UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS
VERITATE SOLA NOBIS IMPONETUR VIRILIS TOGA. 1948

FACULTAD DE INGENIERÍA ELÉCTRICA

Departamento de Electrónica y Telecomunicaciones

TRABAJO DE DIPLOMA

GENERACIÓN DE CÓDIGO HDL PARA LA RADIO DEFINIDA POR SOFTWARE UTILIZANDO MATLAB

Autor: Yaime Fernández Jiménez

Tutor: Ing. Yakdiel Rodríguez-Gallo Guerra

Santa Clara

2013

"Año 55 de la Revolución"

Universidad Central “Marta Abreu” de Las Villas
Facultad de Ingeniería Eléctrica
Departamento de Telecomunicaciones y Electrónica

Generación de código HDL para la Radio Definida por Software utilizando Matlab

Tesis presentada en opción al Título Académico

Autor: Yaime Fernández Jiménez

E-mail: fjimenez@uclv.edu.cu

Tutor: Ing. Yakdiel Rodríguez-Gallo Guerra

E-mail: yrodriguez-gallo@uclv.edu.cu

Santa Clara

2013

“Año 55 de la Revolución”

Hago constar que el presente trabajo de diploma fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de estudios de la especialidad de Ingeniería en Telecomunicaciones y Electrónica, autorizando a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicado sin autorización de la Universidad.

_____

Firma del Autor

Los abajo firmantes certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____

Firma del Tutor

_____ _____

Firma del Jefe de Departamento

Firma del Responsable de Información Científico-Técnica

PENSAMIENTO

Pueblo que se resigna a tecnologías pasadas, sucumbe en el campo de la ignorancia y se entierra en sus ideales.

DEDICATORIA

A mi madre, con quien comparto este sueño y hoy ve coronado el esfuerzo y la dedicación de toda una vida.

A mi papá, por ser tan especial, por apoyarme en todo momento y creer en mí.

A mis abuelitos y hermano, siempre presentes.

AGRADECIMIENTOS

A mis padres y hermano, por ser la inspiración de cada día.

A mi tutor el Ing.Yakdiel Rodríguez-Gallo Guerra, por confiarme la responsabilidad de trabajar en este proyecto.

A toda mi familia que siempre estuvo brindándome apoyo y cariño en sus repetidas muestras de preocupación.

Al claustro de profesores, que durante estos cinco años se han esforzado para hacer de mí, una profesional.

A esos amigos incondicionales y a todos los que me han ayudado en este trabajo, en especial a Nieves.

En fin, a todos los que han sabido enseñarme que la paciencia y el esfuerzo son el camino para lograr verdaderos objetivos y han contribuido a materializar este sueño.

A todos ustedes, gracias.

TAREA TÉCNICA

Para confeccionar el presente trabajo y alcanzar los resultados esperados, fue necesario elaborar las tareas técnicas siguientes:

- Caracterización de la Radio Definida por Software para conocer sus particularidades.
- Identificación de dispositivos de lógica programable que se utilizan para el desarrollo de sistemas en SDR.
- Caracterización de los lenguajes de descripción de hardware y de las herramientas de simulación Matlab y Xilinx ISE.
- Presentación de los pasos a seguir para la generación de código HDL desde la herramienta de simulación Matlab 2011a.
- Confección de aplicaciones de la Radio Definida por Software, para la obtención de código HDL utilizando la herramienta de simulación Matlab 20011a.
- Comprobación del funcionamiento de las aplicaciones desarrolladas e implementación en un FPGA de Xilinx de un circuito compuesto por un generador de secuencia, un codificador convolucional y un decodificador de Viterbi.
- Elaboración del informe final del Trabajo de Diploma.

Firma del Autor

Firma del Tutor

RESUMEN

La Radio Definida por Software (SDR), ha revolucionado las telecomunicaciones en los últimos años. El desarrollo de este concepto junto al aumento vertiginoso de las capacidades de cómputo, permiten que el procesamiento digital de señales se convierta en el núcleo de los equipos de radio actuales. La presente investigación se dedica a la obtención de códigos HDL de aplicaciones de la Radio Definida por Software desarrolladas en Matlab, y a la implementación en el Kit de desarrollo Nexys2 que contiene un FPGA Spartan-3E de Xilinx, de un generador de secuencia, un codificador convolucional y un decodificador de Viterbi. Para ello, se caracterizaron los lenguajes de descripción de hardware, las herramientas de simulación Matlab y Xilinx ISE; y se identificaron los principales campos en que se utiliza la SDR en el mundo. Además, se presentaron los pasos para generar código HDL con la herramienta Simulink HDL Coder de Matlab, evidenciándose como resultado de la investigación, la importancia de este software, de amplia versatilidad, que es capaz de generar eficientemente códigos HDL.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Muchos sistemas de comunicación inalámbricos existen, siendo extensamente usados para propósitos y escenarios diferentes. Dentro de los estándares internacionales actualmente utilizados se encuentran las redes de área personal (PAN), las redes de área locales (LAN), las redes de área metropolitanas (MAN) y las redes de áreas extendidas (WAN)(IEEE Xplore, 2012). En los últimos años la tendencia que existe es que estas tecnologías puedan coexistir simultáneamente en una misma región, lo cual es posible si los dispositivos terminales están preparados para esto (Chen & Prasad, 2009).

Tradicionalmente, el diseño de los equipos de radiocomunicaciones ha sido eminentemente analógico. Filtros, osciladores, moduladores, amplificadores, mezcladores y demás elementos se han diseñado empleando un gran número de componentes electrónicos. Estos equipos definen todas sus funciones mediante su hardware, lo que hace muy rígida su explotación; por lo que a principios de 1990 Joseph Mitola introduce el término de la Radio Definida por Software (SDR) para definir una tecnología de radiocomunicaciones cuya funcionalidad se encuentra especificada en el software, lo cual minimiza la necesidad de realizar modificaciones de hardware durante actualizaciones tecnológicas(Chen & Prasad, 2009).

La implementación de los sistemas de comunicación reconfigurables puede ser realizada a través de componentes programables como son los Arreglos de Compuertas Lógicas Programables (FPGA). Los FPGA permiten alcanzar niveles de velocidad de procesamiento altos debido a su arquitectura interna y a su capacidad de procesar señales (Xilinx, 2012).

Algunas compañías fabricantes de FPGA desarrollan herramientas que facilitan la implementación de sistemas complejos en sus dispositivos. Un ejemplo de ello lo constituye Xilinx con el desarrollo del Xilinx ISE.

El Xilinx ISE (Integrated Software Environment) es una herramienta de diseño profesional de circuitos, que permite realizar un diseño completo basado en lógica programable.

El software Matlab, constituye una herramienta útil para el desarrollo de aplicaciones en distintas ramas de la ciencia. Esta herramienta es usada en universidades y por investigadores debido a las disímiles prestaciones que ofrece, las cuales se han ido incrementando en los últimos años, constituyendo una de ellas, la posibilidad de generar código HDL (Cepero, 2012).

En Cuba las investigaciones realizadas sobre SDR son escasas, principalmente por ser una tecnología emergente en el mercado (Algora, 2011). En la Facultad de Ingeniería Eléctrica de la Universidad Central "Marta Abreu" de Las Villas es uno de los temas en los que se está incursionando actualmente. Se han hecho trabajos anteriores donde se muestra la confección de aplicaciones en SDR, haciéndose uso de las herramientas de simulación System Generator y Xilinx ISE, y no utilizándose las nuevas prestaciones que en las últimas versiones posee el software Matlab para la generación de código HDL, las cuales podrían ser útiles para el desarrollo en el futuro de estos sistemas de las comunicaciones, de ahí la importancia que tiene la realización de este trabajo.

Teniendo en cuenta lo anterior, en el presente trabajo de diploma se plantea el siguiente problema de investigación: ¿Cómo generar código HDL desde la herramienta de simulación Matlab para desarrollar aplicaciones de la Radio Definida por Software?

La investigación tiene como **objeto de estudio** el software Matlab para la obtención de código HDL y el **campo de acción** lo constituye la utilización de esta herramienta de simulación para el desarrollo e implementación de sistemas de la Radio Definida por Software.

Para dar respuesta al problema de investigación se propone como **objetivo general**: Obtener en la herramienta de simulación Matlab código HDL de aplicaciones desarrolladas de la Radio Definida por Software.

Partiendo del objetivo general se derivan los siguientes objetivos específicos:

- Caracterizar la Radio Definida por Software para conocer sus particularidades.
- Caracterizar los Lenguajes de Descripción de Hardware (HDL) para conocer su utilización en el diseño de aplicaciones de la SDR.
- Identificar dispositivos de lógica programable que se utilizan para el desarrollo de sistemas en SDR.
- Caracterizar la herramienta de simulación Matlab para conocer su utilización en la obtención de código HDL.
- Caracterizar la herramienta de simulación Xilinx ISE para la implementación de aplicaciones en un FPGA de Xilinx.
- Presentar los pasos que se deben efectuar para generar código HDL desde la herramienta de simulación Matlab en su versión 2011a.
- Desarrollar aplicaciones de SDR en Matlab para generar su código HDL.
- Comprobar el funcionamiento de las aplicaciones de SDR desarrolladas en Matlab.

De los objetivos específicos surgen las siguientes interrogantes científicas:

- ¿Qué características posee la Radio Definida por Software, así como los dispositivos que se utilizan en su desarrollo?
- ¿Cuáles son los beneficios que proporcionan los HDL para el diseño de aplicaciones de la SDR?
- ¿Cuáles son las prestaciones que brinda el software Matlab para la generación de código HDL?
- ¿Qué utilidad tiene el software Xilinx ISE para la implementación de aplicaciones en FPGA de Xilinx?

- ¿Qué hacer para determinar los pasos a seguir en la generación de código HDL en Matlab?

- ¿Qué aplicaciones de SDR desarrollar en Matlab y qué hacer para comprobar su funcionamiento?

Organización del informe

Para satisfacer los objetivos planteados el trabajo se dividió en: introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos

El primer capítulo aborda las características teóricas de la Radio Definida por Software, las principales aplicaciones en las que se utiliza esta tecnología en el mundo e implementaciones realizadas en dispositivos lógicos programables. Además se describe brevemente el Kit de desarrollo Nexys2.

En el segundo capítulo se muestran los beneficios de los lenguajes de descripción de hardware en el diseño de circuitos digitales, y se ofrece una introducción a los softwares Matlab y Xilinx ISE. Además se presentan los pasos para obtener códigos VHDL y Verilog utilizando el software Matlab (R2011a).

Por último, en el tercer capítulo, se desarrollan dos aplicaciones de la SDR con el software Matlab para obtener código HDL. Se realiza una introducción donde aparecen aspectos como el principio de funcionamiento, y se describen como se implementaron los subsistemas fundamentales de cada diseño. Se comprueba el funcionamiento de las aplicaciones desarrolladas mediante simulaciones y se muestra el funcionamiento del codificador convolucional, el decodificador de Viterbi y el generador de secuencia en el Kit Nexys2.

CAPÍTULO 1. CARACTERÍSTICAS DE LA RADIO DEFINIDA POR SOFTWARE Y DE LOS DISPOSITIVOS LÓGICOS PROGRAMABLES

En este capítulo se abordan los aspectos teóricos esenciales de la SDR tales como concepto, características, ventajas y desventajas de su uso, estructura y aplicaciones prácticas. A continuación se muestran algunas implementaciones de la Radio Definida por Software en dispositivos de lógica programable. Al final se describe el Kit de desarrollo Nexys2.

1.1 Caracterización de la Radio Definida por Software

Los equipos transmisores y receptores de radiocomunicaciones, habitualmente constituidos por hardware, han ido perfeccionándose en los últimos años. En la década de los ochenta y noventa se introdujeron microprocesadores en ellos para el control de sus funciones, y seguidamente la posibilidad de controlarlos desde un ordenador, añadiéndoseles puertos de comunicación o interfaces para la conexión. En la década de los noventa aparecen por primera vez en ellos los chips DSP (Procesadores Digitales de Señal), los cuales permitieron, mediante técnicas digitales realizar filtros, y actualmente son usados en las etapas de FI (Frecuencia Intermedia) de los receptores (Fette, 2009).

En 1991 Joseph Mitola comenzó a investigar y desarrollar un nuevo concepto para los equipos de radiocomunicaciones: la radio definida por software o “radios software” (SR). En 1992, en la "National Telesystem Conference", con el nombre de "Software Radio: Survey, Critical Evaluation and Future Directions", sin realizarse aún la construcción de estos equipos, expone sus ideas, en las cuales se refiere a los SR como un tipo de radio reprogramable o reconfigurable donde un mismo elemento de hardware es capaz de realizar

diferentes funciones, en distintos instantes de tiempo, con la introducción de cambios en su configuración mediante software (Aguilar & Navarro, 2011).

En un sistema SDR ideal (Figura 1.1) la digitalización de la señal se realiza desde la antena; sin embargo, el estado actual de la tecnología y la complejidad de las programaciones hace que no se pueda implementar (Chen et al., 2010).

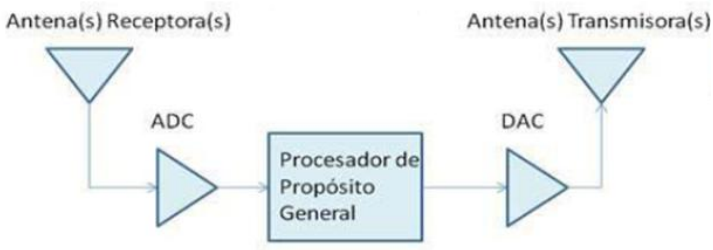


Figura 1.1Diagrama de un sistema SDR Ideal(Romero, 2012).

En la Figura 1.2 se muestra un diagrama general de los bloques fundamentales que conforman un sistema SDR actualmente (Romero, 2012). En él, las cadenas de transmisión y recepción han sido divididas en tres secciones básicas: sección de RF, sección de A/D y sección de software.

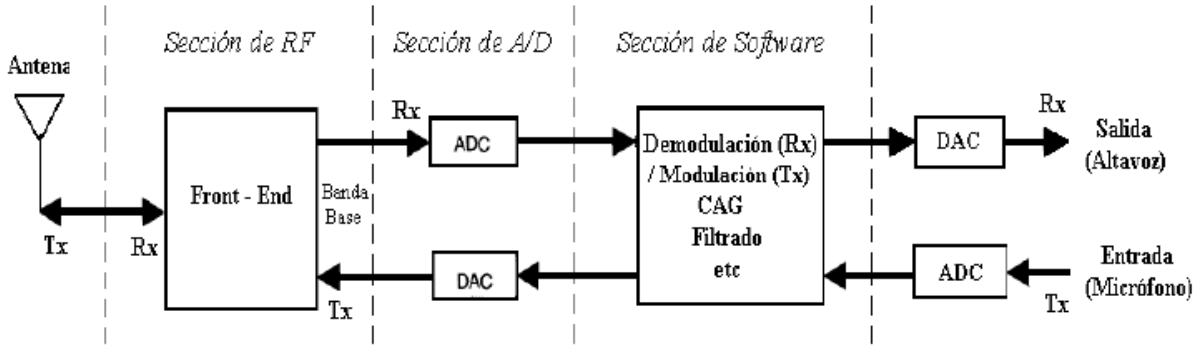


Figura 1.2Diagrama general de un sistema SDR (Romero, 2012).

La sección de RF (también llamada “Front-End”) es la responsable de acondicionar las señales de radiofrecuencia, para que puedan ser digitalizadas por el bloque que le sigue, debido a que en la mayoría de los casos, los ADC (Analog to Digital Converter) y DAC (Digital to Analog Converter) no son lo suficientemente potentes para digitalizar a la frecuencia que se desea trabajar. Las funciones básicas de esta sección son: la conversión

de frecuencia, la selección del canal, el rechazo a interferencias y la amplificación (LOUIS, 2011).

La sección de conversión análoga digital/digital análoga (ADC y DAC) es la encargada de digitalizar la señal a la frecuencia intermedia proveniente de la salida del bloque de RF. En un transceptor SDR los convertidores permiten que todo el procesamiento de la señal sea completamente digital en la etapa de banda base (García, 2012).

La sección de software es considerada por algunos estudiosos del tema como el núcleo de la tecnología SDR, puesto que en ella se ejecuta un programa que maneja los datos digitalizados; realizando sobre las señales funciones como: modulación/demodulación, filtrado de ancho de banda, reducción del ruido, ajuste del control automático de ganancia, entre otras. En esta sección es donde se aprecian las principales ventajas de la tecnología SDR, pues con solo actualizar o sustituir el software, pueden cambiarse las funcionalidades del equipo transceptor, transformándolo para la recepción de señales con características diferentes a las utilizadas anteriormente (Romero, 2012).

1.1.1 Definiciones de la SDR

Varias son las definiciones expuestas sobre SDR por los investigadores. El Foro de SDR, en colaboración con IEEE (Institute of Electrical and Electronic Engineers), la define como la "Radio en la que todas o algunas de las funciones de la capa física están definidas por software" (García, 2012). Por su parte la FCC (Federal Communications Commission) de los Estados Unidos, como organismo regulador de las comunicaciones de ese país, en la página tres de su Definición Regulatoria del 14 de Septiembre del 2001 expresa lo siguiente:

> "…aquellos radios que incluyen un transmisor cuyos parámetros de operación, como banda de frecuencias, tipo de modulación o potencia máxima de salida pueden ser alteradas mediante cambios en software sin realizar cambios a los componentes de hardware que se relacionan con la emisión de radiofrecuencias son denominados SDR" (Romero, 2012).

De acuerdo a Lee Pucker, colaborador de Wireless Innovation Forum, la SDR es una tecnología aplicable en un amplio rango de áreas dentro la industria inalámbrica, capaz de proveer soluciones eficientes y comparativamente baratas a muchos problemas inherentes a arquitecturas de radio tradicionales. En esencia, es un término utilizado para describir una tecnología de radio donde algunas o todas las funciones de la capa física inalámbrica son definidas por software (Algora, 2011).

Para algunos fabricantes de equipos de radio e integradores de sistemas, la SDR establece una familia de productos de radio usando una plataforma con arquitectura común, que permite introducir nuevos productos en el mercado de una forma rápida, trayendo como consecuencia una reducción de los costos de desarrollo, operación y mantenimiento (Mohebbi et al., 2003).

1.1.2 Principales características de la SDR

Las principales características de la técnica SDR son (LOUIS, 2011):

- Reconfigurabilidad: la SDR permite la existencia de múltiples módulos de software juntos, empleando diferentes estándares en el mismo sistema, lo que permite la configuración dinámica, solo a través de la selección apropiada del módulo de software, para poder funcionar.
- Conectividad Ubicua: una de los principales retos para los sistemas de telecomunicaciones es conseguir una conectividad ubicua, que es la capacidad que tiene el usuario para acceder transparentemente a cualquier servicio, en cualquier lugar y momento. Lo que hace la tecnología SDR es facilitar la realización de sistemas de radio de arquitectura abierta.
- Multifuncionalidad: es la capacidad de reconfiguración de un SDR para soportar una variedad grande de servicios en un sistema.
- Facilidad de Actualización: durante el despliegue de una red, los servicios requieren actualizarse o en algunos casos deben introducirse nuevos servicios. Tales cambios se deben realizar sin la interrupción de la operación de la infraestructura existente.

- Movilidad Mundial: la necesidad de transparencia, así como la capacidad de los radios para operar con algunos o preferentemente con todos los estándares en diferentes regiones geográficas del mundo, ha influenciado el crecimiento del concepto de Radio Definido por Software.
- Comparación y Eficiencia de uso de Potencia: SDR ofrece una propuesta de diseño compacto, lo que permite un manejo eficiente de la potencia, especialmente cuando el número de sistemas se incrementa.

1.1.3 Principales ventajas y desventajas de la SDR

Entre las ventajas de la SDR se pueden mencionar (LOUIS, 2011):

- Versatilidad para el desarrollo de diferentes sistemas de transmisión de datos y estándares, debido a que la misma plataforma de hardware podría ser utilizada por cualquier equipo de comunicación, no importa el ancho de banda o el modo de emisión/recepción.
- Sistema Compacto. En la figura 1.2 se aprecia como la sección de software agrupa varias funciones (control de ganancia, modulación, demodulación, etc.).
- Bajo costo de implementación, al brindar un sistema de radiocomunicaciones con menores costos de desarrollo, operación y mantenimiento. El número de componentes electrónicos a utilizar se ve notablemente reducido (pero no la complejidad del sistema SDR), al utilizar software en los productos de radio. ‘’Over-the-air’’ u otra reprogramación remota, permite correcciones de errores que ocurren mientras la radio está en servicio, por lo tanto reduce el tiempo y los costos asociados con la operación y el mantenimiento.

Para los proveedores de servicios de radio, SDR permite:

- Nuevas características y capacidades que se adicionarán a la infraestructura existente sin el requerimiento de gastos significativos.
- El uso de una plataforma de radio común para múltiples mercados, reduce significativamente el apoyo logístico y gastos operativos.

Uno de los inconvenientes de la tecnología SDR es la complejidad de los software, realizados por la comunidad científica internacional, con funcionalidades de equipos como, receptores GPS, receptores de TV Digital, transmisores de ondas medias, radio bases para sistemas de telefonía móvil, entre otros. La dificultad en el trabajo con este tipo de aplicaciones es que, como es evidente, necesita un soporte de hardware muy superior al de una tarjeta de sonido y una PC.

1.2 Aplicaciones de la Radio Definida por Software

En los últimos años el SDR Fórum, grupo independiente formado por la industria, científicos, ingenieros y organismos reguladores, ha sido uno de los principales desarrolladores de la tecnología SDR. Ejemplo de ello es el proyecto presentado en el Foro de Innovación Inalámbrica (Wireless Innovation Forum), el 29 de noviembre de 2012 por Clark Papa (Chen & Prasad, 2009).

El trabajo de Clark Papa llamado "Razor: Arquitectura avanzada del tamaño de un pulgar, de la Radio Definida por Software", presenta un nuevo SDR, pequeño, del tamaño de un pulgar, compuesto por un módulo de procesamiento. El módulo de procesamiento contiene un DSP, 512 MB de RAM, millones de compuertas lógicas y una interfaz USB. La cadena del receptor consiste en un filtro de preselección de cerámica, oscilador local, filtro, convertidor reductor integrado y ADC. Con el proyecto se pueden lograr anchos de banda de hasta 40 MHz y el rendimiento es superior a las implementaciones de conversión directa convencionales. El dispositivo se puede conectar al puerto USB de la PC o funcionar de manera independiente (SDR Forum, 2013).

Actualmente, uno de los campo de aplicación para la SDR es la cuarta generación de tecnologías de telefonía móvil (4G), la cual está basada en el protocolo IP (Protocol Internet). En ella convergen las redes de cables e inalámbricas, y es capaz de proveer velocidades de acceso mayores a 100 Mbit/s en movimiento y 1 Gbit/s en reposo, manteniendo la calidad del servicio (QoS) punto a punto (Clermidy et al., 2009).

El WWRF (Wireless World Research Forum) pretende convertir a 4G en una fusión de tecnologías y protocolos, no sólo un único estándar similar a 3G (tercera generación de

tecnologías de telefonía móvil), que actualmente incluye tecnologías como GSM (Groupe Special Mobile) y CDMA (Code Division Multiple Access) (SDR Forum, 2013).

Por su parte, el ITU (Unión Internacional de Telecomunicaciones) declaró en 2010 que tecnologías de 3G evolucionadas, como lo son WiMax (Worldwide Interoperability for Microwave Access) y LTE (Long Term Evolution), pueden ser consideradas como tecnologías 4G (SDR Forum, 2013).

Otra de las aplicaciones de SDR es la Radio Cognitivo (CR). Como sucesor de la Radio Definida por Software es un sistema de comunicaciones que está al tanto de las condiciones de su estado interno y del ambiente, tales como localización y utilización del espectro de RF. Estos sistemas pueden tomar decisiones sobre su comportamiento comparando esta información con objetivos predefinidos (Chen & Prasad, 2009).

De las iniciativas para el desarrollo de CR a nivel internacional la que más se destaca por su alcance es el proyecto Reconfiguración Extremo-a-Extremo (E2R) de la Unión Europea. El mismo se enfoca en idear, diseñar y validar soluciones de arquitectura, y respaldar algoritmos, protocolos y mecanismos en redes cognitivas, donde la noción de radio cognitivo incluye facilidades de software y radio cognitivo enriquecidas por capacidades de autogestión (Kempf et al., 2006).

Otra aplicación de la tecnología SDR es el diseño de una interfaz de radio basada en OFDM (Orthogonal Frequency Division Multiplexing) con antenas MIMO (Multiple input multiple ouput).Este tipo de antenas aprovecha fenómenos físicos de las señales tales como la propagación multicamino, para incrementar considerablemente la tasa de transferencia y disminuir la tasa de error. La interfaz puede estar contenida en una red de próxima generación (NGN) mediante IMS (IP Mutimedia Subsystem), y como comparte los mismos objetivos que 4G, ambas redes podrían trabajar en conjunto, implantadas en paralelo con un servicio multimedia a través de IP, proporcionando mayor nivel de escalabilidad y con posibilidades de evolución (Vargas et al., 2007).

La aplicación de SDR en sectores como la industria automotriz, utiliza las computadoras y las tecnologías de las comunicaciones para mejorar las experiencias de viaje en los automóviles. Las autopistas del mundo cada vez son más congestionadas, y añadir nuevas

vías para aliviar este fenómeno se vuelve una opción cada día menos atractiva. La telemática ofrece buenas soluciones con la aplicación e integración de geo-localización, comunicaciones y tecnologías de la información dentro de la arquitectura de los vehículos y la infraestructura de las carreteras y autopistas. Esta integración provee conveniencia, seguridad y capacidad de tráfico incrementada sin construcciones nuevas. Un líder en servicios de este tipo en Norteamérica y China es la subsidiaria de General Motors On-Star, una compañía dedicada a ofrecer servicios de comunicaciones, seguridad, conversación manos libres, navegación, y ayuda remota (García, 2012).

Un estándar ya establecido en el mercado a nivel internacional con el propósito de entregar servicios de comunicaciones, seguridad, conversación manos libres, navegación, y ayuda remota es FlexRay, sistema de bus serie digital escalable y tolerante a fallas, diseñado para utilizarse en aplicaciones automotrices. Algunos modelos de autos que utilizan FlexRay son: Audi A8, Bentley Mulsanne, BMW X5, Serie BMW 7, Serie BMW 5 Gran Turismo, Serie BMW 5, y Rolls-Royce Ghost.

1.2.1 Utilización de los dispositivos de lógica programable en la Radio Definida por Software

Las investigaciones sobre SDR se enfocan en la construcción de dispositivos de comunicación reconfigurables, que permitan con solo cambios en el software, influir en el funcionamiento del sistema, por ejemplo, en un sistema al reemplazar el software este puede transformarse en un receptor listo para operar con señales de diferente modulación o con diferentes códigos de corrección de errores, además de permitir su adaptación a los estándares (LAN, WAN) que surgen ante las cambiantes demandas del mercado. Por lo que estos desarrollos coexisten en paralelo con el avance de los dispositivos lógicos programables.

Los dispositivos lógicos programables o PLD (Programmable Logical Device), son circuitos integrados digitales que no tienen una función predefinida por el fabricante, por lo que pueden ser programados por el usuario. Ellos permiten reemplazar grandes diseños digitales que antes se implementaban con componentes discretos como compuertas y flip-

flops, por un solo integrado. Debido a la capacidad lógica que tienen estos dispositivos, sistemas completos pueden desarrollarse sobre un solo circuito integrado (Vallejo & Rodrigo, 2004).

Los dispositivos CPLD (Complex Programmable Logic Device) y FPGA (Arreglo de Compuertas Lógicas Programables) tienen una capacidad lógica de hasta millones de compuertas, incluyendo interfaces programables para varios estándares de interfaces eléctricas. Además, tienen bloques de funciones especiales embebidos entre la lógica programable tales como memoria, multiplicadores o CPU (Unidad Central de Procesamiento) (Bozich, 2005).

Una FPGA es un dispositivo multinivel programable de propósito general. Integra una gran cantidad de dispositivos lógicos programables en un chip. Al utilizar bloques de lógica pre-construidos y recursos de ruteo programables, se pueden configurar para implementar funcionalidades de hardware personalizadas sin tener que utilizar un tablero o un cautín (Cepero, 2012).

Una de las tecnologías SDR desarrolladas es la plataforma GNU, creada a partir del código Pspectra, desarrollado por el MIT (Massachusetts Institute of Technology) en el proyecto Spectrum Ware. Actualmente GNU Radio está programado en C++ y Python, software libre creado en 1998 por Eric Blossom, y está compuesta por un conjunto de herramientas para el procesado de la señal y el control del hardware (Robles, 2011).

GNU Radio ha sido el creador de USRP (Universal Software Radio Peripheral) (Figura 1.3), desarrollado por Matt Ettus. USRP es un sistema de adquisición y generación digital de señales que permite manejar el hardware mediante sentencias. El software GNU Radio se ejecuta principalmente sobre el sistema operativo Linux (Robles, 2011).

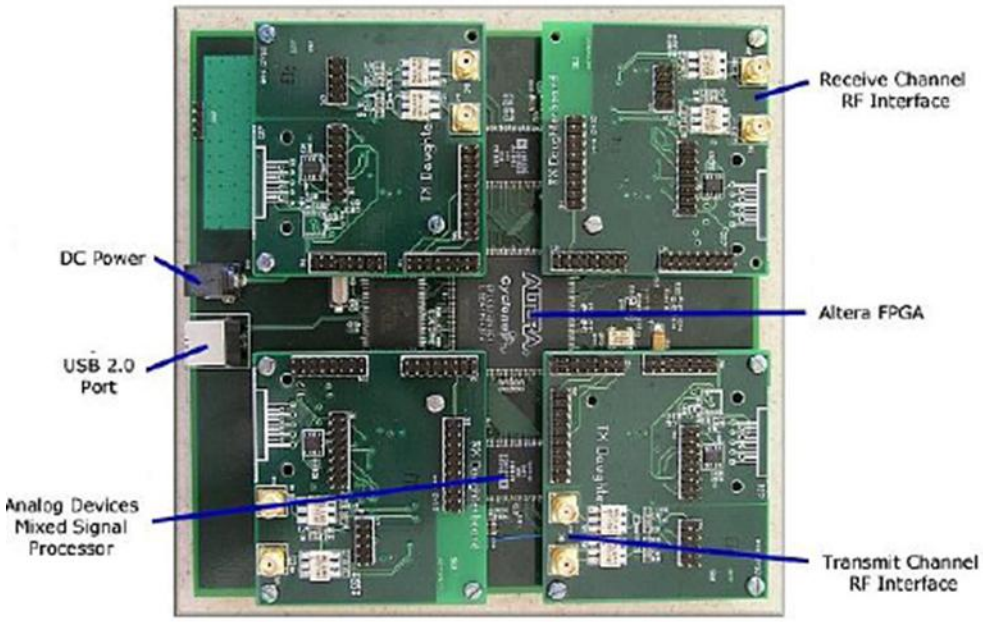


Figura 1.3USRP creado por GNU Radio(Robles, 2011).

Dentro de la USRP, se encuentran varios circuitos integrados, entre ellos cuatro convertidores análogos-digitales de 12 bits y 64Mbits/s, cuatro digitales-análogos de 14 bits y 128Mbits/s, filtros, amplificadores, procesadores e incluye una interfaz USB 2.0 de alta velocidad. No obstante, una de las unidades funcionales importante es el FPGA Cyclone II de la compañía Altera. Este dispositivo de lógica programable es el encargado de realizar los algoritmos de procesamiento digital de señales y sirve como nexo entre las etapas de banda base y banda intermedia. El uso del FPGA es una de las características innovadoras de esta implementación (Robles, 2011).

La combinación de software libre y hardware flexible generada por GNU Radio y el USRP permite a los ingenieros el diseño, desarrollo e implementación de múltiples sistemas de radiocomunicaciones a bajo costo (Blossom, 2004).

Otra tecnología SDR es el sistema SFF SDR DP (Small Form Factor Software Defined Radio Development Platformes), desarrollado por la compañía canadiense Lyrtech. La plataforma está concebida para el diseño y desarrollo de aplicaciones de radio e incorpora módulos de las empresas Texas Instruments y Xilinx.

Figura 1.4 Plataforma de desarrollo SFF SDRDP (Valverde, 2010).

Está compuesta de tres módulos como se puede ver en la Figura 1.5. Ellos son:

• Módulo de Procesado Digital (Digital Processing Module).

• Módulo de Conversión de Datos (Data Conversion Module).

• Módulo de Radiofrecuencia (RF Module).

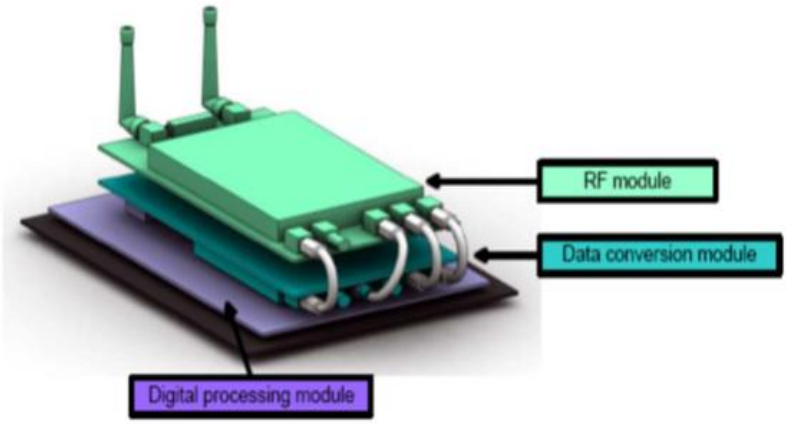


Figura 1.5 Módulos de la Plataforma de Desarrollo(Belanger, 2007).

El Módulo de Procesado Digital está compuesto por dos dispositivos, un DSP y un FPGA. En ambos se puede llevar a cabo la implementación del procesado de señal. El dispositivo DSP es el modelo TMS320DM6446 de Texas Instruments. A este tipo de dispositivos se les conoce como “Systemon Chip” (SoC) porque toda la funcionalidad del dispositivo se programa directamente sobre el mismo chip. El chip está compuesto de un procesador ARM9 de propósito general (GPP), que trabaja a una frecuencia de 594 MHz, y un

procesador TMS320C64x que actúa de núcleo, operando a 297MHz. El DSP también contiene algunas memorias como 64 registros de 32 bits, memoria RAM para datos y programas, y acceso a memoria externa. El FPGA es del tipo Virtex-4 XC4SV35 proporcionada por Xilinx y contiene96x40 bloques lógicos configurables (CLB). Sirve como coprocesador y es quien controla todas las interfaces de entrada/salida de la plataforma. Se puede llevar a cabo una comunicación entre el DSP y el FPGA a través de un protocolo proveniente de la modificación del protocolo VPSS (Video Processing Subsystem protocol), el cual consiste en la interfaz VPFE (Video Processing Front End) usada para transmitir datos de 16 bits de resolución a 75 MHz del FPGA a la entrada del DSP, y en la interfaz VPBE (Video Processing Back End) que se encarga de los datos de salida del DSP y de entrada al FPGA, con la misma resolución y a 75MHz (Belanger, 2007).

El Módulo de Conversión de Datos tiene los conversores análogo-digital (ADC) y digital-analógico (DAC). El DAC es el modelo DAC5687 de Texas Instruments que tiene dos canales que pueden operar simultáneamente con una resolución de 16 bits. El ADC es del tipo ADS5500 también de Texas Instruments, y puede muestrear a una tasa de 125 Mega-muestras/s, con una resolución de hasta 14 bits. En este módulo además hay otro FPGA, en este caso un modelo Virtex-4 XC4VLX25 de Xilinx, que se encarga de ajustar los valores de configuración del conversor de datos. Este FPGA no tiene el propósito de ser reprogramado y únicamente se pueden acceder a los parámetros de configuración a través del DSP del módulo de procesado digital. Se puede seleccionar el reloj de trabajo del conversor de datos, pudiendo utilizar una fuente externa, el reloj de referencia o un VCO (Voltage Controlled Oscillator). Además este módulo, usa un PLL (Phase Locked Loops) para mantener constante la fase y la frecuencia(Valverde, 2010).

El Módulo de Radiofrecuencia está dividido entre la parte del transmisor y la del receptor.

1.3 Principales características del Kit de desarrollo Nexys2

La tarjeta Nexys2 es una plataforma de desarrollo para circuitos, basada en el FPGA Spartan3E de Xilinx. Posee un puerto USB 2.0 de alta velocidad, 16 MB de memoria RAM (Random Access Memory) y ROM (Read Only Memory), varios dispositivos de

entrada/salida y puertos para el desarrollo de sistemas digitales. El puerto USB 2.0 provee de energía a la tarjeta y de una interfaz para la programación, así la tarjeta Nexys2 puede ser utilizada para crear una estación de diseño portable. Los diseños desarrollados en esta tarjeta pueden crecer fácilmente, utilizando los cinco conectores de expansión que posee. En la Figura 1.6 se muestra el diagrama en bloques de esta tarjeta (Cepero, 2012).

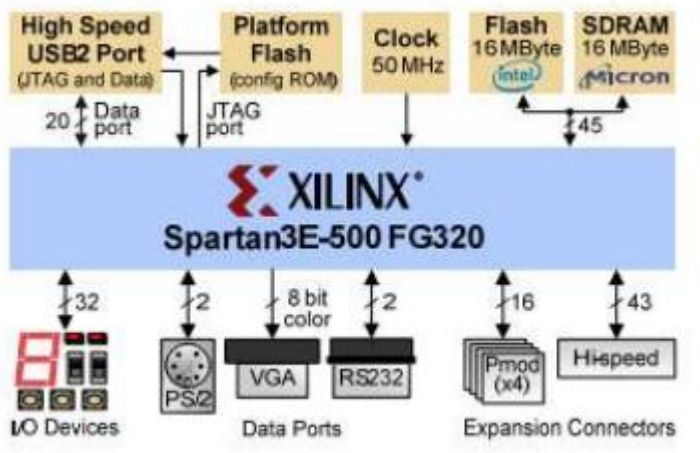


Figura 1.6 Diagrama en bloques del kit Nexys2 (Digilent, 2008).

El Kit proporciona cuatro conectores de 6 pines dispuestos en dos filas que juntos pueden acomodar hasta ocho módulos periféricos Pmod (Peripheral Module). Estos cuatro conectores, de 12 pines en total cada uno, tienen ocho señales de datos, dos pines de tierra, y dos de Vdd respectivamente. En la Figura 1.12 puede observarse el circuito de estos conectores. En el Anexo I se muestra una tabla con la asignación de pines de estos conectores.

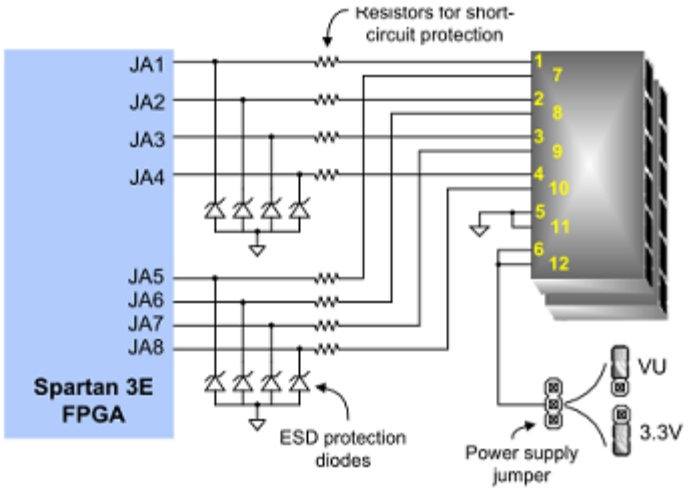


Figura1.12 Circuito de los conectores Pmod (Digilent, 2008).

Todas las señales accesibles por el usuario en la tarjeta son protegidas contra cortocircuitos, asegurando una larga vida de operación en cualquier ambiente (Digilent, 2008).La tarjeta Nexys2 es completamente compatible con todas las versiones de las herramientas de Xilinx ISE. Para más información consultar: Digilent 2008.

1.4 Conclusiones Parciales

En este capítulo se ofreció un acercamiento a los sistemas SDR, al resumirse sus características, funcionamiento, aplicaciones, ventajas, desventajas e implementaciones. La tecnología SDR por las características mencionadas en el capítulo, constituye la alternativa para resolver los problemas relacionados con la incompatibilidad e interoperabilidad entre los diferentes estándares de comunicaciones inalámbricos; y permitir la evolución o la coexistencia de los sistemas móviles sobre la misma infraestructura existente, de manera paralela con el desarrollo de los dispositivos de lógica programable. Además se evidenció la capacidad del Kit Nexys2, como plataforma para el desarrollo de circuitos.

CAPÍTULO 2. LAS HERRAMIENTAS DE SIMULACIÓN EN LA OBTENCIÓN DE CÓDIGOS HDL

Con el creciente desarrollo de la tecnología, la tendencia actual es a lograr sistemas compactos, como lo es la Radio Definida por Software, y contar con lenguajes de descripción de hardware (HDL) es importante, para su modelado y síntesis. El presente capítulo da una introducción a los HDL, especificando sus variantes, beneficios e inconvenientes para el diseño de circuitos digitales. También se caracterizan los software Xilinx ISE y Matlab, y al final se exponen pasos para generar códigos VHDL y Verilog, desde el programa Matlab 2011a.

2.1 Caracterización de los códigos HDL

Los lenguajes de descripción de hardware fueron desarrollados para hacer frente a la creciente complejidad de los diseños electrónicos. Estos códigos modelan la arquitectura y comportamiento de circuitos digitales, y mediante herramientas de software, como Xilinx ISE, estos modelos se sintetizan, para su implementación. Al utilizar HDL para crear un sistema, se está modelando hardware, y no escribiendo software. El software se caracteriza por ser secuencial, es decir, los efectos de una instrucción dependen exclusivamente de los efectos de las instrucciones anteriores. En el hardware, sin embargo, hay muchas tareas que suceden de manera concurrente, muchos parámetros y señales cambian al mismo tiempo, por lo que la variable tiempo predomina (Zamora, 2010).

Los HDL facilitan el diseño de circuitos digitales ya que, entre otros beneficios, proporcionan:

Independencia de la tecnología: El mismo modelo puede ser sintetizado en librerías de distintos vendedores (Chu, 2006).

Soportan tres estilos de descripción básicos: Descripción behavioral, descripción del flujo de datos (data-flow) y descripción estructural. Un diseño puede describirse como una combinación de los tres estilos (Zamora, 2010).

- Se puede verificar la funcionalidad del modelo en el proceso de diseño: La simulación a alto nivel, antes de la implementación a nivel de compuertas, permite verificar la arquitectura y rectificar decisiones en las primeras fases de diseño, con un esfuerzo mucho menor que si se realizase en fases posteriores (Zamora, 2010).
- Sencillez: Como la descripción se centra en la funcionalidad, resulta sencillo comprender qué función realiza el diseño a partir de una descripción HDL que a partir de un esquemático de interconexión de compuertas (Zamora, 2010).
- Ahorro de tiempo: Facilita las correcciones en el diseño debidas a fallos o cambio de especificaciones (Chu, 2006).
- La propia descripción en el lenguaje de alto nivel sirve como especificación del comportamiento del sistema a diseñar (tanto a nivel funcional, como con las restricciones temporales), y de interfaz con el resto del sistema: Los modelos descritos con estos lenguajes, pueden ser verificados fácilmente y de forma precisa por simuladores, definidos en base a estos HDL (Chu, 2006).
- Los "Test Bench" pueden ser escritos en el mismo lenguaje que con el que han sido modelados los diseños (HDL): Esto permite un mejor manejo del modelo, porque se puede asociar el modelo a sus estímulos de simulación. Los "Test Bench" no limitan el uso de dichos estímulos a un determinado simulador, pudiendo ser reutilizados dichos estímulos aunque se use un simulador distinto. Los retardos de propagación y limitaciones temporales, pueden ser descritos con estos HDL (Chu, 2006).
- El lenguaje soporta jerarquía: Un sistema digital puede ser modelado como un conjunto de componentes interconectados. A su vez cada componente puede ser modelado como un conjunto de subcomponentes (Chu, 2006).
- Soporta modelos de tiempos síncronos y asíncronos(Barrios, 2013).
- Posibilidad de implementar distintas técnicas de modelado digital: Descripciones de máquinas de estados finitos (FSM), descripciones algorítmicas y ecuaciones Booleanas.

- No hay limitaciones impuestas por el lenguaje en el tamaño del diseño: Los HDL tienen elementos que permiten el diseño a gran escala, de forma fácil; por ejemplo, componentes, funciones y procedimientos (Chu, 2006).

Actualmente existen varias alternativas de lenguajes de descripción de hardware como son: ABEL (Advanced Boolean Expression Language), AHDL (Altera Hardware Description Language), VHDL (Very High Speed Integrated Circuit Hardware Description Language) y Verilog, siendo estas dos últimas las que destacan en el mundo del desarrollo del hardware digital.

EL lenguaje VHDL nace como un proyecto del Departamento de Defensa (DoD) de EE.UU para disponer de una herramienta estándar, independiente para la especificación (modelado y/o descripción) y documentación de los sistemas electrónicos. El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) lo adopta y estandariza (Gilman, 1986).

El VHDL cuenta con la posibilidad de definir nuevos tipos de datos, facilitando la descripción de circuitos con diversos niveles de abstracción. Posee sentencias de control de flujos como if, for y while. También tiene la capacidad de estructurar el código en subprogramas, funciones o procedimientos, lo que permite afrontar algoritmos complejos. Utiliza y desarrolla bibliotecas de diseño, es decir, incorpora conceptos específicos para el modelado del hardware, como concurrencia y ciclo de simulación (Chu, 2006).

Un proyecto de VHDL puede contener muchos ficheros. El código VHDL usualmente se encuentra en los ficheros con extensión *.vhd (SÁNCHEZ, 2010), con una estructura típica formada por:

- Paquetes: Constantes, tipos de datos, componentes y subprogramas utilizados en varios diseños o entidades.
- Entidades: Interfaces de los componentes.
- Arquitecturas: Implementación de las entidades.

Verilog soporta el diseño, prueba e implementación de circuitos analógicos y digitales, a diferentes niveles de abstracción. Un diseño con este lenguaje de descripción de hardware, consiste de una jerarquía de módulos. Internamente un módulo contiene una lista de cables y registros. Las sentencias concurrentes y secuenciales definen el comportamiento del

módulo, describiendo las relaciones entre los puertos, cables y registros. La sintaxis de este lenguaje es muy parecida a la del lenguaje de programación C (Zamora, 2010).

2.2 Herramientas utilizadas para generar códigos HDL

En ocasiones los diseñadores necesitan herramientas de simulación que permitan no solamente realizar la descripción, sino evaluar la viabilidad y el desempeño de los sistemas implementados, con rapidez y eficiencia. Con este propósito se han creado herramientas de simulación y modelado de alto nivel, como son: Quartus II, System Generator, Xilinx ISE, Matlab, entre otros. A continuación se describen las prestaciones de los dos últimos software mencionados, utilizados en el desarrollo del trabajo.

2.2.1 Caracterización de la herramienta de simulación Matlab

Matlab es el nombre abreviado de "MATrixLABoratory". Fue creado por Cleve Moler en los años setenta del siglo pasado y distribuido por Math Works, desde 1984. Matlab cuenta con los siguientes usos: simular, modelar, crear prototipos, analizar datos y encontrar soluciones a sistemas complejos. Además, cuenta con un toolbox de Matemática Simbólica que soporta: cálculo, simplificaciones y sustituciones, variables de precisión, álgebra lineal, ecuaciones diferenciales ordinarias y lógica booleana. También permite el estudio de sistemas continuos, discretos, lineales y no lineales, mediante descripción interna y externa, en el dominio temporal y frecuencial. Este programa es usado para realizar cálculos numéricos con vectores y matrices; y cuenta con la capacidad de realizar una amplia variedad de gráficos en dos y tres dimensiones. Además tiene un lenguaje de programación propio (Cepero, 2012).

El software Matlab se acompaña de Simulink, el cual proporciona una interfaz gráfica de usuario para construir los modelos como diagramas de bloques. El comportamiento de dichos sistemas se define mediante funciones de transferencia, operaciones matemáticas, elementos de Matlab y señales predefinidas. Después de definir un modelo se puede simular e interactuar con dicha simulación a través de los diferentes menús que ofrece el programa. Además, es posible reconfigurar los parámetros y ver de forma inmediata lo que sucede tras el cambio; pudiendo transferir los resultados de la simulación al espacio de trabajo del programa Matlab, para su posterior procesamiento y visualización. Incluye una

amplia biblioteca de fuentes y herramientas de visualización que lo hacen útil para el diseño de sistemas. Esto supone un cambio radical respecto a otras herramientas de simulación, que requieren una formulación de las ecuaciones en forma de lenguaje o programa (Marrero, 2011).

En el desarrollo de este trabajo, se utiliza Matlab 2011a. Esta versión destaca por incluir herramientas como Matlab Coder, para generar códigos C y C++, Phased Array System Toolbox, para el diseño y simulación de arreglos de fase en Sistemas de Procesamiento de Señales, y Simulink, que cuenta con bloques, en diferentes bibliotecas, capaces de generar código HDL (A pesar de que existen varios tipos de lenguajes HDL, se debe aclarar que Matlab solo es capaz de generar VHDL y Verilog. Además algunas funciones empleadas presentan problemas de compatibilidad con versiones anteriores de este software).

Para más información se sugiere consultar la ayuda del Matlab o ir al sitio de Mathwork (MathWorks, 2013).

2.2.2 Caracterización de la herramienta de simulación Xilinx ISE

El paquete informático de Xilinx ISE (Integrated Software Environment) está formado por un conjunto de herramientas que permiten diseñar, simular y sintetizar circuitos digitales sobre dispositivos lógicos programables de Xilinx. Para llevar un diseño hasta la implementación en el dispositivo programable se siguen una serie de pasos:

- Descripción: Se realiza mediante esquemáticos, diagramas de estado o lenguajes de descripción de hardware.
- Simulación funcional: Antes de realizar la simulación, es necesario generar un banco de pruebas o test bench, con las entradas con que se va a estimular el diseño para comprobar que funciona. Es decir, la simulación sólo va a realizarse para el conjunto de estímulos que se coloquen en el test bench.
- Síntesis: El proceso de síntesis se lleva a cabo mediante Xilinx Synthesis Technology XST, el sintetizador nativo de Xilinx, que genera un fichero de extensión .ngc.
- Implementación: La fase de implementación consiste en la creación del fichero de configuración que una vez cargado en el dispositivo programable, le hará trabajar conforme a las especificaciones de nuestro diseño. Consta de dos partes: la creación de

una base de datos o netlist que recopile tanto el circuito diseñado como sus restricciones, esta es llamada fase de traducción donde se crea un archivo .ngd y la creación del fichero de programación propiamente, de extensión .bit (Rodríguez, 2011).

- Simulación Temporal: Realizar una simulación temporal del diseño, muestra el comportamiento del mismo en el FPGA que se va a utilizar, ya que esta simulación tiene en cuenta los retardos que sufren las señales dentro de dicho dispositivo. Una vez comprobado el correcto funcionamiento del mismo entonces se procede a generar fichero de programación (.bit); el cual se descargará al FPGA.
- Programación: Para descargar el fichero de configuración en el FPGA, ISE utiliza la herramienta IMpact, que requiere un cable que comunica el puerto paralelo de la PC con el Kit Nexys2. Otra vía para descargar el fichero es utilizar el software Digilent Adept, que permite esta comunicación vía USB. En fin, este es un paso sencillo donde una vez que se tiene el fichero de extensión .bit se selecciona dónde se va a cargar dicha configuración directamente, en el FPGA y/o en la memoria PROM del Kit de desarrollo, con el objetivo que dicha configuración no se pierda y al conectarle la alimentación a la tarjeta, esta se cargue directamente desde la memoria PROM si el Kit está configurado de esa manera (Zamora, 2010).

2.3 Pasos para generar código HDL desde la herramienta de simulación Matlab en su versión 2011a

La herramienta Simulink HDL Coder de Matlab genera códigos Verilog y VHDL para la síntesis en dispositivos lógicos programables de hardware digital. A continuación se presentan pasos, para el trabajo con dicha herramienta, utilizando como ejemplo un Filtro FIR Simétrico.

2.3.1 Generacion de código VHDL

Los pasos para utilizar la herramienta Simulink HDL Coder de Matlab comienzan con la inicialización del software. Una vez familiarizado con el entorno de Matlab, se accede al esquemático del filtro y se define un directorio de trabajo, utilizando los comandos:

```
>>mkdir C:\work\sl_hdlcoder_work

>>cd C:\work\sl_hdlcoder_work
```

>>open_system('sfir_fixed')

La función mkdir crea una carpeta con el nombre y la dirección especificada en la sentencia, es decir, crea la subcarpeta sl_hdlcoder_work, de la carpeta work, ubicada en el disco local C, para guardar una copia del modelo utilizado y los códigos generados. La localización de la carpeta no importa, sólo que no debe estar dentro del software. El comando cd permite acceder desde Matlab, a la carpeta creada, que debe permanecer abierta durante el proceso de generación del código, al igual que el modelo, que para abrir utiliza la función open_system. El modelo debe ser salvado, con extensión .mdl en la carpeta de trabajo, para inicializar sus parámetros mediante el comando:

>>hdlsetup ('sfir_fixed')

Cuando se invoca hdlsetup ('sfir_fixed'), se cambian los parámetros del modelo especificado en el argumento de la sentencia, en este ejemplo del filtro, a valores de Matlab que por defecto son útiles, pero pueden no ser apropiados para todas las aplicaciones, por lo que se sugiere consultar en la ayuda del Matlab, en la sección Model and Blocks Parameters de Simulink, la tabla Model Parameters.

Otra vía para la inicialización de parámetros, es acceder a la ventana Configuration Parameters, desde el botón Simulation ubicado en la barra de menú del modelo abierto y configurarlos manualmente. En la Figura 2.1 se observa la ventana Configuration Parameters, con las configuraciones del parámetro solver, predefinidas por Matlab para el Filtro FIR Simétrico.

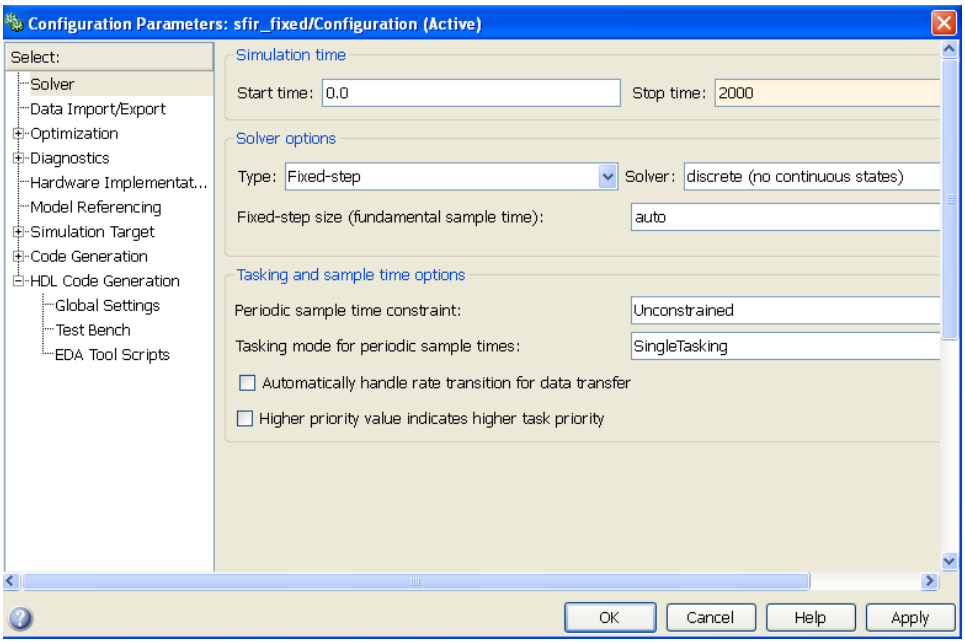


Figura 2.1 Ventana para la configuración de parámetros.

A la izquierda de la ventana Configuration Parameters existe la opción, HDL Code Generation, que permite especificar el subsistema del modelo con el que se desea trabajar, el tipo de HDL a generar, la carpeta de destino para el código. Además, permite comprobar un subsistema o modelo, para la detección de posibles incompatibilidades, por el uso de bloques que no soportan la generación de códigos o el uso ilegal de tipos de datos, y obtener un informe HTML en un navegador web (Figura 2.2) de la comprobación, todo esto a través del llenado o marcación de campos como muestra la Figura 2.3, y del botón Run Compatibility Checker.

Utilizar desde el Command Window de Matlab la función hdlcheck, cuando se tiene el modelo previamente elaborado, es otra vía de realizar la comprobación y obtener el informe HTML. Cuando se trata de diseñar, la función hdllib es la apropiada, porque genera una librería compilada a partir de la distribución de bloques de Simulink que generan código HDL.

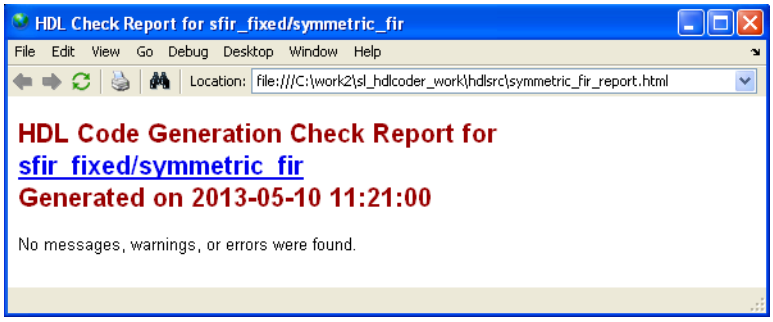


Figura 2.2 Informe HTML de la comprobación del filtro.

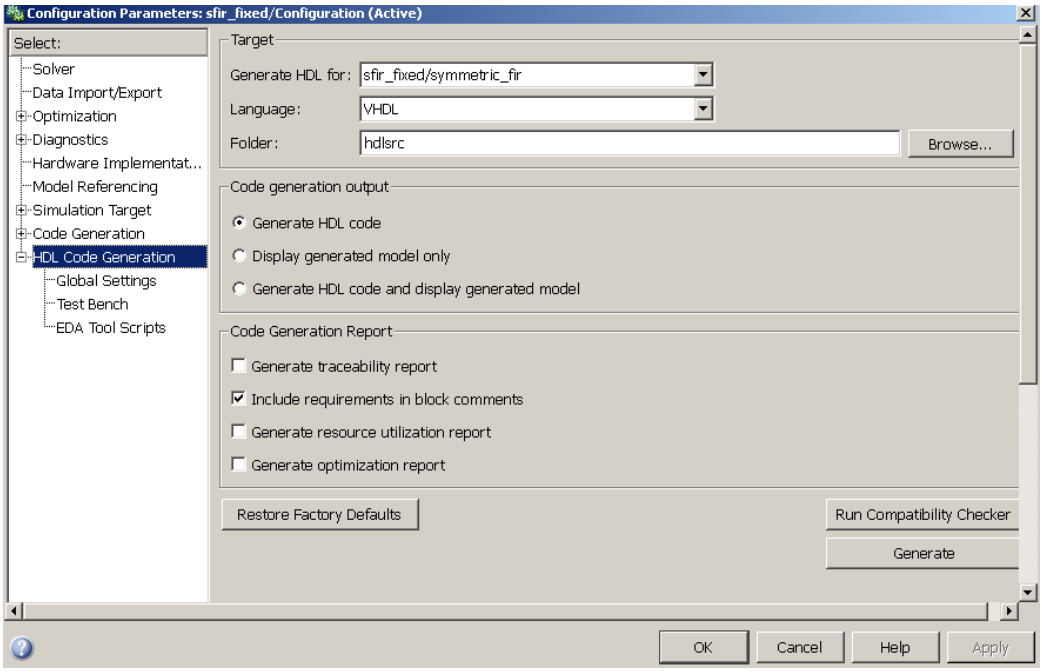


Figura 2.3 Sección HDL Code Generation.

Se debe salvar nuevamente el modelo una vez concluida la parametrización, para continuar con el siguiente paso, la generación del código. Existen varias maneras de generar, como presionar el botón Generate que se muestra en la Figura 2.3, y seleccionar desde el menú del modelo, Tools-HDL/Code Generation/Generate HDL o utilizando el comando:

>>makehdl ('sfir_fixed/symmetric_fir')

Cuando la generación del código no procede, Matlab informa del error, de lo contrario se despliegan mensajes de progreso. En la Figura 2.4 se muestra cómo uno de estos mensajes constituye un hipervínculo al editor de Matlab para ver el código generado. Además en la subcarpeta sl_hdlcoder_work aparecerá la carpeta hdlsrc con los siguientes archivos:

- symmetric_fir.vhd: que contiene una definición de la entidad y arquitectura de RTL (Registred Transfer Level) que lleva a cabo el filtro. Es el fichero con el código VHDL generado.
- symmetric_fir_compile.do: utilizado para compilar el código VHDL generado.
- symmetric_fir_map.txt: brinda una relación de todas las entidades que forman el subsistema.

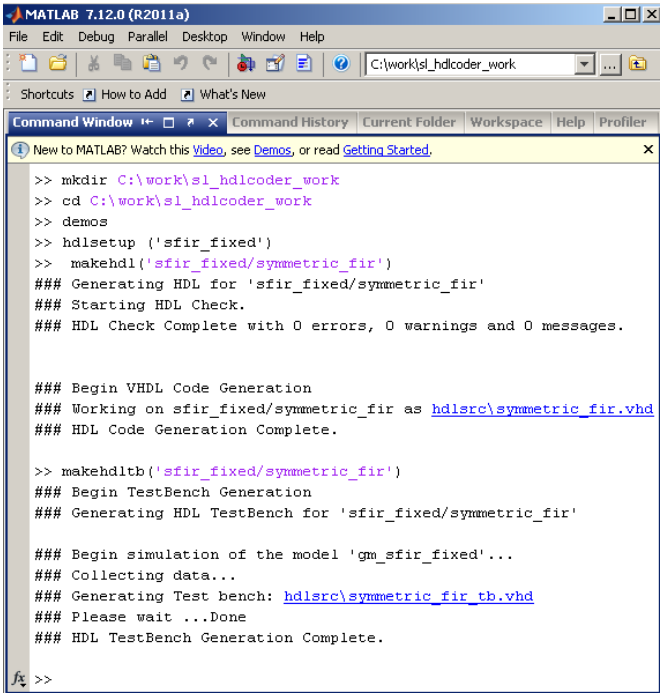


Figura 2.4 Ventana Command Window de Matlab.

De la opción HDL Code Generation se despliegan a su vez, otras opciones para la configuración de variados parámetros, como son:

- Global Settings
- Test Bench
- EDA Tool Scripts

Global Settings brinda las opciones para especificar características detalladas del código generado, y si ciertas optimizaciones son aplicables (Figura 2.5).

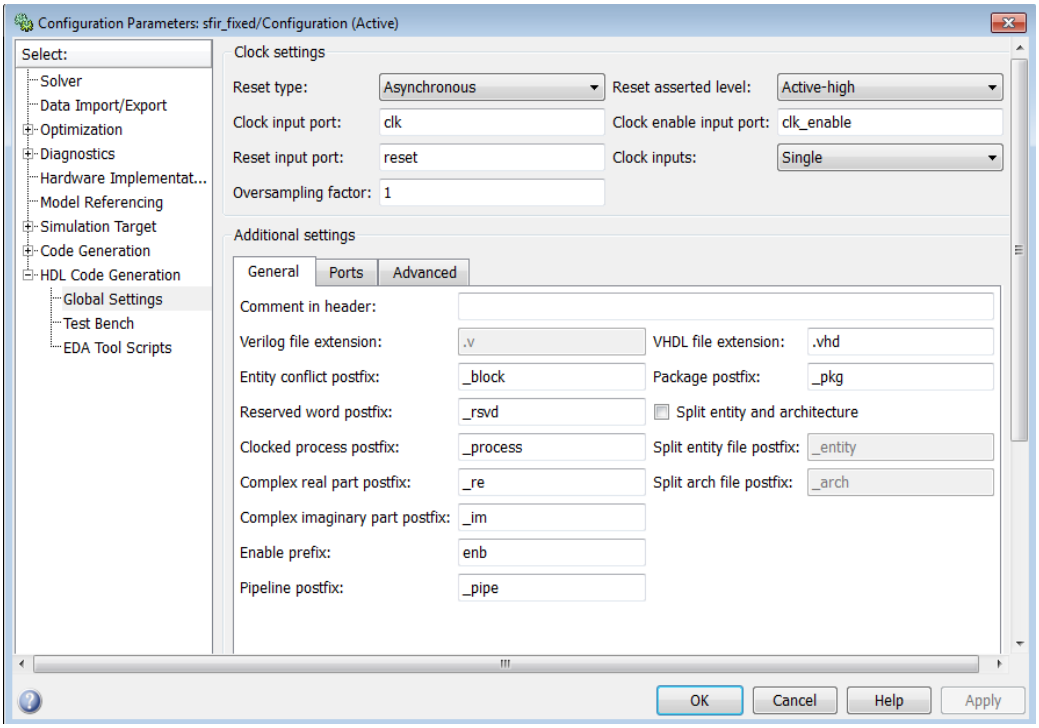


Figura 2.5 Ventana Global Settings de Matlab.

El banco de pruebas (test bench) se utiliza para manejar y verificar el funcionamiento de la entidad a la que se le generó anteriormente el código VHDL. Generar un test bench incluye datos de estímulos, generados por fuentes señaladas conectadas a la entidad bajo la prueba y datos de salida generados por la entidad bajo la prueba. Durante una simulación del test bench, estos datos se comparan con las salidas VHDL del modelo, como comprobación. En la Figura 2.6 se muestran los parámetros que por defecto son útiles para realizar el test bench. Si se desea realizar algún cambio, se hace manualmente siempre teniendo en cuenta las características de la aplicación. El botón Generate Test Bench ejecuta la generación o la utilización desde el Comand Window de Matlab del comando:

>>makehdltb ('sfir_fixed/symmetric_fir')

Los ficheros que genera el test bench en el directorio de trabajo, para el Filtro FIR Simétrico son:

- symmetric_fir_tb.vhd
- symmetric_fir_tb_compile.do
- symmetric_fir_tb_sim.do

La forma de acceder al test bench es idéntica a la utilizada para ver el código VHDL generado con anterioridad.

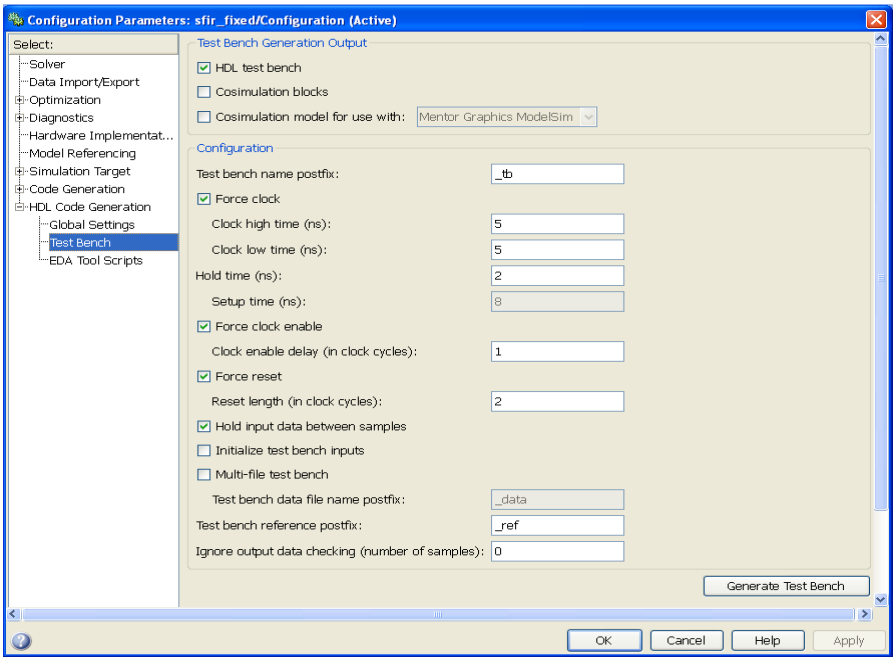


Figura 2.6 Parámetros del test bench.

EDA Tool Scripts permite configurar todas las opciones que controlan la generación de archivos relacionados con los códigos HDL, útiles en la simulación del modelo y con las herramientas de síntesis (Figura 2.7).

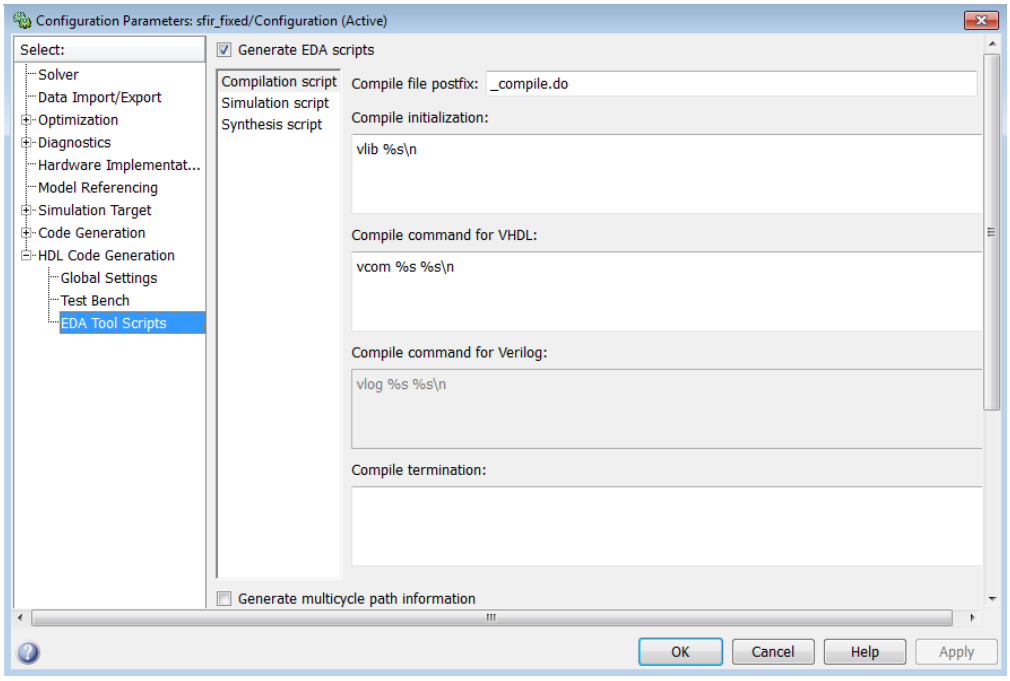


Figura 2.7 Ventana EDA Tool Scripts de Matlab.

Matlab cuenta además con funciones que le permiten prepararse para utilizar Xilinx ISE, porque desde Matlab se pueden crear proyectos para el trabajo con el FPGA. Cuando esto ocurre la ventana Configuration Parameters brinda una nueva opción EDA Link como se muestra en la Figura 2.8, al ejecutar los comandos siguientes.

>>setupxilinxtools

>>makefpgaproject('sfir_fixed')

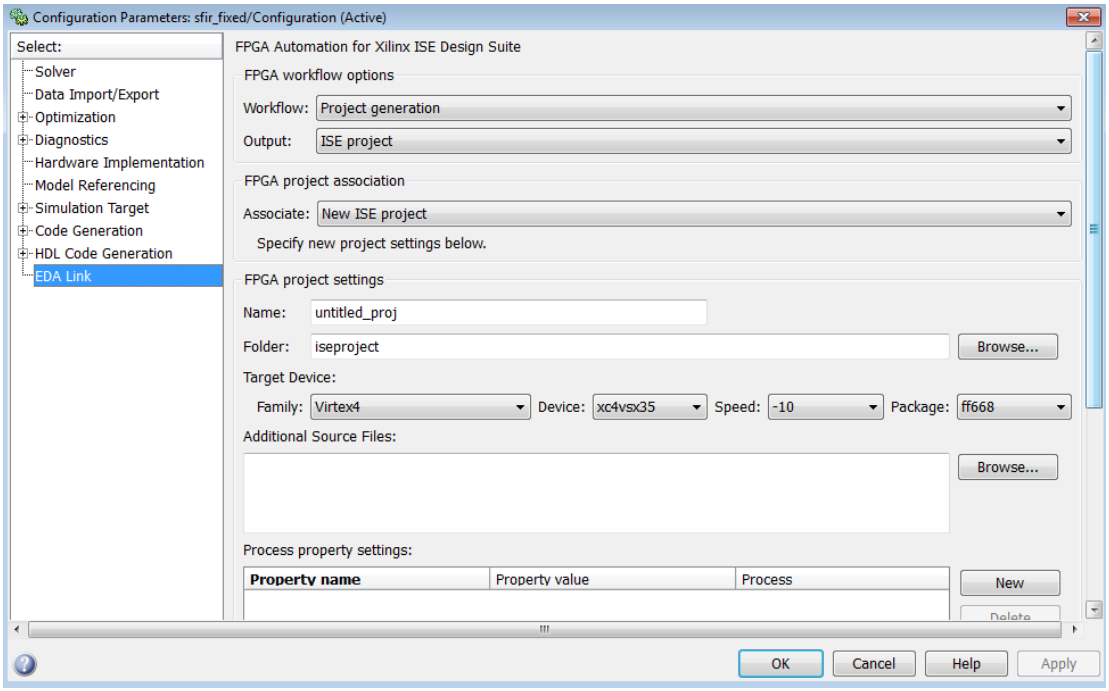


Figura 2.8 Ventana EDA Link de Matlab.

La opción EDA Link permite crear un proyecto para trabajar con la FPGA, para ello se especifica el nombre del proyecto, la carpeta para guardarlo, la familia de FPGA a utilizar, tipo de dispositivo, su velocidad, empaquetamiento, entre otros.

2.3.2 Generacion de código Verilog

Los pasos para obtener código Verilog y su test bench es similar al descrito para lograr códigos VHDL, por lo que solo se hará referencia a las diferencias. Una de las diferencias es la de seleccionar Verilog y no VHDL, en el campo Language (Idioma), en la sección HDL de la ventana de Configuration Parameters, como se muestra en la Figura 2.9.

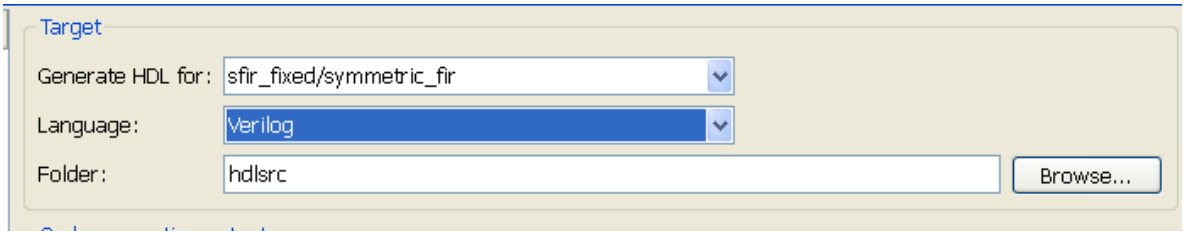


Figura 2.9 Fragmento de la ventana Configuration Parameters.

Si se emplean los comandos makehdl y makehdltb, también debe especificarse el tipo de HDL requerido como se muestra a continuación:

>>makehdl('sfir_fixed/symmetric_fir','TargetLanguage','verilog')

>>makehdltb('sfir_fixed/symmetric_fir','TargetLanguage','verilog')

Los archivos relacionados con el código, son guardados en el directorio de trabajo. La diferencia con respecto a los ficheros obtenidos al generar código VHDL, es que la extensión del archivo que contiene el código Verilog, ahora es .v. Igual sucede con el comando makehdltb, que genera los archivos:

symmetric_fir_tb.v: contiene el código Verilog.

symmetric_fir_tb_compile.do: Esta archivo compila y carga las entidades de symmetric_fir.v y symmetric_fir_tb.v.

symmetric_fir_tb_sim.do: Para inicializar el simulador y ejecutar una simulación.

2.4 Conclusiones Parciales

En este capítulo se caracterizaron los lenguajes de descripción de hardware, por su utilidad para el diseño de circuitos digitales. Se resumieron los aspectos más importantes de los software Matlab y Xilinx ISE, herramientas de alto nivel, para el modelado, simulación e implementación de circuitos en FPGA. Además, se mostraron los pasos para obtener con la herramienta HDL Coder de Matlab códigos Verilog y VHDL.

CAPÍTULO 3. IMPLEMENTACIÓN Y EVALUACIÓN DE LAS APLICACIONES DE SDR DESARROLLADAS

En este capítulo se desarrollan aplicaciones de la Radio Definida por Software utilizando el software Matlab. Para una mejor comprensión de las mismas, se realiza una introducción, donde aparecen aspectos como el principio de funcionamiento y la utilidad práctica. Además, se describe cómo se implementaron los subsistemas de cada diseño. A las aplicaciones seleccionadas se les generó código HDL con la herramienta Simulink HDL Coder de Matlab, y se utilizó el Xilinx ISE para la síntesis de las aplicaciones. Al final se comprueba el funcionamiento, en el Kit de desarrollo Nexys2, de un circuito constituido por un generador de secuencia, un codificador convolucional y un decodificador de Viterbi.

3.1 Aplicaciones de SDR desarrolladas en Matlab

En el Capítulo 2 se caracterizó el software Matlab, indicando las ventajas de Simulink para la implementación de aplicaciones de la SDR, debido a la diversidad de prestaciones que ofrece. A continuación se describen dos de estas aplicaciones, un receptor OFDM con 512 portadoras Streaming I/O FFT y la capa física WirelessMan-OFDM utilizando modulación BPSK (Binary Phase Shift Keying).

3.1.1 Simulación de un Receptor OFDM con 512 portadoras Streaming I/O FFT

La tecnología OFDM (Orthogonal Frequency Division Multiplexing) consiste en enviar un conjunto de ondas portadoras de diferentes frecuencias, donde cada una transporta información, moduladas en QAM (Quadrature Amplitude Modulation) o en PSK (Phase

Shift Keying). Además los canales de banda estrecha de OFDM son ortogonales entre sí, lo que evita el uso de bandas de guardas, proporcionando un uso eficiente del espectro. La multiplexación de portadoras OFDM es muy robusta frente al multicamino, que es habitual en los canales de radiodifusión, frente a las atenuaciones selectivas en frecuencia y a las interferencias de RF (Sidhu et al., 2012).

A continuación se diseña un receptor OFDM con 512 portadoras Streaming I/O FFT. El modelo que se observa en la Figura 3.1, para una mayor comprensión de su funcionamiento, se describe a partir de los siguientes bloques funcionales:

- Fuente de datos
- Transmisor OFDM
- Canal
- Receptor OFDM
- FFT
- Medición de BER (Bit Error Rate)

Estos bloques funcionales, son construidos por bloques de Simulink, que se comentan a continuación.

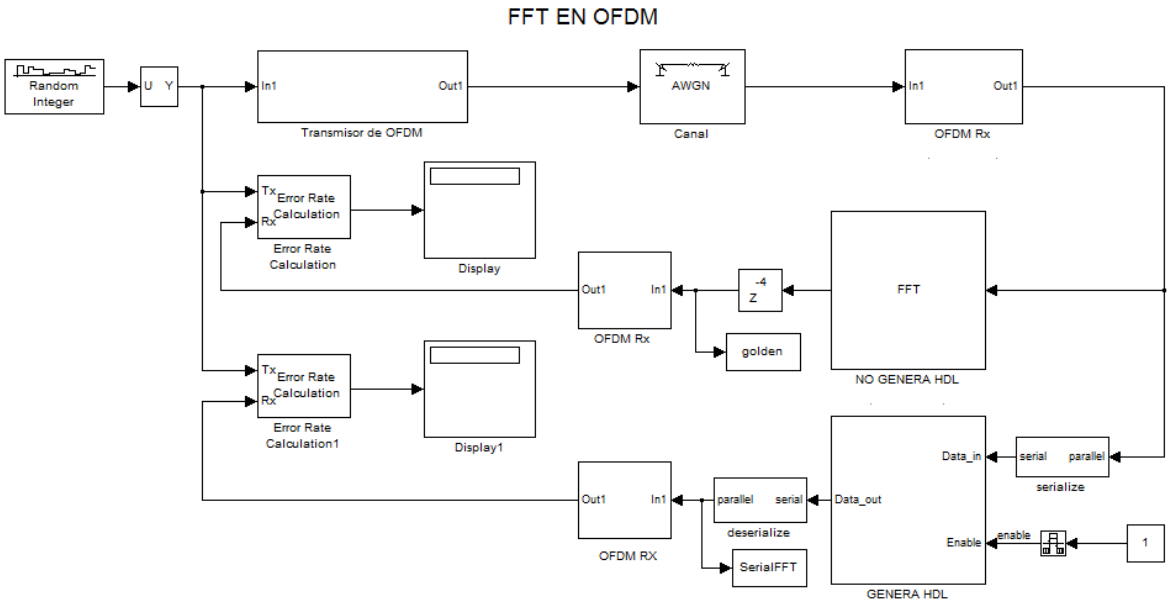


Figura3.1. Receptor OFDM con 512 portadoras Streaming I/O FFT.

La fuente de datos es un generador binario, constituido por el bloque de Simulink, Random Integer, que comanda la tasa de transmisión de datos de todo el sistema. Esta tasa no es la misma con que los datos pasan al canal, ni a la que se recibe en la etapa de recepción, porque la modulación implica colocar los datos en tramas, insertar pilotos, generar preámbulos, entre otras tareas que producen un aumento de la tasa de datos. Entre los parámetros que comandan la producción de bits en este bloque se encuentran:

- Tiempo de Muestreo (Sample Time): es el tiempo que tarda el bloque en generar internamente un bit y el consecutivo.
- Muestras por trama (Samples per frame): es el número de bits que se enviarán juntos en una sola trama, o como lo es para el ambiente de Matlab, un vector columna.

En la Figura 3.2 se muestra el diagrama en bloques de Simulink, del Transmisor OFDM.

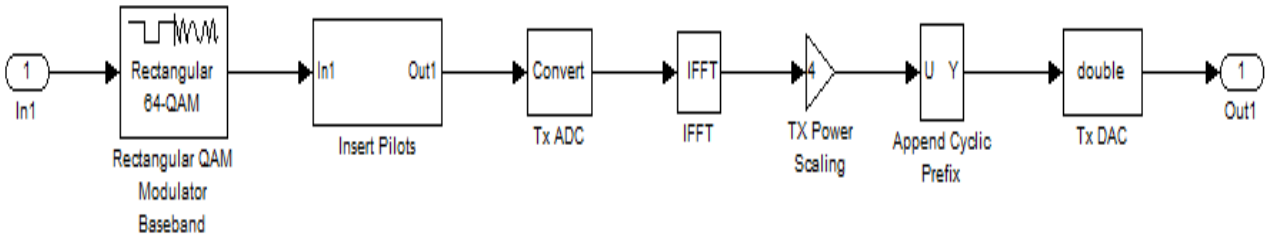


Figura 3.2. Diagrama en bloques del Transmisor OFDM.

En un transmisor OFDM se realizan tareas como:

- La modulación digital I/Q o codificación. El modelo utiliza el modulador 64-QAM de la biblioteca de comunicaciones de Simulink. En la modulación digital de amplitud en cuadratura (QAM), la información está contenida tanto en la amplitud como en la fase de la portadora.
- Elaboración de la trama de acuerdo al estándar. En el modelo es utilizada la inserción de pilotos, aunque existen variantes, como la inserción de guardas en frecuencia y la inserción del nivel DC. Los pilotos tienen diversas utilidades, entre

ellas, facilitar la sincronización y la estimación del canal, además de la detección de desplazamientos en fase y frecuencia.

- Conversión análoga digital y viceversa.
- IFFT (Inverse Fast Fourier Transform), implementada por un bloque de la biblioteca de procesamiento de señales de Simulink, para proveer una manera de modular los datos sobre N subportadoras ortogonales.
- Adición del prefijo cíclico (CP). El CP es utilizado durante el intervalo de guarda, consiste en añadir por delante de las N muestras que emite la IFFT, G muestras que no son más que copias de las últimas G del símbolo.

El canal utilizado es el clásico canal de ruido blanco gaussiano aditivo (AWGN), que es sencillo de implementar de manera inicial con los FPGA (Figura 3.3).

Figura 3.3 Canal AWGN.

Un receptor OFDM realiza funciones similares a las de un transmisor, pero en sentido inverso. En la Figura 3.4 se observa como el primer bloque receptor de la Figura 3.1 solo realiza las funciones:

- Conversión análoga digital.
- Conversión de tramas.
- Extracción del prefijo cíclico.

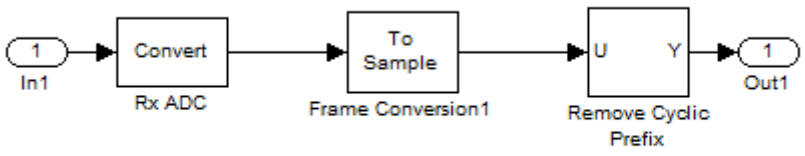


Figura 3.4 Receptor OFDM.

Al bloque receptor le continúa la implementación de dos bloques para realizar la FFT (Fast Fourier Transform), uno de ellos genera HDL y el otro no. Los dos bloques son utilizados para comprobar y comparar el rendimiento del diseño general, debido a las diferencias en sus prestaciones. El bloque que realiza la FFT y además, genera HDL, es la única unidad que soporta la generación de código HDL en el circuito, y en la Figura 3.5 se observan sus parámetros. El bloque que no genera HDL, calcula un vector de entrada de 512 portadoras por cada muestra y opera a una razón de muestreo de 512 muestras/s (Figura 3.6)

Otros bloques empleados en el diseño son los convertidores de paralelo a serie y viceversa, y unidades de demora.

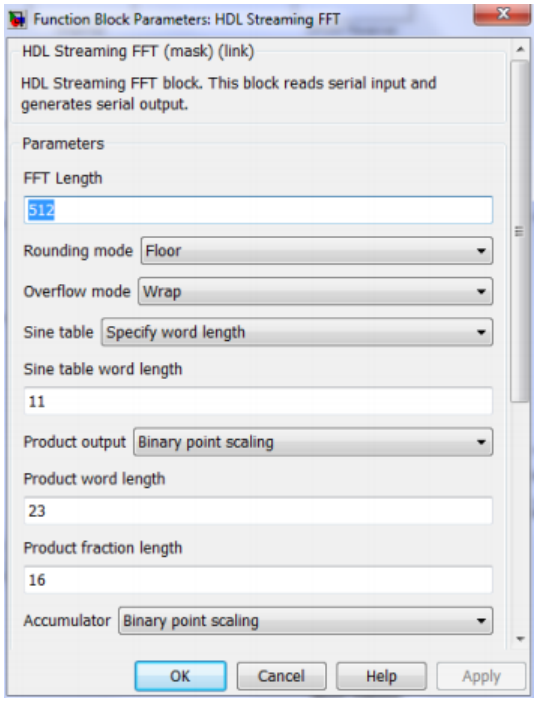


Figura 3.5 Parámetros del bloque que genera HDL.

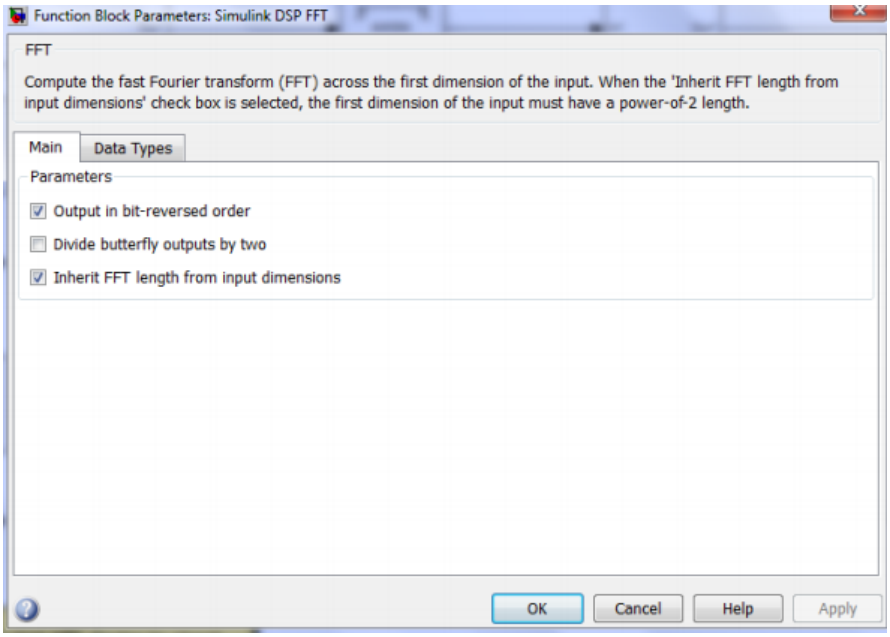


Figura 3.6 Parámetros del bloque que no genera HDL.

Después de realizada la FFT, se encuentra la implementación de un segundo receptor para la detección de símbolos (Figura 3.7)

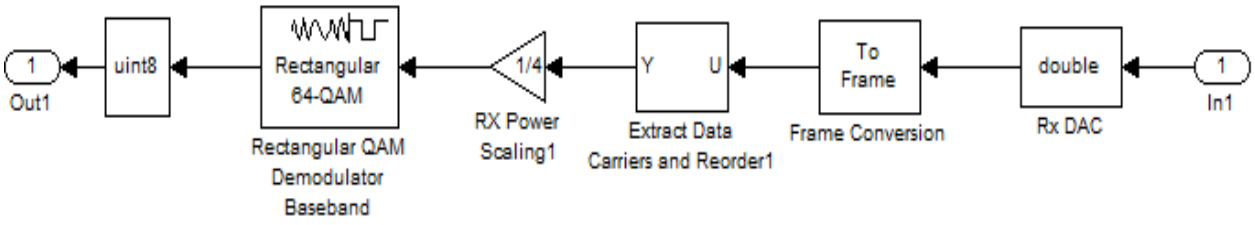


Figura 3.7 Receptor OFDM para la detección de símbolos.

Al final de esta etapa se tienen los datos en forma binaria dentro de tramas. De este modo se tienen los datos exactamente en el mismo formato en que fueron generados, lo cual habilita para hacer la medición de BER.

La medición de BER, es la etapa con la que se cierra el circuito de la comunicación en el esquema general. Está conformada básicamente por el bloque estándar de Simulink denominado Error Rate Calculation, cuya labor es recibir bits enviados y recibidos,

comparar y contar los errores. Sus opciones permiten la sincronización con el sistema tomando en consideración los retardos y los metadatos que no constituyen información enviada. En la Figura 3.8 se muestran los resultados de la simulación, evidenciando que el bloque Genera HDL, además de soportar la generación de código, funciona correctamente, porque los resultados obtenidos con respecto al bloque NO GENERA HDL son idénticos.

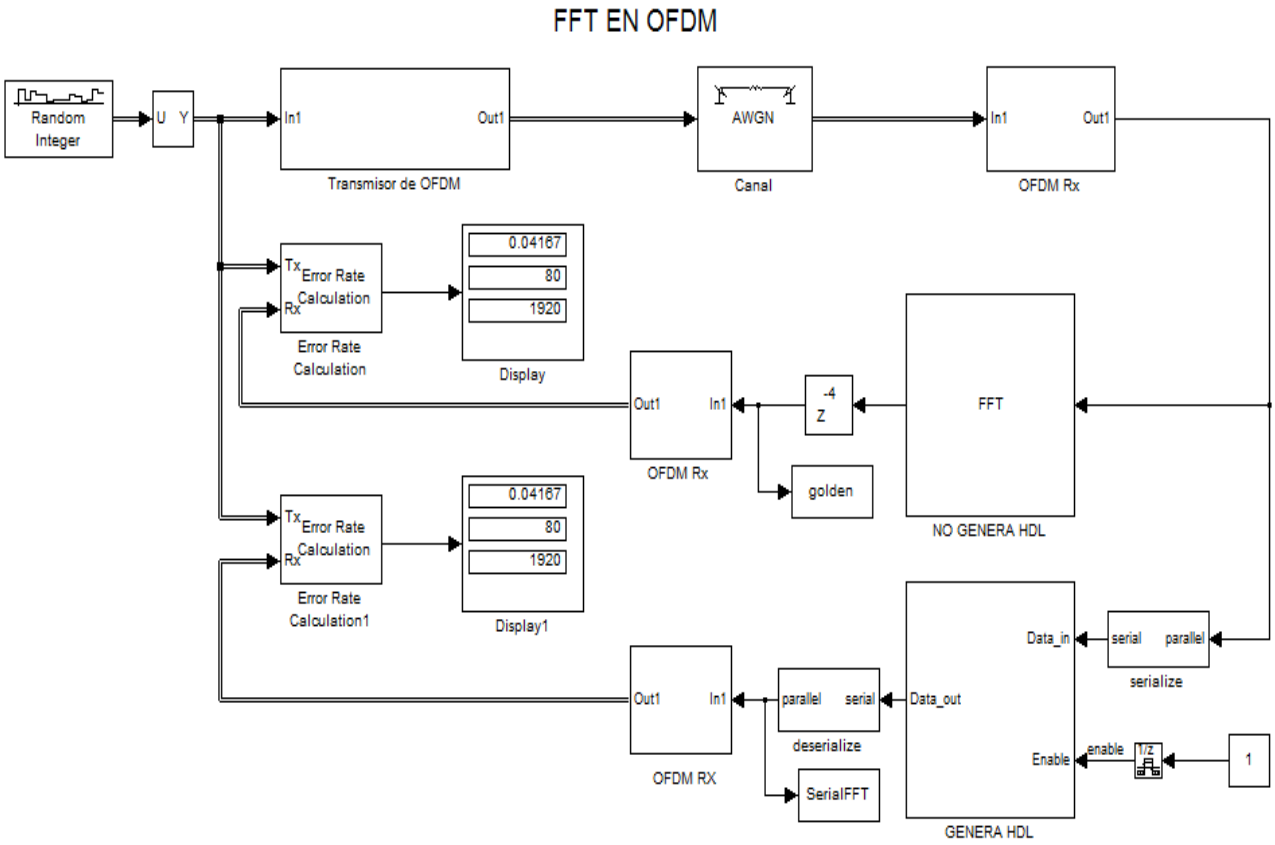


Figura 3.8 Resultados de la Simulación.

3.1.2 Simulación de la capa física WirelessMAN-OFDM utilizando modulación BPSK

La norma WirelessMan, es una forma de denominar al estándar IEEE 802.16D, en la que se basan tecnologías como WiMAX. El grupo de trabajo de la IEEE802.16D se formó en 1998 con el objetivo de desarrollar un estándar para las comunicaciones inalámbricas de banda ancha. El enfoque inicial fue el desarrollo de una línea de vista directa (LoS), basada en la conexión punto a multipunto, para un funcionamiento en las frecuencias comprendidas entre 10 y 66 GHZ (NGOUESSI, 2011).

A continuación se realiza un diseño de la capa física WirelessMan-OFDM utilizando modulación BPSK (Figura 3.9). La aplicación utiliza varios bloques de Simulink, que en su mayoría se describen a continuación.

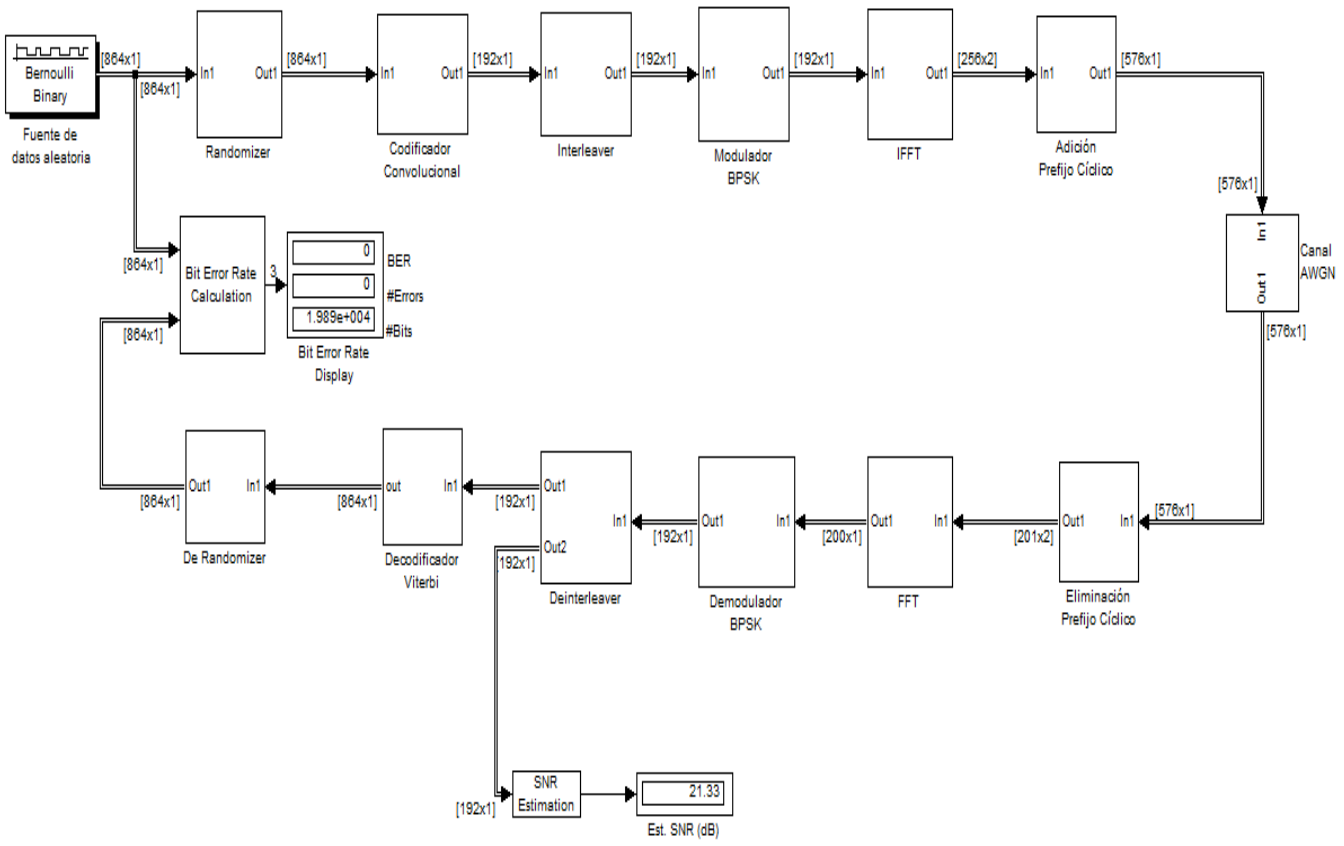


Figura 3.9.La capa física WrelessMan-OFDM utilizando modulación BPSK.

La fuente de datos utilizada, genera números binarios aleatorios que usan distribución de Bernoulli. La distribución de Bernoulli con el parámetro p produce el cero, con probabilidad p, y el uno con probabilidad 1-p. En la Figura 3.10 se observan los parámetros asignados a la fuente de datos.

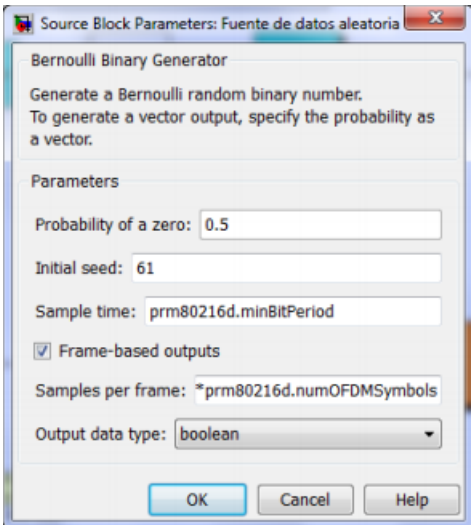


Figura 3.10 Parámetros de la Fuente de Datos.

Los bloques funcionales Randomizer y Derandomizer están constituidos por el bloque de Simulink, PN Sequence Generator y una compuerta lógica XOR (Figura 3.11). El bloque PN Sequence Generator genera una secuencia lineal con retroalimentación, con un polinomio generador como parámetro y se encuentra en la biblioteca de comunicaciones. La compuerta lógica XOR es empleada para realizar la función booleana A'B+AB', y pertenece a la biblioteca de operaciones lógicas.

EL objetivo del Randomizer no es lograr una técnica de corrección de errores, sino un mecanismo de prevención de errores. Se utiliza para conseguir una densidad de potencia uniforme en el ancho de banda de transmisión para cualquier tipo de portadora digital transmitida, convirtiendo cualquier flujo de bits de información, a un número igual de unos y ceros (NGOUESSI, 2011).

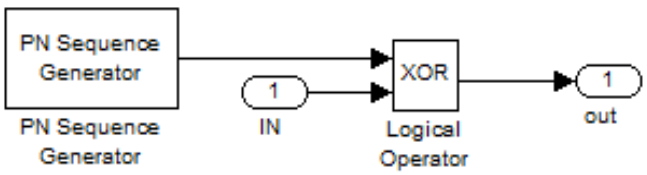


Figura 3.11Randormizer y Derandormizer.

El codificador convolucional utilizado se encuentra en la biblioteca de comunicaciones de Simulink. Un código convolucional es un código lineal, donde la suma de dos palabras de código, no importa cuales, es también una palabra de código. Este código se utiliza para mapear

k bits de información dentro de una palabra de código de *n* bits. La implementación de este tipo de código da una codificación continua, donde la secuencia de bit codificada depende de los bits actuales y de los bits previos (NGOUESSI, 2011).

Interleaver, es un bloque de la biblioteca especial que brinda la IEEE sobre el estándar 802.16D, implementada en Simulink, los valores de sus parámetros se ajustan en dependencia al tipo de modulación utilizada (Figura 3.12). El interleaving, es en informática una forma de ordenar datos de manera no-contigua para mejorar su desempeño y se utiliza en las tecnologías de transmisión de datos digitales, contra ráfagas de errores.

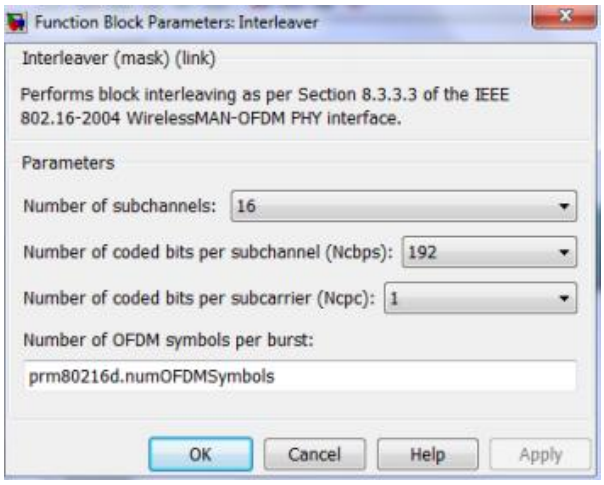


Figura 3.12 Parámetros del bloque Interleaver.

El modulador BPSK se encuentra en la biblioteca de las comunicaciones de Simulink (Figura 3.13).La modulación PSK consiste en codificar los valores binarios como cambios de fase de la señal portadora, la PSK binaria se denomina BPSK. La modulación BPSK utiliza dos estados para las fases +/- π/2.Se realiza con el objetivo de obtener un mejor aprovechamiento del canal de transmisión y proteger la señal del ruido.

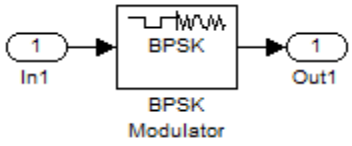


Figura 3.13 Modulador BPSK.

El bloque IFFT (Transformada Inversa de Fourier) utilizado en el diseño que se muestra en la Figura 3.9, pertenece a la biblioteca de procesamiento de señales de Simulink.

Al realizarse la IFFT continúa la adición del prefijo cíclico y seguidamente los datos pasan al canal (AWGN)(estos bloques se comentaron con anterioridad).

Después del canal (AWGN) sigue el inverso de los procesos realizados hasta ahora como son: extracción del prefijo cíclico, FFT, demodulación BPSK, de-interleaver, decodificación de Viterbi y las funciones del bloque de-randormizer.

La unidad de-interleaver, se tomó de la librería especial que brinda la IEEE. El decodificador convolucional que se implementó sigue el algoritmo de Viterbi, y se encuentra en la biblioteca de comunicaciones de Simulink. El algoritmo de decodificación de Viterbi está basado en los diagramas de Trellis, diagramas en forma de red, en donde cada línea horizontal corresponde con uno de los estados del codificador y cada línea vertical corresponde a uno de los niveles del árbol del código.

El último bloque implementado en el circuito general es utilizado para la medición de la BER, que es denominado Error Rate Calculation. Otra estimación que se realiza es la del valor de la relación señal a ruido.

3.2 Generación y simulación del código VHDL de las aplicaciones implementadas

En el siguiente epígrafe se obtienen los códigos VHDL de determinados bloques de las aplicaciones implementadas con anterioridad, utilizando los pasos presentados en el Capítulo 2 para trabajar con la herramienta Simulink HDL Coder de Matlab.

3.2.1 Generación y simulación del código VHDL de un Receptor OFDM con 512 portadoras Streaming I/O FFT

En la aplicación del receptor OFDM con 512 portadoras Streaming I/O FFT, el bloque denominado Genera HDL, que realiza la FFT, es el único bloque que soporta la generación de código HDL.

El bloque Genera HDL, tiene dos puertos de entrada din y start; y tres puertos de salida dout, dvalid y ready. El puerto din permite la entrada de señales de datos complejas y de punto fijo, y start, de control booleana. Cuando el puerto start se encuentra con un “1”lógico, el bloque inicia el procesado de la trama de datos. Dout es el puerto encargado de brindar los datos de salida, y cuando esto ocurre dvalid se pone a “1” lógico. El puerto

ready para indicar que está listo para procesar una nueva trama, pone un "1" lógico. Tanto ready como dvalid trabajan con señales de control booleanas.

Otras especificaciones necesarias en el funcionamiento del bloque son:

- Cooley-Tukey Radix-2 DIF FFT
- Entrada y salida en serie
- Ancho de banda de los datos: 12 bits
- 512 portadoras
- Streaming I/O

Para la generación del código VHDL del bloque FFT Genera HDL, se creó una carpeta denominada work en el disco local C de la PC y una vez realizado el diagrama general fue salvado en dicha carpeta. En la Figura 3.14 se observa la ventana Configuration Parameters, del receptor OFDM en la sección HDL Code Generation. Se accede a la ventana Configuration Parameters para escoger el bloque que se le generará código, el tipo de HDL requerido y la carpeta donde se guardarán los archivos relacionados con el código. Además se seleccionaron todas las opciones de la sección Code Generation Report, para obtener el resumen del informe sobre el proceso de generación de HDL (Figura 3.15). Desde la ventana del reporte se puede acceder tanto al esquemático general como al del bloque FFT Genera HDL.

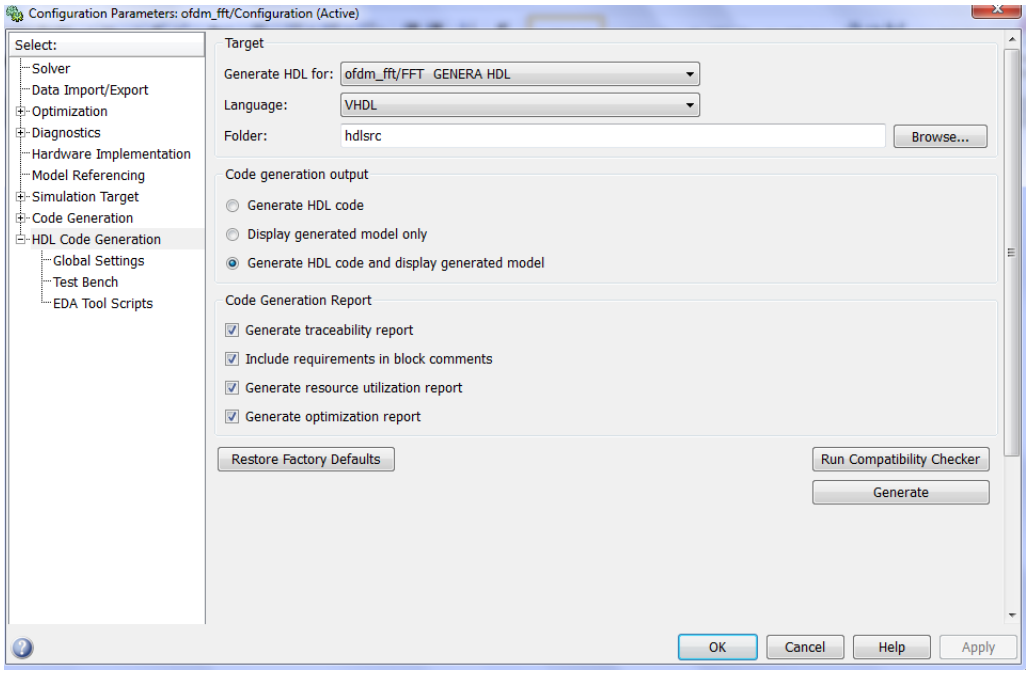


Figura 3.14 Ventana para la configuración de parámetros del bloque GENERA HDL (FFT).

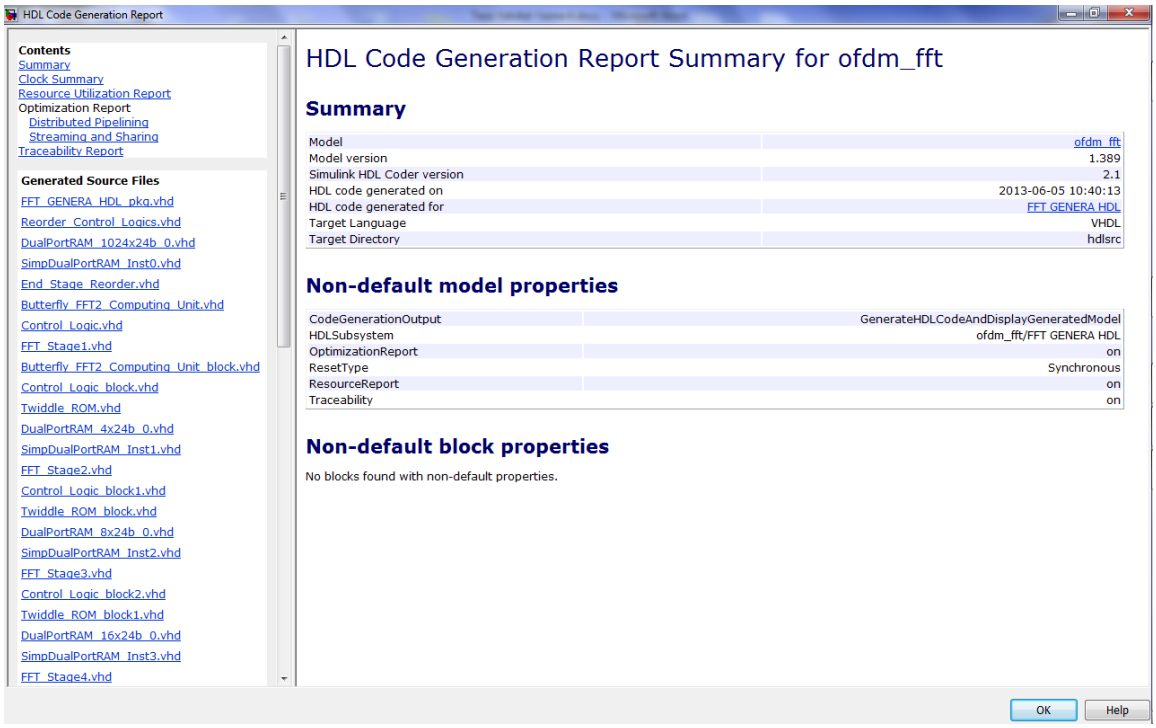


Figura 3.15 Reporte de la Generación de código HDL del bloque GENERA HDL (FFT).

Al especificar estos parámetros propios de la aplicación se realizó un chequeo de compatibilidad, que resulto sin errores y posteriormente la generación del código.

El test bench para la comprobación del código se realizó también desde el propio Matlab, teniendo en cuenta parámetros como un período de 10ns, es decir se especificó 5ns para los campos clock hight time y clock low time (tiempos de reloj de subida y bajada) (Figura2.6).

Una vez logrado el código VHDL y su test bench, este se exportó al software Xilinx ISE, para la síntesis del diseño. Utilizar el Matlab para generar códigos HDL y test bench, agiliza el trabajo, porque si no este debe realizarse manualmente en el Xilinx ISE.

A continuación se describe el trabajo con Xilinx ISE, que comienza con la creación de un nuevo proyecto. En la Figura 3.16 se observa la ventana New Project Wizard, utilizada para dar nombre al proyecto, seleccionar la familia del FPGA empleado, tipo de dispositivo (que especifica la cantidad de compuertas del FPGA), el encapsulado y su velocidad. Además se declara el simulador y la herramienta de síntesis empleada, entre otros parámetros.

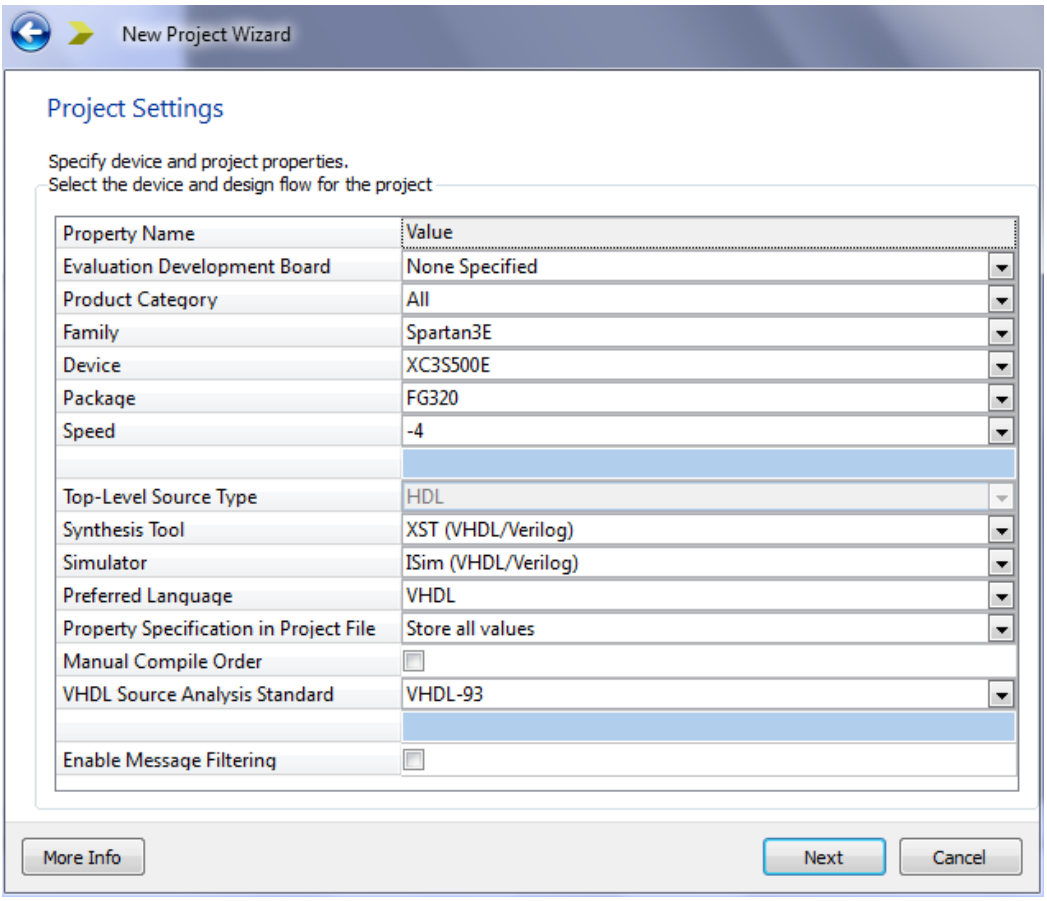


Figura 3.16 Ventana de Xilinx para la creación de un proyecto nuevo.

En la Figura 3.17 se muestra el ambiente de Xilinx, en la sección de implementación, donde se añaden todos los archivos fuentes al programa, es decir todos los archivos .vhd obtenidos a Matlab se copian en Xilinx, todos estos archivos tienen las opciones de simular e implementar, menos el test bench que es solo para simular. La simulación sólo va a realizarse para el conjunto de estímulos que se colocan en el test bench.

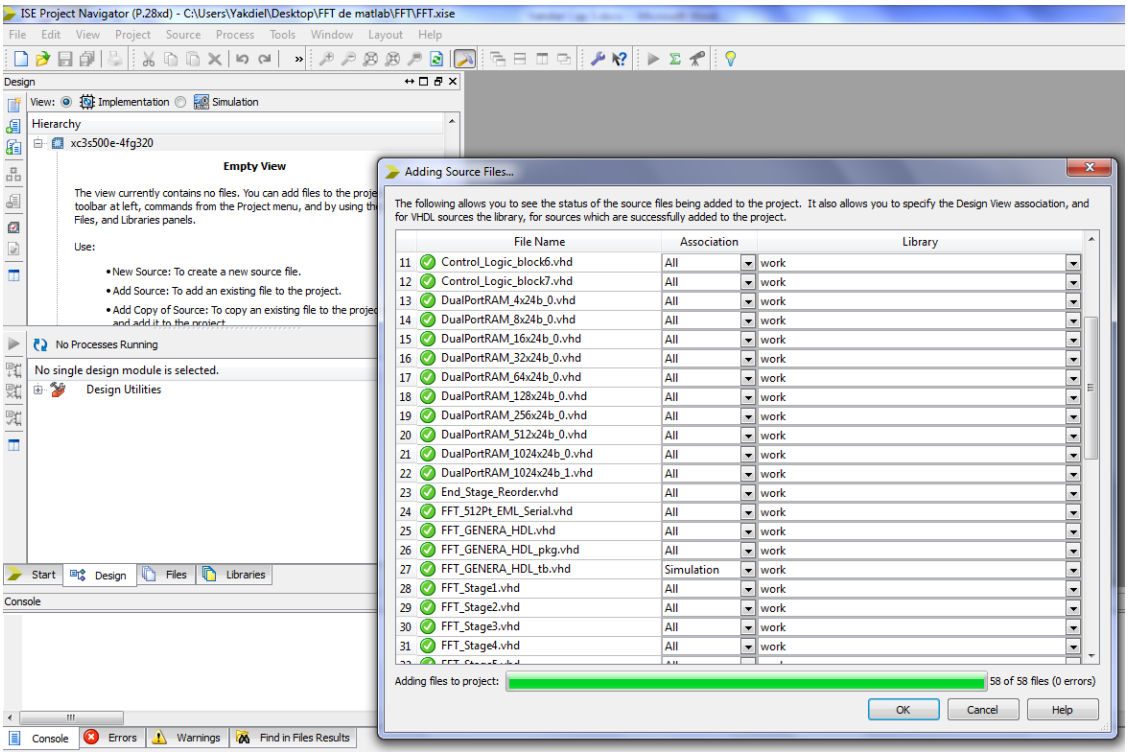


Figura 3.17 Estado de los archivos fuentes al ser agregados al proyecto.

Cuando se añaden todos los archivos fuentes, a la derecha de la ventana principal de Xilinx, se tiene el código VHDL. Una vez realizado el diseño se puede realizar la revisión de la sintaxis del VHDL, como se muestra en la Figura 3.18. Si aparecen errores se debe revisar la sintaxis, corregir los errores y volver a revisarla. Una vez sin errores se está en disposición de realizar la simulación funcional del diseño.

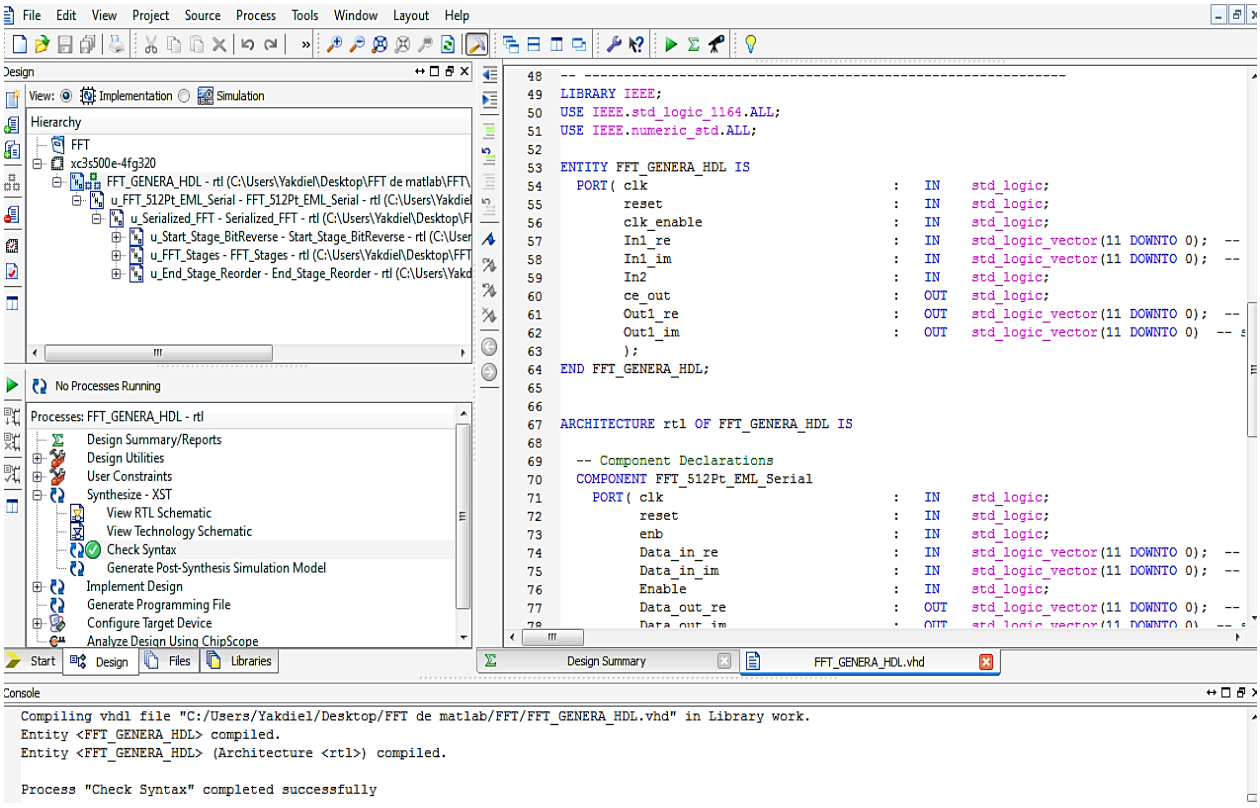
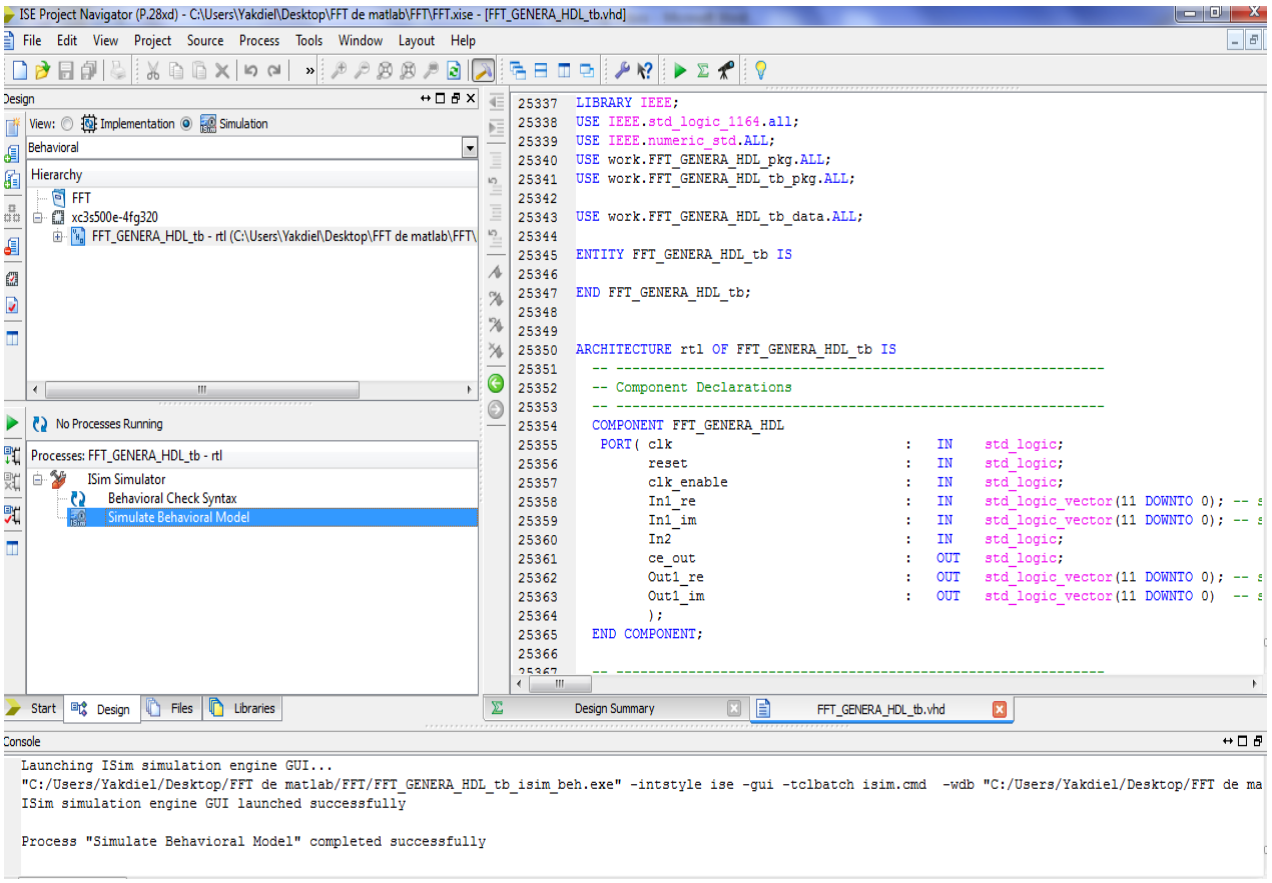


Figura 3.18 Proceso de chequeo de la sintaxis.

Para llevar a cabo la simulación se selecciona en la sección View, Simulation, se escoge el banco de pruebas y se manda a simular mediante el simulador ISim (Figura 3.19).




```
25337  LIBRARY IEEE;
25338  USE IEEE.std_logic_1164.all;
25339  USE IEEE.numeric_std.ALL;
25340  USE work.FFT_GENERA_HDL_pkg.ALL;
25341  USE work.FFT_GENERA_HDL_tb_pkg.ALL;
25342
25343  USE work.FFT_GENERA_HDL_tb_data.ALL;
25344
25345  ENTITY FFT_GENERA_HDL_tb IS
25346
25347  END FFT_GENERA_HDL_tb;
25348
25349
25350  ARCHITECTURE rtl OF FFT_GENERA_HDL_tb IS
25351    -- ------------------------------------------------------------
25352    -- Component Declarations
25353    -- ------------------------------------------------------------
25354    COMPONENT FFT_GENERA_HDL
25355     PORT( clk                          :   IN    std_logic;
25356           reset                        :   IN    std_logic;
25357           clk_enable                   :   IN    std_logic;
25358           In1_re                       :   IN    std_logic_vector(11 DOWNTO 0); -- s
25359           In1_im                       :   IN    std_logic_vector(11 DOWNTO 0); -- s
25360           In2                          :   IN    std_logic;
25361           ce_out                       :   OUT   std_logic;
25362           Out1_re                      :   OUT   std_logic_vector(11 DOWNTO 0); -- s
25363           Out1_im                      :   OUT   std_logic_vector(11 DOWNTO 0)  -- s
25364           );
25365    END COMPONENT;
25366
```

Figura 3.19 Proceso de simulación.

En la Figura 3.20 se muestran los resultados de la simulación, donde están las señales de entrada y salida con sus valores. Por ejemplo, se pude apreciar la señal clk con valor “1”y periodo de 10ns, el reset en “0” y clk enable también con valor “1”.

El comportamiento de las señales en la Figura 3.20 muestra el correcto funcionamiento del bloque FFT, que realiza una convolución circular de dos señales, es decir supone el cálculo de dos elementos de la salida a partir únicamente de dos elementos de la entrada.

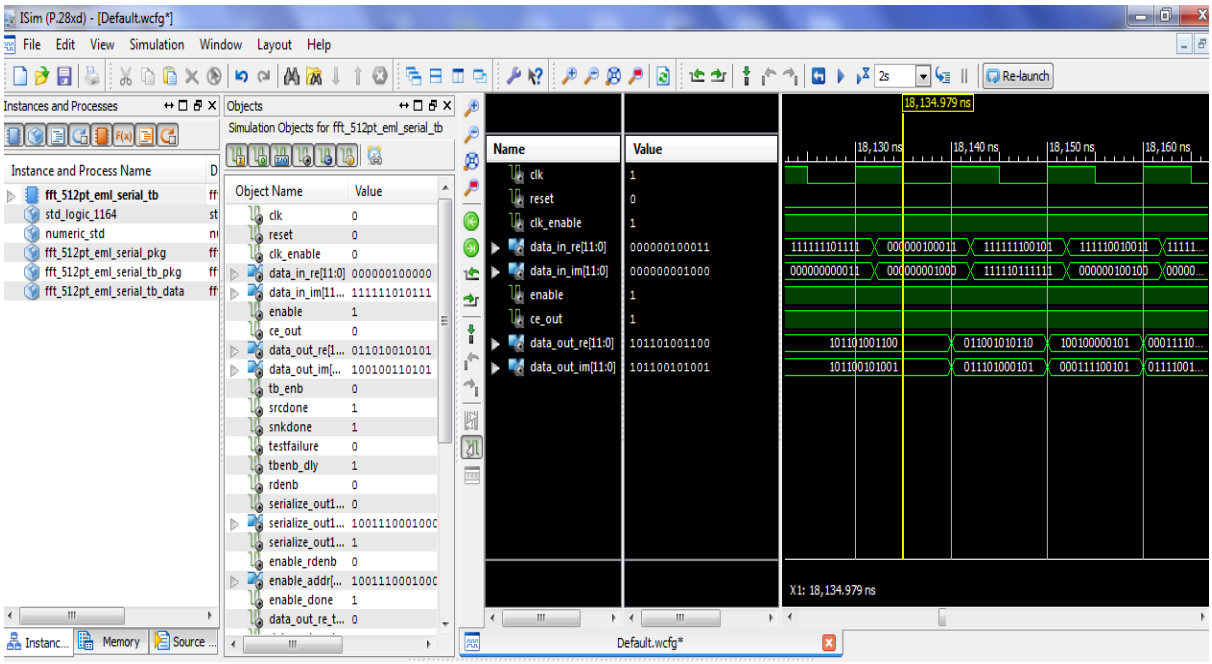


Figura 3.20 Resultados de la simulación del bloque GENERA HDL (FFT) utilizando el simulador ISim.

En la parte inferior de la ventana principal de Xilinx ISE, la opción Design Summary brinda los datos que se encuentran en la Tabla 3.1. Estos datos son resultados del proceso de "traducir" que se encarga de llevar el diseño digital a la lógica y componentes del PLD, en el caso del FPGA a bloques lógicos programables, CLB; tablas de consultas, LUT; bloques de entrada/ salida, IOB; controladores de reloj, DCM etc.

Tabla3.1 Resumen sobre la utilización de dispositivos (valores estimados)

Utilización de la lógica	usado	disponible	utilizado
Número de slices	2659	4656	57%
Número de slices flip flops	2917	9312	31%
Número de LUTs de 4 entradas	3349	9312	35%
Número de IOBs unidos	53	232	22%
Número de BRAM	13	20	65%
Número de MULT 18X18SIOs	20	20	100%

Número de GCLK	1	24	4%

3.2.2 Generación y simulación del código VHDL de la capa física WirelessMan-OFDM, utilizando modulacion BPSK

En el diagrama en bloques de Simulink, realizado de la capa física WrelessMan-OFDM las unidades que soportan la generación de HDL, son el Codificador Convolucional y el Decodificador de Viterbi. Los pasos para obtener el VHDL de cada bloque, son los mismos que se utilizaron con anterioridad, por lo que a continuación solo se harán referencia a las diferencias.

En la Figura 3.21 se muestran los parámetros del Codificador Convolucional que fueron asignados según a lo orientado por la norma IEEE802.16D para la modulación BPSK. El Codificador Convolucional de datos binarios puede procesar múltiples símbolos en un momento y acepta entradas que varían en longitud durante la simulación. El funcionamiento del bloque se gobierna por el parámetro, modo de funcionamiento. En continuo, como modo de funcionamiento, el bloque retiene los estados del codificador al final de cada entrada, para el uso con la próxima trama.

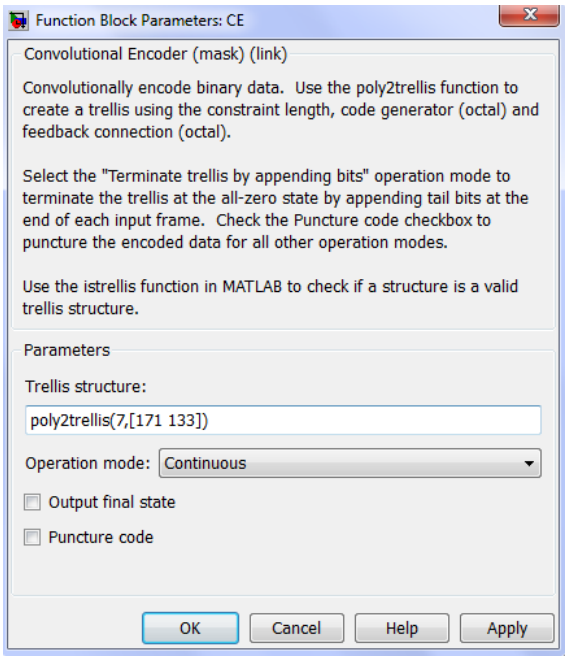


Figura 3.22 Parámetros del codificador convolucional.

Para sus entradas y salidas los puertos soportan datos de tipo double, single, boolean, int8, uint8, int16, uint16, int32, uint32, and ufix1.Los tipos de datos del puerto se heredan de las señales que maneja el bloque.

Para la generación del código VHDL del Codificador Convolucional, se creó una carpeta en el disco local C de la PC y una vez realizado el diagrama general, fue salvado en dicha carpeta. En la Figura 3.22 se observa la ventana Configuration Parameters de la capa física IEEE 802.16D en la sección HDL Code Generation, la diferencia apreciable en esta ventana con respecto a la aplicación anterior es que se escogido el bloque del codificador convolucional para la generación de código HDL, los demás campos de la ventana han sido comentados con anterioridad y su configuración es igual.

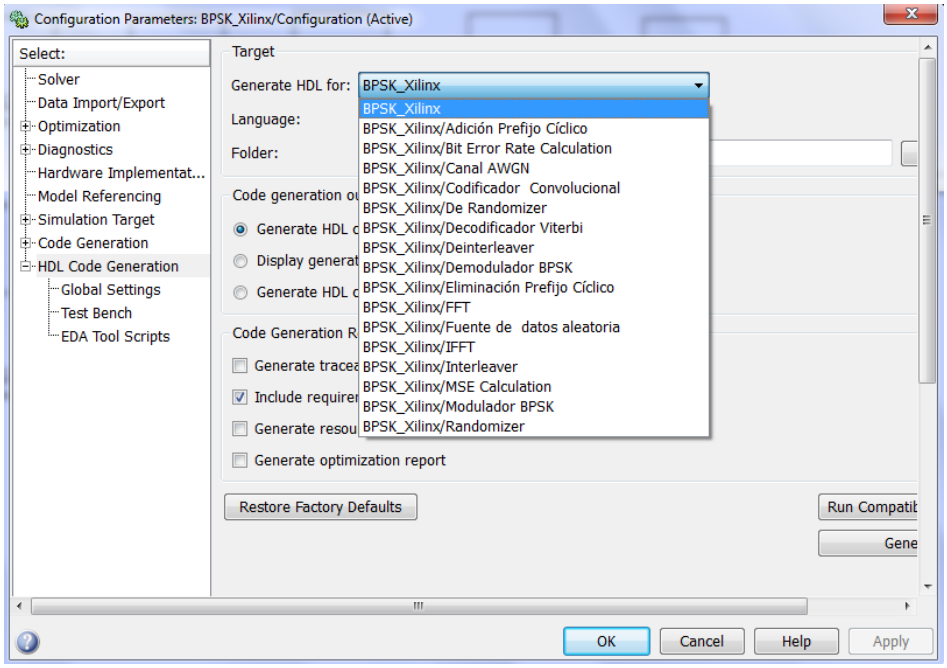


Figura 3.22Ventana para la configuración de parámetros del Codificador Convolucional.

El chequeo de compatibilidad, la generación del código VHDL, la configuración de los parámetros del test bench y generación del test bench se realiza de igual manera a la aplicación anterior.

Cuando el proceso de generación de código VHDL y el test bench en Matlab termina con éxito, se exportan al software Xilinx ISE los archivos con extensión .vhd, donde se siguen

cada una de las etapas del flujo de diseño. Estas etapas se realizaron de igual manera que con el bloque de la aplicación anterior, solo que ahora es para el codificador convolucional.

En la Figura 3.23 se muestra el proceso de chequeo de la sintaxis del código VHDL del codificador convolucional y en la Figura 3.24 se observan los resultados de la simulación.

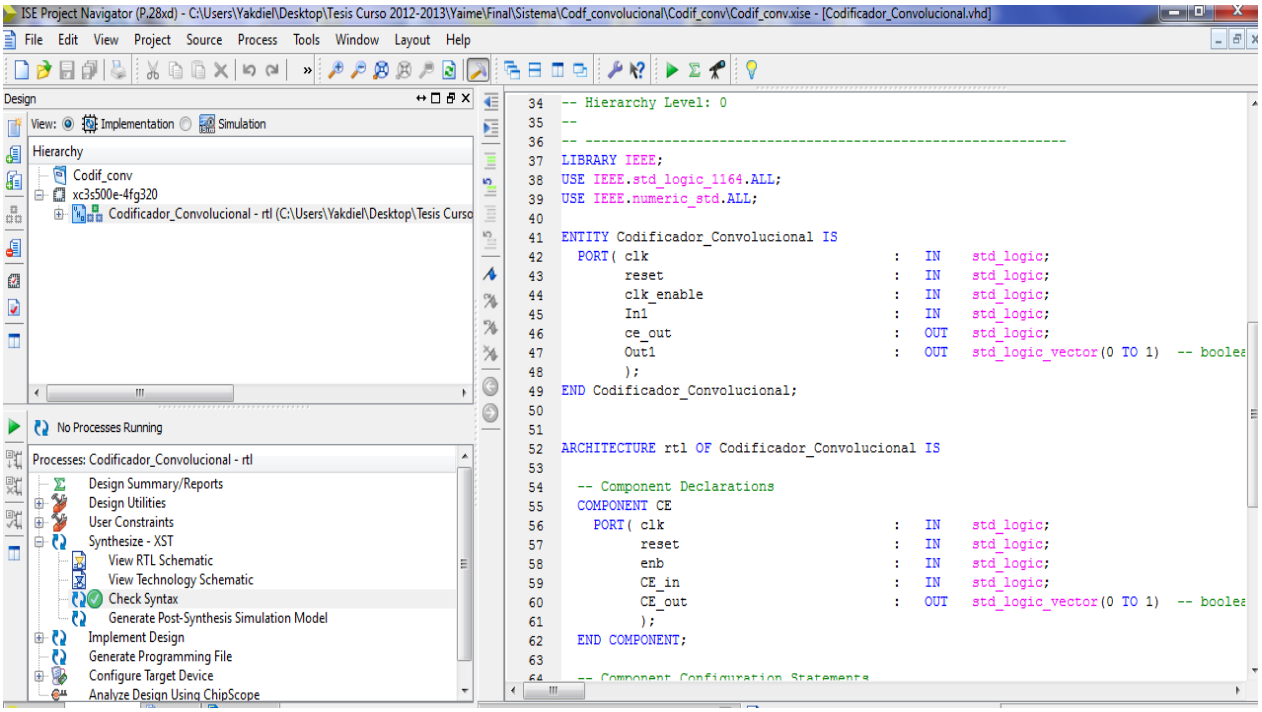


Figura 3.23 Proceso de chequeo de la sintaxis.

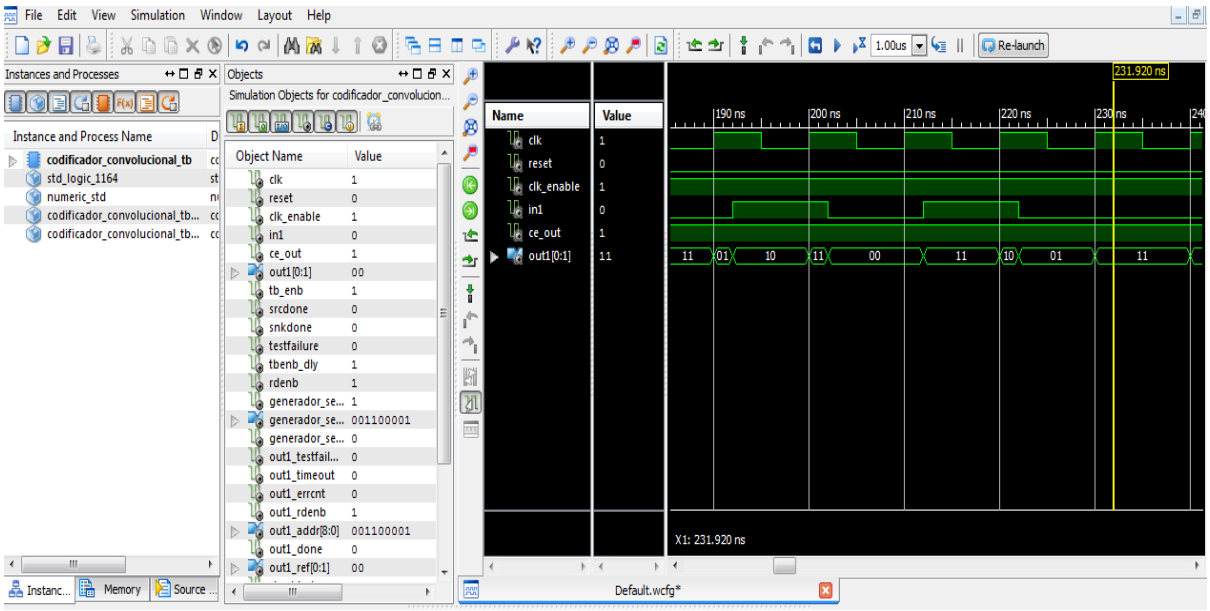


Figura 3.24 Resultados de la simulación del codificador convolucional utilizando el simulador ISim.

La simulación muestra como por cada bit de información se agrega un bit de redundancia.

El consumo de recursos en la tarjeta FPGA para esta aplicación se muestra en la Tabla siguiente:

Tabla3.2 Resumen sobre la utilización de dispositivos (valores estimados)

Utilización de la lógica	usado	disponible	utilizado
Número de slices	3	4656	0%
Número de slices flip flops	6	9312	0%
Número de LUTs de 4 entradas	3	9312	0%
Número de IOBs unidos	7	232	3%
Número de GCLKs	1	24	4%

El decodificador convolucional que se implementó sigue el algoritmo de Viterbi, según indica el estándar de la IEEE. El funcionamiento del bloque se gobierna por el parámetro modo de funcionamiento, que se refiere al método de transición entre tramas de entrada sucesivas (Figura 3.25).

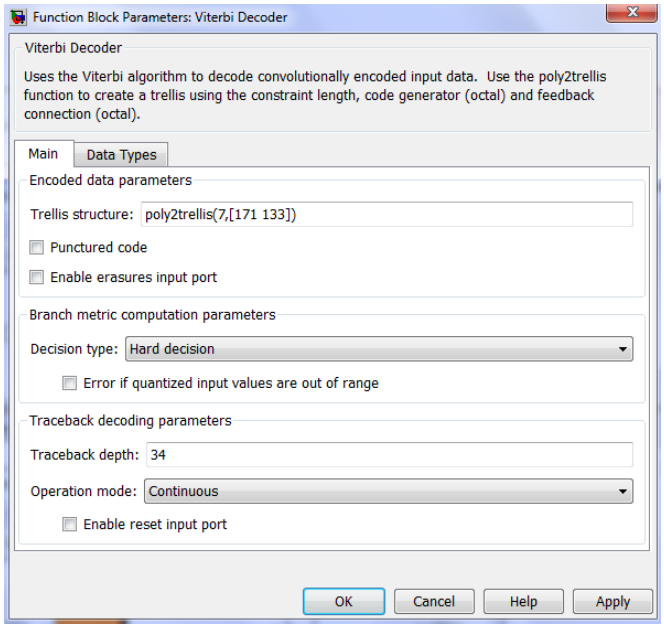


Figura 3.25 Parámetros del decodificador de Viterbi.

Los tipos de datos de las señales de salida pueden ser double, single, boolean, int8, uint8, int16, uint16, int32 y uint32.

Con la herramienta HDL Coder de Matlab, se obtuvo el código VHDL y el test bench del decodificador de Viterbi (Figura 3.26). Este código VHDL fue utilizado por el software Xilinx ISE para realizar todas las etapas del flujo de diseño, tratadas anteriormente. A continuación se muestra en la Figura 3.27 los resultados de la simulación, que muestran el correcto funcionamiento del bloque decodificador.

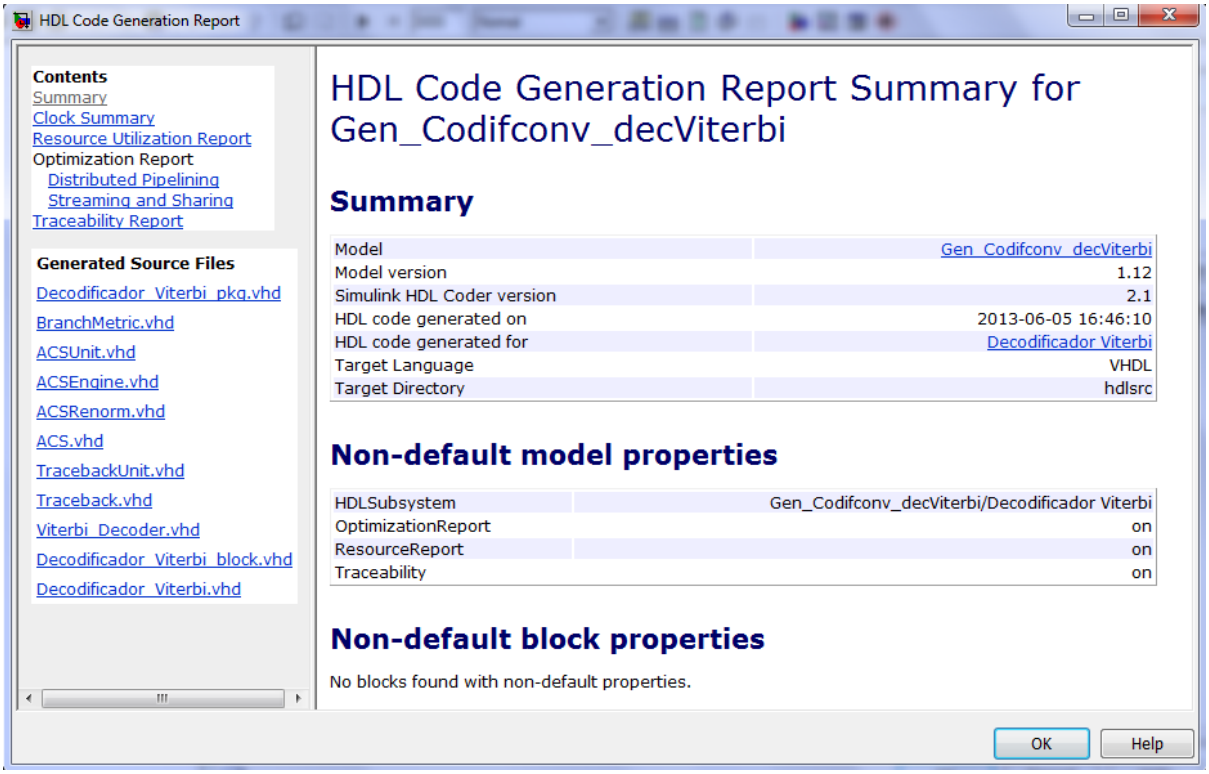


Figura 3.26 Resumen del reporte de la generacion de HDL del decodificador de Viterbi.

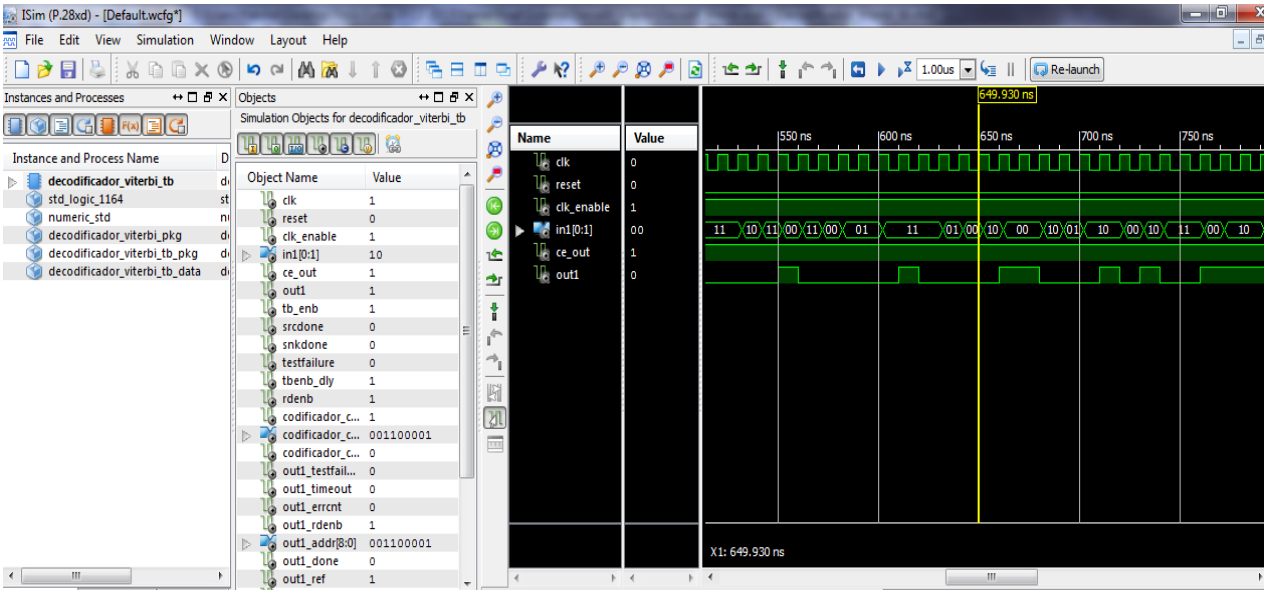


Figura 3.27 Resultados de la simulación del decodificador de Viterbi utilizando el simulador ISim.

El consumo de recursos en la tarjeta FPGA para esta aplicación se muestra en la Tabla siguiente:

Tabla3.3 Resumen sobre la utilización de dispositivos (valores estimados)

Utilización de la lógica	usado	disponible	utilizado
Número de slices	2623	4656	62%
Número de slices flip flops	3329	9312	35%
Número de LUTs de 4 entradas	2965	9312	31%
Número de IOBs unidos	7	232	3%
Número de GCLKs	1	24	4%

3.3 Comprobación del funcionamiento del codificador convolucional y el decodificador de Viterbi

En este epígrafe se realiza la implementación en el Kit de desarrollo Nexys2 de un diseño simple, conformado por un generador de secuencia, un codificador convolucional y un decodificador de Viterbi (Figura3.28). Estos dos últimos bloques mencionados, fueron utilizados en epígrafes anterior por lo que se conoce su funcionamiento, parámetros y se cuenta con sus códigos VHDL y test bench, obtenidos en Matlab y sintetizados con la ayuda de Xilinx ISE.

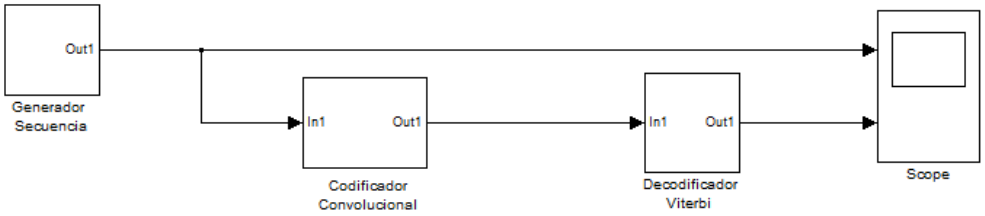


Figura 3.28Esquema del diseño a implementar en el Kit Nexys2.

El primer bloque del circuito de la Figura 3.28 es un Generador de Secuencia, que genera una secuencia de ruido de seudo aleatoria, utilizando un LFSR (Linear Feedback Shift

Register). Uno de sus parámetros es el Generator polynomial que especifica valores de las conexiones del registro de cambio, se pueden entrar estos valores como un vector binario. En la Figura 3.29 se muestra el resto de los parámetros.

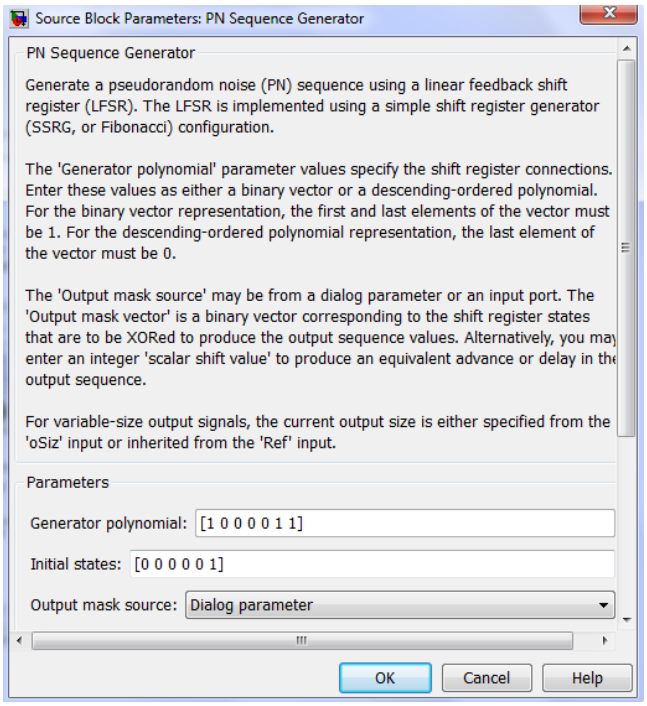


Figura 3.29 Parámetros del Generador de secuencia.

En el diseño de la Figura 3.28 se colocó un scope (osciloscopio), bloque de Simulink para ver el comportamiento de la señal a la salida del generador de secuencia y del circuito. El resultado debe ser señales idénticas, lo que desfasadas en el tiempo, debido a la demora que introducen los bloques intermedios, como se muestra en la Figura 3.30.

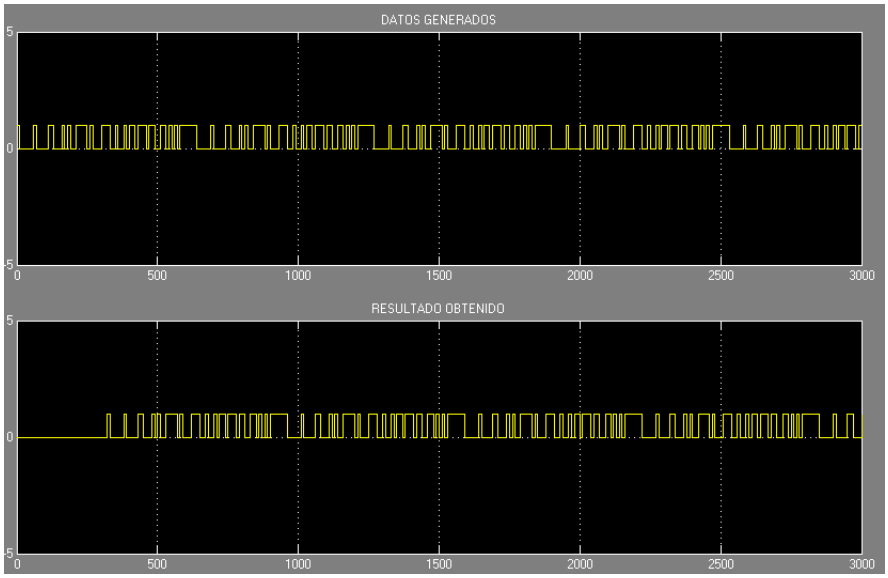


Figura 3.30 Señales a la salida del generador de secuencia y del circuito.

Después de conocido el comportamiento del diseño y los parámetros del bloque generador de secuencia, para proceder con la implementación es necesario obtener el VHDL de dicho bloque, para esto se utilizó la herramienta Simulink HDL Coder de Matlab.

En la Figura 3. 31 se muestra el código VHDL, como parte del reporte sobre la generación de HDL del generador de secuencia.

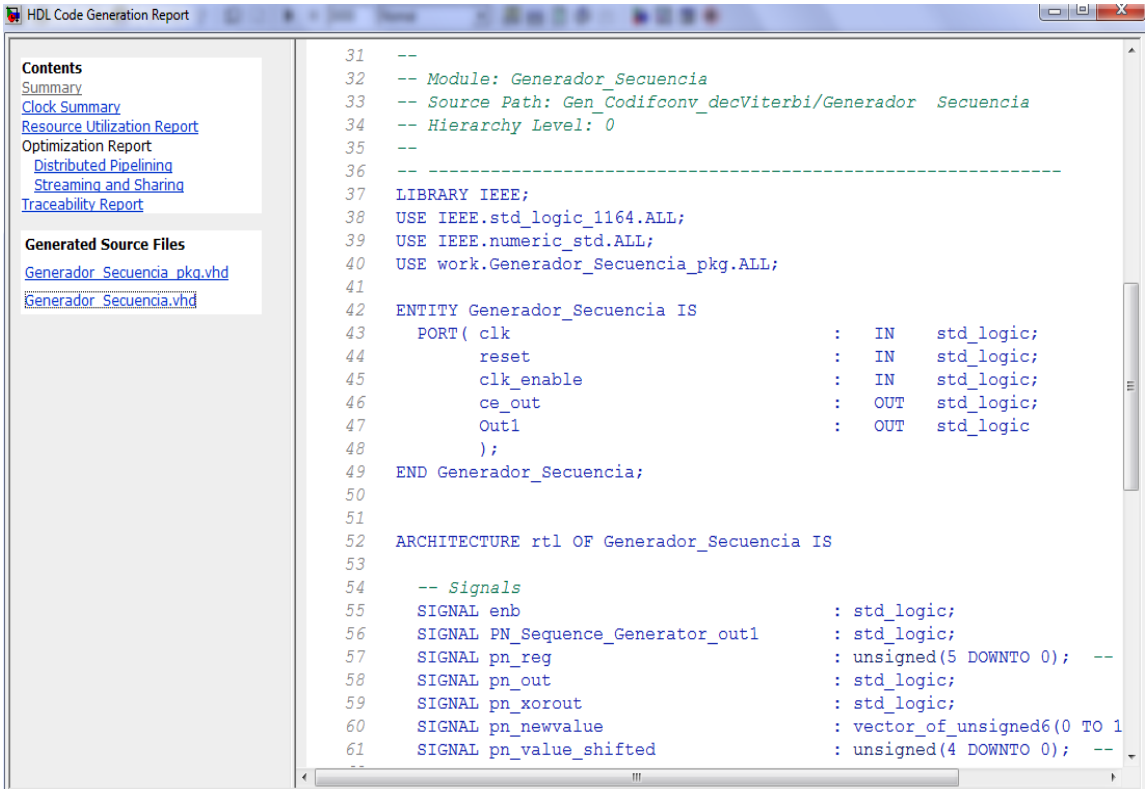


Figura 3.31 Reporte de la generación de código HDL del Generador de Secuencia.

A continuación se describe el trabajo con Xilinx ISE, que comienza con la creación de un nuevo proyecto. En la ventana New Project Wizard, se creó el proyecto Sistema General, se seleccionó la el FPGA Spartan 3E de 500 mil compuertas y encapsulado FG320 a una velocidad de -4. Además se declaró el uso del simulador ISim y la herramienta de síntesis XST, entre otros parámetros.

En la Figura 3.32 se muestra la ventana Adding Source Files, donde se añade el archivo con extensión .vhd del test bench del generador de secuencia al nuevo proyecto de Xilinx.

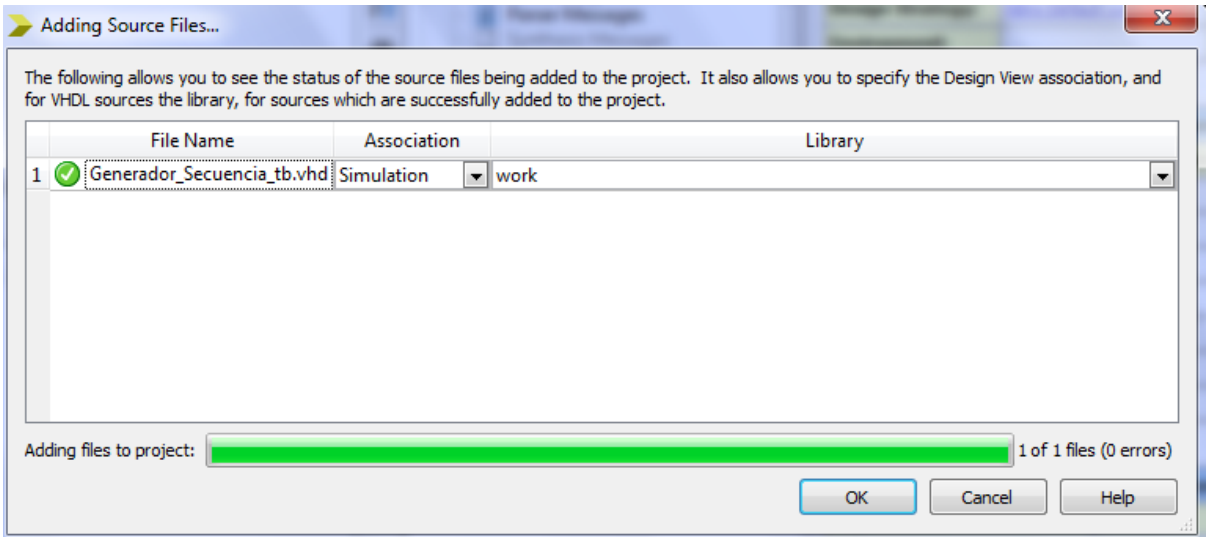


Figura 3.32 Estado de los archivos fuentes al ser agregados al proyecto.

Añadidos los archivos fuentes, se realizó la revisión de la sintaxis y la simulación del diseño, a través del simulador ISim. En la Figura 3.33 se muestran los resultados de la simulación, observándose como la salida, denominada out1, no se repite en el tiempo. Es aleatoria.

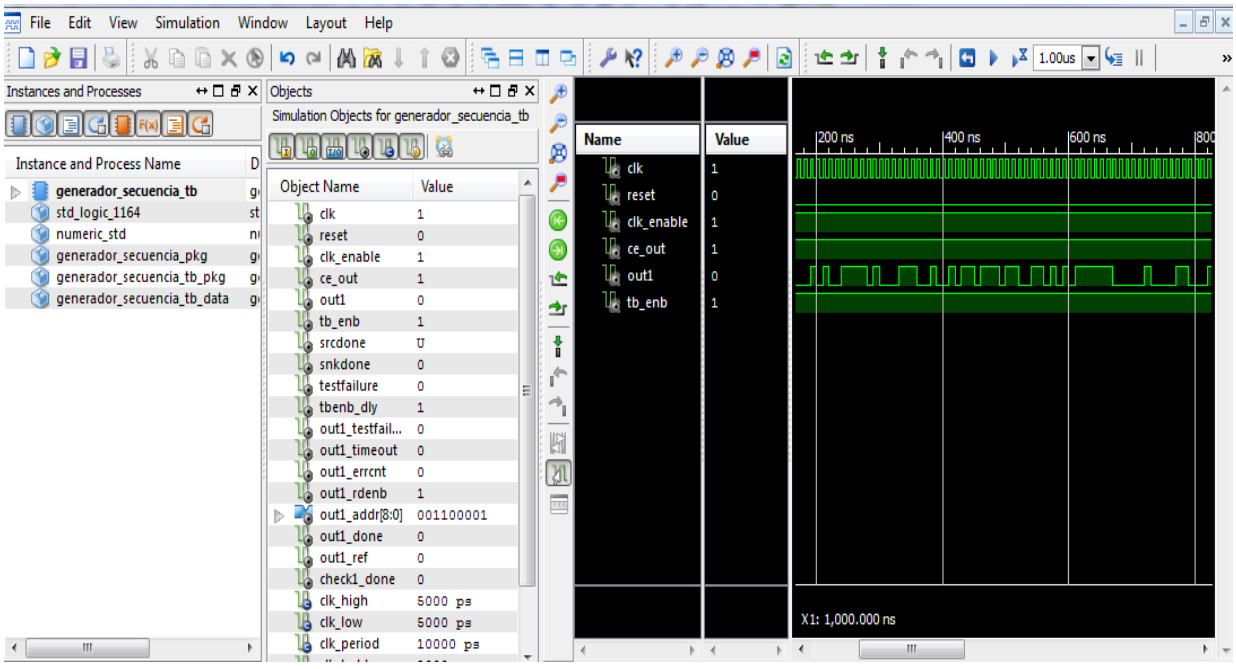


Figura 3.33 Resultados de la simulación del generador de secuencia utilizando el simulador ISim.

En la parte inferior de la ventana principal de Xilinx ISE, la opción Design Summary brinda un resumen del consumo de recursos en la tarjeta FPGA para esta aplicación, que se muestra en la Tabla siguiente.

Tabla3.4 Resumen sobre la utilización de dispositivos (valores estimados)

Utilización de la lógica	usado	disponible	utilizado
Número de slices	3	4656	0%
Número de slices flip flops	6	9312	0%
Número de LUTs de 4 entradas	1	9312	0%
Número de IOBs unidos	5	232	2%
Número de GCLKs	1	24	4%

Una vez revisado el diseño y logrados los resultados esperados se puede crear un símbolo. El símbolo consistirá en un nuevo componente donde el usuario sólo verá un bloque con el mismo número de entradas y de salidas que el esquema original, que englobará a éste y realizará su misma función. Este símbolo que se crea, pasa a formar parte de la librería de componentes del proyecto. La realización de símbolos es una herramienta muy útil a la hora de diseñar circuitos más complicados y que además tengan partes que se repitan. Facilita el diseño jerárquico, permitiendo una mayor claridad a la hora de presentar un circuito y entender su funcionamiento. En la Figura 3.34 se observa el proceso de creación del símbolo esquemático.

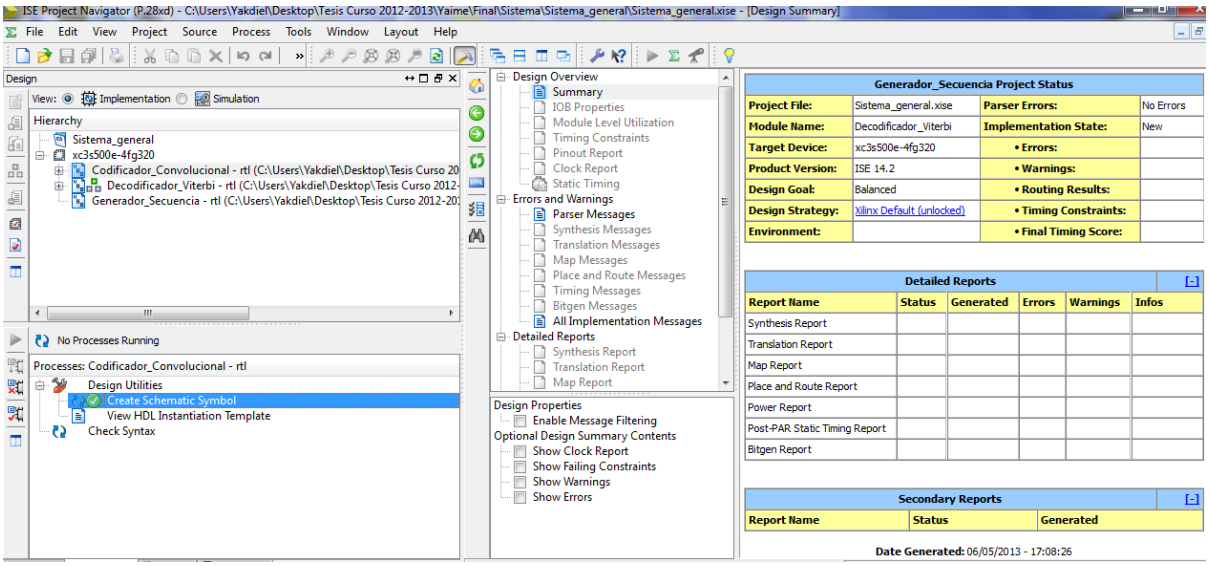


Figura 3.34 Proceso de creación del símbolo.

En la Figura 3.34 se observa además, como el archivo del decodificador de Viterbi es el principal, es decir, no existe una instanciación entre los archivos correspondientes a los tres bloques. La instanciación de los bloques del circuito es lograr una interconexión entre ellos, integrar los bloques para que funcionen como un todo. Se puede determinar por la Figura 3.34 si un archivo es principal o no, por el símbolo de tres cuadritos pequeños organizados en forma de triángulo junto al fichero.

En la Figura 3.35 se muestra la instanciación del Sistema General y el esquemático del circuito. El primer bloque que forma el esquemático es un reductor de frecuencia, porque en él entra la señal de reloj de la tarjeta Nexys2 de 50MHz y sale reducida a 1KHZ, que es la que entra al resto de los bloques. La programación VHDL de este bloque se realizó manualmente, y el valor de un 1KHz de frecuencia se escogió de manera arbitraria, para que la tarjeta del Kit Nexys2 no trabajara al máximo de su capacidad, 50MHZ, y poder visualizar las señales en el osciloscopio Rigol. La señal de Reset es activa en “0” y el Habilitador en “1”.

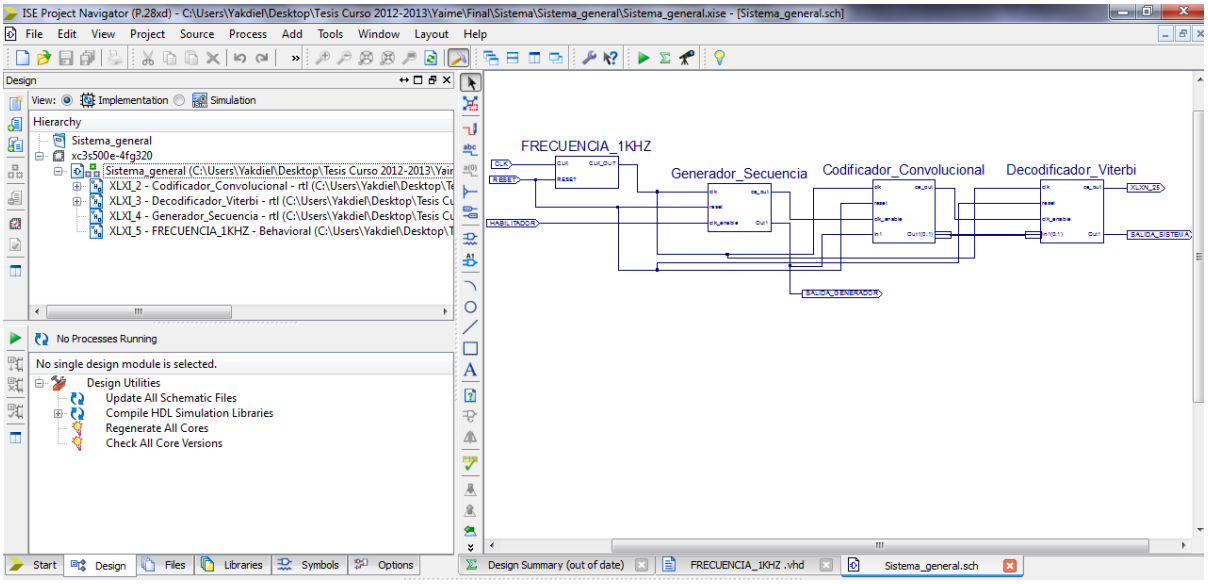


Figura 3.35 Esquemático del Sistema General.

A continuación se realiza el proceso de síntesis del sistema con el empleo de la herramienta XST (Figura 3.36).

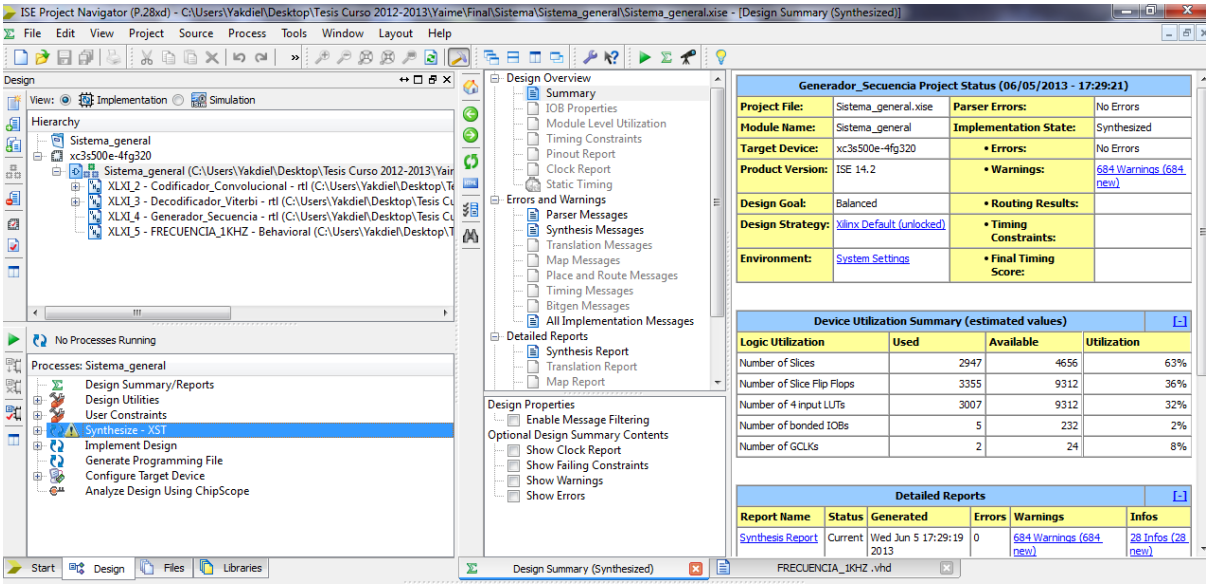


Figura 3.36 Proceso de síntesis del Sistema General utilizando la herramienta XST.

El consumo de recursos en la tarjeta FPGA para esta aplicación se muestra en la Tabla 3.5.

Tabla3.5 Resumen sobre la utilización de dispositivos (valores estimados)

Utilización de la lógica	usado	disponible	utilizado

Número de slices	2947	4656	63%
Número de slices flip flops	3355	9312	36%
Número de LUTs de 4 entradas	3007	9312	32%
Número de IOBs unidos	5	232	2%
Número de GCLKs	2	24	8%

Una vez completado el proceso de síntesis pueden especificarse una serie de restricciones físicas y temporales para poder sintetizar el diseño de manera satisfactoria. En un proyecto con ISE existe una gran variedad de métodos para añadir restricciones. Una de las restricciones es la asignación de pines, que permite asociar pines del FPGA con entradas y salidas específicas del diseño. Esto se logra mediante el subprograma PlanAhead. El mismo tiene un ambiente como el mostrado en la Figura 3.37, y crea un fichero de extensión .ucf que especificará las restricciones impuestas. Para realizar estas asignaciones de pines hay que tener en cuenta el kit de desarrollo que posee la FPGA en la cual se va a sintetizar el diseño ya que en éste se especifican cada uno de los dispositivos que se pueden utilizar y su ubicación.

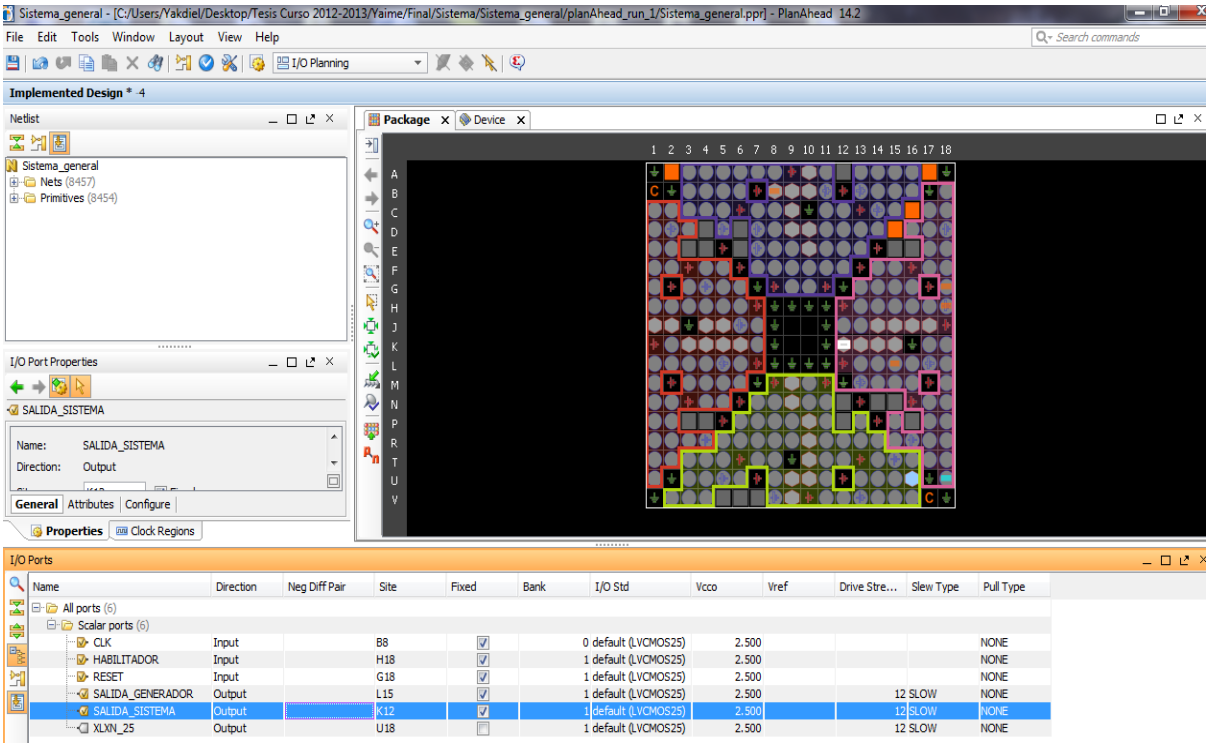


Figura 3.37Asignacion de pines utilizando PlanAhead.

Además el FPGA de la tarjeta Nexys2 debe ser configurada (o programada) por el usuario antes de que pueda realizar cualquier función. Se puede programar de varias maneras: de una PC, usando el puerto USB de la tarjeta o el conector JTAG, o desde una "Platform Flash ROM" en la tarjeta (esta última es también programable vía puerto USB). Un jumper determina qué fuente (PC o ROM) se utilizará para cargar la configuración. La Figura 3.38 muestra el circuito de programación de la tarjeta.

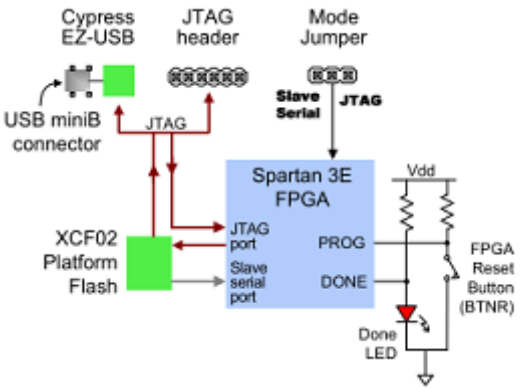


Figura 3.38 Circuito de programación del Kit Nexys2 (Digilent, 2008).

Para la configuración del FPGA en este trabajo se escogió el primer método por ser sencillo, solo demanda de un cable USB del ordenador a la tarjeta (Figura 3.39) y un software para cargar el fichero. Se utilizó Digilent Adept. No se utilizó el conector JTAG debido a que no existe en la Facultad. Luego de conectar el FPGA al ordenador, el software la reconoce y permite buscar el fichero de configuración (con extensión ".bit"), que una vez cargado en el dispositivo programable, lo hará trabajar conforme a las especificaciones del diseño (Figura 3.40).

Figura 3.39 Programación del FPGA de la tarjeta Nexys2.

Figura 3.40 Sistema General implementado en el Kit Nexys2.

En la Figura 3.40 se muestran el Kit Nexys2 conectado a un osciloscopio, para observar la salida del circuito y del generador de secuencia, mostrando ondas cuadradas idénticas, con

un desfasaje de tiempo introducido por los bloques intermedios, comprobando los resultados obtenidos anteriormente en el Matlab, con el bloque se Simulink scope.

3.4 Conclusiones Parciales

En este capítulo se realizaron aplicaciones de SDR en Matlab, a las cuales seles generó su código VHDL utilizando la herramienta Simulink HDL Coder, y se realizaron las etapas del flujo de diseño con Xilinx ISE. Además se comprueba el comportamiento y desempeño del codificador convolucional y del decodificador de Viterbi junto al generador de secuencia.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

En el presente trabajo se desarrollaron aplicaciones de la Radio Definida por Software utilizando el software Matlab y Xilinx ISE, y se comprobó el funcionamiento de un codificador convolucional y un decodificador de Viterbi, junto a un generador de secuencia, en un FPGA Spartan-3E de Xilinx. Durante su realización se arribó a las siguientes conclusiones:

- La Radio Definida por software es una tecnología de rápida evolución, que promete funcionalidad múltiple, relativa a modos de operación y bandas de frecuencias de los dispositivos inalámbricos, mediante actualizaciones del software. Aunque, los software para realizar determinadas funciones, como por ejemplo en equipos de radio bases de telefonía móvil, pueden ser complejos.
- Los sistemas SDR se enfocan en la construcción de equipos de comunicación reconfigurables, por lo que los FPGA son los dispositivos lógico programables utilizados con mayor frecuencia, debido a las facilidades que brinda su arquitectura interna. La evolución de la tecnología SDR y la de los dispositivos lógicos programables deben coexistir de manera paralela.
- MatLab/Simulink es un software que brinda variedad de prestaciones para los programadores, lo cual se evidenció en la obtención de código VHDL de las aplicaciones desarrolladas.
- Los pasos presentados para trabajar con la herramienta Simulink HDL Coder de Matlab muestran una vía eficiente de obtener código HDL de aplicaciones de la SDR.

- Los resultados satisfactorios alcanzados en las simulaciones, así como los obtenidos en la implementación en el Kit de desarrollo Nexys2, corroboraron la efectividad del software Matlab en la generación de código HDL para aplicaciones de la SDR.

Recomendaciones

- Seguir investigando la utilización de herramientas de alto nivel que brinden la posibilidad de generar códigos HDL, como es el caso de Matlab, para el desarrollo de la Radio Definida por Software.

REFERENCIAS BIBLIOGRÁFICAS

Aguilar JH and Navarro A (2011) Radio cognitiva–Estado del arte. Available at: http://bibliotecadigital.icesi.edu.co/biblioteca_digital/handle/10906/5283 (accessed 10/05/13).

Barrios JP (2013) *Diseño Digital*. [Online] Available at: http://digital.fie.uclv.edu.cu/bienvenida.htm (accessed 17/06/13).

Belanger L (2007) Advanced SDR platform eases multiprotocol radio development. *RF DESIGN*. 30 (1), 36.

Blossom E (2004) Exploring GNU radio. *Online][Cited: 10 January 2009.] http://www. gnu. org/software/gnuradio/doc/explorin g-gnuradio. html.*

Bozich EC (2005) *Introducción a los Dispositivos FPGA. Análisis y ejemplos de diseño*. La Plata, Argentina.

Cepero A (2012) *Desarrollo de aplicaciones utilizando PicoBlaze en System Generator y su implementación en una FPGA de Xilinx*. Santa Clara, Cuba: UCLV.

Chen C-Y, Tseng F-H, Chang K-D, Chao H-C and Chen J-L (2010) Reconfigurable software defined radio and its applications. *Tamkang Journal of Science and Engineering*. 13 (1), 29г38.

Chen K-C and Prasad R (2009) *Cognitive radio networks*. Wiley Available at: http://books.google.com/books?hl=en&lr=&id=QJs2rUdsT2cC&oi=fnd&pg=PR5&dq=%22Cognitive+Radio+Networks%22+%22Kwang-Cheng+Chen%22&ots=VTX0wK4tjL&sig=qBPZsAR0Ucyuu5ySojCSUQgMSLY (accessed 17/06/13).

Chu P (2006) *RTL hardware design using VHDL: coding for efficiency, portability, and scalability*. Wiley-Interscience Available at: http://books.google.com/books?hl=en&lr=&id=gVd2yeFHshUC&oi=fnd&pg=PR1&dq=%22RTL+HARDWARE+DESIGN+USING+VHDL%22&ots=7UyObPazuh&sig=RiqXEoso1h3Gottb6BHWEf_rttM (accessed 17/06/13).

Clermidy F, Lemaire R, Popon X, Ktenas D and Thonnart Y (2009) An open and reconfigurable platform for 4G telecommunication: Concepts and application. In: *Digital System Design, Architectures, Methods and Tools, 2009. DSD'09. 12th Euromicro Conference on*. 449–456. Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5350210 (accessed 17/06/13).

Digilent I (2008) Digilent Nexys2 Board Reference Manual.

Fette BA (2009) *Cognitive radio technology*. Academic Press Available at: http://books.google.com/books?hl=en&lr=&id=81q5AMSAMzgC&oi=fnd&pg=PP2&dq=%22Cognitive+Radio+Technology%22+%22bruce%22&ots=vhPnLK_-hV&sig=TRoTJXUsxi5TSqHdcp-XFdd-YQk (accessed 10/05/13).

García CM (2012) *Radio Definido por Software usando MATLAB*. Santa Clara, Cuba: UCLV.

Gilman AS (1986) VHDL-The Designer Environment. *Design & Test of Computers, IEEE*. 3 (2), 42–47.

IEEE Xplore (2012) *IEEE Xplore - Software defined radio on digital communications: A new teaching tool*. [Online] Available at: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6208436&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6208436 (accessed 07/09/12).

Kempf T, Witte EM, Ramakrishnan V, Ascheid G, Adrta M and Antweiler M (2006) An SDR implementation concept based on waveform description. *Frequenz*. 60 (9-10), 171–174.

LOUIS A (2011) *Aplicaciones de la tecnología de Radio definida por Software en los Sistemas Móviles*. Santa Clara, Cuba: UCLV.

Marrero L (2011) *Modelación de Circuitos Digitales Secuenciales Típicos utilizando MATLAB y SIMULINK*. Santa Clara, Cuba: UCLV.

MathWorks (2013) *MathWorks - Worldwide Offices & Representatives*. [Online] Available at: http://www.mathworks.com/company/worldwide/index.html (accessed 10/05/13).

Mohebbi B, Maestre R, Davies M and Kurdahi FJ (2003) A case study of mapping a software-defined radio (SDR) application on a reconfigurable DSP core. In: *Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*. 103–108. Available at: http://dl.acm.org/citation.cfm?id=944676 (accessed 17/06/13).

Robles OI (2011) Diseño e implementación de un modulador OFDM reconfigurable para la tecnología software-defined radio sobre un FPGA. Available at: http://tesis.pucp.edu.pe/repositorio/handle/123456789/270 (accessed 17/06/13).

Rodríguez Y (2011) *Implementación de la capa física WirelessMan-OFDM del estándar*. Santa Clara, Cuba: UCLV.

Romero B (2012) *Contribución al estudio y aplicación de la radio definida por software en Cuba*. La Habana, Cuba: CUJAE.

SÁNCHEZ F (2010) *Diseños de circuitos digitales con VHDL*.

SDR Forum (2013) *Software Defined Radio (SDR), Cognitive Radio (CR), and Dynamic Spectrum Access (DSA)*. [Online] Available at: http://www.wirelessinnovation.org/ (accessed 17/06/13).

Sidhu GS, Kaushal S and Banga VK (2012) Performance Evaluation Of Wimax Network. *International Journal of Engineering*. 1 (8). Available at: http://www.ijert.org/browse/october-2012-edition?download=1367%3Aperformance-evaluation-of-wimax-network&start=140 (accessed 17/06/13).

Vallejo ML and Rodrigo JA (2004) FPGA: Nociones básicas e implementación. *Laboratorio de Diseño Microelectrónico, 4◦ Curso, P94*. Available at: http://cursoshistorico.iteso.mx/moodle/pluginfile.php/68670/mod_resource/content/0/fpga_a2_2004.pdf (accessed 10/05/13).

Valverde C (2010) Implementación de un sistema OFDM en un dispositivo SFF SDR. Available at: http://e-archivo.uc3m.es:8080/handle/10016/11062 (accessed 17/06/13).

Vargas CV, Lopez WE and da Rocha CF (2007) Sistemas de Comunicación Inalámbrica MIMO-OFDM. Available at: http://www.revistasbolivianas.org.bo/scielo.php?script=sci_arttext&pid=S1683-07892007000200009&lng=pt&nrm=iso?iframe=true (accessed 17/06/13).

Xilinx I (2012) *All Programmable Technologies from Xilinx Inc.* [Online] Available at: http://www.xilinx.com/ (accessed 05/07/12).

Zamora R (2010) *Metodología para el diseño de aplicaciones medianas en FPGAs de Xilinx*. Santa Clara, Cuba: UCLV.

ANEXOS

ANEXO I Asignación de pines de los conectores Pmods (Digilent, 2008).

Pmod J		**Pmod JB**		**Pmod JC**		**Pmod JD**	
JA1: L15	JA7:K13	JB1:M13	JB7: P17	JC1: G15	JC7:H15	JD1: J13	JD7:K
JA2: K12	JA8:L16	JB2: R18	JB8: R16	JC2: J16	JC8: F14	JD2:M18	JD8:K
JA3: L17	JA9:M14	JB3 : R15	JB9: T18	JC3: G13	JC9:G16	JD3: N18	JD9: J
JA4: M15	JA10:M16	JB4: T17	JB10: U18	JC4: H16	JC10: J12	JD4: P18	JD10: J