IMPLEMENTACIÓN DE CÓDIGOS DE LÍNEA EN UNA TARJETA DE ENTRENAMIENTO BASADA EN UN FPGA

Checa Romo Viviana Elizabeth, Ing.

Velásquez Córdova José Daniel, Ing.

Álvarez Rueda Robin, PhD.

Escuela Politécnica Nacional
Isabel La Católica 202 y Veintimilla, Quito-Ecuador

***Resumen*— Se describe la implementación de códigos de línea con el lenguaje de programación VHDL en el FPGA de la Spartan-3E Starter Kit Board. En primer lugar se realiza una introducción a los FPGAs y la Spartan-3E Starter Kit Board; también se desarrolla un estudio preliminar de la síntesis, simulación y programación en Xilinx ISE; y se proporciona un panorama de VHDL. Posteriormente se detalla cada código de línea implementado, además se visualizan simulaciones de la codificación en el software Matlab. Luego se describe la elaboración de una interfaz gráfica en Matlab, la misma que permite: introducción de datos, configuración del puerto de comunicación y velocidad de transmisión, escoger el tipo de código de línea a implementarse, observar las señales enviada y recibida posterior a la decodificación en el FPGA, además de desplegar información adicional. A continuación se explica la implementación de cada código de línea en VHDL y se muestra la simulación en Testbench. También se detalla el diseño de un circuito interpretador unipolar-bipolar y un circuito interpretador bipolar-unipolar. Finalmente se visualizan los resultados obtenidos en el FPGA con un osciloscopio que son semejantes a los obtenidos en Matlab.**

***Términos para indexación*— FPGA, Spartan-3e starter kit board, VHDL, códigos de línea, Interfaz gráfica, Matlab.**

Checa R. Viviana E. E-mail: vivi_chk@yahoo.com
Velásquez José Daniel. E-mail: jdanivelasquez@hotmail.com
Álvarez Robin, profesor a tiempo completo de la Facultad de Ingeniería Eléctrica y Electrónica de la Escuela Politécnica Nacional. E-mail: robin.alvarez@epn.edu.ec

I. INTRODUCCIÓN A LOS FPGAs

Un Dispositivo lógico programable o PLD, es un circuito integrado formado por cierto número de compuertas lógicas y/o módulos básicos cuyas conexiones pueden ser personalizadas o programadas por el usuario final para construir circuitos digitales reconfigurables.

Los PLDs se clasifican en: SPLDs (Dispositivo lógico programable simple), CPLDs (Dispositivo Lógico Programable Complejo), y FPGAs (Matrices de Compuertas Programables en Campo).

FPGA, Field Programmable Gate Array, es un PLD, que tiene capacidad lógica muy alta debido a que posee un arreglo bidimensional de bloques lógicos comunicados por medio de una matriz de cables e interruptores cuya interconexión y funcionalidad son programables por los usuarios. El tamaño, estructura, número de bloques, cantidad de conexiones y forma de conectividad difieren de una arquitectura a otra.

La arquitectura de los FPGAs de la familia Spartan 3E consta de 5 elementos fundamentales programables: CLBs (Bloques Lógicos Configurables), IOBs (Bloques de Entrada/Salida), Bloques RAM, Bloques Multiplicadores, DCMs (Administradores de Reloj).

En el mercado se ofertan diferentes tarjetas de evaluación y desarrollo, las cuales incluyen un FPGA y una diversidad de periféricos. Spartan-3E Starter Kit Board es una tarjeta de entrenamiento que dispone principalmente de los componentes: FPGA Spartan-3E XC3S500E de Xilinx, Flash PROM de 4 Mb, CPLD CoolRunner XC2C64A, 4 interruptores deslizantes, 3 entradas para señal de reloj, dos conectores seriales RS-232 (un DB9 hembra y un DB9 macho), una variedad de

conectores de expansión, tres conectores periféricos de 6 pines de entrada y salida.

Xilinx ISE, es un conjunto de herramientas de desarrollo elaborado por Xilinx para facilitar la creación de diseños. El flujo de diseño utilizando el software Xilinx ISE de un módulo hardware sobre un FPGA comprende los siguientes pasos: creación del modelo, síntesis del diseño, implementación del diseño y programación del dispositivo.

HDL (Lenguaje de Descripción de Hardware) define las funciones de los CLBs y las conexiones entre ellos. **VHDL** significa Very high speed integrated circuit Hardware Description Language y es un HDL.

Un código VHDL autónomo está compuesto de 3 partes fundamentales: Declaración de Librerías, Entidad y Arquitectura. La Declaración de Librerías es una colección de partes de código usadas comúnmente. En la declaración se debe incluir todas las librerías que van a ser utilizadas en el diseño. La Entidad especifica todos los puertos de entrada y salida. La Arquitectura contiene el código que describe el comportamiento del circuito.

En VHDL se puede construir: Código Concurrente, código que se ejecute en forma paralela; y Código Secuencial, también llamado código de comportamiento, las secciones de código que se ejecutan en forma secuencial son: procesos, funciones y procedimientos.

II. CÓDIGOS DE LÍNEA

Un **Sistema de Comunicaciones** tiene el propósito de transmitir información desde un emisor hasta un receptor a través de un canal.

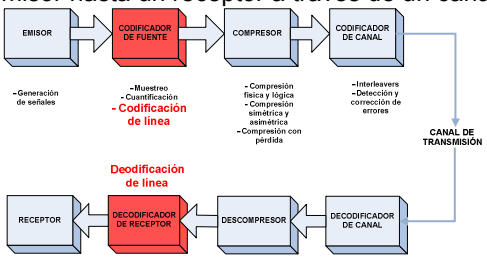

Figura 1: Diagrama de bloques de un Sistema de Comunicaciones.

Dentro del esquema que se presenta en la figura 1, los **Códigos de Línea** se encuentran en el bloque *Codificador de Fuente*, donde desempeñan la función de transformar los símbolos que emite el Emisor en símbolos de un código binario más adecuado para ser transmitido a través de un canal de comunicaciones.

En el presente proyecto se ha considerado desarrollar los códigos de línea que se muestran en la Tabla 1, relacionados a su polaridad.

Tabla 1: Códigos de Línea considerados en el presente proyecto de titulación relacionados a su polaridad.

Códigos unipolares	➢ **NRZ** ➢ **RZ 50 %** ➢ **4B5B**
Códigos Polares	➢ **Diferencial tipo M (NRZI)** ➢ **Diferencial tipo S** ➢ **Bifase L o Manchester** ➢ **Manchester Diferencial** ➢ **CMI**
Códigos Bipolares	➢ **AMI** ➢ **HDB3** ➢ **MLT-3**

Todos los códigos presentados en la tabla anterior, han sido desarrollados en VHDL en base a su algoritmo de codificación y diagrama de flujo de la codificación en Matlab. El código Diferencial Tipo M se lo ha escogido como ejemplo en este documento.

Diferencial Tipo M

En el código diferencial tipo M el dígito binario 1L cambian el nivel de la señal (Si estuvo en –A cambiará a +A), en tanto que los 0L mantienen dicho nivel. El diagrama de flujo de la programación en Matlab se visualiza en la **Figura 2.**

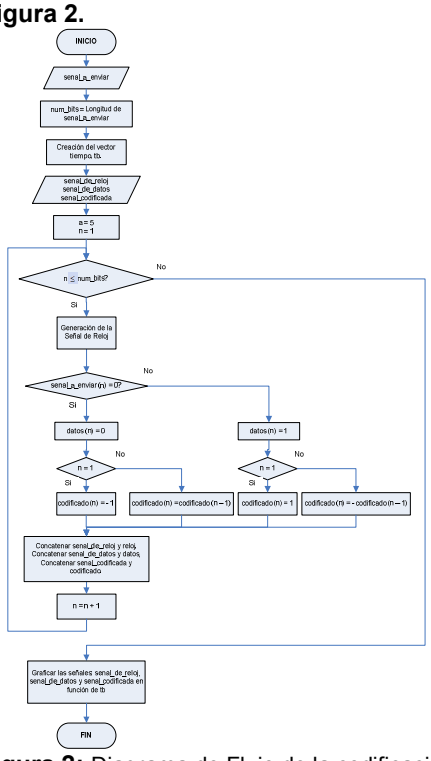

Figura 2: Diagrama de Flujo de la codificación Diferencial Tipo M en Matlab.

La simulación para éste código se la hace en base a los caracteres **a** y **b** que juntos en binario son: 1000011001000110.

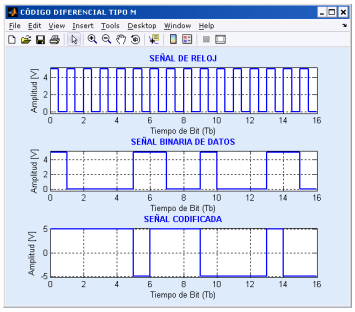

Figura 3: Codificación Diferencial Tipo M en Matlab.

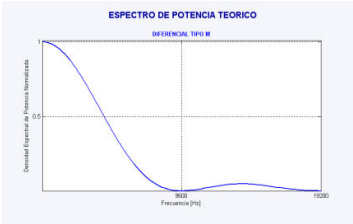

Figura 4: Espectro de potencia teórico para el código Diferencial Tipo M.

III. DESARROLLO DE LA INTERFAZ GRÁFICA

La Interfaz Gráfica se desarrolló en el Software de Matlab, en base a su herramienta GUIDE (Ambiente de desarrollo de Interfaces Gráficas de Usuario), ya que combina un ambiente gráfico con la programación en código.

El propósito de la realización de la Interfaz Gráfica es que el usuario disfrute de un entorno amigable, ante el ingreso de la secuencia de caracteres a ser codificada y el tipo de código de línea a implementarse en el FPGA. Estos caracteres se transmiten del PC al FPGA (Dispositivo que se encarga de la codificación y decodificación) en forma de bits a través del puerto RS-232 (Puerto de Comunicación de la Spartan-3E Starter Kit Board).

Posterior a la decodificación, el PC recibe los caracteres decodificados desde el FPGA, entonces la interfaz gráfica presenta: la secuencia de caracteres antes de ser codificados enviada al FPGA, la señal en forma de bits enviada, la secuencia de caracteres decodificados recibida desde el FPGA y la señal en forma de bits recibida.

El esquema de la Interfaz Gráfica se presenta en la Figura 5.

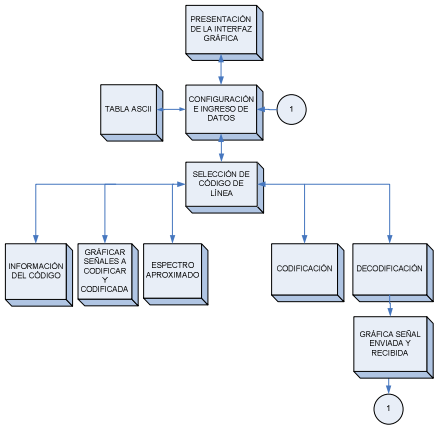

Figura 5: Diagrama de Bloques de la Interfaz Gráfica.

DESARROLLO DE LAS GUIs

Presentación de la Interfaz Gráfica

Al inicializar la Interfaz Gráfica, la primera ventana que se muestra es la de Presentación.

Figura 6: Presentación de la Interfaz Gráfica.

Configuración e Ingreso de Datos

Esta GUI permite al usuario seleccionar las opciones de configuración e ingresar los datos (caracteres) a ser codificados y decodificados en el FPGA.

Figura 7: Configuración e Ingreso de Datos.

Selección del Código de Línea

Para esta GUI, existen tres marcos específicos: **Generales, Simulación** y **Hardware.**

Figura 8: Selección del Código de Línea.

Cada uno de los botones que contiene esta GUI se describen a continuación:

Información de Código de Línea, este botón presenta una Interfaz Gráfica que visualiza una reseña del código de línea escogido.

Visualización del Espectro Teórico, abre una pantalla que muestra la Densidad Espectral de Potencia Teórica luego de la codificación con el código de línea seleccionado.

Graficar Señales a Codificar y Codificada, presenta una ventana que incluye tres gráficas: la señal de reloj, la señal a codificar (originada como resultado de ensamblar los identificadores de inicio y fin de datos y todos los bits de datos producidos por la secuencia de caracteres introducida), y la señal codificada con el código de línea seleccionado.

Enviar Datos al FPGA para la Codificación, al presionar este botón, el PC envía un flujo de bits que contiene un número binario que identifica el código de línea seleccionado que se desea implementar en el FPGA, seguido de los datos a ser codificados.

Ver datos recibidos Decodificados, al presionar este botón Matlab toma los datos binarios que llegan por el pin de recepción del puerto RS-232 y los muestra en las GUIs: Decodificación, y Gráfica de Señal Enviada y Recibida.

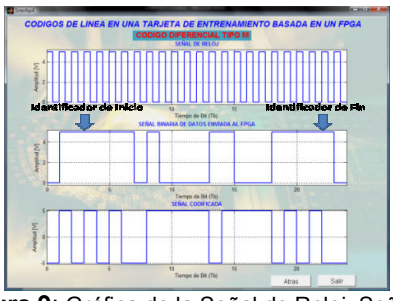

Figura 9: Gráfica de la Señal de Reloj, Señal a Codificar y Señal Codificada.

La Figura 9 muestra el caso específico cuando el carácter introducido por el usuario es “a” y el código de línea seleccionado es Diferencial Tipo M.

Decodificación

Esta pantalla muestra la secuencia de caracteres enviados y recibidos por el FPGA, y su respectivo valor en binario. Además se visualiza la velocidad de transmisión y la velocidad de señal.

Figura 10: Visualización de Caracteres Enviados y Recibidos del FPGA.

Gráfica de señales enviada y recibida

Permite visualizar la ventana de la Figura 11, que muestra tres gráficas: la señal de reloj, la señal enviada, y la señal recibida del FPGA. Esta ventana tiene tres botones: *Ingresar Nuevos Datos,* abre la ventana de Configuración e Ingreso de Datos con el propósito de iniciar una nueva codificación; *Atrás*, abre la ventana Decodificación; *Salir*, cierra la interfaz gráfica.

Figura 11: Gráfica de la Señal de Reloj, Señal Enviada y Señal Recibida del FPGA.

IV. IMPLEMENTACIÓN Y SIMULACIÓN DE LOS CÓDIGOS DE LÍNEA

Para la codificación y decodificación de datos, es necesario considerar los siguiente: Configurar la velocidad en el FPGA, es decir generar un reloj interno de frecuencia determinada que permita llevar a cabo procesos en VHDL; almacenar los datos después de su recepción en el FPGA para su posterior codificación; identificar el tipo de código de línea a implementarse; después del almacenamiento de datos serializarlos para que sean codificados, implementar un Interpretador Unipolar – Bipolar y un Interpretador Bipolar – Unipolar debido a las características de la Spartan – 3E Starter Kit Board (No trabaja con niveles de voltaje negativos); posterior a la decodificación,

ensamblar los datos para que sean enviados a la Interfaz Gráfica; enviar los datos decodificados desde el FPGA a Matlab para la respectiva visualización.

Banco de Trabajo

A continuación se presenta los elementos que conforman el banco de trabajo utilizado en el proyecto.

Hardware

- Tarjeta de entrenamiento SPARTAN 3E STARTER KIT BOARD
- Circuito externo interpretador Unipolar-Bipolar y Bipolar-Unipolar
- 2 Fuentes DC
- Osciloscopio Tektronix TDS 1002B
- Computador con Sistema operativo Windows XP con Service Pack 3, dos puertos USB, memoria RAM, mínimo de 512 Mbytes, y Procesador Pentium 4 o superior
- Cable USB-Serial

Software

- MATLAB 7.1
- Xilinx ISE 10.1

Figura 12: Banco de Trabajo.

Para los propósitos del proyecto, los periféricos utilizados de la tarjeta de entrenamiento son: el conector DB9 hembra (DCE), que permite la transmisión y recepción de datos; los interruptores deslizantes y los pines de entrada y salida adecuados.

La transmisión y recepción de datos entre MATLAB y el FPGA se la hace de manera asincrónica (no hay ninguna relación de tiempo entre MATLAB y el FPGA), debido a que este modo se usa típicamente en transmisión de códigos ASCII a través del puerto RS-232. El flujo de bits que envía Matlab hacia el FPGA está constituido por el siguiente esquema:

Figura 13: Transmisión de señales desde Matlab al FPGA.

Donde el *Stream de Datos* está constituido por el Identificador de Inicio de Datos, Identificador de Código, Datos (caracteres ASCII de 8 bits) e Identificador de Fin de Datos.

Diagrama de Bloques de la Implementación en VHDL

A continuación se muestra el esquema de implementación en VHDL del proyecto.

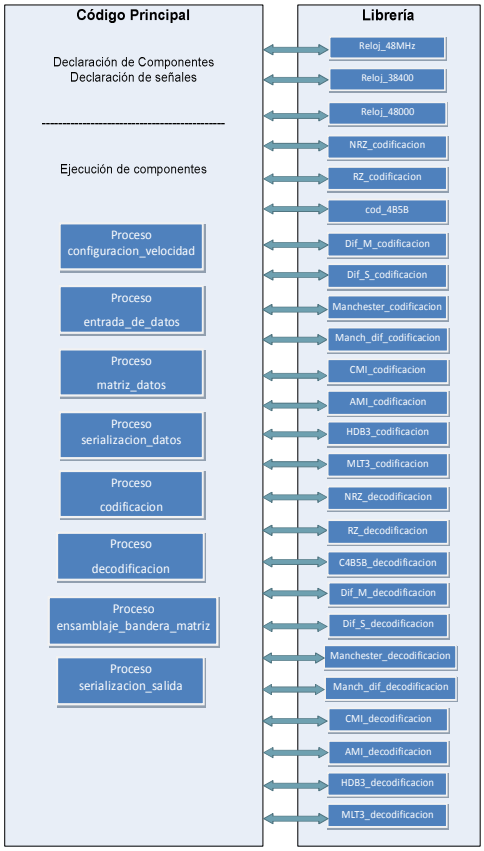

Figura 14: Diagrama de Bloques del programa principal y sus componentes en VHDL.

Componentes de Reloj

La tarjeta de entrenamiento dispone de un único reloj interno de 50 MHz, a partir de este se realizan las derivaciones requeridas por el proyecto.

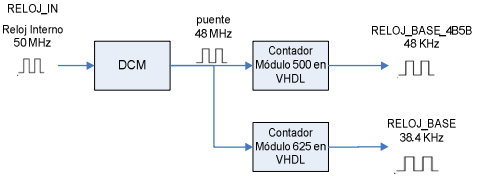

Figura 15: Señales de reloj obtenidas a partir del reloj interno que posee la Tarjeta de Entrenamiento.

Desarrollo de Procesos

Configuración de Velocidad

Este proceso tiene como finalidad obtener en el FPGA señales de reloj de frecuencia adecuada de acuerdo a la velocidad que el usuario ha seleccionado en la Interfaz Gráfica. Las señales de reloj *RELOJ_OUT* y *RELOJ_OUT_Doble* se obtienen a partir de la señal de reloj *RELOJ_BASE*, mientras que *RELOJ_OUT_4B5B* se obtiene a partir de *RELOJ_BASE_4B5B*.

Entrada de Datos

Este proceso realiza una recepción de datos serial, el flujo de bits de datos consiste en 11 bits por carácter, el primero es el bit de inicio, el cual, si está en cero, indica el inicio de la recepción. Los siguientes ocho bits son los de datos. El décimo bit es el de paridad, y finalmente el undécimo bit es el de parada, el mismo que debe ser uno lógico al concluir la transmisión del carácter.

Matriz de Datos

El propósito de este proceso es almacenar en una matriz 100 x 8, los valores de datos, que representan el valor binario de los caracteres ASCII ingresados por el usuario.

Serialización de Datos

Una vez almacenada la señal de datos en la matriz, este proceso permite tomar bit a bit todos los caracteres a ser codificados (desde el bit menos significativo del primer carácter hasta el bit más significativo del último carácter ingresado), obteniéndose la señal a codificar.

Codificación

La elección del Código de Línea que se va a implementar se establece de acuerdo al Identificador de Código, *ID_Codigo*, que forma parte del flujo de bits que se envía de MATLAB a la Spartan – 3E Starter Kit Board. Para la codificación de cada código de línea se la hace en componentes diferentes. La señal de reloj *RELOJ_OUT* se la utiliza para la codificación de los siguientes códigos: NRZ, Diferencial Tipo M, Diferencial Tipo S, AMI, HDB3 y MLT3. En cuanto a los códigos que requieren transición a mitad de tiempo de bit, la señal de reloj para codificación es *RELOJ_OUT_Doble*, y para el código de línea 4B5B se utiliza la señal *RELOJ_OUT_4B5B*.

Para la visualización de la codificación con cada uno de los códigos ya sean unipolares, polares o bipolares, se crean dos señales: *salida1* y *salida2*, las cuales toman valores que constituyen las entradas al circuito externo interpretador unipolar a bipolar, dando como resultado tres niveles de voltaje, - 5, 0 y 5 [V], tal como se muestra en la Tabla 2.

Tabla 2: Niveles de voltaje de acuerdo a las señales de salida

Niveles de Voltaje	salida1	salida2
0 [V]	0L	0L
- 5 [V]	0L	1L
+ 5 [V]	1L	0L
Restringido	1L	1L

A continuación se va presentar la componente de la codificación correspondiente al código Diferencial Tipo M.

Codificación Diferencial Tipo M

La componente *DIF_M_codificacion* permite realizar la codificación Diferencial tipo M. En la Figura 16 se muestra el diagrama de flujo de la programación en VHDL para esta componente.

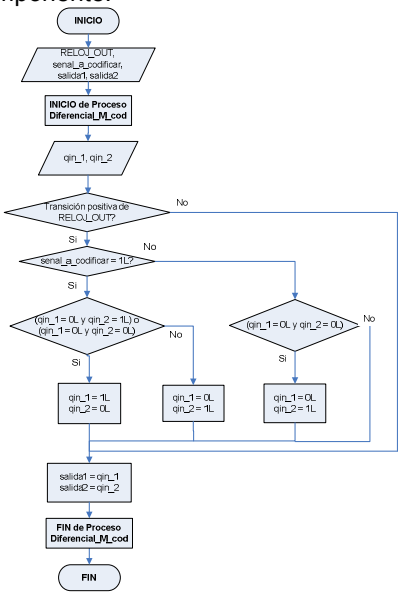

Figura 16: Diagrama de Flujo Codificación Diferencial Tipo M en VHDL.

La simulación de esta componente se la hace con TestBench de Xilinx ISE 10.1. La señal para codificación la constituyen los caracteres *ab* que en binario es 1000110010000110.

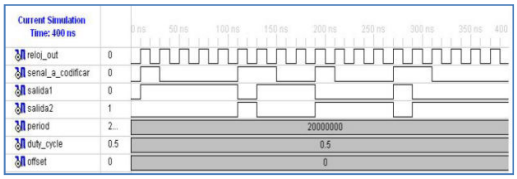

Figura 17: Simulación en Testbench de la codificación Diferencial Tipo M en VHDL.

Decodificación

Para continuar la construcción del diseño en forma jerárquica, la programación en VHDL para la decodificación con cada código de línea también se la realiza en componentes diferentes. Se identifica al código de línea, con el mismo *ID_Codigo* que en la codificación.

Se utilizan dos señales de entrada de datos que representan la señal codificada debido a que el FPGA no puede recibir niveles de voltaje negativos. Los valores que toman estas señales de entrada al FPGA son las salidas del circuito externo interpretador bipolar - unipolar, tal como se muestra en la Tabla 3.

Tabla 3: Niveles de voltaje de acuerdo a las señales entrada_cod1 y entrada_cod2.

Niveles de Voltaje	entrada_cod1	entrada_cod2
Restringido	0L	0L
- 5 [V]	0L	1L
0 [V]	1L	1L
+ 5 [V]	1L	0L

Al igual que en la codificación, se va a tomar como ejemplo el código Diferencial Tipo M para la decodificación.

Decodificación Diferencial Tipo M

La componente *DIF_M_decodificacion* tiene tres señales de entrada y una señal de salida. En la Figura 18 se observa el diagrama de flujo para esta componente.

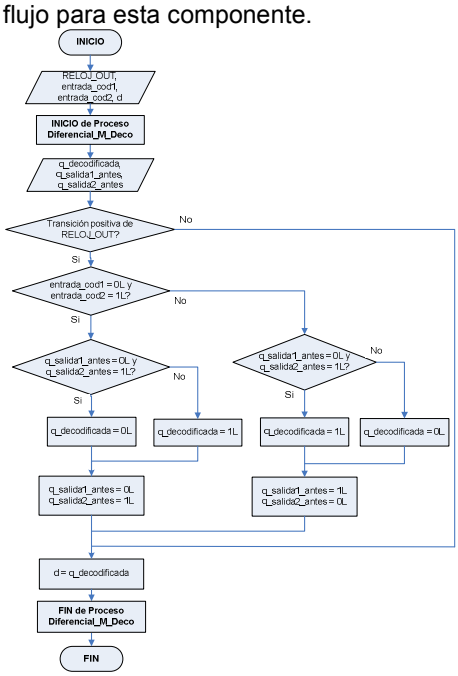

Figura 18: Diagrama de Flujo de la Decodificación Diferencial Tipo M en VHDL.

La simulación de la decodificación se la realiza en TestBench.

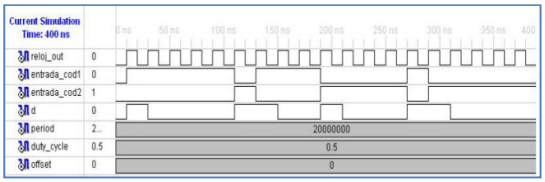

Figura 19: Simulación en Testbench de la decodificación Diferencial Tipo M en VHDL.

Identificación de bandera de inicio, ensamblaje de datos, matriz.

Luego de la decodificación, se procede a almacenar los bits de datos de la señal decodificada en la matriz de decodificación de tamaño 100 x 11. Este proceso incluye en cada fila de la matriz el bit de inicio, bit de paridad y bit de parada, a parte de los 8 bits decodificados.

Serialización y salida de datos.

Este proceso tiene como objetivo el retorno de los datos decodificados como un stream de once bits por carácter a la interfaz gráfica.

Asignación de pines

A las señales de la entidad del programa principal se les asigna varios pines de la Spartan 3E Starter Kit Board. Estos pines pueden ser para señales de entrada o salida, como: reset, pines del puerto RS-232, entradas y salidas de los circuitos interpretadores, señales de reloj, señal a codificar, señal decodificada, entre otras.

Diseño y construcción de los circuitos interpretadores

Debido a que la tarjeta de entrenamiento trabaja con niveles de voltaje LVTTL (TTL de bajo voltaje), únicamente se obtienen voltajes positivos; por consiguiente se requiere la construcción de un circuito externo interpretador unipolar – bipolar y un circuito externo interpretador bipolar – unipolar para los códigos de línea polares y bipolares.

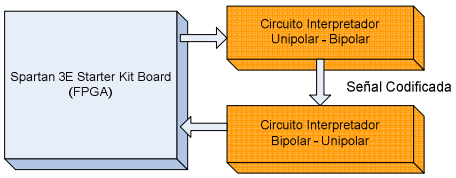

Figura 20: Diagrama de Bloques de la conexión entre el FPGA y los circuitos externos.

Circuito interpretador unipolar – bipolar

Este circuito externo tiene dos señales de entrada binarias y obtiene una señal bipolar de salida, de acuerdo a la Tabla 2, esto se puede efectuar con el diagrama de bloques de la Figura 21. El diagrama de bloques se lo

cumple basándose en amplificadores operacionales.

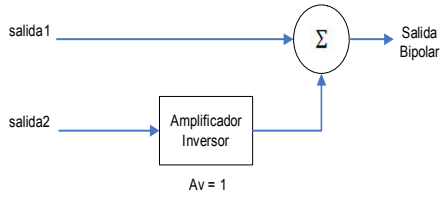

Figura 21: Diagrama de bloques del circuito Interpretador unipolar-bipolar.

En la Figura 21 se aprecia que se deben utilizar dos amplificadores operacionales, uno como amplificador inversor para obtener un nivel de voltaje negativo cuando la entrada al mismo sea un 1L, y el otro como sumador no inversor. El amplificador operacional seleccionado para la implementación de este circuito externo es el LF353, debido a su bajo costo y buena respuesta a altas velocidades (el tiempo de subida y tiempo de bajada está en el orden de las unidades de us), suficiente para que no exista distorsión e ISI (Interferencia inter símbolos) a la mayor velocidad, 19200 bps, en los códigos que tienen transiciones a mitad de bit.

Circuito interpretador bipolar – unipolar

Este circuito tiene como entrada una señal de tres niveles de voltaje -5 [V], 0[V] y +5[V] según sea el caso del código. El circuito interpretador bipolar - unipolar va a ser el encargado de traducir los tres niveles de voltaje en dos salidas binarias que van a ser entradas a la tarjeta de entrenamiento para la decodificación.

En la Figura 22 se observa el diagrama de bloques que representa la estructura del diseño de este circuito.

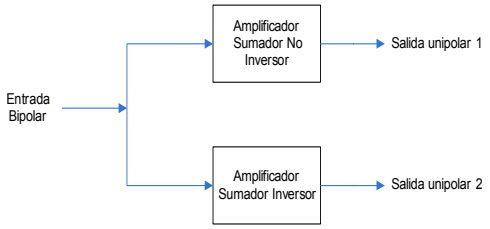

Figura 22: Diagrama de bloques del circuito interpretador bipolar-unipolar.

Caracteres restringidos

Se han utilizado varios identificadores como: de inicio y fin de datos, de velocidad, y de código de cada uno de los códigos de línea utilizados; los cuales corresponden a caracteres que no se utilizan usualmente. Los caracteres restringidos que no deben ser ingresados como datos, corresponden a los identificadores mencionados.

V. RESULTADOS

Con el objetivo de visualizar y almacenar los resultados producidos en la implementación se utiliza el hardware "DS1M12 Osciloscopio y Generador de Funciones USB", instrumento de EasySync.

Visualización de las señales de reloj

Las Figuras 23 y 24 presentan las tres señales de reloj generadas en base a *RELOJ_BASE* y *RELOJ_BASE_4B5B.* Estas señales se producen cuando se ha seleccionado la velocidad estándar de 9600 bps en la interfaz gráfica.

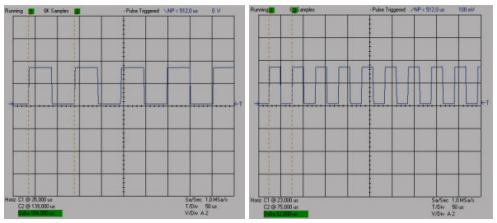

Figura 23: Señales de Reloj RELOJ_OUT y RELOJ_OUT_Doble.

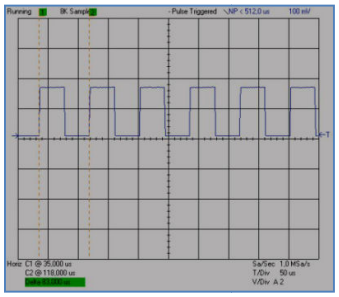

Figura 24: Señal de Reloj RELOJ_OUT_4B5B.

Resultados de funcionamiento

La codificación y decodificación con todos los códigos de línea expuestos se puede realizar a las velocidades 2400, 4800, 9600 y 19200 bps. Sin embargo las pruebas de funcionamiento que se presentan, es a 19200 bps por ser la condición más crítica.

Todas las pruebas de funcionamiento se realizan con el carácter ASCII "a" de valor binario 1000 0110, el bit menos significativo a la izquierda. El carácter de ejemplo está acompañado de dos caracteres adicionales que constituyen los identificadores de inicio y fin de datos.

La Figura 25 muestra la codificación de la señal de datos con el Código Diferencial tipo M en Matlab.

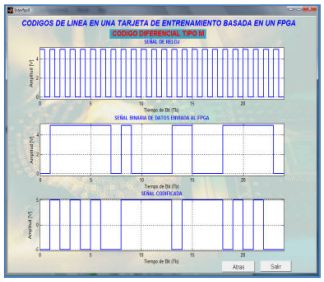

Figura 25:1 Codificación Diferencial tipo M en MATLAB.

En la Figura 26 se muestra la señal de datos a codificar y la señal codificada. Se aprecia que existe un retardo de un tiempo de bit en realizarse la codificación.

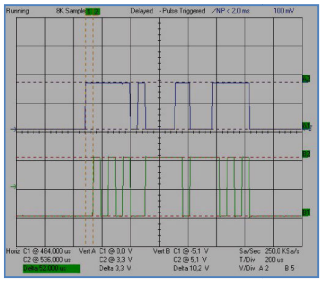

Figura 26: Visualización de la Codificación Diferencial tipo M.

La señal codificada y la señal decodificada se visualizan en la Figura 27. Existe un retardo de un tiempo de bit de la señal decodificada respecto a la señal codificada.

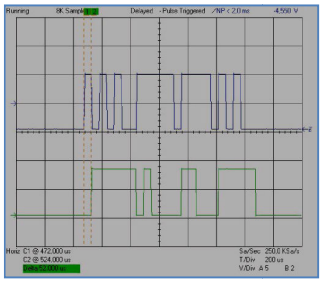

Figura 27: Visualización de la Decodificación Diferencial Tipo M.

Visualización de las señales enviadas desde el FPGA a la interfaz gráfica

En la Interfaz gráfica se grafica los datos recibidos, sin bits de inicio, paridad y parada tal como lo muestra la Figura 28.

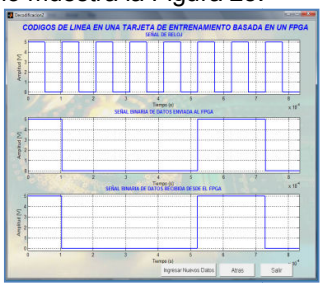

Figura 28: Gráfica de los datos recibidos en la Interfaz Gráfica.

Análisis de Tiempos de Procesamiento

En la Tabla 4 se presenta un resumen de los tiempos de procesamiento totales en la codificación y decodificación en VHDL desde que se obtiene la señal de datos, con cada uno de los códigos de línea que se analiza.

Tabla 4: Tiempos de procesamiento totales de los Códigos de Línea.

Código de Línea	Tiempo de Procesamiento a 19200 bps	Tiempo de Procesamiento a otras velocidades
NRZ	104 µs	2 tiempos de bit
RZ 50%	52 µs	1 tiempo de bit
4B5B	892 µs	Aprox. 17 tiempos de bit
Diferencial Tipo M	104 µs	2 tiempos de bit
Diferencial Tipo S	104 µs	2 tiempos de bit
Manchester	52 µs	1 tiempo de bit
Manchester Diferencial	52 µs	1 tiempo de bit
CMI	78 µs	1, 5 tiempos de bit
AMI	104 µs	2 tiempos de bit
HDB3	520 µs	10 tiempos de bit
MLT3	104 µs	2 tiempos de bit

VI. CONCLUSIONES

El DFS (Sintetizador de Frecuencia) del DCM, es una herramienta que cumple un papel importante en la elaboración de este proyecto, ya que permitió obtener la señal de reloj básica a partir de la cual se desarrollaron otras señales de reloj ocupadas en la transmisión, recepción, codificación, y decodificación de datos en la tarjeta de entrenamiento.

La programación en VHDL no debe contener señales y/o variables que pueden ser omitidas, las listas de sensibilidad deben contener todas las señales que al cambiar de estado permiten que se ejecute un proceso, las señales y variables de los tipos entero y natural deben declararse en rangos numéricos en los cuales se desenvuelven, y al utilizar segmentos secuenciales se debe saber las formas de utilización de condicionales.

Los scripts de MATLAB y los códigos fuente con el lenguaje de programación VHDL son totalmente diferentes. Los scripts de MATLAB son secuenciales, en su lugar VHDL es inherentemente concurrente.

En los resultados obtenidos se visualiza que para cada código de línea en particular, desde que se obtiene la señal a codificar en el FPGA hasta que se la codifica, al igual desde que ingresa la señal codificada hasta que se la decodifica; el tiempo de procesamiento se

mantiene en función de tiempos de bit, es decir no se tiene un tiempo de procesamiento fijo sino depende de la velocidad de transmisión escogida por el usuario.

VII. RECOMENDACIONES

Se recomienda realizar proyectos de sistemas digitales implementados en VHDL con FPGAs, incluso se podría elaborar un laboratorio de circuitos digitales con FPGAs.

Para definir el comportamiento del FPGA se utiliza VHDL, sin embargo se recomienda investigar otros métodos, entre ellos: Verilog, la herramienta System Generator de Simulink, los IP Cores, la herramienta EDK (Embedded Development Kit). En cuanto a hardware se sugiere investigar la familia de FPGAs Virtex 5 debido a que tiene la misma arquitectura pero atributos diferentes que la familia Spartan 3E; además se encuentran embebidos en tarjetas de desarrollo que poseen más periféricos.

Como una ampliación de este proyecto, se propone la implementación de los códigos de línea en base al esquema tradicional de comunicación de datos, con dos tarjetas de entrenamiento, en conjunto con dos computadores que constituyan un transmisor y receptor conectados por un canal de transmisión, y además hacer las pruebas a largas distancias.

REFERENCIAS

- MEYER-BAESE, Uwe; Digital Signal Processing with Field Programmable Gate Arrays. Springer-Verlag Berlin Heidelberg. Alemania. 2001
- PEDRONI, Volnei A.; Circuit Design with VHDL. 1a Edición. 2004
- STALLINGS, William; Data and Computer Communications. 5ª Edición.
- WOODS, Roger; McAllister, John; LIGHTBODY, Gaye; YI, Ying; FPGA-based Implementation of Signal Processing Systems. 1a Edición. 2008
- XILINX, ISE Design Suite 10.1 Release Notes and Installation Guide. 2008.
- XILINX, Spartan-3 Generation FPGA User Guide
- XILINX, Spartan-3E FPGA Family
- XILINX, Spartan-3E Starter Kit Board User Guide
- http://www.opencores.org/
- http://www.fpga4fun.com/
- http://www.isa.cie.uva.es/proyectos/codec/marco1.html

Viviana Checa R. Nació en San Gabriel en 1985. Realizó sus estudios secundarios en la Unidad Educativa Experimental "Manuela Cañizares" - Quito obteniendo el título de Bachiller en Ciencias especialización Físico Matemáticas, sus estudios superiores los realizó en la Escuela Politécnica Nacional - Quito, en la Facultad de Ingeniería Eléctrica y Electrónica con especialización en Ingeniería en Electrónica y Telecomunicaciones.
e-mail: vivi_chk@yahoo.com

Daniel Velásquez C. Nació en Ambato en 1986. Realizó sus estudios secundarios en el Instituto Tecnológico Superior Bolívar obteniendo el título de Bachiller en Ciencias y formando parte del cuadro de honor. Sus estudios superiores los realizó en la Escuela Politécnica Nacional - Quito, en la Facultad de Ingeniería Eléctrica y Electrónica con especialización en Ingeniería en Electrónica y Telecomunicaciones.
e-mail: jdanivelasquez@hotmail.com

Robin Álvarez Rueda. Nació en Cayambe, Ecuador, en 1969. Ingeniero en Telecomunicaciones, graduado de la Escuela Politécnica Nacional, Quito - Ecuador, 1996; M.Sc. en Telecomunicaciones por la Universidad de Cantabria, Santander - España, 2001; Ph.D en Telecomunicaciones por la Universidad Politécnica de Madrid-España, Enero-2006.
e-mail: robin.alvarez@epn.edu.ec