DISEÑO DE PRÁCTICAS PARA LABORATORIO DE COMUNICACIONES DIGITALES BASADO EN SIMULINK Y XILINX SYSTEM GENERATOR

HECTOR FERNANDO FRANCO MORENO

ABDUL YAVER QUINTERO

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS

ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

BUCARAMANGA

2012

DISEÑO DE PRÁCTICAS PARA LABORATORIO DE COMUNICACIONES DIGITALES BASADO EN SIMULINK Y XILINX SYSTEM GENERATOR

HECTOR FERNANDO FRANCO MORENO
ABDUL YAVER QUINTERO

Trabajo de Investigación para optar al título de
Ingeniero Electrónico

Director
MSc. JORGE HERNANDO RAMÓN SUÁREZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2012

AGRADECIMIENTOS

Agradezco a mis padres Antonio y María del Pilar por el apoyo incondicional y el ejemplo de vida basado en los valores que me han brindado. A mis hermanos David y Julián por ser cada uno desde su forma de ser un impulso para sobresalir.

A Dios por ese abrazo cercano y todas las bendiciones que me han rodeado siempre a través de la amistad y la música.

Hector Fernando Franco Moreno

AGRADECIMIENTOS

A Dios que está presente en cada una de mis acciones y me ha regalado tranquilidad y fortuna en la consecución de mis metas.

A toda mi familia por su total e incondicional apoyo, en especial a mi madre Fanny Esther, a mi padre Iván y a mi hermano Fadhul, por inculcarme los valores de responsabilidad y dedicación, además de brindarme todos sus conocimientos y experiencias.

Abdul Yaver Quintero

CONTENIDO

LISTA DE FIGURAS

pág.

LISTA DE ANEXOS

RESUMEN

TÍTULO: DISEÑO DE PRÁCTICAS PARA LABORATORIO DE COMUNICACIONES DIGITALES BASADO EN SIMULINK Y XILINX SYSTEM GENERATOR.[1]

AUTORES: Hector Fernando Franco Moreno y Abdul Yaver Quintero.[2]

PALABRAS CLAVE: Modulación digital, *System Generator*, *Direct digital synthesizer DDS*, *Simulink*, *Xilinx ISE*, *Spartan 3AN*.

DESCRIPCIÓN

En este trabajo se elaboró una propuesta teórico-práctica para el desarrollo del laboratorio de la asignatura comunicaciones digitales usando la herramienta de *Xilinx System Generator* en conjunto con *Simulink* de *Matlab*, donde se pueden diseñan circuitos digitales en el entorno grafico de *Simulink* por medio de bloques especializados de *Xilinx*, los cuales pueden ser exportados a código VHDL para configurar dispositivos lógicos programables.
Teniendo en cuenta el contenido de la asignatura se realizó el diseño, simulación e implementación de las modulaciones OOK, 4ASK, BFSK, 4FSK, 8PSK, 16PSK, 8QAM, 16QAM, PAM y PCM, dejando como resultado final un manual de usuario sobre el funcionamiento de los bloques más comunes de *System Generator* que son utilizados en aplicaciones de comunicaciones digitales, y otro manual de guías de laboratorio para estudiantes, las cuales dentro de su estructura cuentan con una actividad dirigida donde se describe en detalle el procedimiento y los parámetros necesarios para realizar el diseño, simulación e implementación en la tarjeta de entrenamiento *Spartan 3AN* las modulaciones OOK, BFSK, 8PSK, 8QAM, PAM sin retorno a cero y PCM delta. Adicionalmente se plantea como actividad adicional para fomentar el análisis y la creatividad, teniendo como base lo descrito anteriormente, el mismo desarrollo para las modulaciones 4ASK, 4FSK, 16PSK, 16QAM, y PAM con retorno a cero.

---

[1] Proyecto de Grado

[2] Facultad de Ingenierías Físico Mecánicas, Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones, Director: MSc. Jorge Hernando Ramón Sánchez.

ABSTRACT

TITLE: DESIGN OF PRACTICES FOR DIGITAL COMMUNICATIONS LABORATORY BASED ON SIMULINK AND XILINX SYSTEM GENERATOR.[1]

AUTHORS: Hector Fernando Franco Moreno y Abdul Yaver Quintero.[2]

KEY WORDS: Digital modulation, System Generator, *Direct digital synthesizer DDS* Simulink, Xilinx ISE, *Spartan 3AN*.

DESCRIPTION

In this project, a theoretical-practical proposal for the laboratory development of the subject *Digital Communications* was elaborated using the *Xilinx System Generator* tool together with *Matlab Simulink*, where digital circuits can be designed in the Simulink graphical environment using Xilinx specialized blocks, which can be exported to VHDL code to configure programmable logic devices. Taking into account the content of the subject it was done the design, the simulation and the implementation of modulations OOK, 4ASK, BFSK, 4FSK, 8PSK, 16PSK, 8QAM, 16QAM, PAM y PCM, giving as final result a user's manual on the operation of the most common blocks of *System Generator* that are used in digital communication applications and another manual laboratory guide for students, which within its structure have directed activity that describes in detail the procedure and the necessary parameters required for the design, simulation and implementation on Spartan 3AN training card modulations OOK, BFSK , 8PSK, 8QAM, PAM no return to zero and PCM delta. Besides it poses as additional activity to promote the analysis and creativity, taking as a basis what was described above, the same development for 4ASK, 4FSK, 16QAM, 16PSK and PAM modulations with return to zero.

---

[1] Degree project

[2] Faculty of Physics Mechanics Engineering, School of Electrical Engineering, Electronic Engineering and Telecommunications. Director: MSc. Jorge Hernando Ramón Sánchez.

I. Introducción

Xilinx System Generator suministra un conjunto de bloques especiales para transformar los datos dentro de un entorno de simulación o software, que en éste caso es *Simulink* de *MATLAB*, para aplicaciones de hardware que pueden implementarse en diversos *FPGAs* de *Xilinx*, lo que hace a dicha herramienta una de las principales a nivel industrial para el diseño de sistemas de procesamiento digital de señales (DSP) con alto rendimiento.[1]

El propósito del trabajo de investigación es que los estudiantes reconozcan y se familiaricen con la herramienta *Xilinx System Generator* ubicándolos dentro del contexto de las comunicaciones digitales, esto a través de unas guías de laboratorio que tendrán como objetivo comprender ampliamente el funcionamiento de algunas modulaciones digitales y estudiar sus comportamientos reales gracias a la implementación de los sistemas a través del *Kit Spartan 3AN de Xilinx***.**

II. Antecedentes

El Laboratorio de Comunicaciones de la Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones de la Universidad Industrial de Santander cuenta con el modulo especializado 91022-22 de *LabVolt* para comunicaciones digitales, el cual contiene bloques de muestreo, retenedores, sumadores, limitadores, comparadores, generadores de función rampa, filtros, entre otros, y un manual guía de estudio que le permite a los estudiantes configurar, operar y analizar circuitos para transmisión digital de modulaciones PAM, PWM, PPM y PCM.

En el 2001 se desarrolló un proyecto de grado para la Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones, con el título: LABORATORIO VIRTUAL DE COMUNICACIONES USANDO LABVIEW realizado por los estudiantes: Dora María Ballesteros Larrota, Néstor Yesid Mojica Rodríguez y Erick José Vera Mercado. Este proyecto consistió en implementar un software elaborado en lenguaje de programación grafico Labview, el cual incluía la simulación de trece tipos de modulaciones diferentes, guías de laboratorio para cada una de ellas y a su vez un manual de usuario para un manejo general de la herramienta.[2]

En el año 2012 los estudiantes Vivian Paola Triana Galeano y Javier Mauricio Suárez Monsalve desarrollaron el proyecto de grado para la Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones titulado: DISEÑO DEL MANUAL DE PRÁCTICAS PARA UN LABORATORIO DE COMUNICACIONES DIGITALES BASADO EN LA TÉCNICA DE RADIO DEFINIDO POR SOFTWARE. En este trabajo se desarrolló una propuesta para el laboratorio de comunicaciones digitales, en la cual se estudiaron algunas modulaciones como GMSK y PSK y sus variaciones, elaborando para ellas una guía de uso práctico basada en el concepto de radio definido por *software*, implementado de manera inalámbrica utilizando la herramienta de desarrollo GNU Radio.[3]

III. System Generator

System Generator es una herramienta *software* de diseño DSP (*Digital signal processing*) de *Xilinx*, la cual permite combinar la facilidad de diseño y

simulación del entorno grafico de *Simulink* de *MathWorks* y la eficiencia para descripción de hardware de *ISE Design Suite Logic Edition* en FPGA. Permite diseñar sobre un entorno de alto nivel, robusto y fácil de usar, sistemas DSP de alto rendimiento para un determinado *hardware*, mediante funciones lógicas, de memoria y específicas, ideales para el filtrado digital, análisis espectral y comunicaciones digitales.

En esta Herramienta el diseño es creado en el entorno grafico de *Simulink/Matlab*, usando librerías de *Xilinx*, teniendo como ventaja el acceso a la herramienta de simulación de *Simulink* y de esta forma verificar y modificar el funcionamiento del sistema. Posteriormente convierte el modelo de bloques de Xilinx en *Simulink* en una muy eficiente implementación de *hardware* que combina el código VHDL personal con los bloques de propiedad intelectual de *Xilinx* que ya están listos para ejecutarse con alto rendimiento en la *FPGA*.

Esta herramienta requiere para su funcionamiento la sincronización del *software Matlab* de *MathWorks* e ISE de *Xilinx*, esta asociación agrega un complemento de bloques dedicados (*blockset*) de *Xilinx* a la librería de *Simulink*.

Xilinx Blockset: familia de librerías que contiene bloques básicos de *System Generator* para el acceso al *hardware* especifico del dispositivo y otros para procesamiento de señales y algoritmos avanzados de comunicación (registros, operaciones aritméticas, comparadores).

Xilinx Reference Blockset: Contiene bloques compuestos de *System Generator* que implementan un amplio rango de funciones (máquinas de estado, filtrado).

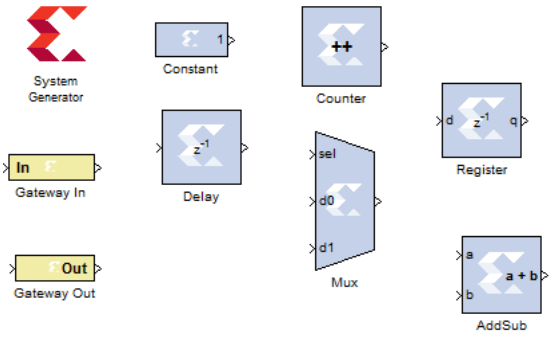

Figura 1. Xilinx Blockset.

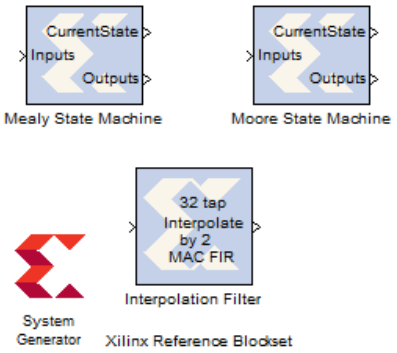

Figura 2. Xilinx Reference Blockset.

IV. Propuesta

La idea central de la investigación es el diseño de unas guías de laboratorio que consisten en la simulación e implementación de algunas modulaciones y transmisiones digitales para que sean parte del laboratorio de la asignatura Comunicaciones Digitales, que en los últimos semestres no ha tenido un espacio reservado para las experiencias en laboratorio y solo se ha trabajado en el aspecto de simulación con un acercamiento a *Simulink*, es por eso que éste es el primer aspecto que motivó la investigación, pues existe una experiencia en *Simulink*, dónde igualmente se diseñan los sistemas que serán implementados a través de la herramienta *System Generator*, lo que hace de éste proyecto de investigación un buen complemento

para lo que se viene trabajando dentro de la asignatura. El segundo aspecto para tener en cuenta en la investigación han sido los recursos con los que cuenta la universidad, entre éstos se encuentran algunos *kits Spartan 3AN* de *Xilinx*, licencia de *Matlab* con *Simulink*, y la facilidad de adquirir la licencia de *Xilinx ISE* con el paquete *System Genarator*, que es gratuita para instituciones académicas, así pues están prestas las condiciones para el desarrollo de laboratorios que involucren no solo simulaciones, sino también la ventaja de implementar y ver un comportamiento real de los sistemas de comunicaciones digitales, gracias a la versatilidad de la herramienta.

Las guías de laboratorio, fundamentadas en las principales modulaciones y transmisiones digitales, poseen una estructura basada en simulación y posterior implementación, a través una actividad dirigida y una actividad adicional; en la primera el profesor encargado de la dirección del laboratorio explicará a los estudiantes un sistema que describe algún tipo de modulación o transmisión digital, su diseño y correcto procesamiento de las señales digitales en el entorno de *Simulink* y posteriormente la respectiva implementación a través de *Xilinx ISE* en el *Kit Sapartan 3AN*; la segunda es una actividad basada en el trabajo dirigido con algunas variaciones que desarrollarán los estudiantes, para consecutivamente entregar un informe detallado con los procedimientos y análisis que dejó la experiencia en el laboratorio.

En el anexo A de éste documento se encuentra la definición de cada uno de los bloques funcionales utilizados en los sistemas diseñados para las guías de laboratorio.

El anexo B de éste artículo contempla todo el procedimiento de instalación de la herramienta *Xilinx System Generator*.

La siguiente es la lista de las guías contempladas en este trabajo de investigación y que corresponden al programa de Comunicaciones Digitales, siguiendo un orden cronológico, es decir, la consecución de los temas vistos teóricamente en la asignatura. También se muestran las actividades planteadas para cada práctica, éstas contemplan un procedimiento de simulación y posterior implementación.

1) Generación de Ondas Seno y Coseno (Práctica introductoria).
   Actividad dirigida: Generación de una onda seno y una onda coseno.
   Actividad adicional: Generación de dos ondas sinusoidales.

2) Modulación Digital ASK.
   Actividad dirigida: Modulación OOK.
   Actividad adicional: Modulación 4ASK.

3) Modulación Digital FSK
   Actividad dirigida: Modulación BFSK.
   Actividad adicional: Modulación 4FSK.

4) Modulación Digital PSK
   Actividad dirigida: Modulación 8PSK.
   Actividad adicional: Modulación 16PSK.

5) Modulación Digital QAM
   Actividad dirigida: Modulación 8QAM.
   Actividad adicional: Modulación 16QAM.

6) Modulación Digital PAM
Actividad dirigida: Modulación PAM sin retorno a cero.
Actividad adicional: Modulación PAM con retorno a cero.

7) Modulación Digital PCM
Actividad dirigida: Modulación PCM Delta con una señal referencia sinusoidal pura.
Actividad adicional: Modulación PCM Delta con una señal referencia distinta a una sinusoidal pura.

En el anexo C de esté documento se encuentra en detalle cada una de las guías de laboratorio, junto a todos los procedimientos realizados y actividades propuestas y desarrolladas utilizando *Xilinx System Generator* e implementando en un *FPGA* del *kit Spartan 3AN*.

V. Resultados

Las simulaciones para las diferentes modulaciones desarrolladas mediante la herramienta *System Generator* en el entorno de *Simulink*, permiten visualizar y verificar las principales características de cada una de ellas, tal y como se viene desarrollando en la asignatura Comunicaciones Digitales hasta el momento. En la figura 3 se muestra el resultado obtenido para la simulación de la modulación por amplitud de pulsos PAM.

Debido a la robustez de la herramienta *System Generator* los resultados de la simulación son consecuentes con los que se obtienen en la implementación, lo cual permite comprobar el funcionamiento de los moduladores en tiempo real. La implementación de la modulación PAM se muestra en la figura 3.

El desempeño de la tarjeta Spartan 3AN implementando las modulaciones descritas anteriormente fue satisfactorio. Midiendo los parámetros de desfase (figura 4) y amplitud en las modulaciones, se obtuvieron errores máximos de 0.99° y 0.1V los cuales representan un error relativo respecto al valor teórico de 8.8% y 9.8% respectivamente.

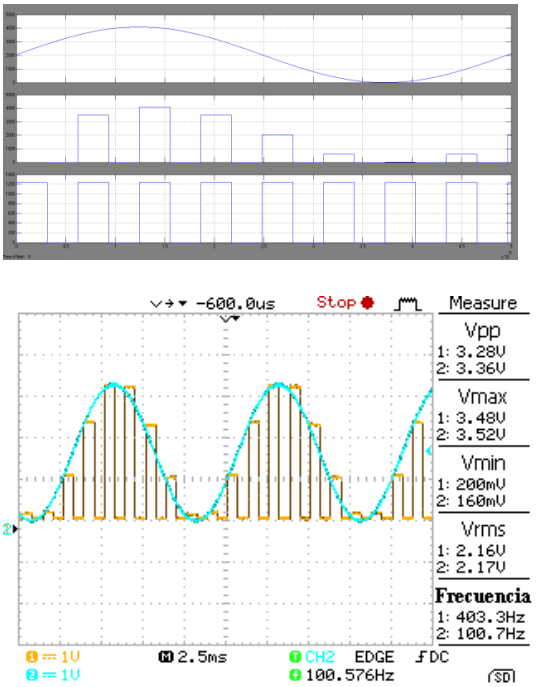

Figura 3. Simulación e implementación modulación PAM

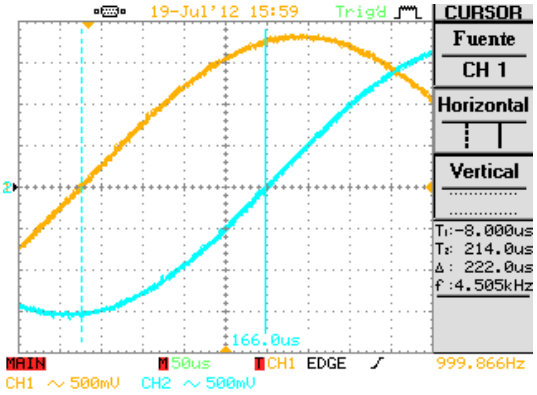

Figura 4. Medición del desfase en la implementación 16 PSK para el dato binario de entrada "0011".

Se realizó una prueba con estudiantes activos de la universidad, donde se desarrollaron todos los procedimientos contemplados en dos de las guías de laboratorio, la primera la correspondiente a la generación de ondas seno y coseno, donde se presentan todos los pasos muy detallados para todo el procesamiento de las señales digitales para efectos de simulación e implementación; la segunda la correspondiente a la modulación por desplazamiento de amplitud ASK, que al igual que las demás prácticas contiene los procedimientos más condensados, pues los detalles se explicaron en guías anteriores. A través del ejercicio de retroalimentación con los estudiantes se realizaron ajustes y adiciones sobre algunos detalles en la explicación de procesos específicos, y de esta manera se validaron satisfactoriamente las prácticas para el laboratorio.

VI. Conclusiones

La propuesta de un laboratorio para la asignatura comunicaciones digitales basada en la herramienta *Xilinx System Generator* es una alternativa viable a desarrollar en la Universidad Industrial de Santander teniendo en cuenta los recursos con que ella cuenta, y que contribuye a enriquecer la experiencia de estudio teórico-práctica de los estudiantes de pregrado de Ingeniería Electrónica porque además del componente de simulación incluye la verificación experimental de las señales en tiempo real.

El Kit de desarrollo de *Xilinx Spartan 3AN* cuenta con los recursos lógicos suficientes para implementar las modulaciones OOK, 4ASK, BFSK, 4FSK, 8PSK, 16 PSK, 8QAM, 16 QAM, PAM y PCM Delta.

En la generación digital de las señales en los Bloques *DDS Compiler* debido a la relación entre sus variables internas para determinar la frecuencia, el incremento del ancho de fase debería tomar valores no enteros, lo cual no es permitido en el bloque, por tal motivo deben ser redondeados al número entero más cercano, lo que provoca que la frecuencia no sea tampoco un número entero y en consecuencia en las modulaciones por amplitud de pulso PAM y por codificación de pulso PCM Delta, el muestreo no se hace en los mismos puntos cada periodo de las señales referencia, generando en el momento de la visualización en tiempo real un efecto de movimiento.

Trabajo futuro

Para ser consecuentes con los objetivos académicos del contenido de la asignatura Comunicaciones Digitales, se propone realizar los sistemas de demodulación a través de *System Generator* sobre el *kit Spartan 3AN* para observar experimentalmente el funcionamiento de los receptores que interpretan las modulaciones (realizadas en este proyecto de investigación), a través de un medio de transmisión cableado.

Adicionalmente para complementar el laboratorio de comunicaciones digitales estudiando los efectos del canal y los factores externos que intervienen en él, se recomienda implementar un sistema de transmisión inalámbrico. Debido a que las tarjetas de desarrollo de *Xilinx* no cuentan con un módulo transmisor inalámbrico integrado directamente, se propone la utilización de módulos externos como el *AD-FMCOMMS1-EBZ* de *Analog Devices*, o el *SDRtx 3U VPX RF transmitter Board* de *SOC-e* los cuales

son compatibles con tarjetas de *Xilinx* como la *Virtex-6 ML605*, la cual a su vez es configurable a través de la herramienta *System Generator*

VII. Agradecimientos

Presentamos nuestra gratitud al Grupo de Investigación en Conectividad y Procesado de Señal (CPS), en especial a nuestro director de proyecto el profesor Jorge Hernando Ramón Suárez. De la misma manera agradecemos al profesor William Salamanca y a la ingeniera Yuri Mejía por su valiosa asesoría, y también al estudiante Álvaro Javier Flórez por su total colaboración en el proceso de validación para las prácticas de laboratorio.

VIII. Referencias

[1] AN INTRODUCTION TO XILINX SYSTEM GENERATOR. Miroslav Knezevic. Reviewed by: Pieter Nuyts, Tom Redant and Nele Reynders. email: miroslav.knezevic@esat.kuleuven.be.

[2] LABORATORIO VIRTUAL DE COMUNICACIONES USANDO LABVIEW. Dora Maria Ballesteros Larrotta, Nestor Yezid Mojica Rodriguez, Erick Jose Vera Mercado; director Jaime Barrero Perez, codirectores Faver Amorocho, Jesus David Acero. Bucaramanga 2001. Universidad Industrial de Santander. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones.

[3] DISEÑO DEL MANUAL DE PRÁCTICAS PARA UN LABORATORIO DE COMUNICACIONES DIGITALES BASADO EN LA TÉCNICA DE RADIO DEFINIDO POR SOFTWARE. Vivian Paola Triana Galeano, Javier Mauricio Suárez Monsalve; director Jorge Hernando Ramón Suárez, codirector Diego Rafael Medina Pulido. Bucaramanga 2012. Universidad Industrial de Santander. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones.

IX. Biografías

Hector Fernando Franco Moreno. Recibió el título de Bachiller académico en la Fundación Colegio UIS, Floridablanca, Santander, Colombia, en el año 2006. Ingeniero Electrónico de la Universidad Industrial de Santander, Colombia. Forma parte del grupo de Investigación en Conectividad y Procesado de Señal (CPS) de la UIS.

Abdul Yaver Quintero. Recibió el título de Bachiller académico en el Instituto Nacional José María Campo Serrano, Aguachica, Cesar, Colombia, en el año 2006. Ingeniero Electrónico de la Universidad Industrial de Santander, Colombia. Forma parte del grupo de Investigación en Conectividad y Procesado de Señal (CPS) de la UIS.

X. ANEXOS

ANEXO A

ANEXO A

Manual Referencia de los Bloques System Generator

1. CONVERTIDOR DIGITAL-ANALOGO (DAC) SPARTAN 3AN

La tarjeta de entrenamiento Starter Kit 3AN incluye un convertidor digital a analógico (DAC) serial con protocolo de comunicación SPI. El dispositivo principal del DAC es el LTC2624, el cual consta de cuatro canales de salida A, B, C y D los cuales se encuentran disponibles en el conector J21 (figura 1).

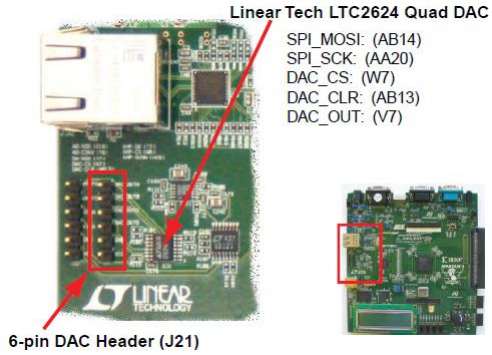

Figura 1. Conector J21. Tomado de: Spartan-3A/3AN FPGA Starter Kit Board User Guide UG334

El FPGA utiliza el protocolo SPI (*Serial Peripheral Interface*) para comunicar valores digitales a cada uno de los canales del DAC. La interfaz de comunicación (figura 2) consta de 4 señales básicas: la primera es el reloj, la cual da el sincronismo para todo el sistema (SPI_SCK); la siguiente es de habilitación (DAC_CS) la cual indica el inicio y fin de la comunicación; debe permanecer normalmente activa en '1' lógico y da inicio a la comunicación al desactivarse '0', la conversión digital a análogo ocurre cuando esta vuelve a activarse en '1'. Los datos digitales se transmiten de forma serial a través de la señal (SPI_MOSI); y finalmente una señal de *reset* asíncrono (DAC_CLR) la cual se activa en bajo '0'.

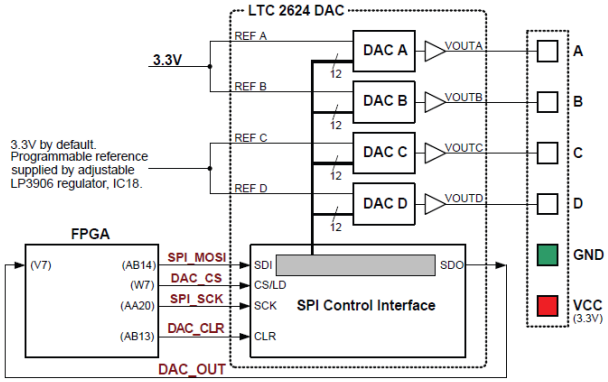

Figura 2. Interfaz de comunicación DAC. Tomado de: Spartan-3A/3AN FPGA Starter Kit Board User Guide UG334

Cada nivel de salida del DAC es el equivalente análogo a un dato de 12 bits D[11:0] sin signo (valores positivos) descrito por la ecuación 1. El voltaje de referencia VREF es diferente entre los cuatro canales del DAC, los canales A y B tienen un voltaje de referencia fijo de 3.3V, los canales C y D tienen una alimentación independiente nominalmente también de 3.3V, pero esta puede ser modificada a otros niveles.

La tensión de referencia tiene una tolerancia de 5% por lo cual hay ligeras variaciones en los niveles de salida.

$$Vout = \frac{D[11:0]}{4096} * VRef \qquad (1)$$

2. SYSTEM GENERATOR

System Generator es una herramienta software de diseño DSP (*Digital signal processing*) de *Xilinx*, la cual permite combinar la facilidad de diseño y simulación del entorno grafico de *Simulink* de *MathWorks* y la eficiencia para descripción de hardware de *ISE Design Suite Logic Edition* en FPGA. Permite diseñar sobre un entorno de alto nivel, sistemas DSP de alto rendimiento para un determinado hardware, mediante funciones lógicas, de memoria y específicas, ideales para el filtrado digital, análisis espectral y comunicaciones digitales.

Esta herramienta requiere para su funcionamiento la sincronización del software *Matlab* de *MathWorks* e ISE de *Xilinx*, la cual agrega un complemento de bloques dedicados de *Xilinx* a la librería de *Simulink*.

En esta Herramienta el diseño es creado en el entorno grafico de *Simulink*, utilizando librerías de *Xilinx*, teniendo como ventaja el acceso a la herramienta de simulación de S*imulink* y de esta forma verificar y modificar el funcionamiento del sistema. Posteriormente convierte el modelo de bloques realizado en una eficiente implementación de hardware que combina el código VHDL personal con la propiedad intelectual de los bloques *Xilinx* que ya están listos para ejecutarse con alto rendimiento en la FPGA.

2.1 OPCIONES COMUNES EN PARÁMETROS DE BLOQUES

Cada uno de los bloques de *SysGen* presenta parámetros configurables propios, a los cuales se puede acceder haciendo doble clic sobre el mismo. Sin embargo hay algunos parámetros básicos que son comunes en la mayoría de los bloques, los cuales se detallan a continuación.

2.1.1 REPRESENTACION ARITMETICA

System Generator maneja los tres tipos de dato aritmético más comunes en aplicaciones DSP: *Double* de punto flotante (formato de *Simulink*), números de punto fijo con y sin signo. El formato de punto flotante no puede ser traducido a hardware, pero si utilizado en simulación.

En los bloques este parámetro permite escoger el tipo de dato que se obtendrá en la salida del mismo, las opciones que ofrece *SysGen* son: Booleano, sin signo y con signo en complemento a dos.

2.1.2 NUMERO DE BITS

Cuando se escoge que la precisión sea definida por el usuario se tiene acceso a este parámetro, el cual permite indicar con cuantos bits se va a representar cada muestra a la salida del bloque. El máximo valor que puede adoptar es 4096 bits.

2.1.3 PUNTO BINARIO

El parámetro punto binario indica el número de bits a la derecha del punto decimal, estableciendo el tamaño de la parte fraccionaria. La posición del punto binario debe estar entre cero y el valor especificado en el número de bits.

Así un número de punto fijo se caracteriza por los tres parámetros analizados anteriormente, la representación aritmética, el número de bits y el punto binario.

2.1.4 PRECISIÓN

El modo fundamental de los bloques de *Xilinx* es precisión aritmética de punto fijo. La mayoría de los bloques de *Xilinx* permiten escoger la precisión de salida, mediante la opción *User defined*, se puede especificar el número de bits y posición de punto binario a la salida del mismo. La opción por defecto es *Full Precision*, la cual quiere decir que a la salida se obtendrá una precisión lo suficientemente grande para presentar el resultado sin errores por redondeo o desbordamiento.

2.1.5 CUANTIFICACIÓN Y DESBORDAMIENTO

Las opciones más comunes para la cuantificación son:

- Truncate: Elimina o descarta los bits que se encuentran a la derecha del bit menos significativo (LSB).
- Round: Permite redondear al valor más cercano con el número de bits disponibles.

Para el desbordamiento *SysGen* ofrece tres opciones:

- Wrap: Descarta los bits que están a la izquierda del bit más significativo (MSB).
- Saturate: Permite saturar al valor positivo más grande o al negativo más pequeño posible según el número de bits.
- Flag: Genera un error durante la simulación indicando desbordamiento.

2.1.6 PERIODO DE MUESTREO

El flujo de datos es procesado a una tasa específica de muestreo gobernada por el reloj principal del sistema, siendo común que cada bloque detecte la tasa de muestreo de entrada y genere la correcta tasa de muestreo a la salida. Sin embargo *SysGen* permite modificar este parámetro con la opción *Specify Explicit Period*, con la cual el usuario ingresa el periodo de muestreo deseado a las salidas del bloque.

2.1.7 LATENCIA

Este parámetro presente en la mayoría de los bloques de *Xilinx*, define el número de periodos de muestreo que tardara la señal en aparecer a la salida del bloque.

2.2 BLOQUES UTILIZADOS EN EL PROYECTO

2.2.1 TOKEN DE SYSTEM GENERATOR

Figura 3. *System Generator Token*

Este bloque debe estar presente en todos los diseños, posee características esenciales para la generación y simulación de los modelos en *Simulink* que contengan elementos del *Blockset* de *Xilinx*. Ejerce el control del sistema pues en sus parámetros están el periodo del reloj para implementaciones en hardware y simulación, además de la invocación del generador de código.

Las diferentes librerías del *Blockset* de *Xilinx* donde se encuentra este bloque son *Basic elements*, e *Index*.

Sus principales parámetros configurables se describen a continuación:

- pestaña *Compilation* (Figura 4):

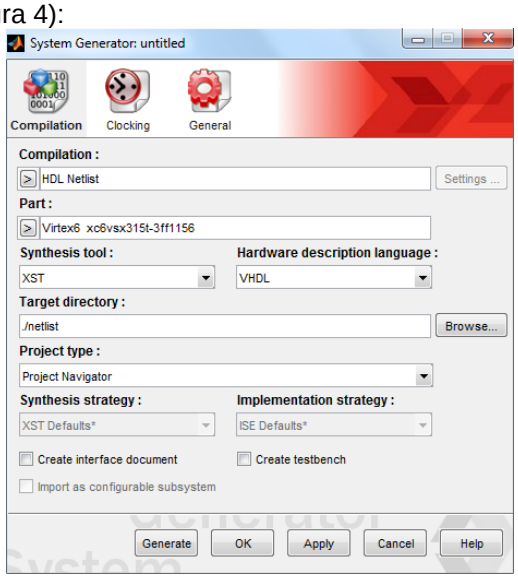

Figura 4. Pestaña *Compilation* del *System Generator Token.*

Compilation: Indica el tipo de compilación a realizar cuando se ejecute el generador de código, en el podemos escoger *Netlist* para crear una fuente en el lenguaje de descripción seleccionado, o *Bitstream* para generar el archivo de programación directamente.

Part: Define el hardware a utilizar, el tipo de FPGA que contiene la tarjeta de entrenamiento.

Target Directory: Permite seleccionar el directorio donde *SysGen* escribirá los resultados de la compilación.

Hardware Description Language: Señala el tipo de lenguaje HDL que se empleara para la compilación (VHDL o Verilog).

Synthsis Tool: especifica la herramienta que se utilizara para la síntesis del diseño. Teniendo tres posibilidades: *Synplity's Synplify Pro*, *Synplify y Xilinx's XST* (Incluida con la instalación de *SysGen*).

Create Testbench: Indica a *SysGen* que cree un archivo de prueba (*HDL Test bench*) con el cual se puede simular en ISE y comparar el resultado de *Simulink*.

- pestaña *Clocking* (Figura 5):

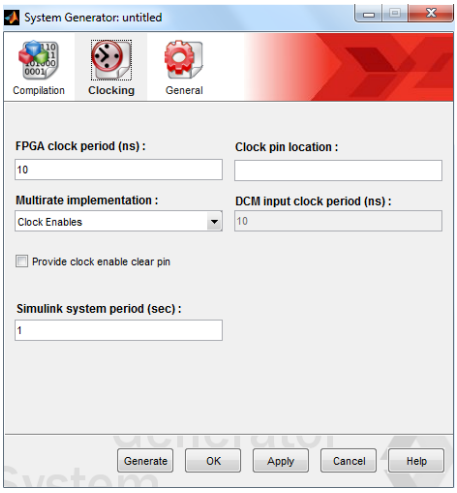

Figura 5. Pestaña *Clocking* del *System Generator Token.*

FPGA Clock Period: Determina el periodo que tomara el reloj en el hardware dentro del FPGA (en nanosegundos).

Clock Pin Location: En él se puede especificar el terminal UCF que corresponde a la señal de reloj a utilizar, siendo esta diferente en cada tarjeta de entrenamiento (Spartan 3AN: E12)

Provide Clock Enable Clear Pin: Proporciona un puerto de habilitación ce_clr en el reloj principal.

Simulink System Period: Define el periodo del sistema en *Simulink* (en segundos). Este es el máximo común divisor de los periodos de muestreo que aparecen en el modelo.

- Pestaña *General (Figura 6):*

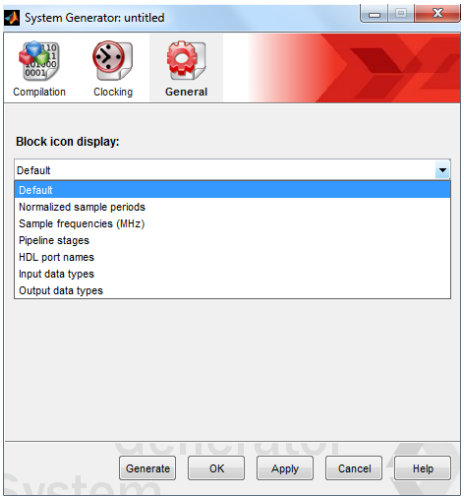

Figura 6. Pestaña *General* del System Generator Token.

Block Icon Display: Especifica la información que se desea visualizar en los bloques cuando se simula el sistema. Existen varias opciones para visualización, una de ellas es *output data types* la cual muestra la representación aritmética del dato al salir de cada bloque, por ejemplo Ufix_8_2, ósea de punto fijo (fix) sin signo (unsigned) de 8 bits y punto binario en el segundo bit. Otras opciones son *Default*, *HDL port names*, *Input data types*.

2.2.2 GATEWAY IN

Figura 7. Bloque Gateway In.

Este bloque (Figura 7) es la interfaz entre los bloques de S*imulink* y los de *SysGen*, permitiendo hacer interconexiones entre ellos debido a que convierte los datos de tipo entero y doble (*Simulink*) al formato de punto fijo (*SysGen)*. A su vez cada uno de estos bloques define una entrada de alto nivel (*Top Level*) en el diseño HDL generado.

Las diferentes librerías del *Blockset* donde encuentra este bloque son: *Basic Elements, Data types e index.*

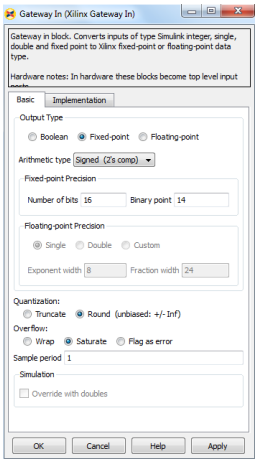

Figura 8. Bloque Gateway In / Basic

En la pestaña *Basic* (Figura 8) se encuentran las opciones comunes descritas en la sección 2.1 mediante las cuales se puede configurar los puertos de entrada y la naturaleza de la información que ingresara al diseño, tales parámetros son el tipo de dato, la representación aritmética, la cantidad de bits, la ubicación del punto binario, la forma de proceder respecto a cuantificación o desbordamiento de datos y el periodo de muestreo.

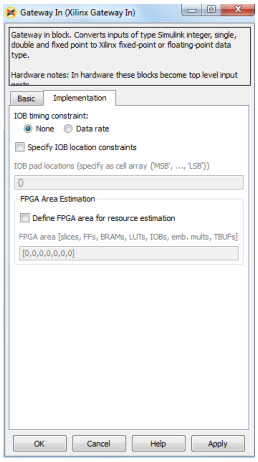

Figura 9. Bloque Gateway In / Implementation

Specify IOB Location Constraints: Si se marca este cuadro, habilita un campo que permite al usuario definir la ubicación de la interfaz que representa este bloque dentro de la tarjeta de entrenamiento.

IOB Pad Locations: Aquí se describe la ubicación de la interfaz como un arreglo de caracteres tipo celda {'MSB',…,'LSB'}.

2.2.3 GATEWAY OUT

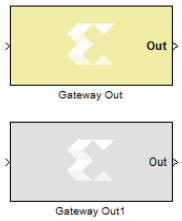

Figura 10. Bloque Gateway Out.

Este bloque es la interfaz entre los bloques de *SysGen* y los de *Simulink*, debido a que convierte los datos de tipo punto fijo de *SysGen*, en tipo doble de *Simulink*.

De acuerdo a su configuración este bloque puede también definir un puerto de salida de alto nivel en el diseño HDL generado por *SysGen* o también ser usado simplemente como un punto de prueba.

Las diferentes librerías del *Blockset* de *Xilinx* donde se encuentran este bloque son *Basic Elements, Data Types, e Index*.

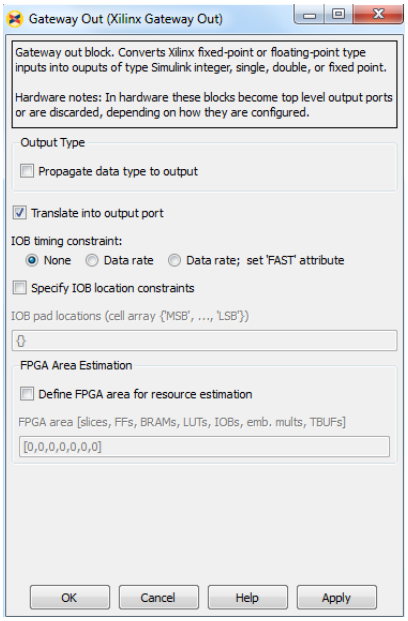

Figura 11. Parámetros Bloque Gateway Out.

Translate into Output Port: En forma predeterminada está habilitada, haciendo que el bloque sea traducido como un puerto de salida de alto nivel en el diseño, pero si se deshabilita esta opción, se previene que llegue a trasladarse a un puerto en hardware. Cuando esto último pase, este bloque se tornara de color gris.

IOB Timing Constraint: Cuando se traslada al *hardware* el bloque *Gateway Out* se construye con un conjunto de *buffers* de entrada/salida IOB. Los tipos de sincronización son: *None, Data Rate* y *Data Rate; Set 'FAST' Attribute*.

Si *None* es seleccionado los *IOBs* hacia los elementos síncronos no se limitan; si se selecciona *Data Rate* los *IOBs* son limitados y funcionaran con un tiempo de muestreo definido que depende de la frecuencia del reloj especificado en el *Token* del sistema. Por último en *Data Rate; Set 'FAST' Attribute* se reduce el retardo producido al seleccionar la opción anterior, pero incrementa el ruido y el consumo de potencia.

Los otros dos parámetros **Specify IOB Location Constraints** e **IOB Pad Locations** se configuran de igual forma que en el bloque Gateway In.

2.2.4 DDS Compiler V4.0

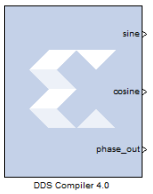

Figura 12. Bloque DDS Compiler v4.0.

Las diferentes librerías del *Blockset* de *Xilinx* donde se encuentra este bloque son: DSP e *Index*.

Este bloque implementa un DDS (*Direct Digital Synthesizer*) también conocido como oscilador controlado numéricamente NCO, el cual es fuente de ondas sinusoidales para diversas aplicaciones.

En términos generales un DDS tiene dos componentes principales, el acumulador de fase y una memoria.

El acumulador de fase se podría considerar sencillamente como un contador binario de N bits, el cual cuando alcanza su valor máximo vuelve a comenzar desde cero, obteniéndose así el comportamiento de una función rampa como se ve en la figura 13.

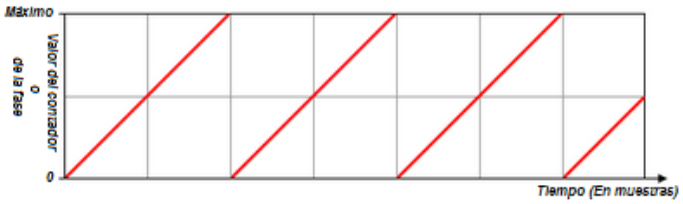

Figura 13. Salida contador binario de N bits en el tiempo.
Tomada de: www.electricdruid.net > Information > Direct Digital Synthesis

Si el contador de N bits realiza saltos, por ejemplo aumenta de a 2 unidades por cada paso de tiempo (Figura 14), se obtendrá a la salida una función rampa con el doble de frecuencia que alcanza su valor máximo en la mitad del tiempo. Así vemos que manipulando los saltos del contador se pueden generar funciones rampa de diferentes valores de pendiente, limitadas por el número de bits N del contador.

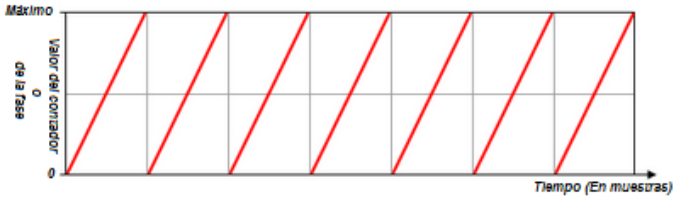

Figura 14. Salida contador binario de N bits con saltos de 2 en 2 en el tiempo.
Tomada de: www.electricdruid.net > Information > Direct Digital Synthesis

La memoria del DDS contiene almacenado un periodo de una onda sinusoidal en 2^N valores. Si la salida del contador de N bits se usa como la dirección de la memoria DDS, cuando el contador haya llegado a su valor máximo se habrá recorrido un periodo de la onda sinusoidal (Figura 15).

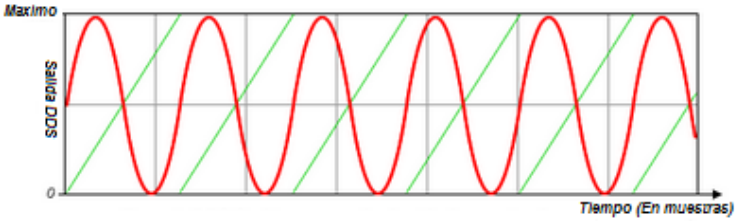

Figura 15. Salida DDS (rojo) vs valor de fase (verde).
Tomada de: www.electricdruid.net > Information > Direct Digital Synthesis

Haciendo un ajuste adecuado entre el número de bits N del acumulador de fase, el tamaño de los saltos que este realiza en cada flanco de reloj y el tamaño de la memoria 2^N, se pueden generar ondas sinusoidales de cualquier frecuencia. En la figura 16 se resume el proceso de generación de ondas por medio de un DDS, en cada flanco de reloj t0, t1,…,t5, el acumulador de fase genera un valor el cual ingresa como dirección a la memoria e indica un determinado punto en el tiempo (primera fila en rojo), obteniéndose así la forma de onda en color azul del centro a la salida del DDS.

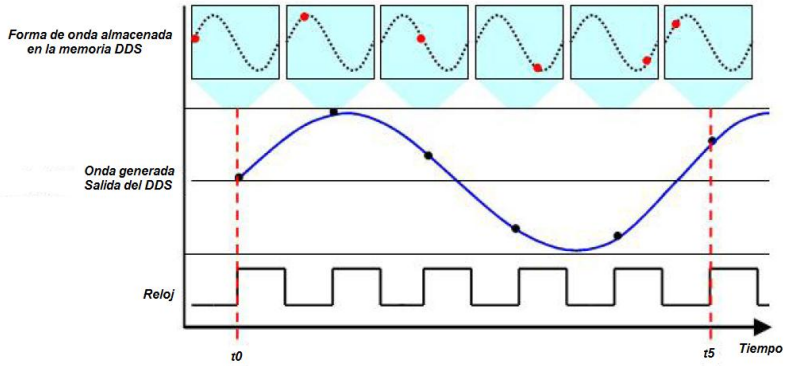

Figura 16. Onda de salida DDS.
Tomada de: "Understanding direct digital synthesis DDS" National Instruments

Hay que resaltar que si por ejemplo se tiene un acumulador de fase de 32 bits, es decir $N=2^{32}$, si estos valores fueran directamente la dirección de la memoria, seria esto correspondería a una memoria de 4gb de capacidad, la cual es muy grande para tarjetas de desarrollo de estudio, por lo cual es común que exista un tercer elemento en la estructura interna de un DDS conocido como SLICER, el cual se encarga de ajustar el ancho de bits de la dirección de memoria al tamaño que esta posea, ya sea mediante redondeo o tomando los bits más significativos y despreciando los demás.

El bloque de *Xilinx* (Figura 17) de *System Generator* emplea una memoria (*look-up table*) que almacena muestras uniformemente espaciadas que representan el periodo de una onda sinusoidal. Un acumulador o integrador digital que es usado para generar un argumento de fase adecuado Θ(n), el cual dirige e indica la secuencia para las muestras almacenadas en el *look-up table* y así obtener la forma de onda de salida deseada.

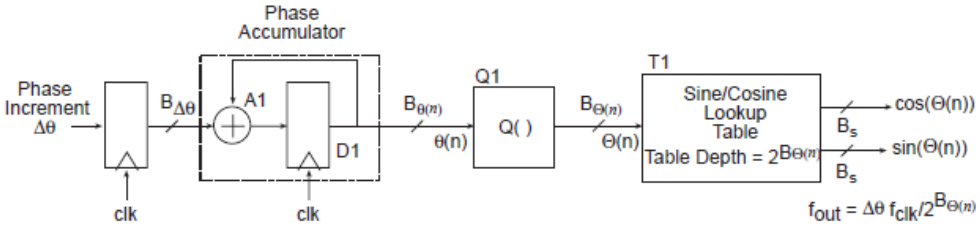

Figura 17. Esquema bloque DDS Compiler v4.0.
Tomada de: *Xilinx* LogiCORE IP DDS Compiler v4.0, DS558 marzo 1 de 2011.

Para entender cómo usar el bloque DDS, es necesario entender como el bloque es implementado en hardware, la figura 17 muestra una vista general de este esquema.

El acumulador de fase (*Phase Accumulator)* consta del sumador A1 y el registro D1, este recibe como entrada un valor de incremento de fase ΔΘ para así ir generando en cada flanco de reloj el argumento de la señal Θ(n) con un ancho de bits BΘ(n). El ancho de Θ(n) es truncado o recortado mediante el *Slicer* Q1 el cual toma la mayor cantidad de los bits más significativos hasta tener el tamaño óptimo y poder usarlo como dirección o índice de la memoria (*sine/cosine Lookup Table)*, asignando el valor fase-espacio en el tiempo.

❖ **Frecuencia de salida**

Puede ser determinada de dos formas según los parámetros de configuración: *Hardware* o *sistema.*

- ***Parámetros de Hardware:***

En la pestaña *basic* se deben especificar el ancho de fase BΘ(n) que corresponde al número de bits a la salida del acumulador, y el ancho de salida Bs que es el número de bits en las salidas sin/Cos de la tabla de búsqueda (rango:3 a 26). En la pestaña *output frecuency* se debe ingresar en binario el valor del incremento de fase ΔΘ, para así determinar la frecuencia de salida según la ecuación 2.

$$f_{out} = \frac{f_{clk} * \Delta_\theta}{2^{B_{\theta(n)}}} \qquad \Delta_\theta = \frac{f_{out} * 2^{B_{\theta(n)}}}{f_{clk}} \qquad (2)$$

- BΘ(n): Ancho de fase (rango: 3 a 48).
- ΔΘ : Valor del incremento de fase: binario sin signo (rango: 0 a $2^{B_{\theta(n)}}$ - 1).
- fclk: frecuencia del reloj del sistema.
- fout: frecuencia de salida deseada.

Por ejemplo para una frecuencia de reloj de 50MHz, un BΘ(n) de 26 bits y un incremento de fase ΔΘ de 1000 (‘1111101000’), se obtendrá una frecuencia de salida de 745Hz.

$$f_{out} = \frac{f_{clk} * \Delta_\theta}{2^{B_{\theta(n)}}} = \frac{50'000.000}{2^{26}} * 1000 \approx 745\ Hz$$

- ***Parámetros de sistema***

En la pestaña *Basic* se deben especificar el rango dinámico SFDR (*Spurious Free Dynamic Range)* el cual corresponde a la diferencia en decibeles entre el armónico fundamental y su siguiente (rango: 18 a 150db); Y la resolución, la cual corresponde al cociente entre fclk y 2^BΘ(n).
En la pestaña *output frecuency* se ingresa directamente el valor de la frecuencia de salida deseada en MHz.
El número de bits a la salida bajo este tipo de configuración teniendo por defecto la opción de corrección de ruido (*noise shapin)* en *None* se obtiene mediante la ecuación 3.

$$Ancho\ de\ salida\ Bsalida = \frac{SFDR}{6} \qquad (3)$$

Por ejemplo para un SFDR de 70, se obtendrán 12 bits de salida.

$$Ancho\ de\ salida\ Bsalida = \frac{70}{6} \approx 12\ bits$$

❖ **Parámetros de configuración particulares del bloque**

- En la pestaña *Basic* (Figura 18) se tienen los siguientes parámetros de configuración:

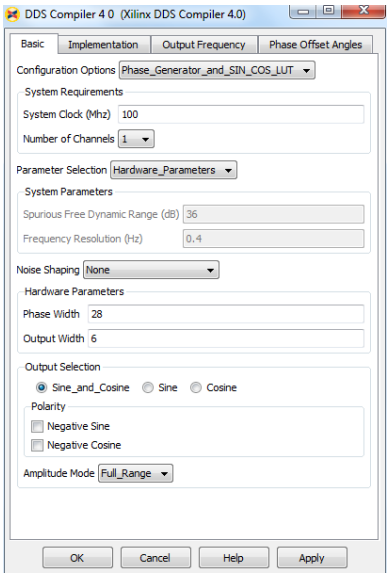

Figura 18. Bloque DDS Compiler v4.0 / Basic.

Configuration options: Permite implementar solo una parte o todos los componentes del DDS. Se puede escoger entre únicamente el generador de fase, o la tabla de búsqueda, o ambas en conjunto. Para la generación de ondas de tipo sinusoidal se recomienda la opcion *Phase_Generator_and_SIN_COS_LUT.*

System Clock: Frecuencia del reloj (en MHz) a la cual se ha de desempeñar el DDS. Es importante resaltar que este valor influye en la frecuencia de salida de las ondas generadas (Ecuación 2) y que este debe ser compatible con la tarjeta de entrenamiento en la que se desee implementar el diseño, por ejemplo la tarjeta Spartan 3AN incluye un oscilador de 50MHz en su terminal E12.

Number of channels: El bloque DDS puede manejar de 1 a 16 canales multiplexados en el tiempo, lo cual disminuye la frecuencia del reloj por canal. Se recomienda dejar en 1.

Parameter selection: Permite escoger bajo que parámetros configurar el bloque y su frecuencia de salida, hay dos opciones parámetros de hardware y parámetros de sistema las cuales ya fueron descritas en la sección frecuencia de salida.

Noise Shaping: En este parámetro se pueden implementar estrategias para mejorar la calidad de las ondas generadas. Se puede aumentar el rango dinámico SFDR mejorando el truncamiento del argumento Θ(n) lo cual involucra un incremento en los recursos requeridos para la implementación. Las opciones son *None*, *Dithering* y *Taylor series corrected*. Se recomienda mantener *None* ya que es la que consume menos recursos y presenta buenos resultados para las frecuencias utilizadas en este proyecto.

Output selection: Permite seleccionar el tipo de onda que se obtendrá a la salida del bloque. Seno, coseno o ambas.

Polarity: Marcando esta opción las salidas del bloque pueden invertirse.

- En la pestaña *implementation* (Figura 19) se tienen:

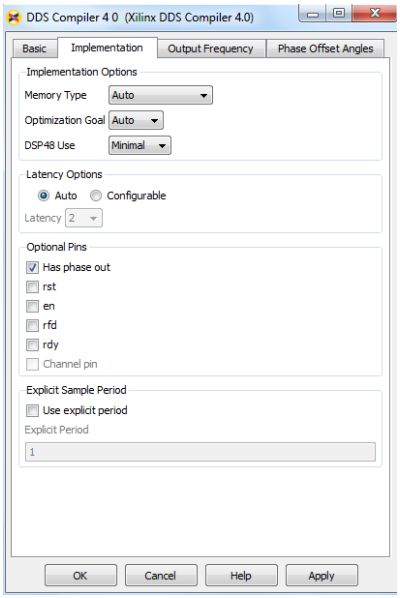

Figura 19. Bloque DDS Compiler v4.0 / Implementation.

Implementation options: Esta sección controla el hardware utilizado para la implementación del bloque. Se puede especificar el tipo de memoria utilizado en la tabla de busqueda SIN/COS y el uso de bloques DSP48 para el acumulador de fase.

Optional pins: Este parametro permite crear entradas o salidas adicionales para configurar el bloque externamente tales como, fase de salida, enable, reset etc…. Se recomienda dejar sin marcar todas las casillas ya que el bloque va a ser configurado en su totalidad internamente.

- En la pestaña *output frecuency (Figura 20)* según el tipo de parámetros de configuración seleccionado para la frecuencia de salida, en esta sección se debe indicar el valor de la frecuencia de salida (MHz) o el valor del incremento de fase ΔΘ en binario.

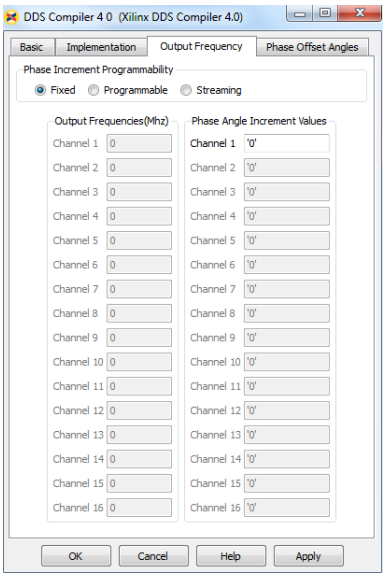

Figura 20. Bloque DDS Compiler v4.0 / Output Frecuency.

2.2.5 MUX

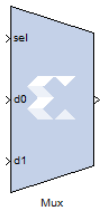

Figura 21. Bloque Mux.

Este bloque (Figura 21) implementa un multiplexor que cuenta con un número de entradas configurable por el usuario y posee una línea de selección (de tipo sin signo), la cual controla la salida que se obtiene en un determinado instante de tiempo.

Las diferentes librerías del *Blockset* de *Xilinx* donde se encuentra este bloque son: *Basic Elements*, *Control Logic* e *Index*.

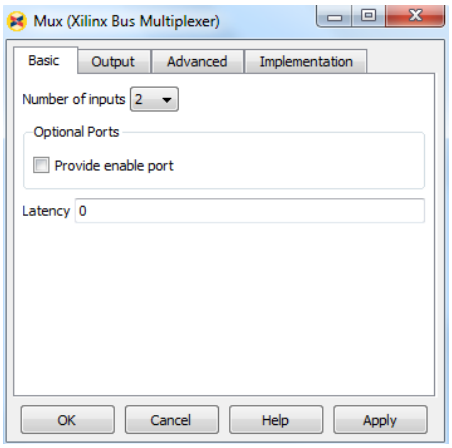

Figura 22. Bloque Mux / Basic.

Number of inputs: Este parámetro permite elegir el número de entradas que se tiene para multiplexar, este valor está comprendido entre 2 y 32.

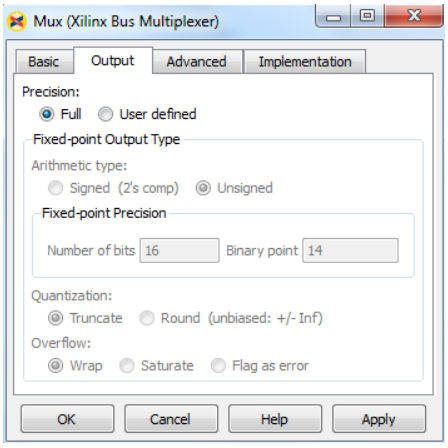

Figura 23. Bloque Mux / Output.

2.2.6 CONSTANT

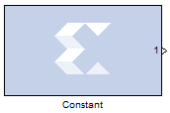

Figura 24. Bloque Constant.

Este bloque (Figura 24) permite implementar el valor de una contante. Este valor debe ser asignado tomando en consideración el número de bits asignado y el tipo de dato ya que puede ser booleano o de punto fijo.

Las diferentes librerías del *Blockset* de *Xilinx* donde se encuentra este bloque son: *Basic Elements, Control Logic, Math* e *Index*.

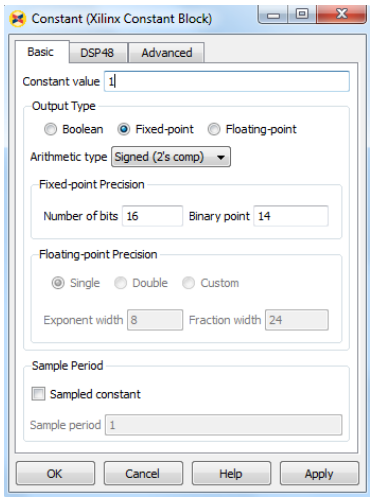

Figura 25. Bloque Constant / Basic.

Constant Value: Permite especificar el valor de la constante.

Output type: En él se determinan la representación aritmética, cantidad de bits y la ubicación del punto binario.

Sample Constant: Al marcar esta opción se asocia un periodo de muestreo con la salida.

2.2.7 REINTERPRET

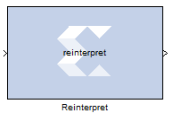

Figura 26. Bloque Reinterpret

Las diferentes librerías del *Blockset* de *Xilinx* donde se encuentra este bloque son *Basic elements*, *Data Types, Floating-point, Math* e *Index*.

Este bloque (Figura 26) fórza a su salida a un nuevo tipo de dato sin alterar la cadena de bits, es decir que la representación binaria pasa a través de el sin cambio alguno, por lo cual este bloque no consume recursos. El número de bits a la salida siempre será el mismo que a la entrada. El bloque permite que el formato de dato sin signo sea reinterpretado con signo o viceversa. Además también permite la reinterpretación de la escala del dato, a través del reposicionamiento del punto binario.

Por ejemplo, si el tipo de dato a la entrada es de 8 bits de ancho, con signo y parte fraccionaria (punto binario) de 2 bits y fuese -10.0 decimal el cual corresponde a 110110.00 en complemento a dos binario.

Si el tipo de dato de salida es reinterpretado como sin signo y 0 bits de parte fraccionaria, se obtendrá 216 decimal o 11011000 en binario.

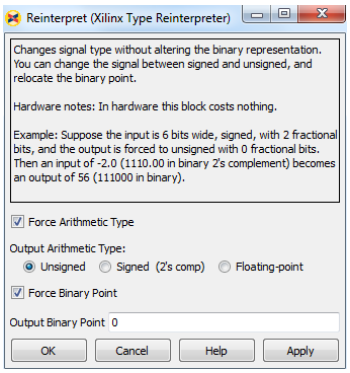

Figura 27. Parámetros específicos bloque Reinterpret

Parámetros específicos del bloque (Figura 27):

Force Arithmetic Type: cuando se selecciona esta opción, se puede escoger el tipo de dato aritmético a la salida, pudiéndose escoger entre sin signo, con signo, o punto flotante.

Forece Binary Point Position: al marcar esta opción, se puede indicar la posición del punto binario, es decir indicar el ancho de la parte fraccionaria que se desea obtener a la salida del bloque.

2.2.8 CONVERT

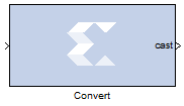

Figura 28. Bloque Convert

Este bloque (Figura 28) convierte cada muestra en su entrada, a un nuevo tipo de dato aritmético. A diferencia del bloque *reinterpret*, este bloque podría llegar a cambiar el número de bits, convierte cualquier tipo de dato en la entrada a cualquier tipo de dato en su salida; el bloque tratará de preservar el valor del dato de entrada si es posible según la configuración que se le especifique, sin embargo se debe tener precaución pues en conversiones de con signo a sin signo, o redondeos se podría alterar el valor de la cadena de bits. Es útil cuando se requiere obtener un tipo de dato aritmético determinado.

Las diferentes librerías del *Blockset* de *Xilinx* donde se encuentra este bloque son *Basic elements*, *Data Types, Floating-point, Math* e *Index*.

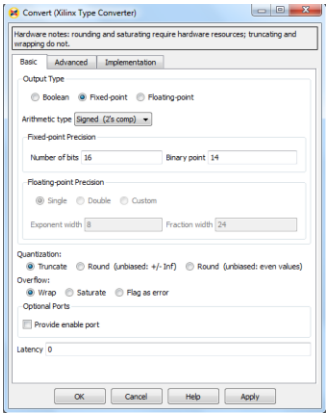

Figura 29. Parámetros básicos del bloque Convert

Los parámetros de configuración para este bloque son los parámetros comunes a los demás, los cuales fueron explicados en la sección previa 2.1, tales como el tipo aritmético de salida, numero de bits, punto binario, comportamiento ante cuantificación, comportamiento para desbordamiento.

2.2.9 DELAY

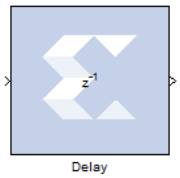

Figura 30. Bloque Delay

Este bloque (Figura 30) permite que los datos de la señal de entrada se presenten después de un cierto tiempo en la salida. Este retardo se especifica por el usuario teniendo en cuenta el periodo de muestreo del sistema. El valor inicial de este bloque es cero y no es configurable.

Las diferentes librerías del *Blockset* de *Xilinx* donde se encuentra este bloque son: *Basic Elements*, *Memory* e *Index*.

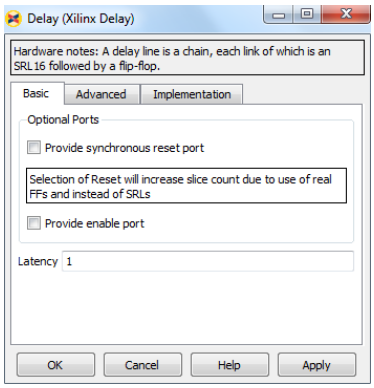

Figura 31. Pestaña Basic bloque Delay

Optional ports: permite agregar un *reset* síncrono y un puerto habilitador *enable.*

Latency: se especifica la cantidad de flancos de reloj que se ha de retardar la señal.

Enable Register Retiming: Al realizar la implementación se usara una cadena de Flip-Flops, si se activa este campo. Cuando se sintetiza de esta forma, se hará uso de una gran cantidad de slices.

2.2.10 COUNTER

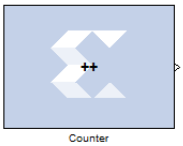

Figura 32. Bloque Counter

El bloque *Counter* (Figura 32) implementa contadores ascendentes, descendentes o ascendentes/descendentes dentro del FPGA. Los contadores generados por este bloque pueden ser limitados a un valor determinado o de cuenta libre; sin embargo en el segundo caso su valor también se verá limitado por el número de bits que maneja en la salida del contador. Adicionalmente, este consume menos recursos que el contador limitado, pues no necesita un comparador.

Las diferentes librerías del *Blockset* de *Xilinx* donde se encuentra este bloque son: *Basic Elements, Control Logic, Math e Index.*

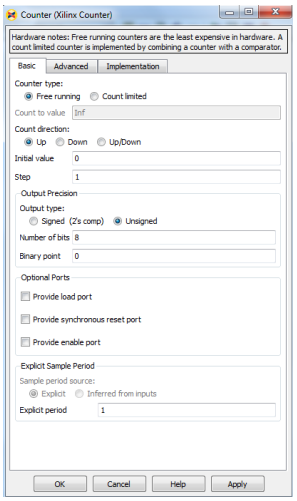

Figura 33. Parámetros específicos bloque Counter

Counter Type: Permite elegir entre contador de cuenta libre o limitado hasta cierto valor.

Count to Value: En este campo se debe colocar el valor al cual se va a limitar el contador. Esta opción permaneces deshabilitada cuando se trata de contador de cuenta libre.

Count Direction: Permite escoger si el contador es ascendente, descendente o ascendente/descendente.

Initial Value: En este campo se debe colocar el valor en el cual se iniciara la cuenta.

Step: Permite colocar el incremente o decremento en la cuenta, dependiendo de la dirección elegida.

Provide Load Port: Esta opción es habilitada solo para contadores de cuenta libre y permite obtener contadores con carga y entradas explicitas.

2.2.11 ADDSUB

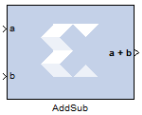

Figura 34. Bloque Addsub

Este bloque (Figura 34) implementa un sumador o un restador dentro del FPGA según se elija o de acuerdo a una señal booleana de entrada.

Las diferentes librerías del *Blockset* de *Xilinx* donde se encuentra este bloque son: *Math e Index*.

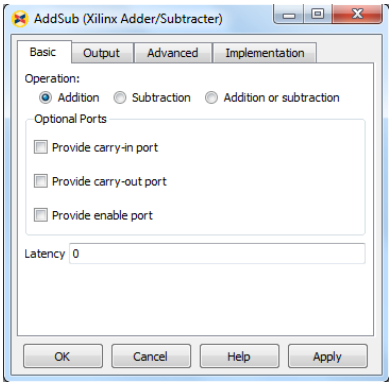

Figura 35. Pestaña Basic bloque Addsub

Operation: Permite definir la operación que realizara el bloque, entre adición, sustracción o adición/sustracción, En este último caso la operación se controla por medio de una señal booleana, de tal forma que se realizara una adición cuando esta señal sea 0 y la sustracción cuando este en 1.

Provide Carry-in Port: Al selecciona esta casilla, se crea una entrada adicional Carry-in (cin).

Provide Carry-out Port: Permite tener una salida de carry-out (cout)

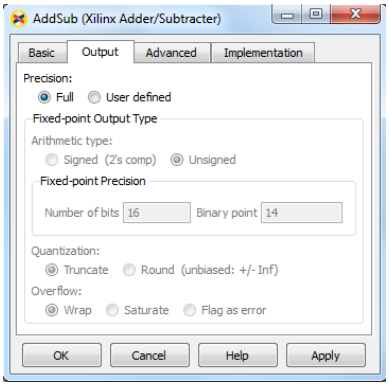

Figura 36. Pestaña Output bloque Addsub

2.2.12 MULT

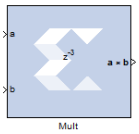

Figura 37. Bloque Mult

Este bloque (Figura 37) implementa un multiplicador, que calcula el producto de los datos que se encuentran en los dos puertos de entrada a y b. este bloque requiere 3 flancos de reloj para realizar el proceso por lo cual agrega un retardo de 3, el cual es indicado gráficamente por el z^{-3}.

Las diferentes librerías del *Blockset* de *Xilinx* donde se encuentra este bloque son *Math e Index*.

En los parámetros de configuración básicos (Figura 38) se puede determinar manualmente el tipo de precisión que se obtendrá a la salida del bloque. La opción por defecto *Full* acomoda el número de bits y tipo de dato aritmético con el fin de que no haya errores por desbordamiento o cuantificación.

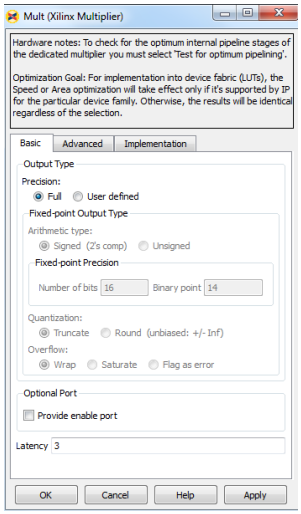

Figura 38. Pestaña Basic bloque Mult

En la pestaña correspondiente a implementación (Figura 39) se pude seleccionar:

Use embedded multipliers: Si se selecciona esta casilla, el multiplicador se implemente con la máxima eficiencia posible, haciendo que mejora la velocidad de procesamiento.

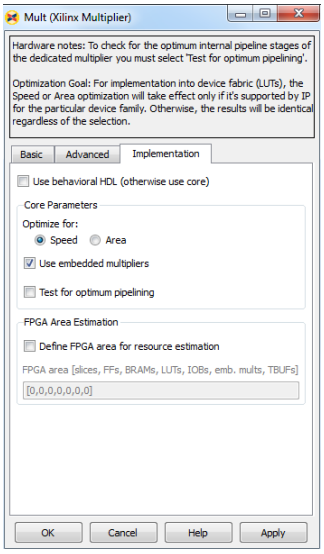

Figura 39. Pestaña Implementation bloque Mult

2.2.13 CMult

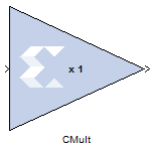

Figura 40. Bloque CMult

Este bloque implementa un operador de ganancia, obteniendo así a su salida el producto de una entrada por una constante.

Las diferentes librerías del *Blockset* de *Xilinx* donde se encuentra este bloque son *Floating-point, Math* e *Index*.

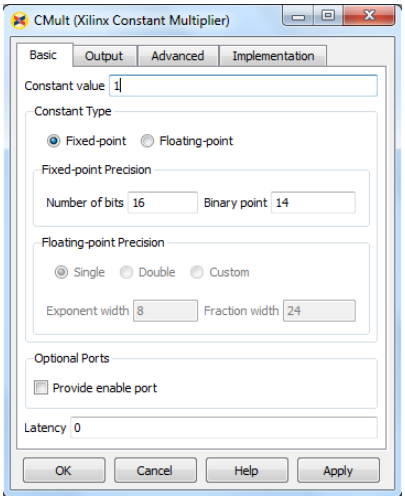

Figura 41. Pestaña Basic bloque CMult

Los parámetros específicos del bloque CMult en la pestaña *Basic* (Figura 41) son:

Value of Constant: Se especifica el valor de la constante, el cual puede ser también una expresión. Si la constante no puede ser expresada exactamente según el tipo aritmético y el número de bits seleccionados, el valor es redondeado y saturado hasta donde le sea posible, un valor positivo es implementado en el formato sin signo, y un valor negativo con signo.

Fixed Point Precision: Determina el número de bits y la ubicación del punto binario en la constante, siendo cero el bit menos significativo.

Provide enable port: Genera una nueva entrada externa de habilitación al bloque.

En la pestaña *Output* (Figura 42) se tiene acceso a los parámetros que permiten escoger el tipo de precisión que se obtendrá a la salida del bloque, estos parámetros se encuentran explicados en la sección 2.1 Opciones Comunes.

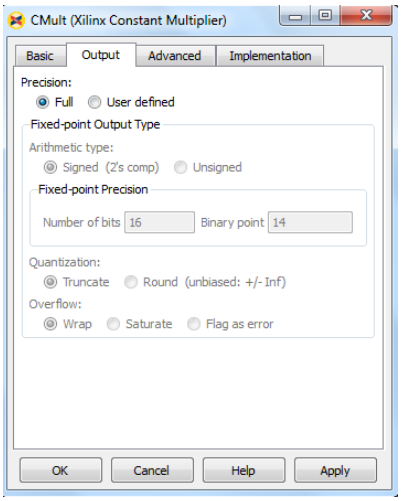

Figura 42. Pestaña Output bloque CMult

En la pestaña *Implementation* (Figura 43) se tienen

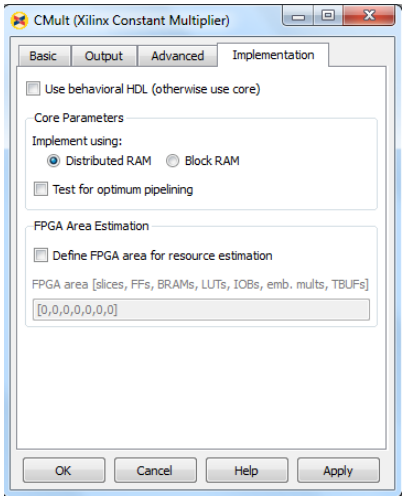

Figura 43. Pestaña *Implementation* bloque CMult

Implement Using: especifica cuando usar en la implementación en hardware del bloque memoria RAM distribuida o bloques RAM.

Test for optimum Pipelining: cuando esta seleccionado, le indica a *SysGen* que haga la implementación en pipeline lo más completa y extensa posible.

2.2.14 SLICE

Figura 44. Bloque Slice

Este bloque (Figura 44) permite extraer una porción de bits de su señal de entrada y crear con ellos un nuevo valor, el cual es presentado a la salida del bloque. El tipo de dato de salida es sin signo con punto binario en la posición cero.

Las diferentes librerías del *Blockset* de *Xilinx* donde se encuentra este bloque son *Basic elements*, *Control Logic*, *Data Types e Index*.

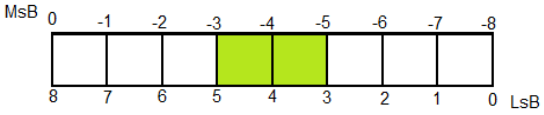

Figura 45. Selección mediante posición de dos bits

El bloque en sus parámetros de configuración (Figura 46) permite varios mecanismos bajo los cuales el rango de bits a extraer puede ser seleccionado. Si el tipo de dato de entrada no es conocido o está sujeto a cambios en el tiempo se puede configurar el bloque para tome como referencia el bit más significativo MSB (*Upper bit location + width*) o el menos significativo (*Lower bit location + width*) y desde allí tome un rango de bits determinado. Si el tipo de dato es conocido se puede especificar también el rango a partir de la ubicación especifica de dos bits (*Two bit locations*) los cuales serían los nuevos MsB y LsB, por ejemplo en la Figura 45 se presenta una señal de 8 bits de la cual se desean extraer a través del bloque *slice* los dos centrales, esto se logra configurando el *offset of top bit* en -3 y el *offset of bottom bit* en 3.

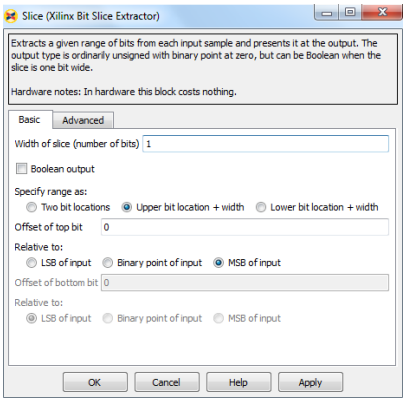

Figura 46. Parámetros básicos del bloque Slice

2.2.15 RELATIONAL

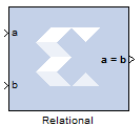

Figura 47. Bloque Relational

Este bloque implementa un comparador, el cual entrega a la salida un bit sin signo, siendo este '1' cuando la comparación es verdadera, y '0' cuando es falsa.

Las diferentes librerías del *Blockset* de *Xilinx* donde se encuentra este bloque son *Basic elements*, *Control Logic, Floating-point, Math* e *Index*.

Las comparaciones que permite implementar en sus parámetros de configuración (Figura 48) son:

Igual a (a=b)
Diferente a (a!=b)
Menor que (a<b)
Mayor que (a>b)
Menor o igual (a<=b)
Mayor o igual (a>=b)

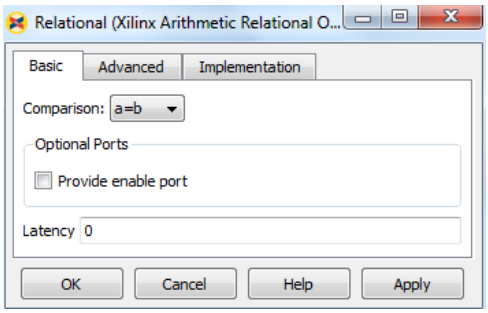

Figura 48. Pestaña Basic bloque Relational

Comparison: se especifica el tipo de comparación a realizar.

Provide enable port: Genera un puerto externo de habilitación.

2.2.16 REGISTER

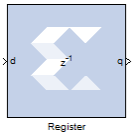

Figura 49. Bloque Register

Este bloque (Figura 49) modela un registro paralelo-paralelo basado en flip flops tipo D, teniendo una latencia de una unidad, es decir que el dato presentado en la entrada aparecerá en la salida después de una muestra de periodo.

El bloque trae por defecto un puerto de entrada para ingresar el dato al registro y otro de salida. Opcionalmente en los parámetros básicos de configuración (Figura 50) se pueden agregar entradas síncronas de *reset*, *enable* y además especificar el valor inicial almacenado en el registro. Al activarse el *reset* el registro asume el valor inicial especificado en los parámetros.

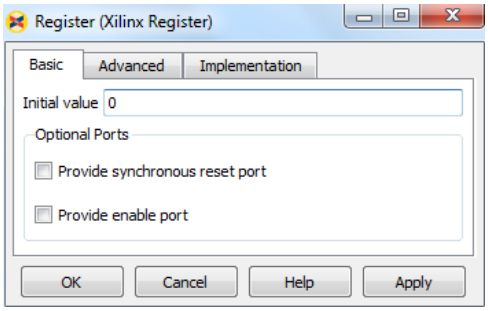

Figura 50. Pestaña Basic bloque Register

Las diferentes librerías del *Blockset* de *Xilinx* donde se encuentra este bloque son *Basic elements*, *Control Logic, Floating-point, Memory* e *Index*.

2.2.17 MCODE

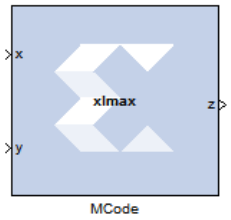

Figura 51. Bloque Mcode

Las diferentes librerías del *Blockset* de *Xilinx* donde se encuentra este bloque son: *Math* e *Index*.

El bloque (Figura 51) permite ejecutar funciones creadas por el usuario en *Matlab* dentro de *Simulink*. Un parámetro en el bloque permite especificar el nombre de la función (archivo .m) de *Matlab* a ejecutarse.

El bloque ejecuta el código para calcular las salidas del mismo cuando se realizan simulaciones en *Simulink*. El mismo código es traducido en forma directa a hardware cuando es implementado.

El bloque Mcode debe cumplir las siguientes tres reglas:

1. El tipo de dato de todas las entradas y salidas del bloque son punto fijo.

2. El bloque debe tener al menos una salida

3. El archivo .m que contiene el código a ejecutar debe estar ubicado en la misma carpeta de trabajo que el archivo de *simulink* .mld que utiliza el bloque.

Parámetros de configuración:

En la pestaña Basic (Figura 52) se especifica el archivo .m que contiene el código a ejecutar en lenguaje de *Matlab*, el botón *Browse* permite seleccionar la ubicación de este archivo. El botón *Edit M-File* permite ver el código y realizar cambios en el.

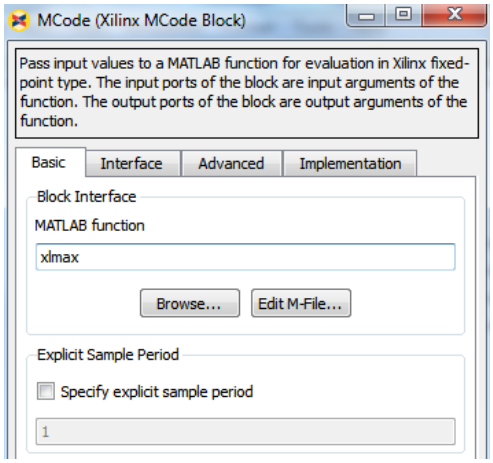

Figura 52. Pestaña Basic/Mcode

En la pestaña Interface (Figura 53) se visualizan las entradas y salidas del bloque, se puede verificar que el código fue bien interpretado y que las entradas y salidas definidas sean compatibles con el tipo de dato de *Xilinx*.

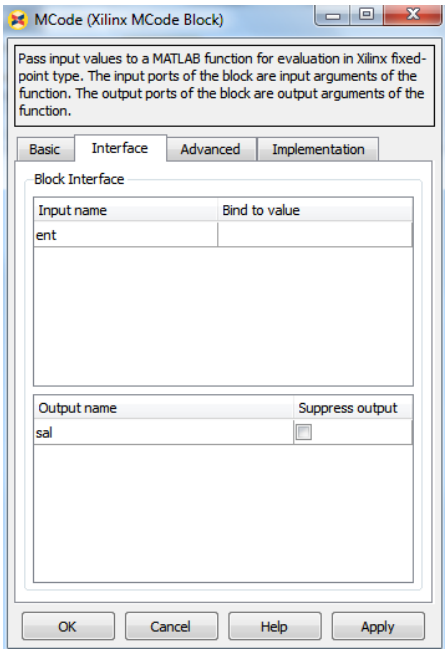

Figura 53. Pestaña Interface/Mcode

El bloque Mcode maneja únicamente ciertos comandos sencillos del lenguaje de *Matlab*, los cuales son suficientes para implementar funciones aritméticas, máquinas de estado y de control.

Comandos y expresiones soportados por el bloque Mcode

- Sentencias de asignacion
- Sentencias simples y complejas de comandos *if/else/elseif/end*
- Sentencias *Switch*
- Expresiones aritmeticas que involucran operaciones de suma o resta
- Multiplicacion y division en potencias de dos
- Los siguiente operadores

<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
==	Igual a
~=	Diferente a
&	And
|	Or
~	Negación (Not)

BIBLIOGRAFÍA

- Jiménez Núñez Sixto Alejando, Panchi Campos Diego Enrique, Ph.D Robin Álvarez R. Diseño e implementación de un modulador y un demodulador N-QAM empleando *Xilinx* ISE, *System Generator* y *Simulink* sobre una tarjeta de entrenamiento basada en un FPGA de *Xilinx*. Facultad de ingeniería eléctrica y electrónica. Escuela Politécnica Nacional, Quito Mayo 2011.

- *Xilinx Blockset Reference Guide.Xilinx System Generator v2.1 for Simulink*, 2007.

- *System Generator for DSP User Guide*, UG640 (v 14.1) Abril 24, 2012.

- Miroslav Knežević, ESAT/SCD-COSIC, Room 01.62, *An introduction to Xilinx System Generator.*

- *Xilinx* LogiCORE IP DDS Compiler v4.0, DS558 marzo 1 de 2011.

- Spartan-3A/3AN FPGA Starter Kit Board User Guide UG334 (v1.1) Junio 19 de 2008.

ANEXO B

INSTALACIÓN XILINX SYSTEM GENERATOR FOR DSP

Para windows 7, Matlab 2011a, Ise Design Suite 13

INTRODUCCIÓN

La Herramienta de *Xilinx System Generator for DSP* trabaja en complemento con el *software Simulink-Matlab*. En esta guía vamos a realizar el proceso de instalación y autorización de la licencia de *ISE Design Suite* incluyendo *System Generator,* asumiendo que *Matlab* 2011a ya se encuentra debidamente instalado en el equipo.

ADQUIRIR LICENCIA

Lo primero que se debe hacer es adquirir una licencia de tipo académica o superior con autorización para utilizar *system generator*, esta puede ser adquirida desde el sitio web de *Xilinx* http://www.xilinx.com/. Para verificar que la licencia adquirida cuenta con los permisos necesarios, el archivo de extensión .lic debe tener incluido a *SysGen* en su sección *"PACKAGE"* como se muestra a continuación:

```
#
PACKAGE System_Edition xilinxd 2012.12 78DCD79AA627 \
COMPONENTS="ChipscopePro ChipscopePro_SIOTK sdk xps
PlanAhead \
ISE ISIM AccelDSP SysGen" OPTIONS=SUITE
#
```

INSTALACIÓN

- Ejecutar el archivo de instalación de *ISE Desing Suite* 13 "setup.exe". se debe abrir una interfaz gráfica de instalación como la de la figura 1.

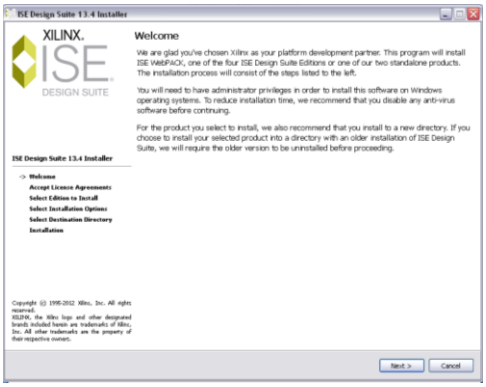

Figura 1. Pantallazo inicial Bienvenida.

- Aceptar los términos de licencia (figura 2), next.

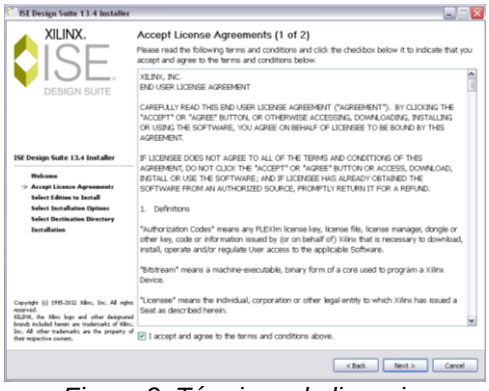

Figura 2. Términos de licencia.

- Seleccionar e Instalar el paquete *"ISE Desing Suite: System Edition"* el cual incluye todas las herramientas de software de *Xilinx*, entre ellas *System Generator for DSP* (figura 3)

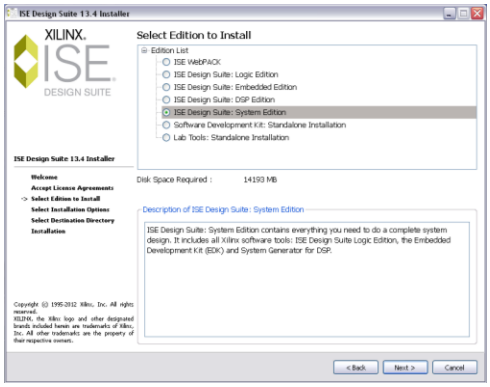

Figura 3. Seleccionar Edición a instalar

- En opciones de instalación, desactive la casilla *Adquire or Manage a licence key (Figura 4) pues* el archivo de licencia con los permisos requeridos ya ha sido obtenido previamente.

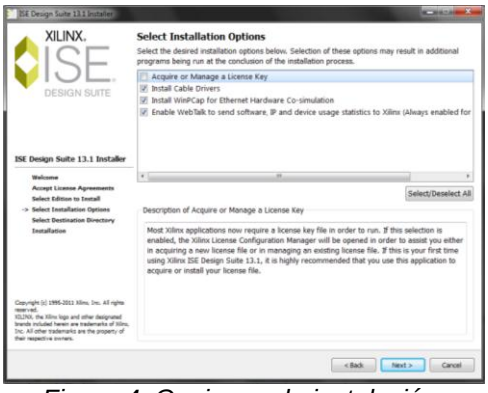

Figura 4. Opciones de instalación.

- Seleccionar la carpeta de destino donde se instalará el software (figura 5). El paquete completo requiere "14.2GB" de espacio en el disco duro aproximadamente. El proceso de instalación debe comenzar.

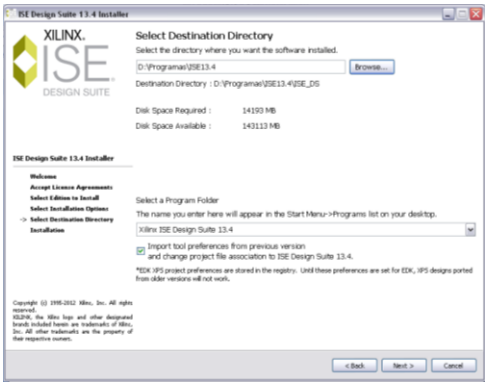

Figura 5. Carpeta de destino.

- Durante la instalación (cerca del 90%) automáticamente se detectan las versiones de *Matlab* que se encuentran instaladas en el equipo (figura 6), seleccionar la versión 2011a que debe estar "no configurada" y seleccionar *OK* para que se sincronice con *system generator.* Con esto termina el proceso de instalación.

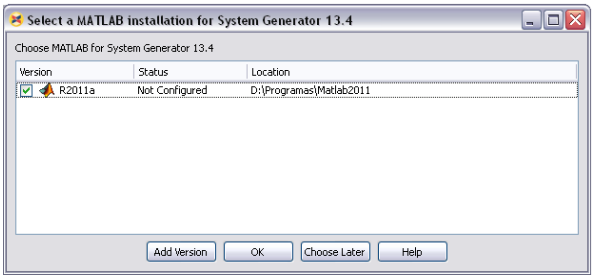

Figura 6. Sincronización Matlab-SysGen.

ACTIVACION DE LA LICENCIA

- Para licenciar debidamente el software instalado diríjase a "Panel de control/sistema/opciones avanzadas del sistema/variables de entorno" y cree una nueva variable de usuario con las siguiente características (figura 7) :

Nombre de la variable: LM_LICENSE_FILE

Valor de la variable: Ubicación completa del archivo de licencia incluyendo su extension .lic

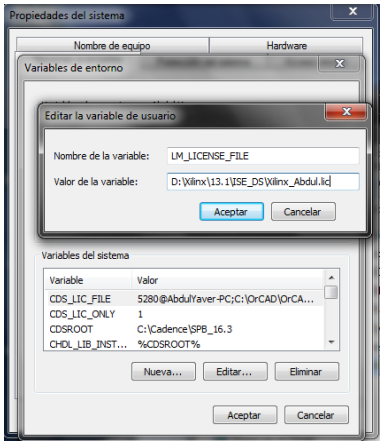

Figura 7. Variables de entorno.

- Para verificar el direccionamiento del archivo de licencia, abrir el *Manage Xilinx Licenses* y debe estar agregada la variable de entorno LM_LICENSE_FILE que se creó previamente (figura 8). Si no lo está, revisar la ubicación del archivo y repetir el paso anterior.

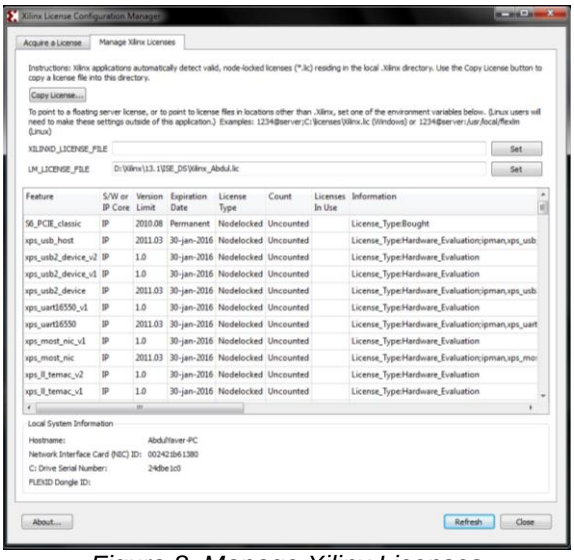

Figura 8. Manage Xilinx Licenses

SIMULINK

- Ejecutar el software *Matlab* y su aplicación *Simulink*. Si es la primera vez se abre una vez instalado *System Generator*, se realiza una actualización y se crean los bloques propios de *Xilinx* (figuras 9 y 10).

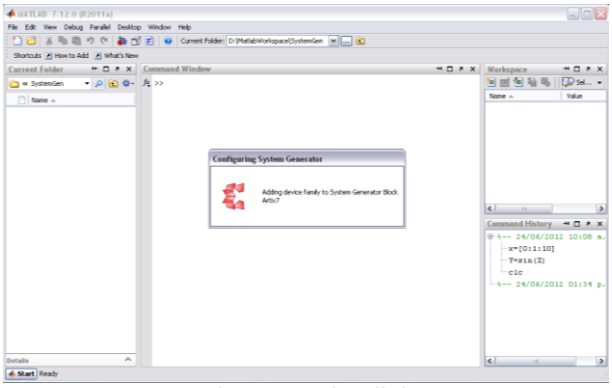

Figura 9. Simulink

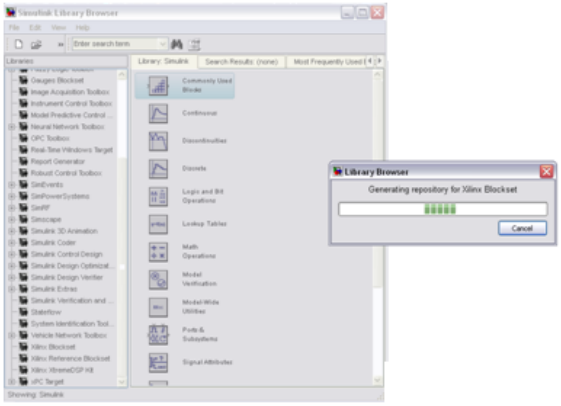

Figura 10. Simulink

Para verificar que los programas ya están instalados y sincronizados correctamente, en la ventana principal (*command prompt)* de *Matlab* escribir: *xlVersion.* Debe aparecer la versión y ubicación de *System Generator* como se muestra a continuación*:*

```
>> xlversion

Available System Generator installations:
Version 13.4.4236 in D:/ISE13.4/ISE_DS/ISE/sysgen
Current version of System Generator is 13.4.4236.
```

Ya en este punto se pueden realizar diseños, simular su comportamiento y exportar el código VHDL para la configuración del Hardware.

Para probar la herramienta vamos a realizar un ejemplo sencillo:

Cree un nuevo proyecto de *Simulink* y realice la conexión mostrada en la figura 11, con los siguientes bloques:

- Xilinx Blockset/Basic Elements/System Generator.
- Xilinx Blockset / Basic Elements / Gateway Out.
- Xilinx Blockset / Basic Elements / Counter.
- Simulink / Display / Scope.

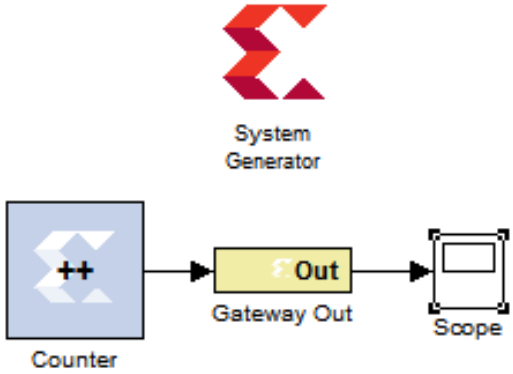

Figura 11. Ejemplo de prueba

El contador viene configurado por defecto como un contador libre de 8 bits, ascendente, de valor inicial 0, con pasos de 1, sin signo y periodo explicito 1. Haciendo doble *click* sobre él se pueden verificar.

Conservando los parámetros de configuración internos que vienen por defecto en cada bloque, simule el diseño para un tiempo de 500 segundos y debe obtenerse en el *Scope* el comportamiento de la figura 12. El contador avanza de uno en uno hasta donde le permiten sus 8 bits, es decir 255, valor en el cual se desborda y vuelve a iniciar desde cero.

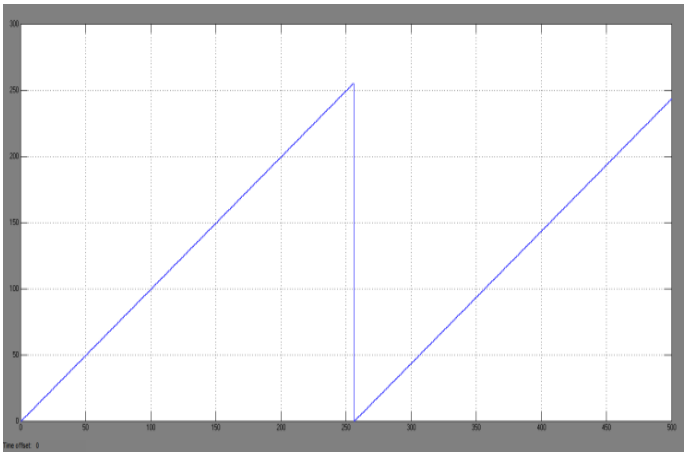

Figura 12. Simulación ejemplo de prueba

ANEXO C

LABORATORIO No 1 DE COMUNICACIONES DIGITALES

Generación De Ondas Seno Y Coseno

Utilizando Xilinx System Generator e implementando en FPGA Spartan 3AN

OBJETIVOS

- Dominar el entorno de *Simulink*, *System* Generator e *ISE* para el desarrollo de sistemas de procesamiento digital de señales.
- Generar ondas seno y coseno usando el bloque *DDS compiler* de *Xilinx* y verificar su comportamiento en el tiempo

BIBLIOGRAFÍA

- *Anexo A del proyecto de grado "Diseño de prácticas para laboratorio de Comunicaciones Digitales basado en Simulink y Xilinx System Generator".*
- *Spartan-3A/3AN FPGA Starter Kit Board User Guide (Datasheet)*
- *LogiCORE IP DDS Compiler v4.0 (Datasheet)*

ACTIVIDAD DIRIGIDA

Se genera una onda Sinusoidal para ver su forma en un *Scope* de Simulink conectando de la manera mostrada en la fig. 1. los siguientes bloques de Xilinx en Simulink:

- Xilinx Blockset/Basic Elements/SystemGenerator.
- Xilinx Blockset/Basic Elements/Gateway Out.
- Xilinx Blockset/DSP/DDS compiler V4.0.

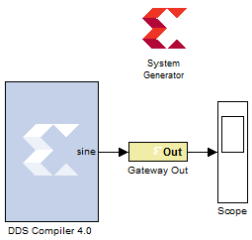

Fig. 1. Conexión para generar una onda sinusoidal.

Podemos acceder a la configuración del bloque *DDS Compiler* haciendo doble clic en él.

En la pestaña *Basic* establecemos el reloj del sistema en 50 MHz; el número de bits para el ancho de fase y ancho de salida de la señal, en 16 y 12 respectivamente; también elegimos como salida solo una onda tipo Seno. Todos los detalles se encuentran en la fig. 2.

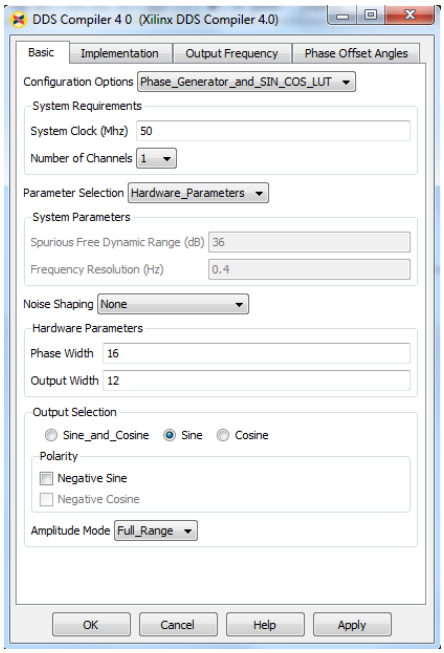

Fig. 2. Pestaña *Basic* del *DDS Compiler 4.0*.

Las fórmulas para establecer la frecuencia de la señal de salida o en su defecto el incremento del ancho de fase son:

$$f_{out} = \frac{f_{clk}\Delta_\theta}{2^{B_{\theta(n)}}} \qquad \Delta_\theta = \frac{f_{out}2^{B_{\theta(n)}}}{f_{clk}} \quad (1)$$

Donde, f_{out} es la frecuencia para la señal de salida, f_{clk} es la frecuencia del reloj de la tarjeta (50 MHz), $B_{\theta(n)}$ es el número de bits para el ancho de fase (16 en nuestro caso, entre más grande mejor resolución para la señal de salida), y Δ_θ es el incremento del ancho de fase.

El valor del incremento del ancho de fase se modifica en la pestaña *output frecuency*, por ejemplo establecimos un valor correspondiente a 32 decimal ('100000' en binario) como lo muestra la fig. 3, y basados en la ecuación (1) se obtiene una frecuencia de salida de 24.414 kHz.

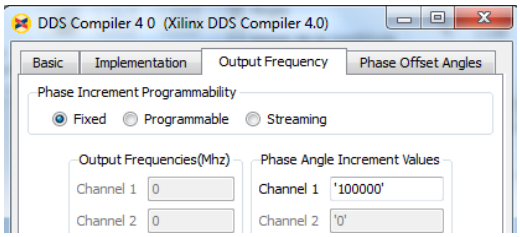

Fig. 3. Pestaña *Output Frequency* del *DDS Compiler 4.0*.

Finalmente, estableciendo un tiempo de 5000 para simulación, teniendo en cuenta que cada segundo en *simulink* corresponde a un flanco de reloj, y pulsando *ejecutar* (►), empieza la compilación, cuando ésta termine, veremos como resultado nuestra onda sinusoidal haciendo doble clic en el *Scope*, esto se puede observar en la fig. 4.

Es importante resaltar que el periodo de la señal en muestras está determinado a partir de la formula:

$$Número\ de\ muestras\ por\ periodo = \frac{f_{clk}}{f_{out}} \quad (2)$$

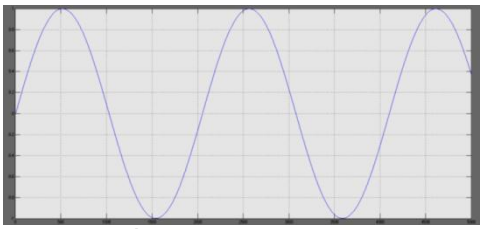

Fig. 4. Simulación de la onda sinusoidal generada.

Ahora vamos a realizar el esquema de la fig. 5. para generar e implementar en la tarjeta Spartan 3AN una onda seno y otra coseno con el bloque *DDS Compiler* (seleccionando *Sine_and_Cosine* en la pestaña *Basic*) de frecuencia 1kHz y amplitud pico a pico de 3.3 Volts. Buscamos adicionalmente los siguientes bloques de Xilinx en Simulink:

- Xilinx Blockset / Basic Elements / Constant.
- Xilinx Blockset / Basic Elements / Reinterpret.
- Xilinx Blockset / Basic Elements / Convert.
- Xilinx Blockset / Math / Addsub.

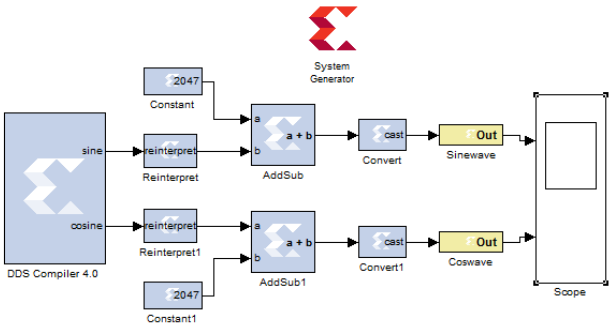

Fig. 5. Conexión para generar una onda senoidal y otra cosenoidal de frecuencia 1kHz.

Para poder visualizar en el osciloscopio las señales generadas en el FPGA, es necesario hacer uso del DAC que viene incluido en la tarjeta. Este maneja protocolo SPI y consta de canales de 12 bits sin signo (0 a 4095). Debido a esto es necesario adecuar las ondas generadas para que puedan ser interpretadas y convertidas por el DAC. Para ellos se sigue el siguiente procedimiento:

Cada una de las señales que genera el DDS son de 12 bits con signo, el bit más significativo (*MSB)* es el signo y los once restantes son de amplitud con punto binario ubicado en el bit 11, es decir que sus valores se encuentran centrados en cero y comprendidos entre -1 y 1 (realmente entre -0.998046875 y 0.998046875).

Éstas señales serán pasadas por un bloque *Reinterpret* que mantendra el bit de signo y ubicara el punto binario en el bit cero o *LSB* de manera que las ondas seguiran centradas en cero y tomaran el rango de -2047 a 2047 (ver fig. 6).

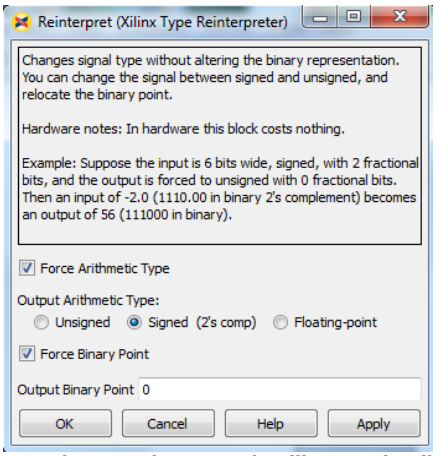

Fig. 6. Bloque Reinterpret de Xilinx en Simulink

Para que las ondas queden oscilando siempre en valores positivos, a través del bloque *AddSub* sumamos una constante de 2047 (ver fig. 7), así la onda oscilará entre 0 y 4094, y será interpretada por el DAC como una onda de 0 a 3.3 Volts.

El bloque Convert (Fig. 8) asegura 12 bits de salida, esto para efectos del ancho de las señales que se generarán como código en VHDL y ademas convierte la señal a Unsigned asegurando un comportamiento entre los valores positivos únicamente, todo esto para poder enviar correctamente la señal al DAC de la tarjeta.

Una vez completado el esquema y configurados los bloques adecuadamente se procede a la generacion del codigo VHDL del diseño realizado. Para esto hacemos doble click en el bloque Token system generator, en él seleccionamos la tarjeta en la que vamos a implementar el proyecto, el lenguaje de descripcion de hardware, la ubicación del archivo

generado y el tipo de compilacion en HDL Netlist, como se muestra en la fig. 9.

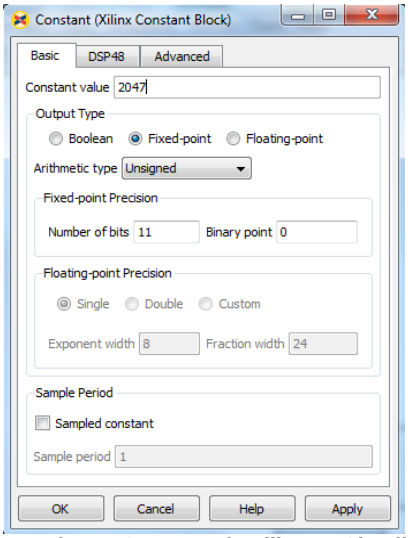

Fig. 7. Bloque Constant de Xilinx en Simulink

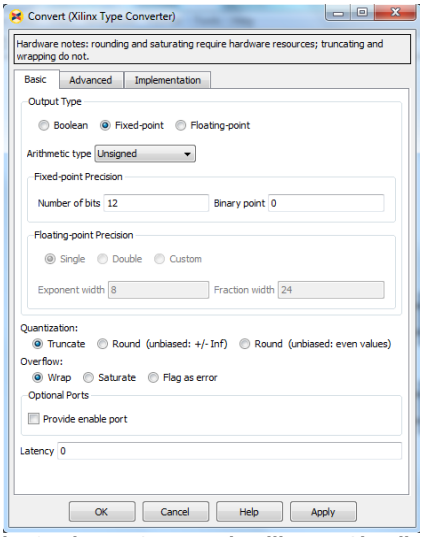

Fig. 8. Bloque Convert de Xilinx en Simulink

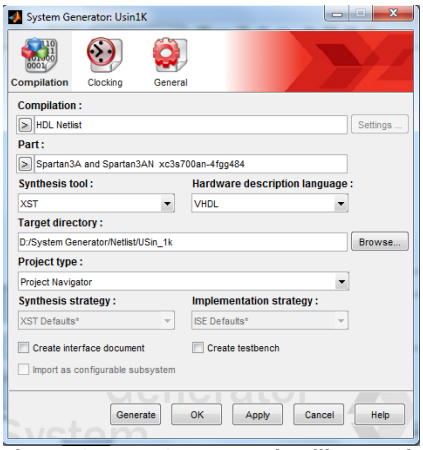

Fig. 9. Bloque System Generator de Xilinx en Simulink.

Adicionalmente en la pestaña Cloking se escoje la frecuencia del reloj que en este caso sera 50MHz o 20ns (ver fig. 10.), pues nuestra tarjeta Spartan 3AN trabaja a dicha frecuencia, y se procede a generar el código haciendo clic en *Generate*.

Una vez terminada la compilación del código, abrimos un nuevo proyecto de ISE y creamos una fuente (módulo VHDL) que a de tener como entradas un reloj principal y dos pulsadores (start, reset) y como salidas un led, y cuatro señales dedicadas al DAC (MOSI, REL, SC, CLR).

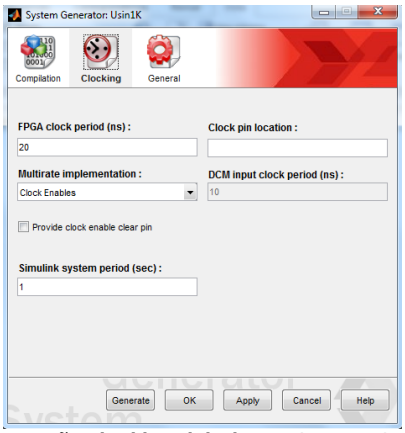

Fig. 10. Pestaña clocking del Bloque System Generator.

Como primera medida agregamos como *component* la fuente “DACmain” (suministrada por el docente) la cual realiza la comunicación SPI de los cuatro canales del DAC de la Spartan 3AN, (ver fig. 11). La salida denominada “LED” prende un diodo LED que confirma que la implementación se realizo satisfactoriamente.

Seguidamente agregamos como *component* la fuente de system generator (.sgp) que generamos desde Simulink, (ver fig. 13.) la cual tendrá las salidas que definimos en el diseño, es decir las ondas seno y coseno, y como entradas un reloj y un *enable*. El *enable* se debe dejar habilitado siempre en ‘1’.

```
COMPONENT DACmain
PORT(
   canal1 : IN std_logic_vector(11 downto 0);
   canal2 : IN std_logic_vector(11 downto 0);
   canal3 : IN std_logic_vector(11 downto 0);
   canal4 : IN std_logic_vector(11 downto 0);
   start_button : IN std_logic;
   rst_button : IN std_logic;
   clk_4ch : IN std_logic;
   OKled : OUT std_logic;
   DAC_MOSI : OUT std_logic;
   DAC_CLR : OUT std_logic;
   DAC_SC : OUT std_logic;
   DAC_CLK : OUT std_logic
   );
END COMPONENT;
```

Fig. 11. Instanciacion DACmain en Xilinx ISE

```
Inst_DACmain: DACmain PORT MAP(
    canal1 => ,
    canal2 => ,
    canal3 => ,
    canal4 => ,
    start_button => ,
    rst_button => ,
    clk_4ch => ,
    OKled => ,
    DAC_MOSI => ,
    DAC_CLR => ,
    DAC_SC => ,
    DAC_CLK =>
);
```

Fig. 11. Instanciacion DACmain en Xilinx ISE

En la fuente *Top module* conectamos directamente el reloj principal *clks* a los demas relojes del sistema (clk_4ch y clk). Las ondas seno y coseno que salen del bloque de system generator se conectan a dos de las entradas del DACmain (canal1 y canal2), los otros dos canales (canal3 y canal4) se conectan a un valor '0'. Los pulsadores de entrada del top module *start* y *reset* se conectan a las entradas *start_button* y *rst_button* de la fuente DACmain. Las salidas del DACmain se conectan a las salidas de la fuente *top module* asi:

OKled => LED
DAC_MOSI => MOSI
DAC_CLR => CLR
DAC_SC => SC
DAC_CLK => REL

En la fig. 12. se puede observar en diagrama de bloques de la fuente principal del proyecto, y en su interior las fuentes de system generator (generada desde simulink) y el DACmain. Los canales 3 y 4 serán asignados a una señal "000000000000" (0 V).

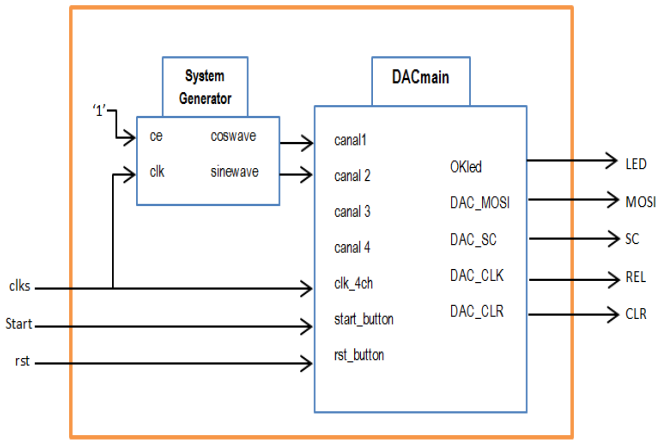

Fig. 12. Esquema de conexión

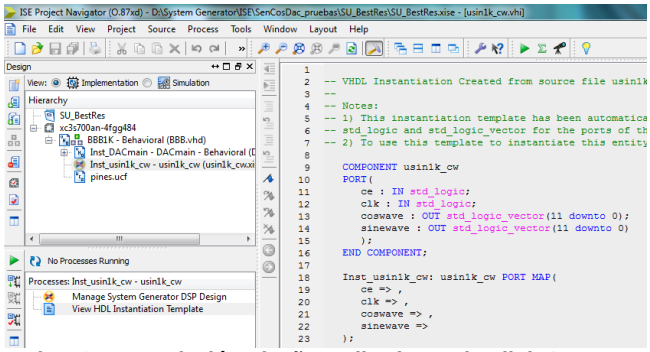

Fig. 13. Instanciación Diseño realizado en simulink-System generator en Xilinx ISE.

A continuacion se debe crear una fuente *constraints* para agregar los *UCF Location Constraints* como se muestra en la fig. 14.

Ahora se puede implementar el diseño y generar el archivo de programacion .bit que finalmente será cargado a la tarjeta a través de la extensión *IMPACT* deshabilitando la opción de verificación (*verify)* y programanto el FPGA únicamente (*Program FPGA only*).

```
# ==== Reloj 50 MHz ====
NET "clks" LOC = "E12"| IOSTANDARD = LVCMOS33 ;
# ==== Leds ====
NET "LED" LOC = "R20" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
# ==== Pulsadores ====
NET "start" LOC = "T16" | IOSTANDARD = LVCMOS33 | PULLDOWN ;
NET "rst" LOC = "T14" | IOSTANDARD = LVCMOS33 | PULLDOWN ;
# ==== DAC ====
NET "MOSI" LOC = "AB14" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "REL" LOC = "AA20" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "SC" LOC = "W7" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "CLR" LOC = "AB13" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
```

Fig. 14. UCFs del proyecto en Xilins ISE

La fig. 15. es la gráfica tomada del osciloscopio en donde se puede verificar el funcionamiento del diseño realizado en simulink, se observan las ondas seno y coseno con frecuencia de 1KHz.

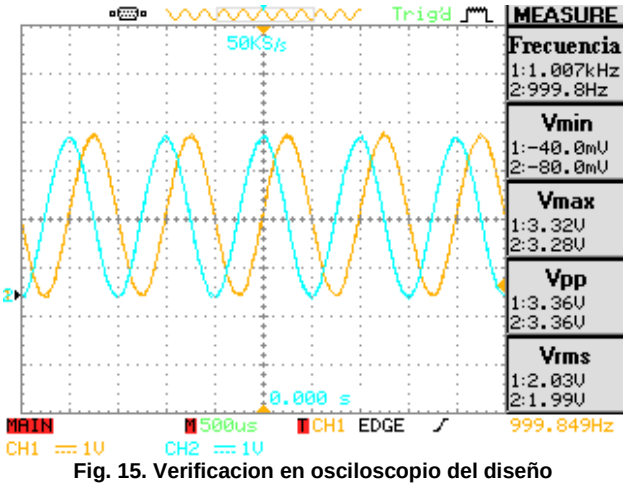

Fig. 15. Verificacion en osciloscopio del diseño

ACTIVIDAD ADICIONAL

Genere dos señales sinusoidales pero con diferente frecuencia, simúlelas en *Simulink* e impleméntelas, la primera a una frecuencia de α kHz, donde α es el último dígito del código estudiantil del primer integrante en orden alfabético (apellidos), la segunda a una frecuencia de β kHz, donde β es el último dígito del código estudiantil del último integrante en ordén alfabético (apellidos), si dicho dígito es cero, se reemplaza por 1.

ANÁLISIS DE RESULTADOS

En grupos de tres personas, se entregará un informe despues de realizada la práctica; éste debe contener:

- Título de la práctica.
- Nombre y código de los estudiantes.
- Objetivos, tanto los señalados en la guía, como los planteados por los estudiantes.
- Análisis y descripción detallada del procedimiento para realizar la actividad adicional.
- Conclusiones
- Observaciones y sugerencias (si las hay) para el mejoramiento de la metodología empleada.

DESARROLLO ACTIVIDAD ADICIONAL

Para visualizar dos señales sinusoidales pero con diferente frecuencia, por ejemplo de 1 kHz y otra de 2 kHz, se introducen dos bloque *DDS Compiler* cada uno con un valor para el incremento del ancho de fase correspondiante a 1342 (010100111110) y 2684 (’101001111100’) respectivamente, es decir una onda será del doble de frecuencia que la otra, y se realiza el esquema tal y como se muestra en la fig. 16. Para que el Scope tenga dos entradas simplemente hacemos clic derecho en él, luego propiedades y digitamos “2” en Inputs.

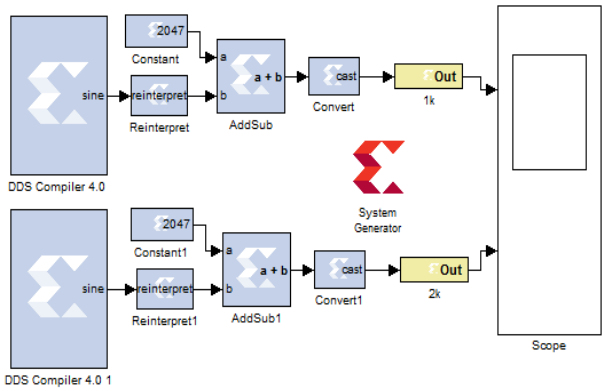

Fig. 16. Conexión para generar dos ondas sinusoidales de diferente frecuencia.

De esta manera se generan dos ondas sinusoidales de diferente frecuencia usando los bloques de Xilinx en Simulink. En la fig. 17 se observa la simulación estableciendo un tiempo de 50000.

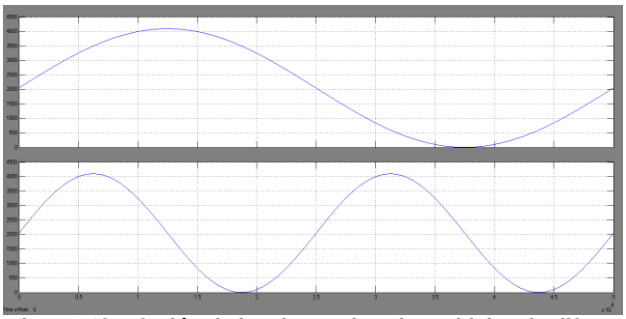

Fig. 17. Simulación de las dos ondas sinusoidales de diferente frecuencia generadas.

Luego siguiendo los procedimientos explicados en la guía se implementan las dos señales con frecuencias de 1kHz y 2kHz respectivamente, asignándolas a dos de los cuatro canales del DAC. Los resultados en el osciloscopio se observa en la fig. 17.

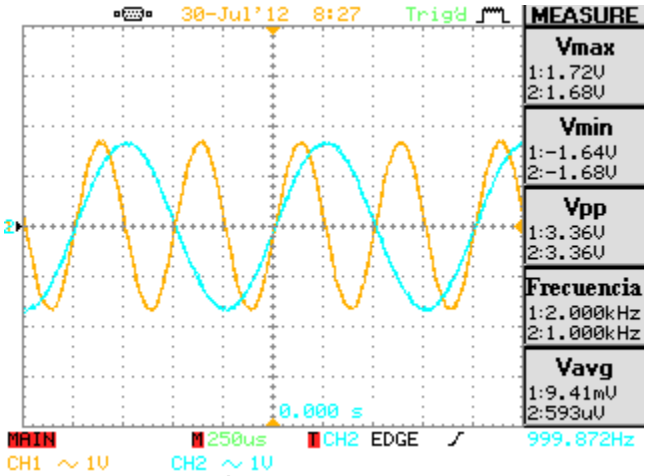

Fig. 17. Verificación en el osciloscopio del diseño.

LABORATORIO No 2 DE COMUNICACIONES DIGITALES

Modulación por Desplazamiento de Amplitud ASK

Utilizando Xilinx System Generator e implementando en FPGA Spartan 3AN

OBJETIVOS

- Reconocer parámetros configurables para el correcto procesamiento de las señales digitales.
- Verificar el comportamiento en el tiempo de la señal de salida correspondiente a la modulación por desplazamiento de amplitud de la señal de referencia.

BIBLIOGRAFÍA

- *Anexo A del proyecto de grado "Diseño de prácticas para laboratorio de Comunicaciones Digitales basado en Simulink y Xilinx System Generator"*
- *Spartan-3A/3AN FPGA Starter Kit Board User Guide (Datasheet)*
- *LogiCORE IP DDS Compiler v4.0 (Datasheet)*
- *Sistemas de Comunicaciones Electrónicas,* ***Wayne Tomasi****, 4 edición, capitulo 12 Comunicaciones Digitales.*

ACTIVIDAD DIRIGIDA

Simulación e implementación modulación OOK

Realizar el montaje de la fig. 1 que describe la modulación OOK, que es la modulación por desplazamiento de amplitud (ASK) más sencilla, con los siguientes bloques de *Xilinx* en *Simulink*:

- Xilinx Blockset / Basic Elements / System Generator.
- Xilinx Blockset / Basic Elements / Gateway Out.
- Xilinx Blockset / Basic Elements / Gateway In.
- Xilinx Blockset / Basic Elements / Constant.
- Xilinx Blockset / DSP / DDS compiler V4.
- Xilinx Blockset / Basic Elements / Mux.
- Simulink / Sources / Counter Free-Running.
- Xilinx Blockset / Basic Elements / Reinterpret.
- Xilinx Blockset / Basic Elements / Convert.
- Xilinx Blockset / Math / CMult.
- Xilinx Blockset / Math / Addsub.

Luego defina una frecuencia de salida de 1 kHz, para esto en el bloque *DDS Compiler* se establece 10 bits de ancho para la señal de salida (es decir de -511 a 511 en valores digitales luego de pasar por el bloque *Reinterpret*), 26 bits para el ancho de fase y un incremento de ancho de fase de 1342 ('010100111110').

Teóricamente la modulación ASK se define para cambios de amplitudes equidistantes y con la inclusión de la señal en modo *Off*. La ecuación (1) define la diferencia de amplitud pico a pico entre cada nivel de la modulación, dependiendo del número de niveles y del voltaje máximo pico a pico que se quiere.

$$\Delta V = \frac{V_{ppMAX}}{N-1} \tag{1}$$

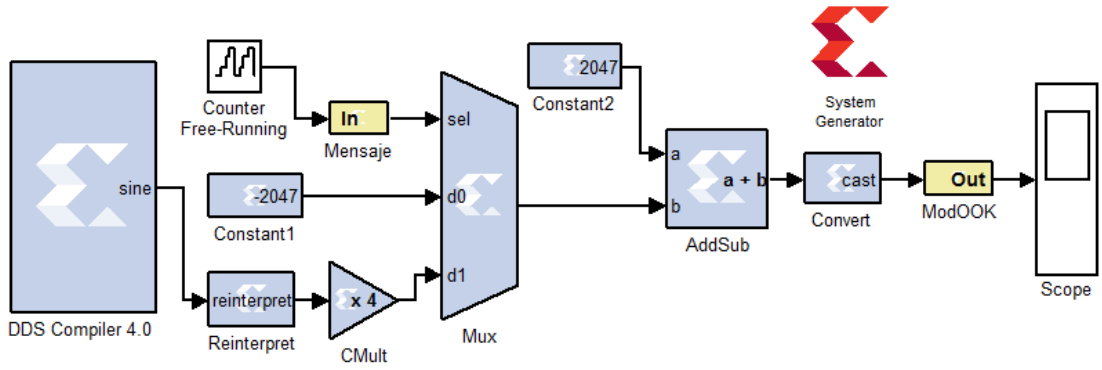

Fig. 1. Conexión para generar Modulación OOK.

Donde N es el número de niveles de amplitud, en éste caso 2 (1 bit) que son el cero y un nivel más que será de 3.3 Volts (voltaje pico a pico máximo que puede interpretar el DAC de la tarjeta, que corresponden a 4096 valores digitales pico a pico).

Los bloques *CMult* multiplican la señal en un factor determinado; para ésta modulación OOK tendremos en la salida valores de 0 Volts o 3.3 Volts pico a pico, por esta razón el bloque *Cmult* es x4, pues 511 x 4 = 2044 (fig 2), que al aplicarle el offset de 2047 se obtiene la señal siempre positiva sobre el eje horizontal con valores comprendidos entre 0 y 4096.

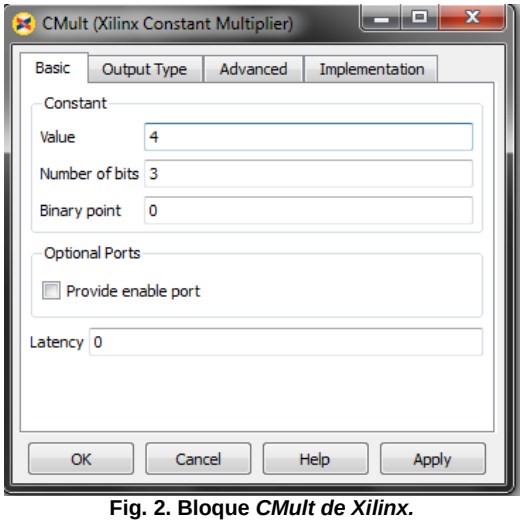

Fig. 2. Bloque *CMult de Xilinx.*

El primer canal del MUX tiene una constante de -2047 para que al ser sumada con el offset quede en 0, y sea nuestro *Off* en la modulación.

La fig. 3 muestra la simulación para la modulación OOK con los niveles de 0 Volts y 3.3 Volts pico a pico, estableciendo como habilitador del *Mux* un contador de 1 bit. El bloque *Gateway In* debe estar configurado para recibir 1 bit también.

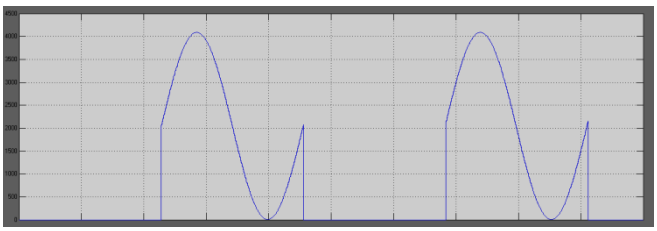

Fig. 3. Simulación de la modulación OOK.

Se genera el código para incluirlo como *Component* en un nuevo proyecto enlazado con el DAC, tal y como se explicó en la práctica anterior, estableciendo una salida (para la señal modulada) y una entrada manejada a través de un *switch*.

Los resultados de la implementación para la modulación OOK se observan en la fig. 4 y en la fig. 5.

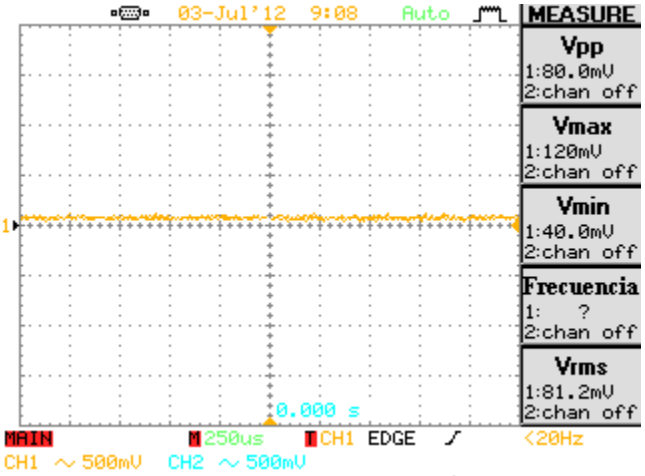

Fig. 4. Modo *Off* para la modulación OOK.

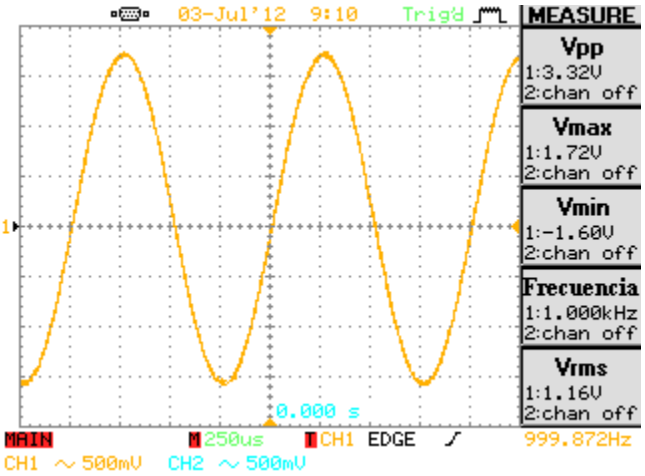

Fig. 5. Señal modulada a 3.3 Volts pico a pico.

ACTIVIDAD ADICIONAL

Simule e implemente una modulación ASK de 4 niveles con las características mostradas en la tabla 1 dependiendo del número asignado para su grupo.

Grupo	Vpp máximo [Volts]	Frecuencia [Hz]
1	3.3	1000
2	3.0	1500
3	2.7	2000
4	2.4	2500
5	2.1	3000
6	1.8	3500
7	1.5	4000
8	1.2	4500
9	0.9	5000
10	0.6	5500

Tabla 1. Vpp máximo y frecuencia para cada grupo.

ANÁLISIS DE RESULTADOS

En grupos de tres personas, se entregará un informe despues de realizada la práctica; éste debe contener:

- Título de la práctica.
- Nombre y código de los estudiantes.
- Objetivos, tanto los señalados en la guía, como los planteados por los estudiantes.
- Análisis y descripción detallada del procedimiento para realizar la actividad adicional.
- Conclusiones
- Observaciones y sugerencias (si las hay) para el mejoramiento de la metodología empleada.

DESARROLLO ACTIVIDAD ADICIONAL

Se desarrollará la actividad para una señal referencia con frecuencia de 1 kHz y un voltaje pico a pico máximo de modulación de 3.3 Volts.

Se realiza el montaje de la fig. 6, dónde los bloques de *CMult* multiplican la señal en un factor determinado para que los cambios en la amplitud de la señal sean equidistantes, sabiendo que la máxima amplitud será de 3.3 Volts. Entonces tomando como referencia la ecuación (1) cada nivel aumentará en 1.1 Volts, es decir, obtendremos amplitudes pico a pico de 0 V, 1.1 V, 2.2 V y 3.3 V.

Por lo anterior los aumentos en posiciones pico a pico serán de 1365, 2730 y 4096, entonces buscamos los factores de multiplicación para 511 que den la mitad de cada uno de dichos valores, por ejemplo para la amplitud de 1.1 Volts, tenemos 511 x K = 1365/2, es decir K = 1.335942, esta será la constante que se pondrá en el *CMult*, pero como este tiene que ser de una cantidad de bits determinada, se tomó de 1.3359375 que corresponden a 8 bits con punto binario después del bit 7 (‘1.0101011’), tal y como se muestra en la fig. 7.

De igual forma se realiza el mismo procedimiento para la constante de la señales de 2.2 V y el resultado es el mostrado en la fig. 8.

En la fig. 9 se observa la simulación de la modulación ASK de 4 niveles con un contador de 2 bits como habilitador para el *Mux*.

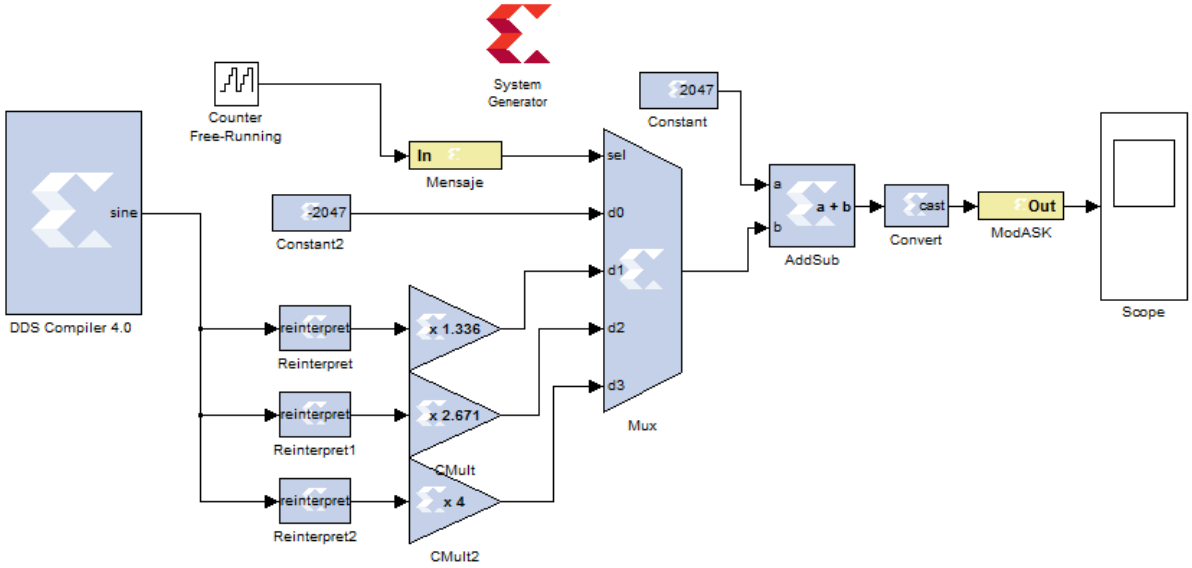

Fig. 6. Conexión para modulación ASK de 4 niveles.

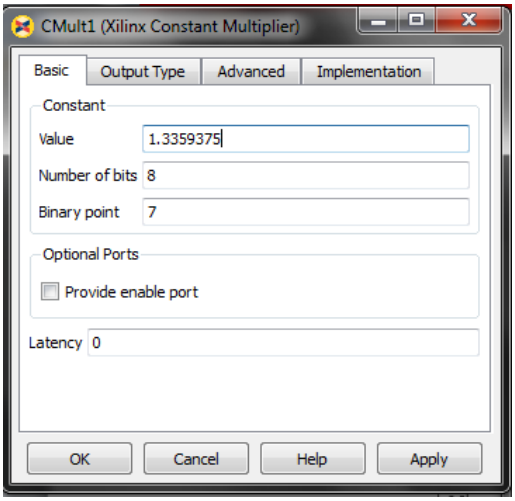

Fig. 7. Bloque CMult para la amplitud pico a pico de 1.1V.

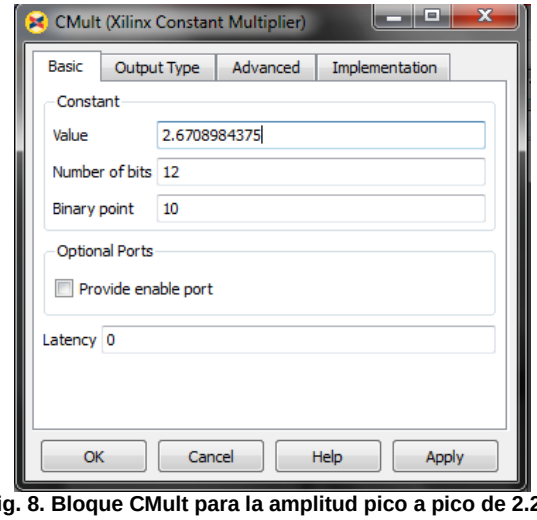

Fig. 8. Bloque CMult para la amplitud pico a pico de 2.2V.

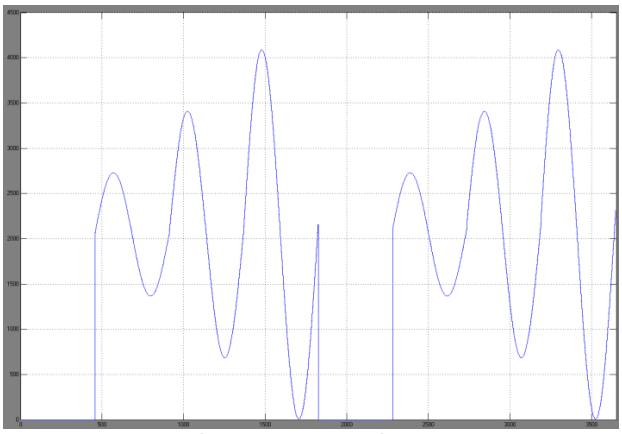

Fig. 9. Simulación de la modulación ASK de 4 niveles.

Los resultados de la implementación de la modulación ASK de 4 niveles de amplitud se observan en la fig. 10, fig. 11, fig. 12 y fig. 13 donde efectivamente los cambios de voltaje pico a pico son de 0, 1.1, 2.2 y 3.3 Volts, tomando una salida para la señal modulada y una entrada de 2 bits controlada por dos *switchs*.

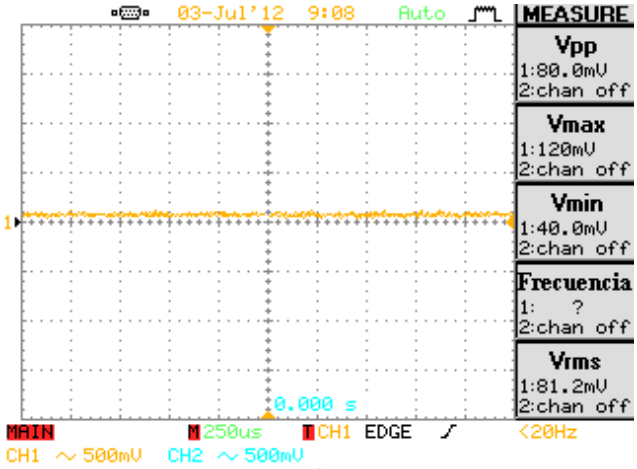

Fig. 10. Implementación ASK (nivel de 0 Vpp).

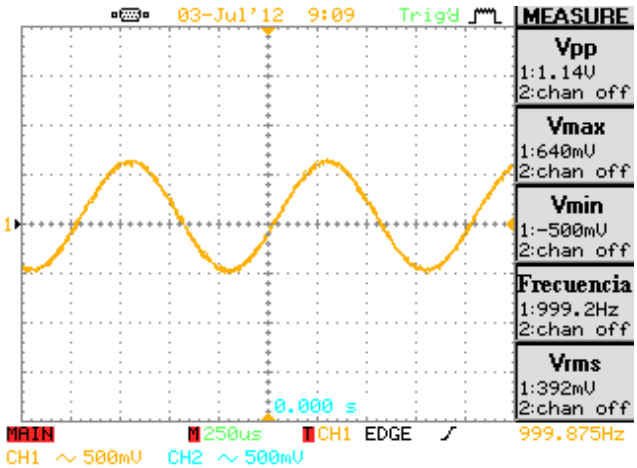

Fig. 11. Implementación ASK (nivel de 1.1 Vpp).

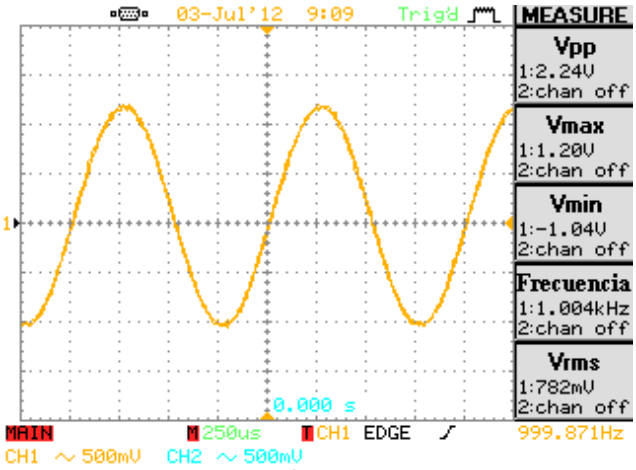

Fig. 12. Implementación ASK de (nivel de 2.2 Vpp).

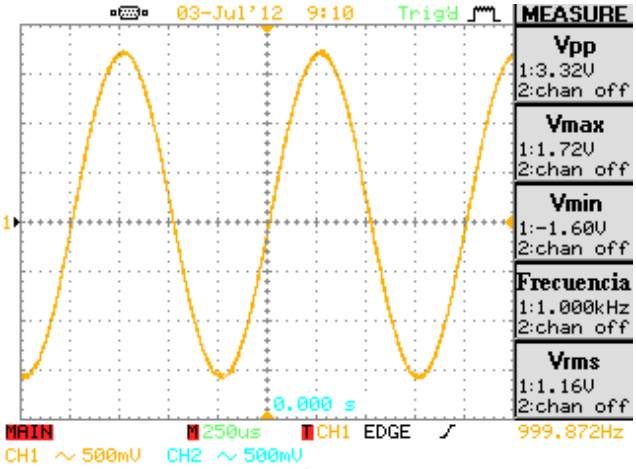

Fig.13. Implementación ASK (nivel de 3.3 Vpp).

LABORATORIO No 3 DE COMUNICACIONES DIGITALES

Modulación por Desplazamiento de Frecuencia FSK

Utilizando Xilinx System Generator e implementando en FPGA Spartan 3AN

OBJETIVOS

- Reconocer parámetros configurables para el correcto procesamiento de las señales digitales.
- Verificar el comportamiento en el tiempo de la señal de salida correspondiente a la modulación por desplazamiento de frecuencia de la señal de referencia.

BIBLIOGRAFÍA

- *Anexo A del proyecto de grado "Diseño de prácticas para laboratorio de Comunicaciones Digitales basado en Simulink y Xilinx System Generator"*
- *Spartan-3A/3AN FPGA Starter Kit Board User Guide (Datasheet)*
- *LogiCORE IP DDS Compiler v4.0 (Datasheet)*
- *Sistemas de Comunicaciones Electrónicas,* ***Wayne Tomasi****, 4 edición, capitulo 12 Comunicaciones Digitales.*

ACTIVIDAD DIRIGIDA

Simulación e implementación modulación 2FSK

Realizar el montaje de la fig. 1 que describe una modulación FSK de dos frecuencias diferentes, con los siguientes bloques de *Xilinx* en *Simulink*:

- Xilinx Blockset /Basic Elements/System Generator.
- Xilinx Blockset /Basic Elements/Constant.
- Xilinx Blockset / Basic Elements / Gateway Out.
- Xilinx Blockset / Basic Elements / Gateway In.
- Xilinx Blockset / DSP / DDS compiler V4.
- Simulink / Sources / Bernoulli Binary Generator.
- Xilinx Blockset / Basic Elements / Reinterpret.
- Xilinx Blockset / Basic Elements / Convert.
- Xilinx Blockset / Math / CMult.
- Xilinx Blockset / Math / Addsub.

Se configura cada bloque *DDS Compiler* con una frecuencia de 5 kHz y 15 kHz respectivamente, con 32 bits para el ancho de fase, un incremento de ancho de fase de 1342 ('010100111110') y 12 bits para la amplitud de la señal de salida (3.3 Vpp luego de pasar por el bloque *Reinterpret*), de tal forma se obtiene una modulación FSK con portadora en 10 kHz y ancho de banda de 10 kHz.

La fig. 2 muestra la simulación del sistema con una señal binaria (Bernoulli Binary Generator) que habilita el *Mux*. El bloque *Gateway In* debe estar configurado para recibir la misma cantidad de bits que vienen del bloque *Bernoulli Binary Generator*.

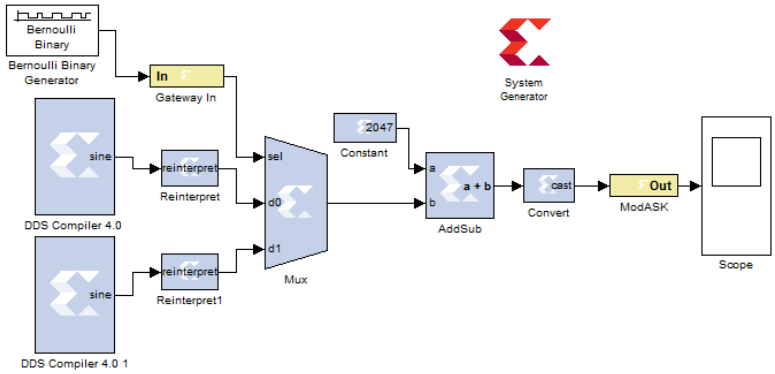

Fig. 1. Conexión para generar la modulación FSK de dos frecuencias.

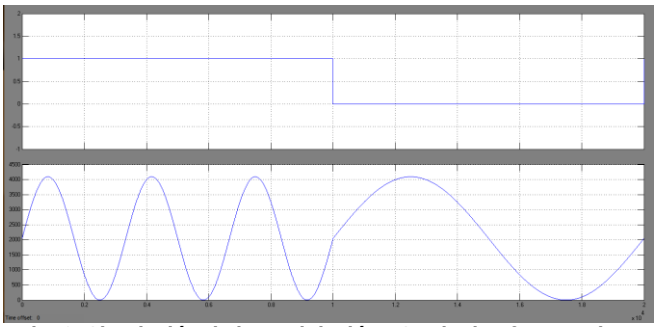

Fig. 2. Simulación de la modulación FSK de dos frecuencias.

Se genera el código para incluirlo como *Component* en un nuevo proyecto enlazado con el DAC.

Los resultados de la implementación para la modulación FSK de dos frecuencias se observan en la fig. 3 y fig. 4, estableciendo una salida para la señal modulada y una entrada de un bit controlada por un *switch*.

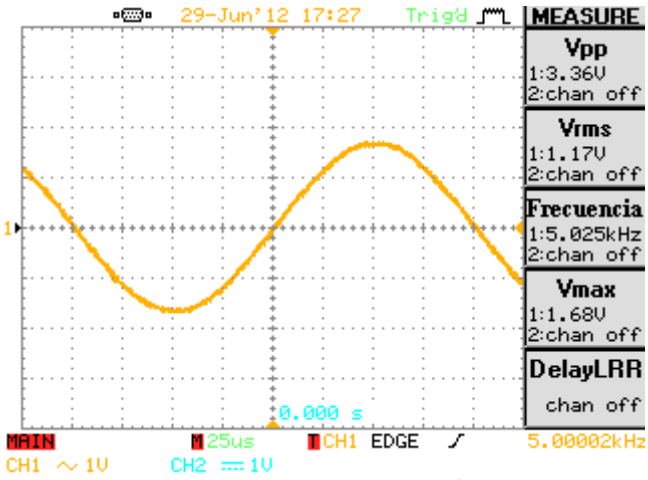

Fig. 3. Modo *Off* para la modulación OOK.

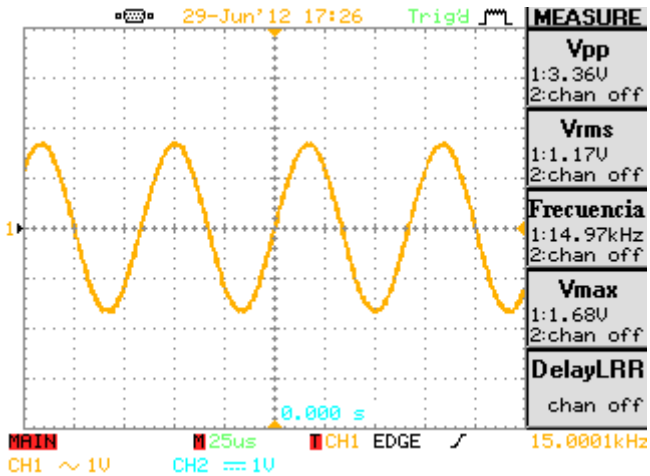

Fig. 4. Señal modulada a 3.3 Volts pico a pico.

ACTIVIDAD ADICIONAL

Simule e implemente una modulación FSK de 4 frecuencias con portadora en la frecuencia mostrada en la tabla 1 dependiendo del número asignado para su grupo, y se establece una distancia de 2 kHz entre cada nivel de frecuencia en la modulación (es decir una ancho de banda de 8 kHz). El voltaje pico a pico será el mostrado en la tabla 1.

Grupo	Vpp [Volts]	Frecuencia [Hz]
1	3.3	5000
2	3.0	5500
3	2.7	6000
4	2.4	6500
5	2.1	7000
6	1.8	5000
7	1.5	5500
8	1.2	6000
9	0.9	6500
10	0.6	7000

Tabla 1. Vpp y frecuencia de portadora para cada grupo.

ANÁLISIS DE RESULTADOS

En grupos de tres personas, se entregará un informe despues de realizada la práctica; éste debe contener:

- Título de la práctica.
- Nombre y código de los estudiantes.
- Objetivos, tanto los señalados en la guía, como los planteados por los estudiantes.
- Análisis y descripción detallada del procedimiento para realizar la actividad adicional.
- Conclusiones
- Observaciones y sugerencias (si las hay) para el mejoramiento de la metodología empleada.

DESARROLLO ACTIVIDAD ADICIONAL

Se desarrollará la actividad para una frecuencia de portadora de 5 kHz y un voltaje pico a pico de 3.3 Volts; entonces vamos a obtener una modulación de la señal con frecuencias de 1, 3, 7 y 9 kHz para las bandas laterales.

Éstas frecuencias son configuradas en cada bloque *DDS Compiler* con 12 bits de amplitud para la salida del Bloque (3.3 Vpp después de pasar por el bloque *Reinterpret*) y 32 bits para el incremento en el ancho de fase. Así pues se realiza la conexión de la fig. 5.

En la fig. 6 se observa la simulación del sistema donde se visualiza los cambios de las 4 frecuencias con un contador de 2 bits como habilitador del *Mux*.

Los resultados de la implementación de la modulación FSK de 4 frecuencias se observan en la fig. 7, fig. 8, fig. 9 y fig. 10 donde efectivamente los cambios de frecuencia son de 1, 3, 7 y 9 kHz, estableciendo una salida para la señal modulada y una entrada de dos bits controlada por dos *switchs*.

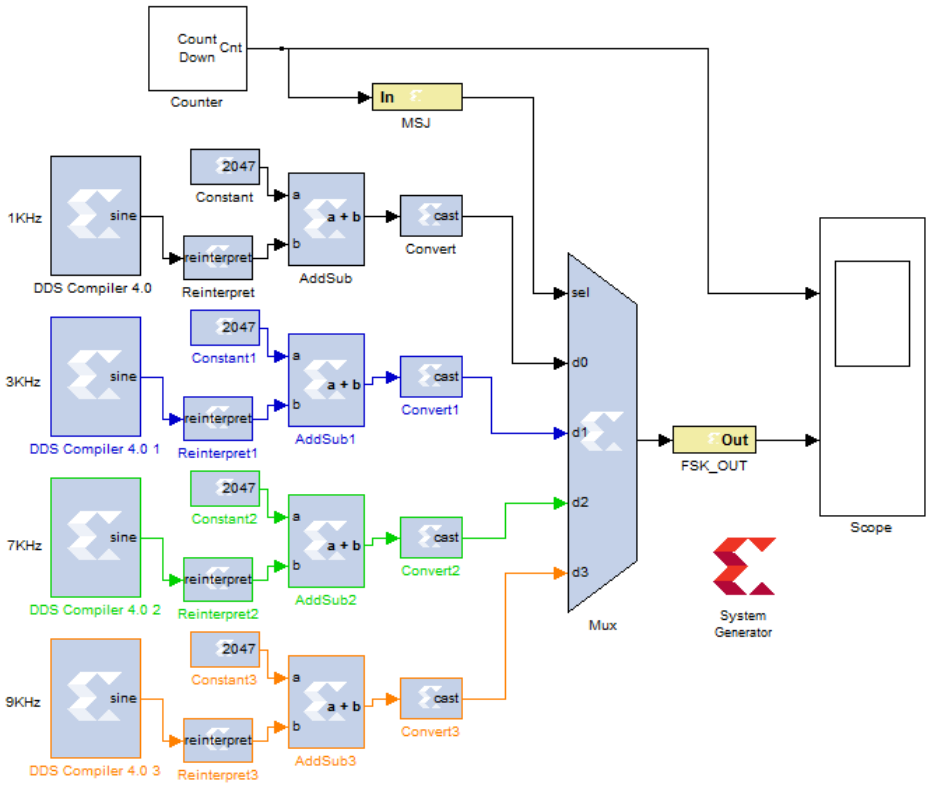

Fig. 5. Conexión para modulación FSK de 4 frecuencias.

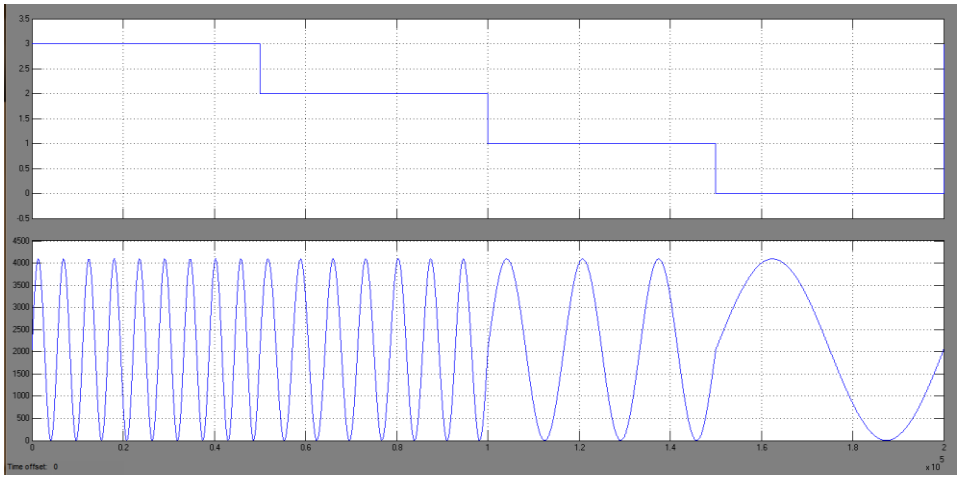

Fig. 6. Bloque CMult para la amplitud pico a pico de 1.1V.

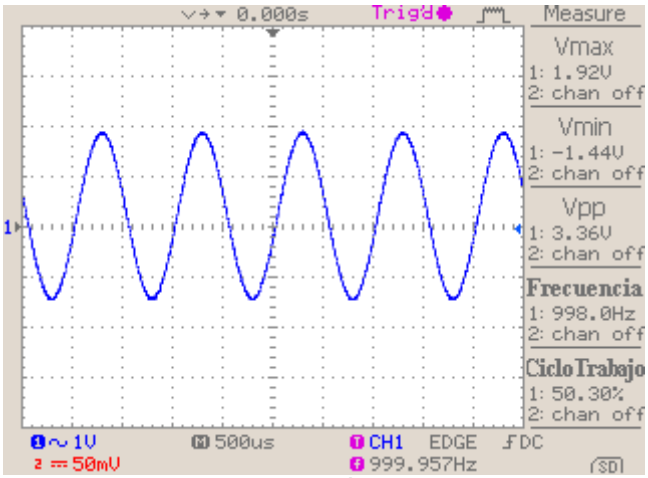

Fig. 7. Implementación 4FSK (1 kHz).

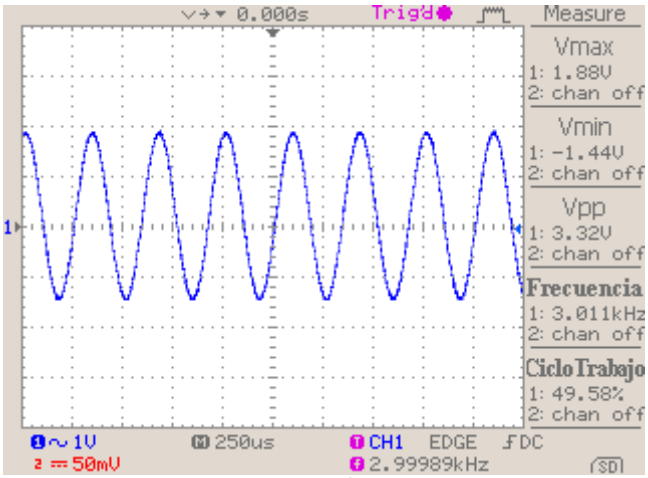

Fig. 8. Implementación 4FSK (3 kHz).

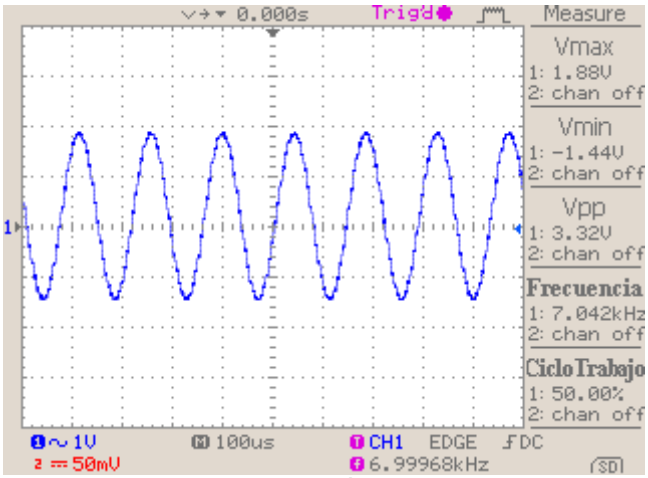

Fig. 9. Implementación 4FSK (7 kHz).

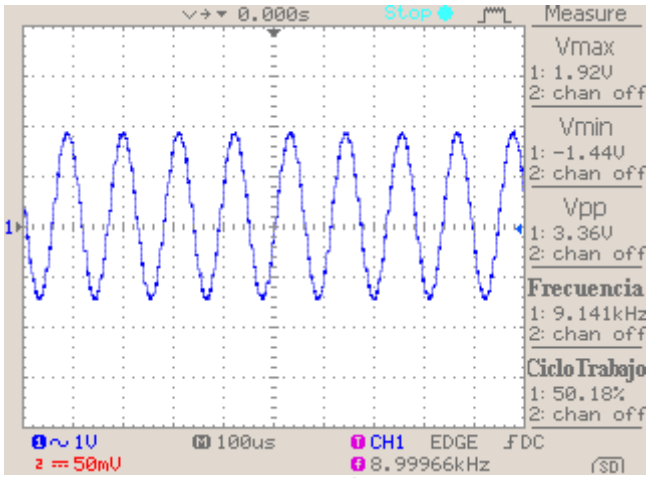

Fig.10. Implementación 4FSK (9 kHz).

LABORATORIO No 4 DE COMUNICACIONES DIGITALES

Modulación por Desplazamiento de Fase PSK

Utilizando Xilinx System Generator e implementando en FPGA Spartan 3AN

OBJETIVOS

- Reconocer parámetros configurables para el correcto procesamiento de las señales digitales.
- Verificar el comportamiento en el tiempo de la señal de salida correspondiente a la modulación por desplazamiento de fase de la señal de referencia
- Medir y comparar respecto a los valores teóricos, los cambios de fase de la señal modulada PSK.

BIBLIOGRAFÍA

- *Anexo A del proyecto de grado "Diseño de prácticas para laboratorio de Comunicaciones Digitales basado en Simulink y Xilinx System Generator"*
- *Spartan-3A/3AN FPGA Starter Kit Board User Guide (Datasheet)*
- *LogiCORE IP DDS Compiler v4.0 (Datasheet)*
- *Sistemas de Comunicaciones Electrónicas,* ***Wayne Tomasi****, 4 edición, capitulo 12 Comunicaciones Digitales.*

ACTIVIDAD DIRIGIDA

Simulación e implementación modulación 8PSK

Teóricamente la modulación 8PSK consiste en modificar el ángulo de una señal sinusoidal sin alterar su amplitud, esto se logra mediante la suma de una señal seno y otra coseno, donde la magnitud de cada componente determina el ángulo en el que se va a encontrar la señal resultante, y como su amplitud siempre deberá ser la misma en la salida, siempre habrá una compensación entre las componentes. Esto se explica de mejor forma al observar la fig. 1 y la tabla 1 donde encontramos el diagrama fasorial dependiendo de la entrada de 3 bits (8 cambios de fase de 45°) y los diferentes ángulos que puede tomar la señal al ser modulada.

Entrada Binaria			Fase de Salida 8PSK
Q	I	C	
0	0	0	-112.5°
0	0	1	-157.5°
0	1	0	-67.5°
0	1	1	-22.5°
1	0	0	112.5°
1	0	1	157.5°
1	1	0	67.5°
1	1	1	22.5°

Tabla 1. Fase de la señal modulada 8PSK.

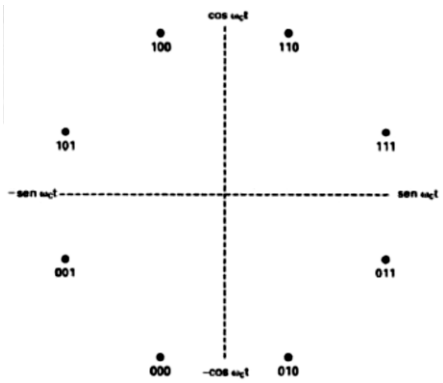

Fig. 1. Diagrama fasorial para la modulación 8PSK.
Tomado de: Sistema de comunicaciones Electrónicas (Wayne Tomasi).

Se realiza el montaje de la fig. 2 que describe una modulación 8PSK, con los siguientes bloques de *Xilinx* en *Simulink*:

- Xilinx Blockset /Basic Elements/System Generator.
- Xilinx Blockset /Basic Elements/Constant.
- Xilinx Blockset /Basic Elements/Mux.
- Xilinx Blockset / Basic Elements / Gateway Out.
- Xilinx Blockset / Basic Elements / Gateway In.
- Xilinx Blockset / DSP / DDS compiler V4.
- Simulink / Sources / Counter Free-Running.
- Xilinx Blockset / Basic Elements / Convert.
- Xilinx Blockset / Math / CMult.
- Xilinx Blockset / Math / Addsub.

El bloque *DDS Compiler* es configurado con una frecuencia de 1 kHz, con 26 bits en el incremento de ancho de fase y 11 bits para la amplitud de la señal de salida, para obtener una onda seno y otra coseno de amplitud igual a la unidad, que luego serán multiplicadas por factores dependientes de la amplitud que se requiera para la señal de salida, en este caso vamos a tomar una amplitud de 2048 valores digitales (3.3 Vpp), es decir las ondas de amplitud igual a la unidad se multiplicaran por factores precisos de tal manera que al sumarse las ondas seno y coseno la amplitud sea 2048 (3.3 Vpp) y la fase la requerida según el mensaje, que corresponde a la entrada en los *Mux* (Datos de entrada Q-I-C, que será un contador de 0 a 7 para efectos de simulación).

En la tabla 2 se muestra el valor real e imaginario de la señal modulada dependiendo de la entrada de 3 bits. Estos son los bloques *Constant* que se observan en la fig. 2. Por ejemplo las componentes para la entrada '000' son: Real = 2048*Sen(247.5) e Imaginario = 2048*Cos(247.5).

Entrada Binaria			Teoría		Seno	Coseno
			Mag.	Fase	Real	Imaginario
0	0	0	2048	247,5	-783,74	-1892,11
0	0	1	2048	202,5	-1892,11	-783,74
0	1	0	2048	292,5	783,74	-1892,11
0	1	1	2048	337,5	1892,11	-783,74
1	0	0	2048	112,5	-783,74	1892,11
1	0	1	2048	157,5	-1892,11	783,74
1	1	0	2048	67,5	783,74	1892,11
1	1	1	2048	22,5	1892,11	783,74

Tabla 2. Coeficientes para componentes Seno y Coseno.

En la figura 3 se visualiza la simulación de la modulación 8PSK con un contador de 0 a 7 como entrada para los *Mux*. Allí se puede observar la señal modulada, el comportamiento del contador, y las señales multinivel correspondiente a los coeficientes de la parte real e imaginaria respectivamente.

Se realiza la implementación en la tarjeta Spartan 3AN. Los resultados se muestran de la fig. 4 a la fig. 11, estableciendo 4 salidas (señal referencia marcada en azul, señal modulada marcada en amarillo, nivel de la parte real y nivel de la parte imaginaria de la señal modulada, estas últimas hacen referencia a la envolvente compleja) y una entrada de tres bits que será controlada por tres *switchs*.

De la fig. 12 a la fig. 19 se observa la medición de la diferencia temporal entre la señal de referencia y la modulada, siempre ubicando la primera en la parte derecha, pues todos los ángulos se tomaron positivos en nuestra tabla 2.

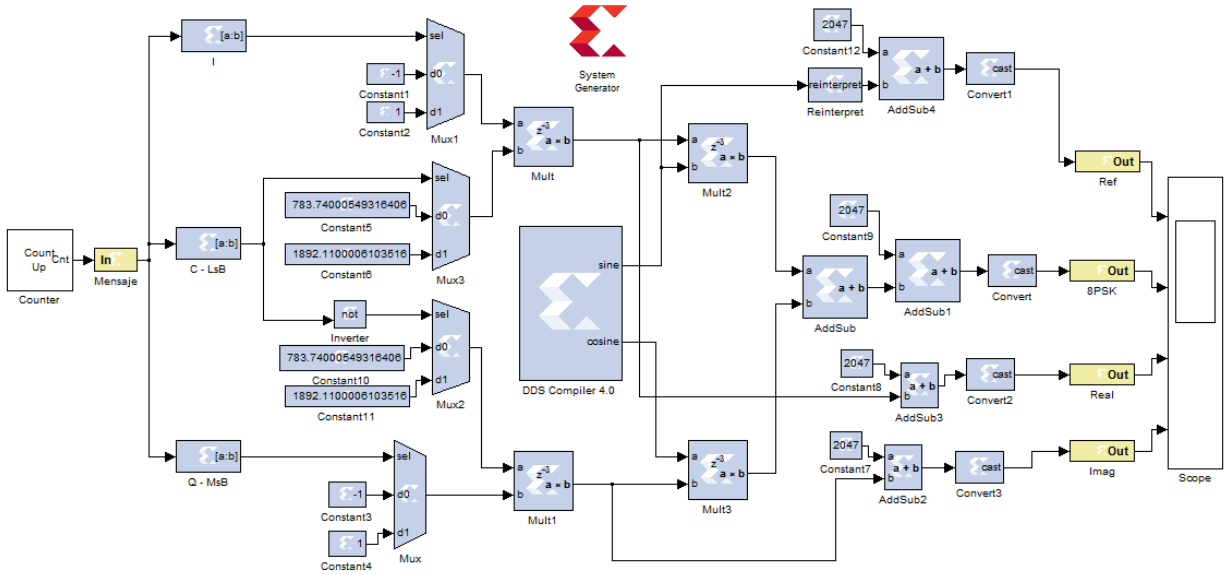

Fig. 2. Conexión para generar la modulación 8PSK.

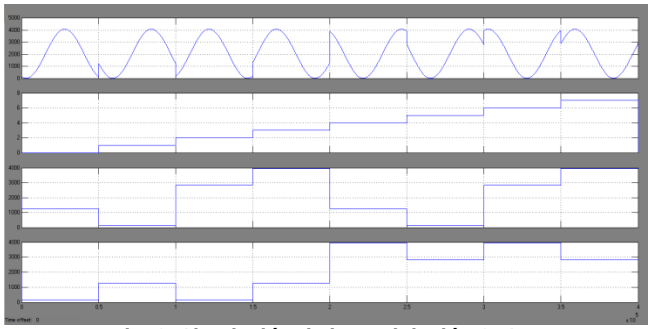

Fig. 3. Simulación de la modulación 8PSK.

Por ejemplo para la señal modulada con una entrada '000' la diferencia temporal Δt es de 690 micro segundos, entonces la diferencia angular $\Delta\varphi$ será igual a 690*(360/0.001) = 248.4°. Es decir:

$$\Delta\varphi = \frac{\Delta t * 360}{0.001} \qquad (2)$$

La fig. 20 muestra el diagrama de constelación para la modulación 8PSK, éste se obtuvo haciendo una superposición de todas las gráficas que arroja el modo XY en el osciloscopio usando como señales comparadas los niveles de coeficientes real e imaginario, dependientes de la entrada de tres bits.

La tabla 3 muestra cada ángulo de desfase experimental obtenido y el error relativo con respecto al valor teórico.

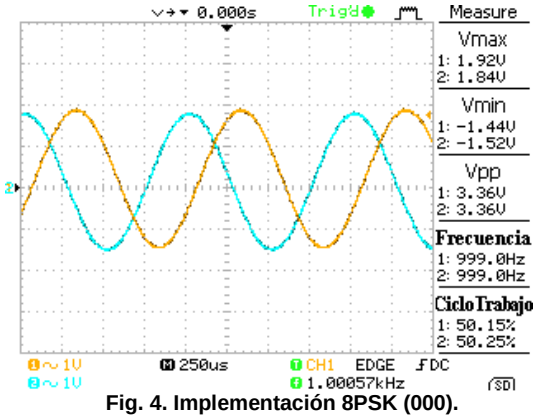

Fig. 4. Implementación 8PSK (000).

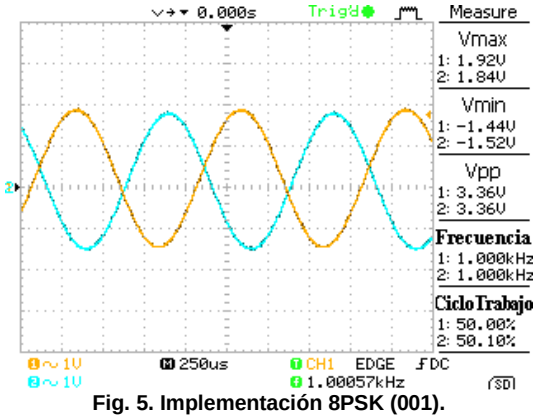

Fig. 5. Implementación 8PSK (001).

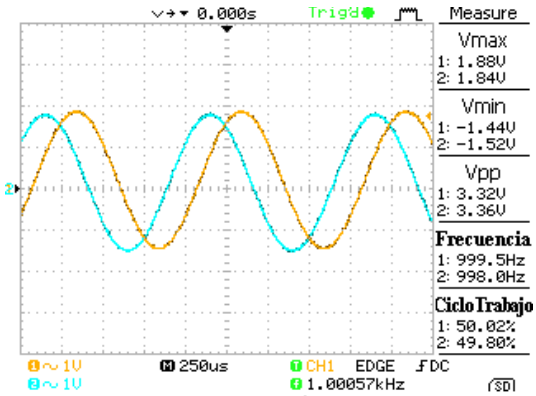

Fig. 6. Implementación 8PSK (010).

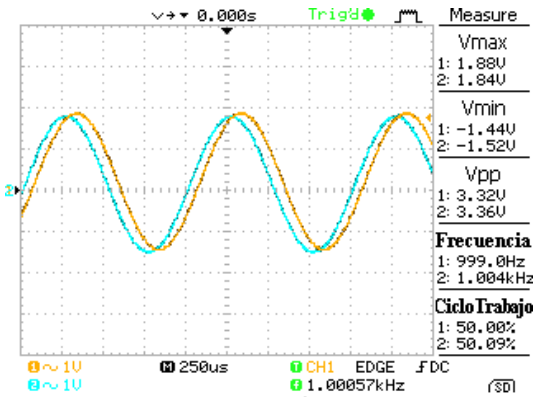

Fig. 7. Implementación 8PSK (011).

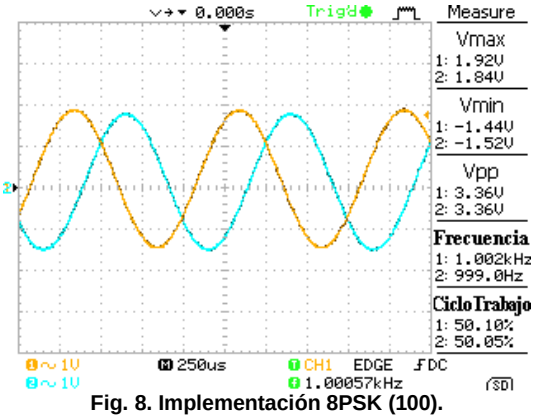

Fig. 8. Implementación 8PSK (100).

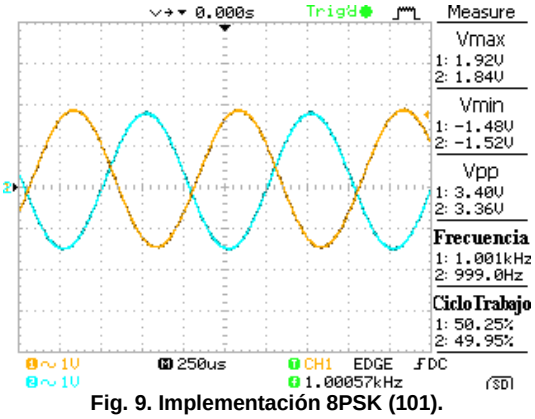

Fig. 9. Implementación 8PSK (101).

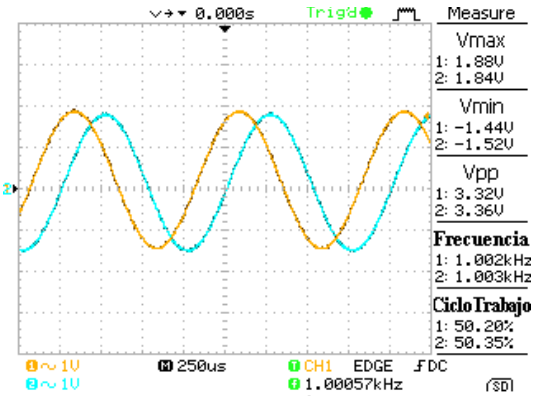

Fig. 10. Implementación 8PSK (110).

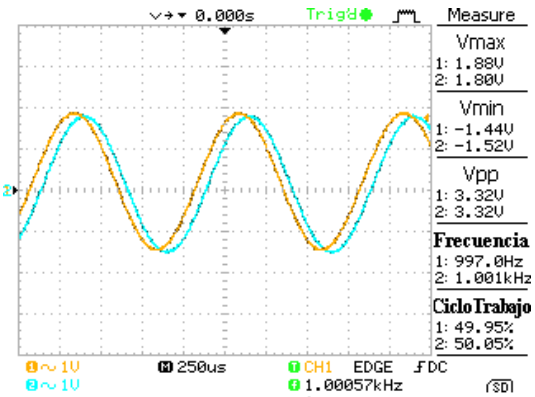

Fig. 11. Implementación 8PSK (111).

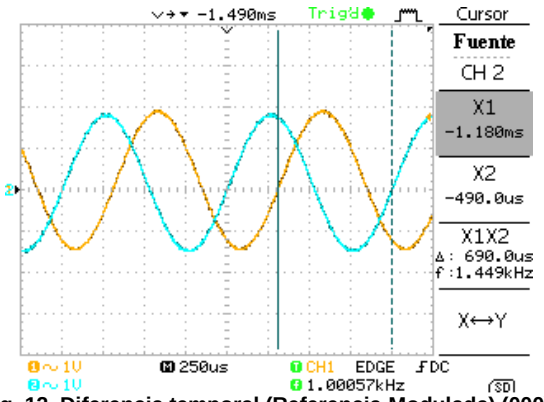

Fig. 12. Diferencia temporal (Referencia-Modulada) (000).

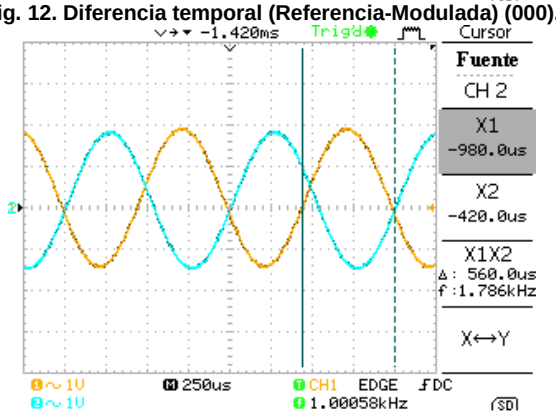

Fig. 13. Diferencia temporal (Referencia-Modulada) (001).

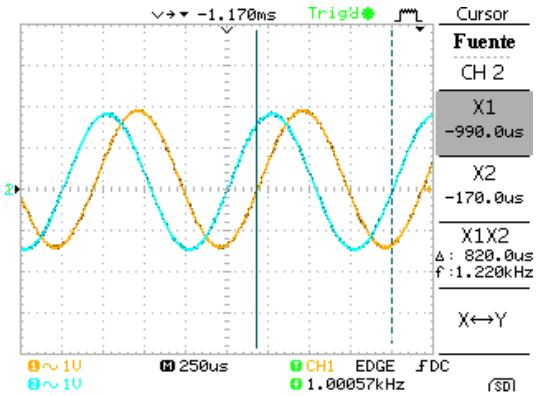

Fig. 14. Diferencia temporal (Referencia-Modulada) (010).

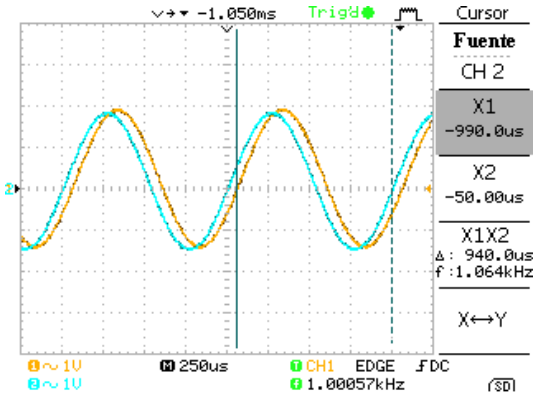

Fig. 15. Diferencia temporal (Referencia-Modulada) (011).

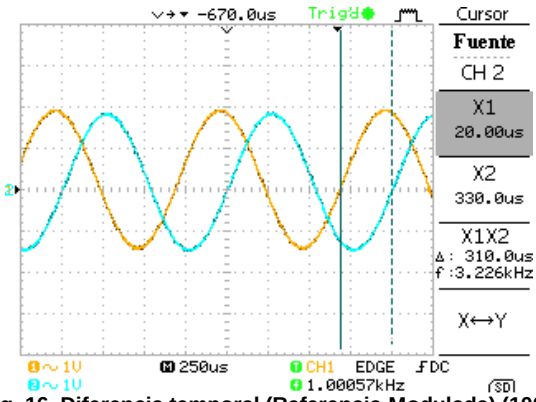

Fig. 16. Diferencia temporal (Referencia-Modulada) (100).

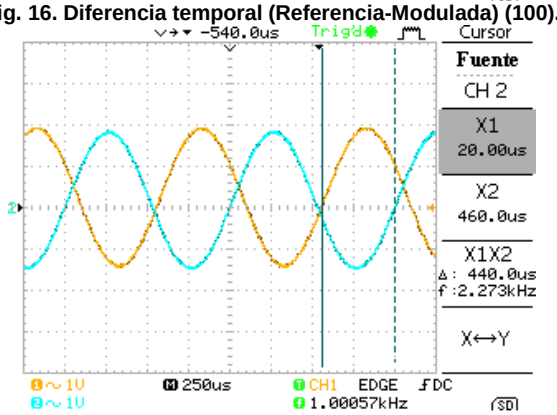

Fig. 17. Diferencia temporal (Referencia-Modulada) (101).

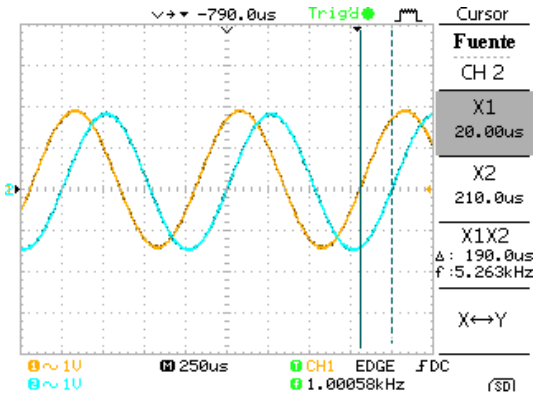

Fig. 18. Diferencia temporal (Referencia-Modulada) (110).

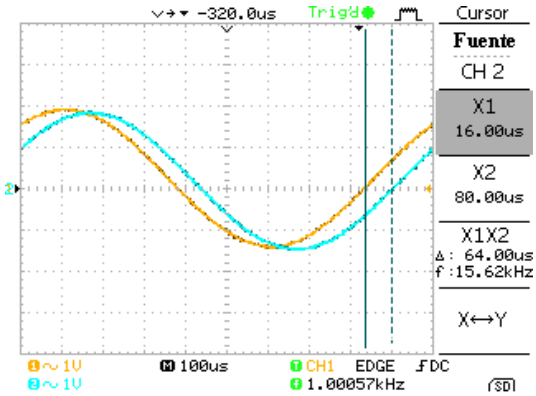

Fig. 19. Diferencia temporal (Referencia-Modulada) (111).

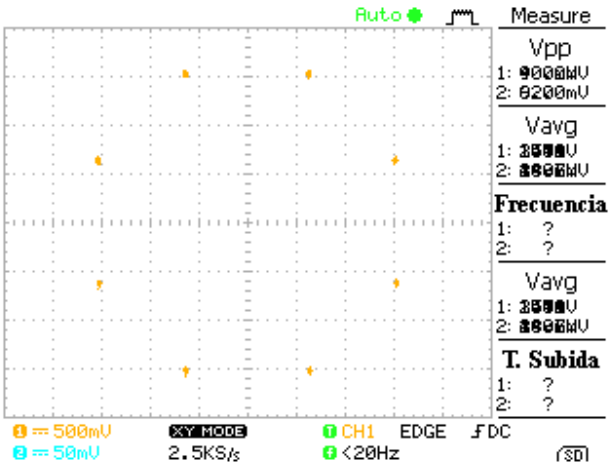

Fig. 20. Diagrama de constelación para la modulación 8PSK.

Entrada Binaria			Fase	Δt [seg]	Δφ [°]	Error relativo
0	0	0	247,5	6,90E-04	248,4	0,36%
0	0	1	202,5	5,60E-04	201,6	0,44%
0	1	0	292,5	8,20E-04	295,2	0,92%
0	1	1	337,5	9,40E-04	338,4	0,27%
1	0	0	112,5	3,10E-04	111,6	0,80%
1	0	1	157,5	4,40E-04	158,4	0,57%
1	1	0	67,5	1,90E-04	68,4	1,33%
1	1	1	22,5	6,40E-05	23,04	2,40%

Tabla 3. Ángulo de desfase experimental. Error relativo porcentual.

ACTIVIDAD ADICIONAL

Simule e implemente una modulación 16PSK para una señal referencia de frecuencia y voltaje pico a pico igual al mostrada en la tabla 4 dependiendo del número asignado para su grupo. Tenga en cuenta la información que revela la fig. 21 y la tabla 5, y recurra al texto guía para poder establecer de manera correcta las constantes que corresponden a la modulación 16PSK (coeficientes de las componentes seno y coseno)

Grupo	Vpp [Volts]	Frecuencia [Hz]
1	3.3	1000
2	3.0	1500
3	2.7	2000
4	2.4	2500
5	2.1	3000
6	1.8	3500
7	1.5	4000
8	1.2	4500
9	0.9	5000
10	0.6	5500

Tabla 4. Vpp asignado para cada grupo.

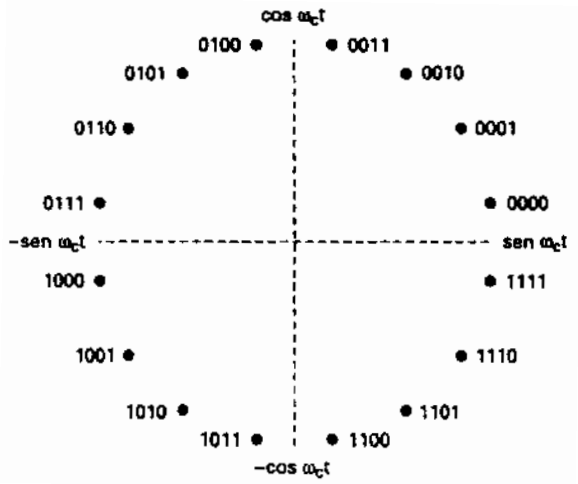

Fig. 21. Diagrama fasorial para la modulación 16PSK.
Tomado de: Sistema de comunicaciones Electrónicas (Wayne Tomasi).

Entrada Binaria				Fase de Salida 8PSK
0	**0**	**0**	**0**	11,25°
0	**0**	**0**	**1**	33,75°
0	**0**	**1**	**0**	56,25°
0	**0**	**1**	**1**	78,75°
0	**1**	**0**	**0**	101,25°
0	**1**	**0**	**1**	123,75°
0	**1**	**1**	**0**	146,25°
0	**1**	**1**	**1**	168,75°
1	**0**	**0**	**0**	191,25°
1	**0**	**0**	**1**	213,75°
1	**0**	**1**	**0**	236,25°
1	**0**	**1**	**1**	258,75°
1	**1**	**0**	**0**	281,25°
1	**1**	**0**	**1**	303,75°
1	**1**	**1**	**0**	326,25°
1	**1**	**1**	**1**	348,75°

Tabla 5. Fase de la señal modulada 16PSK.

ANÁLISIS DE RESULTADOS

En grupos de tres personas, se entregará un informe despues de realizada la práctica; éste debe contener:

- Título de la práctica.
- Nombre y código de los estudiantes.
- Objetivos, tanto los señalados en la guía, como los planteados por los estudiantes.
- Análisis y descripción detallada del procedimiento para realizar la actividad adicional.
- Conclusiones
- Observaciones y sugerencias (si las hay) para el mejoramiento de la metodología empleada.

DESARROLLO ACTIVIDAD ADICIONAL

Se desarrollará la actividad con una frecuencia para la señal de referencia de 1 kHz y amplitud en la salida de 3.3 Volts pico a pico.

En la tabla 6 se muestra el valor real e imaginario de la señal modulada dependiendo de la entrada de 4 bits. Estos son los bloques *Constant* que se observan en el diagrama del sistema completo de modulación 16PSK que se visualiza en la fig. 22.

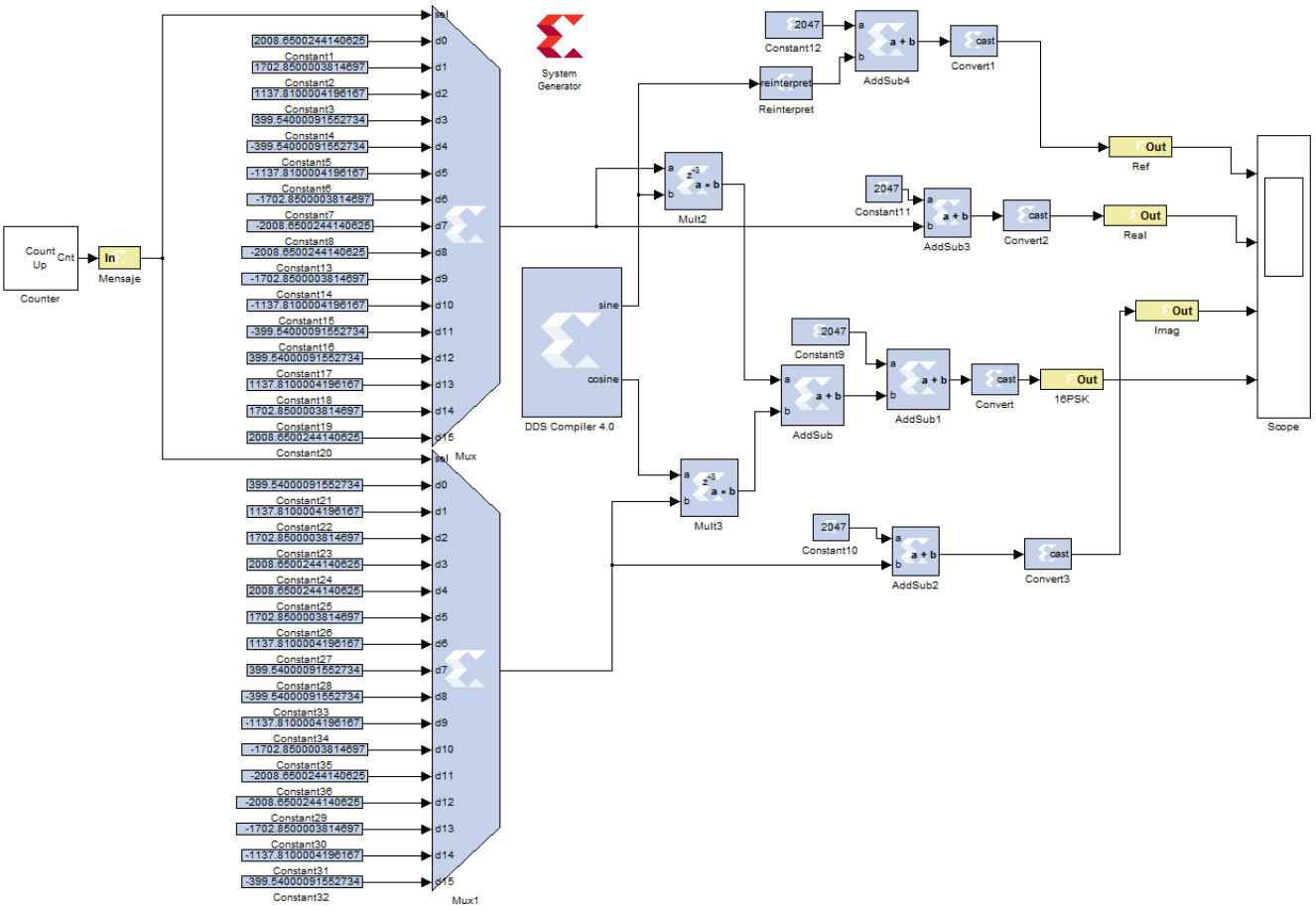

Fig. 22. Conexión para generar la modulación 16PSK.

Entrada Binaria				Teoría		Seno	Coseno
				Mag.	Fase	Real	Imaginario
0	0	0	0	2048	11,25	2008,65	399,54
0	0	0	1	2048	33,75	1702,85	1137,81
0	0	1	0	2048	56,25	1137,81	1702,85
0	0	1	1	2048	78,75	399,54	2008,65
0	1	0	0	2048	101,25	-399,54	2008,65
0	1	0	1	2048	123,75	-1137,81	1702,85
0	1	1	0	2048	146,25	-1702,85	1137,81
0	1	1	1	2048	168,75	-2008,65	399,54
1	0	0	0	2048	191,25	-2008,65	-399,54
1	0	0	1	2048	213,75	-1702,85	-1137,81
1	0	1	0	2048	236,25	-1137,81	-1702,85
1	0	1	1	2048	258,75	-399,54	-2008,65
1	1	0	0	2048	281,25	399,54	-2008,65
1	1	0	1	2048	303,75	1137,81	-1702,85
1	1	1	0	2048	326,25	1702,85	-1137,81
1	1	1	1	2048	348,75	2008,65	-399,54

Tabla 6. Coeficientes para componentes Seno y Coseno.

En la fig. 23 se observa la simulación de la modulación 16PSK con un contador de 0 a 15 como entrada (4 bits).
De la fig. 24 a la fig. 39 se observa la implementación de la modulación 16PSK dependiendo de la entrada de 4 bits controlada por cuatro *switchs* y estableciendo cuatro salidas (señal referencia, señal modulada, nivel de la parte real y nivel de la parte imaginaria para la señal modulada).

De la fig. 40 a la fig. 55 se visualiza la medición de la diferencia temporal entre la señal de referencia y la modulada, siempre ubicando la primera en la parte derecha, pues todos los ángulos se tomaron positivos en nuestra tabla 6.

En la fig. 56 se observa el diagrama de constelación para la modulación 16PSK

La tabla 7 muestra cada ángulo de desfase experimental obtenido y el error con respecto al valor teórico.

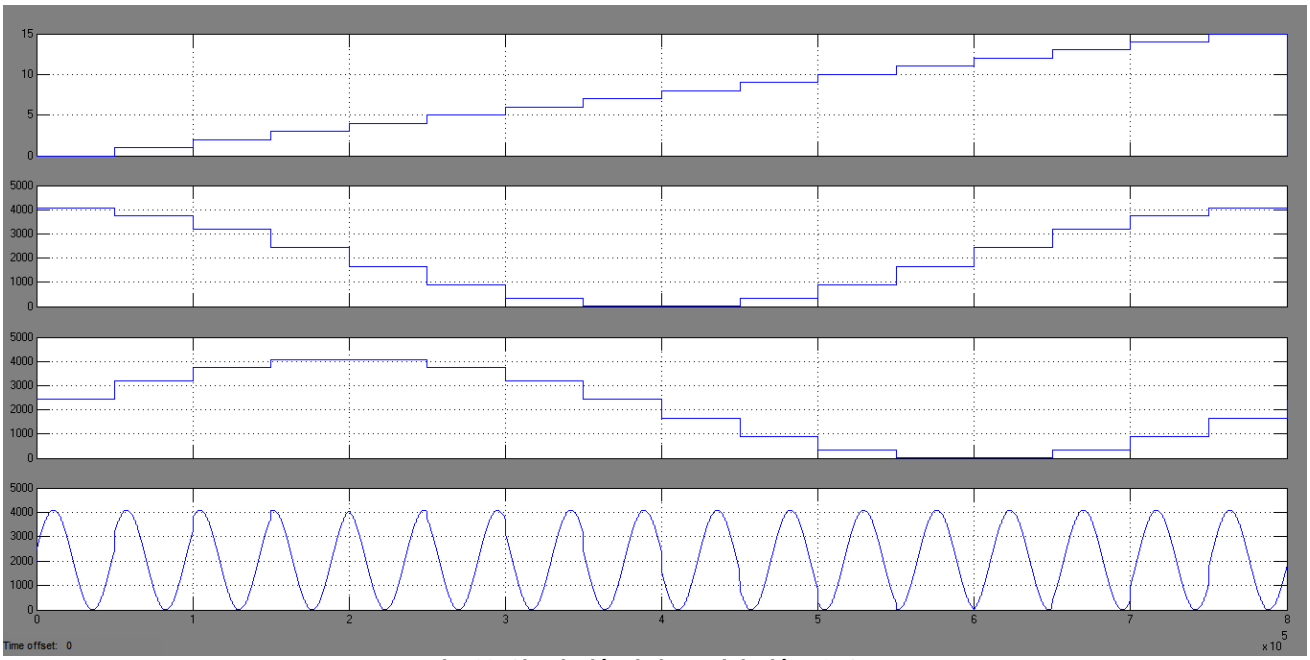

Fig. 23. Simulación de la modulación 16PSK.

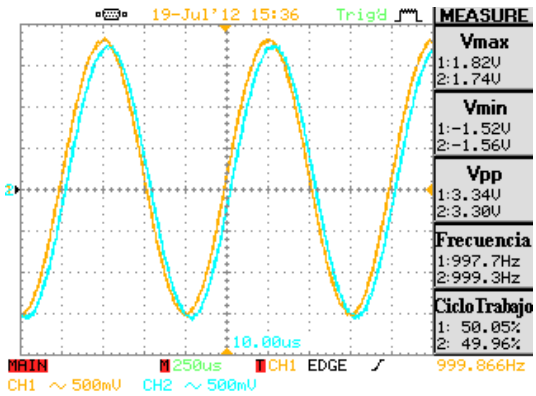

Fig. 24. Implementación 16PSK (0000).

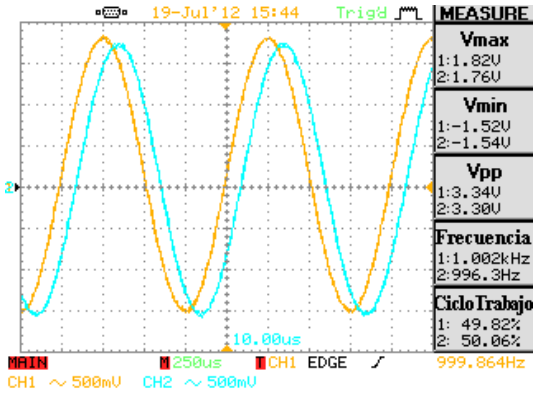

Fig. 25. Implementación 16PSK (0001).

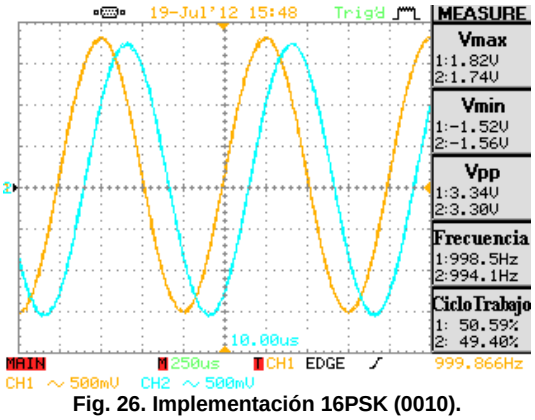

Fig. 26. Implementación 16PSK (0010).

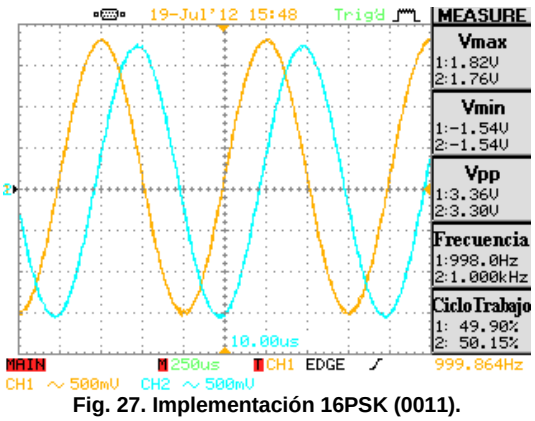

Fig. 27. Implementación 16PSK (0011).

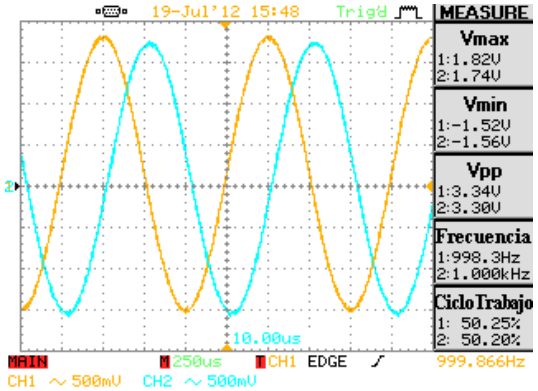

Fig. 28. Implementación 16PSK (0100).

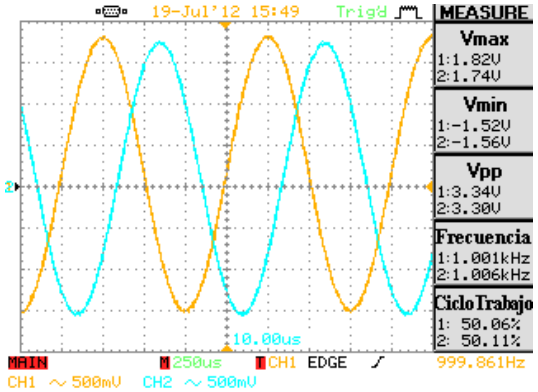

Fig. 29. Implementación 16PSK (0101).

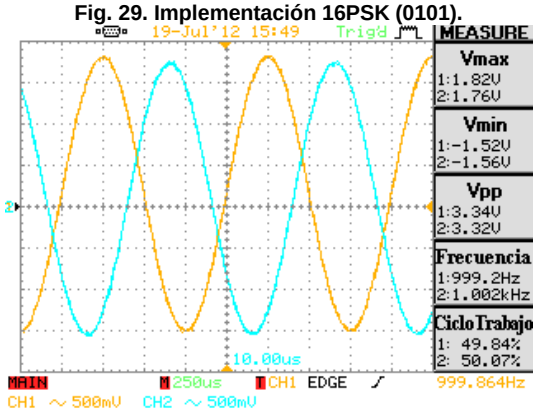

Fig. 30. Implementación 16PSK (0110).

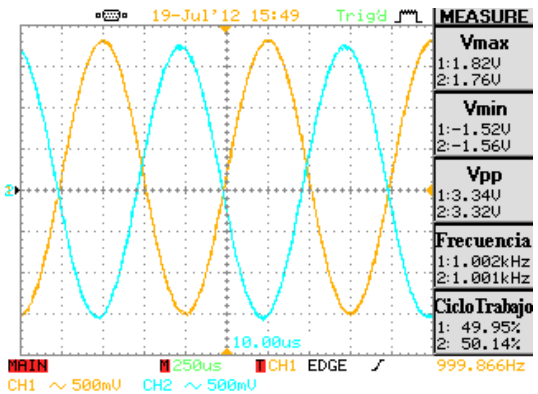

Fig. 31. Implementación 16PSK (0111).

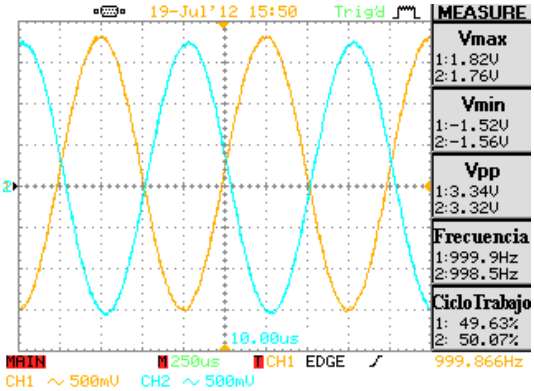

Fig. 32. Implementación 16PSK (1000).

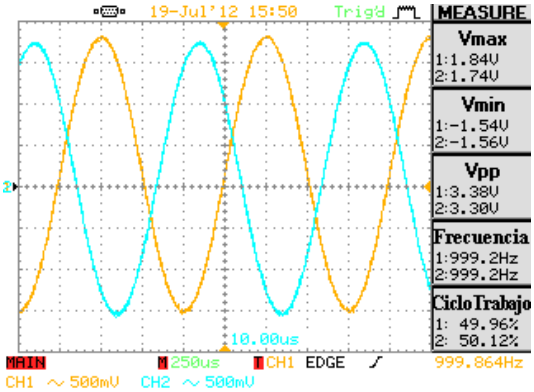

Fig. 33. Implementación 16PSK (1001).

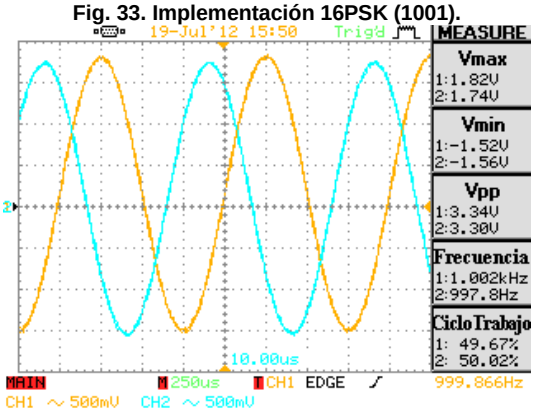

Fig. 34. Implementación 16PSK (1010).

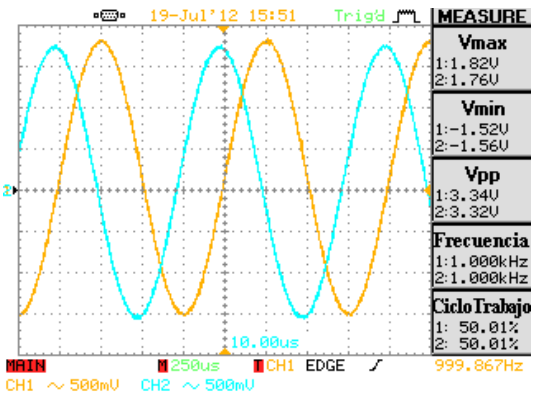

Fig. 35. Implementación 16PSK (1011).

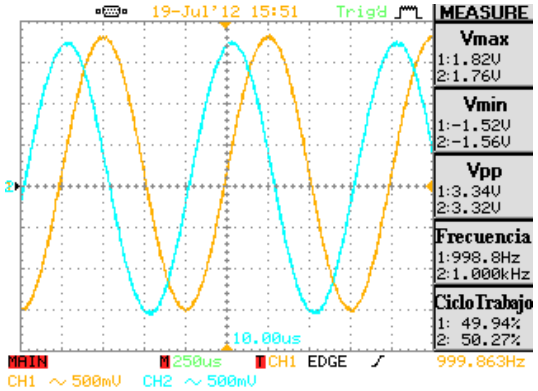

Fig. 36. Implementación 16PSK (1100).

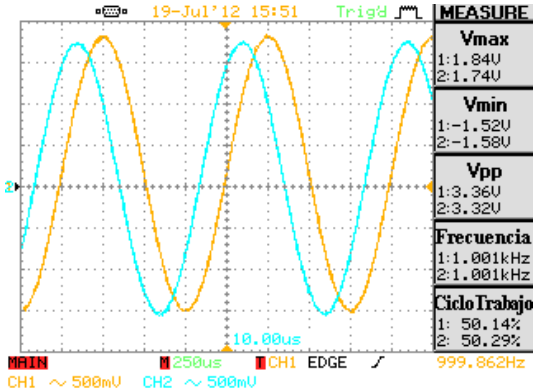

Fig. 37. Implementación 16PSK (1101).

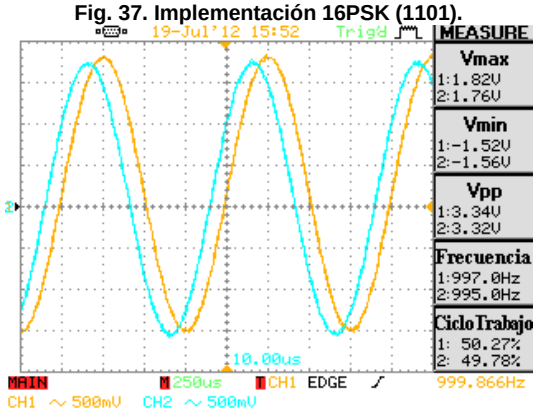

Fig. 38. Implementación 16PSK (1110).

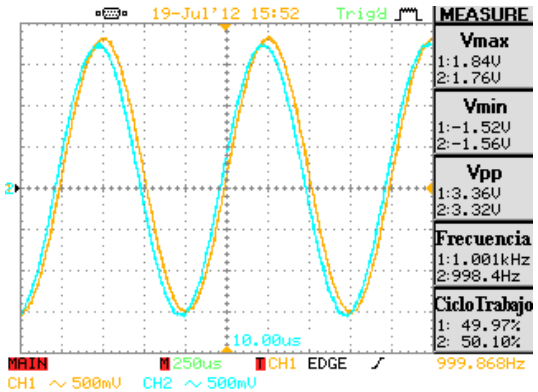

Fig. 39. Implementación 16PSK (1111).

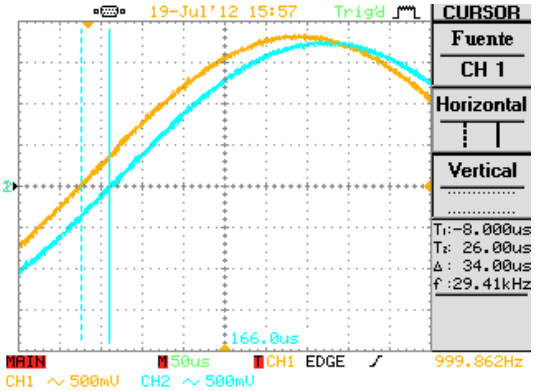

Fig. 40. Diferencia temporal (Referencia-Modulada) (0000).

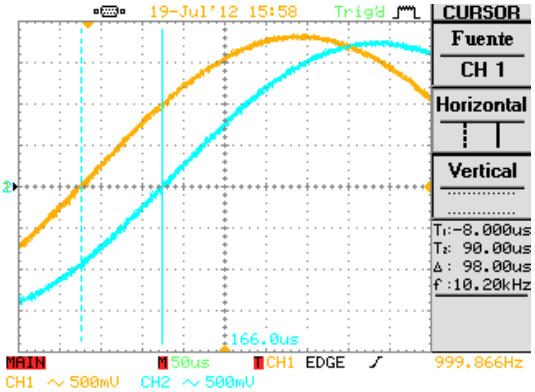

Fig. 41. Diferencia temporal (Referencia-Modulada) (0001).

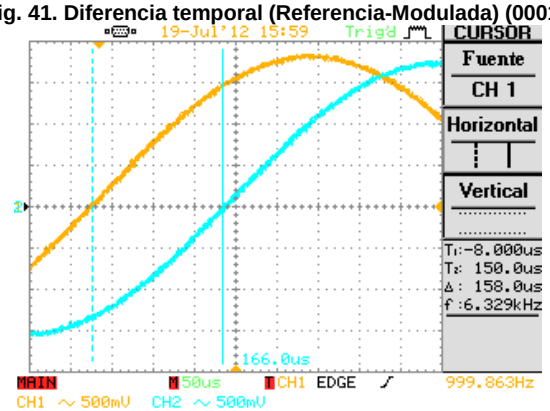

Fig. 42. Diferencia temporal (Referencia-Modulada) (0010).

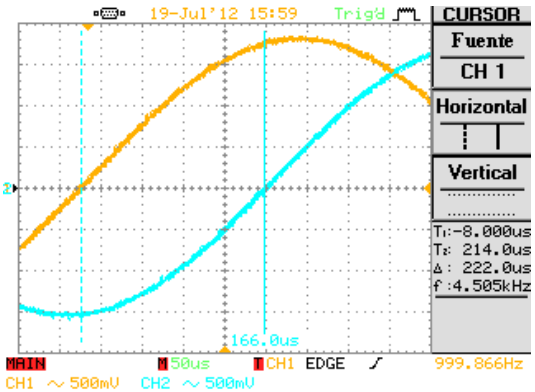

Fig. 43. Diferencia temporal (Referencia-Modulada) (0011).

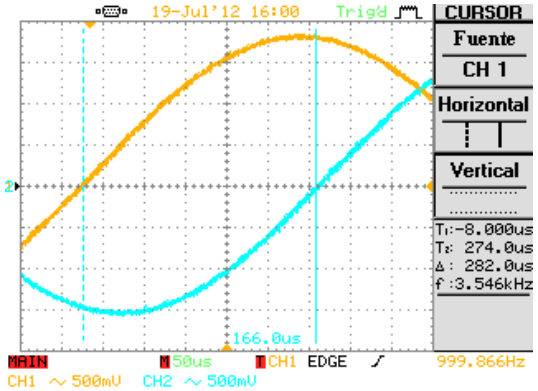

Fig. 44. Diferencia temporal (Referencia-Modulada) (0100).

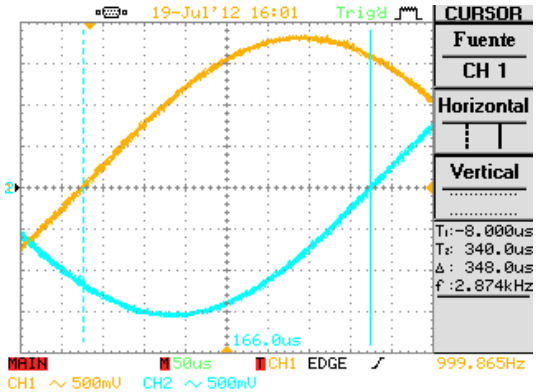

Fig. 45. Diferencia temporal (Referencia-Modulada) (0101).

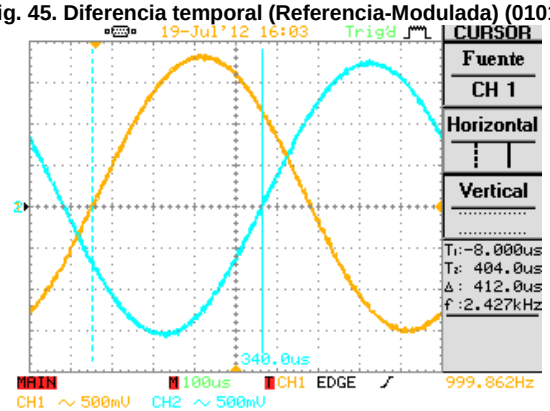

Fig. 46. Diferencia temporal (Referencia-Modulada) (0110).

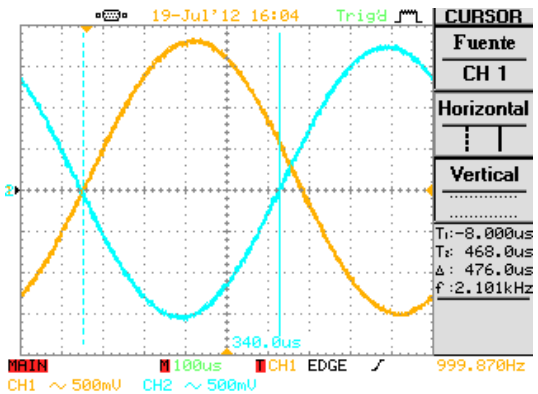

Fig. 47. Diferencia temporal (Referencia-Modulada) (0111).

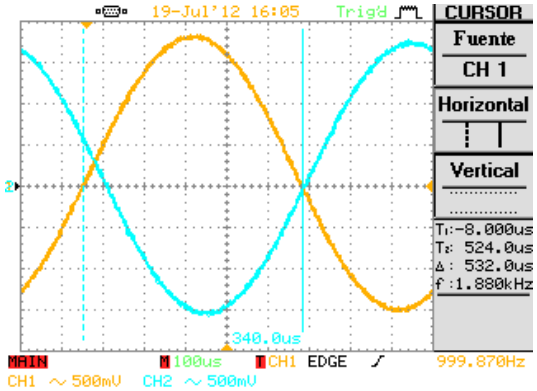

Fig. 48. Diferencia temporal (Referencia-Modulada) (1000).

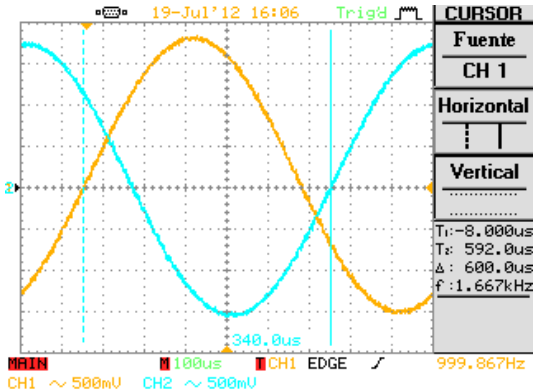

Fig. 49. Diferencia temporal (Referencia-Modulada) (1001).

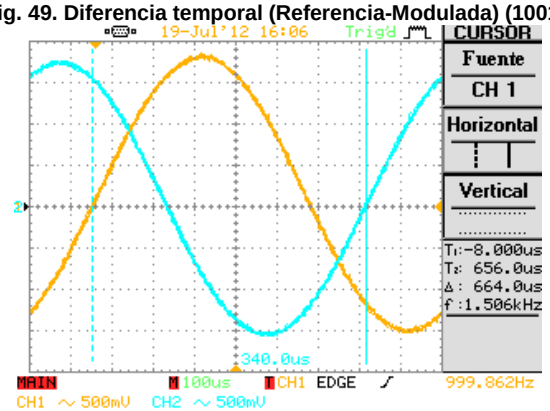

Fig. 50. Diferencia temporal (Referencia-Modulada) (1010).

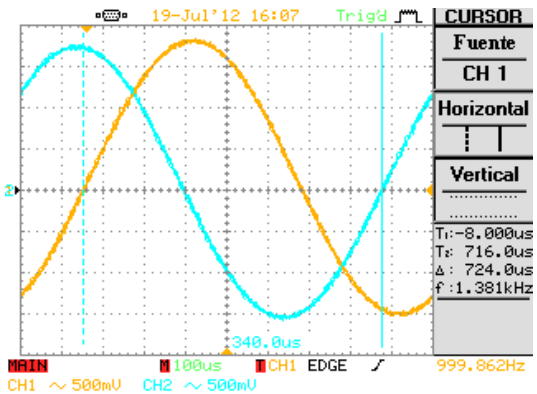

Fig. 51. Diferencia temporal (Referencia-Modulada) (1011).

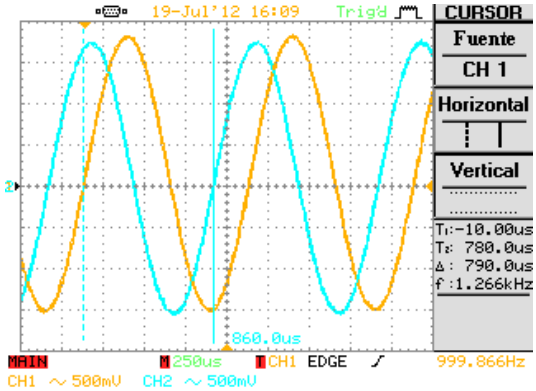

Fig. 52. Diferencia temporal (Referencia-Modulada) (1100).

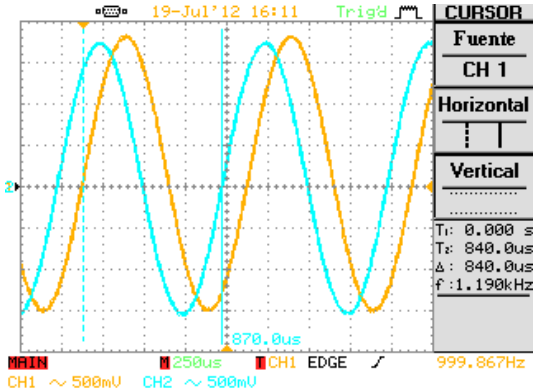

Fig. 53. Diferencia temporal (Referencia-Modulada) (1101).

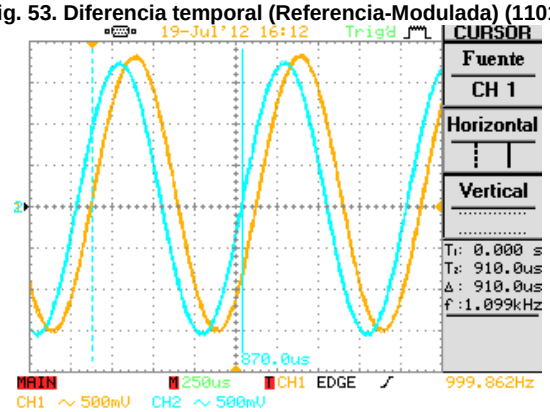

Fig. 54. Diferencia temporal (Referencia-Modulada) (1110).

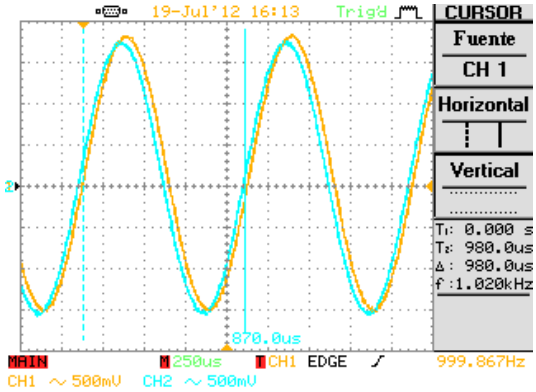

Fig. 55. Diferencia temporal (Referencia-Modulada) (1111).

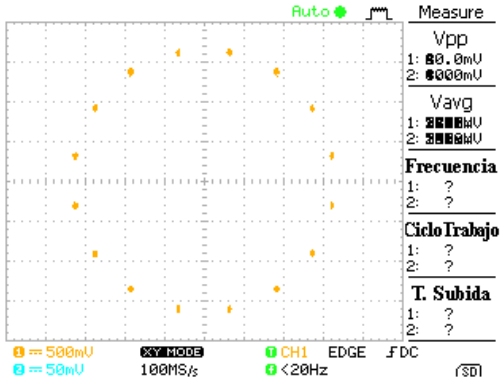

Fig. 56. Diagrama de constelación para la modulación 16PSK.

Entrada Binaria				Fase	Δt [seg]	Δθ [°]	Error relativo
0	0	0	0	11,25	3,40E-05	12,24	8,80%
0	0	0	1	33,75	9,80E-05	35,28	4,53%
0	0	1	0	56,25	1,58E-04	56,88	1,12%
0	0	1	1	78,75	2,22E-04	79,92	1,49%
0	1	0	0	101,25	2,82E-04	101,52	0,27%
0	1	0	1	123,75	3,48E-04	125,28	1,24%
0	1	1	0	146,25	4,12E-04	148,32	1,42%
0	1	1	1	168,75	4,76E-04	171,36	1,55%
1	0	0	0	191,25	5,32E-04	191,52	0,14%
1	0	0	1	213,75	6,00E-04	216	1,05%
1	0	1	0	236,25	6,64E-04	239,04	1,18%
1	0	1	1	258,75	7,24E-04	260,64	0,73%
1	1	0	0	281,25	7,90E-04	284,4	1,12%
1	1	0	1	303,75	8,40E-04	302,4	0,44%
1	1	1	0	326,25	9,10E-04	327,6	0,41%
1	1	1	1	348,75	9,70E-04	349,2	0,13%

Tabla 7. Ángulo de desfase experimental. Error relativo porcentual.

LABORATORIO No 5 DE COMUNICACIONES DIGITALES

Modulación de amplitud en cuadratura QAM

Utilizando Xilinx System Generator e implementando en FPGA Spartan 3AN

OBJETIVOS

- Reconocer parámetros configurables para el correcto procesamiento de las señales digitales.
- Verificar el comportamiento en el tiempo de la señal de salida correspondiente a la modulación de amplitud en cuadratura QAM de la señal de referencia.
- Medir y comparar respecto a los valores teóricos, los cambios de amplitud y fase de la señal modulada QAM.

BIBLIOGRAFÍA

- *Anexo A del proyecto de grado "Diseño de prácticas para laboratorio de Comunicaciones Digitales basado en Simulink y Xilinx System Generator"*
- *Spartan-3A/3AN FPGA Starter Kit Board User Guide (Datasheet)*
- *LogiCORE IP DDS Compiler v4.0 (Datasheet)*
- *Sistemas de Comunicaciones Electrónicas,* ***Wayne Tomasi****, 4 edición, capitulo 12 Comunicaciones Digitales.*

ACTIVIDAD DIRIGIDA

Simulación e implementación 8QAM

En la modulación de amplitud en cuadratura QAM, la información digital a transmitir va contenida tanto en la amplitud como en la fase de la señal portadora. Estas variaciones en amplitud y ángulo se logran mediante la suma de una señal seno y otra coseno de amplitudes adecuadamente seleccionadas.
Cuando la información a transmitir consta de 3 bits (Q, I y C), existen ocho posibles símbolos, es decir ocho combinaciones diferentes de amplitud y fase, las cuales se muestran de manera más clara en la tabla 1 y la figura 1. Debido a esto este tipo de modulación recibe el nombre de 8QAM.
Se debe tener en cuenta que el máximo valor posible a implementar según las características del DAC de la tarjeta Spartan 3AN, es de $3.3/2 = 1.65$ Vp, de la tabla 1 vemos que el valor de amplitud más grande es 1.848 Vp el cual está fuera de este rango, por tal motivo es necesario hacer un escalamiento en las amplitudes de la señal modulada 8QAM, se escogió arbitrariamente un factor de 2/3 el cual satisface esta condición y los resultados están consignados en la tercer columna de la tabla 1.

Entrada Binaria			Amplitud 8QAM	Ajuste de amplitud 8QAM	Fase 8QAM
Q	I	C			
0	0	0	0,765	0,510	225
0	0	1	1,848	1,232	225
0	1	0	0,765	0,510	315
0	1	1	1,848	1,232	315
1	0	0	0,765	0,510	135
1	0	1	1,848	1,232	135
1	1	0	0,765	0,510	45
1	1	1	1,848	1,232	45

Tabla 1. constantes de la señal modulada 8QAM.

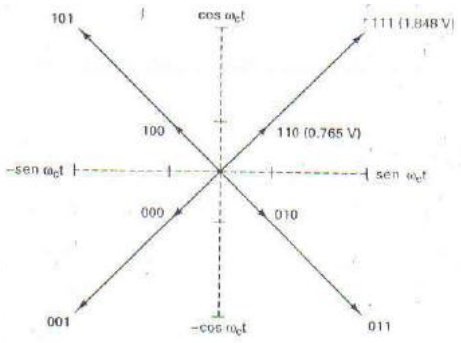

Fig. 1. Diagrama fasorial para la modulación 8QAM.
Tomado de: Sistema de comunicaciones Electrónicas (Wayne Tomasi).

Sabiendo esto, se realiza el montaje de la fig. 2 que describe una modulación 8QAM, con los siguientes bloques de *Xilinx* en *Simulink*:

- Xilinx Blockset /Basic Elements/System Generator.
- Xilinx Blockset /Basic Elements/Constant.
- Xilinx Blockset /Basic Elements/Mux.
- Xilinx Blockset / Basic Elements / Gateway Out.
- Xilinx Blockset / Basic Elements / Gateway In.
- Xilinx Blockset / DSP / DDS compiler V4.
- Xilinx Blockset / Basic Elements / Convert.
- Xilinx Blockset / Math / Mult.
- Xilinx Blockset / Math / Addsub.
- Xilinx Blockset / Basic Elements / slice.
- Xilinx Blockset / Basic Elements / Reinterpret.

La señal de entrada (Mensaje-*Gateway in*) de 3 bits es separada en los 3 canales independientes Q (MsB), I y C (LsB) mediante los bloques *Slice* que se configuran para extraer un determinado bit del vector, por ejemplo para obtener el bit central *I* los parámetros son -1 a partir del MSB y 1 apartir del LsB, tal como se muestra en la fig 3, los parámetros para los bits Q y C son 0 y 2, y, -2 y 0 respectivamente.

El bloque *DDS Compiler* es configurado con una frecuencia de 1 kHz, con 26 bits en el incremento de ancho de fase $B_{\theta_{(n)}}$ y 12 bits de salida, para obtener una onda seno y otra coseno de amplitud igual a la unidad, que luego serán multiplicadas por factores dependientes de la amplitud y fase que se requiera para la señal de salida. Por ejemplo para el caso en que la entrada sea el dato "000", de la tabla 1 se sabe que se desea una amplitud de 0.51Vp y una fase de 225°, haciendo la conversión a la resolución del DAC la amplitud 0.51Vp corresponde con 633 valores digitales, A partir del diagrama fasorial de la fig 1 se pueden determinar la magnitud de las ondas seno (parte real) y coseno (parte imaginaria) cuya suma generan la señal deseada como se muestra a continuación:

$$real = Amp\ Seno = 633 * \mathrm{COS}(225) = -447.6114$$
$$imaginaria = Amp\ Cos = 633 * SIN(225) = -447.6114$$

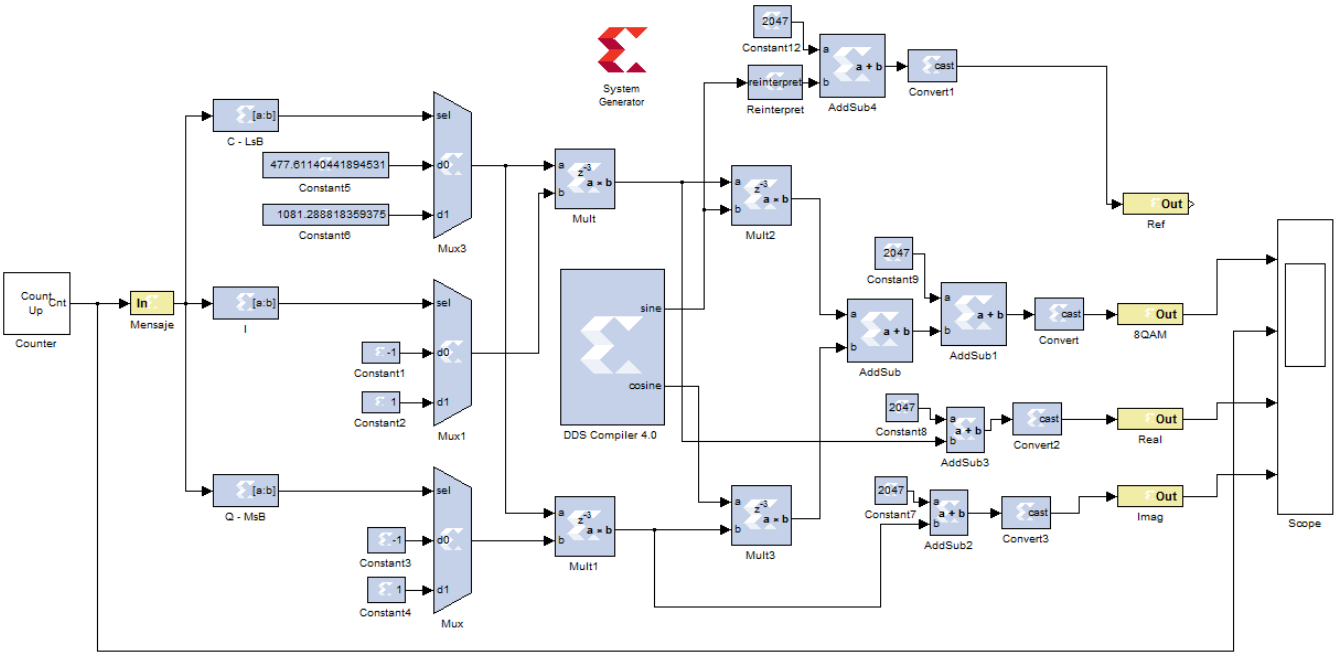

Fig. 2. Conexión para generar la modulación 8QAM.

En la tabla 2 se muestra el valor real e imaginario de la señal modulada dependiendo de la entrada de 3 bits. El valor absoluto de estos valores se ingresa como una constante en el diseño, las cuales son seleccionadas a través de un *Mux* controlado por el bit C de entrada. El signo de las constantes lo determinan los bits de entrada Q/I los cuales también controlan multiplexores con valores fijos de 1 y -1, posteriormente multiplicando mediante el bloque *Mult* (parámetros de configuración por defecto) la salida de los *Mux* controlados por C/I se obtiene la constante que multiplica a la onda seno, y de la misma manera con los *Mux* que corresponden a C/Q la constante de la onda coseno.

Finalmente la onda Modulada 8QAM se obtiene sumando las ondas resultantes de Multiplicar las señales de amplitud unitaria generadas por el bloque *DDS Compiler* con sus respectivas constantes.
Como salidas para la implementación del diseño se van a fijar la señal modulada 8QAM, sus componentes real e imaginaria, es decir el nivel de la parte real y nivel de la parte imaginaria de la señal modulada, que referencia a la envolvente compleja, y una señal sinusoidal tomada directamente del bloque DDS con desfase 0° con el fin de tener una referencia para realizar mediciones de desfase.

Entrada Binaria			Teoría		Seno	Coseno
			Mag.	Fase	Real	Imaginario
0	0	0	0,510	225	-447,6114	-447,6114
0	0	1	1,232	225	-1081,2888	-1081,2888
0	1	0	0,510	315	447,6114	-447,6114
0	1	1	1,232	315	1081,2888	-1081,2888
1	0	0	0,510	135	-447,6114	447,6114
1	0	1	1,232	135	-1081,2888	1081,2888
1	1	0	0,510	45	447,6114	447,6114
1	1	1	1,232	45	1081,2888	1081,2888

Tabla 2. Coeficientes para componentes Seno y Coseno.

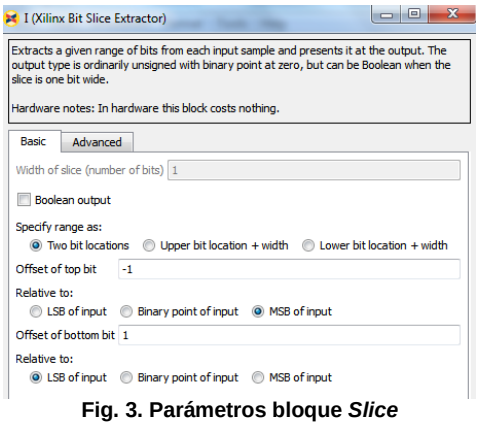

Fig. 3. Parámetros bloque *Slice*

En la figura 4 se visualiza la simulación para un tiempo de 400.000 segundos de la modulación 8QAM con un contador de 0 a 7 con pasos de 50.000 como mensaje de entrada, para así tener un periodo exacto de la señal modulada por cada uno de los valores de entrada. 50M/1K=50.000 muestras por periodo. Allí se puede observar el comportamiento del contador y la señal modulada respectivamente. En la figura 5 se muestra el resultado de una simulación teórica tomada del texto Wayne Tomasi.

Se realiza la implementación en la tarjeta Spartan 3AN ejecutando la debida instanciación de las fuentes del DAC y *System Generator,* y asignando cada uno de los bits de la señal de entrada mensaje a un *switch* y las cuatro salidas antes mencionadas a cada uno de los canales del DAC. Los resultados de la implementación para cada una de los posibles símbolos de entrada (“000” a “111”) se muestran de la fig. 6 a la fig. 13, en donde la señal en color amarillo corresponde a la sinusoidal de referencia, y la de color azul a la señal modulada 8QAM.

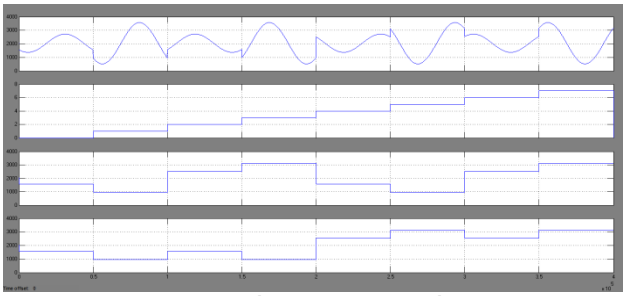

Fig. 4. Simulación de la modulación 8QAM.

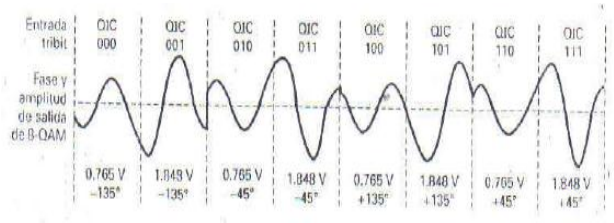

Fig 5. Simulación teórica 8QAM

Tomado de: Sistema de comunicaciones Electrónicas (Wayne Tomasi).

De la fig. 14 a la fig. 21 se observa la medición de la diferencia temporal entre la señal de referencia y la modulada, siempre ubicando la primera en la parte derecha, pues todos los ángulos se tomaron positivos en la tabla 2. A partir de esta medición se puede determinar el desfase de la señal 8QAM, por ejemplo para la señal modulada con una entrada ‘000’ la diferencia temporal Δt es de 630 micro segundos, entonces la diferencia angular $\Delta\varphi$ será igual a 630*(360/0.001) = 226.8°. Es decir:

$$\Delta\varphi = \frac{\Delta t * 360}{T = periodo} \qquad (2)$$

La fig. 22 muestra el diagrama de constelación correspondiente a la señal 8QAM. La tabla 3 muestra cada ángulo de desfase experimental obtenido y el error relativo con respecto al valor teórico.

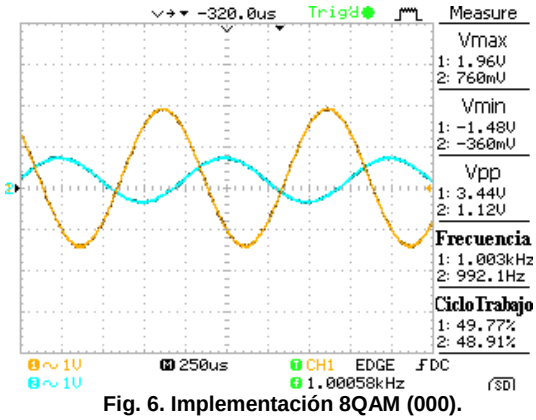

Fig. 6. Implementación 8QAM (000).

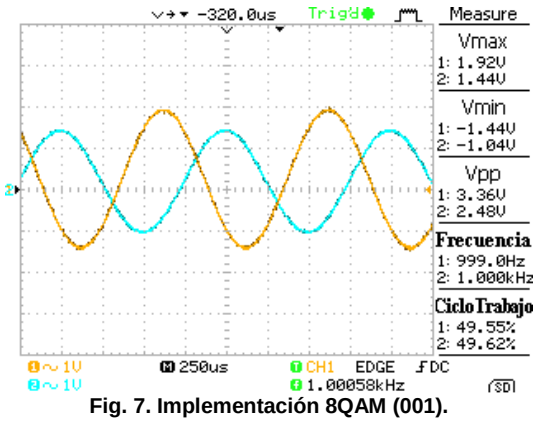

Fig. 7. Implementación 8QAM (001).

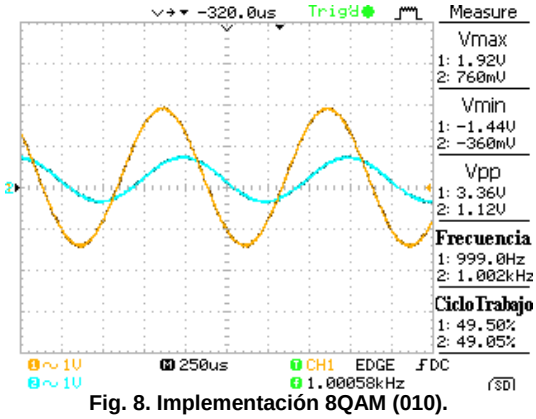

Fig. 8. Implementación 8QAM (010).

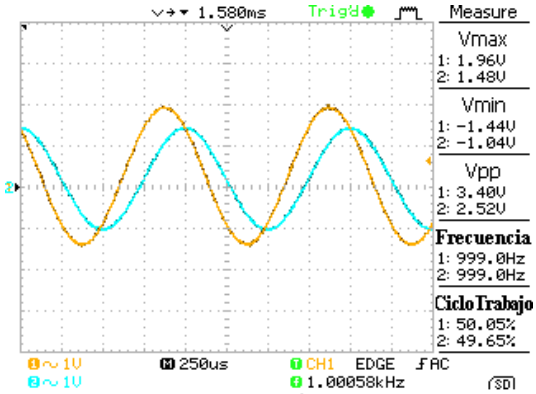

Fig. 9. Implementación 8QAM (011).

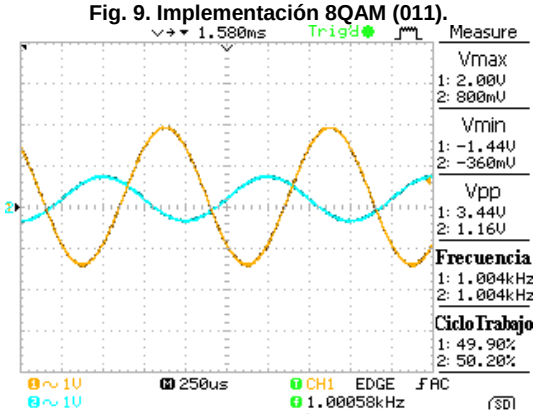

Fig. 10. Implementación 8QAM (100).

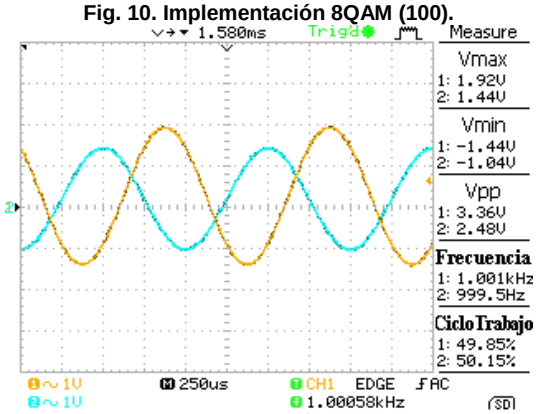

Fig. 11. Implementación 8QAM (101).

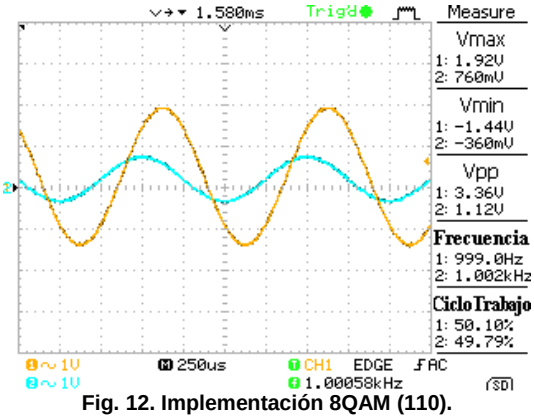

Fig. 12. Implementación 8QAM (110).

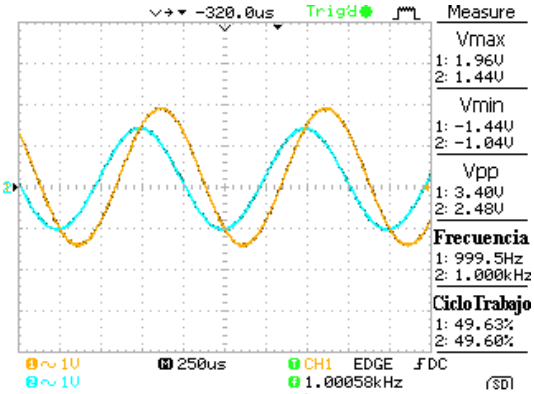

Fig. 13. Implementación 8QAM (111).

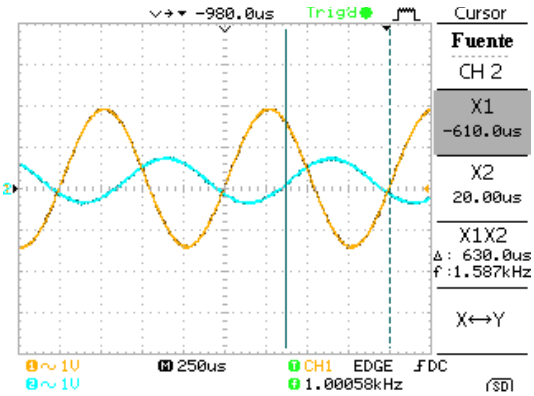

Fig. 14. Diferencia temporal (Referencia-Modulada) (000).

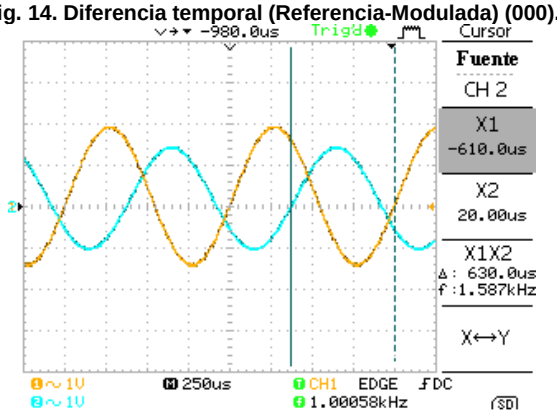

Fig. 15. Diferencia temporal (Referencia-Modulada) (001).

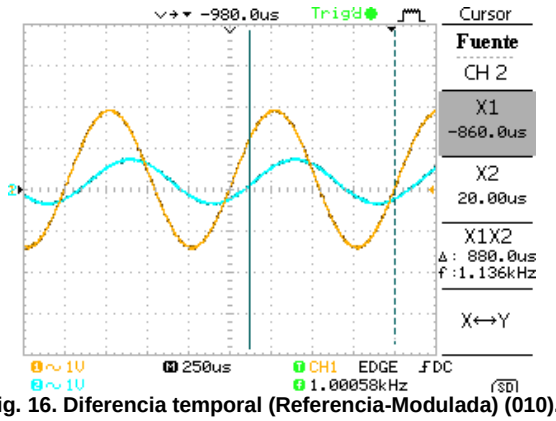

Fig. 16. Diferencia temporal (Referencia-Modulada) (010).

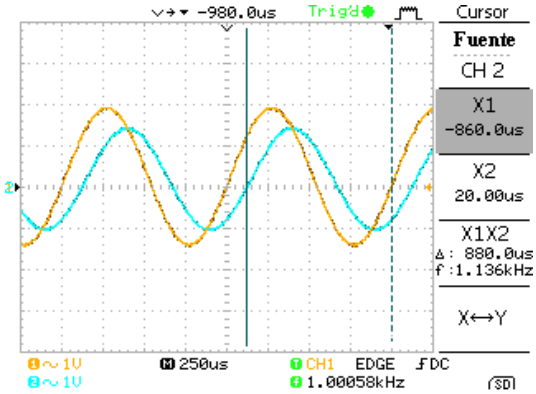

Fig. 17. Diferencia temporal (Referencia-Modulada) (011).

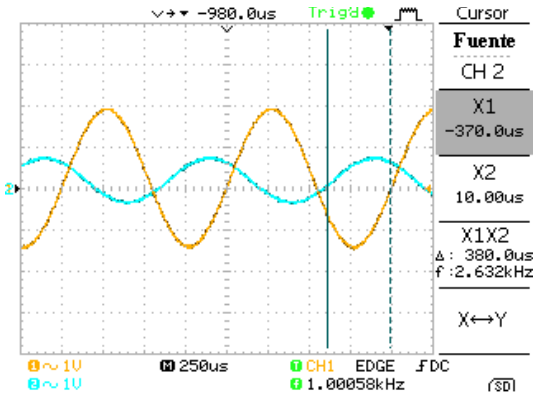

Fig. 18. Diferencia temporal (Referencia-Modulada) (100).

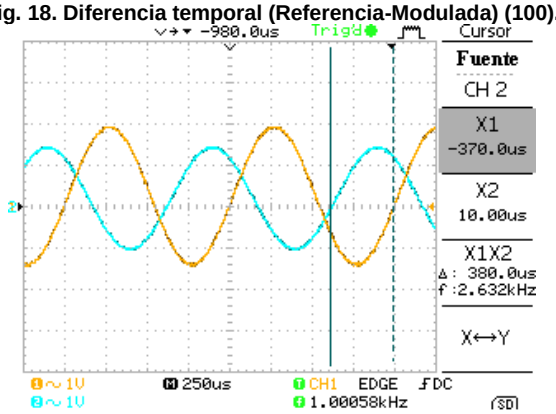

Fig. 19. Diferencia temporal (Referencia-Modulada) (101).

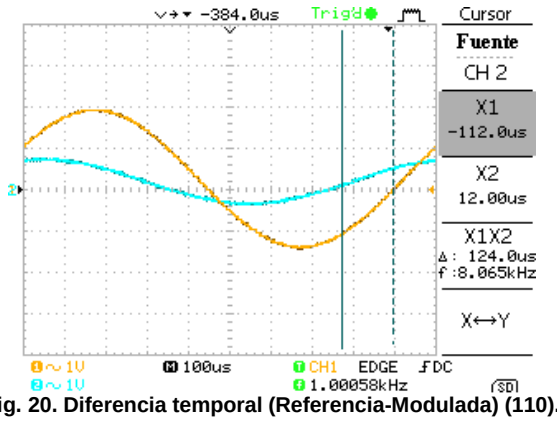

Fig. 20. Diferencia temporal (Referencia-Modulada) (110).

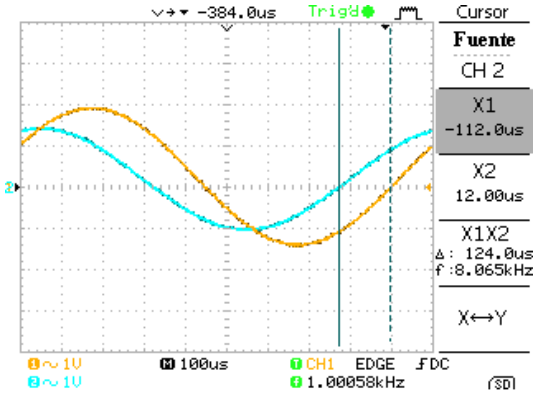

Fig. 21. Diferencia temporal (Referencia-Modulada) (111).

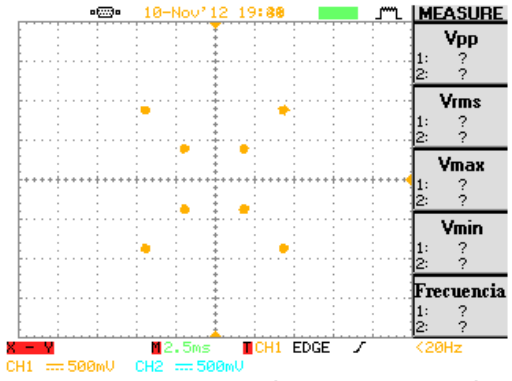

Fig. 22. Diagrama de constelación para la modulación 8QAM.

Entrada Binaria			Fase	Δt [seg]	Δφ [°]	Error relativo
0	0	0	225	6,30E-04	226,8	0,80%
0	0	1	225	6,30E-04	226,8	0,80%
0	1	0	315	8,80E-04	316,8	0,57%
0	1	1	315	8,80E-04	316,8	0,57%
1	0	0	135	3,80E-04	136,8	1,33%
1	0	1	135	3,80E-04	136,8	1,33%
1	1	0	45	1,24E-04	44,64	0,80%
1	1	1	45	1,24E-04	44,64	0,80%

Tabla 3. Ángulo de desfase experimental. Error relativo porcentual.

ACTIVIDAD ADICIONAL

Simule e implemente una modulación 16QAM para una señal portadora de frecuencia igual a la mostrada en la tabla 4 dependiendo del número asignado para su grupo. Tenga en cuenta la información que revela en la tabla 5, y recurra al texto guía para poder establecer de manera correcta las constantes que corresponden a la modulación 16QAM (coeficientes de las componentes seno y coseno)

Grupo	Frecuencia [Hz]
1	1000
2	1500
3	2000
4	2500
5	3000
6	3500
7	4000
8	4500
9	5000
10	5500

Tabla 4. Frecuencia portadora asignada para cada grupo.

Entrada Binaria				Fase 16QAM	Amp 16QAM
0	**0**	**0**	**0**	-135	0,311
0	**0**	**0**	**1**	-165	0,850
0	**0**	**1**	**0**	-45	0,311
0	**0**	**1**	**1**	-15	0,850
0	**1**	**0**	**0**	-105	0,850
0	**1**	**0**	**1**	-135	1,161
0	**1**	**1**	**0**	-75	0,850
0	**1**	**1**	**1**	-45	1,161
1	**0**	**0**	**0**	135	0,311
1	**0**	**0**	**1**	165	0,850
1	**0**	**1**	**0**	45	0,311
1	**0**	**1**	**1**	15	0,850
1	**1**	**0**	**0**	105	0,850
1	**1**	**0**	**1**	135	1,161
1	**1**	**1**	**0**	75	0,850
1	**1**	**1**	**1**	45	1,161

Tabla 5. Fase y magnitud de la señal modulada 16QAM.

ANÁLISIS DE RESULTADOS

En grupos de tres personas, se entregará un informe despues de realizada la práctica; éste debe contener:

- Título de la práctica.
- Nombre y código de los estudiantes.
- Objetivos, tanto los señalados en la guía, como los planteados por los estudiantes.
- Análisis y descripción detallada del procedimiento para realizar la actividad adicional.
- Conclusiones
- Observaciones y sugerencias (si las hay) para el mejoramiento de la metodología empleada.

DESARROLLO ACTIVIDAD ADICIONAL

Se desarrollará la actividad con una frecuencia para la señal portadora de 1 kHz. En la tabla 6 se muestra el valor real e imaginario de la señal modulada dependiendo de la entrada de 4 bits. Según el texto guía para este tipo de modulación basta con dos valores de constantes, por tal motivo y teniendo en cuenta la proximidad de los resultados obtenidos 272,955 y 273,062, 1019,081 y 1018,974 se realizó una aproximación y se seleccionaron como los valores de las constantes a implementar en los bloques *Constant* a 272,955 y 1019,081, siendo sus parámetros 18bits con punto binario en 9, y 20 bits con punto binario en 10 respectivamente. El sistema completo de la modulación 16QAM se visualiza en la fig. 23, en ella el dato de entrada de 4 bits es separado en 4 canales independientes Q,Q',I,I'. Los bits I' y Q' determinan la magnitud y los I y Q el signo de las constantes que han de multiplicar las ondas seno y coseno, las cuales posteriormente se suman y por medio de sus contribuciones individuales generan la amplitud y fase requeridas en la modulación.

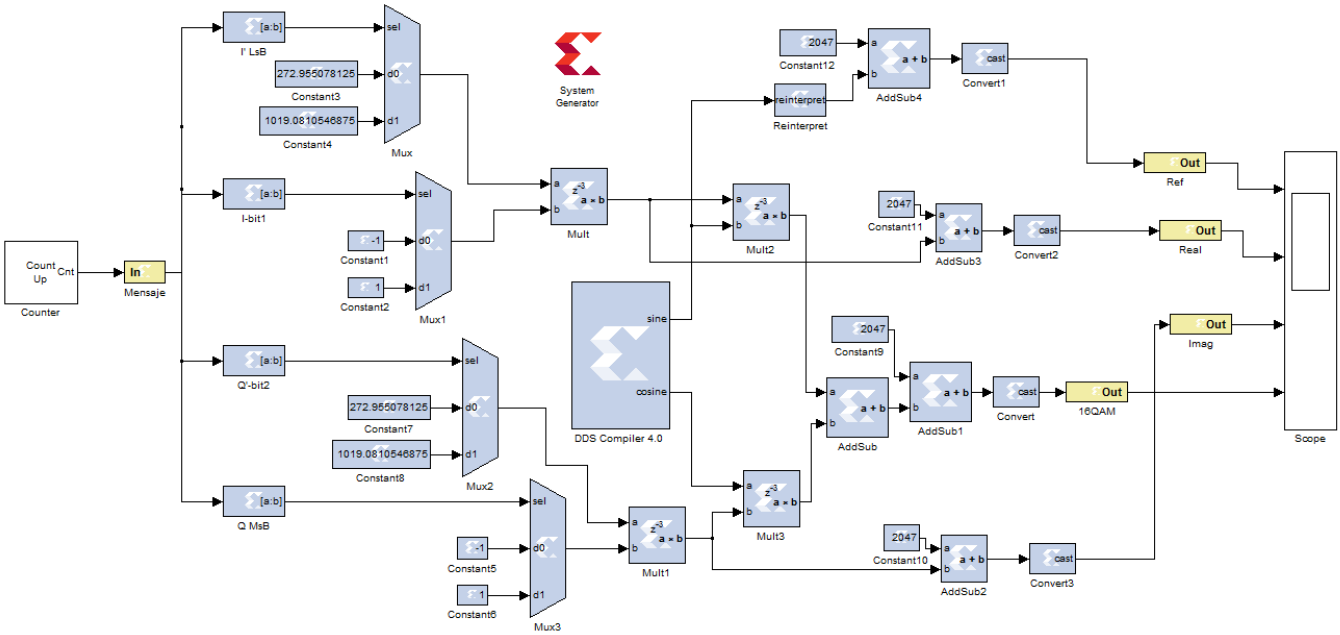

Fig. 23. Conexión para generar la modulación 16QAM.

Entrada Binaria				Teoría		Seno	Coseno
				Mag DAC	Fase	Real	Imaginario
0	0	0	0	386	-135	-272,955	-272,955
0	0	0	1	1055	-165	-1019,081	-273,062
0	0	1	0	386	-45	272,955	-272,955
0	0	1	1	1055	-15	1019,081	-273,062
0	1	0	0	1055	-105	-273,062	-1019,081
0	1	0	1	1441	-135	-1018,974	-1018,974
0	1	1	0	1055	-75	273,062	-1019,081
0	1	1	1	1441	-45	1018,974	-1018,974
1	0	0	0	386	135	-272,955	272,955
1	0	0	1	1055	165	-1019,081	273,062
1	0	1	0	386	45	272,955	272,955
1	0	1	1	1055	15	1019,081	273,062
1	1	0	0	1055	105	-273,062	1019,081
1	1	0	1	1441	135	-1018,974	1018,974
1	1	1	0	1055	75	273,062	1019,081
1	1	1	1	1441	45	1018,974	1018,974

Tabla 6. Coeficientes para componentes Seno y Coseno.

En la fig. 24 se observa la simulación de la modulación 16QAM teniendo como entrada un contador de 0 a 15 (4 bits) con pasos de 50.000, para un tiempo de 800.000 segundos.
De la fig. 25 a la fig. 40 se observa la implementación de la modulación 16QAM dependiendo de la entrada de 4 bits controlada por cuatro *switchs* y estableciendo como salida la señal referencia, señal modulada, también se asignan canales para el nivel de la parte real y el nivel de la parte imaginaria de la señal modulada, que hacen referencia a la envolvente compleja, al igual que para la señal referencia. En ellas se puede verificar la amplitud correspondiente a cada símbolo de entrada.

De la fig. 41 a la fig. 56 se visualiza la medición de la diferencia temporal entre la señal de referencia con fase 0° y la modulada 16QAM, para el posterior cálculo de los desfases y la comparación con los valores teóricos, los cuales se encuentran condensados en la tabla 7, la cual muestra cada ángulo de desfase experimental obtenido y el error relativo con respecto al valor teórico.

La fig. 57 muestra el diagrama de constelación para la modulación 16QAM.

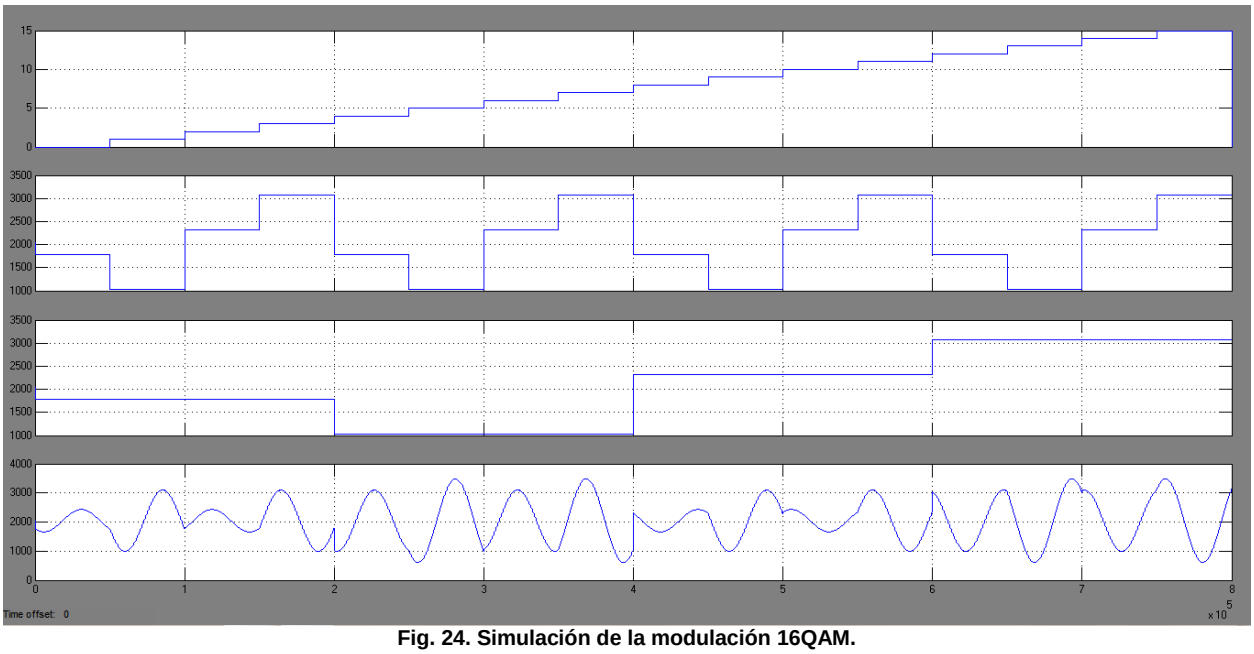

Fig. 24. Simulación de la modulación 16QAM.

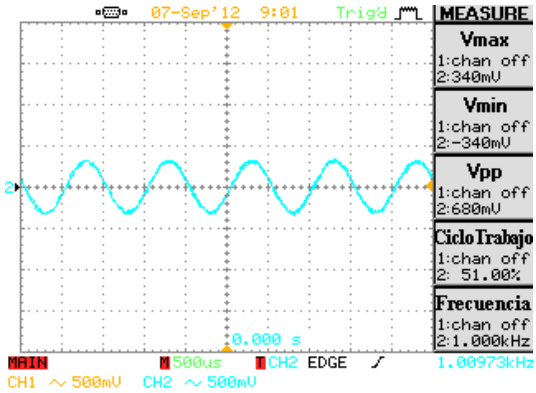

Fig. 25. Implementación 16QAM (0000).

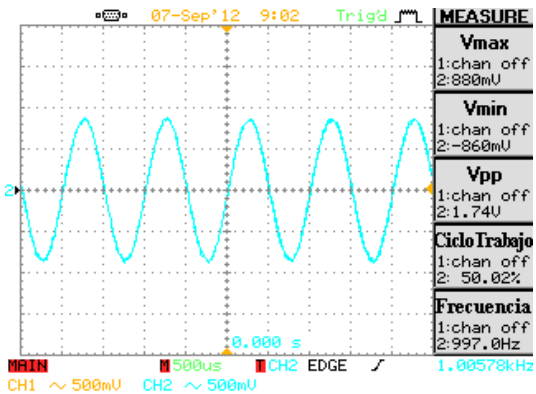

Fig. 26. Implementación 16QAM (0001).

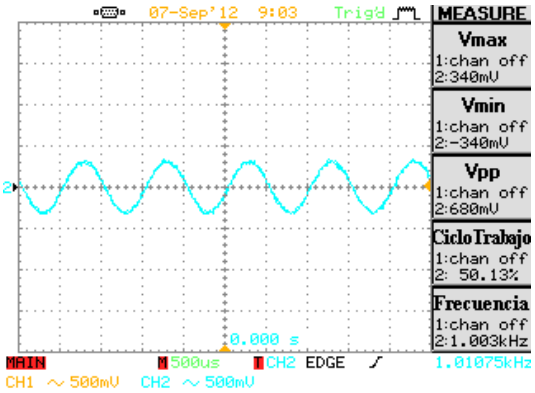

Fig. 27. Implementación 16QAM (0010).

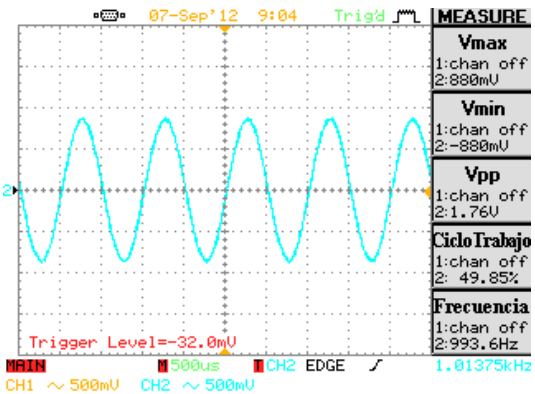

Fig. 28. Implementación 16QAM (0011).

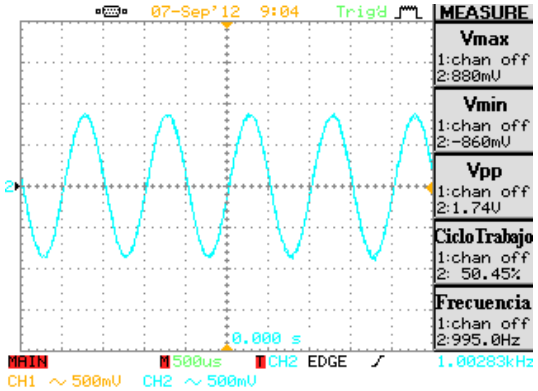

Fig. 29. Implementación 16QAM (0100).

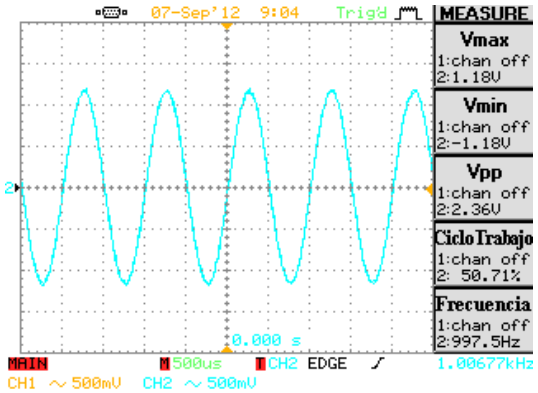

Fig. 30. Implementación 16QAM (0101).

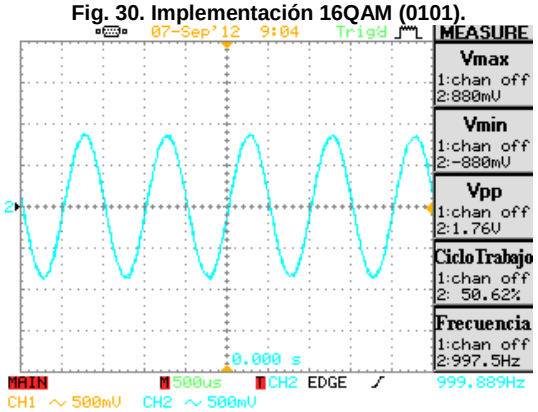

Fig. 31. Implementación 16QAM (0110).

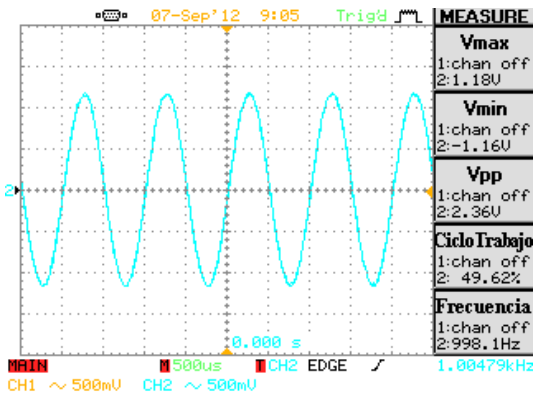

Fig. 32. Implementación 16QAM (0111).

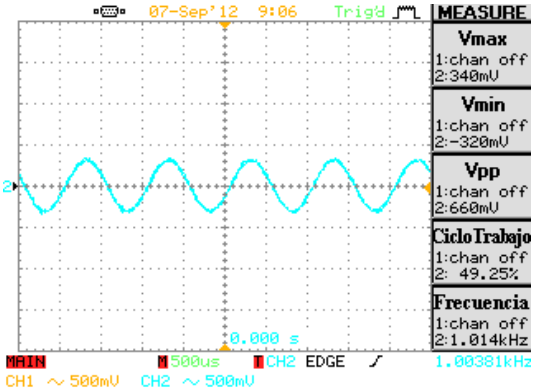

Fig. 33. Implementación 16QAM (1000).

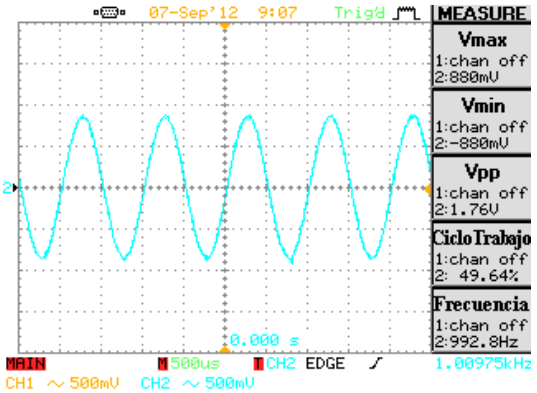

Fig. 34. Implementación 16QAM (1001).

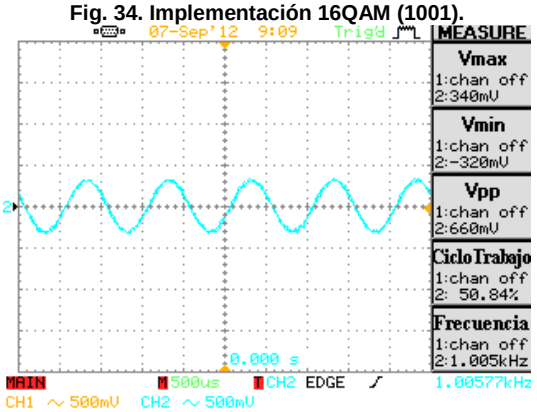

Fig. 35. Implementación 16QAM (1010).

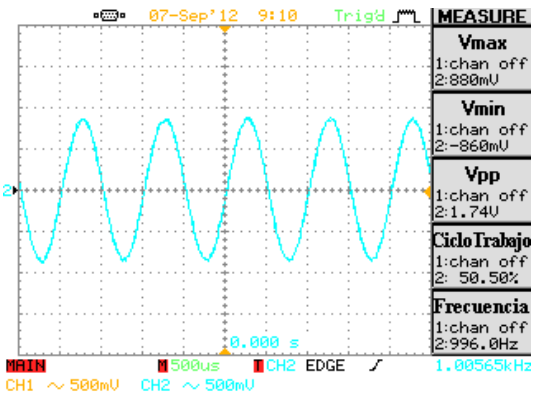

Fig. 36. Implementación 16QAM (1011).

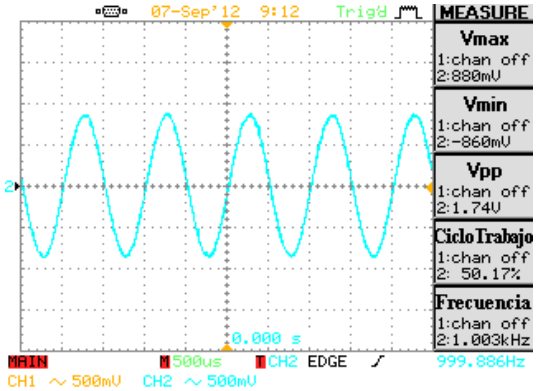

Fig. 37. Implementación 16QAM (1100).

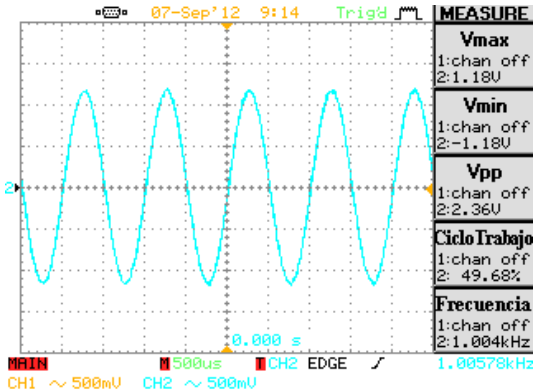

Fig. 38. Implementación 16QAM (1101).

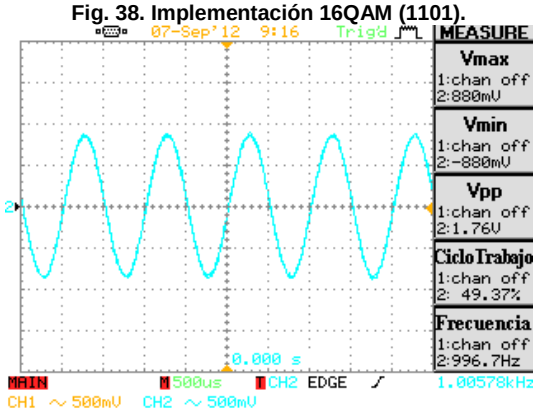

Fig. 39. Implementación 16QAM (1110).

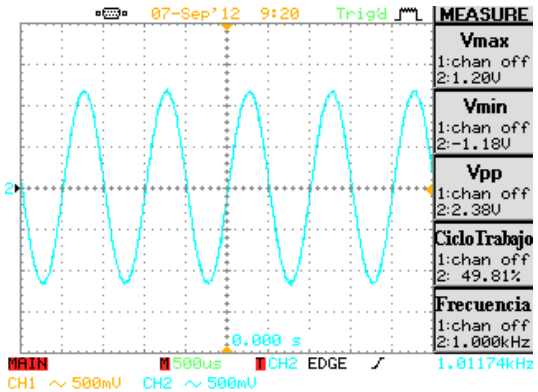

Fig. 40. Implementación 16QAM (1111).

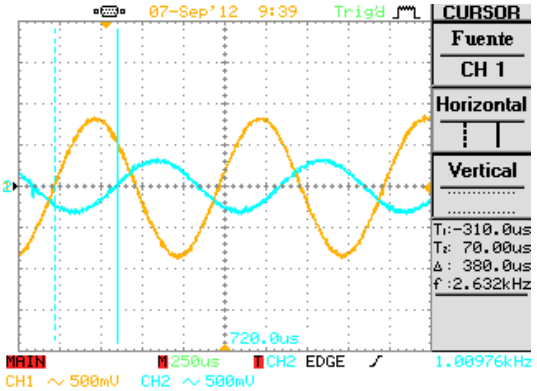

Fig. 41. Diferencia temporal (Referencia-Modulada) (0000).

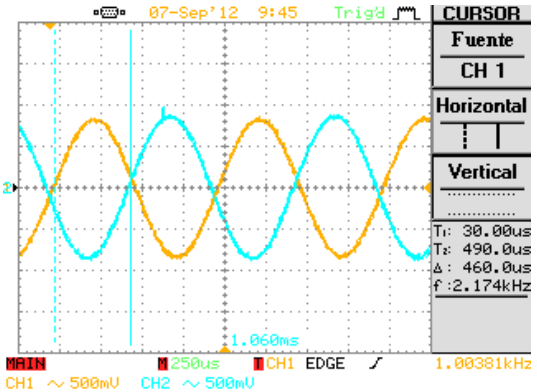

Fig. 42. Diferencia temporal (Referencia-Modulada) (0001).

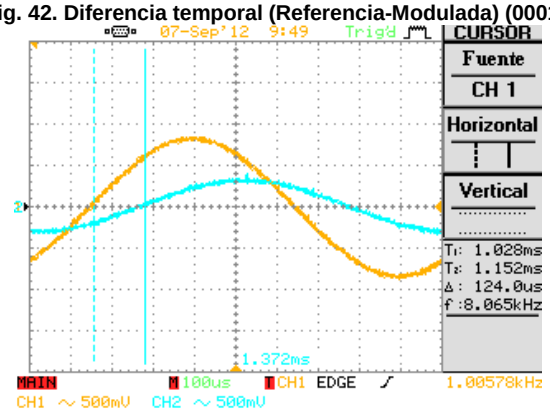

Fig. 43. Diferencia temporal (Referencia-Modulada) (0010).

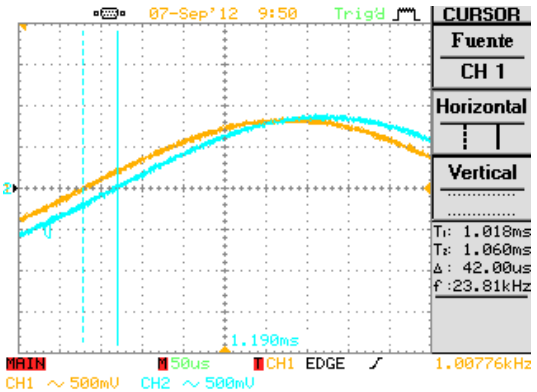

Fig. 44. Diferencia temporal (Referencia-Modulada) (0011).

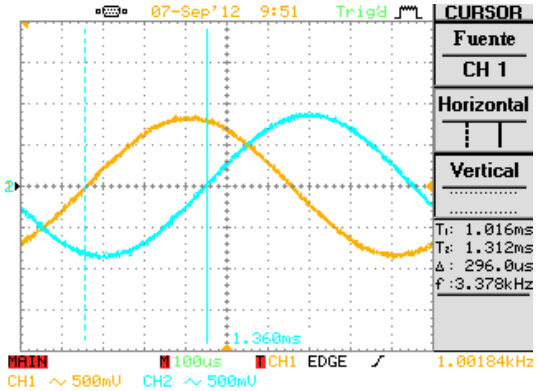

Fig. 45. Diferencia temporal (Referencia-Modulada) (0100).

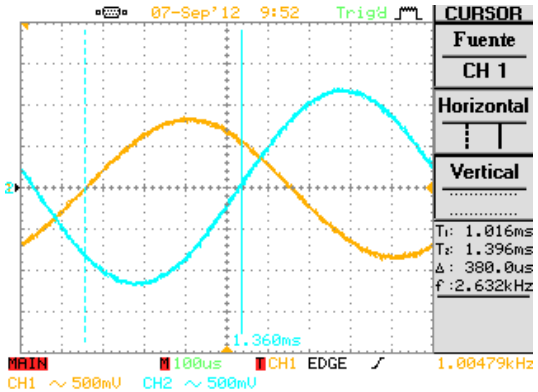

Fig. 46. Diferencia temporal (Referencia-Modulada) (0101).

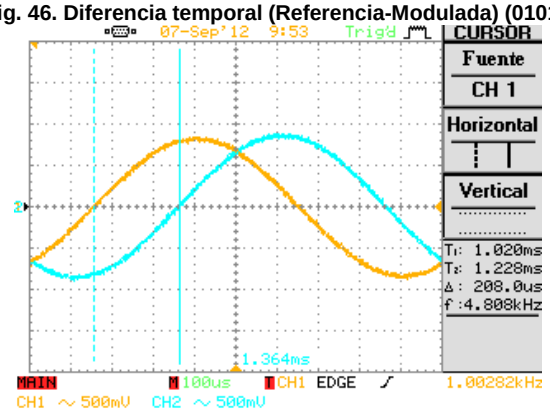

Fig. 47. Diferencia temporal (Referencia-Modulada) (0110).

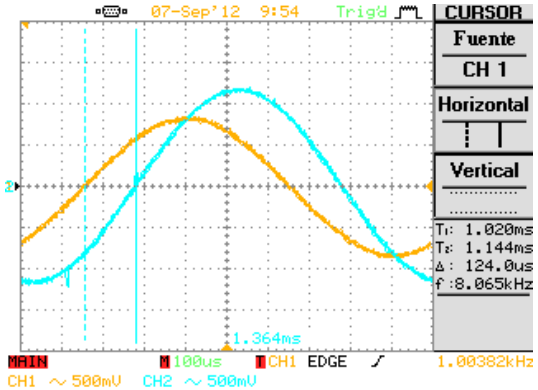

Fig. 48. Diferencia temporal (Referencia-Modulada) (0111).

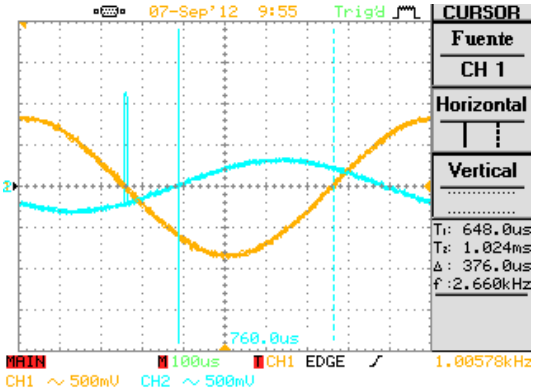

Fig. 49. Diferencia temporal (Referencia-Modulada) (1000).

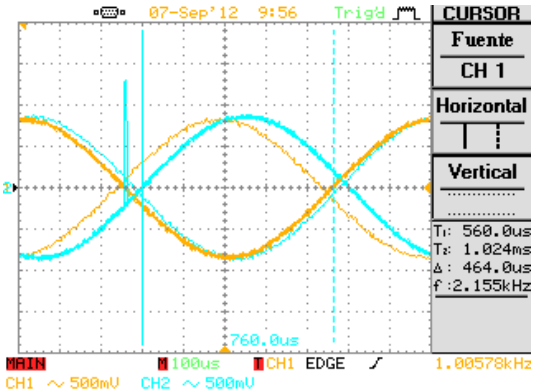

Fig. 50. Diferencia temporal (Referencia-Modulada) (1001).

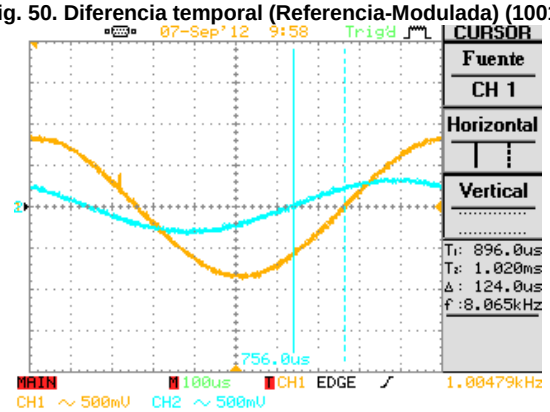

Fig. 51. Diferencia temporal (Referencia-Modulada) (1010).

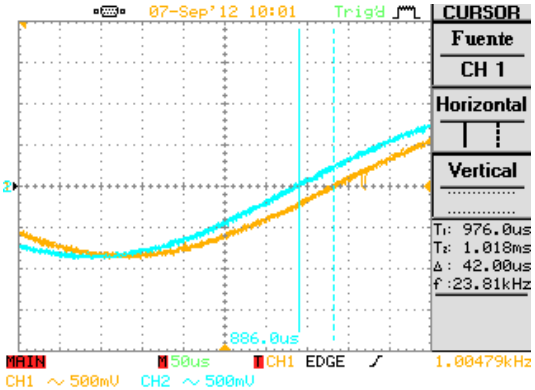

Fig. 52. Diferencia temporal (Referencia-Modulada) (1011).

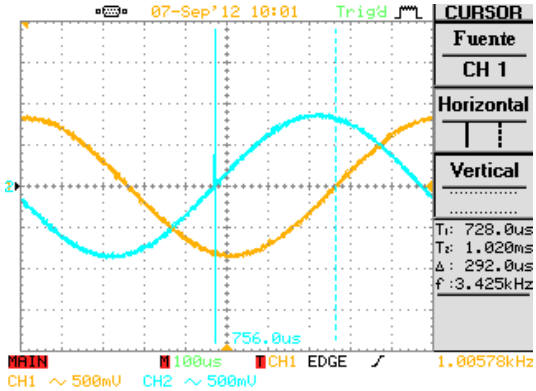

Fig. 53. Diferencia temporal (Referencia-Modulada) (1100).

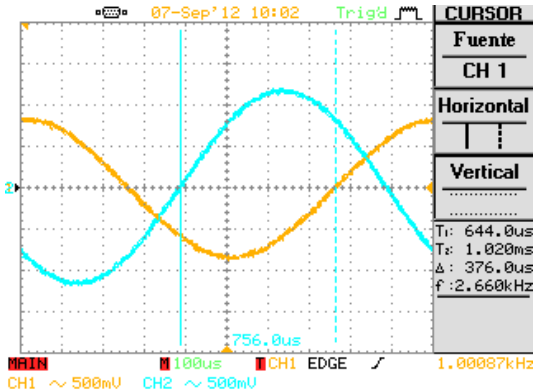

Fig. 54. Diferencia temporal (Referencia-Modulada) (1101).

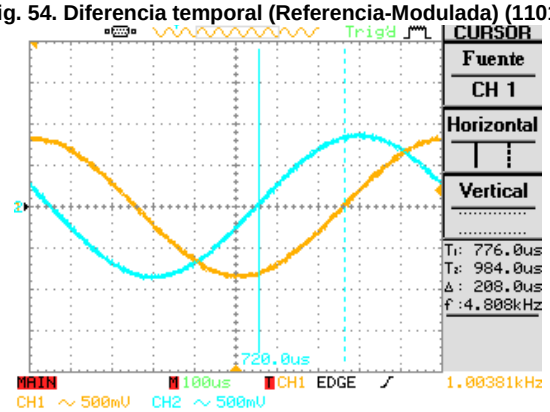

Fig. 55. Diferencia temporal (Referencia-Modulada) (1110).

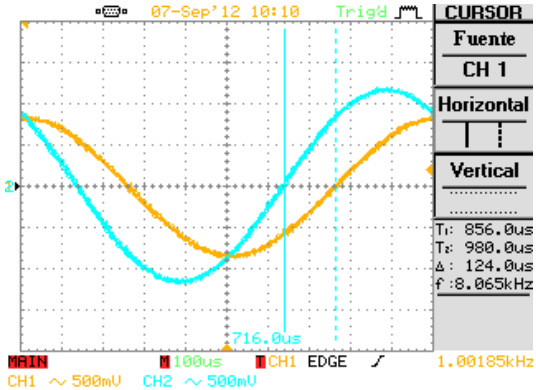

Fig. 56. Diferencia temporal (Referencia-Modulada) (1111).

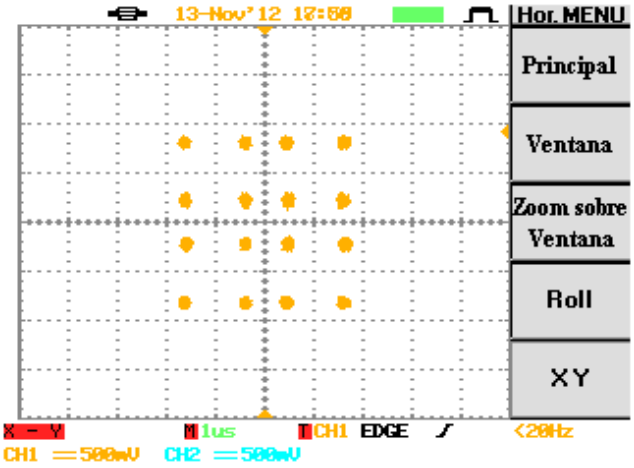

Fig. 56. Diagrama de constelación para la modulación 16QAM.

Entrada Binaria				Fase	Δt [seg]	Δθ [°]	Error relativo
0	0	0	0	-135	3,80E-04	-136,8	1,33%
0	0	0	1	-165	4,60E-04	-165,6	0,36%
0	0	1	0	-45	1,24E-04	-44,64	-0,80%
0	0	1	1	-15	4,20E-05	-15,12	0,80%
0	1	0	0	-105	2,96E-04	-106,56	1,49%
0	1	0	1	-135	3,80E-04	-136,8	1,33%
0	1	1	0	-75	2,08E-04	-74,88	-0,16%
0	1	1	1	-45	1,24E-04	-44,64	-0,80%
1	0	0	0	135	3,76E-04	135,36	0,27%
1	0	0	1	165	4,64E-04	167,04	1,24%
1	0	1	0	45	1,24E-04	44,64	-0,80%
1	0	1	1	15	4,20E-05	15,12	0,80%
1	1	0	0	105	2,92E-04	105,12	0,11%
1	1	0	1	135	3,76E-04	135,36	0,27%
1	1	1	0	75	2,08E-04	74,88	-0,16%
1	1	1	1	45	1,24E-04	44,64	-0,80%

Tabla 7. Ángulo de desfase experimental. Error relativo porcentual.

UNIVERSIDAD INDUSTRIAL DE SANTANDER
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
Perfecta Combinación entre Energía e Intelecto

LABORATORIO No 6 DE COMUNICACIONES DIGITALES

Modulación por Amplitud de Pulso PAM

Utilizando Xilinx System Generator e implementando en FPGA Spartan 3AN

OBJETIVOS

- Reconocer parámetros configurables para el correcto procesamiento de las señales digitales.
- Verificar el comportamiento en el tiempo de la señal de salida correspondiente a la modulación por amplitud de pulso de la señal de referencia.

BIBLIOGRAFÍA

- *Anexo A del proyecto de grado "Diseño de prácticas para laboratorio de Comunicaciones Digitales basado en Simulink y Xilinx System Generator"*
- *Spartan-3A/3AN FPGA Starter Kit Board User Guide (Datasheet)*
- *LogiCORE IP DDS Compiler v4.0 (Datasheet)*
- *Sistemas de Comunicaciones Electrónicas,* ***Wayne Tomasi****, 4 edición, capitulo 15 Transmisión Digital.*

ACTIVIDAD DIRIGIDA

Modulación PAM sin retorno a cero.

Realizar el montaje de la fig. 1 que describe una modulación PAM con muestreo sin retorno a cero, con los siguientes bloques de *Xilinx* en *Simulink*:

- Xilinx Blockset /Basic Elements/System Generator.
- Xilinx Blockset /Basic Elements/Register.
- Xilinx Blockset /Basic Elements/Delay.
- Xilinx Blockset /Basic Elements/Constant.
- Xilinx Blockset / Basic Elements / Gateway Out.
- Xilinx Blockset / Basic Elements / Gateway In.
- Xilinx Blockset / DSP / DDS compiler V4.
- Xilinx Blockset / Basic Elements / Counter.
- Xilinx Blockset / Basic Elements / Reinterpret.
- Xilinx Blockset / Basic Elements / Convert.
- Xilinx Blockset / Math / Mult.
- Xilinx Blockset / Math / CMult.
- Xilinx Blockset / Math / Relational.
- Xilinx Blockset / Math / Addsub.

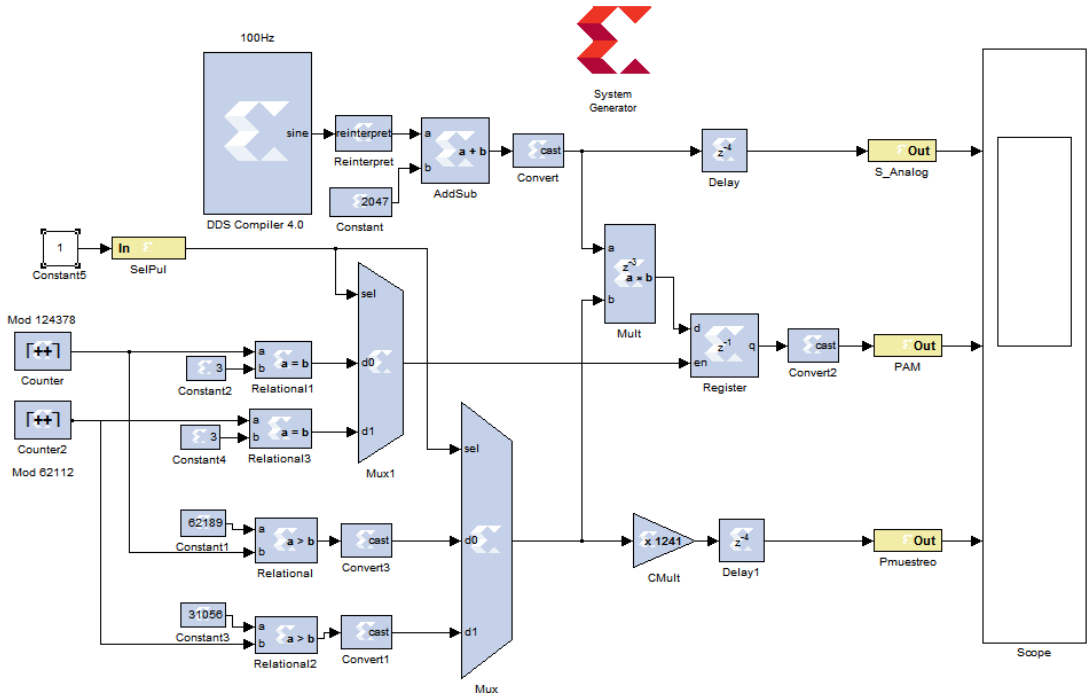

Fig. 1. Conexión para generar la modulación PAM con muestreo sin retorno a cero.

Se configura el bloque *DDS Compiler* con una frecuencia de 100Hz, con 26 bits para el ancho de fase, un incremento de ancho de fase de 1342 (‘10000110’) y 12 bits para la amplitud de la señal de salida (3.3 Vpp luego de pasar por el bloque *Reinterpret*). Se utilizan los *Muxs* para elegir entre dos frecuencias de muestreo diferentes para la señal de referencia. El módulo de cada contador se establece dependiendo de la cantidad requerida de muestras por periodo de la señal de referencia. Para este caso vamos a establecer las opciones de 4 y 8 muestras por periodo de la señal de referencia, la siguiente es la ecuación que establece el módulo del contador dependiendo de la cantidad de muestras por periodo.

$$Módulo\ del\ contador = \frac{50\ MHz}{100\ Hz * \#\ de\ muestras} \qquad (2)$$

De esta forma los módulos de los contadores son de 124378 y 62112 respectivamente y se configura por ejemplo para el contador módulo 124378 tal como se muestra en la fig. 2.

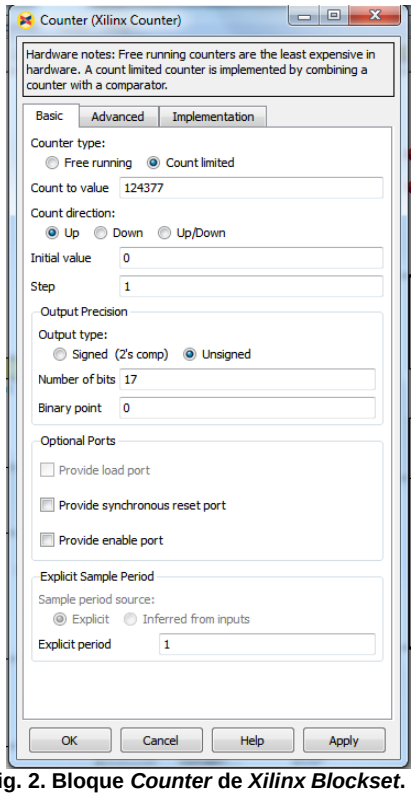

Fig. 2. Bloque *Counter* de *Xilinx Blockset*.

Cada contador entra a un bloque comparador (*Relational*) tipo a>b, dónde “a” es una constante de valor igual a la mitad del módulo del contador y “b” es el contador. Lo anterior establece una señal en ‘1’ durante la primera mitad del contador y un ‘0’ en la otra mitad. Dicha señal se multiplica por la señal referencia, el resultado es una señal que toma el valor de la referencia durante la primera mitad del contador y cero en la otra mitad, que luego se introduce en un registro que funciona como retenedor de orden cero, que es habilitado por un pulso que se genera cuando el contador es igual a 3, esto porque el bloque de multiplicación (*Mult*) genera un retraso de 3 muestras. La salida del Registro corresponde a la señal PAM sin retorno a cero. Como el registro posee un retardo de una muestra, al sumarlo con los tres retardos del bloque *Mult* se obtienen cuatro retardos que serán los aplicados mediante los bloques *Delay* a la señal de referencia y a la señal de muestreo, esta última es multiplicada por 1241 para lograr un valor de 1 V y ser visible al momento de implementar, pues si se deja tal como estaba no podría percibirse, todo esto para que todas las señales coincidan en el tiempo.

La fig. 3 y la fig. 4 muestra la simulación del sistema con una señal de entrada de un bit (‘0’ o ‘1’ respectivamente) establecida por una constante. El bloque *Gateway In* debe estar configurado para recibir la misma cantidad de bits que vienen del bloque de la constante. La primera parte de la gráfica se observa sin respuesta porque la señal sinusoidal demora en aparecer algunos flancos del reloj principal de 50 MHz, pero es solo para la primera muestra, en estado estable se mantiene el correcto muestreo.

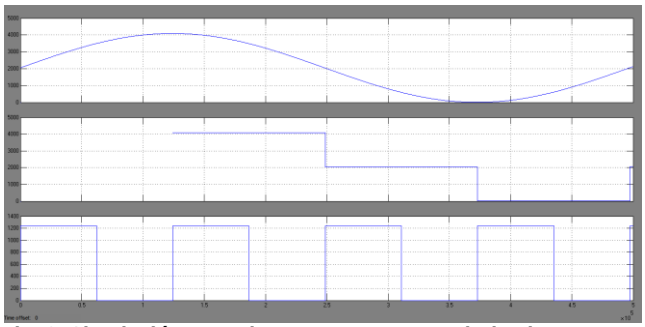

Fig. 3. Simulación PAM de 4 muestras por periodo sin retorno a cero.

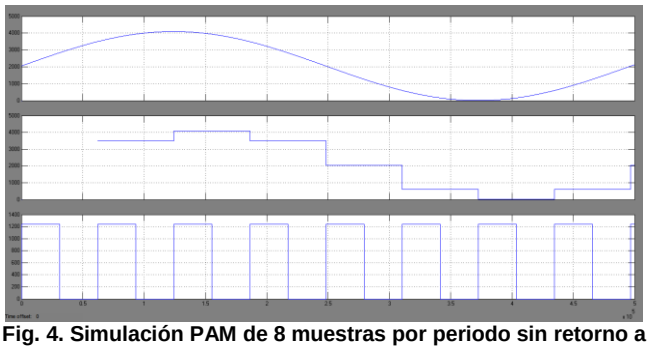

Fig. 4. Simulación PAM de 8 muestras por periodo sin retorno a cero.

Se genera el código para incluirlo como *Component* en un nuevo proyecto enlazado con el DAC.

Los resultados de la implementación para la

modulación PAM se observan en la fig. 5, fig. 6, fig. 7 y fig. 8, estableciendo tres salidas para la señal referencia, la señal modulada, y el tren de pulsos para muestreo y una entrada de un bits controlada por un *switch* para establecer la cantidad de muestras por periodo.

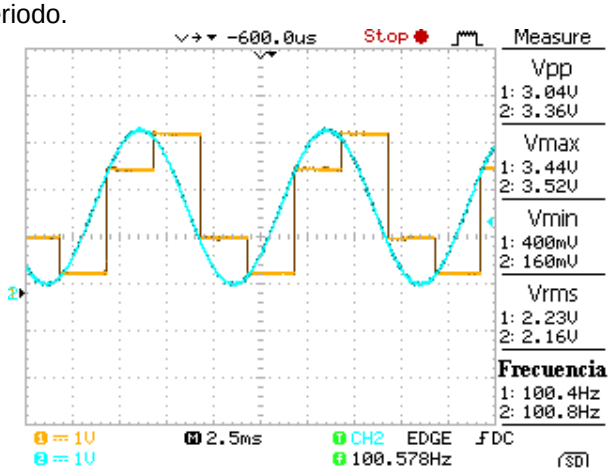

Fig. 5. Señal referencia y PAM (4 muestras por periodo).

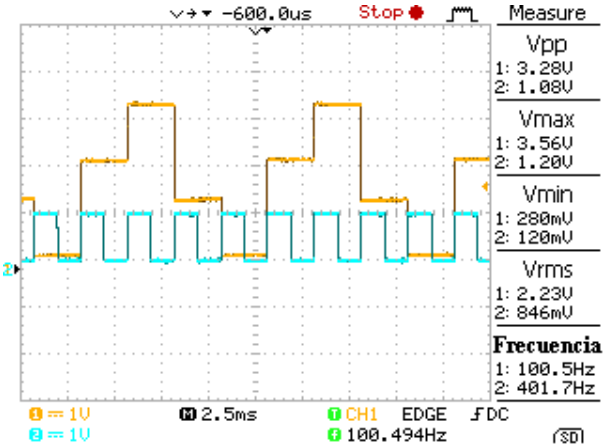

Fig. 6. Tren de pulsos y PAM (4 muestras por periodo).

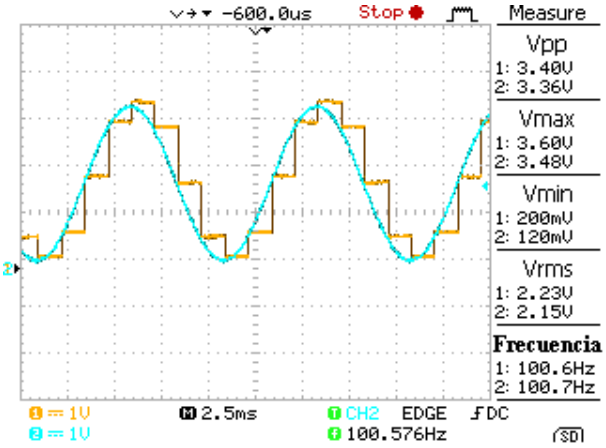

Fig. 7. Señal referencia y PAM (8 muestras por periodo).

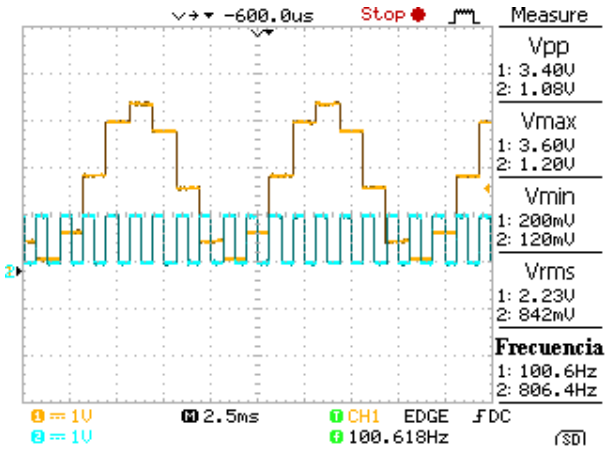

Fig. 8. Tren de pulsos y PAM (8 muestras por periodo).

ACTIVIDAD ADICIONAL

Simule e implemente una modulación PAM **con retorno a cero**, con valores para la amplitud de la señal referencia, frecuencia de la misma, y cantidad de muestras por periodo igual a los mostrados en la tabla 1 dependiendo del número asignado para su grupo.

Grupo	Vpp [Volts]	Frecuencia [Hz]	Muestras por periodo
1	3.3	100	4 y 8
2	3.0	200	4 y 8
3	2.7	400	4 y 8
4	2.4	100	4 y 6
5	2.1	200	4 y 6
6	3.3	400	4 y 6
7	3.0	100	6 y 8
8	2.7	200	6 y 8
9	2.4	400	6 y 8
10	2.1	600	4 y 8

Tabla 1. Vpp, frecuencia y cantidad de muestras por periodo para cada grupo.

ANÁLISIS DE RESULTADOS

En grupos de tres personas, se entregará un informe despues de realizada la práctica; éste debe contener:

- Título de la práctica.
- Nombre y código de los estudiantes.
- Objetivos, tanto los señalados en la guía, como los planteados por los estudiantes.
- Análisis y descripción detallada del procedimiento para realizar la actividad adicional.
- Conclusiones
- Observaciones y sugerencias (si las hay) para el mejoramiento de la metodología empleada.

DESARROLLO ACTIVIDAD ADICIONAL

Se desarrollará la actividad para una frecuencia de 100 Hz, un voltaje pico a pico de 3.3 Volts y se tendrá para elegir entre 4 u 8 muestras por periodo.

La parte inferior del diagrama de bloques es la misma generación del tren de pulsos para la modulación PAM sin retorno a cero, lo diferente está en cómo se habilita el registro, pues en ahora se utilizaron contadores con módulo igual a la mitad del valor del módulo de los contadores de la parte inferior, de esta forma el registro estará habilitado por un pulso cuando el tren de pulsos de muestreo esté en '1' y también cuando esté en '0', dicho pulso habilitador estará cuándo el contador es igual a 3 por efectos del retardo de tres muestras del bloque *Mult*. Para el resto de la configuración del sistema la explicación es la misma que ara la modulación PAM sin retorno a cero.

La fig. 10 y la fig. 11 muestra la simulación del sistema con una señal de entrada de un bit ('0' o '1' respectivamente) establecida por una constante. El bloque *Gateway In* debe estar configurado para recibir la misma cantidad de bits que vienen del bloque de la constante. La primera parte de la gráfica se observa sin respuesta porque la señal sinusoidal demora en aparecer algunos flancos del reloj principal de 50 MHz, pero es solo para la primera muestra, en estado estable se mantiene el correcto muestreo.

Los resultados de la implementación para la modulación PAM con retorno a cero se observan en la fig. 12, fig. 13, fig. 14 y fig. 15, estableciendo tres salidas para la señal referencia, la señal modulada, y el tren de pulsos para muestreo y una entrada de un bits controlada por un *switch* para establecer la cantidad de muestras por periodo.

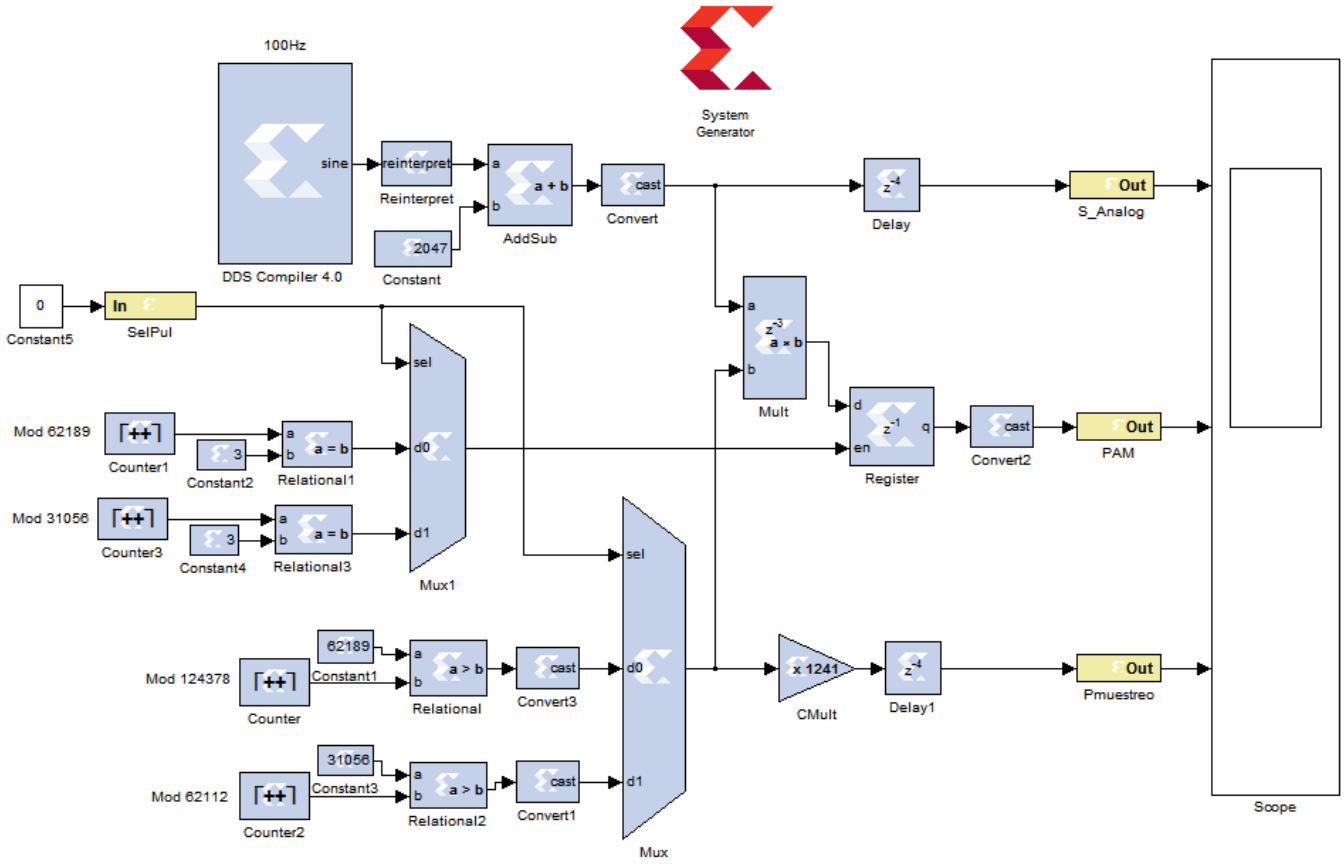

Fig. 9. Conexión para generar la modulación PAM con muestreo con retorno a cero.

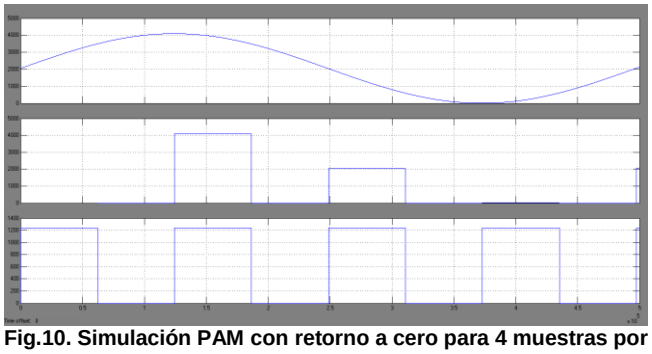

Fig.10. Simulación PAM con retorno a cero para 4 muestras por periodo.

Fig.11. Simulación PAM con retorno a cero para 8 muestras por periodo.

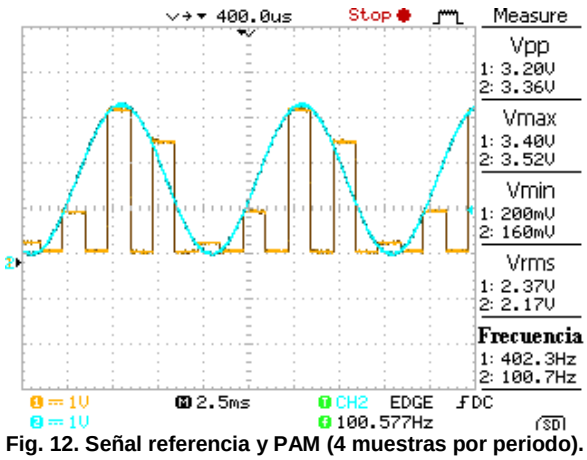

Fig. 12. Señal referencia y PAM (4 muestras por periodo).

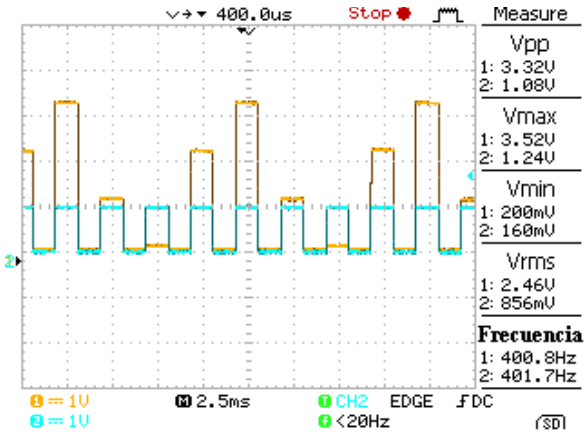

Fig. 13. Tren de pulsos y PAM (4 muestras por periodo).

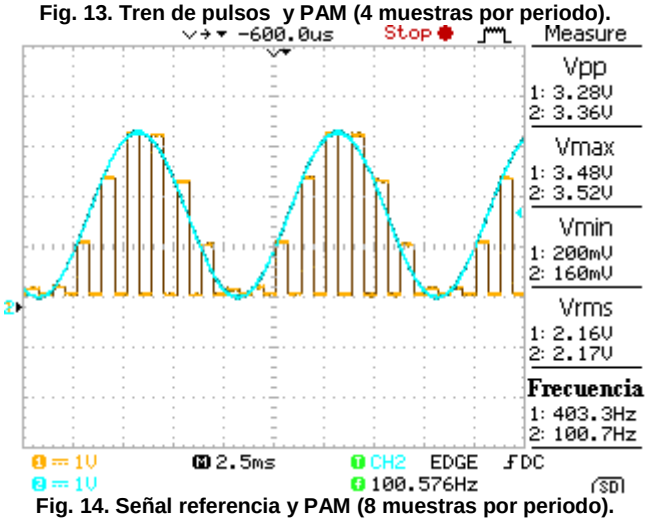

Fig. 14. Señal referencia y PAM (8 muestras por periodo).

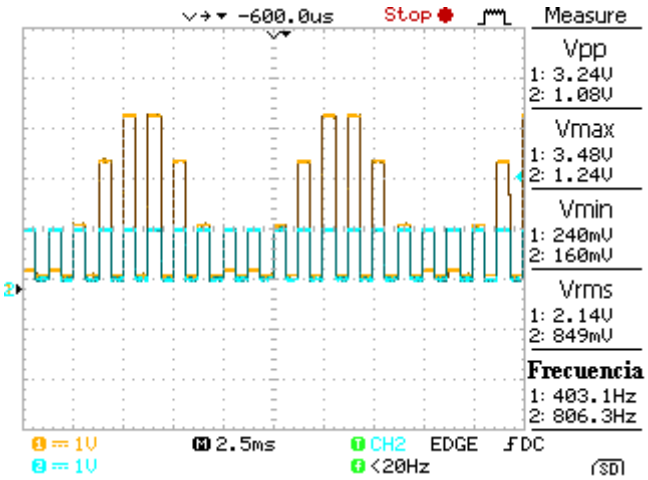

Fig. 15. Tren de pulsos y PAM (8 muestras por periodo).

LABORATORIO No 7 DE COMUNICACIONES DIGITALES

Modulación por Codificación de Pulso PCM

Utilizando Xilinx System Generator e implementando en FPGA Spartan 3AN

OBJETIVOS

- Reconocer parámetros configurables para el correcto procesamiento de las señales digitales.
- Verificar el comportamiento en el tiempo de la señal de salida correspondiente a la modulación por codificación de pulso de la señal de referencia.

BIBLIOGRAFÍA

- *Anexo A del proyecto de grado "Diseño de prácticas para laboratorio de Comunicaciones Digitales basado en Simulink y Xilinx System Generator"*
- *Spartan-3A/3AN FPGA Starter Kit Board User Guide (Datasheet)*
- *LogiCORE IP DDS Compiler v4.0 (Datasheet)*
- *Sistemas de Comunicaciones Electrónicas,* ***Wayne Tomasi****, 4 edición, capitulo 15 Transmisión Digital.*

ACTIVIDAD DIRIGIDA

Modulación PCM Delta

Realizar el montaje de la fig. 1 que describe una modulación PCM Delta que se constituye en interpretar una señal PAM de tal manera que se muestre un '1' lógico cuando el dato actual sea mayor al dato anterior, y un '0' lógico cuando el dato actual sea menor al dato anterior.

Se utilizan los siguientes bloques de *Xilinx* en *Simulink*:

- Xilinx Blockset /Basic Elements/System Generator.
- Xilinx Blockset /Basic Elements/Register.
- Xilinx Blockset /Basic Elements/Delay.
- Xilinx Blockset /Basic Elements/Constant.
- Xilinx Blockset / Basic Elements / Gateway Out.
- Xilinx Blockset / Basic Elements / Gateway In.
- Xilinx Blockset / DSP / DDS compiler V4.
- Xilinx Blockset / Basic Elements / Counter.
- Xilinx Blockset / Basic Elements / Reinterpret.
- Xilinx Blockset / Basic Elements / Convert.
- Xilinx Blockset / Math / Mult.
- Xilinx Blockset / Math / CMult.
- Xilinx Blockset / Math / Relational.
- Xilinx Blockset / Math / Addsub.
- Xilinx Blockset / Index / MCode.

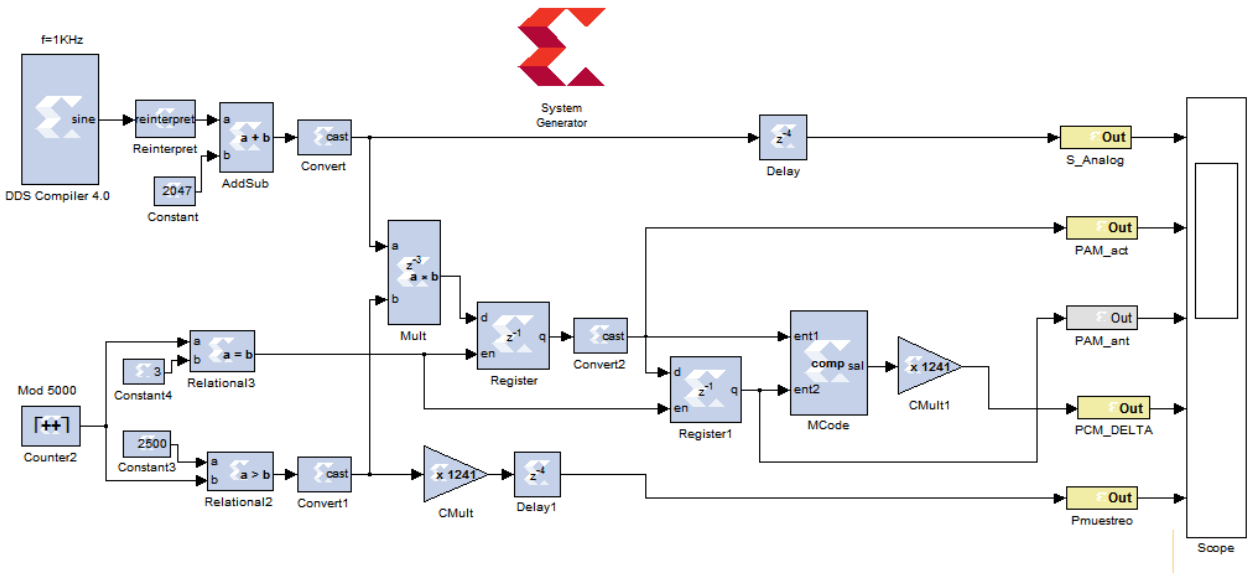

Fig. 1. Conexión para generar la modulación PCM.

El bloque *DDS Compiler* se configura para una frecuencia de 1 kHz, y el contador de módulo 5000 para obtener 10 muestras por periodo de la señal referencia.

La primera parte del sistema, hasta la salida del primer registro, es exactamente la modulación por amplitud de pulsos PAM sin retorno a cero, ésta será la entrada de un registro habilitado por la misma señal que el anterior registro, esto con el fin de guardar el dato actual y el dato anterior, que serán las entradas del bloque *MCode* que compara estos dos valores para determinar si el dato actual es mayor o menor que el dato anterior, dependiendo de esto la salida del comparador será un '1' o un '0' respectivamente. Ésta última señal será la modulación PCM Delta que se multiplica por 1241 para poder ser visualizada en el osciloscopio con un valor de 1 Volt si es '1' o 0 Volts si es un '0'. El código que representa dicho comparador y que se encuentra dentro del bloque *MCode* es el siguiente:

```
function sal = comp(ent1,ent2)
if ent1 > ent2
    sal = 1;
  else
    sal = 0;
  end
end
```

La fig. 3 muestra la simulación del sistema, en orden descendente se observa la señal referencia, PAM para ver el dato anterior, PAM para ver el dato actual, la señal modulada PCM y el tren de pulsos de muestreo. La primera parte de la gráfica se observa sin respuesta porque la señal sinusoidal demora en aparecer algunos flancos del reloj principal de 50 MHz, pero es solo para la primera muestra, en estado estable se mantiene el correcto muestreo.

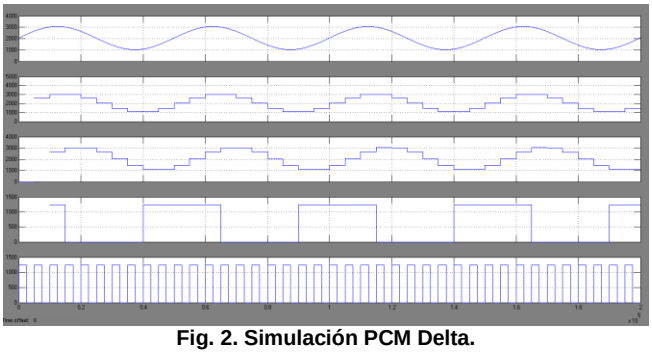

Fig. 2. Simulación PCM Delta.

Se genera el código para incluirlo como *Component* en un nuevo proyecto enlazado con el DAC.

Los resultados de la implementación para la modulación PCM se observan en la fig. 3, fig. 4, y fig. 5, estableciendo cuatro salidas para la señal referencia, la señal modulada PAM del dato actual, la señal modulada PCM y el tren de pulsos para muestreo.

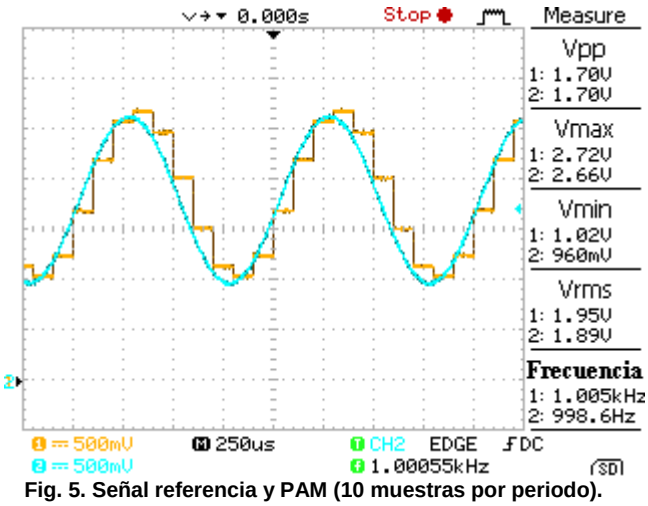

Fig. 5. Señal referencia y PAM (10 muestras por periodo).

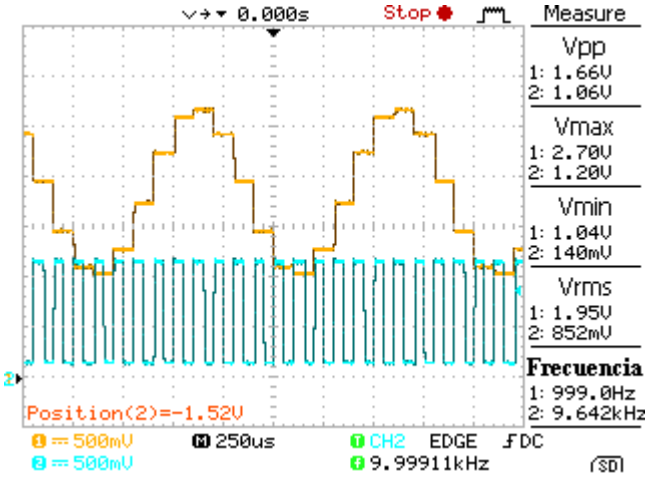

Fig. 6. Tren de pulsos y PAM (10 muestras por periodo).

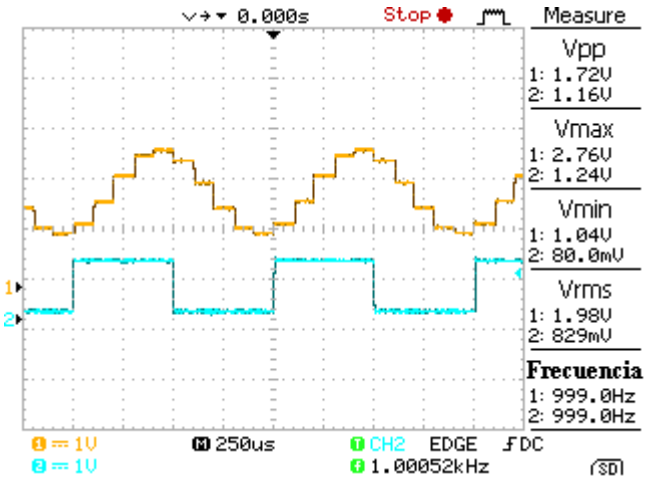

Fig. 7. Señal modulada PCM Delta y PAM.

ACTIVIDAD ADICIONAL

Simule e implemente una modulación PCM Delta tomando como entada una señal diferente a una sinusoidal pura, con 10 muestras por periodo de la señal referencia, con valores para la amplitud de la señal referencia igual a los mostrados en la tabla 1 dependiendo del número asignado para su grupo.

Grupo	Vpp [Volts]
1	3.3
2	3.0
3	2.7
4	2.4
5	2.1
6	3.3
7	3.0
8	2.7
9	2.4
10	2.1

Tabla 1. Vpp para cada grupo.

ANÁLISIS DE RESULTADOS

En grupos de tres personas, se entregará un informe despues de realizada la práctica; éste debe contener:

- Título de la práctica.
- Nombre y código de los estudiantes.
- Objetivos, tanto los señalados en la guía, como los planteados por los estudiantes.
- Análisis y descripción detallada del procedimiento para realizar la actividad adicional.
- Conclusiones
- Observaciones y sugerencias (si las hay) para el mejoramiento de la metodología empleada.

DESARROLLO ACTIVIDAD ADICIONAL

Se desarrollará la actividad para una amplitud pico a pico de 3.3 Volts y tomando como señal referencia la suma (a través de un bloque *AddSub*) de dos ondas sinusoidales generadas por dos bloques *DDS Compiler*, con frecuencias de 1 kHz y 3 kHz respectivamente, el resto del sistema es la misma configuración de la actividad dirigida. El diagrama completo se observa en la fig. 8

La fig. 9 muestra la simulación del sistema, en orden descendente se visualizan las graficas de la señal de referencia, PAM para ver el dato anterior, PAM para ver el dato actual, la modulación PCM delta y el tren de pulsos de muestreo. La primera parte de la gráfica se observa sin respuesta porque la señal sinusoidal demora en aparecer algunos flancos del reloj principal de 50 MHz, pero es solo para la primera muestra, en estado estable se mantiene el correcto muestreo.

Los resultados de la implementación para la modulación PCM se observan en la fig. 10, fig. 11 y fig. 12, estableciendo cuatro salidas para la señal referencia, la señal modulada PAM del dato actual, la señal modulada PCM y el tren de pulsos para muestreo.

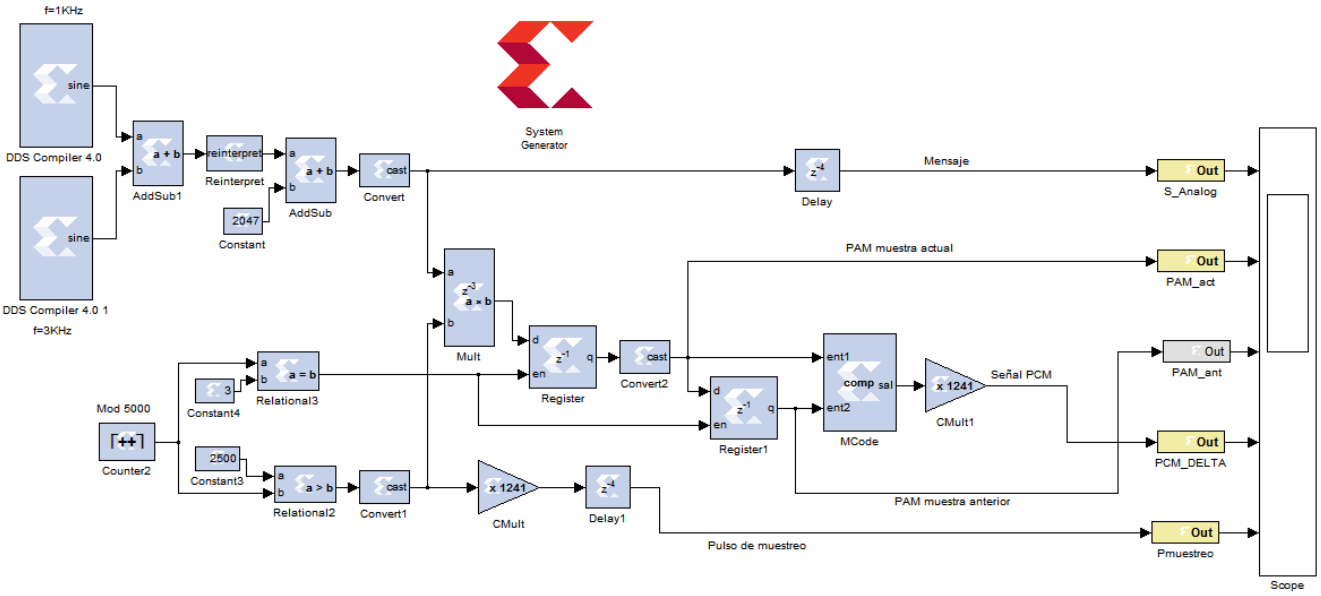

Fig. 8. Conexión para generar la modulación PCM Delta.

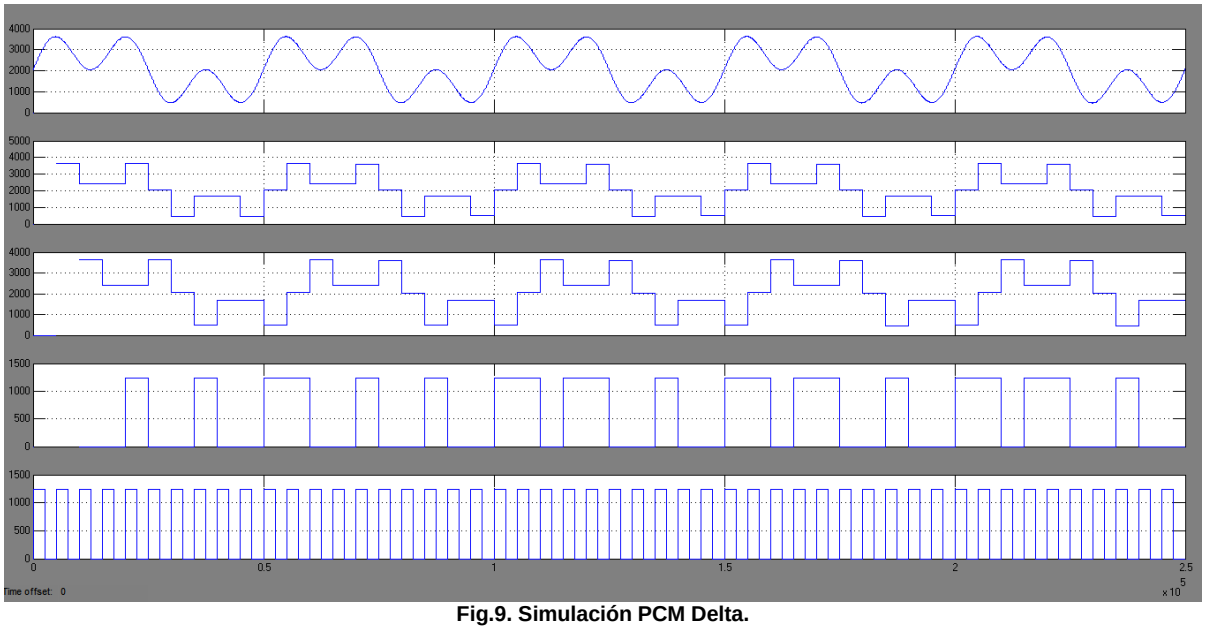

Fig.9. Simulación PCM Delta.

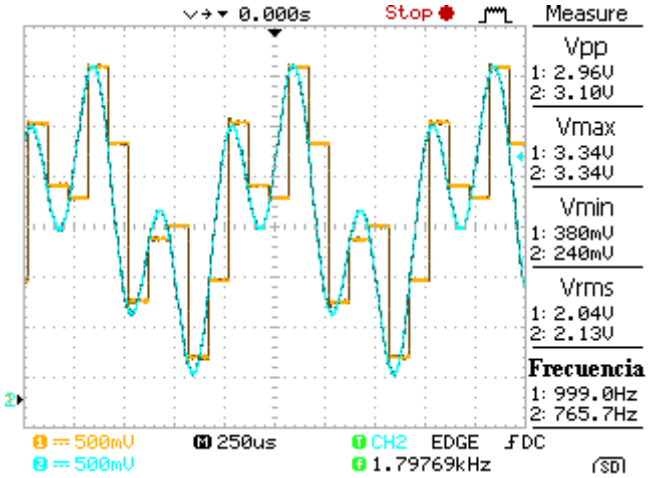

Fig. 10. Señal referencia y PAM (10 muestras por periodo).

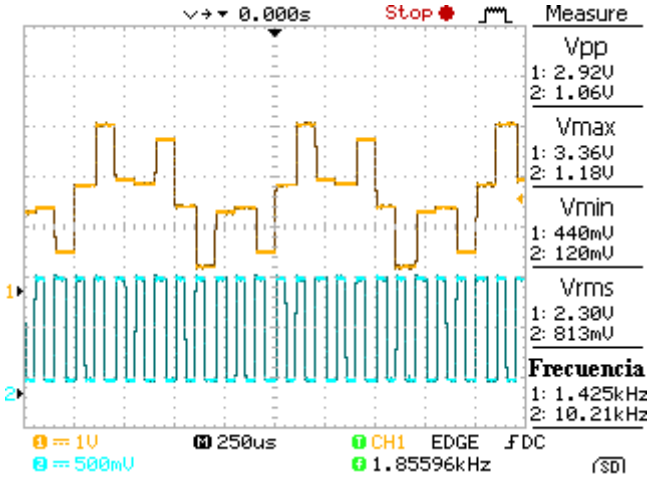

Fig. 11. Tren de pulsos y PAM (10 muestras por periodo).

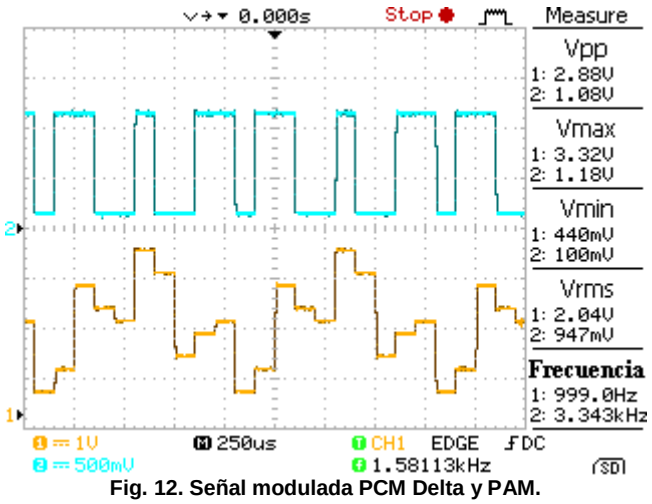

Fig. 12. Señal modulada PCM Delta y PAM.