1	El transmisor debe soportar los anchos de banda de LTE: 1.25, 2.5, 5, 10 y 20 MHz.
R2	El transmisor debe soportar las tasas de transmisión establecidas para los anchos de banda del requerimiento R1: 1.92, 3.84, 7.63, 15.36 y 30.72 MHz.
R3	El reloj mínimo de sistema debe ser de 61.44 MHz.
R4	La transmisión de la trama debe ser continua. Es decir, el flujo de datos no debe ser interrumpido una vez que la trama comience a ser transmitida.
R5	El transmisor debe soportar los esquemas de modulación de datos: 4/16/64-QAM
R6	El transmisor debe modular los pilotos en BPSK
R7	El transmisor debe ser reconfigurable en términos de longitud trama, longitud de prefijo cíclico, número de portadoras nulas, cantidad y valor de los pilotos
R8	La señal transmitida debe tener un SQNR superior a 50 dB
R9	Las longitudes de IFFT deben ser de 128, 256, 512, 1024 y 2048 muestras.
R10	Con la frecuencia de reloj del requerimiento R3, el tiempo de escritura y procesamiento de la IFFT debe ser menor a 4096 ciclos de reloj.

Figura 4.2: Estructura de las camas de prueba del **procesador FFT/IFFT VL**

4.3. Procesador FFT de longitud Variable

4.3.1. Camas de prueba del Procesador FFT de longitud Variable

Se generaron camas de prueba para las 4 diferentes versiones del módulo **procesador FFT/IFFT VL**. El objetivo es determinar el tiempo de procesamiento del módulo, así como el valor de SQNR que existe entre la salida del modelo y la del módulo. En el apéndice A se describe la metodología para obtener estas camas de prueba de manera automatizada. Las acciones realizadas durante cada prueba son mostradas en la figura 4.2. El desempeño de todas las longitudes del **procesador FFT/IFFT VL** fue evaluado mediante 5 camas de prueba, una para cada longitud. Por lo tanto, se generaron 10 camas de prueba para cada versión del procesador, 5 para la FFT y 5 para la IFFT.

Durante las camas de prueba, las muestras son ingresadas proporcionando una dirección que esta en del rango de 0 a N (Longitud de IFFT). Sin embargo, el módulo decodifica estas direcciones a través de los 4 niveles y al final, en el nivel 0, aplica la operación bit reversal [1]. En consecuencia, el módulo almacena las muestras de entrada en una dirección distinta a la proporcionada, provocando un efecto de reordenamiento. Por esta razón, durante las camas de prueba no se precargó la memoria del procesador, pues esto no hubiera permitido examinar el reordenamiento. Así entonces, es de suma importancia que en cada cama se someta a prueba el ingreso de las muestras.

Lo que motivó la generación de las camas de prueba de manera automatizada fue que el número de muestras es muy grande. Por ello se elaboró un script que leyera las muestras desde un archivo, las pasara a punto fijo y las insertara dentro de la cama de prueba correspondiente. Por otro lado, las camas de prueba escriben las muestras de salida en un archivo. Este archivo se compara muestra a muestra, con la salida proporcionada por el modelo del **procesador FFT/IFFT VL** para obtener la relación señal a ruido de cuantificación (SQNR) promedio.

4.3.2. Resultados del Procesador FFT de longitud Variable

4.3.2.1. Métrica de SQNR

El SQNR existente entre la salida del módulo, en punto fijo, y la salida del modelo, en punto flotante, es reportado en la tabla 4.2. Esta tabla es representada de manera gráfica en la figura 4.3. Dónde se puede apreciar que a medida que la longitud de IFFT aumenta, el SQNR disminuye. Esto se debe a que una mayor longitud de IFFT requiere un escalamiento de igual magnitud. Dicho escalamiento es realizado a partir de escalamientos parciales. Por lo tanto, habrán más escalamientos parciales entre mayor sea el escalamiento total. En consecuencia, una mayor cantidad de escalamientos parciales producirá más pérdidas por truncamiento.

Los resultados de SQNR son idénticos en las 4 versiones, pues la diferencia entre las versiones está en la estrategia que siguen para disminuir el tiempo de procesamiento.

Tabla 4.2: SQNR obtenido en las 4 versiones del **procesador FFT/IFFT VL**

Longitud de FFT [muestras]	SQNR [dB]
128	66.5889
256	63.8945
512	60.8342
1024	57.735
2048	54.599

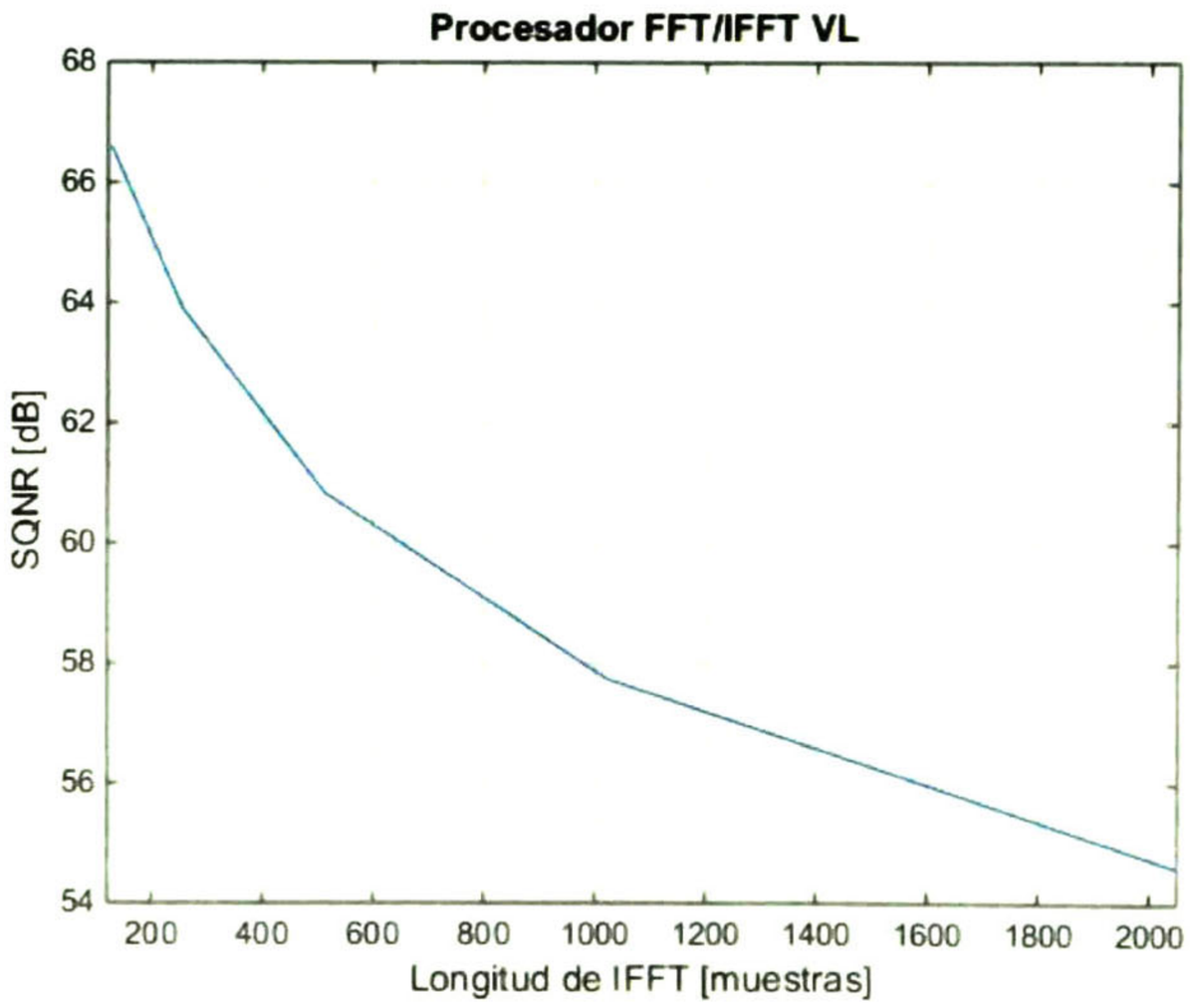


Figura 4.3: SQNR obtenido en las 4 versiones del **procesador FFT/IFFT VL**

4.3.2.2. Tiempo de procesamiento

Para el desempeño del **procesador FFT/IFFT DEBF**, se consideraron dos tiempos de procesamiento; el primero es el tiempo total de procesamiento, el cual es el tiempo empleado para procesar la FFT/IFFT desde que **start** es activada hasta que **busy_done** declara proceso realizado. El segundo es el tiempo de procesamiento parcial, el cual es más corto que el primero porque en este caso, el tiempo utilizado para realizar LBFS no se considera como tiempo de procesamiento, sino como tiempo utilizado para proporcionar el resultado.

El tiempo de procesamiento total (T_t) en el **FFT/IFFT processor** de nivel 0 es determinado como:

$$T_t(0) = s \cdot b \cdot c + \log_2(FFT\_length(0)) + 1 \tag{4.1}$$

Tabla 4.3: Tiempo de procesamiento para diferentes longitudes de FFT/IFFT

Longitud de FFT [muestras]	Tiempo de procesamiento [ciclos de reloj]				
	LV4 SEBF (ShBF)	LV4 SEBF	LV5 SEBF	LV4 DEBF T_t	LV4 DEBF T_p
128	904	904	523	904	
256	2059	1161	780	1033	904
512	2572	1674	1293	1290	1032
1024	3597	2699	2318	1803	1290
2048	5646	4748	4367	2828	1804

donde s es el número de etapas utilizadas para procesar una FFT de nivel 0, b es el número de mariposas por etapa ejecutadas en el nivel 0, c es el número de ciclos de reloj utilizados para calcular una operación mariposa, la cual tiene un valor de 2 en esta implementación, y *FFT_length* fue definida en la ecuación (3.4).

Similarmente, el tiempo de procesamiento total (T_t) para los niveles superiores está dado como:

$$T_t(l) = T_t(l-1) + 2^{l-1} \cdot b \cdot c + 1 \tag{4.2}$$

donde l es un número entero que representa el nivel y está en el rango de 1 a Lv.

El tiempo de procesamiento parcial (T_p) para el nivel 0 no es considerado porque este no es un processor de longitud variable, por lo tanto no ejecuta una LBFS utilizando dos mariposas externas. En el caso del nivel 1, (T_p) es igual a $T_t(0)$. Por otro lado, para los niveles superiores, T_p está definido como:

$$T_p(l) = T_p(l-1) + 2^{l-2} \cdot b \cdot c + 1 \tag{4.3}$$

en este caso, l va desde 2 hasta Lv.

La tabla 4.3 muestra los resultados del tiempo de procesamiento obtenidos mediante las camas de prueba: Esta tabla se muestra de manera gráfica en la figura 4.4.

Se puede apreciar que para las dos longitudes de menor tamaño, la versión de 5 niveles con mariposa externa única **LV5 SEBF** es más rápida que el resto de las implementaciones. Sin embargo, como se verá mas adelante, es la que mayor número de mariposas utiliza. Además, para el transmisor OFDM se requiere que la velocidad se enfoque a las IFFTs de mayor longitud. Por lo tanto, la versión de 5 niveles queda totalmente descartada, pues sólo supera a la versión de mariposa externa única de 4 niveles **LV4 SEBF** por 381 ciclos de reloj. Es decir, no es viable duplicar el número de mariposas siendo que la ganancia en tiempo de procesamiento es poca. El desempeño de las dos implementaciones de 4 niveles con mariposa externa única **SEBF (ShBF)** y **SEBF** es similar en términos de tiempo de procesamiento. Sin

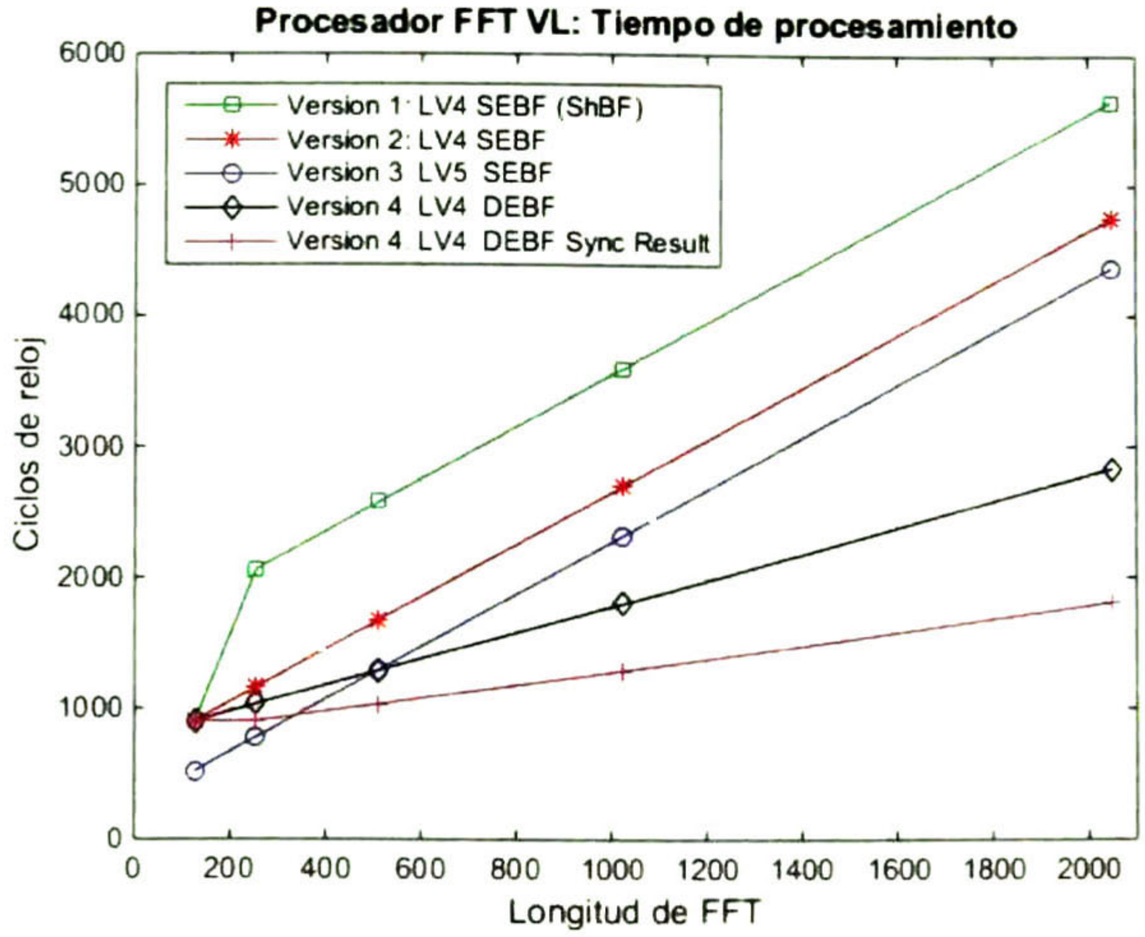


Figura 4.4: Tiempo de procesamiento de las distintas versiones del procesador FFT/IFFT implementado

embargo, ambas son descartadas porque la versión de 4 niveles con mariposa externa doble **DEBF**, las supera considerablemente durante la prueba de mayor longitud. La versión **DEBF** de cuatro niveles, tiene el balance entre consumo de mariposas y velocidad de cálculo. Es decir, en promedio, este procesador es el más rápido de las cuatro versiones y no requiere tantas mariposas como la versión de 5 niveles.

4.3.2.3. Mariposas requeridas

En la tabla 4.4 se reporta el número de mariposas requeridas por las distintas versiones del **procesador FFT/IFFT** implementado. Esta tabla es representada de manera gráfica en la figura 4.5. Se puede apreciar que la versión de 5 niveles con mariposa externa única **LV5 SEBF**, es la que más mariposas requiere. Las versiones de 4 niveles con mariposa externa única, **SEBF** y **SEBF (ShBF)**, requieren menor cantidad de mariposas. Sin embargo, su tiempo de procesamiento es relativamente elevado. Finalmente, la versión de doble mariposa externa, **DEBF**, justifica la cantidad de mariposas requeridas, al ser la versión con el menor tiempo de procesamiento para longitud de 2048 muestras.

4.3.2.4. Resultados de síntesis

La tabla 4.5 muestra los resultados de síntesis del **procesador DEBF FFT/IFFT** de 4 niveles. Se configuró el grado de velocidad con un valor de -3 y se seleccionó la opción **Timing Performance** como objetivo del diseño.

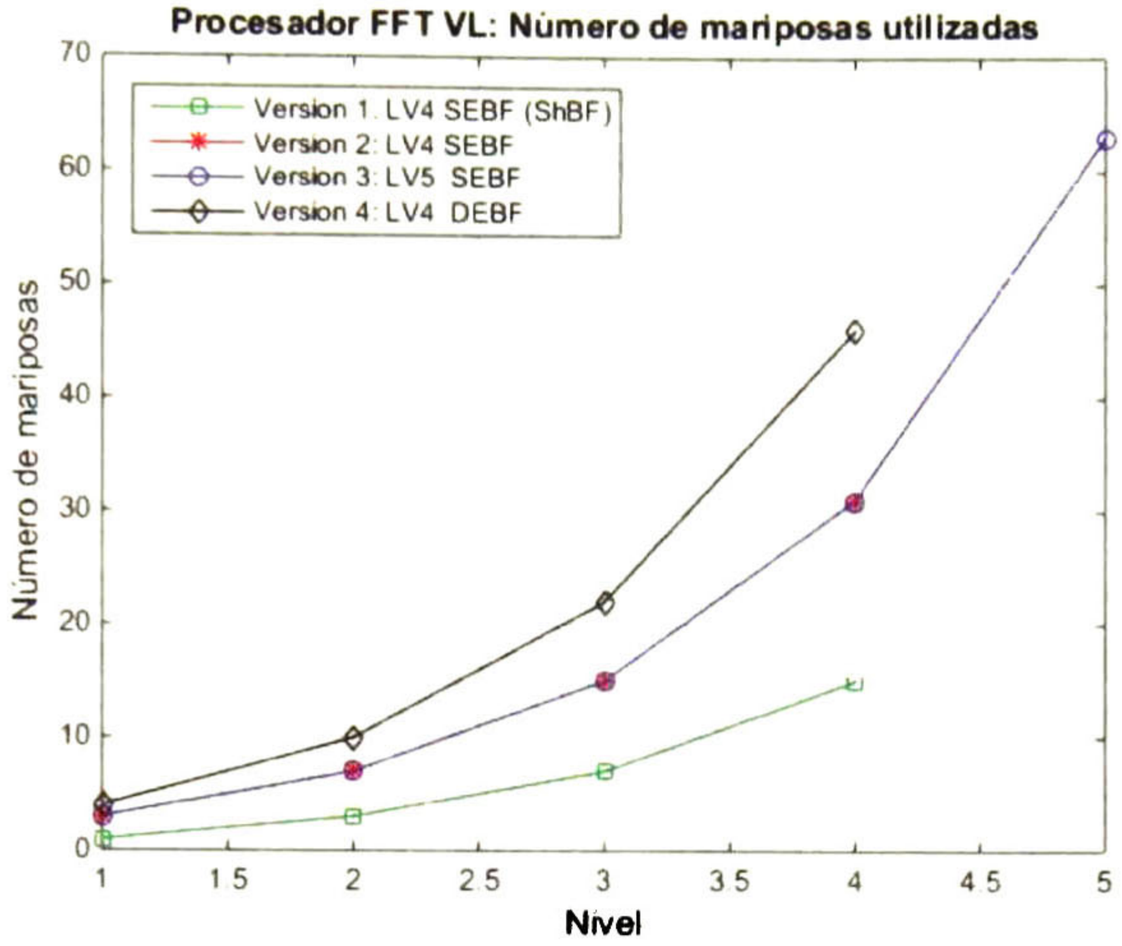


Figura 4.5: Número de mariposas utilizadas por las distintas versiones del **procesador FFT/IFFT**

Tabla 4.4: Número de mariposas requeridas por las distintas versiones del **procesador FFT/IFFT**

Nivel	Número de mariposas			
	LV4 SEBF (ShBF)	LV4 SEBF	LV5 SEBF	LV4 DEBF
1	1	3	3	4
2	3	7	7	10
3	7	15	15	22
4	15	31	31	46
5			63	

Tabla 4.5: Resultados de síntesis del **procesador FFT/IFFT**

Familia	Frecuencia máxima [MHz]	Número de registros	Número de LUTs	Número de bloques RAM
Spartan 6	34.778	6134 (127 %)	69191 (2882 %)	4 (33 %)
Virtex 6	61.115	6192 (7 %)	21235 (51 %)	32 (23 %)
Kintex7	69.855	6192 (8 %)	21269 (45 %)	32 (20 %)

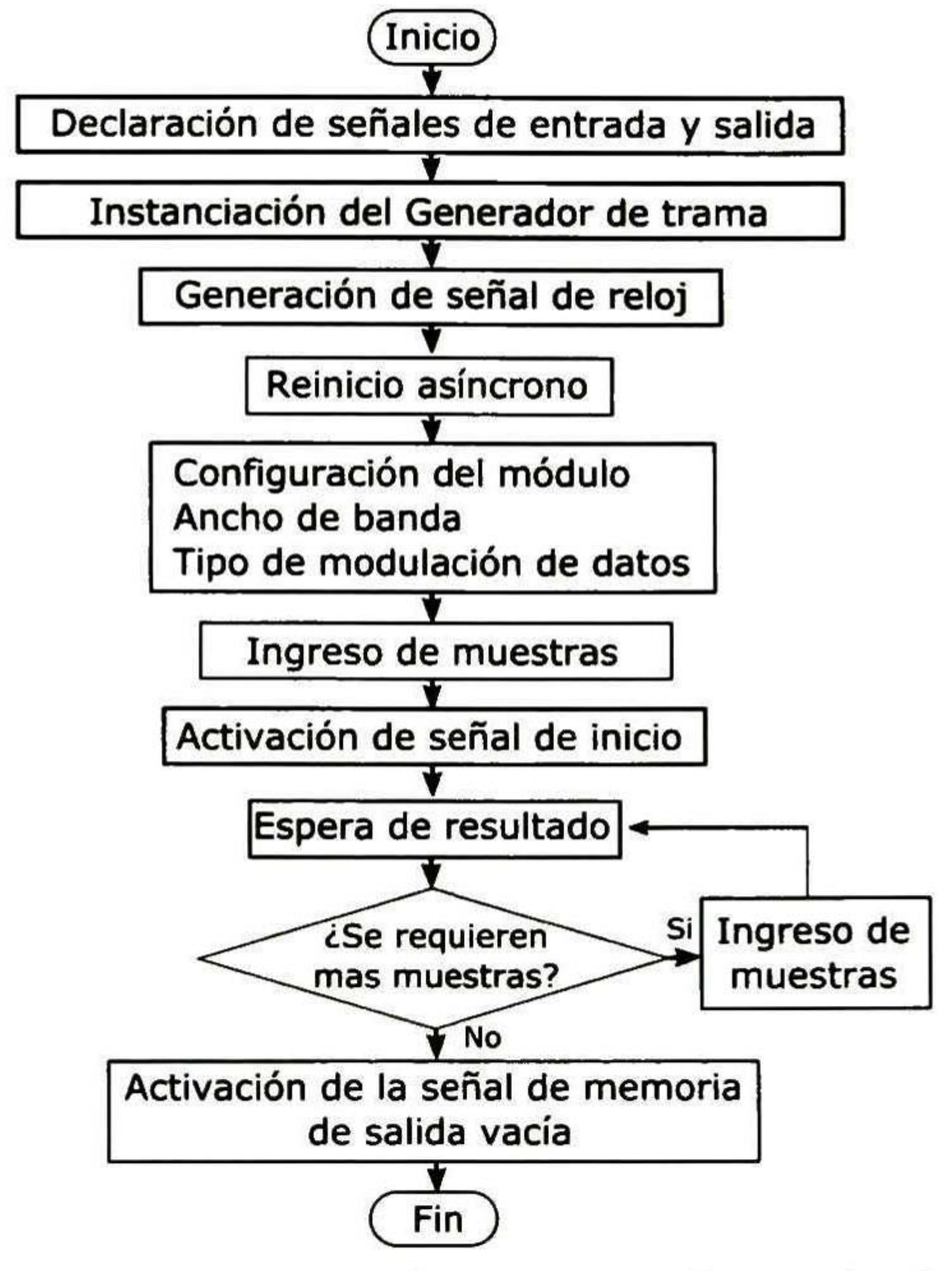


Figura 4.6: Estructura de las camas de prueba de **FC**

4.4. Conformador de trama

4.4.1. Camas de prueba del Conformador de trama

Las camas elaboradas para el conformador de trama (**FC**) permitieron probar la generación de la secuencia de ceros, datos y pilotos que posteriormente es modulada y convertida a símbolo OFDM. En cada ancho de banda se sometieron a prueba los 3 tipos de modulación de datos. Por lo tanto se elaboraron 15 camas de prueba para este módulo. En la figura 4.6 se muestra la secuencia de acciones realizadas por las camas de prueba de **FC**.

Los registros de configuración de **FC** fueron pre-cargados con los valores requeridos por el estándar LTE. Por esta razón, en la figura 4.6 no se muestra una etapa donde se configure la trama, la ranura, el número de portadoras nulas, los pilotos, ni la longitud del prefijo cíclico. Sin embargo, se requiere que los datos de entrada sean ingresados durante la cama de prueba. La trama de LTE requiere de una gran cantidad de datos. En contraste, se cuenta con una memoria de datos con profundidad limitada. Debido a esto, se decidió que fuera posible habilitar el inicio de **FC** sin necesidad de haber ingresado todos los datos requeridos para conformar la trama. De este modo, es posible ingresar más datos aunque la generación de trama esté en progreso. Esto evita el desbordamiento de la memoria de datos.

4.4.2. Plan de pruebas y resultados del Conformador de trama

4.4.2.1. Plan de pruebas

En la tabla 4.6 se muestra el plan de pruebas utilizado para verificar al conformador de trama (**FC**) mediante las camas de prueba de la sección 4.4.1.

4.4.2.2. Resultados

A continuación se presentan los resultados obtenidos de las pruebas mencionadas en la tabla 4.6. Durante la prueba 1, se configuraron 20 ranuras de tiempo y 7 símbolos OFDM dentro de cada ranura. Como resultado se observaron 140 símbolos OFDM generados a lo largo de la trama. Los resultados de la prueba 2 permitieron identificar un error en el modulador 64-QAM. Se realizó una prueba aislando a los moduladores BPSK y QAM reportados en [5]. Se encontró que todas las constelaciones son generadas correctamente, excepto la de 64-QAM. En esta constelación sólo se generaban 62 puntos. Por esta razón, el modulador 64-QAM fue modificado y corregido en este trabajo.

El resultado de la prueba 5 fue exitoso al ver que las direcciones no utilizadas para portadoras nulas fueron utilizadas para insertar datos y pilotos. También se verificó que la secuencia de datos y pilotos correspondiera con la distancia y el offset entre pilotos que se configuró. En particular, se verificó que en cada ranura de tiempo, el primer y quinto símbolo OFDM incluyeran un piloto cada 5 subportadoras. Además, configurando al primer símbolo con un *pilotOffset* de 0 y al quinto símbolo con un *pilotOffset* de 3, se verificó que las posiciones de los pilotos, en el quinto símbolo, estuvieran desfasadas 3 subportadoras respecto al primero, tal como se muestra en la cuadrícula LTE de la figura 1.8.

Debido a que el resultado de las pruebas restantes fue exitoso, se garantiza que **FC** es capaz de proporcionar la secuencia de símbolos, que al modularse, conforman al símbolo OFDM. Además de generar las dirección correspondientes de cada símbolo. Los símbolos OFDM generados, consideran las regiones de portadoras nulas, las cuales son utilizadas como bandas de guarda. Por otro lado, se comprobó que este módulo es capaz de replicar los **patrones** de generación de símbolo OFDM, mencionados en la tabla 2.3. Así, dos tipos principales de símbolo OFDM pueden generarse, los que incluyen datos y pilotos o aquellos que sólo incluyen datos. La reconfigurabilidad de este módulo permite modificar el tamaño de la trama, el tamaño de las bandas de guarda, la distancia entre los pilotos (*pilotDist*) y la posición a partir de la cual, los pilotos comienzan a incluirse (*pilotOffset*).

4.4.2.3. Resultados de síntesis

La tabla 4.7 muestra los resultados de síntesis del **`Conformador de trama`** para distintas familias de FPGAs. Se configuró el grado de velocidad con un valor de -3 y se seleccionó la opción **Timing Performance** como objetivo del diseño.

Tabla 4.6: Plan de pruebas del **`Conformador de trama`**

Núm. de prueba	Prueba	Descripción
1	Generar todos los símbolos OFDM configurados	Esta prueba consiste en verificar que al configurar N_{Slots} ranuras de tiempo y N_{Sy} símbolos por ranura, un total de N_{Slots} N_{Sy} símbolos OFDM sean generados.
2	Generar las 4 constelaciones del modulador BPSK-QAM	Verificar que los pilotos sean modulados utilizando BPSK y que los datos sean modulados mediante 4, 16 o 64-QAM, según indique la señal **dataModulation**.
3	Verificar que el número de portadoras nulas (N_{nc}) y la longitud de FFT (N) cambian en función del ancho de banda **BW**	El valor de N_{nc} debe corresponder con el valor configurado. En este caso los valores son: para **BW** $= 0$, $N = 128$, $N_{nc} = 53$; para **BW** $= 1$, $N = 256$, $N_{nc} = 106$; para **BW** $= 2$, $N = 512$, $N_{nc} = 212$; para **BW** $= 3$, $N = 1024$, $N_{nc} = 424$; y finalmente, para **BW** $= 4$, $N = 2048$, $N_{nc} = 848$.
4	Verificar la posición de las portadoras nulas dentro del símbolo OFDM	Dada la longitud de IFFT (N) y el número de portadora nulas (N_{nc}), se debe verificar que las primeras $\frac{N_{nc}}{2} - 1$ direcciones, del símbolo OFDM, tengan asociada una muestra con valor 0. Lo mismo se espera para las últimas $\frac{N_{nc}}{2}$ direcciones. La última muestra con valor cero se debe insertar en la parte media del símbolo OFDM, es decir, en la dirección $\frac{N}{2}$. El resto de las direcciones deben ser utilizadas para datos y pilotos o sólo datos.
5	Verificar la posición de los datos y pilotos dentro del símbolo OFDM	Manteniendo *pilotOffset* en 0, activar la señal *enaPilots* y verificar que una secuencia alternada de datos y pilotos sea incluida en el símbolo OFDM. La manera de conformar esta secuencia es insertando un piloto cada *pilotDist*-1 subportadoras/direcciones. El seguimiento de esta secuencia se debe realizar desde las primeras direcciones, aunque estas sean utilizadas para portadoras nulas. Esto permite determinar si se debe incluir un dato o un piloto cuando se llegue a una dirección que no está destinada para portadoras nulas. La única portadora nula que pausa el seguimiento de esta secuencia, es la muestra central.
6	Verificar la posición de los datos dentro del símbolo OFDM	Desactivar la señal *enaPilots* y verificar que datos modulados sean insertados en las direcciones que no son utilizadas para portadoras nulas. Ningún piloto debe ser incluido.
7	Verificar el patrón de los 7 símbolos OFDM dentro de cada ranura	Verificar que el símbolo OFDM 0 sea generado con el **patrón 1**; el símbolo 4 con el **patrón 2**; y los símbolos OFDM 1, 2, 3, 5 y 6 con el **patrón 3**.

Tabla 4.7: Resultados de síntesis del **`Conformador de trama`**

Familia	Frecuencia máxima [MHz]	Número de registros	Número de LUTs
Virtex 6	232.456	332 (0 %)	2492 (5 %)
Kintex7	237.254	332 (0 %)	2492 (5 %)

4.5. Transmisor OFDM

4.5.1. Camas de prueba

Se realizaron 15 camas de prueba al transmisor OFDM. Las camas de pruebas cubren las 15 combinaciones que surgen a partir de los 5 anchos de banda y los 3 tipos de modulación de datos.

En cada cama se prueba se estableció el ancho de banda **BW** y el esquema de modulación de datos **dataModulation**. Posteriormente se ingresaron las muestras de entrada y se inició el proceso con la señal **start**. En este caso, sólo se observaron las señales de interés mencionadas en el plan de pruebas de la sección 4.5.2, poniendo fin a la cama de prueba. La señal de salida no se exportó durante las camas de prueba porque se prefirió utilizar cosimulación para manejar la gran cantidad de símbolos transmitidos.

4.5.2. Plan de pruebas y resultados

En la tabla 4.8 se muestra el plan de pruebas utilizado para verificar al **`Transmisor OFDM`** mediante las camas de prueba de la sección 4.5.1.

4.5.3. Cosimulación del transmisor

Se utilizó Simulink y Matlab para realizar la cosimulación del transmisor OFDM. Para realizar esto, el estimulo de entrada al igual que la respuesta del modelo de la sección 2.2, es decir, las señales **Tx_bits** y **Tx_signal** fueron capturadas y posteriormente utilizadas en el bloque de cosimulación del transmisor OFDM. Durante la cosimulación, la señal de salida del transmisor fue comparada con la del modelo (**Tx_signal**) para obtener un histograma de SQNR y su valor promedio. Además se utilizó un analizador de espectros para ver la señal transmitida por el bloque de cosimulación.

Tabla 4.8: Plan de pruebas del **`Procesador OFDM Reconfigurable`**

Núm. de prueba	Prueba	Descripción
1	Verificar la tasa de transmisión para **BW** = 1.25 MHz	Fijar el valor de la señal **BW** en 0 y corroborar que la señal transmitida cambie cada 32 ciclos de reloj
2	Verificar la tasa de transmisión para **BW** = 2.5 MHz	Fijar el valor de la señal **BW** en 1 y corroborar que la señal transmitida cambie cada 16 ciclos de reloj
3	Verificar la tasa de transmisión para **BW** = 5 MHz	Fijar el valor de la señal **BW** en 2 y corroborar que la señal transmitida cambie cada 8 ciclos de reloj
4	Verificar la tasa de transmisión para **BW** = 10 MHz	Fijar el valor de la señal **BW** en 3 y corroborar que la señal transmitida cambie cada 4 ciclos de reloj
5	Verificar la tasa de transmisión para **BW** = 20 MHz	Fijar el valor de la señal **BW** en 4 y corroborar que la señal transmitida cambie cada 2 ciclos de reloj
6	Verificar que la bandera **full** no se active durante la trasmisión	Verificar el valor de la señal **full** en las 5 FIFO's que se encuentran dentro del *buffer* de salida. Esta señal debe permanecer en **0** durante toda la transmisión de la trama
7	Verificar que la bandera **empty** no se active durante la trasmisión	Para **BW** = 0, se debe verificar el valor de la señal **empty** en la FIFO **x**. Para el resto de valores de **BW**, se debe verificar el valor de la señal **empty** en las FIFO's **a**, **b**, **c** y **d**. Estas FIFO's se encuentran dentro del *buffer* de salida. La señal **empty** no debe valer **1** sino hasta el final de la transmisión.

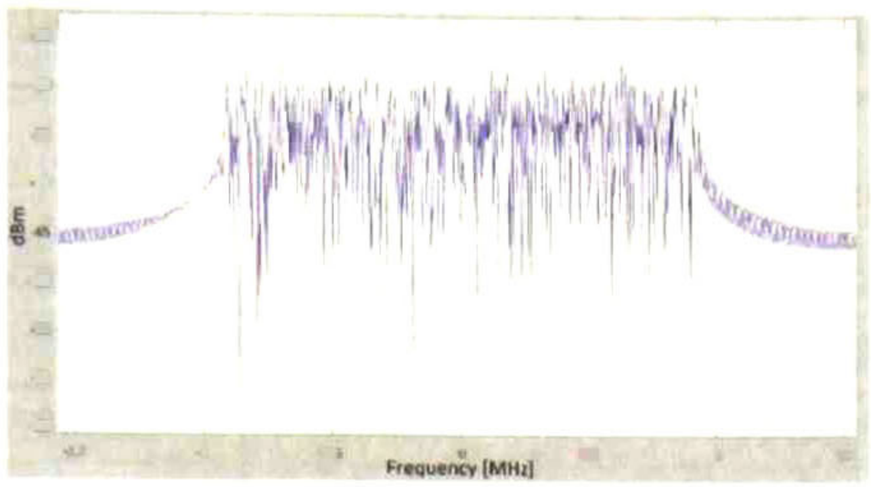


Figura 4.7: Espectro de los símbolos OFDM generados por el transmisor propuesto

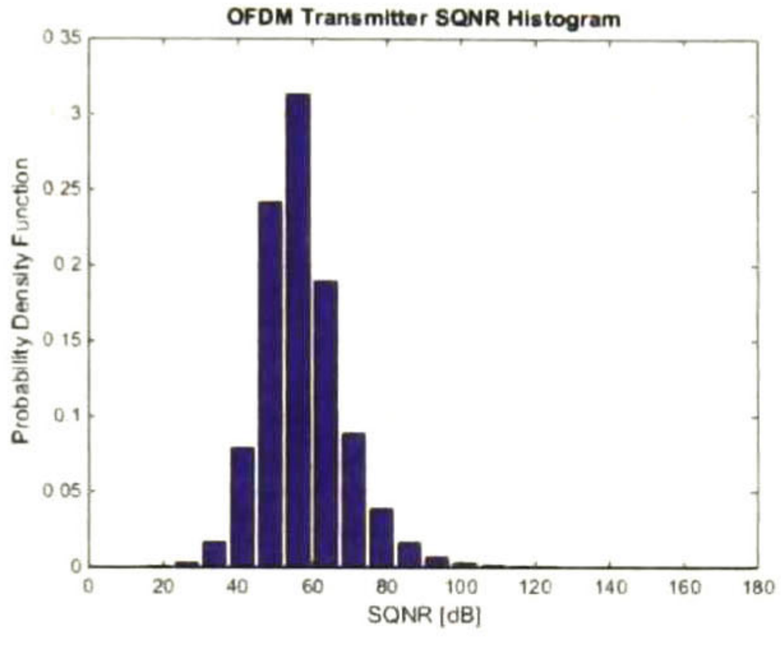


Figura 4.8: Histograma de SQNR de la señal de OFDM transmitida

4.5.4. Resultados

El **procesador FFT/IFFT** implementado pudo generar sin problema los símbolos OFDM para todas las tasas de transmisión requeridas por LTE. Las pruebas 1-5 de la tabla 4.8 permitieron verificar que las tasas de transmisión, requeridas por LTE, fueran alcanzadas por el transmisor. La prueba 6 garantiza que el *buffer* de salida no caiga en overflow y por lo tanto no haya pérdida de símbolos OFDM.

Finalmente, la prueba 7 garantiza que la transmisión de la trama no se ve interrumpida, pues el *buffer* de salida siempre cuenta con símbolos que transmitir.

La señal transmitida por el **Transmisor OFDM** fue comparada con la del modelo de la sección 2. El resultado de la comparación es mostrado en histograma de SQNR de la figura 4.8, donde se aprecia que el SQNR promedio es de 56 dB.

El espectro de la señal transmitida es mostrado en la figura 4.7. Se puede ver el ancho de banda de la señal, así como sus correspondientes bandas de guarda.

4.5.5. Resultados de síntesis

El transmisor OFDM fue implementado en una FPGA Xilinx Kintex7: XC7K70T. Los resultados de síntesis son mostrados en la tabla 4.9.

Tabla 4.9: Resultados de síntesis del transmisor OFDM propuesto.

FPGA Xilinx Kintex7: XC7K70T			
Frecuencia de operación			61.44 MHz
Recurso de la FPGA	Disponible	Utilización	Porcentaje
Número de Registros	82000	6233	7 %
Número de LUTs	41000	22240	54 %
Número de Bloques RAM/FIFO	135	24	17 %
Número de DSP48E1s	240	184	76 %

4.6. Discusión de Resultados

La frecuencia máxima a la que está limitado el procesador FFT/IFFT es de 69 MHz. Sin embargo, la tasa de transmisión más alta en LTE es de 30.72 MHz, la cual corresponde al ancho de banda de 20 MHz. En estas condiciones, la frecuencia máxima de reloj que se puede proponer es del doble de la tasa de transmisión más alta, es decir, el reloj debe ser de 61.44 MHz. La razón es porque el reloj del sistema tiene que ser un múltiplo de 30.72 MHz, pero no puede sobrepasar los 69 MHz. Cuando el ancho de banda es de 20 MHz, las señales transmitidas deben salir cada 2 ciclos de reloj, debido al reloj de 61.44 MHz. Por lo tanto, las 2048 muestras de un símbolo OFDM serán transmitidas en 4096 ciclos de reloj. Como consecuencia, la siguiente IFFT debe ser escrita y procesada en menos de 4096 ciclos de reloj. De lo contrario, el *buffer* de salida se quedará vació después de haber transmitido las primeras 2048 señales. Como el procesador FFT/IFFT SEBF se demora 2048 ciclos de reloj en escribir las muestras de entrada y 4748 ciclos en procesar, no es viable utilizarlo en la implementación del transmisor, pues sobrepasa los 4096 ciclos de reloj requeridos. Esto motivó la implementación del procesador DEBF, la cual además de reducir el tiempo de procesamiento, comienza a proporcionar el resultado antes de que el módulo termine de calcular toda la IFFT. El procesador FFT/IFFT DEBF cumple con la restricción de 4096 ciclos de reloj, pues demora 2048 ciclos en escribir las muestras de entrada y 1804 ciclos en comenzar a proporcionar el resultado. Por esta razón, el procesador DEBF fue utilizado en la implementación del transmisor.

Para el transmisor OFDM se debe considerar que debido a que el analizador de espectros promedia la señal y a que los símbolos OFDM promediados difieren unos de otros en el sentido de que algunos incluyen pilotos y otros no, el analizador de espectros presenta una señal donde no se puede hacer distinción clara entre los datos y los pilotos. Sin embargo, la métrica de SQNR garantiza que el transmisor OFDM tiene el desempeño esperado por el modelo en punto fijo de la sección 2.

Capítulo 5

Conclusiones y Trabajo a Futuro

5.1. Conclusiones

- Se logró implementar al **procesador FFT/IFFT** de longitud variable, al **Conformador de trama** y al **transmisor OFDM** acorde a los requerimientos del estándar LTE.

- Las 5 longitudes del **procesador FFT/IFFT DEBF** permiten que el **transmisor OFDM** alcance 5 de los 6 anchos de banda requeridos en LTE.

- La reconfigurabilidad del **conformador de trama** permite modificar en número de símbolos OFDM dentro de la trama, la cantidad de portadoras nulas, la longitud del prefijo cíclico y las posiciones en las que datos y pilotos son distribuidos en un símbolo OFDM. Por lo tanto, el **transmisor OFDM** no esta limitado a generar tramas de LTE únicamente, sino que modificando estos parámetros de configuración, se pueden obtener diferentes tramas.

- Si se desea utilizar la versión del **procesador FFT/IFFT SEBF** en el transmisor, entonces es necesario utilizar un *buffer* de salida con una profundidad específica y relativamente grande. El **procesador FFT/IFFT SEBF** implementado está limitado por una frecuencia máxima de 69 MHz. Una consecuencia de esto es que, por sí sólo no puede mantener la tasa de transmisión más alta de LTE (30.72 MHz). Para poder alcanzar esta tasa, es necesario acoplarle un *buffer* de salida con profundidad de 2048×8 localidades. Con esta profundidad, es posible almacenar 8 símbolos OFDM antes de iniciar la transmisión. Esto garantiza que el *buffer* pueda proporcionar los datos a transmitir sin caer en *underflow*, mientras que el **procesador FFT/IFFT SEBF** prepara el siguiente símbolo OFDM.

- Ventajas de la arquitectura multinúcleo del **procesador FFT/IFFT**.

La arquitectura multinúcleo del **procesador FFT/IFFT** es my regular, lo que permite ensamblar múltiples niveles de manera repetitiva y relativamente sencilla.

Si se desea expandir al **procesador FFT/IFFT** para obtener de una FFT/IFFT del doble de longitud, sólo se necesita agregar un nivel a la arquitectura sin necesidad de modificar internamente a los niveles inferiores.

Esta arquitectura puede entregar el resultado durante la última etapa de mariposas. Es decir, la latencia es reducida, pues no es necesario terminar con el cálculo para empezar a proporcionar el resultado.

Desventajas de la arquitectura multinúcleo del **procesador FFT/IFFT**.

La principal desventaja del **procesador FFT/IFFT** SEBF y DEBF, dentro del transmisor OFDM, es que no pueden iniciar el cálculo hasta que todas las muestras de entrada sean ingresadas. Esto incrementa la latencia que existe al formar un símbolo OFDM, pues no sólo es el tiempo necesario para calcular la FFT/IFFT, sino también el tiempo para cargar los datos de entrada al **procesador FFT/IFFT**.

Agregar un nivel a la implementación implica duplicar el número de **mariposas** del nivel previo. Por lo tanto, una mayor cantidad de niveles incrementa el consumo de área.

- Manteniendo fija la configuración de punto fijo utilizada para la implementación del **procesador FFT/IFFT** SEBF y DEBF, se concluyó que a medida que la longitud de FFT/IFFT aumenta, el SQNR disminuye. No obstante, los **procesadores FFT/IFFT** alcanzaron métricas de SQNR superiores a 54 dB y en el mejor de los casos 66.5 dB.

- Para disminuir la profundidad del *buffer* de salida, se requiere de un **procesador FFT/IFFT** más rápido. La idea de utilizar un **procesador FFT/IFFT SEBF** de 5 niveles para este propósito queda descartada, pues, la diferencia de velocidades entre las implementaciones de nivel 5 y nivel 4 es de 381 ciclos de reloj. Por lo tanto, utilizar un **procesador FFT/IFFT SEBF** de 5 niveles sólo incrementará el consumo de área sin ofrecer un incremento significativo de velocidad.

- La arquitectura multinúcleo de doble mariposa externa **DEBF** reduce considerablemente el tiempo del procesamiento de la FFT/IFFT de 2048 puntos. A diferencia de la implementación de 5 niveles de mariposa única **SEBF**; la implementación de 4 niveles con doble mariposa externa, supera por 1924 ciclos de reloj a su homóloga de mariposa única. Esto se traduce como un incremento de velocidad de aproximádamente 40 %. Por lo tanto, de los procesadores implementados en este trabajo, la versión DEBF es la más apropiada para incluirse dentro del diseño final del transmisor.

- El transmisor OFDM implementado es capaz de generar símbolos OFDM de 5 anchos de banda diferentes, soportando 3 esquemas de modulación de datos (4,16 y 6-QAM) y manteniendo una métrica de SQNR superior a 50 dB.

5.2. Trabajo a futuro

- Agregar la longitud 1536 al **procesador FFT/IFFT** para cubrir todas las longitudes de FFT requeridas por el estándar LTE. Esta longitud puede ser alcanzada mediante un algoritmo de *radix* mezclado, en particular, el algoritmo FFT *radix*-3 que combina tres FFT´s de longitud $N/3$ para producir una de longitud N [9].

- Expandir el software desarrollado para la generación automática del **procesador FFT/IFFT SEBF** para que soporte no sólo el *radix*-2, con el que ya se cuenta, sino también el algoritmo *radix*-3.

- Implementar técnicas que permitan incrementar la frecuencia máxima del **procesador FFT/IFFT SEBF**. En particular, superar las limitaciones en frecuencia que los multiplicadores generan a medida que se incrementa su ancho de palabra. Con un reloj de 120 MHz, el procesador FFT/IFFT SEBF podría mantener alimentado al *buffer* de salida sin riesgo de *underflow*.

- Implementar un procesador FFT/IFFT basado en *pipeline* para iniciar el cálculo sin necesidad de haber ingresado todas las muestras previamente. Esto permitiría reducir el tiempo de formación de símbolos OFDM y en consecuencia, se podría reducir el tamaño del *buffer* de salida. Además, se reduciría el riesgo de *underflow*.

- Diseñar e implementar la interfaz para el **Transmisor OFDM Reconfigurable**.

Apéndice A

Generación del procesador FFT/IFFT de longitud variable

A.1. Justificación

Los parámetros configurables del módulo se muestran en la tabla A.1. Cabe destacar que si se desea modificar el número máximo de muestras (N) de la FFT, no basta con modificar el valor del parámetro *log2_num_muest* en el nivel Lv, también es necesario modificar diversos archivos de diseño, entre ellos están los decodificadores de tamaño **deco_size** y las memorias **ROM_Wn** que se encuentran dentro de cada nivel del módulo FFT/IFFT. La razón es porque los factores rotatorios asociados a los 5 tamaños de FFT se encuentran almacenados en dichas ROM, de manera que, modificar N implica modificar la profundidad de las ROM y los valores almacenados en ellas. Es por ello que surge la necesidad de contar con un programa que ensamble los diferentes niveles del procesador FFT/IFFT multinúcleo de manera automatizada.

Tabla A.1: Parámetros del módulo **proc_fft_with_EBF_LV4**

Parámetros	
log2_num_muest	logaritmo base 2 del número de muestras a transformar
data_with	longitud de palabra de los datos de entrada y salida
enteros	cantidad de bits utilizados para la parte entera de los datos de entrada y salida

A.2. Introducción

Para poder generar los niveles de manera automática y obtener la implementación de una FFT de longitud variable, se debe contar con al menos los archivos de diseño del nivel 0, el cual es la base para la generación de los niveles superiores. El procesador utilizado para conformar al nivel 0, es un procesador FFT de longitud estática, el cual fue elaborado y reportado en [1]. Al desarrollar la implementación del procesador de nivel 1, se identificó la existencia de un patrón regular al conectar los elementos que lo componen, de modo que a partir de él, se desarrolló una plantilla que pudiese adaptarse a los niveles superiores. Así, se elaboraron funciones y programas en Matlab que utilizaran esta plantilla modificando parámetros específicos de los diferentes niveles. La tabla A.2 muestra las funciones necesarias para la generación de los módulos de niveles superiores. Mientras que los programas de Matlab que utilizan estas funciones para generar automáticamente todos los archivos de diseño del módulo FFT/IFFT de longitud variable se muestran en la tabla A.3.

Tabla A.2: Funciones auxiliares de Matlab relacionadas con el procesador FFT/IFFT

Descripción	Función de Matlab
FFT representada en punto fijo	FFT_FP.m
Divisor de muestras pares e impares	Feeder_Rx2.m
Binario char a numérico	binc2binn.m
Binario numérico a char	binn2binc.m
Número complejo a punto fijo hexadecimal	complex2ram.m
decimal a complemento a 2	dec2tc.m
FFT de longitud variable	fft_procc_VL.m
Rpresentación punto fijo para ROM (Wn)	fi_wn2.m
Generador de Testbench para FFT/IFFT	gen_TB_FFT.m
Generador del decodificador de tamaño	gen_deco_size_LV.m
Generador del módulo FFT completo	gen_proc_with_EBF_LV.m
Generador del módulo FFT sin BF	gen_procc_fft_wihout_BF_LV.m
Generador de memoria ROM (Wn)	genera_rom_verilog_v2.m
Hexadecimal a punto fijo	hex2fp.m
Punto fijo a hexadecimal	fp2hex.m
Error cuadrático medio	mse.m
Lector de archivo con muestras	read_samples_file.m
Lector de archivo con muestras	read_samples_file_v2.m
Complemento a 2 a decimal	tc2dec.m
Complemento a 2 en hexadecimal	tc_hex.m
Escritor de archivo com muestras	write_samples.m

Tabla A.3: Programas para la generación de los archivos de diseño del procesador FFT/IFFT

Descripción	Programa
Generador de muestras aleatorias complejas	*gen_all_random_samples.m*
Generador de archivos de diseño, archivos con las muestras en tiempo y frecuencia representadas en punto fijo hexadeciamal y camas de prueba.	*gen_fft_core.m*

A.3. Modo de empleo del programa: gen_fft_core.m

En Matlab, se debe establecer el path principal con la ruta donde se encuentra el programa *gen_fft_core.m*, además, se debe añadir al path la carpeta donde se encuentran las funciones mencionadas en la tabla A.2. Los pasos para obtener todos los archivos de diseño y simulación son los siguientes:

1. Establecer una longitud máxima N y la cantidad de niveles Lv que se desean para el procesador FFT/IFFT de nivel Lv. Las longitudes disponibles están determinadas por la ecuación (3.4).

2. Generación archivos con las muestras complejas en decimal.
   Abrir el programa *gen_all_random_samples.m* y modificar los **parámetros ajustables**: N y $cd\_niveles$, acorde al punto 1. Tras ejecutar el programa, se generarán diversos archivos nombrados como: x_t_N.txt, los cuales son los valores de las muestras en tiempo que se usarán en las diferentes camas de prueba.

3. Generación de los siguientes archivos:

   - Archivos de diseño de los niveles superiores del modulo FFT/IFFT
   - Archivo con las muestras generadas en el punto 2, representadas en punto fijo.
   - Camas de prueba

   Abrir el programa: *gen_fft_core.m* y modificar los **parámetros ajustables**: N y $cd\_niveles$. Tras ejecutar el programa se generarán 3 carpetas:

   CORE_FFT: Archivos de diseño: Memorias ROM (Wn) y niveles superiores.

   - **TB_FFT**: Muestras en tiempo y frecuencia para las *Lv+1* longitudes de FFT disponibles.
   - **Testbench**: Camas de prueba generadas a partir de los archivos de la carpeta **TB_FFT**.

De esta manera, para expandir un módulo FFT de nivel 0 al nivel superior Lv, basta con incluir el contenido de la carpeta **CORE_FFT** y **Testbench**, al proyecto en hardware y establecer como Top-level al módulo de nivel Lv.

Bibliografía

[1] Eduardo Romero Aguirre. *Arquitecturas Digitales de Procesamiento de Señales para Sistemas de Comunicación con Entrenamiento Implícito.* PhD thesis, Cinvestav Unidad Guadalajara, 2012.

[2] B. Bautista-Contreras, R. Parra-Michel, R. Carrasco-Alvarez, and E. Romero-Aguirre. A sdr architecture based on fpga for multi-standard transmitter. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pages 1266–1269, Dec 2013.

[3] LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels. Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation (3GPP TS 36.211 version 12.3.0 Release 12). Technical report, European Telecommunications Standards Institute., 2014.

[4] D. Cohen. Simplified control of fft hardware. In *IEEE Trans. Acoust., Speech Signal Processing, Vol. ASSP-24, pp. 577-579*, volume 2, pages II–833–6 Vol.2, May 1976.

[5] Benjamin Bautista Contreras. Diseño e implementación de un transmisor reconfigurable para sistemas de comunicaciones con entrenamiento explícito e implícito. Master's thesis, Cinvestav Unidad Guadalajara, 2013.

[6] C.I. Cox. *An Introduction to LTE LTE, LTE-advanced, SAE, VoLTE and 4G Mobile Communications.* 2014.

[7] E. Dahlman, S. Parkvall, and J. Skold. *4G: LTE/LTE-Advanced for Mobile Broadband: LTE/LTE-Advanced for Mobile Broadband.* Elsevier Science, 2011.

[8] International Telecommunication Union. ITU Paves Way for Next-Generation 4G Mobile Technologies. [online] Disponible en: `www.itu.int/net/pressoffice/press_releases/2010/40.aspx`, 2010.

[9] Inc. Software Optimization of DFTs Freescale Semiconductor and IDFTs Using the StarCore SC3850 DSP Core. Documento número: AN3680. [online] Disponible en: `http://www.freescale.com/files/dsp/doc/app_note/AN3680.pdf`, 2010.

[10] J. Garcia and R. Cumplido. On the design of an FPGA-based OFDM modulator for IEEE 802.11a, 2005. Electrical and Electronics Engineering, 2005 2nd International Conference on.

[11] Chung-Ping Hung, Sau-Gee Chen, and Sau-Gee Chen. Design of an efficient variable-length fft processor. In *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, volume 2, pages II–833–6 Vol.2, May 2004.

[12] Y.-T. Lin, P.-Y. Tsai, and T.-D. Chiueh. Low-power variable-length fast fourier transform processor. *Computers and Digital Techniques, IEE Proceedings* -, 152(4):499–506, July 2005.

[13] Qingwang Lu, Xin'an Wang, and JiuChong Niu. A low-power variable-length FFT processor base on radix-24 algorithm. In *Microelectronics & Electronics, 2009. PrimeAsia 2009. Asia Pacific Conference on Postgraduate Research in*, pages 129–132, 2009.

[14] R.G. Lyons. *Understanding Digital Signal Processing*. Prentice-Hall accounting series. Pearson Education, 2010.

[15] Y. Ma. An effective memory addressing scheme for fft processors. In *IEEE Trans. on Signal Processing, Vol. 47 Issue: 3, pp. 907-911*, volume 2, pages II–833–6 Vol.2, May 1999.

[16] Y. Ma and L. Wanhammar. A hardware efficient control of memory addressing for high-performance fft processors. In *IEEE Trans. on Signal Processing, Vol. 48 Issue: 3, pp. 917-921*, volume 2, pages II–833–6 Vol.2, May 2000.

[17] J.G. Proakis and D.G. Manolakis. *Digital Signal Processing*. Prentice Hall international editions. Pearson Prentice Hall, 2007.

[18] Ericsson AB. June 2014 Ericsson Mobility Report. [online] Disponible en: `http://www.ericsson.com/res/docs/2014/ericsson-mobility-report-june-2014.pdf`, 2014.

[19] D. Revanna, O. Anjum, M. Cucchi, R. Airoldi, and J. Nurmi. A scalable FFT processor architecture for OFDM based communication systems. In *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIII), 2013 International Conference on*, pages 19–27, 2013.

[20] E. Romero-Aguirre, R. Parra-Michel, R. Carrasco-Alvarez, and A.G. Orozco-Lugo. Architecture based on array processors for data-dependent superimposed training channel estimation. In *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*, pages 303–308, 2011.

[21] S.W.A. Shah, H.S. Okleh, A.S. Hussaini, R.A. Abd-Alhameed, J.M. Noras, and J. Rodriguez. Fpga implementation of modern wireless communication system. In *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2012 IEEE 17th International Workshop on*, pages 388–392, Sept 2012.

[22] S.W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing.* California Technical Pub., 1997.

[23] Inc. The MathWorks. ifftshift. [online] Disponible en: `http://www.mathworks.com/help/matlab/ref/ifftshift.html`, 2015.

[24] K.-J. Yang, S.-H. Tsai, and G.C.H. Chuang. MDC FFT/IFFT Processor with Variable Length for MIMO-OFDM Systems. 21(4):720–731, 2013.

[25] H. Zarrinkoub. *Understanding LTE with MATLAB: From Mathematical Modeling to Simulation and Prototyping.* Wiley Desktop Editions. Wiley, 2014.

[26] Beiling Zhang and Xuan Guo. A novel reconfigurable architecture for generic OFDM modulator based on FPGA. In *Advanced Communication Technology (ICACT), 2014 16th International Conference on*, pages 851–854, 2014.

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

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

Diseño e implementación digital de un transmisor reconfigurable para sistemas OFDM

del (la) C.

LEONARDO OROZCO GALVAN

el día 20 de Agosto de 2015.

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

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

Dr. Roberto Carrasco Álvarez
Profesor Investigador
Universidad de Guadalajara

CINVESTAV - IPN
Biblioteca Central
SSIT0013361