ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES

"ANÁLISIS E IMPLEMENTACIÓN DE SISTEMAS DE MODULACIÓN DIGITAL ASK, FSK, M-PSK Y M-QAM MEDIANTE LA PROGRAMACIÓN DE CÓDIGO VHDL UTILIZANDO LA TECNOLOGÍA FPGA"

TESIS DE GRADO

PREVIO A LA OBTENCIÓN DEL TÍTULO DE:

INGENIERO EN ELECTRÓNICA, TELECOMUNICACIONES Y REDES

PRESENTADO POR:

GONZALO RAMIRO SÁNCHEZ VELOZ

RIOBAMBA – ECUADOR

2014

A toda mi familia que me han apoyado y han sido el pilar fundamental de mis primeros triunfos.

A mi tutor de tesis el Ing. Neiser Ortiz, al Ing. José Guerra que me han guiado y colaborado para la culminación de este proyecto.

A mis amigos que con su apoyo incondicional me han brindado sus consejos y ánimos para seguir adelante en la vida.

GONZALO

A LA MEMORIA DE MI PADRE.

GONZALO

FIRMAS RESPONSABLES Y NOTAS

NOMBRE	FIRMA	FECHA
Ing. Iván Menes **DECANO FACULTAD INFORMÁTICA Y ELECTRÓNICA**	_____	_____
Ing. Wilson Baldeón **DIR. ESC. ING. ELECTRÓNICA EN TELECOMUNICACIONES Y REDES**	_____	_____
Ing. Neiser Ortíz **DIRECTOR DE TESIS**	_____	_____
Ing. José Guerra **MIEMBRO DEL TRIBUNAL**	_____	_____
DIRECTOR CENTRO DE DOCUMENTACIÓN	_____	_____

NOTA DE LA TESIS

_____

RESPONSABILIDAD DEL AUTOR

Yo, Gonzalo Ramiro Sánchez Veloz, soy responsable de las ideas, doctrinas y resultados expuestos en esta Tesis y el patrimonio intelectual de la tesis de Grado pertenece a la ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO.

_____

Gonzalo Ramiro Sánchez Veloz

ÍNDICE DE ABREVIATURAS

ACRÓNIMO	SIGNIFICADO
SSI	Small Scale Integration (Integración a Pequeña Escala).
MSI	Medium Scale Integration (Integración a Mediana Escala).
LSI	Large Scale Integration (Integracion a Gran Escala).
FPGA	Field Programmable Gate Array (Arreglo de Compuertas de Campos Programables).
PLD	Dispositivos Lógicos Programables.
ASIC	Application Specific Integrated Circuit (Circuito Integrado para Aplicaciones específicas).
CPU	Central Processing Unit (Unidad Central de Procesamiento).
ASK	Amplitude Shift Keying (Modulación por Desplazamiento de Amplitud).
FSK	Frequency Shift Keying (Modulación por Desplazamiento de Frecuencia).
PSK	Phase Shift Keying (Modulación por Desplazamiento de Fase).
QAM	Quadrature Amplitude Modulation (Modulación de Amplitud en Cuadratura).
VHDL	Very High Speed Integrated Circuit Hardware Description Language (Lenguaje de Descripción de Hardware de Circuitos Integrados de Muy Alta Velocidad).
SPLD	Dispositivos Lógicos Programables Simples.
CPLD	Dispositivos Lógicos Programables Complejos.
PLA	Arreglo Lógico Programable.
LUT	Lookup Table.
SRAM	Static Random Access Memory (Memoria Estática de Acceso Aleatorio).
IOB	Bloques de Entrada y/o Salida.

CLB Bloques Lógicos Configurables.

ROM Read Only Memory (Memoria de Solo Lectura).

PROM Programmable Read Only Memory (Memoria Programable de solo Lectura).

EPROM Erasable Programmable Read Only Memory (ROM programable borrable).

EEPROM Electrically Erasable Programmable Read Only Memory (ROM programmable y borrable electricamente).

SDRAM Synchronous Dynamic Random Access Memory (Memoria sincrónica de acceso aleatorio).

CMOS Complementary Metal Oxide Semiconductor (Semiconductor Complementario de Oxido Metálico).

MIN Metal Insulator Metal (metal aislante metal).

OTP One Time Programmable (programables una sola vez).

CAD Computer Aided Design (Diseño Asistido por Computadora).

HDL Hardware Description Language (Lenguaje de Descripción de Hardware).

IEEE Institute of Electrical and Electronics Engineers (Instituto de Ingenieros Electrónicos y Eléctricos).

DSP Digital Signal Processor (Procesador Digital de Señales).

PET Tomografía por Emisión de Positrones.

CT Tomografía Computarizada.

ADC Analog to Digital Converter (Convertidor Analógico Digital).

DAC Digital to Analog Converter (Convertidor Digital Analógico).

TTL Transistor, Transistor Logic (Lógica Transistor a Transistor).

SoC System on a Chip (Sistema en un solo Chip).

TSMC TAIWAN SEMICONDUCTOR MANUFACTURING COMPANY.

IC	Circuito Integrado.
ISE	Integrated Software Enviroment (Entorno de Software Integrado).
EDA	Electronic Design Automation (Diseño Electrónico Automatizado).
GUI	Graphic User Interface (Interfaz Gráfica de Usuario).
RTL	Resistor Transistor Logic (Lógica de Resistencia Transistor).
ESD	Electrostatic Discharge (Descarga Electrostática).
DDR	Double Data Rate (Tasa de Datos Dobles).
JTAG	Join Test Action Group (Grupo de Acción Conjunta de Pruebas).
VGA	Video Graphics Array (Portadora Grafica de Video).
PWM	Pulse Width Modulation (Modulación por Densidad de Pulsos).
LCD	Liquit Crystal Display (Pantalla de Cristal Líquido).
DNA	Device Number Authentication (Numero de Autenticación del Dispositivo).
LVTTL	Low Voltage TTL (Bajo Voltaje TTL).
SMT	Surface Mount Technology (Tecnología de Montaje Superficial).
GPIO	General Purpose Input/Output (Entrada/Salida de Propósito General).
EDK	Embedded Develpment Kit (Kit de Desarrollo Integrado).
HDMI	High Definition Multimedia Interface (Interfaz Multimedia de Alta Definición).
DCM	Digital Clock Manager (Gestor digital de reloj).
PLL	Phase Locked Loop (Bloqueador de bucles de fase).
DLL	Delay Locked Loop (Bloqueador de bucles de retardo).
VHDC	Very High Density Cable (Cable Denso de Alta Velocidad).
OLED	Organic Light Emitting Diode (Diodo Orgánico de Emisión de Luz).
HID	Human Interface Device (Dispositivo de Interfaz Humana).

QSPI Quad Serial Peripheral Interface (Interfaz periférica Serial Quad).

NTSC National Television System Committee (Comisión Nacional de Sistema de Televisión).

PAL Phase Alternating Line (Línea de Fase Alternada).

IrDA Infrared Data Association (Asociación de Datos Infrarrojos).

API Application Programming Interfase (Interfaz de Programación de Aplicaciones).

HSMC High Speed Mezzanine Card (Conector de Borde de Placa de Alta Velocidad).

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Debido al auge tecnológico y a su constante desarrollo se opta por abaratar costos, tiempo y espacio al momento de implementar aplicaciones relacionadas a la electrónica digital. Por consiguiente se busca la tecnología adecuada para reemplazar de una manera óptima y sencilla a los grandes sistemas digitales que usan circuitos lógicos SSI, MSl y LSI, es aquí en donde nos adentramos en el mundo de las tarjetas programables FPGA (Field Programmable Gate Array) y así reemplazar dichos elementos, obteniendo una mayor versatilidad al momento de la implementación de cualquier tipo de sistemas convencional digital y así también nos permite analizar y simular los diseños implementados.

Un FPGA es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada mediante un lenguaje de descripción especializado. Su lógica programable nos permite configurar estructuras sencillas, tales como las funciones de compuertas lógicas como sistemas complejos.

Las FPGA son el resultado de la convergencia de dos tecnologías diferentes, los dispositivos lógicos programables (PLDs) y los circuitos integrados de aplicación específica (ASIC).

Los dispositivos lógicos programables (PLDs) son circuitos integrados digitales que no tienen una función predefinida por el fabricante. Su función puede ser definida (o programada) por el usuario. Permiten reemplazar grandes diseños digitales que antes se implementaban con componentes discretos (como compuertas y flip-flops) por un solo integrado. Debido a la gran capacidad lógica que tienen los dispositivos modernos, sistemas completos pueden desarrollarse sobre un solo circuito integrado.

Los dispositivos actuales (FPGAs) tienen una capacidad lógica de hasta millones de compuertas, incluyen interfaces programables para varios estándares de interfaces eléctrica y tienen bloques de funciones especiales embebidos entre la lógica programable tales como memoria, multiplicadores o CPUs completas. Esta gran capacidad y variedad de los dispositivos los hace sumamente útiles a la hora de crear prototipos, desarrollo rápido de nuevos productos, para los productos que deben ser reconfigurables por naturaleza o productos que se producen en bajos volúmenes y para los cuales no es económicamente viable crear un integrado a medida.

Por lo antes mencionado surge la necesidad de buscar nuevas soluciones en hardware e incrementar la competencia de los profesionales de la FACULTAD DE

INFORMÁTICA Y ELECTRÓNICA y en especial a los alumnos de la carrera de INGENIERÍA EN ELECTRÓNICA, TELECOMUNICACIONES Y REDES de la ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO en esta área del conocimiento. Razón por la cual se desarrolla este proyecto que tiene como objetivo diseñar e implementar los diversos tipos de modulaciones digitales y así poder analizar de una manera adecuada y funcional dichas modulaciones, utilizando la tecnología FPGA y demás dar a conocer la potencialidad de esta tecnología.

CAPÍTULO I

MARCO REFERENCIAL

1.1 INTRODUCCIÓN

En este primer capítulo se describe la situación actual de la tecnología y los avances tecnológicos relacionados en electrónica digital, también se describen los objetivos planteados en la tesis, además se presentan los motivos por los que se ha desarrollado el presente proyecto de grado.

1.2 ANTECEDENTES

Debido al auge tecnológico y a su constante desarrollo se opta por abaratar costos, tiempo y espacio al momento de implementar aplicaciones relacionadas a la electrónica digital. Por consiguiente se busca la tecnología adecuada para reemplazar de una manera óptima y sencilla a los grandes sistemas digitales que usan circuitos lógicos SSI, MSl y LSI, es aquí en donde nos adentramos en el mundo de las tarjetas programables FPGA (Field Programmable Gate Array) y así reemplazar dichos elementos, obteniendo una mayor versatilidad al momento de la implementación de cualquier tipo de sistemas convencional digital y así también nos permite analizar y simular los diseños implementados.

Un FPGA es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada mediante un lenguaje de descripción especializado. Su lógica programable nos permite configurar estructuras sencillas, tales como las funciones de compuertas lógicas como sistemas complejos.
Las FPGA son el resultado de la convergencia de dos tecnologías diferentes, los dispositivos lógicos programables (PLDs) y los circuitos integrados de aplicación específica (ASIC).

Los dispositivos lógicos programables (PLDs) son circuitos integrados digitales que no tienen una función predefinida por el fabricante. Su función puede ser definida (o programada) por el usuario. Permiten reemplazar grandes diseños digitales que antes se implementaban con componentes discretos (como compuertas y flip-flops) por un solo integrado. Debido a la gran capacidad lógica que tienen los dispositivos modernos, sistemas completos pueden desarrollarse sobre un solo circuito integrado.

Los dispositivos actuales, FPGAs tienen una capacidad lógica de hasta millones de compuertas, incluyen interfaces programables para varios estándares de interfaces eléctrica y tienen bloques de funciones especiales embebidos entre la lógica programable tales como memoria, multiplicadores o CPUs completas. Esta gran capacidad y variedad de los dispositivos los hace sumamente útiles a la hora de crear prototipos, desarrollo rápido de nuevos productos, para los productos que deben ser reconfigurables por naturaleza o productos que se producen en bajos volúmenes y para los cuales no es económicamente viable crear un integrado a medida.

Los FPGA por ser reprogramables ofrecen una gran flexibilidad de diseño y una disminución significativa de los costos de implementación, ya que en el Ecuador existe una falta de desarrollo tecnológico para sistemas de Telecomunicaciones debido al alto costo de inversión para el desarrollo de circuitos integrados para estos fines, por tal razón todas las soluciones tecnológicas son cerradas y a pesar de todos los conocimientos en electrónica digital, resulta difícil llegar a su implementación.

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

Analizar e implementar los sistemas de modulación digital ASK, FSK, M-PSK y M-QAM mediante la programación en código VHDL utilizando la tecnología FPGA.

1.3.2 OBJETIVO ESPECIFICO

- Investigar la potencialidad y el funcionamiento de las tarjetas FPGA.
- Investigar los dos proveedores más sobresalientes en tecnología FPGA y determinar la herramienta de desarrollo más adecuada para el diseño, haciendo referencia a Kits Educaciones.
- Analizar, programar y simular los diferentes tipos de modulación digital que se utilizan en las transmisiones digitales y en procesamiento de señales.
- Implementar en diseño y visualizar los resultados obtenidos de las diferentes modulaciones digitales programadas en la tarjeta FPGA.

1.4 MARCO HIPOTÉTICO

1.4.1 HIPÓTESIS

Con la tecnología FPGA se desarrollara e implementara los sistemas de modulación digital y así se analizara dichas señales y además determinara la potencialidad de las tarjetas FPGA en una de sus diversas aplicaciones de manera, fácil, sencilla y acorde a la actualidad.

CAPITULO II

POTENCIALIDADES Y FUNCIONAMIENTO DE LA TECNOLOGÍA FPGA

2.1 INTRODUCCIÓN

En este capítulo se conoce de los dispositivos lógicos programables y su evolución hasta los Field Programmable Gate Arrays (FPGA) que es nuestro caso de estudio, analizando su arquitectura, tecnología, sus maneras de programación e implementación y las áreas en donde se puede aplicar esta tecnología.

2.2 INTRODUCCIÓN A LAS FPGAs

Los dispositivos FPGA son un conjunto de circuitos integrados digitales que contienen bloques lógicos reconfigurables con interconexiones entre ellos. Su nombre referencial puede ser “campo programable” porque su programación se da “en campo”, a diferencia de los dispositivos en los cuales las funciones internas están fuertemente “amarradas” por el fabricante. Esto quiere decir que los FPGAs pueden ser configurados tanto en el laboratorio como también posteriormente en sus ubicaciones definitivas de operación.

Un FPGA es un dispositivo multinivel programable de propósito general. Integra una gran cantidad de dispositivos lógicos programables en un chip. El tamaño y velocidad de los FPGA es comparable a los circuitos integrados para aplicaciones específicas (ASICs), pero los FPGA son más flexibles, su ciclo de diseño es más corto y los recursos necesarios generan menores gastos. La adopción de chips FPGA en la industria ha sido impulsada por le echo de que los FPGAs combinan lo mejor de los ASICs y de los sistemas basados en procesadores, los FPGA llevan a cabo diferentes operaciones de manera paralela, por lo que estas no necesitan competir por los mismos recursos.

A comparación de los dispositivos ASIC (Circuitos Integrados para Aplicaciones Específicas) los FPGA dan un mayor campo de oportunidades a los diseñadores para la creación de proyectos ya que integra una gran cantidad de dispositivos lógicos programables en un chip, teniendo la ventaja de ser reprogramables, son más flexibles, su ciclo de diseño es más corto y los recursos necesarios generan menores gastos, mientras que los ASIC no lo son, aunque éstos últimos son más veloces y consumen menos energía que los FPGA.

En el nivel más alto, los FPGAs son chips de silicio reprogramables, son completamente reconfigúrales y al instante toma una nueva ‘’ personalidad ‘’ cuando se compila un distinto diseño de circuito. Anteriormente solo los Ingenieros con un amplio y profundo conocimiento en el diseño de hardware digital podía trabajar con la tecnología FPGA, el avance de la tecnología ha dado cavidad al desarrollo de herramientas de diseño de alto nivel cambiando de esta manera las reglas de programación de los FPGAs, las nuevas tecnologías de diseño convierten los diagramas a bloques gráficos, o hasta el código ANSI C a circuito de hardware digital.

2.3 CONCEPTOS FUNDAMENTALES Y ORÍGENES DE LA TECNOLOGÍA

La invención de los Circuitos Integrados ha dado cavidad al desarrollo de nuevos chips y entre ellos los dispositivos lógicos programables. Un PLD es un chip de uso general para implementar circuitos lógicos. Incluye un conjunto de elementos de circuito lógico que pueden adaptarse de diferentes formas. Un PLD puede considerarse una ‘‘caja negra’’ que contiene compuertas lógicas e interruptores programables, como se ilustra en la figura II-1. Estos últimos permiten que las compuertas lógicas en el interior del PLD se conecten juntas para implementar el circuito lógico que se necesite.

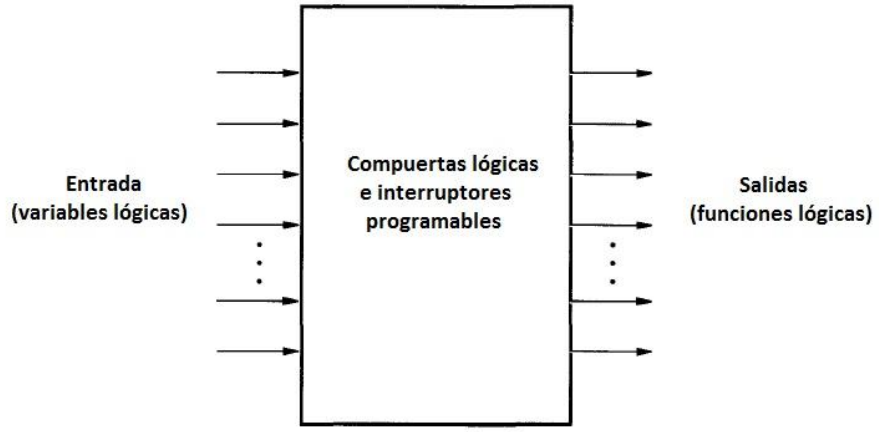


Figura II-1: Dispositivo lógico programable como caja negra

Hay diferentes tipos de PLD comerciales como: los SPLD (Dispositivos Lógicos Programables Simples), los CPLD (Dispositivos Lógicos Comerciales Complejos) y los actuales conocidos como los FPGA.

El primero en desarrollarse de la familia de los SPLDs fue el PLA (arreglo lógico programable), cuya estructura general se presenta en la figura II-2. Con base en la idea de que las funciones lógicas se pueden realizar en forma de suma de productos, un PLA comprende un juego de compuertas AND que alimenta un conjunto de compuertas OR. Como se muestra en la figura, las entradas de PLA, X1,…, Xn pasa por un grupo de buffers (que proporcionan tanto el valor verdadero como el complemento de cada entrada) hacia un bloque de circuito llamado plano AND, o arreglo AND. El plano AND produce un juego de términos producto P1,…, Pk, cada uno de los cuales puede configurarse para implementar cualquier función AND de X1,…, Xn. Los términos producto sirven como entradas a un plano OR, que produce las salidas f1,…, fm. Cada salida puede configurarse para realizar cualquier suma de P1,…, Pk y, por ende, cualquier función de suma de productos de las entradas al PLA.

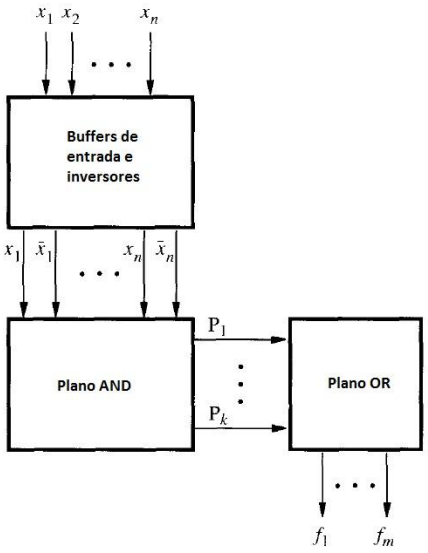


Figura II-2: Estructura general de un PLA

Un diagrama más detallado de un pequeño PLA se muestra en la figura II-3, tiene tres entradas, cuatro términos producto y dos salidas. Cada compuerta AND en el plano AND posee seis entradas, que corresponden a las versiones verdadera y complementada de las tres señales de entrada. Cada conexión a una compuerta AND en programable; la señal que conecta a una compuerta AND se indica con una línea ondulada, y la que no se conecta, con una línea quebrada. Los circuitos se diseñan de tal modo que cualesquiera entradas no conectadas de la compuerta AND no afecten la salida de la compuerta AND.

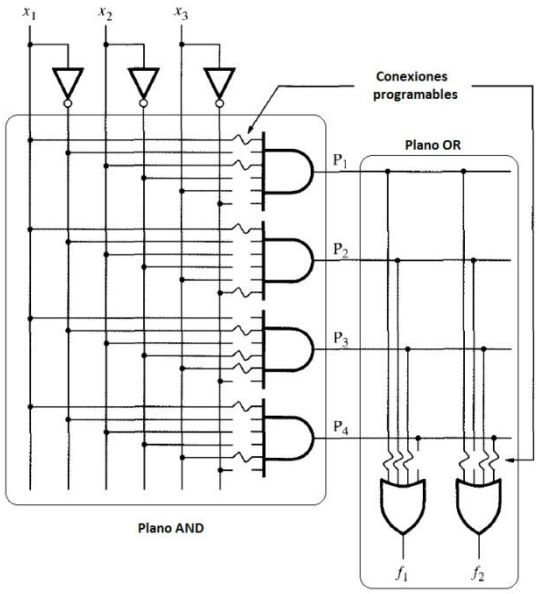


Figura II-3: Diagrama de nivel de compuerta de un PLA

Los PLA disponibles en el comercio viene en tamaños más grandes, los parámetros típicos son 16 entradas, 32 términos producto y ocho salidas.
El PLA es eficiente en términos del área necesaria para implementarlo en un chip de circuito integrado. Por ello con frecuencia se incluyen PLA como parte de chips más grandes, como los microprocesadores.

En un PLA los planos AND y OR son programables. Históricamente, los interruptores programables planteaban dos dificultades a sus fabricantes: eran difíciles de fabricar correctamente y reducían la velocidad de rendimiento de los circuitos implementados en los PLA. Tales desventajas llevaron al desarrollo de un dispositivo similar en el que el plano AND es programable pero el plano OR es fijo. Este chip se conoce como PAL (Dispositivo de lógica de arreglo programable), puesto que son más simples de fabricar y por tanto menos costosos que los que los PLA, aparte de ofrecer mejor rendimiento, las PAL se han vuelto populares en la aplicaciones prácticas. En la figura II-4 se representa un ejemplo de PLA con tres entradas, cuatro términos producto y dos salidas.

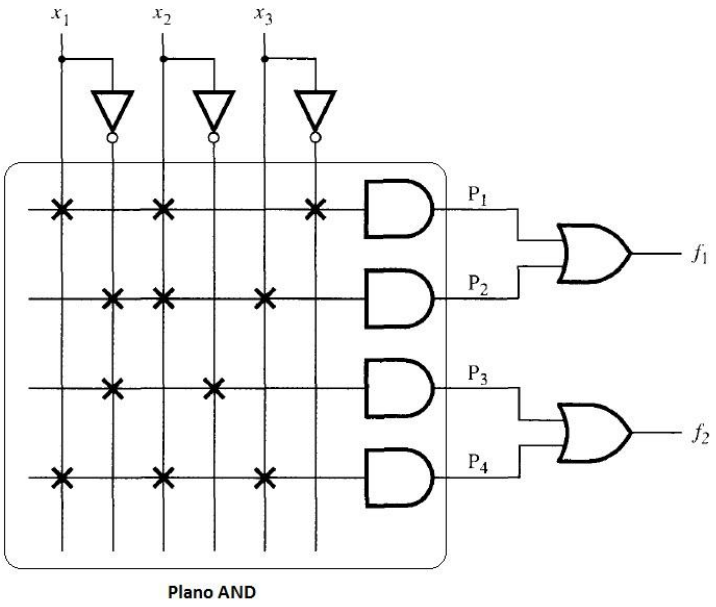


Figura II-4: Ejemplo de una PAL

En la figura se bosqueja como una sola línea horizontal unida a un símbolo de compuerta AND. Las posibles entradas a la compuerta AND se dibujan como líneas verticales que cortan la línea horizontal. En cualquier cruce de una línea vertical y horizontal puede hacerse una conexión programable, la cual se indica con una X.

Para compensar su menor flexibilidad, las PAL se fabrican de diversos tamaños, con varios números de entradas y salidas, y diferentes números de entradas a las compuertas OR.

Los SPLDs son útiles para implementar una amplia variedad de pequeños circuitos digitales. Tanto los PLA y los PAL pueden usarse para implementar circuitos que no requieren más que el número de entradas, términos producto y salidas que se ofrecen en el chip específico. Estos chips están limitados a tamaños muy modestos, y soportan una cantidad de entradas más salidas que en combinación no exceden de 32.
Para la implementación de circuitos que precisan más entradas y salidas se pueden emplear múltiples PLA o PAL o bien un tipo más moderno de chip, llamado CPLD.
Un CPLD comprende múltiples bloques de circuitos en un solo chip, con recursos de cableado interno para conectarlos. Cada bloque de circuito es similar a un PLA o a una PAL; nos referimos a ellos como bloques parecidos a PAL. En la figura II-5 se representa un ejemplo de CPLD.

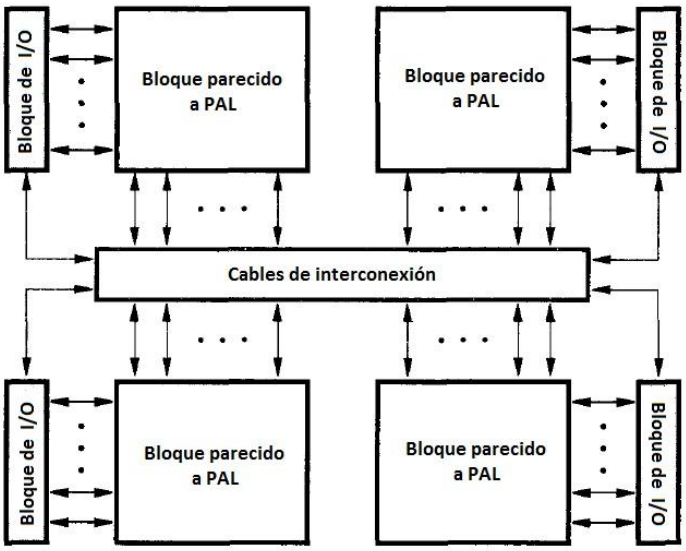


Figura II-5: Estructura de un dispositivo lógico programable complejo (CPLD)

En la figura II-5 se incluyen cuatro bloques parecidos a PAL que se conectan a un conjunto de cales de interconexión. Cada bloque parecido a PAL también está conectado a un subcircuito etiquetado bloque de entrada/salida (I/O, input/output) que a su vez está unido a varios de los pines de entrada y salida del chip.
Los cables de interconexión contienen interruptores programables que se usan para conectar los bloques parecidos a PAL. Cada uno de los cables horizontales puede conectarse a alguno de los verticales que cruza.

Los SPLD y CPLD son útiles para implementar una amplia variedad de circuitos lógicos. Excepto por el CPLD, estos dispositivos son más bien pequeños y adecuados solo para aplicaciones hasta cierto punto simples. Incluso para los CPLD, nada más pueden acomodarse circuitos lógicos moderadamente grandes en un solo chip. Para implementar circuitos mayores conviene usar un tipo de chip con una capacidad lógica mayor. Un Arreglo de Compuertas de Campos Programables es un dispositivo lógico programable que soporta la implementación de circuitos lógicos hasta cierto punto grandes.

A mediados de los 80s fueron introducidas la FPGA, las cuales se diferencian de los CPLD en su arquitectura, tecnología y costos. Estos dispositivos fueron creados principalmente para la implementación de circuitos de alto rendimiento.

Los FPGA son muy diferentes de los SPLD y de los CPLD, ya que no contienen planos AND y OR. En vez de ello, los FPGA ofrecen bloques lógicos para la implementación de las funciones requeridas. La estructura general de un FPGA se ilustra en la figura II-6. Contiene tres tipos principales de recursos: bloques lógicos, bloques de I/O para conectar a los pines del paquete, y cables de interconexión e interruptores.

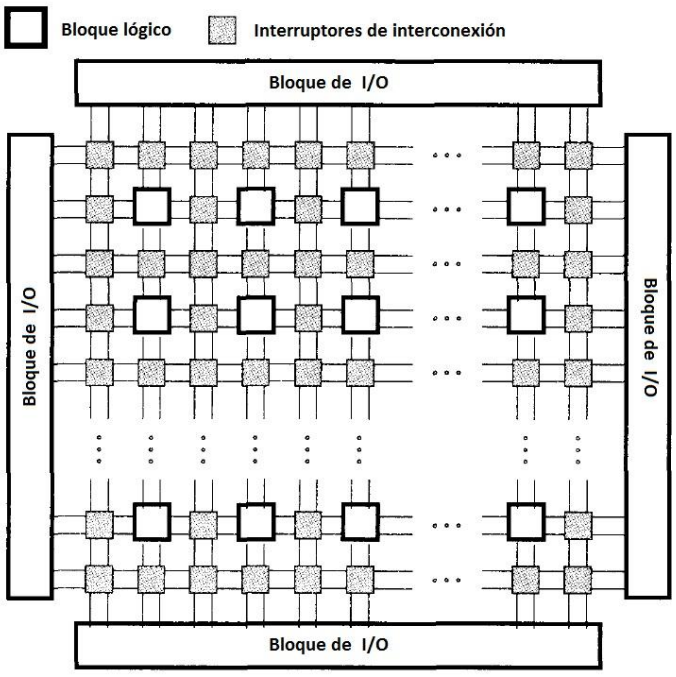


Figura II-6: Estructura general de un FPGA

Los bloques lógicos están dispuestos en un arreglo bidimensional, en tanto que los cables de interconexión están organizados como canales de enrutamiento horizontales

y verticales entre filas y columnas de bloques lógicos. Los canales de enrutamiento contienen cables e interruptores programables que permiten que los bloques se interconecten de muchas formas.

Los FPGA son dispositivos de ciento de miles, e incluso millones de compuertas lógicas, son dispositivos muy flexibles que pueden trabajar a altas frecuencias y con capacidad de procesamiento en paralelo. Además presentan recursos especiales para implementar de forma eficiente funciones aritméticas (comparadores, sumadores, contadores, etc.), mientas que los CPLD carecen de estos.

El núcleo es la parte del equipo, de la CPU (unidad central de proceso), que lee y ejecuta las instrucciones. Originalmente los ordenadores se desarrollaron con un único procesador por núcleo único, pero actualmente los procesadores con dos, cuatro, o incluso dieciséis núcleos son comunes.

Sin embargo, el Dr Win Vanderbauwhede y sus colegas de la Universidad de Massachusetts Lowell (EE.UU) han creado un procesador que contiene más de un millar de núcleos en un solo chip. Para ello los científicos utilizaron un chip llamado FPGA, que como todos los microchips contienen millones de transistores. El Dr. Vanderbauwhede ha dividido los transistores dentro del chip en pequeños grupos y programar cada uno para realizar una tarea diferente. Mediante la creación de más de 1000 mini-circuitos en el chip FPGA, los investigadores han convertido de hecho el chip en un procesador 100 núcleos, con cada núcleo de trabajo dotado de sus propias instrucciones.

2.4 ARQUITECTURA DE LOS DISPOSITIVOS FPGA

Como visualizamos anteriormente en la figura II-6 la estructura interna de un FPGA consiste en arreglos de bloques lógicos que se comunican entre sí a través de canales de conexión verticales y horizontales.

2.4.1 BLOQUES LÓGICOS CONFIGURABLES

Cada bloque lógico en un FPGA tiene un pequeño número de entradas y salidas, estos bloques lógicos son capaces de implementar funciones lógicas, tanto combinacionales como secuenciales.

En el marcado hay varios productos FPGA, que presentan diferentes tipos de bloques lógicos. El más usado de estos es una tabla de consulta (LUT, lookup table), que es una memoria SRAM, que contiene celdas de almacenamiento que sirven para implementar una pequeña función lógica, las funciones lógicas se precalculan y se almacenan en la memoria de la LUT Cada celda puede contener un solo valor lógico, 0 o 1. El valor almacenado se produce como la salida de la celda de almacenamiento. En la figura II-7 se muestra la estructura de una pequeña LUT.

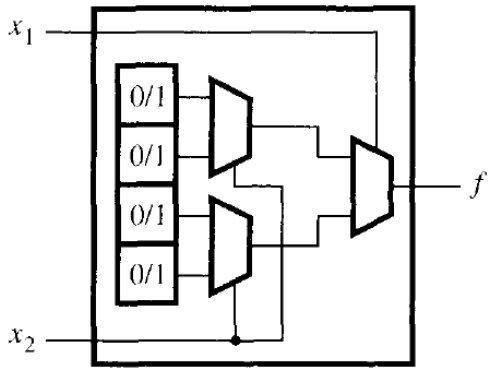


Figura II-7: Circuito para una LUT de dos entradas

Tiene dos entradas, X1 y X2, y una salida, f. Es capaz de implementar cualquier función lógica de dos variables. Como una tabla de verdad de dos variables tiene cuatro filas, esta LUT posee cuatro celdas de almacenamiento. Una celda corresponde al valor de salida de cada fila de la tabla de verdad. Las variables de entrada X1 y X2 se usan como las entradas de selección de tres multiplexores, los cuales, según la valoración de X1 y X2, eligen el contenido de una de las cuatro celdas como la salida de la LUT.

Los FPGA pueden tener circuitos extras, además de las LUTs, en cada bloque lógico. Por ejemplo, se puede incluir flip-flops dentro de un bloque lógico como se muestra en la figura II-8. La estructura en sí de cada bloque varía de acuerdo al fabricante.

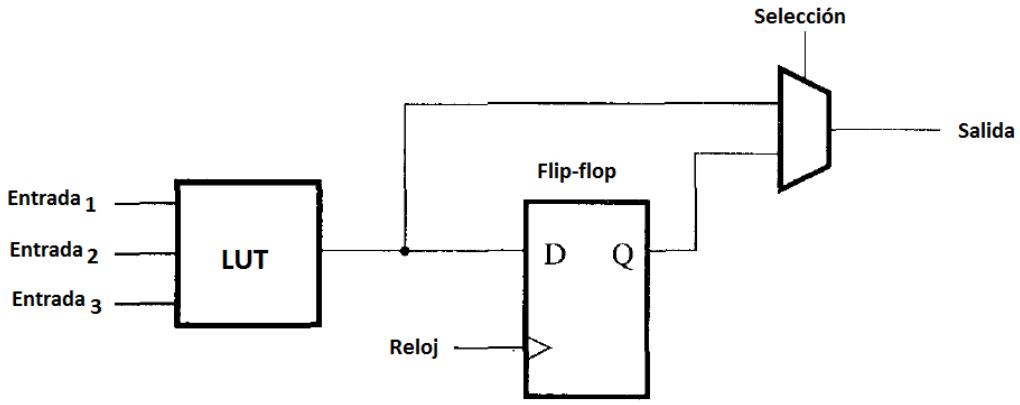


Figura II-8: Bloque CLB

2.4.2 BLOQUES DE I/O PARA CONECTAR LOS PINES DEL PAQUETE

Estos bloques de entrada y/o salida (IOBs) tienen la función de permitir el paso de una señal hacia el interior o hacia el exterior del dispositivo. Cada uno de los bloques disponibles puede ser configurado independientemente para funcionar como entrada, salida o línea bidireccional. En la figura II-9 se muestra un esquema de un IOB.

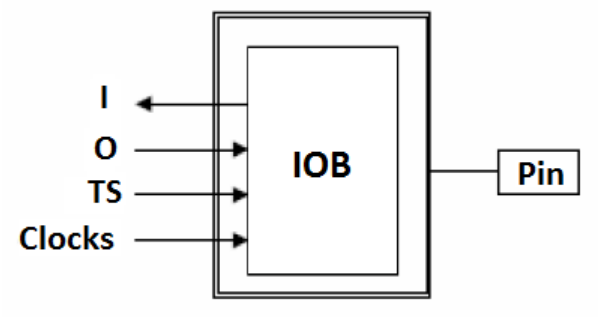


Figura II-9: Bloque lógico de entrada y/o salida

.

Dependiendo del fabricante estos bloques pueden ser considerados o no parte del bloque lógico, y se pueden implementar con flip-flops, latches, o circuitos combinacionales y en algunos casos la salida que presenta es triestado (TS).

2.4.3 CABLES DE INTERCONEXIÓN E INTERRUPTORES

Los cables de interconexión están organizados como canales de enrutamiento horizontales y verticales entre filas y columnas de los bloques lógicos como se muestra en la figura II-10. Los canales de enrutamiento contienen cables e interruptores programables que permiten que los bloques lógicos se interconecten de

muchas formas, además de interconectar a los bloques lógicos entre si también se conectan con los bloques de I/O.

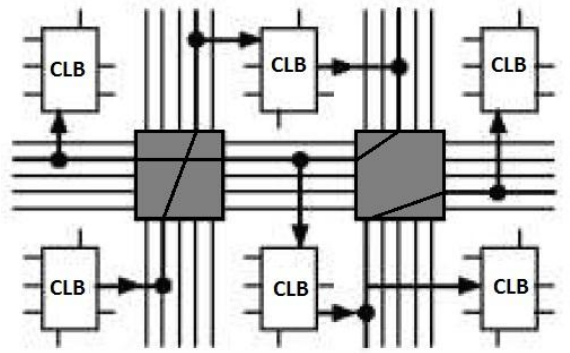


Figura II-10: Interruptores de interconexión

2.5 TECNOLOGÍAS DE PROGRAMACIÓN DE LOS FPGAS

Cada FPGA se basa en una tecnología de programación diferente, pero esta depende de su arquitectura y del fabricante. Las tecnologías de programación que han sido utilizadas incluyen EPROM, EEPROM, Flash, SRAM y anti-fusibles. De estos métodos solo el Flash, memoria estática y los enfoques anti-fusibles se utilizan ampliamente en FPGAs modernas.

2.5.1 TECNOLOGÍA DE CELDAS DE MEMORIA ESTÁTICA

Las celdas de memoria estática son la base para la tecnología de programación SRAM, es una pequeña celda que se usa para mantener la configuración de cada parte configurable del FPGA. La tecnología de programación SRAM se ha convertido en el método dominante para los FPGA debido a sus dos ventajas principales: son reprogramables y el uso de la tecnología de proceso CMOS estándar, de esta manera utilizan la última tecnología CMOS disponible y por lo tanto beneficiarse de la mayor integración, las ultimas velocidades más altas y el consumo de potencia dinámica inferior de nuevos procesos con geometrías mínimas menores.

Debido a la enorme cantidad de memorias SRAM que se producen para el mercado digital, permite lograr costos de producción muy bajos, muy alta performance y trabajar con un proceso de fabricación muy amortizado y de gran rendimiento.

Las celdas de memoria tipo SRAM pueden ser reprogramadas un sinnúmero de veces, del mismo modo un FPGA basado en celdas de configuración SRAM puede ser reprogramado un infinito número de veces, aun cuando el FPGA ya este montado y soldado en un circuito impreso.

La gran desventaja de las celdas SRAM es que por ser volátiles deben ser reprogramadas cada vez que se reinicie el sistema. El contenido de estas celdas se pierde cuando se deja de suministrar energía. Para ello es necesario el uso especial de una memoria externa no volátil, que mantiene el archivo de configuración del FPGA. Otra desventaja es que, debido a que la selección del camino de conexión entre los diferentes bloques lógicos, se basan en celdas SRAM, se provocan grandes retardos de ruteo, lo que es un problema en diseños que requieren un rendimiento muy alto.

Para algunas aplicaciones, como sistemas médicos de emergencia, este tiempo de lectura de la memoria y configuración es muy largo (50 – 500ms). Para otras aplicaciones, como interfaces series, este tiempo pasa desapercibido.

2.5.2 TECNOLOGÍA FLASH/EEPROM

Una alternativa que responde a algunas de las deficiencias de la tecnología basada en SRAM es el uso de tecnología de programación de puerta flotante que inyecta carga en una puerta que ‘‘flota’’ por encima del transistor. Este enfoque es utilizado en celdas de memoria Flash o EEPROM.

Esta tecnología de programación basada en Flash ofrece varias únicas ventajas, las más importante es su no volatilidad. Esta característica elimina el uso de una memoria externa que carguen las configuraciones de programación a diferencia de la tecnología de programación SRAM, esto hace que un dispositivo basado en flash puede funcionar después de haber sido encendido sin tener que esperar la carga de los datos. Su tamaño es más reducido que el de una celda SRAM, aunque sin llegar al tamaño reducido de una anti-fusible y además son reprogramables.

Una desventaja de los FPGA basados en Flash es que no pueden ser reprogramados infinitamente. Además, la mayoría de dispositivos basados en flash son configurados para una función específica, en este caso un FPGA con tecnología Flash es más que

suficiente y requiere de una tecnología de proceso CMOS no estándar, también la velocidad de programación es bastante más lenta que en el caso de una SRAM.

Una tendencia que ha surgido es el uso de la combinación de la memoria flash con la tecnología de programación SRAM. Marcas reconocidas como Xilinx y Altera utilizan esta combinación, la memoria flash en el chip se utiliza para proporcionar almacenamiento no volátil, mientras que las celdas SRAM se utilizan para controlar los elementos programables en el diseño. Esto equilibra los problemas con la volatilidad de los FPGA basados en celdas SRAM, tales como el costo de los dispositivos de almacenamiento adicionales, mientras se mantiene la capacidad de reconfiguración de un infinito número de veces de los dispositivos basados en SRAM.

2.5.3 TECNOLOGÍA ANTI-FUSIBLE

Otro tipo de FPGA usa una celda llamada anti-fusible como celda básica de configuración. Esta celda consiste en una estructura microscópica que, a diferencia de un fusible regular, está normalmente abierta. Está compuesta de tres capas, la conductora arriba y abajo, y la aisladora en el medio (Metal-Insulator-Metal, MIM). Para configurar el dispositivo se hace circular una cierta cantidad de corriente (~5mA), lo que causa una gran potencia de disipación en un área muy pequeña, provocando el derretimiento del aislante dieléctrico entre los dos electrodos conductores formando una unión permanente muy fina. Esta característica hace que los FPGA anti-fusible no sean volátiles, la cual es la principal ventaja. Además su tamaño es muy reducido, su no volatilidad hace que el dispositivo funcione al instante una vez encendido, esto reduce los costes del sistema ya que no se requiere de una memoria adicional para el almacenamiento de la información de programación. También, debido a esta tecnología los retardos de conexión entre los bloques lógicos son muy reducidos, por lo que el rendimiento de estos dispositivos es bastante elevado, esto permite que el FPGA pueda ser utilizado en situaciones que requieran la operación inmediatamente después de encender el aparato

También hay desventajas respecto a esta tecnología de programación la más considerable es que solo se puede programar una sola vez, lo que es conocido técnicamente como One Time Programmable (OTP). El hecho de que estos FPGA sean OTP crea un proceso de verificación muy meticuloso de la lógica a ser programada, a fin de no tener que descartar este dispositivo por errores de diseño, por lo tanto el rendimiento después de la programación será menor que el rendimiento del

100% de SRAM o de los dispositivos de puerta flotante, ya que algunos fallos solo se descubren después de la programación.

Otra desventaja de los FPGAs anti-fusible es que requieren un proceso CMOS no estándar, que son típicamente muy por detrás en los procesos de fabricación que se pueden adoptar en comparación con los FPGAs basados en SRAM.

A continuación en la Tabla II-1 se muestra la comparación de las tres tecnologías de programación utilizadas actualmente de un FPGA.

	SRAM	**Flash**	**Anti-Fusible**
Volatilidad	Si	No	No
Reprogramable	Si	Si	No
Tamaño del elemento	Grande	Mediano	Pequeño
Proceso de fabricación	Estándar CMOS	CMOS no estándar	CMOS no estándar

Tab.II-I Comparación de las tecnologías de programación de los FPGA

2.6 FORMAS DE DESARROLLO E IMPLEMENTACIÓN

El punto de partida en el proceso de diseñar un circuito lógico es la concepción de lo que se supone debe hacer este y el planteamiento de su estructura general. Esto lo efectúa manualmente el diseñador, pues se requiere experiencia de diseño e intuición. El resto del proceso de diseño se realiza con el auxilio de las herramientas CAD. La primera etapa de este proceso supone ingresar en el sistema CAD una descripción del circuito que se va a diseñar. Esta etapa se denomina ‘‘ingreso de diseño’’. Describiremos dos métodos de ingreso de diseño: el uso de lenguaje esquemático o captura esquemática y la escritura de código fuente en un lenguaje de descripción de hardware.

2.6.1 LENGUAJE ESQUEMÁTICO

Un circuito lógico puede describirse dibujando las compuertas lógicas e interconectándolas con cables. Las herramientas CAD para el ingresar el diseño de un circuito de esta manera se llama herramienta de captura esquemática. La palabra esquemática se refiere al diagrama de un circuito en el que los elementos de éste,

como las compuertas lógicas, se muestras como símbolos gráficos y las conexiones entre tales elementos se indican con líneas.

Una herramienta de captura esquemática usa las funciones graficas de una computadora; por su parte, el ratón permite al usuario trazar un diagrama esquemático. Para facilitar la inclusión de compuertas en el esquema la herramienta ofrece un juego de símbolos gráficos que representan compuertas de varios tipos con diferentes números de entradas. Este juego de símbolos se llama biblioteca. Las compuertas de la biblioteca pueden importarse al esquema del usuario, y la herramienta brinda una forma gráfica de interconectarlas para crear un circuito lógico.

Es posible representar cualesquiera subcircuitos como símbolos gráficos e incluirse en el esquema. En la práctica es común que el usuario de un sistema CAD cree un circuito que comprenda otros circuitos más pequeños. Este método se conoce como diseño jerárquico y provee una buena forma de manejar la complejidad propia de los circuitos grandes. Aunque el lenguaje esquemático es simple de usar, se vuelve engorrosa cuando enfrenta circuitos grandes.

2.6.2 LENGUAJE DE DESCRIPCIÓN DE HARDWARE (HDL)

Un lenguaje de descripción de hardware (HDL) es similar a un lenguaje de programación típico, salvo que el HDL sirve para describir hardware en lugar de un programa que la computadora ejecutara. Hay muchos HDL comerciales. Algunos son sujetos a un derecho de propiedad, lo que significa que los ofrece una compañía y solo pueden usarse para implementar circuitos en la tecnología que esa compañía ofrece.

El IEEE es un organismo mundial que promueve actividades técnicas para el beneficio de la sociedad. Una de ellas supone el desarrollo de normas que definan cómo usar ciertos conceptos tecnológicos de modo adecuado para un gran grupo de usuarios. Dos HDL son normas del IEEE: VHDL (Very High Speed Integrated Circuit Hardware Description Language), lenguaje de descripción de hardware de circuitos integrados de muy alta velocidad y Verilog VHDL. Ambos lenguajes tienen amplio uso en la industria.

En comparación con realizar captura esquemática, el uso de VHDL da varias ventajas. Puesto que lo apoyan la mayor parte de los organismos de ofrecen tecnología de hardware digital, VHDL brinda portabilidad de diseño. Un circuito especificado en VHDL puede implementarse en diferentes tipos de chips y con herramientas CAD

ofrecidas por diferentes compañías, sin necesidad de cambiar la especificación de VHDL. La portabilidad de diseño es una ventaja importante porque la tecnología de circuitos digitales cambia con rapidez. Al utilizar un lenguaje estándar el diseñador puede centrarse en la funcionalidad del circuito deseado sin preocuparse mucho por los detalles de la tecnología que a la postre usara para la implementación.

El ingreso de diseño de un circuito lógico se efectúa mediante la escritura de código VHDL. Las señales de circuito pueden representarse como variables en el código fuente, y las funciones lógicas se expresan mediante la asignación de valores a dichas variables. El código fuente de VHDL es texto llano, lo que facilita al diseñador incluir en él la documentación que explique cómo funciona el circuito. Esta característica, aunada al hecho de que VHDL se usa ampliamente, alienta a compartir y reutilizar los circuitos descritos en VHDL. Esto permite el desarrollo más rápido de productos nuevos en casos donde el código VHDL existente puede adaptarse para diseñar circuitos nuevos.

Similar al modo en que los circuitos grandes se manejan en la captura esquemática, el código VHDL puede escribirse en forma modular que facilite el diseño jerárquico. El diseño de circuitos lógicos pequeños y grandes pueden representarse e eficientemente en código VHDL. Este lenguaje se usa para definir circuitos como microprocesadores con millones de transistores.

El ingreso de diseño en VHDL puede combinarse con otros métodos. Por ejemplo, es posible usar una herramienta de captura esquemática en la que un subcircuito del esquema se describa con VHDL.

2.7 HERRAMIENTAS PARA LA PROGRAMACIÓN DE FPGAS

En la actualidad para el desarrollo de sistemas combinacionales complejos se requiere de un software que asista al diseñador. Para diseñar un circuito lógico se requiere varias herramientas CAD. Casi siempre están empaquetadas en un sistema CAD.

Los PLD del tipo FPGA son un buen ejemplo de la complejidad que se puede alcanzar, esta complejidad no sería alcanzable sin la ayuda de un entorno con herramientas que asistan en el proceso de diseño, simulación, síntesis del resultado y configuración del hardware.

Existen varias herramientas de los distintos fabricantes de tecnología FPGA, dependiendo del fabricante se pueden encontrar estas herramientas gratuitamente o pagadas.

Para verificar la lógica creada por un programador de FPGA, es común escribir un banco de código en HDL o en Lenguaje Esquemático para abarcar y practicar el diseño de FPGA al afirmar entradas y verificar salidas. El proceso de diseño ya sea en HDL o en lenguaje esquemático se ejecuta en un entorno de simulación que modela el comportamiento de temporización de hardware del chip FPGA y muestra al diseñador todas las señales de entrada y salida para validación de pruebas. El proceso de diseñar el circuito y ejecutar la simulación generalmente requiere de tiempo.

Una vez que ha creado un diseño de FPGA y lo ha verificado, necesita integrarlo en una herramienta de compilación que toma la lógica basada en texto y después de varios pasos bastantes complejos, sintetiza su diseño en un archivo de configuración o escritura de bits que contiene información sobre como los componentes deben estar cableados. Como parte de este proceso manual de varios pasos, usted generalmente requiere especificar un mapeo de los nombres de señales hasta los pines en el chip FPGA que está usando.

2.8 APLICACIONES GENERALES

El rango de aplicaciones de las FPGA es muy amplio, debido a la versatilidad y a la flexibilidad de estos dispositivos. En el grafico II-11 se puede apreciar la distribución de las aplicaciones de las FPGA.

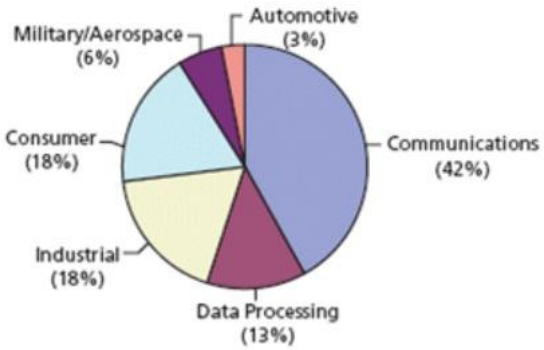


Figura II-11: Distribución de las aplicaciones de los FPGA

La principal aplicación de las FPGA está orientada al procesamiento digital de señales (DSP), la cual es empleada en comunicaciones, procesado de datos, etc. La elección de una FPGA para aplicaciones de tratamiento de señal se debe a su alta frecuencia de trabajo, a su capacidad de procesamiento en paralelo, y a su bajo precio en comparación con los ASICS.

De esta aplicación se derivan una gran variedad de aplicaciones de las FPGA, citándose algunas de ellas a continuación:

2.8.1 SISTEMAS DE VISIÓN ARTIFICIAL

En el mundo actual existen cada vez en más número dispositivos que disponen de un sistema de visión artificial. Ejemplo de esto son las cámaras de videovigilancia, robots, etc. Muchos de estos dispositivos precisan de un sistema para conocer su posición, reconocer los objetos de su entorno, reconocer rostros de personas, y poder actuar e interactuar con ellos de la forma adecuada. Esta característica requiere manejar unos volúmenes de imágenes muy elevados, tratar dichas imágenes para detectar objetos, reconocer rostros, etc., en la gran mayoría de ocasiones en tiempo real.

2.8.2 SISTEMAS DE IMÁGENES MÉDICAS

Cada vez con más frecuencias se están empleando las FPGAs para el tratamiento de imágenes biomédicas obtenidas mediante procesos de PET, escáner CT, rayos X, imágenes tridimensionales, etc. Estos sistemas de visión médica cada vez precisan de más resolución y de una capacidad de procesamiento mayor, incluso muchas necesitan poder desarrollarse en tiempo real, por lo que las prestaciones que ofrecen las FPGA de frecuencia y procesamiento en paralelo se adaptan muy bien a estas necesidades.

2.8.3 PROCESAMIENTO DE SEÑALES

Los componentes especializados como los multiplicadores, mejoran de manera significativa el uso y la eficiencia de los FPGAs para las aplicaciones de DSP. Existen multiplicadores de hasta 18 bits por 18 bits. Estos bloques, específicamente diseñados para operaciones de información de DSP y de análisis señales, incluyen circuitos integrados para multiplicaciones y sumas. Los FPGA son óptimos para DSP debido a que la mayoría de procesos son en paralelo, permitiendo acelerar la

aplicación de procesamiento digital de señales hasta 1000 veces, además utiliza altas tasas de muestreo y gran ancho de banda para datos y realiza operaciones de punto fijo.

2.8.4 RADIO DEFINIDA POR SOFTWARE

De forma tradicional, una radio consta de una antena, encargada de recibir y enviar una señal, y un hardware encargado de procesar esa señal, filtrarla, modificar su frecuencia, etc. Este hardware no podía modificar de forma notable la funcionalidad para la cual había sido diseñada. En la actualidad gran parte de esta funcionalidad se traslada a un dispositivo eléctrico, que con frecuencia suele ser una FPGA, pudiendo limitarse la parte analógica a una antena y a los convertidores ADC y DAC, como muestra la figura II-12.

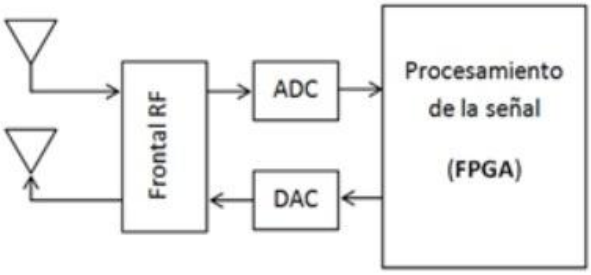


Figura II-12: Esquema de una radio definida por software

.

La principal ventaja de este tipo de radio es que su funcionalidad viene definida por el diseño del software, de forma que su modificación o actualización es sencilla y no precisa de la sustitución de ningún elemento de hardware.

2.8.5 CODIFICACIÓN Y ENCRIPTACIÓN

La seguridad en el envío de mensajes es fundamental en la vida diaria, por ejemplo a la hora de enviar un email o de realizar una compra por internet, y lo es más aún en el ámbito militar, aeronáutico y gubernamental. En este terreno, la encriptación eficiente y segura de mensajes se convierte en un objetivo prioritario. Las FPGAs pueden aportar en este terreno su capacidad de mejorar grandes volúmenes de información y sus bloques optimizados para realizar operaciones aritméticas.

2.8.6 RADIOASTRONOMÍA

La radioastronomía es la ciencia que se encarga de estudiar los fenómenos que ocurren en el espacio mediante la captación de la radiación electromagnética procedente de éste. De forma similar a las aplicaciones anteriores, precisa del procesamiento de una gran cantidad de información en el que la FPGA puede aportar todo su potencial.

2.8.7 RECONOCIMIENTO DE VOZ

El reconocimiento de la persona que habla es una técnica empleada en seguridad, sistemas de recuperación de información, etc., y se espera que en el futuro su ámbito de aplicaciones aumente. En este contexto, la FPGA resulta muy eficiente a la hora de realizar la comparación de la voz de una persona con unos patrones previamente almacenados.

2.8.8 AERONÁUTICA Y DEFENSA

Además de las mencionadas previamente, existen multitud de aplicaciones aeronáuticas y de defensa que emplean FPGAs debido a las buenas características que éstas ofrecen.

CAPÍTULO III

HERRAMIENTAS DE DESARROLLO Y PROVEEDORES DE TECNOLOGÍA FPGA

3.1 INTRODUCCIÓN

En este capítulo se analizaran las dos empresas líderes en tecnología programable FPGA haciendo énfasis a todos los beneficios que nos puede proporcionar en el ámbito académico, razón por la cual se ha escogido analizar las empresas de Xilinx y Altera ya que las dos son grandes proveedores de productos para el diseño y desarrollo de sistemas combinacionales.

3.2 INTRODUCCIÓN A LA TECNOLOGÍA FPGA

El primer fabricante de los dispositivos FPGA fue Xilins, Ross Freeman y Bernard Vonderschmitt co-fundadores de esta empresa inventaron las FPGA en el año 1984, y surgen como una evolución de los CPLDs. Los dispositivos de Xilinx se mantienen como uno de los más populares en compañías y grupos de investigación. Otras empresas también se han dedicado a la fabricación de dispositivos programables, en el mercado existen un sin número de proveedores de tecnología FPGA.

En la actualidad la tecnología para el diseño de sistemas digitales ha avanzado mucho, el diseñador buscara alternativas nuevas de desarrollo a diferencia de lo comúnmente usado como son los circuitos estándar de función fija TTL o CMOS. La creación de circuitos lógicos está orientada a crear circuitos generalmente programables con el propósito de poder satisfacer las necesidades de los todos los sectores del mercado y la posibilidad de acercar el mundo de los dispositivo programables a más sectores, ya que es más sencillo programar un chip que diseñarlo a nivel de compuertas, permitiendo a que el diseñador opte pon diversas maneras de solución ante un determinado problema y así no tener soluciones tecnológicas cerradas y tener una fácil implementación del diseño en hardware.

3.3 MERCADO ACTUAL DE LAS FPGA

El uso de dispositivos FPGA está consolidado a nivel mundial. Los porcentajes de utilización de estos dispositivos respecto a tecnologías anteriores varían según el tipo de industria. Por ejemplo, destacan los altos porcentajes de utilización en la industria aeroespacial (64%) o industrias de video (62%), mientras que el sector de la automoción se sigue inclinando por tecnologías anteriores basadas en ASICs.

El empleo de FPGAs no conlleva la exclusión del resto de ASICs. En muchos casos, las primeras son empleadas en la elaboración de prototipos de los ASIC finales. Otro punto importante de las FPGA es la invasión en el mercado de los DSP. Actualmente, cerca de un 42% de los usuarios de FPGA emplean estos tipos de chips en la realización de diferentes aplicaciones de DSP.

En el mercado existen varios proveedores de productos FPGA como son: Actel, Altera, Atmel, Cypress, Lattice, Quicklogic, Xilinx.

El objetivo es encontrar entornos educativos adecuados de las dos empresas más sobresalientes en tecnología FPGA para su utilización tanto en prácticas de laboratorio como en posibles proyectos adicionales, se valora específicamente aquellas compañías que sigan el siguiente perfil:

- Compañías que ofrezcan un programa educativo en el que se dan facilidades a las instituciones académicas para el acceso a sus productos y recursos, consiguiendo importantes descuentos en la adquisición de tarjetas educacionales destinadas al laboratorio.

- Disponer de un software libre de desarrollo de sistemas digitales, descargable desde la página web correspondiente del proveedor, con el objeto que esté disponible para cualquier persona interesada en el aprendizaje de sistemas digitales utilizando la tecnología FPGA.
- Los kits de desarrollo deben obtener a más del chip FPGA elementos que permitan diseñar un completo sistema combinacional como: chip de memoria, elementos indicadores (leds, displays), elementos de entrada (interruptores, pulsadores), diferentes puertos de entrada y salida, cables de interconexión, drivers correspondientes y sus respectivos manuales para la ayuda y entendimiento de los dispositivos.

Según los requisitos mencionados, se decide centrar el estudio en las compañías Altera y Xilinx, descartando el resto de compañías, debido a la razón que no ofrecen un programa educativo para la adquisición de sus productos o no ofrecen un software libre de desarrollo. Además los productos ofrecidos por estas compañías no están orientados a la realización de prácticas de laboratorio a nivel de inicialización, pese a que son productos potentes, ya que no ofrecen la variedad de recursos necesarios para la realización de un completo sistema combinacional.

3.3.1 ALTERA

La compañía Altera fue fundada en el año 1983 por Robert Hartman, Michael Magranet, Paul Newhagen y Jim Sansbury, tiene su sede en San José, California, y emplea a más de 3000 personas en más de 20 países, teniendo puntos de distribución en todos los continentes del mundo.

Altera Corporation es el pionero en soluciones de lógica programable, permitiendo a compañías de sistemas y semiconductores innovarse de forma rápida y rentable, diferenciarse y ganar en sus mercados. Altera ofrece FPGAs, SoCs con sistemas de procesadores embebidos, CPLDs y ASICs en combinación con la herramientas de software, propiedad intelectual, procesadores embebidos y atención al cliente para proporcionar soluciones programables de alto valor a más de 16000 clientes en todo el mundo.

Al mantener alianzas sólidas y a largo plazo con los proveedores de tecnología líderes en la industria, tales como Intel y Taiwan Semiconductor Manufacturing Company (TSMC), Altera ofrece a sus clientes garantía de calidad y entrega a tiempo de sus

productos. Además Altera ofrece diversos tipos de FPGAs dividiendo a estas en tres diferentes familias, de las FPGA de alto rendimiento hay que destacar las FPGA de la familia ‘‘Stratix’’, la familia ‘‘Arria’’ de gama media y de las FPGA de bajo coste hay que mencionar a la ‘‘Cyclone’’ como la familia más vendida.

Altera también ofrece el software Quartus II, dirigido al diseño y simulación de circuitos lógicos. Aunque su software soporta extensivamente VHDL y Verilog como principales lenguajes, Altera es el desarrollador del lenguaje de descripción de hardware conocido como AHDL.

3.3.2 XILINX

Xilinx es la mayor empresa del mundo encargada de la investigación y el desarrollo de los chip conocidos como FPGA. Fue fundada por Ross Freeman, inventor de las FPGA, y también por Bernie Vonderscmitt y Jim Barnett en 1984; con base, al igual que otras empresas del sector, en Silicon Valley (California) Estados Unidos. Hoy en día la base reside en San Jose (California), mientras que se encuentran subsede en América del Sur, Europa, Asia, África y Australia.

En la actualidad, la empresa Xilinx es el proveedor líder a nivel mundial de todas las FPGAs programables, SoC y 3D ICs, que son usados en una gran variedad de aplicaciones: telecomunicaciones, automoción, productos de consumo, industria militar y otros campos.

Las líneas de producción de Xilinx incluyen las series Virtex, Kintex y Artix, cada uno incluye configuraciones y modelos optimizados para diferentes aplicaciones. Con la introducción de la serie 7 de Xilinx en junio del 2010, la empresa se ha trasladado a tres grandes familias de productos FPGA, la gama alta por la familia Virtex, Kintex de gama media y la familia de Artix de bajo costo, y se retiró la marca Spartan que termina con la serie 6 de Xilinx FPGA.

Por otra parte, la empresa Xilinx también se encarga del desarrollo del software ISE Design Suite, para dotar a los clientes de las herramientas necesarias para desarrollar proyectos con los dispositivos que ellos venden.

3.4 SOFTWARE

La herramienta de desarrollo ISE (Integrated Software Environment) de la empresa Xilinx constituye un verdadero entorno EDA (Electronic Design Automation). La figura III-1 representa el esquema de los componentes más importantes del ISE y la secuencia en que se utilizan.

La interfaz gráfica de usuario (GUI: Graphic User Interface) se denomina Proyect Navigator y facilita el acceso a todos los componentes del proyecto. Los diseños de usuario se pueden introducir mediante diferentes formatos. Los más utilizados son: los esquemáticos, los grafos de estados y las descripciones hardware en VHDL. Una vez compilados los diseños se puede simular su comportamiento a nivel funcional o a nivel temporal. A nivel funcional no tiene en cuenta los retardos provocados por el hardware y a nivel temporal simula el diseño teniendo en cuenta cómo se va a configurar el hardware.

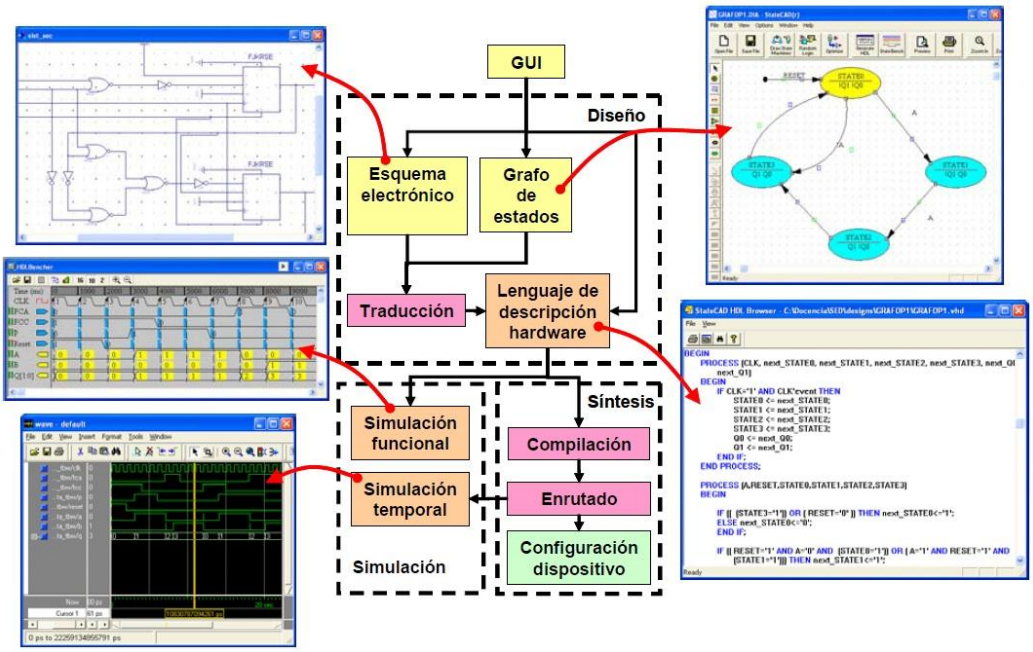


Figura III-1: Esquema de los componentes de la herramienta de desarrollo ISE

Otra herramienta CAD de la empresa Altera es Quartus II para el análisis y la síntesis de diseños realizados en HDL. Quartus II permite al diseñador compilar sus diseños, realizar análisis temporales, examinar diagramas RTL y configurar el dispositivo de destino con el programador, en la figura III-2 se muestra las características principales.

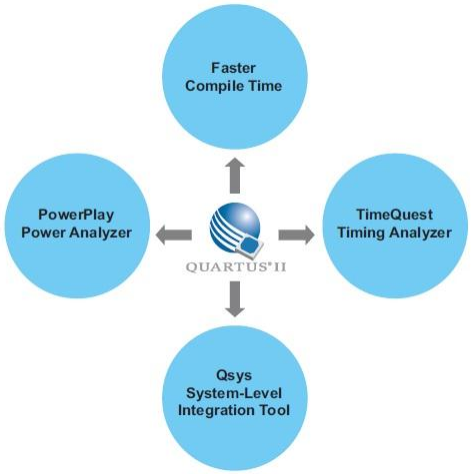


Figura III-2: Características principales del software Quartus II

Además el software proporciona un entorno de diseño completo, multiplataforma, que se adapta fácilmente a sus necesidades específicas de diseño. Se trata de un entorno completo para el diseño de un sistema programable en un chip (SOPC). El software Quartus II incluye soluciones para todas las fases del diseño de FPGA y CPLD. Además permite usar la interfaz gráfica y la interfaz de línea de comandos para cada fase del flujo de diseño. Puede utilizar una de estas interfaces para todo el flujo, o puede utilizar diferentes opciones en diferentes fases, en la figura III-3 se muestra en entorno de desarrollo de Quartus II.

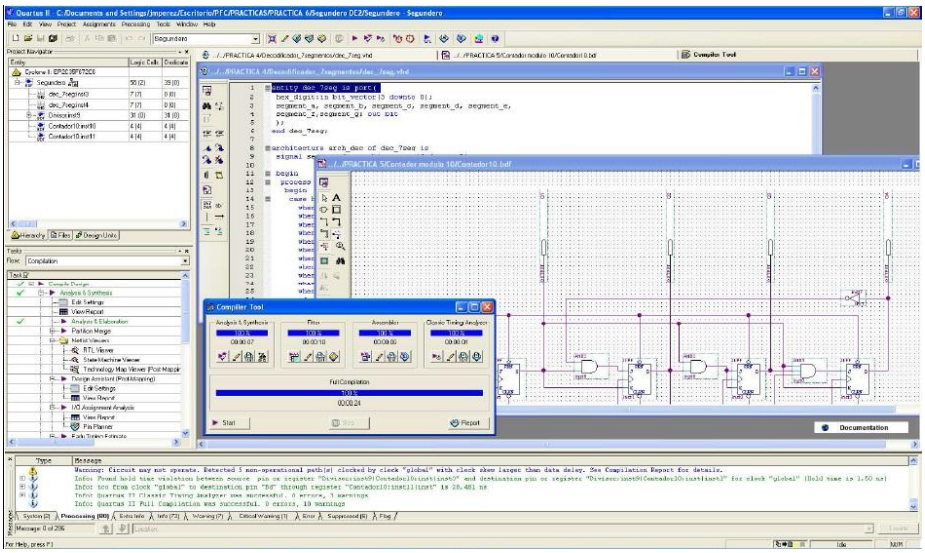


Figura III-3: Entorno de Quartus II

3.5 TARJETAS EDUCACIONALES DE ALTERA Y XILINX.

Un FPGA es un dispositivo programable basado en bloques lógicos y celdas de almacenamiento interconectadas entre sí de tal manera que permite el diseño de funciones lógicas complejas. Cada FPGA, dependiendo del fabricante y el modelo tendrá una serie de características únicas de cada chip.

Una tarjeta de educacional FPGA consta de varios elementos y dispositivos soldados a la placa base como en un sistema embebido dependiendo del fabricante podemos encontrar Kits de desarrollo con diferentes singularidades dependiendo el propósito al que se le vaya a otorgar.

Como mencionamos se analizaran los Kits de desarrollo que presten un entorno adecuado para prácticas de laboratorio haciendo énfasis a lo dispuesto anteriormente.
La empresa Xilinx ofrece una serie de Kits con tecnología FPGA los cuales q describirán a continuación:

BASYS 2 SPARTAN-3E FPGA BOARD

La tarjeta Basys2 es un diseñador de circuitos y plataforma de implementación que cualquiera puede utilizar para ganar experiencia al momento de elaborar circuitos digitales reales, como se muestra en la figura III-4. Construido con un FPGA Spartan-3E de Xilinx y un controlador USB AT90USB2 de Atmel, la tarjeta Basys2 proporciona un completo hardware listo para usarse adecuado para circuitos que van desde dispositivos lógicos básicos asta controladores complejos. Una larga colección de dispositivos E/S y todos los circuitos de soporte requeridos están incluidos, por lo que un sinnúmero de diseños se puede crear sin necesidad de ningún otro componente.

Cuatro conectores estándar de expansión permiten crear diseños más allá de la tarjeta basys2 utilizando placas universales, diseños de usuarios, tarjetas de circuitos o Pmods (los Pmods son módulos económicos digitales y analógicos E/S que ofrecen conversores A/D & D/A, controladores de motor, censores de entrada y muchas otras características). Las señales de los conectores de 6-pines están protegidas contra daños por ESD y cortocircuitos, garantizando una larga vida útil en cualquier entorno. La tarjeta Basys2 funciona a la perfección con todas las versiones de las herramientas de Xilinx ISE, incluyendo el WebPack gratuito, además no requiere otras fuentes de alimentación o cables de programación.

CARACTERÍSTICAS

- FPGA Spartan 3-E de Xilinx, 100K o 250K compuertas.
- Multiplicadores de 18-bits, bloques RAM de doble puerto de 72Kbits de velocidad y velocidades de operación superiores a 500MHz.
- Puerto USB2 full-speed para configuración FPGA y datos de transferencia.
- Plataforma Flash ROM XCF02 para almacenamiento y configuración FPGA.
- Frecuencia de oscilación configurable por el usuario (25, 50 y 100 MHz), socket externo oscilador secundario.
- Tres reguladores de voltaje (1.2V, 2.5V y 3.3V) que permite usarse como suplemento de energía externa de 3.5V – 5.5V.
- 8 LEDs, displays de 4 digitos de 7 segmentos, 4 pulsadores, 8 switches, un puertp PS/2 y un puerto VGA de 8-bits.
- 4 cabeceras de expansión de 6-pines E/S.

PRECIO

- Particular: $ 89.
- Académico: $64.99.

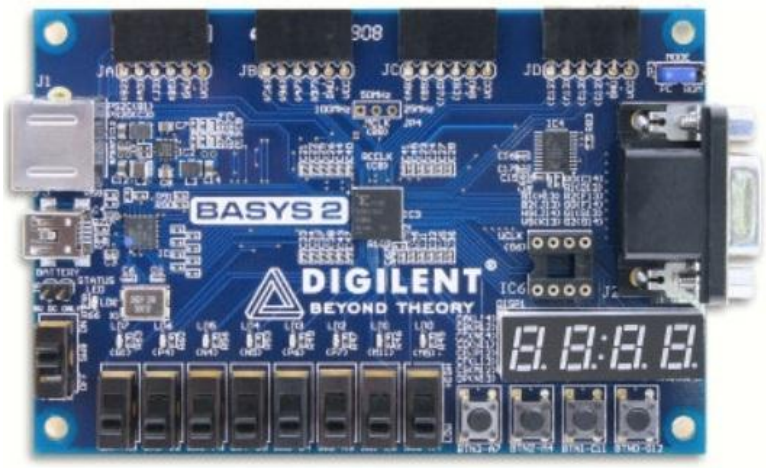


Figura III-4: Tarjeta Basys 2 Spartan 3-E de Xilinx

SPARTAN-3AN STARTER KIT

Esta mesa general de evaluación incluye una tarjeta de red Ethernet PHY 10/100, circuitos integrados ADC/DCA y opciones de conectividad como se muestra en la figura III-5. Los subsistemas robustos le permiten:

- Diseños de referencia Multiboot pre programados en una sola sesión.
- Innumerables aplicaciones prototipo con soporte de 26 entradas y salidas estándar en una sola terminal.
- Conector para memoria DDR2 para una rápida transferencia de datos utilizando el controlador de interfaz de memoria.
- Menor tiempo de desarrollo con diseño de archivos y esquemas pre-verificados.
- Proteja sus diseños con características exclusivas de Spartan-3AN.

INCLUYE EN EL KIT:

- Placa de desarrollo.
- Fuente de alimentación 100-240 V, 50-60 Hz con adaptador universal.
- Software ISE WebPack y Software de evaluación ISE Foundation.
- Guía de inicio rápido.
- Cable de programación.
- Garantía del producto.

CARACTERÍSTICAS PRINCIPALES:

Dispositivos Xilinx:

- Spartan-3AN (XC3S700AN-FG484).
- Plataforma Flash (XCF04S-VOG20C).

Relojes:

- Oscilador de cristal 50 MHz.
- Ranura opcional de reloj.

Memoria:

- Plataforma Flash PROM de 4Mbit.
- SDRAM DDR2 de 32M x 16.
- Parallel Flash de 32Mbit.
- Dispositivo Flash SPI de 2 x 16 Mbit.

Dispositivos de interfaz analógica:

- Convertidor D/A de 4 canales.
- Convertidos A/D de 2 canales.
- Amplificador de señal.

Conectores e interfaces:

- Ethernet 10/100 PHY
- Puerto de descarga USB JTAG.

- Dos puestos serial RS-232 de 9 pines.
- Puerto PS/2 estilo de teclado/ratón.
- Conector VGA de 15 pines de 4.096 colores.
- Una FX2 de 100 pines y dos conectores de expansión de 6 pines.
- Un Mini-jack estéreo de audio PWM.
- Selector de funciones pulsador/giratorio.
- Ocho salidas LED individuales.
- Cuatro Interruptores, cuadro pulsadores.
- 2 líneas LCD de 16 caracteres.

SOLICITUDES DIRIGIDAS

- **Mercado:** Telecom / Datacom, Servidores, Almacenamiento, Consumo.
- **Aplicaciones:** Prototipos en genera.

PRECIO: $ 199.

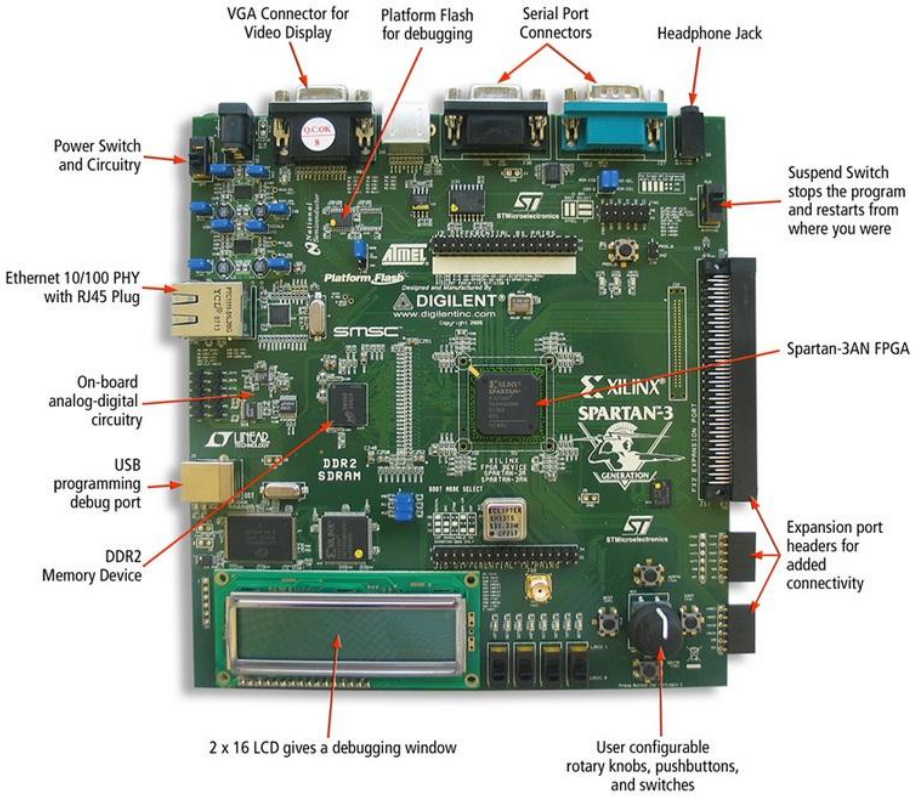


Figura III-5: Tablero de desarrollo Spartan-3AN

.

SPARTAN-3A STARTER KIT

El kit de trabajo Spartan-3A proporciona acceso instantáneo a un dispositivo FPGA con características tales como: modo de ahorro de energía (suspender), opciones de E/S de alta velocidad, interfaz de memoria SDRAM DDR2, soporte de configuración de productos Flash y la protección FPGA/IP utilizando seguridad DNA del dispositivo, en la figura III-6 se muestra el kit completo.

INCLUYE EN EL KIT:

- Tablero de desarrollo.
- Fuente de alimentación 100-240V, 50/60Hz con adaptador universal.
- Software ISE WebPack y software de evaluación ISE Foundation.
- Guía de inicio rápido.
- Cable de programación.
- Garantía del producto.

CARACTERÍSTICAS PRINCIPALES:

Dispositivos Xilinx:

- Spartan-3A (XC3S700A-FG484).
- Plataforma Flash (XCF04S-VOG20C).

Relojes:

- Oscilador de cristal de 50 MHz.
- Ranura opcional de reloj.

Memoria:

- Plataforma Flash PROM de 4Mbit.
- SDRAM DDR2 de 32M x 16.
- Parallel Flash de 32 Mbit.
- Dispositivo Flash SPI de 2-16Mbit.

Dispositivos de interfaz analógica:

- Convertidor D/A de 4 canales.
- Convertidor A/D de 2 canales.
- Amplificador de señal.

Conectores e interfaces:

- Ethernet 10/100 PHY.
- Puerto de descarga USB JTAG.

- Dos puertos serial RS-232de 9 pines.
- Puerto PS/2 estilo teclado/ratón.
- Conector VGA de 15 pines capacidad para 4.096 colores.
- Un FX2 de 100 pines y dos conectores de expansión de 6 pines.
- Un Mini-Jack estéreo de audio PWM.
- Selector de funciones pulsador/giratorio.
- Ocho salidas LED individuales.
- Cuatro interruptores.
- Cuatro pulsadores.

Display:

- Un LCD de 2 líneas de 16 caracteres.

SOLICITUDES DIRIGIDAS

- **Mercado:** Telecom / Datacom, Servidores, Almacenamiento, Consumo.
- **Aplicaciones:** Prototipos en genera.

PRECIO: $ 189.

Figura III-6: Kit de desarrollo Spartan-3A

SPARTAN-3A DSP 1800A EDITION

Evalúe sus aplicaciones DSP utilizando el entrenador versátil XtremeDSP suministrado por FPGA Spartan-3A DSP 1800A, en la figura III-7 se muestra esta placa de desarrollo.

Esta plataforma de evaluación proporciona acceso a las prestaciones de la familia Spartan-3A DSP y soporta periféricos, conectores e interfaces estándar de la industria. Está diseñado para su uso con el sistema generador de Xilinx DSP y con la herramienta de diseño ISE.

INCLUYE EN EL KIT:

- Tablero de desarrollo Spartan-3A DSP 1800A XtremeDSP.
- Software ISE WebPACK y el software de evaluación ISE Foundation.
- Fuente de alimentación 100-240V, 50-60 Hz con adaptador universal.
- No incluye cable USB de descarga y debe pedírselo por separado.

CARACTERÍSTICAS PRINCIPALES

Dispositivos Xilinx:

- FPGA Spartan-3A DSP XC3SD1800A-4FGG676C.

Relojes:

- Oscilador LVTTL SMT de 125 MHz.
- Socket para oscilador LVTTL.
- Oscilador LVTTL SMT de 25.175 MHZ (reloj de video).
- Reloj Ethernet de 25 Mhz (accesible al FPGA).

Memoria:

- SDRAM DDR2de 128 MB (32M x 32).
- Configuración Flash Parallel / PBI de 16M x 8.
- Configuración / Almacenamiento Flash SPI de 64 Mb (con 4 selectores SPI extras).

Interfaces:

- Ethernet 10/100/1000 PHY.
- Puerto JTAG programación / configuración.
- Puerto RS232.
- Puerto VGA de bajo costo.
- líneas selectoras SPI.

Botones y Switches:

- 8 indicadores LED.
- DIP switch de 8 posiciones.
- botones pulsadores.
- Pulsador de reset.

Entradas y salidas de expansión:

- 2 Digilent de 6 pines.
- 2 conectores de expansión EXP.
- Conector GPIO de 30 pines (puede ser usado para sistemas ACE tarjeta flash auxiliar compacta).

Configuración y depuración:

- JTAG.
- Modulo conector sistema ACE.

SOLICITUDES DIRIGIDAS

- **Mercados:** Wireless, Automotriz, Industrial, Consumo, Medicina, Militar / Aeroespacial, Servidores, Almacenamiento y Telecom / Datacom Systems.
- **Aplicaciones:** Prototipos en general, Procesos embebidos, Video Digital, DSP, Procesamiento de imágenes, Comunicaciones Digitales y co-procesamiento.

PRECIO: $ 595.

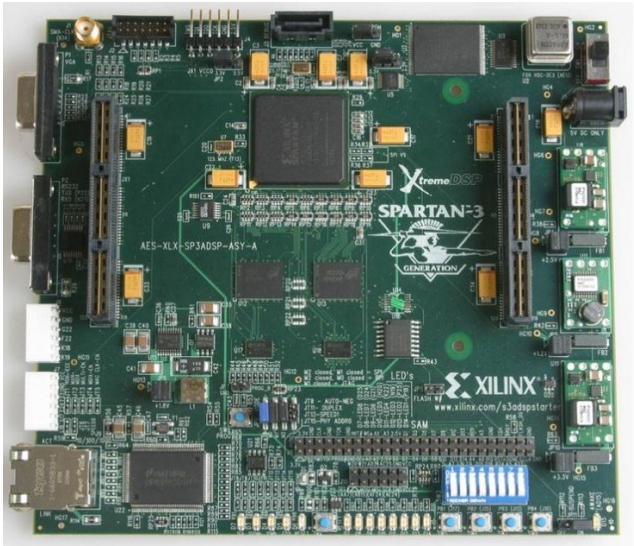


Figura III-7: Placa de desarrollo Spartan-3A DSP 1800A Edition

SPARTAN-3E STARTER KIT

El kit de alto volumen Spartan-3E ofrece a los diseñadores acceso instantáneo a las capacidades de la plataforma de la familia Spartan-3E.
El FPGA Spartan-3E es una completa placa de desarrollo que brinda soluciones y da a los diseñadores acceso instantáneo a las capacidades de la familia Spartan-3E. El kit completo incluye, placa, fuente de alimentación, software, CD (notas de aplicaciones, documentos, datasheet) y cable USB, en la figura III-8 se muestra la placa entrenadora Spartan-3E.

INCLUYE EN EL KIT:

- Placa de desarrollo.
- Fuete de alimentación universal 100-240V, 50/60 Hz.
- Software ISE WebPACK y software de evaluación ISE foundation, y el Kit de desarrollo integrado (EDK).
- Manual: introducción a la lógica programable Diseño de Inicio Rápido.
- CD Kit Inicializador.
- Cable USB.

CARACTERÍSTICAS PRINCIPALES

Dispositivos Xilinx:

- FPGA Spartan-3E (XC3S500E-4FG320C).
- CPLD CoolRunner-II (XC2C64A-5VQ44C).
- Plataforma Flash (XCF04S-VO20C).

Rejo:

- Oscilador de cristal de 50 MHz.

Memoria:

- Parallel Flash de 128 Mbit.
- Flash SPI de 16 Mbit.
- SDRAM DDR2 de 64 Mbyte.

Conectores e interfaces:

- Ethrenet 10/100 PHY
- Puerto USB JTAG de descarga.
- Dos puertos serial RS-232 de 9 pines.
- Puerto PS/2 estilo teclado/ratón, codificador rotativo con pulsador.
- Cuatro interruptores deslizantes.

- Ocho salidas de LED individuales.
- Cuatro botones de contacto momentáneo.
- Puerto de conexión de expansión de 100 pines.
- Tres conectores de expansión de 6 pines.

Display:

- LCD de 2 líneas de 16 caracteres.

SOLICITUDES DIRIGIDAS:

- **Mercado:** Consumidor, Telecom/Datacom, Servidores, Almacenamiento.
- **Aplicaciones:** Prototipos en general.

PRECIO:

- **Particular:** $ 395.
- **Académico:** $ 199.

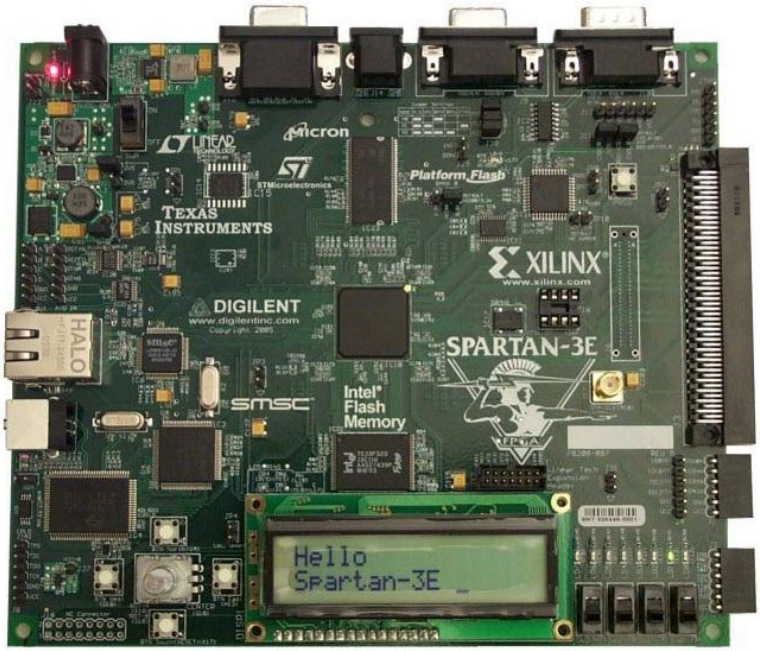


Figura III-8: Placa de desarrollo Spartan-3E

.

ATLYS SPARTAN-6 FPGA DEVELOPMENT KIT

La placa de circuitos Atlys como se muestra en la figura III-9, es una plataforma completa lista para el uso en el desarrollo de circuitos digitales basada en la familia en la familia FPGA de Xilinx Spartan-6 LX45. Los circuitos en la placa base son periféricos de gama alta, incluye Gbit Ethernet, HDMI, Matriz de memoria DDR2 de 128Mbyte, puestos de audio y USB hacen que el módulo Atlys sea el anfitrión ideal para sistemas completos diseñados con procesadores embebidos con el MicroBlaze de Xilinx. Atlys es totalmente compatible con todas las herramientas CAD de Xilinx, incluyendo ChipScope, EDK y el WebPack gratuito, por lo que los diseños se pueden llevan a cabo sin costos adicionales.

CARACTERÍSTICAS PRINCIPALES.

- El Spartan-6 esta optimizada para la lógica de alto rendimiento y ofrece:
- 6882 partes cada una con 4 LUTs de 6 entradas y 8 flip-flops.
- Bloque rápido RAM de 2.1Mbits.
- fichas de reloj (8 DCMs & 4PLLs).
- loops de faces cerradas.
- 58 partes DSP.
- Velocidades de reloj mayores de 500MHz.
- FPGA Spartan-6 LX45, de 324 pines de paquete BGA.
- Memoria DDR2 de 128 Mbyte de 16 bits de datos de ancho.
- Un conector Vmod (VHDC de alta velocidad).
- Un conector Pmod de 12 pines.
- Puerto Ethernet 10/100/1000 PHY.
- 2 puertos USB de programación y datos de transferencia.
- 2 puertos de entrada HDMI y 2 puertos de salida HDMI.
- Codec de audio AC-97 con línea de entrada, línea de salida, micrófono y auriculares.
- Memoria Flash SPI de 16Mbyte x 4 para configuración y almacenamiento de datos.
- Oscilador CMOS de 100MHz.
- GPIO, incluye 8 LEDs, 6 botones y 8 interruptores deslizantes.
- Se suministra con fuente de alimentación de 20W y cable USB.

PRECIO

- **Particular:** $ 399.
- **Académico:** $219.

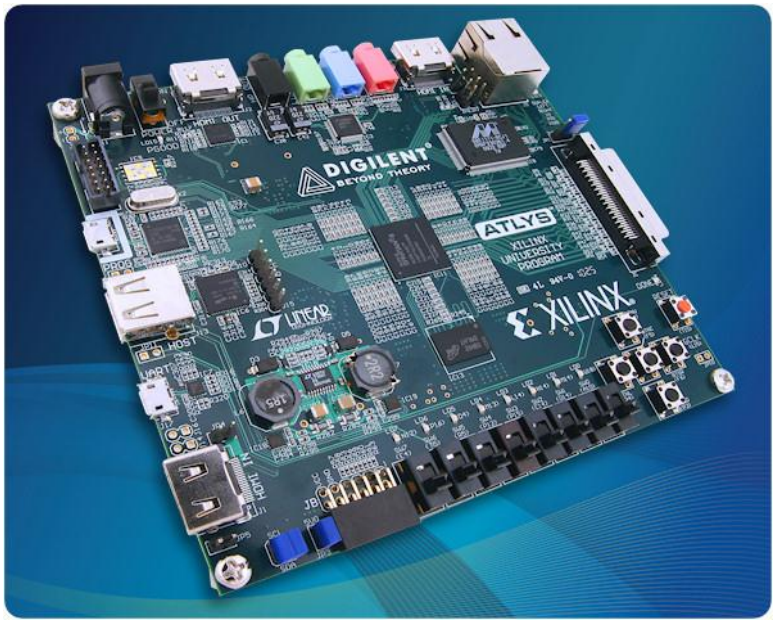


Figura III-9: Placa de desarrollo Atlys Spartan-6

ANVYL SPARTAN-6 FPGA DEVELOPMENT BOARD

La Anvyl FPGA es una plataforma de desarrollo completa, lista para usarse en el desarrollo de circuitos digitales basados en una velocidad de gardo-3 de la familia FPGA Spartan-6 LX45 de Xilinx. La amplia FPGA, cuenta con un puerto Ethernet de 100-mbps, video HDMI, memoria DDR2 de 128MB, pantalla táctil LCD de 4,3'', pantalla OLED de 128x32 pixeles, varios controladores USB HID y el códec de audio, esto hace que la Anvyl sea una plataforma ideal capaz de soportar diseños de procesadores embebidos basados en el MicroBlaze de Xilinx.

El Anvyl es compatible con las herramientas CAD de Xilinx, incluyendo ChipScope, EDK y el ISE WebPack gratuito, por lo que los diseños se pueden completar sin costo adicional. La tarjera mide 27.5cm x 21cm y se la muestra en la figura III-10.

CARACTERÍSTICAS

- La Spartan-6 esta optimizada para un alto rendimiento lógico y ofrece:
- FPGA Spartan-6 LX45 (XC6SLX45-CSG484-3).
- 6.822 slices, cada cual contiene 4 entradas LUTs y 8 Flip-flops.

- Bloque rápido RAM de 2.1Mbits.
- 4 fichas de reloj (8 DCMs y 4 PLLs).
- 58 Slices DSP.
- Velocidades de reloj mayores de 500MHz.
- Memoria SDRAM DDR2 de 128MB.
- Memoria SRAM de 2MB.
- Memoria Flash QSPI de 16MB para configuración y datos de almacenamiento.
- Puerto Ethernet 10/100 PHY.
- Salida de Video HDMI.
- Puerto VGA de 12-bits.
- LCD táctil de 4.3''.
- 3 Displays dobles de 7 segmentos.
- Codec de Audio con línea de entrada, salida, micrófono y auriculares.
- Oscilador de cristal de 100MHz.
- Puerto USB2 para programación y dispositivo USB-HID (mouse/teclado).
- Puerto USB-JTAG circuito con función USB-UART.
- Teclado con 16 teclas etiquetadas (0-F).
- GPIO: 14 LEDs (10 rojos, 2 amarillos, 2 verdes), 8 switches deslizantes, 8 DIP switches en 2 grupos y 4 pulsadores.
- Tablero con 10 entradas/salidas digitales.
- 32 I/O distribuidos en el conector de expansión de 40 pines.
- 7 conectores Pmod de 12 pines con 56 I/O en total.
- Fuente de alimentación de 20W y cable USB.

PRECIO

- **Particular:** $ 539.
- **Académico:** $ 349.

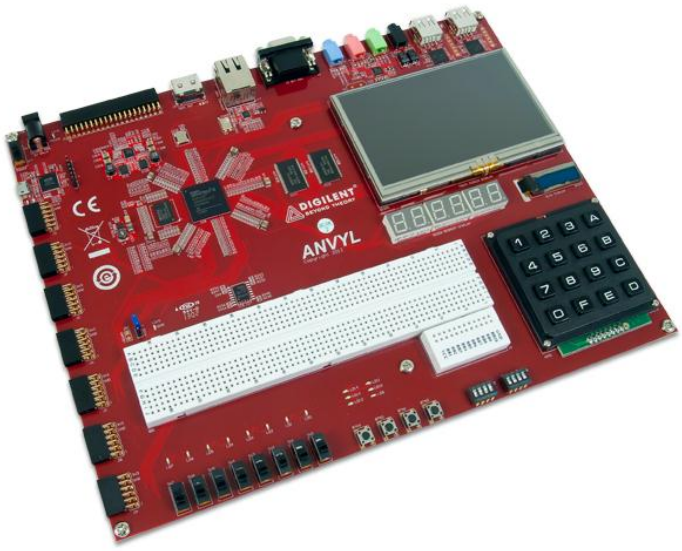


Figura III-10: Tarjeta de desarrollo educacional Anvyl Spartan-6 de Xilinx

Los Kits de desarrollo de Altera proporcionan un entorno de diseño completo y de alta calidad para los ingenieros. Una amplia variedad de kits le ayudan a simplificar el proceso de diseño y reducir el tiempo de lanzamiento al mercado. Los kits de desarrollo incluyen software, diseños de referencia, cables y hardware programable.

Entre su amplia variedad de productos que Altera ofrece se encuentran los kits educacionales únicamente para el área académica, que está a cargo del programa académico de Altera ofreciendo beneficios y ventajas a los estudiantes al momento de diseñar circuitos digitales, estos kits se presentan a continuación:

ALTERA DE0 BOARD

El tablero de desarrollo y educacional DE0 está diseñado en un tamaño compacto con todas las herramientas esenciales para los usuarios novatos que desean adquirir conocimientos en el área de la lógica digita y se lo puede apreciar en la figura III-11. Está equipado con el dispositivo FPGA de Altera Cyclone III 3C16, el cual ofrece 15.408 LEs (Elementos Lógicos). El tablero ofrece 346 pines de entrada y salida, y contiene un amplio conjunto de características que lo hacen adecuado para ser utilizado en la Universidad y en cursos en general, así como el desarrollo de sistemas digitales complejos. El DE0 de Altera combina el bajo consumo de energía, el bajo costo, y el alto rendimiento del FPGA Cyclone III para controlar las varias funciones de la tarjeta DE0. El módulo de desarrollo y educacional DE0 incluye software, diseños de

referencia, y los accesorios necesarios para garantizar un acceso sencillo al usuario en la evaluación de la DE0.

ESPECIFICACIONES

FPGA

Cyclone III 3C16 FPGA.

- 15.408 LEs.
- 56 bloques de memoria embebido cada uno con 9Kbits. (M9K)********
- 504Kbits de memoria.
- 56 multiplicadores embebidos.
- 4 PLLs.
- 346 pines de entrada y salida.
- Paquete BGA de 484 pines.

MEMORIA

SDRAM

- Un chip de memoria síncrona RAM de 8Mbyte.

Memoria Flash.

- Memoria Flash NOR de 4Mbyte.
- Nodo Myte de soporte (8 bits) / palabras (16 bits).

SD card socket.

- Proporciona acceso a ambas tarjetas SPI y SD card.

INTERFACES

Circuito incorporado USB Blaster.

- USB Blaster de programación.
- Uso del CPLD EPM240 de Altera.

Dispositivo de configuración serial Altera.

- Chip Serial EEPROM EPCS4 de Altera.

Interruptores de pulsador.

- 3 pulsadores.

Interruptores deslizantes.

- 10 interruptores deslizantes.

Entrada de reloj.

- Oscilador de 50MHz.

Salida VGA.

- Resistor de red DAC de 4 bits.
- Conector D-sub de alta densidad de 15 pines.

- Soporte mayor a 1280x1024 a 60Hz.

Puertos seriales.

- Un puerto RS-232 (conector serial de salida DB-9).
- Un puerto PS/2 (a través de un cable PS/2 Y se puede utilizar para conectar un teclado y un ratón a la vez).

Dos cabeceras de expansión de 40 pines.

- El Cyclone III 72 I/O, puede ser utilizado como 8 líneas de potencia y líneas de tierra, a través de los dos conectores de expansión de 40 pines.
- La cabecera de 40 pines está diseñada para aceptar un cable estándar plano de 40 pines que se utilizan en los discos duros.

CONTENIDO DEL KIT.

El paquete DE0 incluye:

- Tarjeta DE0 de Altera.
- Cable USB.
- CD DE0 que incluye:
  - Software Quartus II Subscription & Web Edition y el software Nios II edición de evaluación de diseños embebidos.
  - Manual de usuario DE0, Guía de inicio rápido.
- Cubierta acrílica DE0.
- Adaptador DC 7.5V/0.8A.

PRECIO

- **Particular:** $119
- **Académico:** $ 79

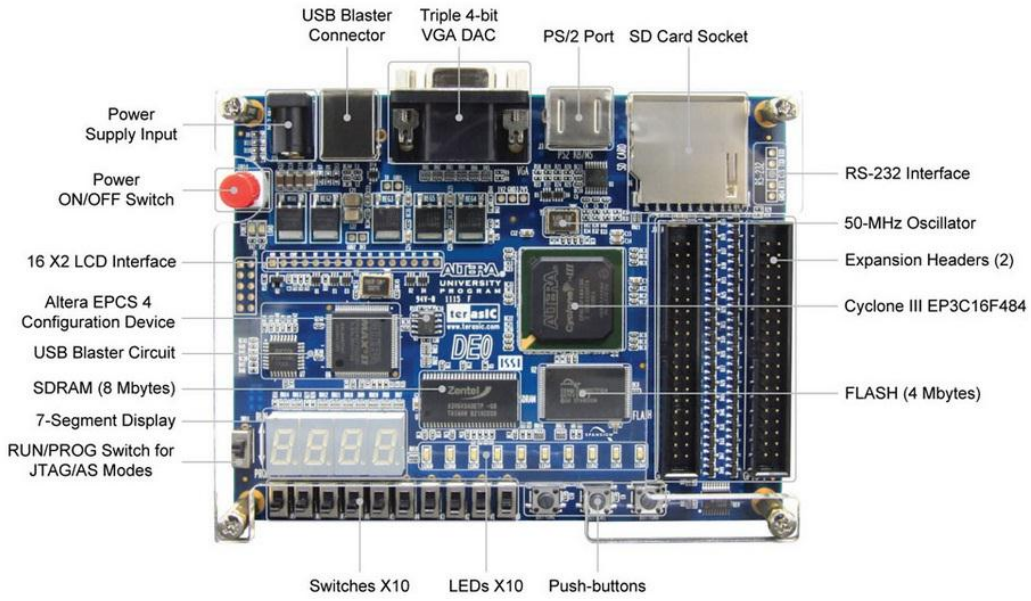


Figura III-11: Tablero de desarrollo DE0 de Altera

DE0-NANO DEVELOPMENT BOARD

El módulo DE0-Nano presenta una plataforma de desarrollo de tamaño compacto adecuado para el diseño de circuitos prototipos tales como robots y proyectos ‘‘portátiles’’. La placa está diseñada para ser utilizada en aplicaciones más sencillas posibles, utilizando el dispositivo Cyclone IV con más de 22.320 LEs.

El DE0-Nano como se muestra en la figura III-12, cuenta con una colección de interfaces que incluyes dos cabeceras externas GPIO para extender diseños, incluye dispositivos de memoria SDRAM y EEPROM para almacenamiento de datos y buffer de tramas, y periféricos en general como LEDs y pulsadores.

Las ventajas de la placa DE0-Nano incluyen su tamaño y peso, así como su capacidad de ser reconfigurado sin necesidad de llevar hardware superfluous, distinguiéndose de otras placas de desarrollo de propósito general. Además para los diseños móviles donde la energía portátil es crucial, la DE0-Nano ofrece a los diseñadores tres opciones combinadas de energía, incluyendo un puerto mini USB AB, dos pines de alimentación en la cabecera externa y dos pines DC de 5V.

ESPECIFICACIONES

FPGA Cyclone IV EP4CE22F17C6N

- 22.320 elementos lógicos (LEs).
- Memoria embebida de 594 Kbits.
- Multiplicador embebido 66 de 18x18.
- PLLs de propósito general.
- 153 pins máximos de entrada y salida.

Configuración y elementos Set-Up

- Circuito de programación USB-Blaster.
- Dispositivo de configuración serial (EPCS).

Cabeceras de expansión

- 2 cabeceras de 40 pines (GPIO) que provee 72 pines de entrada y salida.
- 2 pines de alimentación de 5V, 2 pines de alimentación de 3.3V y 4 pines de tierra.
- 1 cabecera de 26 pines que provee 16 pines digitales I/O y 8 pines analógicos de entrada para censores analógicos, etc.

Dispositivos de memoria

- SDRAM de 32 MB.
- EEPROM I2C de 2Kb.

Entradas y salidas

- 8 LEDs verdes.
- 2 pulsadores.
- 4 Dip Switches en linea.

Sensor G

- ADI ADXL345, 3-axis acelerómetro de alta resolución (13 bits).

Convertidos A/D.

- Convertidor A/D 12 bits, NS ADC128S022, 8 canales.
- 50 ksps a 200 ksps.

Reloj

- Reloj oscilador de 50MHz.

Alimentador de energía

- Puerto Mini USB tipo AB de 5V.
- 2 pines DC en la cabecera GPIO de 5V.
- 2 pines de alimentación en la cabecera externa de 3.6 – 5.7 V.

CONTENIDO DEL KIT

- Placa DE0-Nano de Altera.
- Cable Mini USB tipo B.
- CD del sistema DE0-Nano.
- CD completo de diseño de Altera Suite Free Package.
- Guía de inicio rápido DE0-Nano.

PRECIO

- **Particular:** $ 79.
- **Académico:** $ 59.

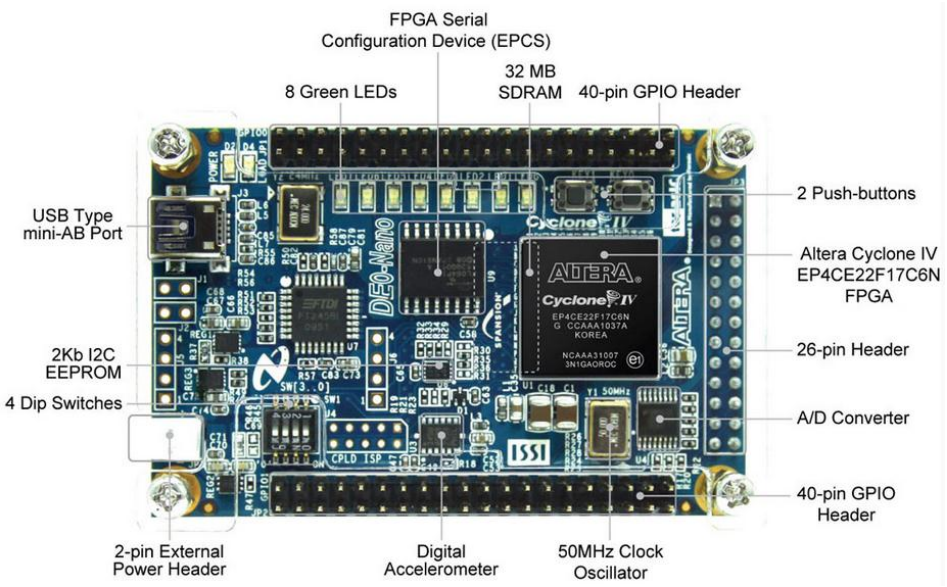


Figura III-12: Placa DE0-Nano (vista superior)

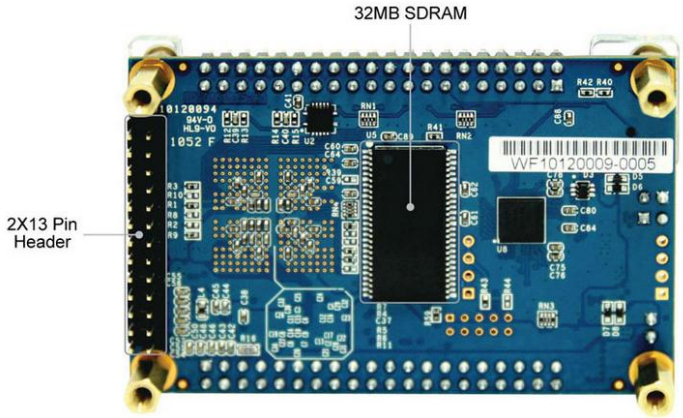


Figura III-13: Placa DE0-Nano (vista inferior)

ALTERA DE1 BOARD

EL propósito del tablero de desarrollo educacional DE1 de Altera es proporcionar una vía ideal para el diseño avanzado de prototipos en multimedia, almacenamiento y redes.

Utiliza la tecnología de última generación tanto en hardware como en herramientas CAD y exponer a los diseñadores a una amplia gama de temas. La tarjeta ofrece un exuberante conjunto de características que lo hacen adecuado para su uso en un entorno de laboratorio de la universidad y cursos de nivel superior, para una variedad de proyectos de diseño, así como para el desarrollo de sistemas digitales completos, en la figura III-14 se muestra la tarjera educacional DE1. Altera proporciona un conjunto de materiales para la tarjeta DE1, incluyendo tutoriales, ejercicios de laboratorio, y demostraciones ilustradas.

ESPECIFICACIONES

La tarjeta DE1 ofrece a los usuarios muchas características para permitir diferentes proyectos de desarrollo multimedia. La selección de componentes de realiza de acuerdo al diseño más popular en volumen de producción de artículos multimedia. La plataforma DE1 permite a los usuarios comprender rápidamente todos los trucos para diseñar proyectos para la industria.

- FPGA de Altera Cyclone II 2C20 con 20000 LEs.
- Dispositivo de configuración serial EPCS4 Cyclone II 2C20 de Altera.
- Puerto USB Blaster de programación y controlador de usuario API.
- Soporta modo JTAG y modo AS.
- Memoria SDRAM de 8Mbyte (1Mx4x16).
- Memoria Flash de 4Mbyte.
- Memoria SRAM de 512Kbyte (256Kx16).
- Socket SD Card,
- 4 Pulsadores.
- 10 Switches DPDT.
- 8 LEDs verdes.
- 10 LEDs rojos.
- displays de siete segmentos.
- Reloj oscilador de 50MHz, oscilador de 24MHz, oscilador de 27MHz y ranura de reloj externo.

- CODEC de audio CD-Quality de 24-bits con línea de entrada, línea de salida y entrada de micrófono.
- DAC VGA (4-bits R-2R por canal) con conector VGA de salida.
- Conector transmisor/receptor RS-232 de 9 pines.
- Conector PS/2 teclado/ratón.
- 2 Cabecera de expansión de 40 pines.
- CD-ROM DE1 el cual contiene ejemplos con código de inicio.

CONTENIDO DEL KIT

- Módulo DE1 de Altera.
- Cable USB.
- Adaptador DC 7.5V / 0.8A.
- DVD Altera Quartus II.
- CD-ROM DE1 Altera.

PRECIO

- **Particular:** $ 150.
- **Académico:** $ 125.

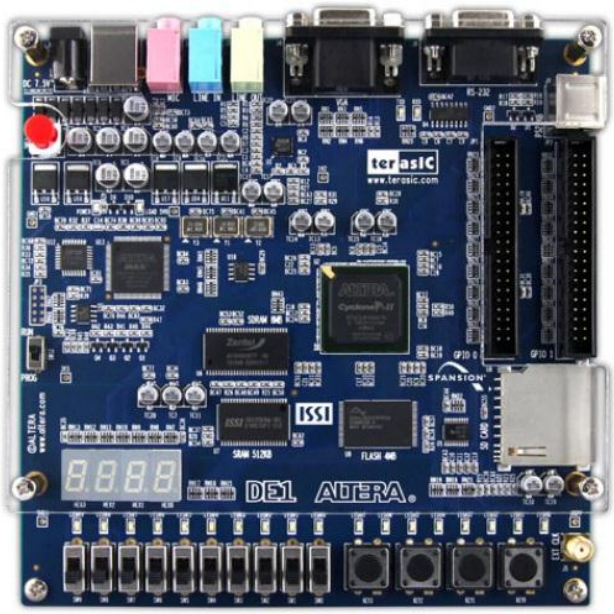


Figura III-14: Tarjera Educacional DE1 de Altera

ALTERA DE2 BOARD

El propósito de la tarjeta de desarrollo educacional DE2 de Altera es proveer una vía ideal para el diseño avanzado de prototipos en multimedia, almacenamiento y en redes, en la figura III-15 se muestra una gráfica del módulo DE2.

Utiliza la tecnología de última generación tanto en hardware como en herramientas CAD para exponer a los diseñadores a una amplia gama de temas. La tarjeta ofrece un conjunto de características que lo hacen adecuado para el uso en un entorno de laboratorio de la universidad y en cursos de nivel superior, para el desarrollo de una amplia variedad de proyectos, así como para el desarrollo de sistemas digitales complejos. Altera proporciona un conjunto de materiales de apoyo para la placa DE2, incluyendo tutoriales, ejercicios de laboratorio y demostraciones ilustradas.

ESPECIFICACIONES

La tarjeta DE2 ofrece a los usuarios muchas características que permiten el desarrollo de diversos proyectos multimedia. La selección de componentes se ha hecho de acuerdo a los diseños más populares en volumen de producción de artículos multimedia como DVD, VCD y MP3. La plataforma DE2 permite a los usuarios entender rápidamente todos los trucos para el diseño de proyectos multimedia reales para la industria.

- FPGA Cyclone II 2C35 con 35000 LEs.
- Dispositivo de configuración serial EPCS16 para Cyclone II 2C35.
- Puerto USB Blaster para programación y controlador de usuario API.
- Soporta modo JTAG y modo AS.
- Memoria SDRAM de 8Mbyte (1Mx4x16).
- Memoria SRAM de 512Kbyte (256Kx16).
- Memoria Flash de 4Mbyte.
- Socket SD Card.
- 4 pulsadores.
- 18 Switches DPDT.
- 9 LEDs verdes.
- 18 LEDs rojos.
- Un LCD de 16x2.
- Oscilador de 50-MHz y 27-MHz (decodificador de TV).

- CODEC de audio CD-Quality de 24-bits con línea de entrada, línea de salida y micrófono de entrada.
- DAC VGA (10-bits high-speed triple DACs) con conector de salida VGA.
- Decodificador de TV (NTSC/PAL) y conector de entrada de TV.
- Puerto Ethernet 10/100 con controlador.
- Controlador USB Host/Slave con conector USB tipo A y tipo B.
- Conector RS-232 emisor/receptor de 9 pines.
- Conector PS/2 teclado/raton.
- Transmisor/receptor IrDA.
- 2 Cabeceras de expansión de 40 pines con diodo de protección.
- CD-ROM Lab DE2 el cual contiene ejemplos de ejercicios con código de inicio, incluye: controlador SDRAM y Flash, CD-Quality Music Player, VGA y TV Lab, lector SD Card, Labs de comunicación RS-232/PS-2, NIOSII y panel de control API.

CONTENIDO DEL KIT

- Tarjeta DE2 de Altera.
- CD DE2 de Altera – versión 2.0.
- DVD software Quartus II con Nios II – versión 11.1
- Adaptador DC de 9V/1.3A.
- Cable USB.

PRECIO

- **Particular:** $ 495.
- **Académico:** $ 269.

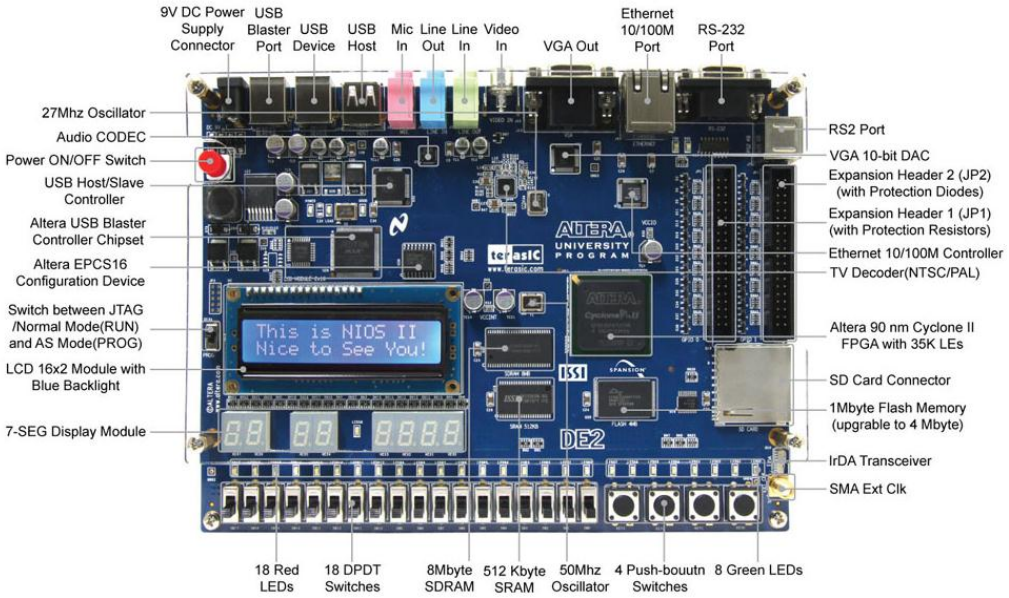


Figura III-15: Tarjeta Educacional DE2 de Altera

ALTERA DE2-115 DEVELOPMENT AND EDUCATIONAL BOARD

La serie DE2 han sido consideradas como las primeras tarjetas de desarrollo educacionales por distinguirse con una gran cantidad de interfaces para adaptarse a diferentes necesidades de aplicaciones. Extendiendo su liderazgo y éxito, Terasic anuncia la última DE2-115 que cuenta con el dispositivo Cyclone IV E. En respuesta a una mayor gama de bajo costo versátil impulsado por la demanda de video móvil, voz, acceso a datos y a la necesidad de imágenes de alta calidad, el nuevo DE2-115 ofrece un equilibrio óptimo de bajo costo, bajo consumo de energía y un amplio suministro de las capacidades de la lógica, la memoria y DSP.

El dispositivo Cyclone EP4CE115 equipado en la características de la DE2-115 cuenta con 114.480 elementos lógicos (LEs), el más grande ofrecido por la serie de la familia Cyclone IV E, hasta 3,9 Mbits de RAM y 266 multiplicadores. Además, ofrece una combinación sin precedentes de bajo consumo y funcionalidad, y menor consumo de energía en comparación con la generación anterior de los dispositivos Cyclon.

El DE2-115 adopta similares características de la serie anterior DE2 principalmente de la DE2-70, así como interfaces adicionales para soportar protocolos convencionales como Gigabit Ethernet (GbE). Un conector High-Speed Mezzanine Card (HSMC) es

dispuesto para soportar funciones adicionales y conectividad vía tarjetas HSMC y cables, en la figura III-16 se muestra la tarjeta DE2-115 de Altera.

Para el desarrollo de prototipos ASIC a gran escala, se puede realizar una conexión con dos o más tarjetas FPGA utilizando el cable HSMC a través del conector HSMC.

ESPECIFICACIONES

- FPGA Cyclone IV EP4CE115 con 114.480 LEs.
- Memoria embebida de 3.888Kbits.
- 266 multiplicadores embebidos de 18x18.
- PLLs de propósito general.
- 528 entradas y salidas de Usuario.
- Dispositivo de configuración serial EPCS64.
- Puerto y circuito de configuración USB-Blaster.
- Modo de configuración JTAG y AS.
- Memoria SDRAM de 128MB (32Mx32bits).
- Memoria SRAM de 2MB (1Mx16).
- Memoria Flash de 8MB (4Mx16).
- Memoria EEPROM de 32Xbits.
- 18 switches y 4 pulsadores.
- 18 LEDs rojos y 9 LEDS verdes.
- 8 Displays de 7 segmentos.
- Codificador/Decodificador CODEC de audio de 24-bits, con línea de entrada, salida y entrada de micrófono.
- 1 modulo LCD de 16x2.
- Tres osciladores de 50MHz.
- Conector SMA (entrada/salida externo de reloj).
- SD Card que proporciona modo SPI y modo SD de 4-bits para el acceso a SD Card.
- Dos puertos Gigabit Ethernet 10/100/1000, soporta Ethernet IP cores.
- Una High-Speed Mezzanine Card (HSMC) de 172 pines, configurable entrada/ salida (niveles de voltaje: 3.3/2.5/1.8/1.5V).
- Puerto USB 2.0 tipo A y tipo B, soporta transferencia de datos a máxima y baja velocidad.
- Puerto de expansión de 40 pines, configurable I/O estándar (niveles de voltaje: 3.3/2.5/1.8/1.5V).

- DAC VGA (High-Speed Triple DACs) con Conector VGA de salida.
- Conector serial RS-232 con control de flujo.
- Conector PS/2 para conexión Mouse/Teclado.
- Modulo conector infrarrojo (control remoto).
- TV decoder (NTSC/PAL/SECAM) con conector de entrada de TV.
- Línea de entrada de energía DC con regulador LM3150MH.

CONTENIDO DEL KIT

- Tarjeta DE2-115 de Altera.
- DVD completo Design-Suite para Windows.
- CD del sistema DE2-115 de Altera.
- Adaptador de energía DC 12V/2A.
- Cable de programación USB.
- Control remoto.

PRECIO

- **Particular:** $ 595.
- **Académico:** $ 299.

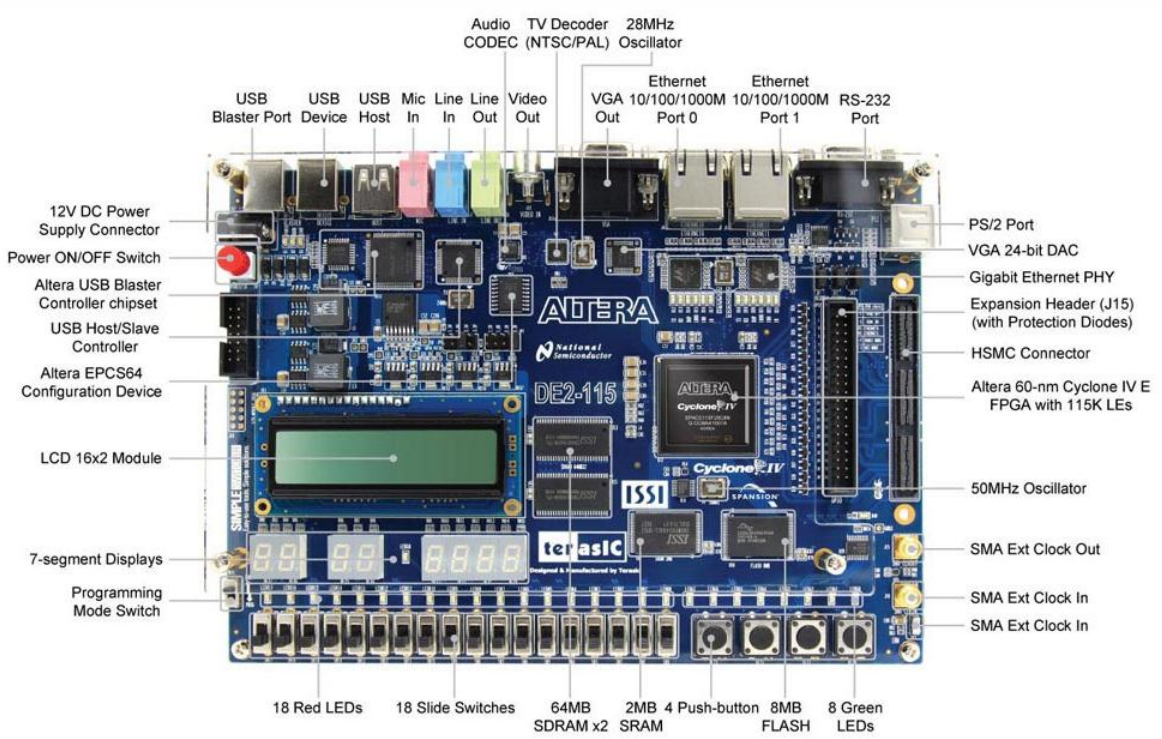


Figura III-16: Tarjeta Educacional DE2-115 de Altera (Vista superior)

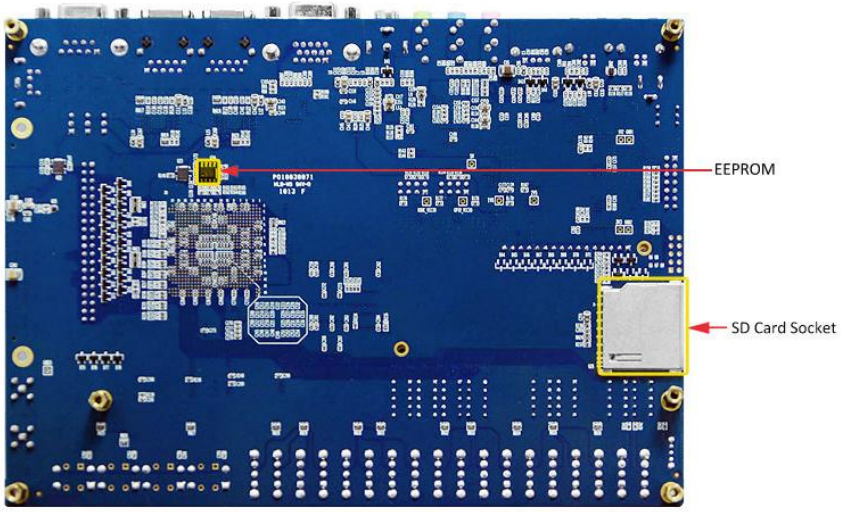


Figura III-17: Tarjeta Educacional DE2-115 de Altera (Vista Inferior)

3.6 ANÁLISIS COMPARATIVO ENTRE LAS TARJETAS XILINX Y ALTERA.

Xilinx y Altera son dos grandes empresas líderes a nivel mundial en la fabricación de dispositivos programables FPGA y en el desarrollo de herramientas CAD, sin embargo dichas empresas han de competir en el sector de las FPGA a nivel mundial, estas dos son las que más sobresalen.

Se ha optado por explicar de forma concisa las particularidades de estas dos empresas al ser estas las que nos ofrecen todas las características y herramientas necesarias para realzar esta tesis, tanto a nivel de hardware con sus FPGA, como a nivel de software con sus herramientas CAD y con sus facilidades a nivel educativo.

Analizando las ofertas de las dos empresas, cabe destacar que todas las tarjetas tienen la capacidad de ser utilizadas bajo el software libre ofrecido por las respectivas compañías, tanto con el software ISE que ofrece la empresa Xilinx, como con la herramienta de desarrollo Quartus II de la empresa Altera.
Las tarjetas básicas de las dos empresas, Basys2 de Xilinx y DE0-Nano de Altera presentan características suficientes para realizar el tema de la tesis, sin embargo al momento de realizar otros diseños en el laboratorio presenta importantes carencias, especialmente a cantidad de memoria, puertos de comunicación e interfaces visuales de salida.

La tarjeta educacional DE0 ofrecida por la compañía Altera, es una placa dirigida a estudiantes para el ensayo y realización de sencillos diseños de apoyo, ofreciendo menores recursos para la realización de un programa de prácticas de laboratorio, al igual que la tarjeta Spartan-3E Starter de Xilinx presenta características semejantes. En cuanto a los dispositivos programables FPGA integrados en las respectivas tarjetas, las familias Cyclone III de la empresa Altera y Spartan 3 por parte de compañía Xilinx, presentan dispositivos con características similares. Ambas familias presentan FPGAs capaces de implementar aplicaciones con alto volumen de datos y alta flexibilidad en los diseños y todo ello a un bajo coste.

La tarjeta DE2-115 de Altera presenta características y recursos muy superiores a los necesarios para la utilización en un laboratorio de prácticas, lo que además conlleva el consecuente aumento de precio de la placa. Al igual que la placa de desarrollo Anvyl Spartan-6 de Xilinx consta de las mismas peculiaridades he incluso superiores, pero al igual su costo es mucho más elevado.

Centrado el análisis en las tarjetas de nivel intermedio como son las DE1, DE2, Spartan-3AN Starter, Spartan-3A Starter y Atlys Spartan-6, obtienen las siguientes conclusiones:

- Tanto los de las tarjetas de Altera como las de Xilinx cuentan con núcleos de procesadores integrados, NIOS II en el caso de FPGAs de Altera y MicroBlaze-PicoBlaze para los FPGAs de Xilinx.
- Las tarjetas de Xilinx ofrecen mayor cantidad de memoria RAM y memoria Flash, si bien las tarjetas de Altera tienen la capacidad de adaptar memorias externas.
- Tanto las tarjetas de Altera y Xilinx cuentan con puertos de comunicación de entrada/salida, puertos de audio y video.
- Las tarjetas de Altera tienen un mayor número de LEDs, Displays, LCD, interruptores y pulsadores, al contrario de las tarjetas Spartan-3A, Spartarn-3AN y la Atlys Spartan-6 que no tienen muchos de estos elementos.
- Las tarjetas de Xilinx ofrecen un precio más reducido que las tarjetas de Altera.

No se ha tomado en cuenta para este análisis la tarjeta Spartan-3A DSP 1800A ya que a más de ser un entrenador de circuitos digitales es una plataforma exclusivamente para el procesamiento digital de señales.

Además las tarjetas educacionales de ambas compañías prestan diversos dispositivos de expansión, a través de distintos productos según la finalidad o aplicación que se le quiera dar.

En este caso, dado que el fin de esta tesis es analizar las diferentes modulaciones digitales y ofrecer al laboratorio un equipo entrenador para el diseño y programación en el ámbito FPGA, que cuente con características y recursos suficientes para dar una facilidad a los alumnos en cuanto a puertos de comunicación de E/S, elementos visuales de salida para el programador y puertos de expansión, por tal motivo se ha escogido para la realización de esta tesis la tarjera de desarrollo educacional DE2 de Altera.

CAPITULO IV

MODULACIÓN DIGITAL

4.1 INTRODUCCIÓN

En este capítulo se conoce lo que son las comunicaciones digitales y las modulaciones digitales, los tipos de modulaciones, sus características y particularidades, una vez comprendidas se proceden al diseño en código VHDL de dichas modulaciones y a su análisis en el simulador.

4.2 INTRODUCCIÓN A LA ELECTRÓNICA DIGITAL

En esencia, las comunicaciones eléctricas son la transmisión, recepción y procesamiento de información usando circuitos electrónicos. Se define a la información como el conocimiento o las señales inteligentes comunicados o recibidos. La figura IV-1 muestra un diagrama simplificado de bloques de un sistema de comunicaciones electrónicos, que comprende tres secciones principales: una fuente, un destino y un medio de transmisión. La información se propaga a través de un sistema de comunicaciones en forma de símbolos que puede ser analógicos (proporcionales), como la voz humana, la información de la imágenes de video, o la música; o bien pueden ser digitales (discretos), como los números codificados en sistema binario, los códigos alfanuméricos, los símbolos gráficos, los códigos de operación de microprocesadores o la información de base de datos. “Sin embargo, y con mucha

frecuencia, la información de la fuente no es adecuada para transmitirse en su forma original y se debe convertir a una forma más adecuada antes de transmitirla. Por ejemplo con los sistemas de comunicaciones digitales, la información analógica se convierte a forma digital antes de transmitirla, y con los sistemas de comunicaciones analógicos, los datos digitales se convierten en señales analógicas antes de transmitirlos.

Los sistemas tradicionales de comunicaciones eléctricas, que usan técnicas convencionales de modulación analógica, como los de modulación de amplitud (AM), modulación de frecuencia (FM) y modulación de fase (PM) se están sustituyendo rápidamente por sistemas de comunicación digital, las modernos, que tienen varias y notables ventajas sobre los sistemas analógicos tradicionales: facilidad de procesamiento, facilidad de multiplexado e inmunidad al ruido.

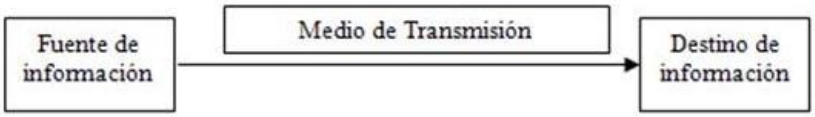


Figura IV-1: Diagrama de bloques simplificado de un sistema de comunicaciones eléctricas

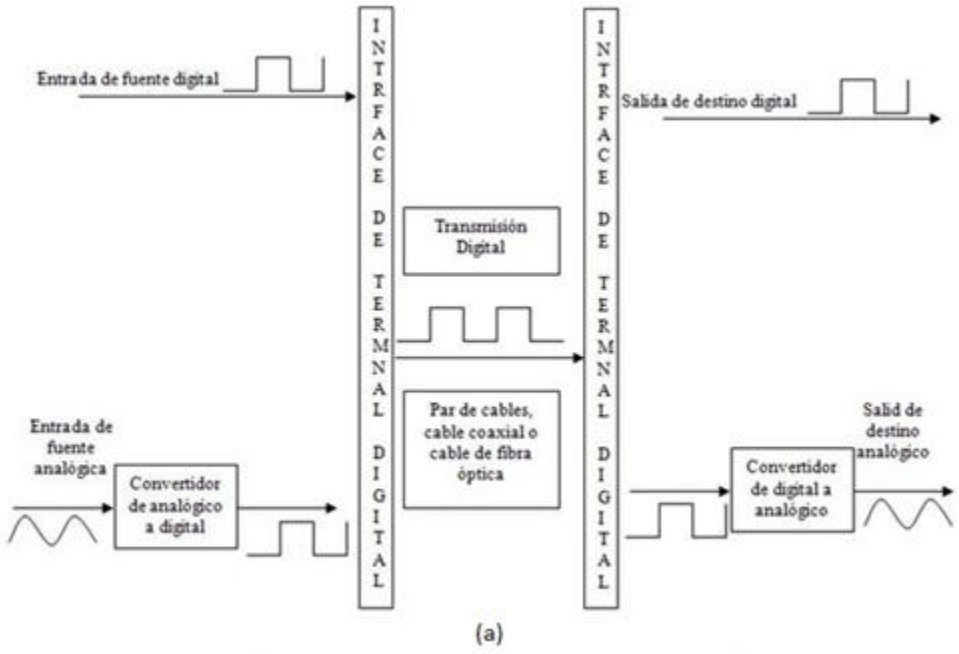
4.3 MODULACIÓN DIGITAL

Los sistemas digitales de comunicación o sistemas de comunicaciones digitales incluyen a aquellos en los que hay portadoras analógicas de frecuencia relativamente alta, que se modulan mediante señales de información digital de relativamente baja frecuencia, y a los sistemas que manejan la transmisión de pulsos.
El termino comunicaciones digitales abarca una gran área de técnicas de comunicaciones, que incluyen la transmisión digital y el (o la) radio digital. Se aplica a la transmisión de pulsos digitales entre dos o más puntos en un sistema de comunicaciones. La radio digital es la transmisión de portadoras analógicas moduladas digitalmente entre dos o más puntos de un sistema de comunicaciones.

Los sistemas digitales de transmisión requieren una instalación física entre el transmisor y el receptor, como un par de hilos metálicos, un cable coaxial o un cable de fibra óptica. En los sistemas digitales de radio, el medio de transmisión podría ser el

espacio libre, la atmosfera terrestre o una instalación física, como un cable metálico o de fibra óptica.

La figura IV-2 muestra diagramas simplificados de bloques de un sistema de transmisiones digitales y uno de radio digital. En el primero, la fuente original de información puede estar en forma digital o analógica. Si esta en forma analógica debe convertirse en pulsos digitales antes de la transmisión, y reconvertirse a la forma analógica en el extremo de recepción. En un sistema digital de radio, la eñal moduladora de entrada y la señal desmodulada de salida son pulsos digitales. Éstos se podrían originar en un sistema digital de transmisión, o en la fuente original digital, como puede ser una computadora central, o bien estas en la codificación binaria de una señal analógica.



(a)

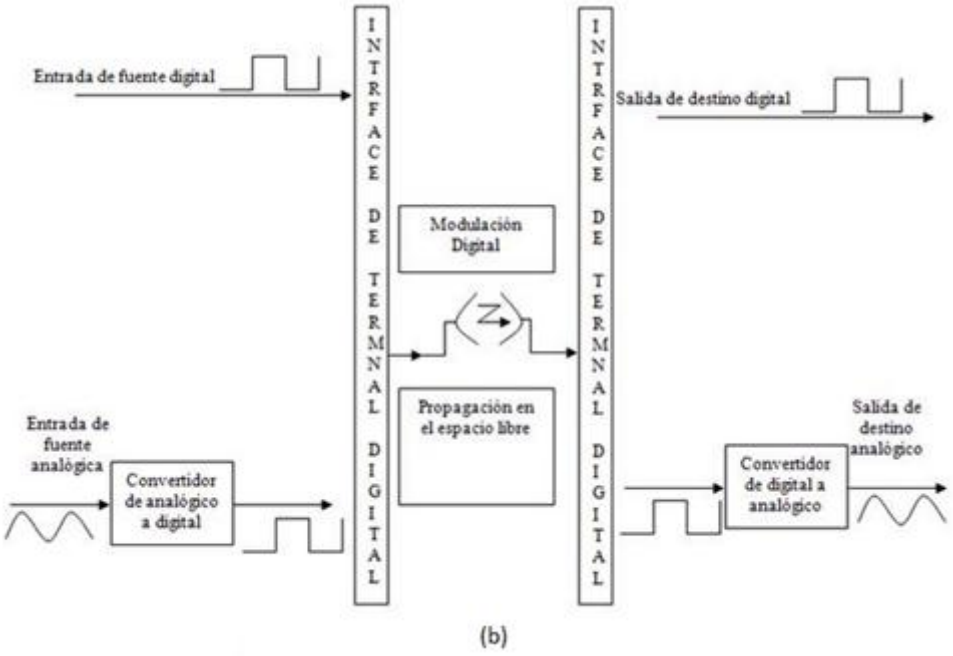


Figura IV-2: Sistema digital de comunicaciones: (a) transmisión digital; (b) radio digital

4.3.1 MODULACIÓN ASK

La técnica de modulación digital más sencilla es la modulación digital de amplitud (ASK, Amplitudes-shift keying), es una modulación de amplitud donde la señal moduladora (datos) es digital. Los dos valores binarios se representan con dos amplitudes diferentes y es usual que una de las dos amplitudes sea cero; es decir uno de los dígitos binarios se representa mediante la presencia de portadora a amplitud constante, y el otro digito se representa mediante la ausencia de la señal portadora.
En este caso la señal moduladora:

$$f(t) = f(t) = \begin{cases} 1 \ \ para\ un\ \ ''1'' binario \\ 0 \ \ para\ un\ \ ''0''\ binario \end{cases}$$

Señal portadora:

$$\mathrm{Cos}\, Wc\ t$$

Señal modulada:

$$f_{ASK}(t) = f(t) Cos\ Wc\ t$$

La señal modulada se obtiene multiplicando las señales, moduladora con la señal portadora, como muestras la gráfica IV-3 siguiente.

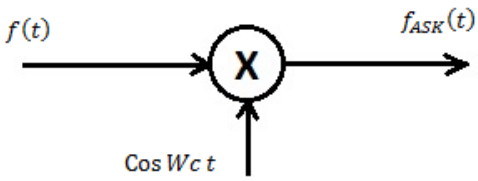


Figura IV-3: Diagrama de bloques del Modulador ASK.

La grafica IV-4 muestra en resumen las señales de entrada al multiplicador, tanto la señal moduladora, como la señal portadora y obteniendo a la salida del multiplicador las señal modulada ASK.

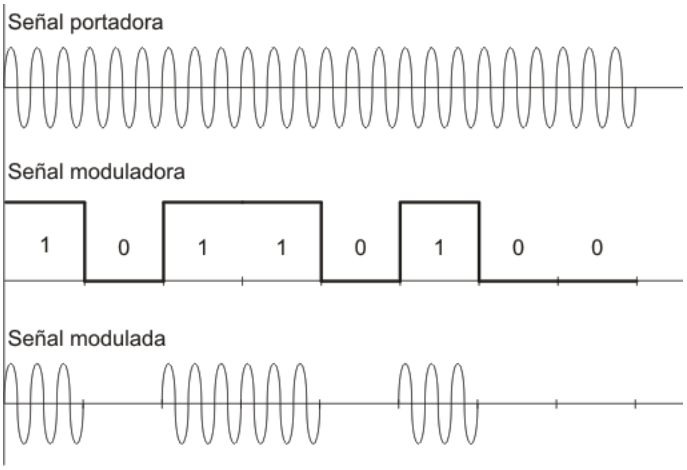


Figura IV-4: Señales de entrada y salida del modulador ASK

La técnica ASK se utiliza para la transmisión de datos digitales en fibras ópticas, en los transmisores con LED, la expresión de la señal modulada sigue siendo válida. Es decir, un elemento de señal se representa mediante un pulso de luz, mientas que el otro se representa mediante la ausencia de luz. Los transmisores laser tienen normalmente un valor de desplazamiento, (vías), que hace que el dispositivo emita una señal de alta intensidad para representar un elemento y una señal de menor amplitud para representar al otro.

4.3.2 MODULACIÓN FSK

La manipulación por desplazamiento de frecuencia (FSK, de frecuency-shift keying) es otro tipo relativamente sencillo y de baja eficiencia de modulación digital. La FSK es una forma de modulación de ángulo, de amplitud constante, parecido a la modulación convencional de frecuencia (FM), pero la señal moduladora es una señal binaria que

varía entre dos valores discretos de voltaje, y no es una forma de onda analógica que cambie continuamente.

Para conseguir una modulación FSK, nos vamos a servir de una señal digital binaria de información: $f(t)$, con unos niveles de tensión de 0 a 1 volt y anchura de bit Tb, como muestra la figura IV-6 (a).
Dos señales portadoras de alta frecuencia, ambas de amplitud A Volt pero con frecuencias diferentes:

Portadora 1: $A.Cos\ W_1\ t$

Portadora 2: $A.Cos\ W_0\ t$

El siguiente diagrama de bloques de la figura IV-5 muestra el proceso que conlleva un modulador FSK:

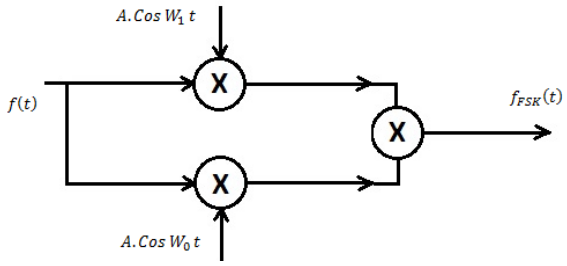


Figura IV-5: Modulador FSK

De esta manera, la función de la señal FSK va a ser:

$$f_{FSK}(t) = \begin{cases} A\ Cos\ W_1\ t & Si\ f(t) = 1\ v. \quad (1\ lógico) \\ A\ Cos\ W_0\ t & Si\ f(t) = 0\ v. \quad (0\ lógico) \end{cases}$$

Como se muestra en la figura IV-6 (b) la señal de salida del modulador será:

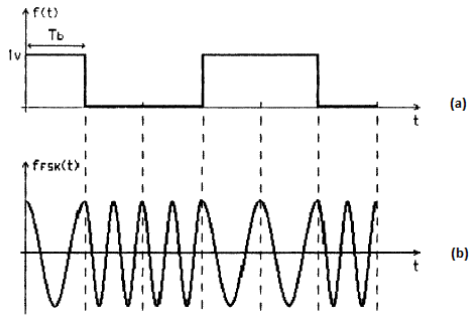


Figura IV-6: Señales de la modulación FSK: (a) Señal binaria de información; (b) Señal modulada FSK

4.3.3 MODULACIÓN M-PSK

La manipulación por desplazamiento de fase (PSK, por phase-shift keying) es otra forma de modulación digital angular de amplitud constante. Se parece a la modulación convencional de fase, excepto que en la PSK la señal de entrada es una señal digital binaria, y es posible tener una cantidad limitada de fases de salida.
PSK o también llamada modulación por desplazamiento binario de fase (BPSK, Binary phase shift keying), es una técnica de modulación digital en la que la información se va a modular en fase, es decir, dependiendo de los valores de la entrada digital, la señal analógica modulación va a tener una u otra fase de salida.

Vamos a considerar una señal binaria que contiene la información $f\ (t)$, a transmitir, con unos niveles de tensión de $\pm$ 1 Volt, y de anchura de bit Tb como muestra la figura IV-8 (a). También consideramos una señal portadora de alta frecuencia $Cos\ W_C\ t$. El siguiente diagrama de bloques de la figura IV-7 muestra el proceso de una modulador PSK.

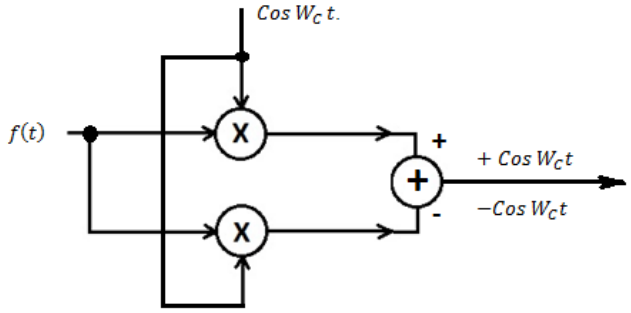


Figura IV-7: Diagrama de bloques del modulador PSK

La función de la señal modulada PSK la definimos como:

$$f_{PSK}(t) = f(t)Cos\ W_C\ t = \begin{cases} +\ Cos\ W_C t & Si\ f(t) = +1\ V. \quad (1\ lógico) \\ -Cos\ W_C t & Si\ f(t) = -1\ V. \quad (0\ lógico) \end{cases}$$

En consecuencia la señal de salida del modulador PSK sera como se muestra en la figura IV-8 (b).

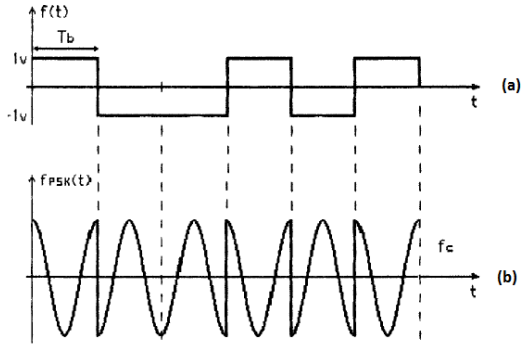


Figura IV-8: Señales de la modulación PSK: (a) Señal binaria de información; (b) Señal modulada PSK

.

El modulador trabaja de la siguiente forma:

Cuando la entrada se corresponde con un 0 lógico, la fase absoluta de salida de la señal PSK es 180º. Si la entrada es un 1 lógico entonces la fase de salida va a ser 0º, como indica la figura IV-9.

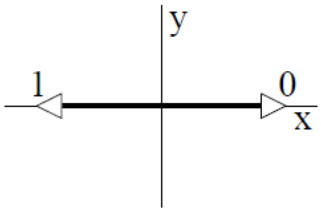


Figura IV-9: Diagrama de constelación BPSK

.

CODIFICACIÓN M-ARIA

M-ario es un término derivado de la palabra binario. M solo es un digito que representa la cantidad de condiciones o combinaciones posibles para determinada cantidad de variables binarias. Las dos técnicas de modulación digital que se han descrito hasta ahora (FSK binaria y BPSK) son sistemas binarios; codifican bits individuales y solo hay dos condiciones posibles de salida. La FSK produce 1 lógico o frecuencia de marca, o un 0 lógico o frecuencia de espacio, y la BPSK produce una fase de 1 lógico o una fase de 0 lógico. Los sistemas FSK y BPSK son M-arios en los que M=2.

MPSK (M-ary phase shift keying) no es más que una extensión de la técnica de modulación digital PSK. Si la señal PSK tenía dos posibles fases de salida (fases absolutas), en MPSK la señal modulada tiene M fases posibles para una misma

frecuencia portadora. Además, igual que ocurría con la modulación PSK, las señales analógicas MPSK tiene una amplitud constante.

Matemáticamente tenemos que:

$$M = 2^k$$

Dónde:

k = número de bits.

M = número de combinaciones de salida posibles de k bits.

Entonces lo que se hace es convertir grupos de k bits de información de información en una señal analógica de amplitud constante y con 2^k fases posibles.

MODULACIÓN QPSK

Quaternary Phase Shift Keying, como se deduce de su nombre, es un tipo de modulación MPSK en la que M=4, es decir la señal portadora de frecuencia W_C puede tener 4 fases de salida diferentes, y por consiguiente k=2 ($4=2^2$). Entonces en QPSK los datos de entrada binarios están compuestos por grupos de 2 bits que reciben el nombre de dibits y que producen 4 posibles combinaciones: 00, 01, 10 y 11.

El circuito de la figura IV-10 es un modulador QPSK que se sirve de una señal portadora $Sen\, W_C t$, y en el que entra una señal digital de información $f(t)$ secuencial con niveles de tensión de $\pm 1\, V$.

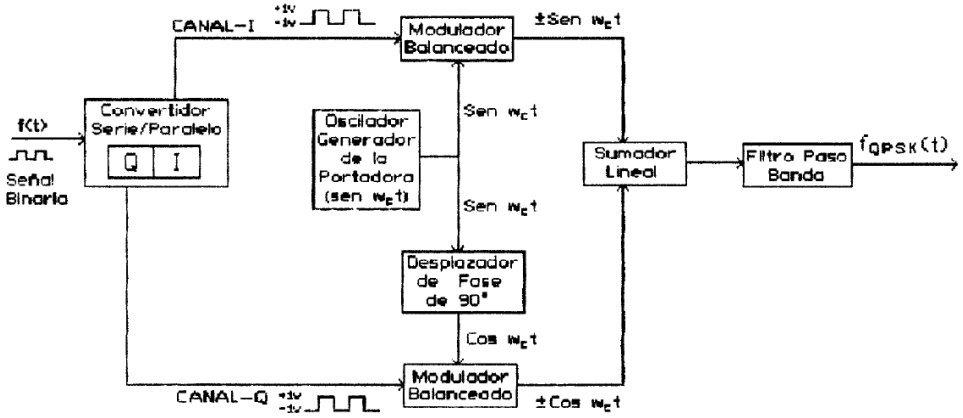


Figura IV-10: Modulador QPSK

En el modulador distinguimos dos canales Canal I y Canal Q, cada uno de los cuales conducirá uno de los bits desde el convertidor serie-paralelo a su modulador balanceado correspondiente, un bit se dirige al canal I y el otro al canal Q, un

modulador balanceado opera igual que un multiplicador analógico. El bit I modula una portadora que está en fase con el oscilador de referencia (de ahí el nombre de ‘‘I’’ para el canal ‘‘en fase’’), y el bit Q modula una portadora que esta 90º fuera de fase o en cuadratura con la portadora de referencia (de ahí su nombre de ‘‘Q’’ para el canal de ‘‘cuadratura’’). El bit I, cuyo nivel de tensión puede ser de -1 ó +1 Volt, es multiplicado por la señal portadora $Sen\, W_C\, t$. Mientras, el bit Q se multiplica por la portadora desplazada en fase 90º ($Cos\, W_C\, t$). La salida de ambos moduladores balanceados se suman linealmente para dar lugar a la señal QPSK.

El filtro paso de banda que se coloca a la salida del modulador QPSK lo que hace es eliminar los armónicos no significativos de la señal modulada para no interferir con otras señales que pudieran transmitirse por ese mismo canal.

Los valores que puede tomar la señal de salida $f_{QPSK}(t)$ son los presentados en la tabla IV-I.

Entrada binaria		$f_{QPSK}(t)$	Fase de salida de la señal QPSK
Q	I		
0	0	$-\cos w_c t - \text{sen } w_c t$	−135°
0	1	$-\cos w_c t + \text{sen } w_c t$	−45°
1	0	$+\cos w_c t - \text{sen } w_c t$	+135°
1	1	$+\cos w_c t + \text{sen } w_c t$	+45°

Tab. IV-I Tabla de verdad de la modulación QPSK

Su diagrama de fasores y su constelación de puntos son los representados en la figura IV-11.

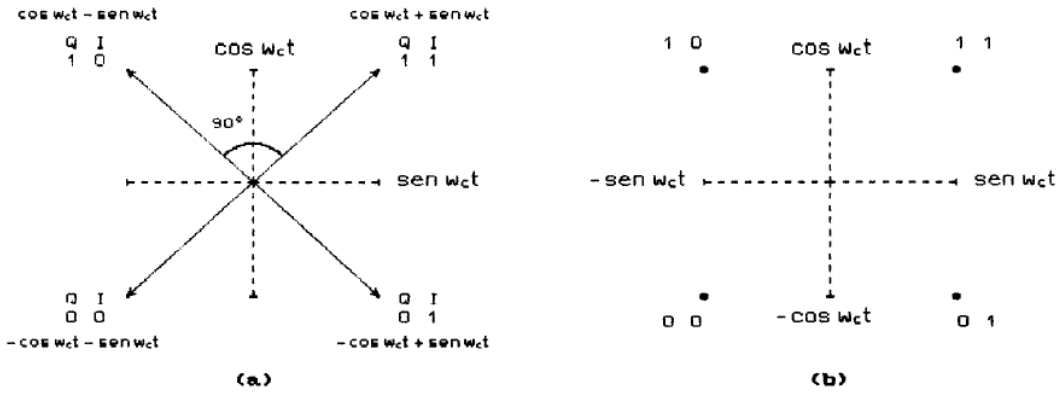


Figura IV-11: Modulación QPSK: (a) Diagrama de fasores; (b) Constelacion de puntos

En la modulación QPSK, como se puede observar en la figura IV-11 (a), la separación angular entre fases de salida adyacentes es de 90º. Además, para este modulador, cada dibit diferente del adyacente en un solo bit. Este sistema de codificación recibe el nombre de Código de Gray.

MODULACIÓN 8PSK

Eight-phase PSK, es una técnica de modulación digital MPSK en la que M=8, siendo entonces k=3 ($8 = 2^3$). Estos grupos de 3 bits de información reciben el nombre de tribits. Asi pues, la señal analógica modulada tendrá 8 posibles fases. En la figura IV-12 se muestra un modulador 8PSK.

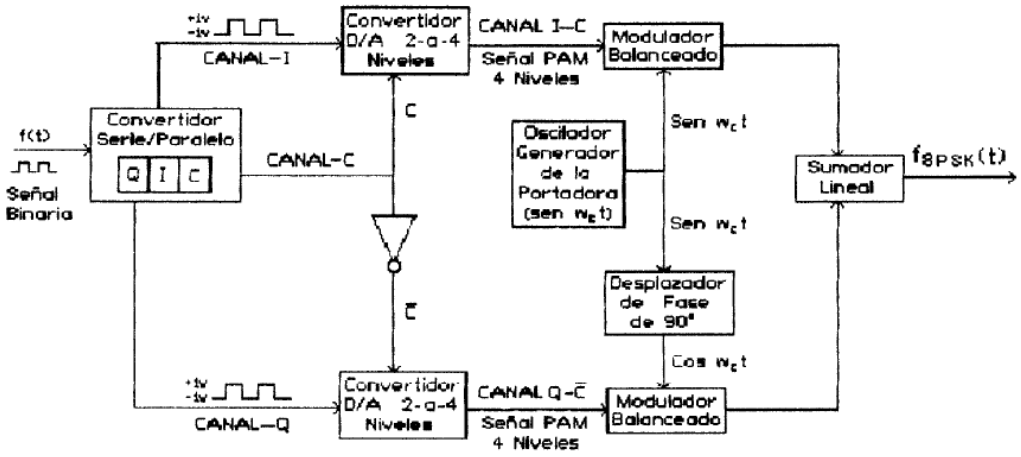


Figura IV-12: Modulador 8PSK

Los convertidores D/A (digital/analógico), convierten dos entrada digitales binarias en una señal analógica PAM de 4 niveles de tensión, como se muestra en la figura IV-13. El circuito que queda a partir de los convertidores es, básicamente, un modulador QPSK.

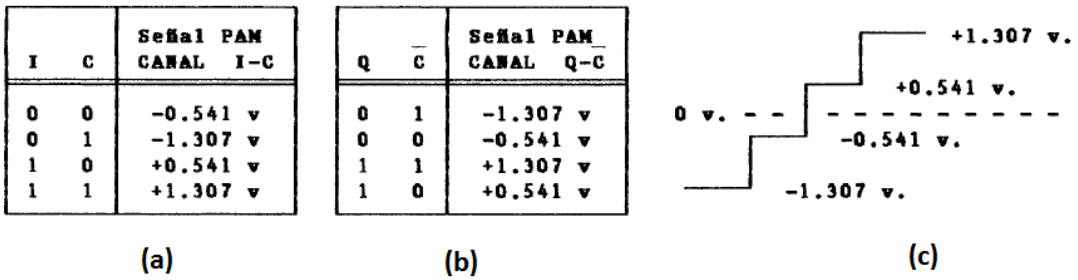


Figura IV-13: (a) Salida convertidor D/A Canal I-C; (b) Salida convertidor D/A Canal Q-/C; (c) Niveles PAM

Las dos señales PAM obtenidas entran en sendos modulares balanceados, que las multiplican por la portadora, la señal PAM del Canal I-C por $Sen\, W_C t$ y la del canal Q-/C por $Cos\, W_C t$. Las salidas de los moduladores balanceados se suman linealmente, dando la señal modulada 8PSK. Los posibles valores que puede alcanzar esta señal aparecen en la tabla IV-II, su diagrama de fasores y su constelación de puntos están representados en la figura IV-14.

Entrada binaria			Señal de salida 8PSK	Fase de salida
Q	I	C	$f_{8PSK}(t)$	de la señal 8PSK
0	0	0	$-1.307 \cos w_c t - 0.541 \text{ sen } w_c t$	-112.5^o
0	0	1	$-0.541 \cos w_c t - 1.307 \text{ sen } w_c t$	-157.5^o
0	1	0	$-1.307 \cos w_c t + 0.541 \text{ sen } w_c t$	-67.5^o
0	1	1	$-0.541 \cos w_c t + 1.307 \text{ sen } w_c t$	-22.5^o
1	0	0	$+1.307 \cos w_c t - 0.541 \text{ sen } w_c t$	$+112.5^o$
1	0	1	$+0.541 \cos w_c t - 1.307 \text{ sen } w_c t$	$+157.5^o$
1	1	0	$+1.307 \cos w_c t + 0.541 \text{ sen } w_c t$	$+67.5^o$
1	1	1	$+0.541 \cos w_c t + 1.307 \text{ sen } w_c t$	$+22.5^o$

Tab. IV-II Tabla de verdad de la modulación 8PSK

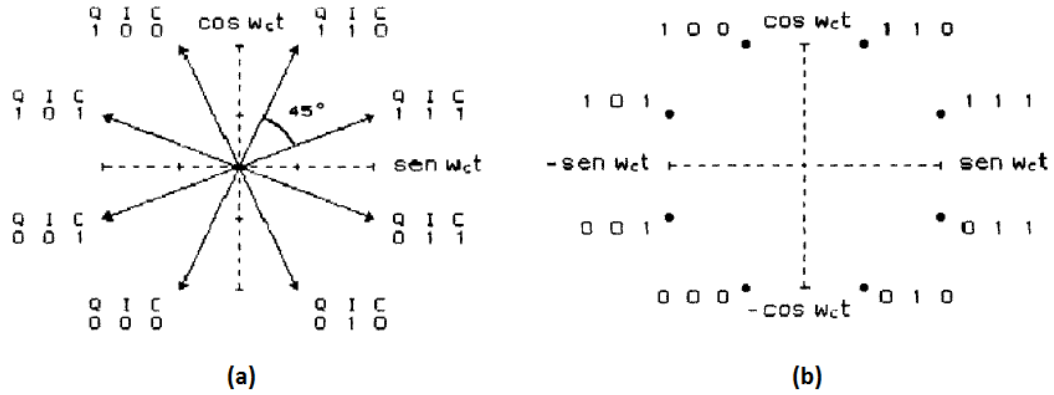


Figura IV-14: Modulación 8PSK: (a) Diagrama de fasores; (b) Constelación de puntos

Al ser una modulación MPSK, la amplitud de la señal modulada $f_{8PSK}(t)$ es constante. Si nos fijamos en la figura IV-14 (b), observamos que cada tribit difiere del adyacente en un solo bit. Cuando ocurre esto se dice que se está utilizando un Código Gray.

Si nos fijamos ahora en la figura IV-14 (a), vemos que la separación angular entre dos fases adyacentes de salida es de 45º, la mitad que en QPSK.

MODULACIÓN 16PSK

Sixteen-phase PSK, es una técnica de modulación digital MPSK en la que M=16 y por tanto existen 16 fases posibles en la señal de salida. El modulador 16PSK trata los datos en grupos de k=4 bits que reciben el nombre de quadbits ($16 = 2^4$). Además como todas las técnicas demodulación MPSK, la amplitud de la señal analógica de salida siempre permanece constante.

La fase de salida no cambiara hasta que los 4 bits al completo hayan entrado en el modulador. En la tabla IV-III se muestra la tabla de verdad de las posibles fases de salida de la señal 16PSK y en la figura IV-15 su constelación de puntos.

Como se puede deducir de la tabla IV-III, la separación angular entre dos fases de salida adyacentes es solo 22’5º, que es la mitad de la existente en 8PSK (45º) y un cuarto de la QPSK (90º). En el siguiente apartado se estudia las consecuencias de esta característica.

Señal binaria	Fase de Salida	Señal binaria	Fase de Salida
0 0 0 0	11.25°	1 0 0 0	191.25°
0 0 0 1	33.75°	1 0 0 1	213.75°
0 0 1 0	56.25°	1 0 1 0	236.25°
0 0 1 1	78.75°	1 0 1 1	258.75°
0 1 0 0	101.25°	1 1 0 0	281.25°
0 1 0 1	123.75°	1 1 0 1	303.75°
0 1 1 0	146.25°	1 1 1 0	326.25°
0 1 1 1	178.75°	1 1 1 1	348.75°

Tab. IV-III Modulación 16PSK - Tabla de verdad de la señal analógica de salida

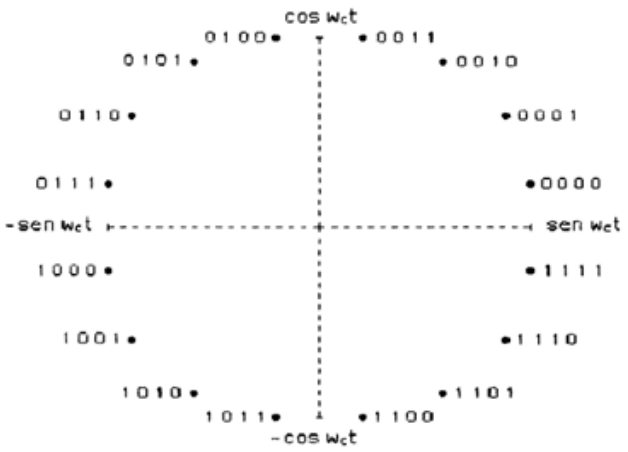


Figura IV-15: Modulación 16PSK - Constelación de puntos

CONSIDERACIONES SOBRE MPSK

El ancho de banda mínimo necesario para transmitir una señal PMSK es igual al necesario para mandar un mensaje modulado en PSK, es decir:

$$B_{MPSK} = 2\,B_F \qquad \text{(Ec.4-1)}$$

Donde, B_F = Ancho de banda de la señal digital de información $f(t)$

Además sabemos que la relación entre la velocidad de transmisión en baudios (símbolos/seg) en el caso ideal, velocidad de Nyquist, y el ancho de banda es:

$$V_S = 2\,B_F \qquad \text{(Ec.4-2)}$$

Para la modulación MPSK hemos visto que en un símbolo se transmiten k bits de información, por consiguiente:

$$V_b = k\,V_S \qquad \text{(Ec. 4-3)}$$

Donde V_b es la velocidad de transmisión en bits por segundo (bps).

Al sustituir la Ec.4-2 y Ec.4-3 en la Ec.4-1 obtenemos que:

$$B_{MPSK} = 2B_F = 2\frac{V_S}{2} = V_S = \frac{V_b}{k}$$

Hemos llegado a la conclusión de que el ancho de banda mínimo necesario para transmitir una señal MPSK es V_b/k, siendo Vb a su vez la velocidad de entrada (bps) en el modulador de los bits de la señal digital a transmitir. Entonces sí, para un canal con ancho de banda B_{MPSK}, la velocidad a la que se puede transmitir un mensaje es Vs, en el caso ideal, la velocidad en bps a la que se transmitirá será $V_b = kV_S$.

Este tipo de modulación está orientada a conseguir que, para un canal con un ancho de banda determinado, la velocidad en bps a la que se pueda transmitir sea mayor que la conseguida para ese mismo canal con otros tipos de modulación, como por ejemplo ASK, FSK o PSK.

Ahora podríamos preguntarnos cual de entre todas las modulaciones MPSK es la que consigue una mayor inmunidad al ruido. Cuanto menor sea M ($M = 2^k$), menos afectado por el ruido se verá el sistema. Esto se puede demostrar fácilmente ya que para un sistema QPSK, con separación angular 90º, la señal modulada puede experimentar un desplazamiento de fase (debido al ruido) de casi $\pm 45°$ durante la transmisión, y todavía guardar la integridad de la información. Entonces, cuanto menor

sea la separación entre fases de salida adyacentes, menor será el margen de confianza y por tanto menos inmune al ruido. Si recordamos 8PSK tenía una separación entre fases de 45º y para 16PSK era de 22'5º, así que sus márgenes de confianza quedan reducidas $\pm 22'5°$ y $\pm 11'25°$ respectivamente.

Por ultimo conviene señalar que si los códigos de k-bits (dibits, tribits, quadbits, etc) de dos fases de salida adyacentes difieren entre sí en un solo bit, disminuye el número de errores en la transmisión, números de bits erróneos. Si una señal experimenta un desplazamiento de fase durante la transmisión, debido al ruido, hasta una fase adyacente, el error se produciría en solo uno de los bits recibidos. Este tipo de código recibe el nombre de Código de Gray o Código de distancia máxima. Ejemplos de estos códigos son los mostrados en la figura IV-11 (b) y figura IV-14 (b).

4.3.4 MODULACIÓN M-QAM.

La modulación de amplitud en cuadradura (QAM, por quadrature amplitude modulation) es una forma de modulación digital, donde la información digital está contenida tanto en la amplitud como en la fase de la portadora transmitida.

Eight QAM es una técnica de codificación M-aria en la que M=8 y por consiguiente k=3 ($8 = 2^3$). Entonces en 8QAM, los datos de entrada binarios están compuestos por grupos de 3 bits que reciben el nombre de tribits.

En la figura IV-16 aparece un circuito modulador 8QAM. El convertidor D/A (digital/analógico) empleado, transforma 2 entradas digitales binarias en una señal analógica PAM de 4 niveles de tensión. La tabla de verdad del convertidor es la que se muestra en la tabla IV-IV.

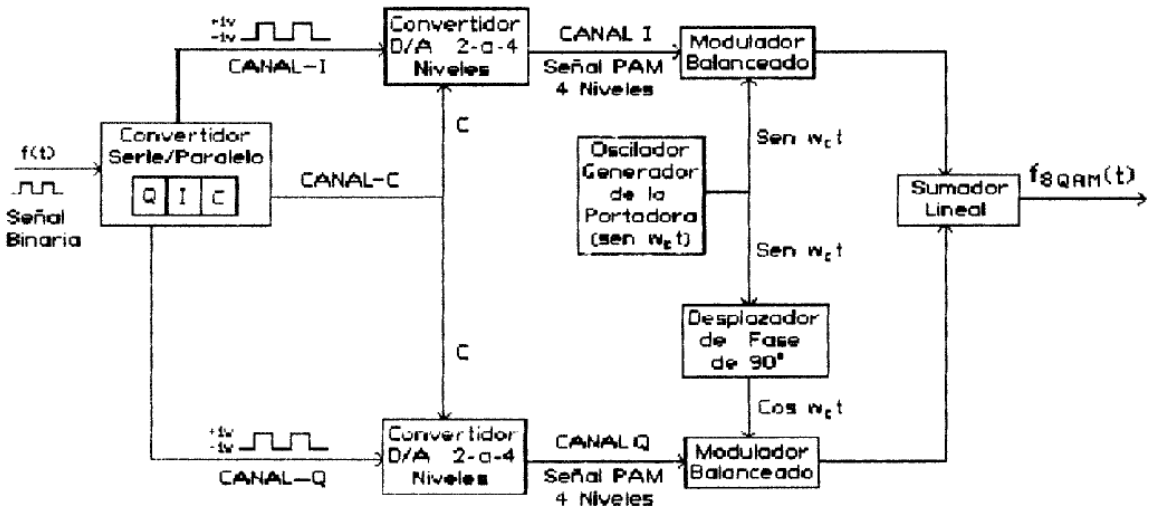


Figura IV-16: Modulador 8QAM

I/Q	C	Señal PAM de Salida
0	0	−0.541 v
0	1	−1.307 v
1	0	+0.541 v
1	1	+1.307 v

Tab. IV-IV Tabla de verdad del convertidor D/A 2-a-4 niveles

Los valores que puede tomar la señal moduladora 8QAM aparecen en la tabla IV-V. El diagrama de fasores y la constelación de puntos de esta técnica de modulación están representados en la figura IV-17 (a) y figura IV-17 (b) respectivamente.

Entrada binaria			Señal de salida 8PSK	Señal de salida 8QAM	
Q	I	C	$f_{8QAM}(t)$	Amplitud	Fase
0	0	0	$-0.541\cos w_c t - 0.541 \operatorname{sen} w_c t$	0.765 v	-135^o
0	0	1	$-1.307\cos w_c t - 1.307 \operatorname{sen} w_c t$	1.848 v	-135^o
0	1	0	$-0.541\cos w_c t + 0.541 \operatorname{sen} w_c t$	0.765 v	-45^o
0	1	1	$-1.307\cos w_c t + 1.307 \operatorname{sen} w_c t$	1.848 v	-45^o
1	0	0	$+0.541\cos w_c t - 0.541 \operatorname{sen} w_c t$	0.765 v	$+135^o$
1	0	1	$+1.307\cos w_c t - 1.307 \operatorname{sen} w_c t$	1.848 v	$+135^o$
1	1	0	$+0.541\cos w_c t + 0.541 \operatorname{sen} w_c t$	0.765 v	$+45^o$
1	1	1	$+1.307\cos w_c t + 1.307 \operatorname{sen} w_c t$	1.848 v	$+45^o$

Tab. IV-V Tabla de verdad de la modulación 8QAM

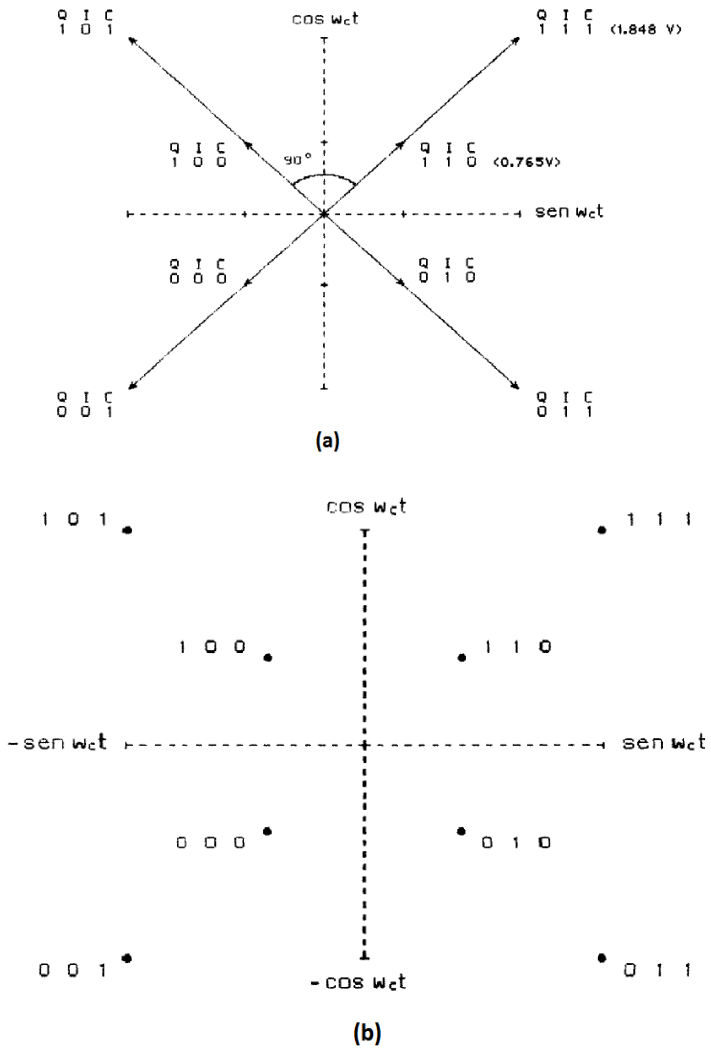


Figura IV-17: Modulación 8QAM: (a) Diagrama de fasores; (b) Constelación de puntos

En la figura IV-17 se comprueba que, es este caso, se ha modulado tanto en amplitud como en fase. Además, hemos conseguido que la separación angular entre dos fases de salida adyacentes sea de 90º, y que para salidas con igual fase, por ejemplo 110 y 111, exista una diferencia de amplitud de 1’083 V, margen bastante considerable. Comparándola con la modulación 8PSK, se observa un mejor comportamiento frente al ruido. Esto se puede demostrar: 8QAM al tener una separación angular entre fases adyacentes de 90º, permite un margen de confianza de $\pm 45°$, en cambio para 8PSK la separación minima entre dos fases posibles, fases adyacentes, es de 45º, por lo que su margen de confianza queda reducido a $\pm 22'5°$. De esta manera, la señal 8QAM puede experimentar un desplazamiento de fase debido al ruido de casi $\pm 45°$ durante la transmisión y guardar todavía la integridad de la información, mientras que 8PSK solo soporta ruidos cuyos desplazamiento de fase no superan los $\pm 22'5°$.

También se puede advertir en la figura IV-17 (b) que este modulador utiliza un Código de Gray, ya que cada tribit difiere del adyacente en un solo bit.

MODULACIÓN 16QAM

Sixteen QAM, técnica de modulación QAM en la que M=16. Asi pues, el modulador 16QAM trata los datos de entrada en grupos de k=4 bits que reciben el nombre de quadbits ($16 = 2^4$). En la figura IV-18 aparece un circuito modulador 16QAM.

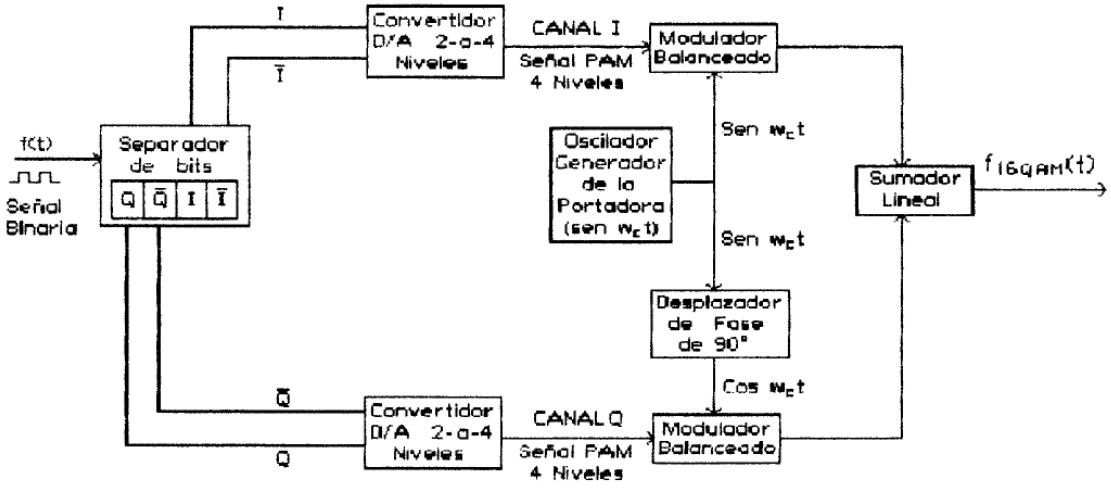


Figura IV-18: Modulador 16QAM

Una vez que entran los 4 bits de información de la señal binaria, se separan en dos grupos de 2 bits cada uno, canal I: I-/I, canal Q: Q-/Q. estos bits entran en un convertidor D/A que transforma 2 entradas digitales en una señal analógica PAM de 4 niveles de tensión (tal como se muestra en las tablas IV-VI (a) y (b)). Los moduladores balanceados son multiplicadores analógicos, cuya salida es el producto de las señales de entrada. El modulador de arriba multiplica la señal PAM del canal I por la portadora $Sen\, W_C t$, mientras que el de abajo multiplica la señal PAM del canal Q por $Cos\, W_C t$, portadora desplazada en fase 90º.

Posteriormente ambas señales resultantes se suman linealmente para dar lugar a la señal de 16QAM, $f_{16QAM}(t)$.

I	$\overline{I}$	Señal PAM CANAL I
0	0	-0.220 v
0	1	-0.821 v
1	0	+0.220 v
1	1	+0.821 v

(a)

Q	$\overline{Q}$	Señal PAM CANAL Q
0	0	-0.220 v
0	1	-0.821 v
1	0	+0.220 v
1	1	+0.821 v

(b)

Tab. IV-VI Tabla de verdad de los convertidores D/A de 2-a-4 niveles: (a) Canal I; (b) Canal Q

En la tabla IV-VII aparecen todas las posibles salidas de la señal 16QAM para sus respectivos quadbits de entrada.

Entrada binaria				Señal de salida 8PSK	Señal de salida 8QAM	
Q	Q	I	I	$f_{16QAM}(t)$	Amplitud	Fase
0	0	0	0	$-0.220\cos w_c t - 0.220 \operatorname{sen} w_c t$	0.311 v	-135^o
0	0	0	1	$-0.220\cos w_c t - 0.821 \operatorname{sen} w_c t$	0.850 v	-165^o
0	0	1	0	$-0.220\cos w_c t + 0.220 \operatorname{sen} w_c t$	0.311 v	-45^o
0	0	1	1	$-0.220\cos w_c t + 0.821 \operatorname{sen} w_c t$	0.850 v	-15^o
0	1	0	0	$-0.821\cos w_c t - 0.220 \operatorname{sen} w_c t$	0.850 v	-105^o
0	1	0	1	$-0.821\cos w_c t - 0.821 \operatorname{sen} w_c t$	1.161 v	-135^o
0	1	1	0	$-0.821\cos w_c t + 0.220 \operatorname{sen} w_c t$	0.850 v	-75^o
0	1	1	1	$-0.821\cos w_c t + 0.821 \operatorname{sen} w_c t$	1.161 v	-45^o
1	0	0	0	$+0.220\cos w_c t - 0.220 \operatorname{sen} w_c t$	0.311 v	$+135^o$
1	0	0	1	$+0.220\cos w_c t - 0.821 \operatorname{sen} w_c t$	0.850 v	$+165^o$
1	0	1	0	$+0.220\cos w_c t + 0.220 \operatorname{sen} w_c t$	0.311 v	$+45^o$
1	0	1	1	$+0.220\cos w_c t + 0.821 \operatorname{sen} w_c t$	0.850 v	$+15^o$
1	1	0	0	$+0.821\cos w_c t - 0.220 \operatorname{sen} w_c t$	0.850 v	$+105^o$
1	1	0	1	$+0.821\cos w_c t - 0.821 \operatorname{sen} w_c t$	1.161 v	$+135^o$
1	1	1	0	$+0.821\cos w_c t + 0.220 \operatorname{sen} w_c t$	0.850 v	$+75^o$
1	1	1	1	$+0.821\cos w_c t + 0.821 \operatorname{sen} w_c t$	1.161 v	$+45^o$

Tab. IV-VII Tabla de verdad de la modulación 16QAM

El diagrama de fasores y la constelación de puntos de esta tipo de modulación están representados en la figura IV-19. Si nos fijamos en la figura IV-19 (a), comprobamos que la separación angular entre dos fases de salida adyacentes es de 30º, que frente a los 22’5º de la modulación 16PSK hacen que 16QAM presente una mayor inmunidad al ruido. Además, el diagrama de la constelación de puntos nos muestra que este modulador utiliza un Codigo Gray.

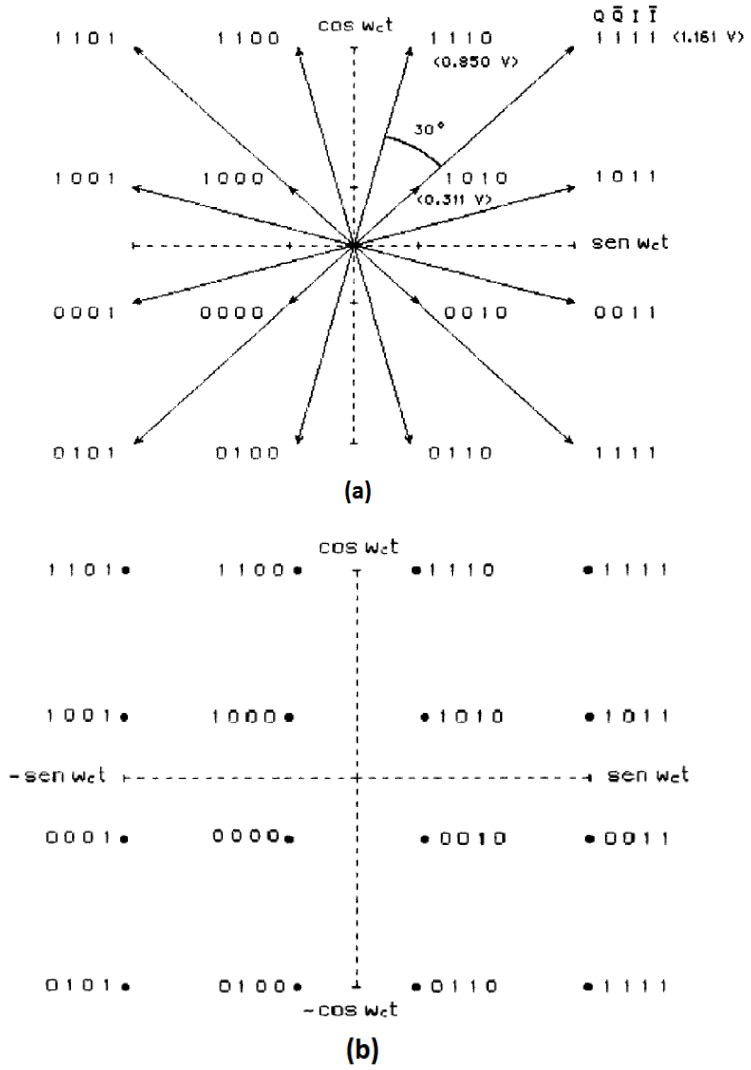


Figura IV-19: Modulación 16QAM: (a) Diagrama de fasores; (b) Constelación de puntos

CONSIDERACIONES SOBRE QAM

El ancho de banda mínimo necesario para transmitir una señal QAM es el doble del ocupado por la señal digital original que porta la información.

$$B_{QAM} = 2B_F \quad \text{(Ec.5-1)}$$

Donde, B_F = ancho de banda de la señal digital de información f(t).

Además sabemos que la velocidad de Nyquist, velocidad de transmisión en baudios (símbolos/seg) en el caso ideal, es:

$$V_S = 2B_F \qquad \text{(Ec.5-2)}$$

En la modulación QAM hemos visto que en un símbolo se transmiten k bits de información ($M = 2^k$), por lo que:

$$V_b = kV_S \qquad \text{(Ec.5-3)}$$

Donde V_b es la velocidad de transmisión en los bits por segundo (bps).

Si sustituimos las Ec.5-2 y Ec.5-3 en la Ec.5-1 tenemos:

$$B_{QAM} = 2B_F = 2\frac{V_S}{2} = V_S = \frac{V_b}{k}$$

Así pues, el ancho de banda mínimo necesario para transmitir una señal QAM es V_b/k, siendo Vb la velocidad (bps) a la que entran los bits de la señal digital de información a transmitir en el modulador. Entonces sí, para un canal con ancho de banda B_{QAM}, la velocidad a la que se puede transmitir un mensaje es Vs, en el caso ideal, la velocidad en bps a la que se transmitirá es $V_b = kV_S$.

Este tipo de modulación, al igual que MPSK, está orientada a conseguir que, para un canal con un ancho de banda determinado, la velocidad en bps a la que se pueda transmitir sea mayor que la conseguida pata ese mismo canal con otros tipos de modulación, como por ejemplo ASK, FSK o PSK.

La utilización de un Código Gray, también llamado de Distancia Máxima, es preferible ya que disminuye el número de errores en la transmisión, número de bits erróneos. Si debido al ruido una señal experimenta un desplazamiento en su fase hasta una fase adyacente, el error producido afectaría a uno solo de los bits recibidos.

Comparando entre si los distintos sistemas QAM se tiene que cuando menor es M, mejor comportamiento frente al ruido presenta. Veamos los sistemas 8QAM y 16QAM. La separación angular en 8QAM entre dos fases de salida adyacentes es 90º por lo que la señal modulada puede sufrir un desplazamiento de fase de casi $\pm 45°$ (debido al ruido) durante la transmisión, y guardar la integridad de la información. En cambio para 16QAM el margen de confianza se reduce a $\pm 15°$, al ser 30º su separación mínima entre fases adyacentes.

4.4 DISEÑO Y PROGRAMACIÓN EN CÓDIGO VHDL.

La disponibilidad de herramientas basadas en computadoras influyó enormemente en el proceso de diseño en numerosos entornos. Por ejemplo, el enfoque global del

diseño de un automóvil es similar al de un mueble o una computadora. Hay que seguir ciertos pasos en el ciclo de desarrollo si el producto final ha de satisfacer los objetivos establecidos. A continuación se presenta un ciclo de desarrollo típico en los términos más generales. El diagrama de la figura IV-20 bosqueja un proceso de desarrollo típico. Supóngase que se trata de desarrollar un producto que satisfaga ciertas expectativas. Los requisitos más obvios son que el producto tiene que funcionar de manera adecuada, ha de satisfacer un nivel de desempeño esperado.

El proceso empieza con la definición de las especificaciones del producto. Se identifican sus características esenciales y se establecen un método aceptable para evaluarlas una vez implementadas en el producto final. Las especificaciones han de ser lo suficientemente concretas para garantizar que el producto desarrollado satisfará las expectativas generales, pero no innecesariamente restrictivas (es decir, no deben impedir opciones de diseño que puedan conducir a ventajas imprevistas).
A partir de un conjunto completo de especificaciones es necesario definir la estructura general de un diseño inicial del producto. Este paso es difícil de automatizar. Suele realizarlo un diseñador humano porque no existe una estrategia obvia para desarrollar la estructura global de un producto: se requiere considerable experiencia e intuición en el diseño.

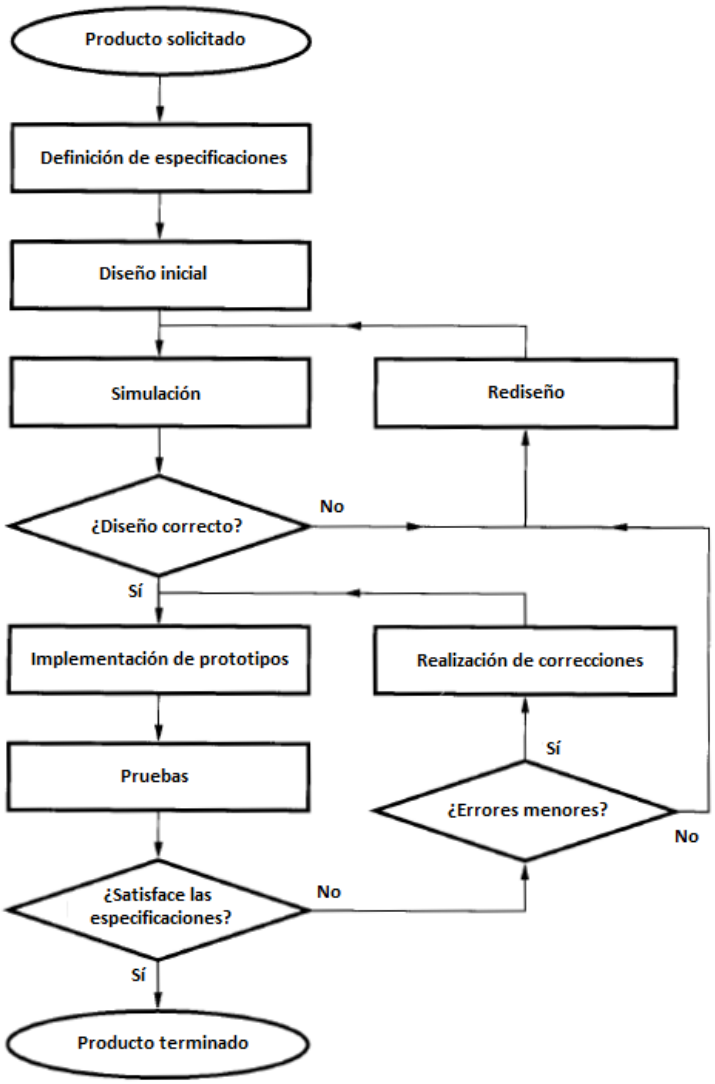


Figura IV-20: El proceso de desarrollo

Después de establecer la estructura general, se usa el software para afinar los detalles. Una vez concluido el diseño inicial los resultados se confrontan con las especificaciones originales. El software permite a los diseñadores simular el comportamiento de productos increíblemente complejos y tales simulaciones se usan para determinar si el diseño obtenido satisface las especificaciones requeridas. Si se encuentran errores, entonces se realizan los cambios adecuados y se repite la verificación del nuevo diseño mediante simulación. Aunque algunos errores de diseño pueden escapar de la detección mediante la simulación, así todos los problemas se descubren de esta forma, salvo los más sutiles.

Cuando la simulación indica que el diseño es correcto se construye un prototipo físico completo del producto. El prototipo se pone a prueba de manera rigurosa para

comprobar su conformidad con las especificaciones. Cualesquiera errores revelados en las pruebas han de corregirse. Los errores pueden ser menores y con frecuencia es posible eliminarlos con pequeñas enmiendas directas en el prototipo del producto. En caso de grandes errores es preciso rediseñar el producto y repetir los pasos entes explicados. Cuando el prototipo pasa todas las pruebas, el producto se juzga bien diseñado y puede irse a producción.

El proceso de diseño se lo realiza con VHDL que es un lenguaje complejo y refinado. Aunque aprender todas sus funciones es una tarea atemorizante, para usarlo en la síntesis basta conocer algunas de ellas.

En el decenio de 1980, los rápidos avances en la tecnología de los circuitos integrados impulsaron el desarrollo de prácticas estándar de diseño para los circuitos digitales. VHDL se creó como parte de tal esfuerzo y se convirtió en el lenguaje estándar industrial para describir circuitos digitales, principalmente porque es un estándar oficial de la IEEE. En 1987 se adoptó la norma original para VHDL, llamada IEEE 1076. En 1993 se adoptó una norma revisada, la IEEE 1164.

En sus orígenes, VHDL tenía dos propósitos centrales. Primero, servía como lenguaje de documentación para describir la estructura de circuitos digitales complejos. Como estándar oficial del IEEE, ofreció una forma común de documentar los circuitos diseñados por varias personas. Segundo, VHDL aportó funciones para modelar el comportamiento de un circuito digital, lo que permitió emplearlo como entrada para programas que entonces se usaban para simular la operación del circuito.

En años recientes, aparte de usarlo para documentación y simulación, VHDL también se volvió popular para el ingreso de diseño en sistemas CAD. Las herramientas CAD se utilizan para sintetizar el código de VHDL en una implementación de hardware del circuito descrito. El compilador de VHDL traduce ese código en un circuito lógico. Cada señal lógica del circuito se representa en el código de VHDL como un objeto de datos. Así como las variables declaradas en cualquier lenguaje de programación de alto nivel tienen tipos asociados, enteros o caracteres, por ejemplo los objetos de datos en VHDL pueden ser de varios tipos. La norma original de VHDL, la IEEE 1076, incluye un tipo de datos llamado BIT. Un objeto de este tipo es adecuado para representar señales digitales, pues solo puede tener dos valores, 0 y 1.

4.4.1 INGRESO DEL DISEÑO

Cualquier proceso de diseño comprende una secuencia básica de tareas que se efectuaran en varias situaciones, como se muestra en la figura IV-21. Si se supone que se tiene un concepto primario acerca de lo que hay que lograr en el proceso de diseño final, el primer paso consiste en generar un diseño inicial para al terminar obtener el un diseño general final.

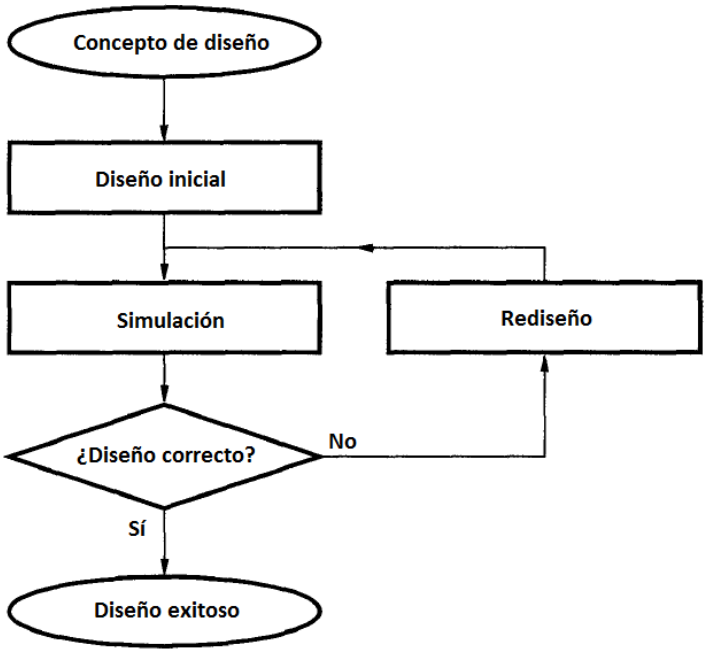


Figura IV-21: Ciclo de diseño básico

La síntesis en el proceso por el que se genera un circuito lógico a partir de una especificación inicial que puede proporcionarse en forma de diagrama esquemático o de código escrito en un lenguaje de descripción de hardware. Con base en esa especificación las herramientas CAD de síntesis generan implementaciones eficientes de circuitos.

El proceso de traducción, o compilación, del código de VHDL en un circuito de compuertas lógicas forma parte de la síntesis. La salida es un conjunto de expresiones lógicas que describen las funciones lógicas necesarias para realizar el circuito.

Sin importar el tipo de ingreso de diseño que se use, las expresiones lógicas iniciales producidas por las herramientas de síntesis no tendrán una forma óptima, ya que reflejan lo que el diseñador ingresa en las herramientas CAD. Es imposible que un diseñador produzca manualmente resultados óptimos para circuitos grandes. Por

ende, una de las tareas importantes de las herramientas de síntesis es manipular el diseño del usuario a fin de generar de manera automática un circuito equivalente, pero mejor.
El desempeño de un circuito sintetizado puede evaluarse construyendo y probando físicamente el circuito, pero también mediante la simulación.

GENERADOR DE SEÑAL SINUSOIDAL

Fue necesario implementar un generador de onda sinusoidal pues este es un elemento esencial para generar todas las modulaciones digitales, para lo cual se tuvo que digitalizar la señal analógica a razón de que un FPGA solo puede generar señales de tipo lógico, es decir estado alto y bajo.

Para digitaliza una señal sinusoidal se debe seguir un proceso, el cual consta de muestrear una señal en diferentes puntos de la onda a intervalos regulares de tiempo, dichas muestras se convierten a valores discretos pre establecidos según el código utilizado, en este caso utilizaremos un código binario.

Conociendo la función que gobierna a la señal sinusoidal seno y utilizando la herramienta CAD Matlab nos es posible realizar un programa en el cual se tomó 50 muestras a un periodo de tiempo con una resolución de 1 byte (8 bits), así obteniendo de esta manera una tabla de 8 bits con 50 muestras, generando así la onda seno digitalmente.
En el anexo A se encuentra el código y la simulación en Matlab para calcular los valores decimales para 50 muestras, los mismos que deben ser convertidos a números binarios.

El código en VHDL de este generador de onda se implementó como una librería para ser llamada en cada modulación digital y se lo describe a continuación:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity osc is
        port (
                contador : in natural;
                onda : out std_logic_vector (7 downto 0));
end osc;
architecture gen_osc of osc is
```

```
signal seno : std_logic_vector(7 downto 0);
begin
    --  tabla de la señal con 50 muestras.
        p1: process(contador)
                begin
                case (contador) is
                        when 0  => seno <= "01111111";
                        when 1  => seno <= "10001111";
                        when 2  => seno <= "10011111";
                        when 3  => seno <= "10101110";
                        when 4  => seno <= "10111101";
                        when 5  => seno <= "11001010";
                        when 6  => seno <= "11010111";
                        when 7  => seno <= "11100010";
                        when 8  => seno <= "11101011";
                        when 9  => seno <= "11110011";
                        when 10 => seno <= "11111001";
                        when 11 => seno <= "11111101";
                        when 12 => seno <= "11111111";
                        when 13 => seno <= "11111111";
                        when 14 => seno <= "11111101";
                        when 15 => seno <= "11111001";
                        when 16 => seno <= "11110011";
                        when 17 => seno <= "11101011";
                        when 18 => seno <= "11100010";
                        when 19 => seno <= "11010111";
                        when 20 => seno <= "11001010";
                        when 21 => seno <= "10111101";
                        when 22 => seno <= "10101110";
                        when 23 => seno <= "10011111";
                        when 24 => seno <= "10001111";
                        when 25 => seno <= "01111111";
                        when 26 => seno <= "01101111";
                        when 27 => seno <= "01100000";
                        when 28 => seno <= "01010000";
                        when 29 => seno <= "01000010";
                        when 30 => seno <= "00110100";
                        when 31 => seno <= "00101000";
                        when 32 => seno <= "00011101";
                        when 33 => seno <= "00010100";
                        when 34 => seno <= "00001100";
                        when 35 => seno <= "00000110";
                        when 36 => seno <= "00000010";
                        when 37 => seno <= "00000000";
                        when 38 => seno <= "00000000";
                        when 39 => seno <= "00000010";
                        when 40 => seno <= "00000110";
                        when 41 => seno <= "00001100";
                        when 42 => seno <= "00010100";
                        when 43 => seno <= "00011101";
                        when 44 => seno <= "00101000";
                        when 45 => seno <= "00110100";
                        when 46 => seno <= "01000010";
                        when 47 => seno <= "01010000";
                        when 48 => seno <= "01100000";
```

```
                when 49 => seno <= "01101111";
                when others => seno <=  "00000000";
            end case;
            onda <= seno;
        end process;
end gen_osc;
```

La frecuencia de la señal de salida está dada por la frecuencia del reloj dividida por el número de muestras que tomamos:

$$f_{señal} = \frac{f_{clk}}{\#\ de\ muestras}$$

$$f_{señal} = \frac{50\ MHz}{50}$$

$$f_{señal} = 1\text{MHz}$$

A continuación se describe el código en VHDL de cada una de las modulaciones digitales:

MODULACIÓN ASK EN CÓDIGO VHDL

```
library ieee;
library libreria;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use libreria.osc.osc;

entity ask is
port (
        clk : in std_logic;
        reset : in std_logic;
        info : in std_logic;
        out_ask : out std_logic_vector (7 downto 0));
end ask;

architecture gen_ask of ask is
signal salida : std_logic_vector(7 downto 0);
signal contador : natural;
signal f : integer:=1;
begin

        p2: process(clk, reset)
                begin
                        if (reset='1') then
                                contador <= 0;
                        elsif (clk'event and clk='1') then
                                if (contador = 49) then
                                        contador <= 0;
                                else
```

```
                                        contador <= contador + f;
                                end if;
                        end if;
                end process;
        u1: osc port map(contador, salida);
         with info select
         out_ask <= "00000000" when '0',
                        salida when others;
end gen_ask;
```

MODULADOR FSK EN CÓDIGO VHDL

```
library ieee;
library libreria;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use libreria.osc.osc;

entity fsk is
port(
        clk, reset : in std_logic;
        info : in std_logic;
        out_fsk : out std_logic_vector(7 downto 0));
end fsk;

architecture gen_fsk of fsk is
signal contador : natural;
begin
        p1: process(clk, reset, info)
        variable freq1: natural:=1;
        variable freq2: natural:=2;
                begin
                        if (reset='1') then
                                contador <= 0;
                        elsif (clk'event and clk='1') then
                                if (contador = 49) or (contador = 48) then
                                        contador <= 0;
                                elsif (info = '0') then
                                        contador <= contador + freq1;
                                else
                                        contador <= contador + freq2;
                                end if;
                        end if;
                end process;
                oscilador: osc port map (contador, out_fsk);
end gen_fsk;
```

MODULADOR BPSK EN CÓDIGO VHDL.

```
library ieee;
library libreria;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
```

```
use libreria.osc.osc;

entity bpsk is
port(
        clk, reset : in std_logic;
        info : in std_logic;
        out_bpsk : out std_logic_vector(7 downto 0));
end bpsk;
architecture gen_bpsk of bpsk is
signal contador : natural;
begin
                p1: process(clk, reset, info)
                variable frec : integer:=1;
                begin
                        if (reset='1') then
                                contador <= 0;
                        elsif (clk'event and clk='1') then
                                if info = '1' then
                                        if (contador = 49) or (contador = 48) then
                                                contador <= 0;
                                        else
                                                contador <= contador + frec;
                                        end if;
                                else
                                        if (contador = 0)then
                                                contador <= 49;
                                        else
                                                contador <= contador - frec;
                                        end if;
                                end if;
                        end if;
                end process;
                oscilador: osc port map(contador, out_bpsk);
end gen_bpsk;
```

MODULADOR QPSK EN CÓDIGO VHDL.

Para la modulación QPSK se generó un oscilador diferente ya que este cuenta con cuatro fases distintas según sea su código de información. El código en VHDL de este oscilador se muestra en el anexo B.

```
library ieee;
library libreria;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
use libreria.osc.osc;

entity qpsk is
port(
        clk, reset : in std_logic;
        info : in std_logic_vector(1 downto 0);
        out_qpsk : out std_logic_vector(7 downto 0));
```

```
end qpsk;

architecture gen_qpsk of qpsk is
signal contador0, contador1, contador01, contador10: natural;
signal salida0, salida1, salida01, salida10 : std_logic_vector(7 downto 0);
begin
	p1: process(reset, clk)
	variable conta0, conta1, conta01, conta10   : natural;
	variable frec : integer:=1;
	begin
		if (reset='1') then
			conta0 := 0;
			conta1 := 0;
			conta01 := 0;
			conta10 := 0;
		elsif (clk'event and clk='1') then
			if info = "00" then
				conta1 := 0;
				conta01 := 0;
				conta10 := 0;
				if (conta0 = 49) or (conta0 = 48) then
					conta0 := 0;
				else
					conta0 := conta0 + frec;
				end if;
			elsif info = "11" then
				conta0 := 0;
				conta01 :=0;
				conta10 :=0;
				if (conta1 = 49) or (conta1 = 48)  then
					conta1 := 0;
				else
					conta1:= conta1 + frec;
				end if;
			elsif info = "01" then
				conta0:=0;
				conta1:=0;
				conta10:=0;
				if (conta01 = 49) or (conta01 = 48) then
					conta01 :=0;
				else
					conta01:= conta01 + frec;
				end if;
			else
				conta0:=0;
				conta1:=0;
				conta01:=0;
				if (conta10 = 49) or (conta10 = 48)   then
					conta10 := 0;
				else
					conta10 := conta10 + frec;
				end if;
			end if;
		end if;
		contador0 <= conta0;
```

```
                contador1 <= conta1;
                contador01 <= conta01;
                contador10 <= conta10;
        end process;

oscilador: osc port map(contador0, contador1,contador01, contador10, salida0,
salida1, salida01, salida10);
        with info select
        out_qpsk <=   salida0 when "00",
                      salida01 when "01",
                      salida10 when "10",
                      salida1 when others;
end gen_qpsk;
```

MODULADOR 8-QAM EN CÓDIGO VHDL

Como hay dos amplitudes de transmisión posibles en 8-AQM y cada una de ellas tiene cuatro fases distintas según la señal de información de entrada, se generó un oscilador diferente a los utilizados en las otras modulaciones. El código de este oscilador en código VHDL se muestra en el anexo C.

```
library ieee;
library libreria;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
use libreria.osc.osc;

entity ocho_qam is
port(
                clk, reset : in std_logic;
                info : in std_logic_vector(2 downto 0);
                out_8qam : out std_logic_vector(7 downto 0));
end ocho_qam;

architecture gen_ocho_qam of ocho_qam is
signal contador000, contador010, contador100, contador110: natural;
signal contador001, contador011, contador101, contador111: natural;
signal salida000, salida010, salida100, salida110 : std_logic_vector(7 downto
0);
signal salida001, salida011, salida101, salida111 : std_logic_vector(7 downto
0);
begin
        p1: process(reset, clk, info)
        variable frec: integer:=1;
        begin
                if (reset='1') then
                contador000 <= 0; contador010 <= 0; contador100 <= 0;
                contador110 <= 0; contador001 <= 0; contador011 <= 0;
                contador101 <= 0; contador111 <= 0;
                elsif (clk'event and clk='1') then
                        if info = "000" then
```

```
            contador010 <= 0; contador100 <= 0;
            contador110 <= 0; contador001 <= 0;
            contador011 <= 0; contador101 <= 0;
            contador111 <= 0;
            if (contador000 = 49) or (contador000 = 48) then
                  contador000 <= 0;
            else
                  contador000 <= contador000 + frec;
            end if;
      elsif info = "010" then
            contador000 <= 0; contador100 <= 0;
            contador110 <= 0; contador001 <= 0;
            contador011 <= 0; contador101 <= 0;
            contador111 <= 0;
            if (contador010 = 49) or (contador010 = 48) then
                  contador010 <= 0;
            else
                  contador010 <= contador010 + frec;
            end if;
      elsif info = "100" then
            contador000 <= 0; contador010 <= 0;
            contador110 <= 0; contador001 <= 0;
            contador011 <= 0; contador101 <= 0;
            contador111 <= 0;
            if (contador100 = 49) or (contador100 = 49) then
                  contador100 <=0;
            else
                  contador100 <= contador100 + frec;
            end if;
      elsif info = "110" then
            contador000 <= 0; contador010 <= 0;
            contador100 <= 0; contador001 <= 0;
            contador011 <= 0; contador101 <= 0;
            contador111 <= 0;
            if (contador110 = 49) or (contador110 = 48) then
                  contador110 <= 0;
            else
                  contador110 <= contador110 + frec;
            end if;
      elsif info = "001" then
            contador000 <= 0; contador010 <= 0;
            contador100 <= 0; contador110 <= 0;
            contador011 <= 0; contador101 <= 0;
            contador111 <= 0;
            if (contador001 = 49) or (contador001 = 48) then
                  contador001 <= 0;
            else
                  contador001 <= contador001 + frec;
            end if;
      elsif info = "011" then
            contador000 <= 0; contador010 <= 0;
            contador100 <= 0; contador110 <= 0;
            contador001 <= 0; contador101 <= 0;
            contador111 <= 0;
            if (contador011 = 49) or (contador011 = 48) then
```

```
                                        contador011 <= 0;
                                else
                                        contador011 <= contador011 + frec;
                                end if;
                        elsif info = "101" then
                                contador000 <= 0; contador010 <= 0;
                                contador100 <= 0; contador110 <= 0;
                                contador001 <= 0; contador011 <= 0;
                                contador111 <= 0;
                                if (contador101 = 49) or (contador101 = 48) then
                                        contador101 <= 0;
                                else
                                        contador101 <= contador101 + frec;
                                end if;
                        else
                                contador000 <= 0; contador010 <= 0;
                                contador100 <= 0; contador110 <= 0;
                                contador001 <= 0; contador011 <= 0;
                                contador101 <= 0;
                                if (contador111 = 49) or (contador111 = 48) then
                                        contador111 <= 0;
                                else
                                        contador111 <= contador111 + frec;
                                end if;
                        end if;
                end if;
        end process;

oscilador: osc port map(contador000, contador010, contador100, contador110,
contador001, contador011, contador101, contador111, salida000, salida010,
salida100, salida110, salida001, salida011, salida101, salida111);

        with info select
                out_8qam  <= salida000 when "000",
                             salida010 when "010",
                             salida100 when "100",
                             salida110 when "110",
                             salida001 when "001",
                             salida011 when "011",
                             salida101 when "101",
                             salida111 when others;
end gen_ocho_qam;
```

4.4.2 SIMULACIÓN DEL DISEÑO

Después del diseño inicial, el siguiente paso es la simulación del diseño. Para que el diseño tenga éxito es preciso tener adecuadas condiciones de entrada que puedan aplicarse al diseño que se simula y más tarde al producto final que se someterá a pruebas. Al aplicar estas condiciones de entrada el simulador intenta comprobar que el producto diseñado se desempeñara como se requiere según las especificaciones del producto original. Si la simulación revela algunos errores hay que cambiar el diseño a

fin de superarlos. La versión rediseñada se simula de nuevo para determinar si los errores desaparecieron. Este ciclo se repite hasta que la simulación indica un buen diseño.

Altera provee la herramienta de simulación ModelSim- Altera Starter Edition software para probar los diseños del programador antes de descargarlos en cualquiera de sus productos.

A continuación se presentan las gráficas de cada una de las modulaciones digitales antes de proceder a la descarga de este código en el módulo educacional de desarrollo DE2 de Altera. En la figura IV-22 se muestra la simulación de la modulación ASK en código VHDL.

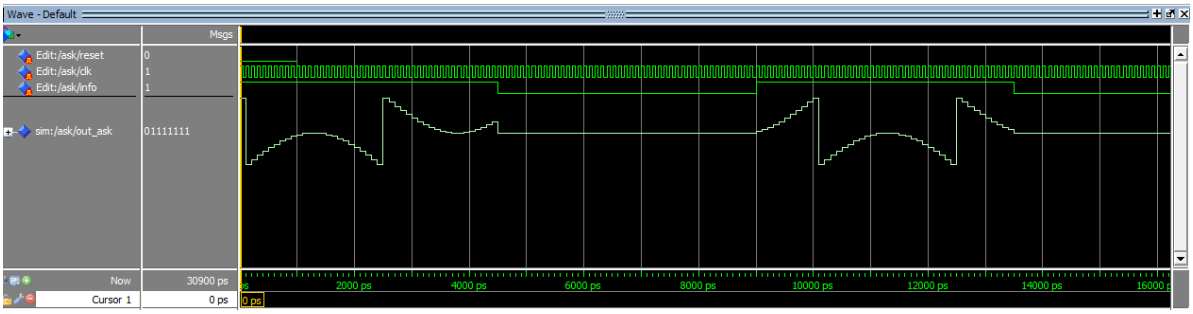


Figura IV-22: Modulación ASK

Como se puede observar la gráfica de la onda modulada presenta una notoria desviación, sin embargo no es ninguna falla de programación simplemente el software muestra de esa manera los datos.

Como se describió y calculó anteriormente la frecuencia de la señal portadora es de 1MHZ. En la figura IV-23 se presenta la simulación del modulador FSK en el software ModelSim de Altera.

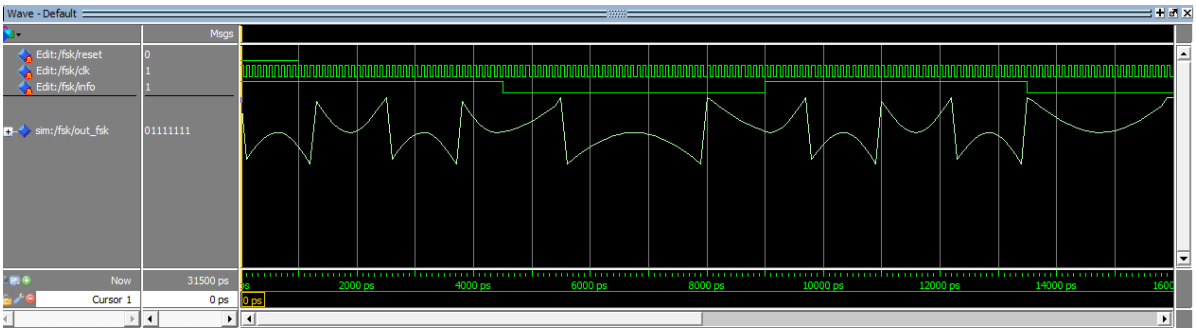


Figura IV-23: Modulación FSK

En la gráfica de la modulación FSK es evidente cuando existe un cambio de la señal de entrada la frecuencia de la señal modulada cambia.

Frecuencia en alto:

$$f_{señal} = \frac{f_{clk}}{\#\ de\ muestras}$$

$$f_{señal} = \frac{50MHz}{25}$$

$$f_{señal} = 2MHz$$

Frecuencia en bajo:

$$f_{señal} = \frac{f_{clk}}{\#\ de\ muestras}$$

$$f_{señal} = \frac{50MHz}{50}$$

$$f_{señal} = 1\text{MHz}$$

En la figura IV-24 se muestra la modulación BPSK, en la cual es claro ver el desfase de 180º cuando existe un cambio de la señal de entrada.

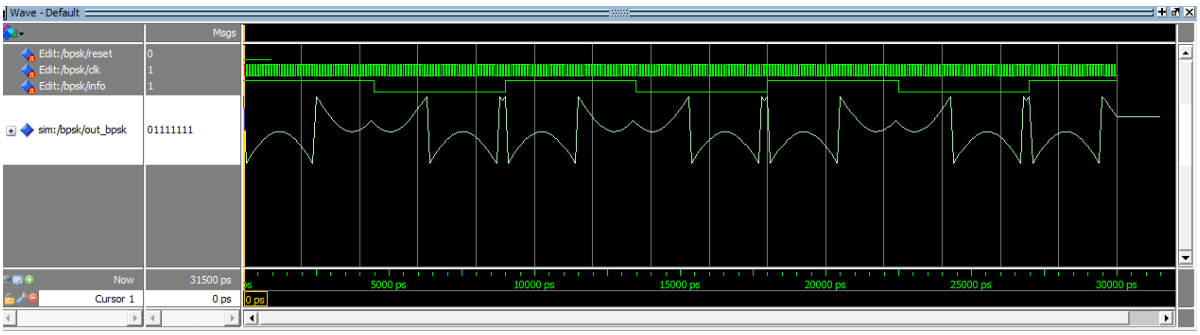


Figura IV-24: Modulación BPSK

Frecuencia de la señal portadora:

$$f_{señal} = \frac{f_{clk}}{\#\ de\ muestras}$$

$$f_{señal} = \frac{50MHz}{50}$$

$$f_{señal} = 1\text{MHz}$$

El modulador QPSK presenta 4 fases de salida dependiendo de la señal de información de entrada, en la figura IV-25 se muestra la modulación en código VHDL de este pido de modulación digital.

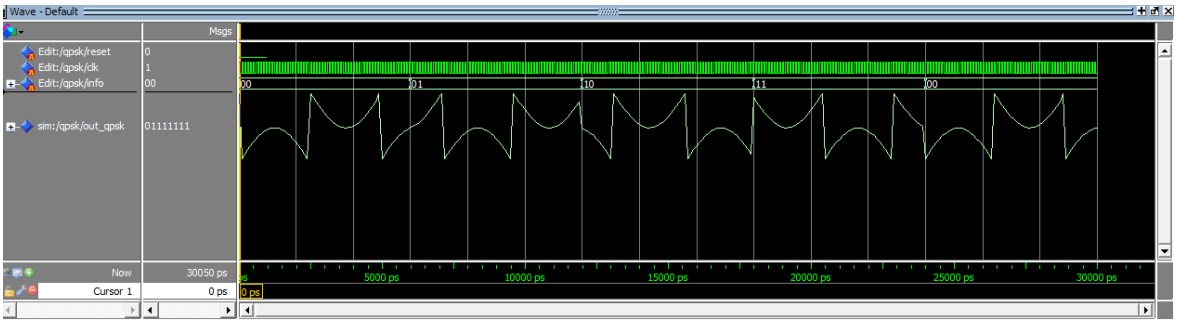


Figura IV-25: Modulación QPSK

Frecuencia de la señal portadora:

$$f_{señal} = \frac{f_{clk}}{\#\ de\ muestras}$$

$$f_{señal} = \frac{50MHz}{50}$$

$$f_{señal} = 1\text{MHz}$$

En la figura IV-26 se muestra la simulación del modulador 8-QAM en código VHDL.

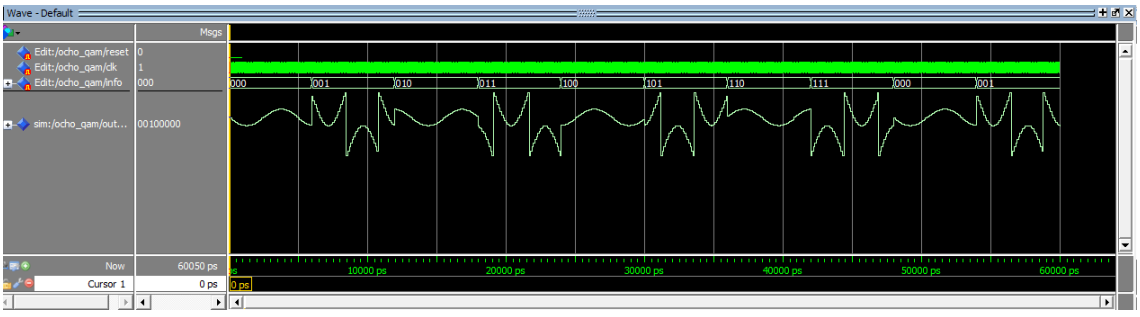


Figura IV-26: Modulación 8-QAM

Es visible notar tanto el cambio de fase como de amplitud en la señal modulada en el modulador 8-QAM.

Se tomó un total de 50 muestras para las 2 portadoras teniendo una frecuencia de 1Mhz, tanto en cuanto la amplitud si varia.

$$f_{señal} = \frac{f_{clk}}{\#\ de\ muestras}$$

$$f_{señal} = \frac{50MHz}{50}$$

$$f_{señal} = 1\text{MHz}$$

CAPÍTULO V

ANÁLISIS E INTERPRETACIÓN DE RESULTADOS.

5.1 INTRODUCCIÓN

Para comprobar la hipótesis de la presente tesis, se hará una comparación entre la tarjeta de desarrollo FPGA y los módulos de comunicaciones digitales KING Instrument Electronics CO., LTD, existentes en el laboratorio de telecomunicaciones de la Escuela de Ingeniería en Electrónica en Telecomunicaciones y Redes.

5.2 IMPLEMENTACIÓN FÍSICA DEL SISTEMA DIGITAL

5.2.1 PRUEBAS FÍSICAS EN LA TARJETA FPGA.

Una vez realizada las simulaciones de las modulaciones digitales se procede a la implementación en hardware, el cual se lo realizo en el FPGA Cyclone II EP2C35F672C6 de la compañía Altera, y la evaluación de resultados fue llevada a cabo mediante el uso de la tarjeta de desarrollo DE2 de la misma compañía, con el objetivo de demostrar y observar físicamente el diseño de un modulador digital, para la visualización de las modulaciones digitales se utilizó una sección del módulo en donde se tiene un circuito integrado AD7123 el cual consta de 3 DAC de 10 bits cada uno, el

mismo que es utilizado en las aplicaciones de video a través del puerto VGA, el cual se utilizó para visualizar en un osciloscopio las modulaciones, en la figura V-1 se muestra la transferencia del archivo de configuración .sof que se crea después de haber compilado y asignado los pines de entrada y salida al chip FPGA.

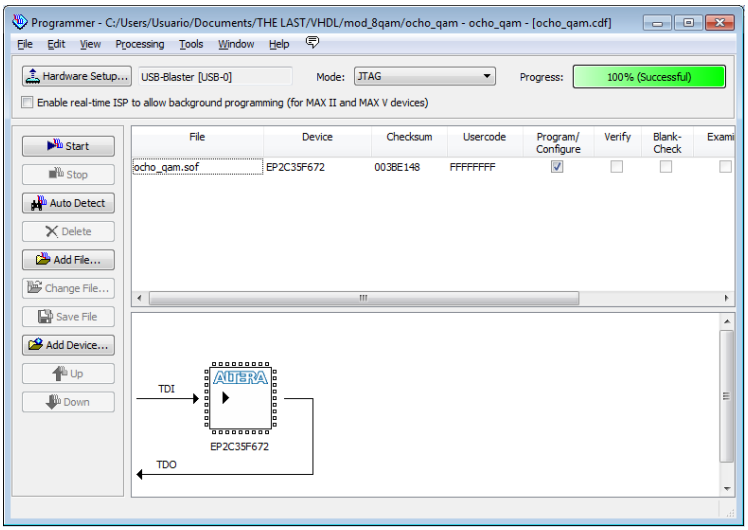


Figura V-1: Transferencia del Archivo de Configuración a la FPGA

El uso de recursos del FPFA Cyclone II se muestra en la gráfica V-2 y es muy visible observar que el total de elementos lógicos ocupados para la programación no ocupa más de un 5% del total de elementos lógicos del chip, además el uso de la memoria dentro del dispositivo fue del 0%.

Flow Summary

Flow Status	Successful - Tue Feb 11 14:41:53 2014
Quartus II 32-bit Version	11.1 Build 173 11/01/2011 SJ Web Edition
Revision Name	ocho_qam
Top-level Entity Name	ocho_qam
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	1,120 / 33,216 (3 %)
Total combinational functions	1,120 / 33,216 (3 %)
Dedicated logic registers	248 / 33,216 (< 1 %)
Total registers	248
Total pins	16 / 475 (3 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Figura V-2: Reporte de recursos utilizados

Este resultado demuestra que el diseño en código VHDL ha sido eficiente y enfocado básicamente en disminuir el consumo de potencia, ya que menor sea la cantidad de recursos utilizados, menor será la potencia consumida por el dispositivo.

Una vez descargado el archivo de configuración en la tarjeta de desarrollo como se muestra en la figura V-3 se procede a analizar dichas modulaciones y ver si es el resultado fue el esperado en el osciloscopio, dichas graficas se pueden observar acontinuacion:

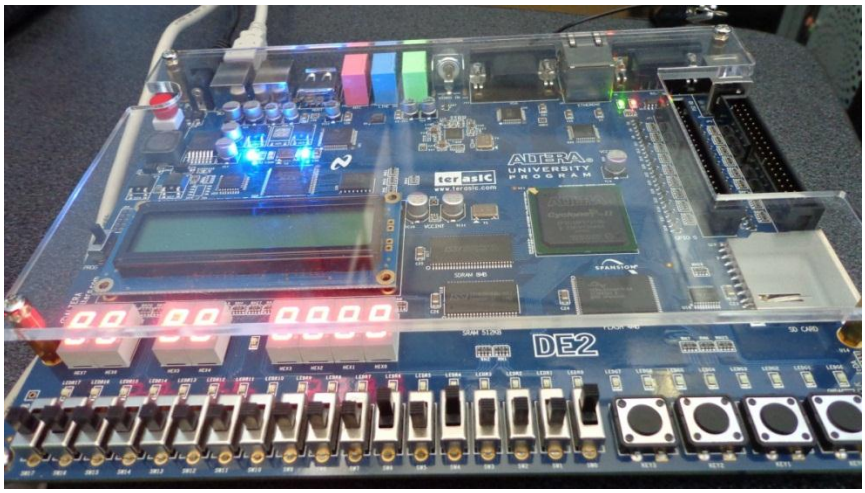


Figura V-3: Entrenador DE2 programado

Un detalle muy importante es que la frecuencia de oscilación de reloj interna de la tarjeta FPGA es de 50 MHz, por tal motivo es imposible ver un cambio de fase de las ondas a diferencia de las simulaciones mostradas en el software ModelSim, en donde se puede tener un control del tiempo. Sin embargo se ha utilizado una señal de clock externa para poder apreciar las gráficas en el osciloscopio como se muestra a continuación:

MODULACIÓN ASK

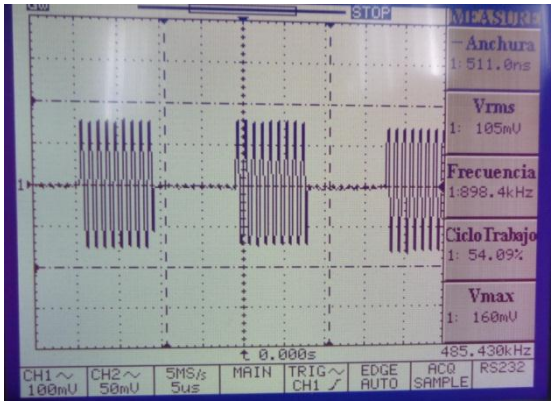


Figura V-4: Modulación ASK

Parámetros utilizados en la modulación ASK:

Frecuencia portadora: 1MHz (señal sinusoidal).

Frecuencia de información: 55KHz (señal digital).

MODULACIÓN FSK

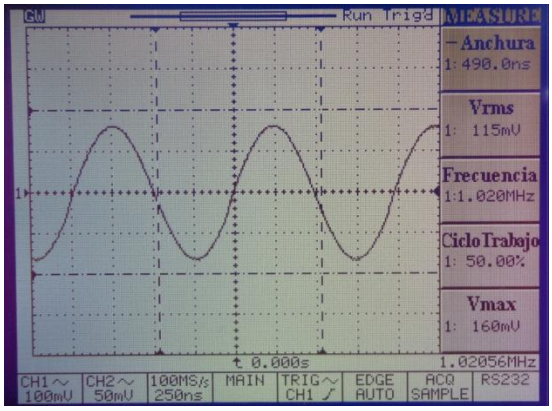


Figura V-5: Modulación FSK en bajo

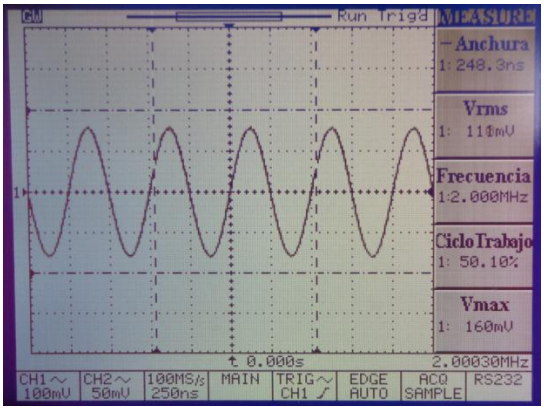


Figura V-6: Modulación FSK en alto

Parámetros del modulador FSK:

Frecuencia de portadora uno para nivel bajo: 1MHz (señal sinusoidal).

Frecuencia de portadora dos para nivel alto: 2MHz (señal sinusoidal).

MODULACIÓN BPSK

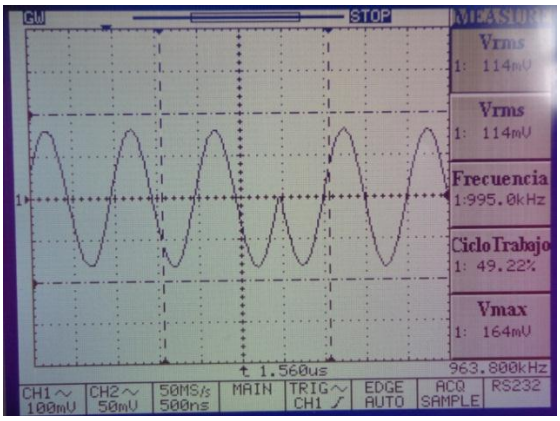


Figura V-7: Modulación BPSK

Parámetro del modulador BPSK:

Frecuencia portadora: 1MHz (señal sinusoidal).

Frecuencia de información: 110KHz (señal digital).

MODULACIÓN QPSK

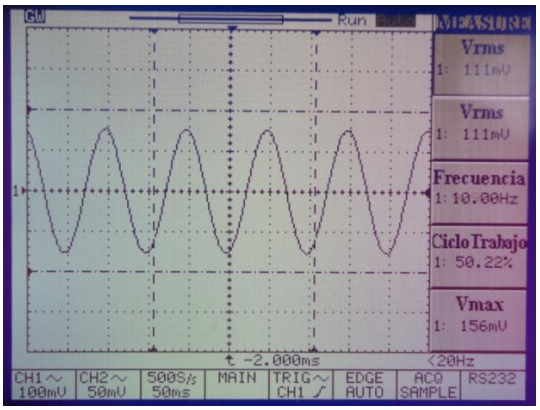


Figura V-8: Modulador QPSK con desfase de -45º

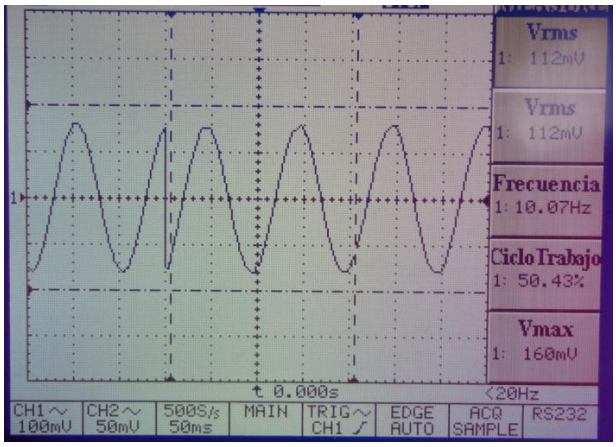


Figura V-9: Modulador QPSK con desfase de -135º

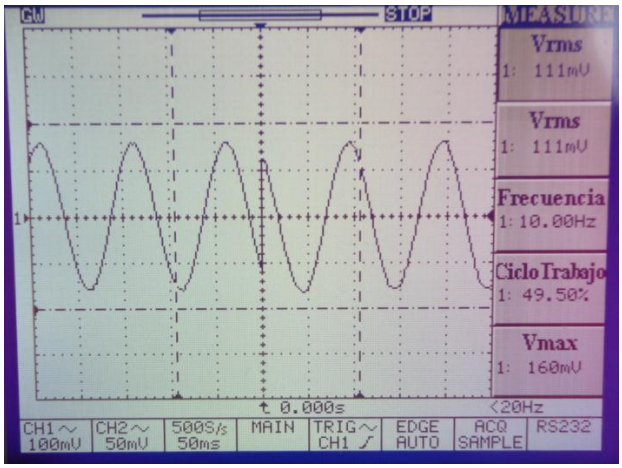


Figura V-10: Modulador QPSK con desfase de +45º

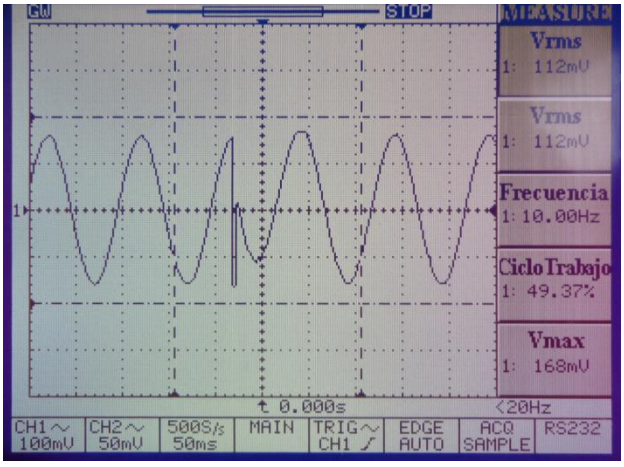


Figura V-11: Modulador QPSK con desfase de +135º.

Parámetros del modulador QPSK:

Frecuencia de oscilación de reloj: 500Hz.

MODULACIÓN 8-QAM

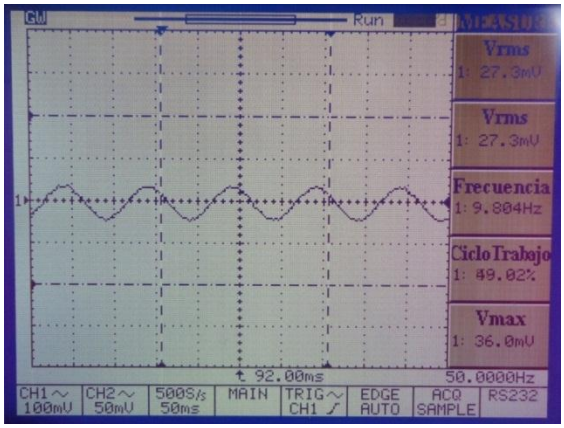


Figura V-12: Modulador 8-QAM con amplitud A y desfase de -135º

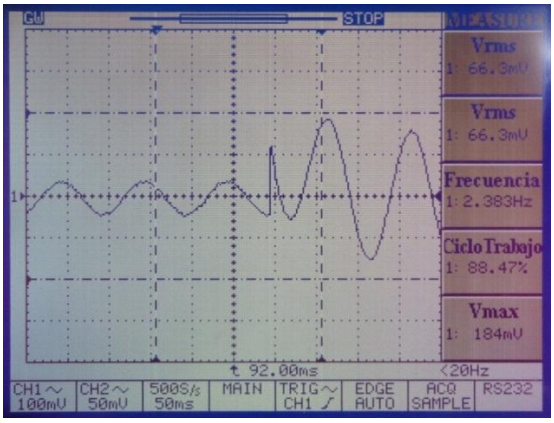


Figura V-13: Modulador 8-QAM con amplitud B y desfase de -135º.

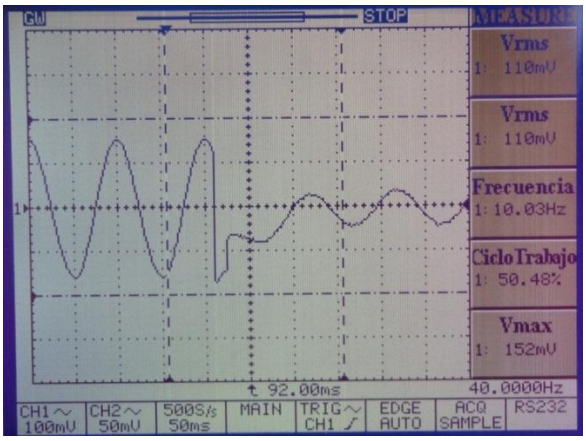


Figura V-14: Modulador 8-QAM con amplitud A y desfase de -45º

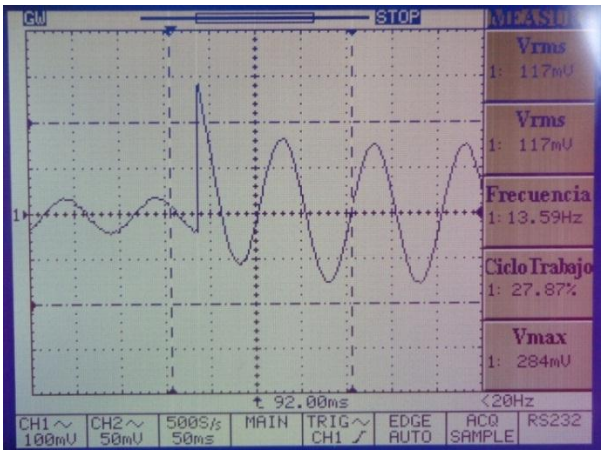


Figura V-15: Modulador 8-QAM con amplitud B y desfase de -45º

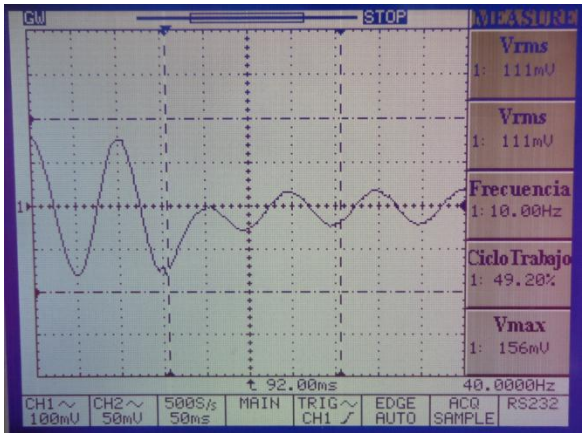


Figura V-16: Modulador 8-QAM con amplitud Ay desfase de +135º

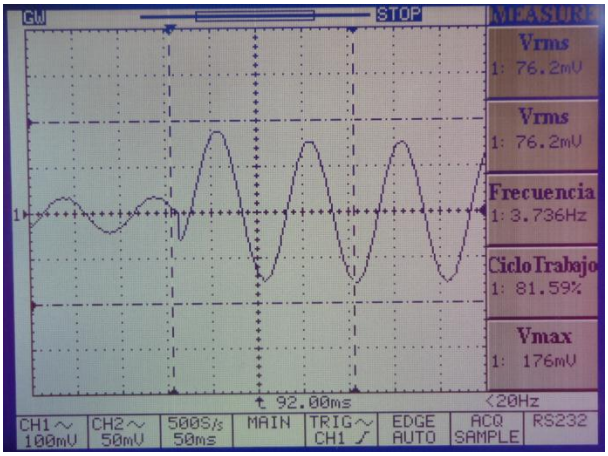


Figura V-17: Modulador 8-QAM con amplitud B y desfase de +135º

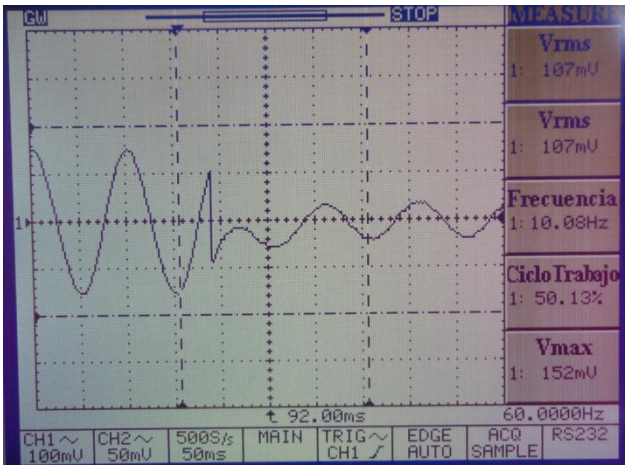


Figura V-18: Modulador 8-QAM con amplitud A y desfase de +45º

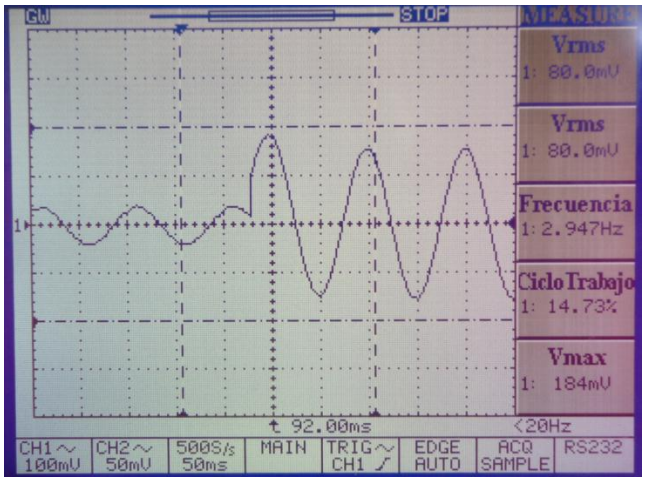


Figura V-19: Modulador 8-QAM con amplitud B y desfase de +45º

.

Parámetros de la modulación 8-QAM:
Frecuencia de oscilación de reloj 500Hz.

5.2.2 PRUEBAS FÍSICAS EN MÓDULOS DE COMUNICACIONES DIGITALES DEL LABORATORIO DE COMUNICACIONES DE LA EIE

El laboratorio de comunicaciones de la EIE-TR cuenta con laboratorios KING Instrument Electronics de la K & H MFG. CO., LTD los cuales son módulos de pruebas para las modulaciones ASK, FSK y PSK/QPSK, los cuales tienen sus propias características, diseños y parámetros los cuales se detallan en anexo D, además se probaran estos módulos para la comparación de las dos tecnologías:

MODULO KL 94005 - MODULADOR ASK

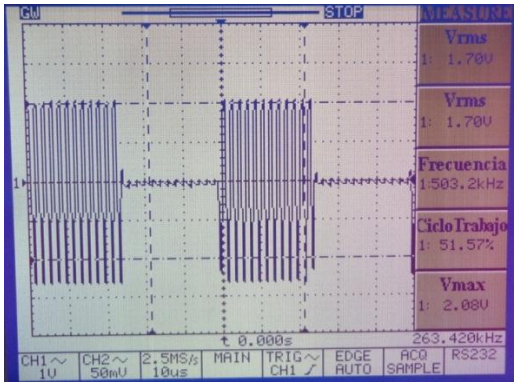


Figura V-20: Modulación ASK

Parámetro del modulador:

Frecuencia de portadora: 500KHz.

Frecuencia de información: 20KHz.

MODULO KL 94003 - MODULADOR FSK

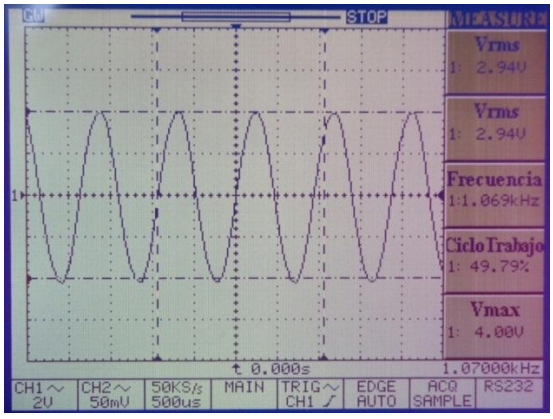


Figura V-21: Modulación FSK con señal en alto

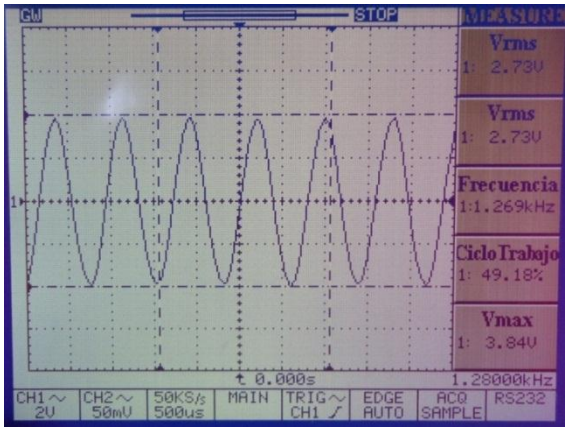


Figura V-22: Modulación FSK con señal en bajo

Parámetros del modulador FSK:

Frecuencia de señal en alto: 1070Hz.

Frecuencia de señal en bajo: 1270Hz.

MODULO KL 94006 - MODULADOR PSK/QPSK

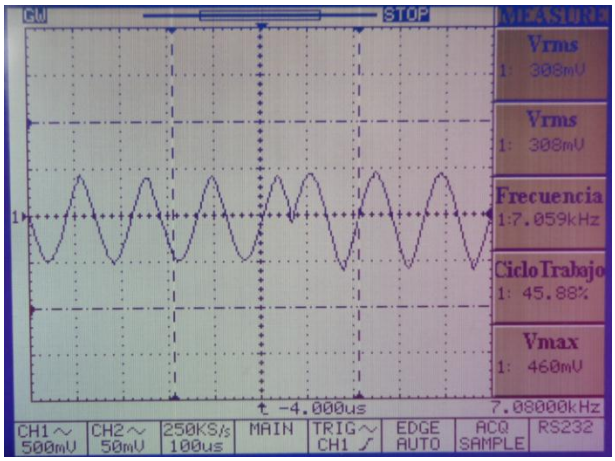


Figura V-23: Modulación PSK/QPSK

Parámetros del modulador PSK/QPSK:

Frecuencia de portadora: 1Hz.

5.3 ANÁLISIS DE RESULTADOS

Una vez realizadas las modulaciones con la tecnología FPGA y con los laboratorios de KING Instrument Electronics se procede hacer un análisis realizando una escala de valorización con los niveles alto, medio, bajo y nula.

Valor cualitativo	Valor cuantitativo
Alto	3
Medio	2
Bajo	1
Nula	0

Tab. V-I Valorización para el indicador

La valorizaran se realiza de acuerdo al indicador: Calidad de la señal mostrada en el osciloscopio de las modulaciones digitales.

Indicador	Valor cualitativo
Buena	Alto
Media	Medio
Mala	bajo
No se muestra.	Nula

Tab. V-II Valorización del indicador

VALORACIONES

La valoración se la realiza a cada una de las modulaciones digitales:

Señales mostradas en el osciloscopio		
Modulaciones	**King instrument electronics**	**De2**
ASK	Buena	Buena
FSK	Buena	Buena
BPSK	Buena	Buena
QPSK	Mala	Buena
8-QAM	No se muestra	Buena

Tab. V-III Resultados del indicador

CALIFICACIÓN

Basándose en la tabla 5.2, y en la valorización para el indicador de la tabla 5.1 se tiene:

Señales mostradas en el osciloscopio		
Modulaciones	**King instrument electronics**	**De2**
ASK	3	3
FSK	3	3
BPSK	3	3
QPSK	1	3
8-QAM	0	3

Tab. V-IV Calificación del indicador

Para obtener los porcentajes se utilizan las siguientes formulas:

X= King instrument electronics

Y = De2.

Para X:

$$Pa(X) = \sum C1(X) + C2(X) + C3(X) + C4\,(X) + C5(X)$$

Dónde:

Pa(X): Valor acumulativo de King instrument electronics.

C1(X): Valor de la modulación ASK de King instrument electronics.

C2(X): Valor de la modulación FSK de King instrument electronics.

C3(X): Valor de la modulación BPSK de King instrument electronics.

C4(X): Valor de la modulación QPSK de King instrument electronics.

C5(X): Valor de la modulación 8-QAM de King instrument electronics.

Por tanto:

$$Pa(X) = 3 + 3 + 3 + 1 + 0$$

$$Pa(X) = 10$$

Para Y:

$$Pa(Y) = \sum C1(Y) + C2(Y) + C3(Y) + C4\,(Y) + C5(Y)$$

Dónde:

Pa(Y): Valor acumulativo de De2.

C1(Y): Valor de la modulación ASK de De2.

C2(Y): Valor de la modulación FSK de De2.

C3(Y): Valor de la modulación BPSK de De2.

C4(Y): Valor de la modulación QPSK de De2.

C5(Y): Valor de la modulación 8-QAM de De2.

Por tanto:

$$Pa(Y) = 3 + 3 + 3 + 3 + 3$$

$$Pa(Y) = 15$$

Determinamos el porcentaje parcia total con las siguientes ecuaciones:

$$\mathrm{PpT(X)} = \left(\frac{\mathrm{Cn(X)}}{\mathrm{Vm}}\right) * 100\% \quad , \quad \mathrm{PpT(Y)} = \left(\frac{\mathrm{Cn(Y)}}{\mathrm{Vm}}\right) * 100\%$$

Dónde:

PpT(X): Porcentaje parcial total de King instrument electronics.

PpT(Y): Porcentaje parcial total de De2.

Vm: Valor máximo de indicadores = 15.

De lo que se tiene la siguiente tabla:

	Calidad de la visualización			
	King instrument electronics		**De2**	
	Valor (X)	**% PpT**	**Valor (Y)**	**% PpT**
C1: ASK	3	20	3	20
C2: FSK	3	20	3	20
C3: BPSK	3	20	3	20
C4: QPSK	1	6.66	3	20
C5: 8-QAM	0	0	3	20

Tab. V-V Valores y porcentajes de cada modulación

Para obtener el porcentaje total de cada uno de los entrenadores se utilizan las siguientes formulas:

$$PT(X) = \left(\frac{Pa(X)}{Vm}\right) * 100\% \quad , \quad PT(Y) = \left(\frac{Pa(Y)}{Vm}\right) * 100\%$$

Dónde:

PT(X): Porcentaje total de King instrument electronics.

PT(Y): Porcentaje total de De2.

Por consiguiente:

$$PT(X) = \left(\frac{10}{15}\right) * 100\%$$

$$PT(X) = 66.66\%$$

$$PT(Y) = \left(\frac{15}{15}\right) * 100\%$$

$$PT(Y) = 100\%$$

Por tanto se tiene la tabla final:

	Calidad de visualización en el osciloscopio	
	Valor (Pa)	**% (PT)**
King instrument electronics (X)	**10**	**66.66**
De2 (Y)	**15**	**100**

Tab. V-VI Valores y porcentajes finales del indicador

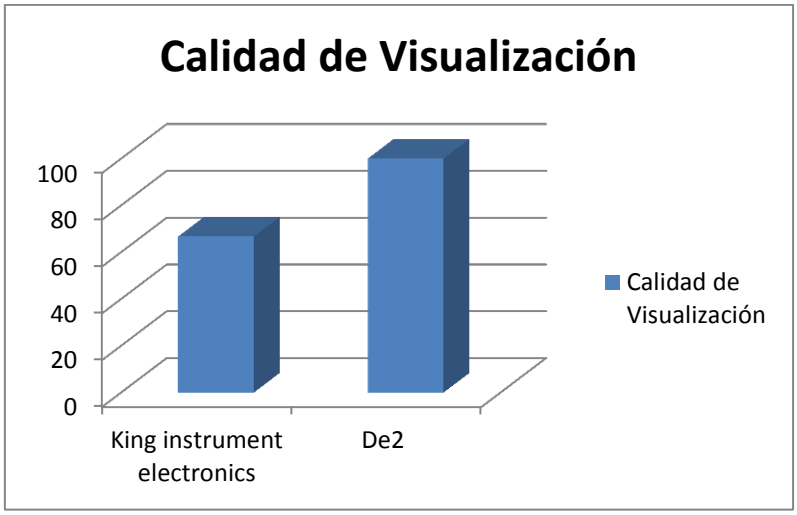


Figura V-24: Resultado final del indicador

5.4 COMPROBACIÓN DE LA HIPÓTESIS.

De los resultados expuestos en la tabla 5.6 se puede concluir que la tarjeta de desarrollo educacional DE2 de Altera es un elemento potencial para el análisis de la modulaciones digitales y se acepta la hipótesis planteada.

CONCLUSIONES

- Los dispositivos programables en su gran mayoría han demostrado un gran rendimiento y facilidad al momento de implementar grandes sistemas digitales, tanto los dispositivos FPGA y sus alternativas de desarrollo hace que el programador tenga una amplia versatilidad al momento de desarrollar cualquier sistema digital, además con sus herramientas de simulación provistas por sus fabricantes hacen un poderoso entrenador para el ámbito académico.

- Altera Corporation posee soluciones muy completas y de buen desempeño en soluciones de lógica programable y además cuenta con un programa universitario completo para introducir a los estudiantes a la tecnología digital, brindando así: hardware, software y material de aprendizaje; en hardware nos brinda lo que son los tableros educacionales de desarrollo diseñados especialmente para los laboratorios de investigación los cuales están a un precio por debajo de los costes de fabricación; tanto en cuanto al soporte de software consiste en la herramienta de diseño Quartus II y en herramienta de simulación el ModelSim-Altera los cuales se pueden descargar de manera gratuita de la página web de la compañía al igual que el material para el aprendizaje.

- Mediante el lenguaje de descripción de hardware VHDL, la herramienta de diseño Quartus II y el software de simulación ModelSim se diseñaron y analizaron las modulaciones digitales las cuales fueron examinadas y comparadas con otros sistemas de modulación adquiridos del laboratorio de comunicaciones de la EIE-TR, en donde se concluyó que la tecnología FGPA es una latente alternativa para el entrenamiento y para el acondicionamiento de los laboratorios de la EIE-TR.

- Los resultados obtenidos de la implementación de las modulaciones digitales en la tarjeta de desarrollo DE2 de Altera nos indica la factibilidad técnica y operativa para la realización de las practicas demostrativas referentes a sistemas y modulaciones digitales, igualmente las herramientas de soporte para el desarrollo del tema y el procedimiento de la programación en VHDL, son muy prácticos y confiables.

- La tecnología FPGA es un elemento potencial para el análisis de las modulaciones digitales obteniendo un 100% de efectividad al momento de analizarlas.

RECOMENDACIONES

- Es aconsejable tener un conocimiento en lo que se refiere a programación en alto nivel, ya que el lenguaje de programación en VHDL es sintácticamente similar a estos lenguajes de programación y así tener una facilidad al momento de tratar con el lenguaje de descripción de hardware VHDL.

- Se recomienda que se conozca el funcionamiento de los componentes del módulo entrenador a manipular, y también sus hojas de datos ya que se pueden presentar problemas en lo que se refiere a niveles de voltaje y corriente.
- Para una mejor visualización en el osciloscopio de la señal de salida es recomendable utilizar una señal externa de reloj a frecuencias bajas en el orden de los Hz, para que el ojo humano pueda apreciar y diferenciar las señales moduladas.

- Se recomienda la adquisición de entrenadores con tecnología FPGA como son los proporcionados por la compañía ALTERA para equipar los laboratorios de digitales y de comunicaciones de la EIE de la ESPOCH, ya que son unos equipos muy robustos, versátiles y acorde a la actualidad, los mismos que ayudaran a abaratar costos, tiempo y espacio al momento de realizar cualquier aplicación relacionada a sistemas digitales y de telecomunicaciones.

- Se recomienda tener mucho cuidado con la manipulación de los elementos del módulo de desarrollo educacional DE2 ya que son dispositivos delicados y un exceso de voltaje o por simple estática podría causar el fallo o daño en dichos elementos.

RESUMEN

El análisis e implementación de sistemas de modulación digital ASK, FSK, M-PSK y M-QAM mediante la programación en código VHDL utilizando la tecnología Field Programmable Gate Array (FPGA) servirá para verificar la potencialidad de esta tecnología y equipar el laboratorio de Comunicaciones de la Escuela de Ingeniería en Electrónica en Telecomunicaciones y Redes de la Escuela Superior Politécnica de Chimborazo.

Mediante el método deductivo se recopila y selecciona la información necesaria para investigar y poder conocer el funcionamiento, programación y características de la tecnología FPGA el cual nos servirá para el diseño e implementación de las modulaciones digitales en tarjetas programables con dicha tecnología.

Con los resultados obtenidos se demuestra que la tecnología FPGA obtiene un 100% de efectividad al momento de analizar las señales con respecto a los equipos de pruebas del laboratorio de comunicaciones de la Facultad de Informática y Electrónica que solamente tuvieron un 66% de calidad de visualización en el osciloscopio.

Con la implementación concluimos la eficacia de las modulaciones digitales en la tarjeta de desarrollo DE2 de ALTERA Corporation, demostrando prácticamente la teoría, igualmente las herramientas de soporte de desarrollo del tema y el procedimiento de la programación en VHDL, son muy prácticos y confiables.

Se recomienda la adquisición de entrenadores con tecnología FPGA para equipar los laboratorios de digitales y comunicaciones de la EIE-TR de la ESPOCH, siendo equipos robustos, versátiles y confiables, los mismos que ayudarán, abaratar costos, tiempo y espacio al momento de realizar cualquier aplicación relacionada a la electrónica digital.

ABSTRACT

Analysis and implementation of digital modulation systems ASK (Amplitude Shift Keying) FSK (Frequency Shift Keying) M-PSK (Phase Shift Keying) M-QAM (Quadrature Amplitude Modulation) programmatically in code VHDL using Field Programmable Gate Array (FPGA) technology will be used to verify the potential of this technology and to equip the laboratory Communications of Escuela de Ingeniería Electrónica en Telecomunicaciones y Redes de la Escuela Superior Politécnica de Chimborazo.

Using the deductive method compiles and selects the information needed to investigate and get to know the operation, programming and features of the FPGA technology which will serve for the design and implementation of digital modulations in programmable cards with this technology.

The results demonstrate that the FPGA technology obtains 100% effectiveness when analyzing signals with respect to laboratory test equipment Communication, of Facultad de Informática y Electrónica which had only 66% of the viewing oscilloscope.

It is concluded with the implementation, the efficacy of digital modulations in DE2 development card of ALTERA Corporation, showing practically theory, also support tools of theme development and the procedure of VHDL (Very high speed integrated circuit hardware description language) programming, are very practical and reliable.

The acquisition of trainers whit FPGA technology is recommended to equip the laboratories of digital and communications of the EIE-TR of ESPOCH, being robust, reliable, and versatile equipment that help reducing costs, time and space at the moment application related to digital electronics.

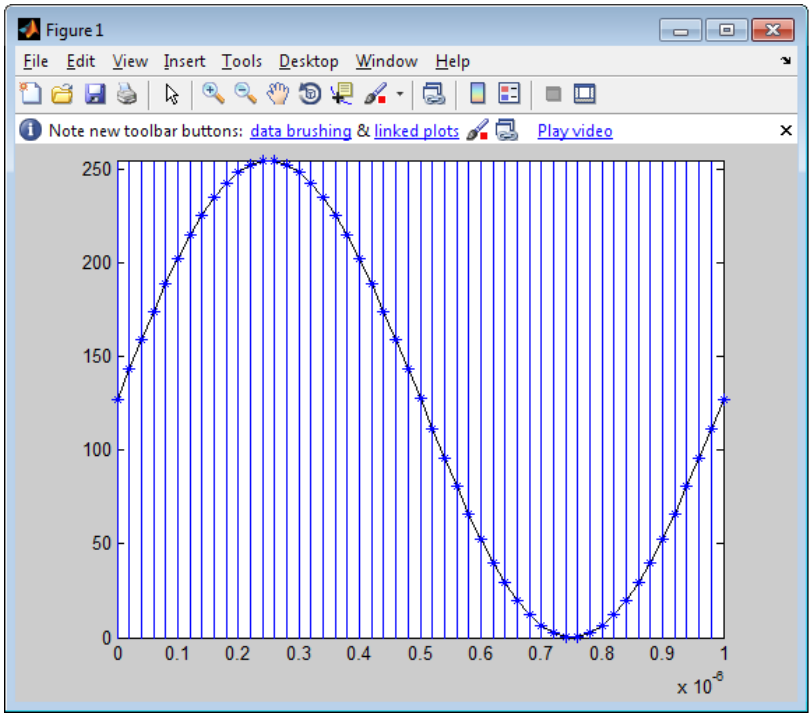
ANEXOS

ANEXO A

CÓDIGO Y SIMULACIÓN EN MATLAB PARA OBTENER UN NÚMERO DE 1 BYTE CON 50 MUESTRAS.

```
clear
frec = 10^6;
t = 0:(1/frec)/50:(1/frec);
y = (255/2)*(sin(2*pi*frec*t)+1);
plot(t,y,'-k*','LineWidth',1,'MarkerEdgeColor','b')
hold on
axis([0 (1/frec) 0 255])

for n = 0:49
    xl = [(n*((1/frec)/50)) (n*((1/frec)/50))];
    yl = [0 255];
    line(xl,yl);
    (255/2)*(sin(2*pi*frec*(n*((1/frec)/50)))+1)
end
```



ANEXO B

CÓDIGO VHDL DEL OSCILADOR PARA LA MODULACIÓN QPSK.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity osc is
	port (
		contador0 : in natural;
		contador1 : in natural;
		contador01 : in natural;
		contador10 : in natural;
		onda0, onda1, onda01, onda10 : out std_logic_vector (7 downto 0));
end osc;

architecture gen_osc of osc is
signal seno0, seno1, seno10, seno01 : std_logic_vector(7 downto 0);
begin

	p00: process(contador0)
		begin
		case (contador0) is
			when 0  => seno0 <= "01111111";
			when 1  => seno0 <= "10001111";
			when 2  => seno0 <= "10011111";
			when 3  => seno0 <= "10101110";
			when 4  => seno0 <= "10111101";
			when 5  => seno0 <= "11001010";
			when 6  => seno0 <= "11010111";
			when 7  => seno0 <= "11100010";
			when 8  => seno0 <= "11101011";
			when 9  => seno0 <= "11110011";
			when 10 => seno0 <= "11111001";
			when 11 => seno0 <= "11111101";
			when 12 => seno0 <= "11111111";
			when 13 => seno0 <= "11111111";
			when 14 => seno0 <= "11111101";
			when 15 => seno0 <= "11111001";
			when 16 => seno0 <= "11110011";
			when 17 => seno0 <= "11101011";
			when 18 => seno0 <= "11100010";
```

```
                when 19 => seno0 <= "11010111";
                when 20 => seno0 <= "11001010";
                when 21 => seno0 <= "10111101";
                when 22 => seno0 <= "10101110";
                when 23 => seno0 <= "10011111";
                when 24 => seno0 <= "10001111";
                when 25 => seno0 <= "01111111";
                when 26 => seno0 <= "01101111";
                when 27 => seno0 <= "01100000";
                when 28 => seno0 <= "01010000";
                when 29 => seno0 <= "01000010";
                when 30 => seno0 <= "00110100";
                when 31 => seno0 <= "00101000";
                when 32 => seno0 <= "00011101";
                when 33 => seno0 <= "00010100";
                when 34 => seno0 <= "00001100";
                when 35 => seno0 <= "00000110";
                when 36 => seno0 <= "00000010";
                when 37 => seno0 <= "00000000";
                when 38 => seno0 <= "00000000";
                when 39 => seno0 <= "00000010";
                when 40 => seno0 <= "00000110";
                when 41 => seno0 <= "00001100";
                when 42 => seno0 <= "00010100";
                when 43 => seno0 <= "00011101";
                when 44 => seno0 <= "00101000";
                when 45 => seno0 <= "00110100";
                when 46 => seno0 <= "01000010";
                when 47 => seno0 <= "01010000";
                when 48 => seno0 <= "01100000";
                when 49 => seno0 <= "01101111";
                when others => seno0 <=  "00000000";
        end case;
        onda0 <= seno0;
end process;

p10: process(contador10)
        begin
        case (contador10) is
                when 0 => seno10 <= "11111111";
                when 1 => seno10 <= "11111101";
                when 2 => seno10 <= "11111001";
                when 3 => seno10 <= "11110011";
                when 4 => seno10 <= "11101011";
                when 5 => seno10 <= "11100010";
```

```
when 6 => seno10 <= "11010111";
when 7 => seno10 <= "11001010";
when 8 => seno10 <= "10111101";
when 9 => seno10 <= "10101110";
when 10 => seno10 <= "10011111";
when 11 => seno10 <= "10001111";
when 12 => seno10 <= "01111111";
when 13 => seno10 <= "01101111";
when 14 => seno10 <= "01100000";
when 15 => seno10 <= "01010000";
when 16 => seno10 <= "01000010";
when 17 => seno10 <= "00110100";
when 18 => seno10 <= "00101000";
when 19 => seno10 <= "00011101";
when 20 => seno10 <= "00010100";
when 21 => seno10 <= "00001100";
when 22 => seno10 <= "00000110";
when 23 => seno10 <= "00000010";
when 24 => seno10 <= "00000000";
when 25 => seno10 <= "00000000";
when 26 => seno10 <= "00000010";
when 27 => seno10 <= "00000110";
when 28 => seno10 <= "00001100";
when 29 => seno10 <= "00010100";
when 30 => seno10 <= "00011101";
when 31 => seno10 <= "00101000";
when 32 => seno10 <= "00110100";
when 33 => seno10 <= "01000010";
when 34 => seno10 <= "01010000";
when 35 => seno10 <= "01100000";
when 36 => seno10 <= "01101111";
when 37 => seno10 <= "01111111";
when 38 => seno10 <= "10001111";
when 39 => seno10 <= "10011111";
when 40 => seno10 <= "10101110";
when 41 => seno10 <= "10111101";
when 42 => seno10 <= "11001010";
when 43 => seno10 <= "11010111";
when 44 => seno10 <= "11100010";
when 45 => seno10 <= "11101011";
when 46 => seno10 <= "11110011";
when 47 => seno10 <= "11111001";
when 48 => seno10 <= "11111101";
when 49 => seno10 <= "11111111";
when others => seno10 <=  "00000000";
```

```
        end case;
        onda10 <= seno10;
end process;

        p01: process(contador01)
        begin
        case (contador01) is
                when  0 => seno01 <= "00000000";
                when  1 => seno01 <= "00000010";
                when  2 => seno01 <= "00000110";
                when  3 => seno01 <= "00001100";
                when  4 => seno01 <= "00010100";
                when  5 => seno01 <= "00011101";
                when  6 => seno01 <= "00101000";
                when  7 => seno01 <= "00110100";
                when  8 => seno01 <= "01000010";
                when  9 => seno01 <= "01010000";
                when 10 => seno01 <= "01100000";
                when 11 => seno01 <= "01101111";
                when 12 => seno01 <= "01111111";
                when 13 => seno01 <= "10001111";
                when 14 => seno01 <= "10011111";
                when 15 => seno01 <= "10101110";
                when 16 => seno01 <= "10111101";
                when 17 => seno01 <= "11001010";
                when 18 => seno01 <= "11010111";
                when 19 => seno01 <= "11100010";
                when 20 => seno01 <= "11101011";
                when 21 => seno01 <= "11110011";
                when 22 => seno01 <= "11111001";
                when 23 => seno01 <= "11111101";
                when 24 => seno01 <= "11111111";
                when 25 => seno01 <= "11111111";
                when 26 => seno01 <= "11111101";
                when 27 => seno01 <= "11111001";
                when 28 => seno01 <= "11110011";
                when 29 => seno01 <= "11101011";
                when 30 => seno01 <= "11100010";
                when 31 => seno01 <= "11010111";
                when 32 => seno01 <= "11001010";
                when 33 => seno01 <= "10111101";
                when 34 => seno01 <= "10101110";
                when 35 => seno01 <= "10011111";
                when 36 => seno01 <= "10001111";
                when 37 => seno01 <= "01111111";
```

```
                when 38 => seno01 <= "01101111";
                when 39 => seno01 <= "01100000";
                when 40 => seno01 <= "01010000";
                when 41 => seno01 <= "01000010";
                when 42 => seno01 <= "00110100";
                when 43 => seno01 <= "00101000";
                when 44 => seno01 <= "00011101";
                when 45 => seno01 <= "00010100";
                when 46 => seno01 <= "00001100";
                when 47 => seno01 <= "00000110";
                when 48 => seno01 <= "00000010";
                when 49 => seno01 <= "00000000";
                when others => seno01 <=  "00000000";
        end case;
        onda01 <= seno01;
end process;

p11: process(contador1)
        begin
        case (contador1) is
                when 0 => seno1 <= "01111111";
                when 1 => seno1 <= "01101111";
                when 2 => seno1 <= "01100000";
                when 3 => seno1 <= "01010000";
                when 4 => seno1 <= "01000010";
                when 5 => seno1 <= "00110100";
                when 6 => seno1 <= "00101000";
                when 7 => seno1 <= "00011101";
                when 8 => seno1 <= "00010100";
                when 9 => seno1 <= "00001100";
                when 10 => seno1 <= "00000110";
                when 11 => seno1 <= "00000010";
                when 12 => seno1 <= "00000000";
                when 13 => seno1 <= "00000000";
                when 14 => seno1 <= "00000010";
                when 15 => seno1 <= "00000110";
                when 16 => seno1 <= "00001100";
                when 17 => seno1 <= "00010100";
                when 18 => seno1 <= "00011101";
                when 19 => seno1 <= "00101000";
                when 20 => seno1 <= "00110100";
                when 21 => seno1 <= "01000010";
                when 22 => seno1 <= "01010000";
                when 23 => seno1 <= "01100000";
                when 24 => seno1 <= "01101111";
```

```
                when 25 => seno1 <= "01111111";
                when 26 => seno1 <= "10001111";
                when 27 => seno1 <= "10011111";
                when 28 => seno1 <= "10101110";
                when 29 => seno1 <= "10111101";
                when 30 => seno1 <= "11001010";
                when 31 => seno1 <= "11010111";
                when 32 => seno1 <= "11100010";
                when 33 => seno1 <= "11101011";
                when 34 => seno1 <= "11110011";
                when 35 => seno1 <= "11111001";
                when 36 => seno1 <= "11111101";
                when 37 => seno1 <= "11111111";
                when 38 => seno1 <= "11111111";
                when 39 => seno1 <= "11111101";
                when 40 => seno1 <= "11111001";
                when 41 => seno1 <= "11110011";
                when 42 => seno1 <= "11101011";
                when 43 => seno1 <= "11100010";
                when 44 => seno1 <= "11010111";
                when 45 => seno1 <= "11001010";
                when 46 => seno1 <= "10111101";
                when 47 => seno1 <= "10101110";
                when 48 => seno1 <= "10011111";
                when 49 => seno1 <= "10001111";
                when others => seno1 <=  "00000000";
            end case;
            onda1 <= seno1;
        end process;

end gen_osc;
```

ANEXO C

CÓDIGO EN VHDL DEL GENERADOR DE ONDAS DE DOS AMPLITUDES DE CUATRO FASES CADA UNA.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity osc is
	port (
		contador000, contador010, contador100, contador110 : in natural;
		contador001, contador011, contador101, contador111 : in natural;
		fase000, fase010, fase100, fase110 : out std_logic_vector (7 downto 0);
		fase001, fase011, fase101, fase111 : out std_logic_vector(7 downto 0));
end osc;

architecture gen_osc of osc is
signal onda000, onda010, onda100, onda110 : std_logic_vector(7 downto 0);
signal onda001, onda011, onda101, onda111 : std_logic_vector(7 downto 0);
begin

	p110: process(contador110)
		begin
		case (contador110) is
			when  0 => onda110 <= "00100000";
			when  1 => onda110 <= "00100011";
			when  2 => onda110 <= "00100111";
			when  3 => onda110 <= "00101011";
			when  4 => onda110 <= "00101111";
			when  5 => onda110 <= "00110010";
			when  6 => onda110 <= "00110101";
			when  7 => onda110 <= "00111000";
			when  8 => onda110 <= "00111010";
			when  9 => onda110 <= "00111100";
			when 10 => onda110 <= "00111101";
			when 11 => onda110 <= "00111110";
			when 12 => onda110 <= "00111111";
			when 13 => onda110 <= "00111111";
			when 14 => onda110 <= "00111110";
			when 15 => onda110 <= "00111101";
			when 16 => onda110 <= "00111100";
			when 17 => onda110 <= "00111010";
```

```
                when 18 => onda110 <= "00111000";
                when 19 => onda110 <= "00110101";
                when 20 => onda110 <= "00110010";
                when 21 => onda110 <= "00101111";
                when 22 => onda110 <= "00101011";
                when 23 => onda110 <= "00100111";
                when 24 => onda110 <= "00100011";
                when 25 => onda110 <= "00100000";
                when 26 => onda110 <= "00011100";
                when 27 => onda110 <= "00011000";
                when 28 => onda110 <= "00010100";
                when 29 => onda110 <= "00010000";
                when 30 => onda110 <= "00001101";
                when 31 => onda110 <= "00001010";
                when 32 => onda110 <= "00000111";
                when 33 => onda110 <= "00000101";
                when 34 => onda110 <= "00000011";
                when 35 => onda110 <= "00000010";
                when 36 => onda110 <= "00000001";
                when 37 => onda110 <= "00000000";
                when 38 => onda110 <= "00000000";
                when 39 => onda110 <= "00000001";
                when 40 => onda110 <= "00000010";
                when 41 => onda110 <= "00000011";
                when 42 => onda110 <= "00000101";
                when 43 => onda110 <= "00000111";
                when 44 => onda110 <= "00001010";
                when 45 => onda110 <= "00001101";
                when 46 => onda110 <= "00010000";
                when 47 => onda110 <= "00010100";
                when 48 => onda110 <= "00011000";
                when 49 => onda110 <= "00011100";
                when others => onda110 <="00000000";
        end case;
        fase110 <= onda110;
end process;

p000: process(contador000)
        begin
        case (contador000) is
                when 0 => onda000 <= "00100000";
                when 1 => onda000 <= "00011100";
                when 2 => onda000 <= "00011000";
                when 3 => onda000 <= "00010100";
                when 4 => onda000 <= "00010000";
```

```
when 5 => onda000 <= "00001101";
when 6 => onda000 <= "00001010";
when 7 => onda000 <= "00000111";
when 8 => onda000 <= "00000101";
when 9 => onda000 <= "00000011";
when 10 => onda000 <= "00000010";
when 11 => onda000 <= "00000001";
when 12 => onda000 <= "00000000";
when 13 => onda000 <= "00000000";
when 14 => onda000 <= "00000001";
when 15 => onda000 <= "00000010";
when 16 => onda000 <= "00000011";
when 17 => onda000 <= "00000101";
when 18 => onda000 <= "00000111";
when 19 => onda000 <= "00001010";
when 20 => onda000 <= "00001101";
when 21 => onda000 <= "00010000";
when 22 => onda000 <= "00010100";
when 23 => onda000 <= "00011000";
when 24 => onda000 <= "00011100";
when 25 => onda000 <= "00100000";
when 26 => onda000 <= "00100011";
when 27 => onda000 <= "00100111";
when 28 => onda000 <= "00101011";
when 29 => onda000 <= "00101111";
when 30 => onda000 <= "00110010";
when 31 => onda000 <= "00110101";
when 32 => onda000 <= "00111000";
when 33 => onda000 <= "00111010";
when 34 => onda000 <= "00111100";
when 35 => onda000 <= "00111101";
when 36 => onda000 <= "00111110";
when 37 => onda000 <= "00111111";
when 38 => onda000 <= "00111111";
when 39 => onda000 <= "00111110";
when 40 => onda000 <= "00111101";
when 41 => onda000 <= "00111100";
when 42 => onda000 <= "00111010";
when 43 => onda000 <= "00111000";
when 44 => onda000 <= "00110101";
when 45 => onda000 <= "00110010";
when 46 => onda000 <= "00101111";
when 47 => onda000 <= "00101011";
when 48 => onda000 <= "00100111";
when 49 => onda000 <= "00100011";
```

```
                when others => onda000 <="00000000";
        end case;
        fase000 <= onda000;
end process;

p100: process(contador100)
        begin
        case (contador100) is
                when 0 => onda100 <= "00000000";
                when 1 => onda100 <= "00000001";
                when 2 => onda100 <= "00000010";
                when 3 => onda100 <= "00000011";
                when 4 => onda100 <= "00000101";
                when 5 => onda100 <= "00000111";
                when 6 => onda100 <= "00001010";
                when 7 => onda100 <= "00001101";
                when 8 => onda100 <= "00010000";
                when 9 => onda100 <= "00010100";
                when 10 => onda100 <= "00011000";
                when 11 => onda100 <= "00011100";
                when 12 => onda100 <= "00100000";
                when 13 => onda100 <= "00100011";
                when 14 => onda100 <= "00100111";
                when 15 => onda100 <= "00101011";
                when 16 => onda100 <= "00101111";
                when 17 => onda100 <= "00110010";
                when 18 => onda100 <= "00110101";
                when 19 => onda100 <= "00111000";
                when 20 => onda100 <= "00111010";
                when 21 => onda100 <= "00111100";
                when 22 => onda100 <= "00111101";
                when 23 => onda100 <= "00111110";
                when 24 => onda100 <= "00111111";
                when 25 => onda100 <= "00111111";
                when 26 => onda100 <= "00111110";
                when 27 => onda100 <= "00111101";
                when 28 => onda100 <= "00111100";
                when 29 => onda100 <= "00111010";
                when 30 => onda100 <= "00111000";
                when 31 => onda100 <= "00110101";
                when 32 => onda100 <= "00110010";
                when 33 => onda100 <= "00101111";
                when 34 => onda100 <= "00101011";
                when 35 => onda100 <= "00100111";
                when 36 => onda100 <= "00100011";
```

```
                when 37 => onda100 <= "00100000";
                when 38 => onda100 <= "00011100";
                when 39 => onda100 <= "00011000";
                when 40 => onda100 <= "00010100";
                when 41 => onda100 <= "00010000";
                when 42 => onda100 <= "00001101";
                when 43 => onda100 <= "00001010";
                when 44 => onda100 <= "00000111";
                when 45 => onda100 <= "00000101";
                when 46 => onda100 <= "00000011";
                when 47 => onda100 <= "00000010";
                when 48 => onda100 <= "00000001";
                when 49 => onda100 <= "00000000";
                when others => onda100 <="00000000";
        end case;
        fase100 <= onda100;
end process;

p010: process(contador010)
        begin
        case (contador010) is
                when 0 => onda010 <= "00111111";
                when 1 => onda010 <= "00111110";
                when 2 => onda010 <= "00111101";
                when 3 => onda010 <= "00111100";
                when 4 => onda010 <= "00111010";
                when 5 => onda010 <= "00111000";
                when 6 => onda010 <= "00110101";
                when 7 => onda010 <= "00110010";
                when 8 => onda010 <= "00101111";
                when 9 => onda010 <= "00101011";
                when 10 => onda010 <= "00100111";
                when 11 => onda010 <= "00100011";
                when 12 => onda010 <= "00100000";
                when 13 => onda010 <= "00011100";
                when 14 => onda010 <= "00011000";
                when 15 => onda010 <= "00010100";
                when 16 => onda010 <= "00010000";
                when 17 => onda010 <= "00001101";
                when 18 => onda010 <= "00001010";
                when 19 => onda010 <= "00000111";
                when 20 => onda010 <= "00000101";
                when 21 => onda010 <= "00000011";
                when 22 => onda010 <= "00000010";
                when 23 => onda010 <= "00000001";
```

```
                when 24 => onda010 <= "00000000";
                when 25 => onda010 <= "00000000";
                when 26 => onda010 <= "00000001";
                when 27 => onda010 <= "00000010";
                when 28 => onda010 <= "00000011";
                when 29 => onda010 <= "00000101";
                when 30 => onda010 <= "00000111";
                when 31 => onda010 <= "00001010";
                when 32 => onda010 <= "00001101";
                when 33 => onda010 <= "00010000";
                when 34 => onda010 <= "00010100";
                when 35 => onda010 <= "00011000";
                when 36 => onda010 <= "00011100";
                when 37 => onda010 <= "00100000";
                when 38 => onda010 <= "00100011";
                when 39 => onda010 <= "00100111";
                when 40 => onda010 <= "00101011";
                when 41 => onda010 <= "00101111";
                when 42 => onda010 <= "00110010";
                when 43 => onda010 <= "00110101";
                when 44 => onda010 <= "00111000";
                when 45 => onda010 <= "00111010";
                when 46 => onda010 <= "00111100";
                when 47 => onda010 <= "00111101";
                when 48 => onda010 <= "00111110";
                when 49 => onda010 <= "00111111";
                when others => onda010 <="00000000";
        end case;
        fase010 <= onda010;
end process;

p111: process(contador111)
        begin
        case (contador111) is
                when 0  => onda111 <= "01111111";
                when 1  => onda111 <= "10001111";
                when 2  => onda111 <= "10011111";
                when 3  => onda111 <= "10101110";
                when 4  => onda111 <= "10111101";
                when 5  => onda111 <= "11001010";
                when 6  => onda111 <= "11010111";
                when 7  => onda111 <= "11100010";
                when 8  => onda111 <= "11101011";
                when 9  => onda111 <= "11110011";
                when 10 => onda111 <= "11111001";
```

```
                when 11 => onda111 <= "11111101";
                when 12 => onda111 <= "11111111";
                when 13 => onda111 <= "11111111";
                when 14 => onda111 <= "11111101";
                when 15 => onda111 <= "11111001";
                when 16 => onda111 <= "11110011";
                when 17 => onda111 <= "11101011";
                when 18 => onda111 <= "11100010";
                when 19 => onda111 <= "11010111";
                when 20 => onda111 <= "11001010";
                when 21 => onda111 <= "10111101";
                when 22 => onda111 <= "10101110";
                when 23 => onda111 <= "10011111";
                when 24 => onda111 <= "10001111";
                when 25 => onda111 <= "01111111";
                when 26 => onda111 <= "01101111";
                when 27 => onda111 <= "01100000";
                when 28 => onda111 <= "01010000";
                when 29 => onda111 <= "01000010";
                when 30 => onda111 <= "00110100";
                when 31 => onda111 <= "00101000";
                when 32 => onda111 <= "00011101";
                when 33 => onda111 <= "00010100";
                when 34 => onda111 <= "00001100";
                when 35 => onda111 <= "00000110";
                when 36 => onda111 <= "00000010";
                when 37 => onda111 <= "00000000";
                when 38 => onda111 <= "00000000";
                when 39 => onda111 <= "00000010";
                when 40 => onda111 <= "00000110";
                when 41 => onda111 <= "00001100";
                when 42 => onda111 <= "00010100";
                when 43 => onda111 <= "00011101";
                when 44 => onda111 <= "00101000";
                when 45 => onda111 <= "00110100";
                when 46 => onda111 <= "01000010";
                when 47 => onda111 <= "01010000";
                when 48 => onda111 <= "01100000";
                when 49 => onda111 <= "01101111";
                when others => onda111 <=  "00000000";
        end case;
        fase111 <= onda111;
end process;

p011: process(contador011)
```

```
begin
case (contador011) is
        when 0 => onda011 <= "11111111";
        when 1 => onda011 <= "11111101";
        when 2 => onda011 <= "11111001";
        when 3 => onda011 <= "11110011";
        when 4 => onda011 <= "11101011";
        when 5 => onda011 <= "11100010";
        when 6 => onda011 <= "11010111";
        when 7 => onda011 <= "11001010";
        when 8 => onda011 <= "10111101";
        when 9 => onda011 <= "10101110";
        when 10 => onda011 <= "10011111";
        when 11 => onda011 <= "10001111";
        when 12 => onda011 <= "01111111";
        when 13 => onda011 <= "01101111";
        when 14 => onda011 <= "01100000";
        when 15 => onda011 <= "01010000";
        when 16 => onda011 <= "01000010";
        when 17 => onda011 <= "00110100";
        when 18 => onda011 <= "00101000";
        when 19 => onda011 <= "00011101";
        when 20 => onda011 <= "00010100";
        when 21 => onda011 <= "00001100";
        when 22 => onda011 <= "00000110";
        when 23 => onda011 <= "00000010";
        when 24 => onda011 <= "00000000";
        when 25 => onda011 <= "00000000";
        when 26 => onda011 <= "00000010";
        when 27 => onda011 <= "00000110";
        when 28 => onda011 <= "00001100";
        when 29 => onda011 <= "00010100";
        when 30 => onda011 <= "00011101";
        when 31 => onda011 <= "00101000";
        when 32 => onda011 <= "00110100";
        when 33 => onda011 <= "01000010";
        when 34 => onda011 <= "01010000";
        when 35 => onda011 <= "01100000";
        when 36 => onda011 <= "01101111";
        when 37 => onda011 <= "01111111";
        when 38 => onda011 <= "10001111";
        when 39 => onda011 <= "10011111";
        when 40 => onda011 <= "10101110";
        when 41 => onda011 <= "10111101";
        when 42 => onda011 <= "11001010";
```

```
                when 43 => onda011 <= "11010111";
                when 44 => onda011 <= "11100010";
                when 45 => onda011 <= "11101011";
                when 46 => onda011 <= "11110011";
                when 47 => onda011 <= "11111001";
                when 48 => onda011 <= "11111101";
                when 49 => onda011 <= "11111111";
                when others => onda011 <=  "00000000";
        end case;
        fase011 <= onda011;
end process;

        p101: process(contador101)
        begin
        case (contador101) is
                when  0 => onda101 <= "00000000";
                when  1 => onda101 <= "00000010";
                when  2 => onda101 <= "00000110";
                when  3 => onda101 <= "00001100";
                when  4 => onda101 <= "00010100";
                when  5 => onda101 <= "00011101";
                when  6 => onda101 <= "00101000";
                when  7 => onda101 <= "00110100";
                when  8 => onda101 <= "01000010";
                when  9 => onda101 <= "01010000";
                when 10 => onda101 <= "01100000";
                when 11 => onda101 <= "01101111";
                when 12 => onda101 <= "01111111";
                when 13 => onda101 <= "10001111";
                when 14 => onda101 <= "10011111";
                when 15 => onda101 <= "10101110";
                when 16 => onda101 <= "10111101";
                when 17 => onda101 <= "11001010";
                when 18 => onda101 <= "11010111";
                when 19 => onda101 <= "11100010";
                when 20 => onda101 <= "11101011";
                when 21 => onda101 <= "11110011";
                when 22 => onda101 <= "11111001";
                when 23 => onda101 <= "11111101";
                when 24 => onda101 <= "11111111";
                when 25 => onda101 <= "11111111";
                when 26 => onda101 <= "11111101";
                when 27 => onda101 <= "11111001";
                when 28 => onda101 <= "11110011";
                when 29 => onda101 <= "11101011";
```

```
                when 30 => onda101 <= "11100010";
                when 31 => onda101 <= "11010111";
                when 32 => onda101 <= "11001010";
                when 33 => onda101 <= "10111101";
                when 34 => onda101 <= "10101110";
                when 35 => onda101 <= "10011111";
                when 36 => onda101 <= "10001111";
                when 37 => onda101 <= "01111111";
                when 38 => onda101 <= "01101111";
                when 39 => onda101 <= "01100000";
                when 40 => onda101 <= "01010000";
                when 41 => onda101 <= "01000010";
                when 42 => onda101 <= "00110100";
                when 43 => onda101 <= "00101000";
                when 44 => onda101 <= "00011101";
                when 45 => onda101 <= "00010100";
                when 46 => onda101 <= "00001100";
                when 47 => onda101 <= "00000110";
                when 48 => onda101 <= "00000010";
                when 49 => onda101 <= "00000000";
                when others => onda101 <=  "00000000";
        end case;
        fase101 <= onda101;
end process;

p001: process(contador001)
        begin
        case (contador001) is
                when 0 => onda001 <= "01111111";
                when 1 => onda001 <= "01101111";
                when 2 => onda001 <= "01100000";
                when 3 => onda001 <= "01010000";
                when 4 => onda001 <= "01000010";
                when 5 => onda001 <= "00110100";
                when 6 => onda001 <= "00101000";
                when 7 => onda001 <= "00011101";
                when 8 => onda001 <= "00010100";
                when 9 => onda001 <= "00001100";
                when 10 => onda001 <= "00000110";
                when 11 => onda001 <= "00000010";
                when 12 => onda001 <= "00000000";
                when 13 => onda001 <= "00000000";
                when 14 => onda001 <= "00000010";
                when 15 => onda001 <= "00000110";
                when 16 => onda001 <= "00001100";
```

```
                when 17 => onda001 <= "00010100";
                when 18 => onda001 <= "00011101";
                when 19 => onda001 <= "00101000";
                when 20 => onda001 <= "00110100";
                when 21 => onda001 <= "01000010";
                when 22 => onda001 <= "01010000";
                when 23 => onda001 <= "01100000";
                when 24 => onda001 <= "01101111";
                when 25 => onda001 <= "01111111";
                when 26 => onda001 <= "10001111";
                when 27 => onda001 <= "10011111";
                when 28 => onda001 <= "10101110";
                when 29 => onda001 <= "10111101";
                when 30 => onda001 <= "11001010";
                when 31 => onda001 <= "11010111";
                when 32 => onda001 <= "11100010";
                when 33 => onda001 <= "11101011";
                when 34 => onda001 <= "11110011";
                when 35 => onda001 <= "11111001";
                when 36 => onda001 <= "11111101";
                when 37 => onda001 <= "11111111";
                when 38 => onda001 <= "11111111";
                when 39 => onda001 <= "11111101";
                when 40 => onda001 <= "11111001";
                when 41 => onda001 <= "11110011";
                when 42 => onda001 <= "11101011";
                when 43 => onda001 <= "11100010";
                when 44 => onda001 <= "11010111";
                when 45 => onda001 <= "11001010";
                when 46 => onda001 <= "10111101";
                when 47 => onda001 <= "10101110";
                when 48 => onda001 <= "10011111";
                when 49 => onda001 <= "10001111";
                when others => onda001 <=  "00000000";
            end case;
            fase001 <= onda001;
        end process;

end gen_osc;
```

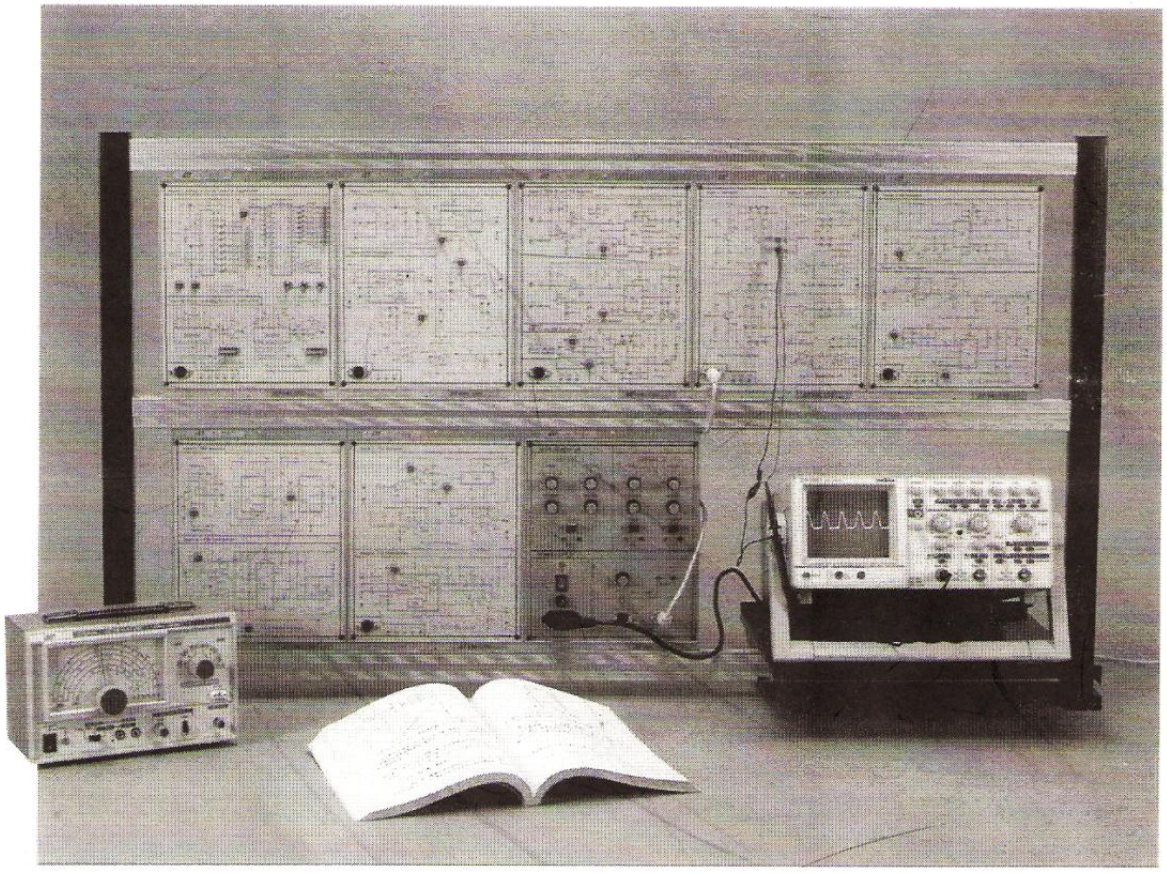
ANEXO D

Basic Communication Trainer - Experiment Manual of KING Instrument Electronics CO., LTD.

KL-900A

Basic Communication Trainer

MODULE EXPERIMENT MANUAL



K&H MFG CO., LTD.
5F, No. 8, Sec. 4 Tzu-Chiang Rd., San Chung City 241, Taipei Hsien, Taiwan R.O.C.
TEL：886-2-2286-0700 FAX：886-2-2287-3066
E-Mail：education@kandh.com.tw WEB：http://www.kandh.com.tw

Chapter 13

FSK MODULATORS

13.1 OBJECTIVES

1. Understanding the principle of frequency-shift keying (FSK) modulation.
2. Measuring FSK signals.
3. Implementing an FSK modulator with LM566.

13.2 DISCUSSION OF FUNDAMENTALS

In digital transmission repeaters can regenerate digital signals and improve the ability against noise interference, and the use of encoding techniques can provide debugging and correction functions. But digital signals often occur distortions due to its high-frequency components are easily attenuated for a long distance transmission. To improve this disadvantage, a particular processing (modulation) is need for this purpose. Frequency-shift keying (FSK) is a type of FM in which the modulating signal (digital signal) shifts the output between two predetermined frequencies - usually termed the mark and space frequencies. The relationship between FSK and digital signals is shown in Fig. 13-1. The FSK frequency f_1 corresponds to the digital input high, and the f_2 represents the digital low.

FSK technique is widely used for the transmission of Teletype information. FSK standards have evolved for the years. For radio Teletype, the frequency of 2124Hz represents mark or 1, and 2975 Hz represents space or 0.

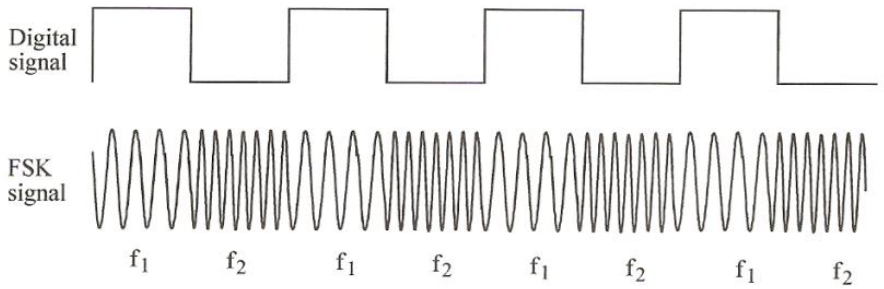


Fig.13-1 Relationship between digital and FSK signals

For data transmission over telephone and landlines, the commonly used frequencies are:

space = 1270 Hz
mark = 1070 Hz

and

space = 2225 Hz
mark = 2025 Hz

Notice that the frequency difference (gap) of FSK signal equals 200 Hz.

The FSK modulator is used to convert the digital signal (square wave) into the analog signal having two different frequencies corresponding to the input levels. In this experiment, we use the frequencies of 1070 Hz and 1270 Hz to represent space and mark, respectively. A voltage-controlled Oscillator (VCO) can easily generate these two frequencies. A practical FSK modulator using the LM566 VCO is shown in Fig.13-2. In such cases, the oscillating frequency of LM566 can be found by

$$f_0 = \frac{2}{R_{10}C_5}\left(\frac{V_{cc} - V_{in}}{V_{cc}}\right)$$

where V_{cc} is the power voltage applied to LM566 pin 8, and V_{in} is the VCO control voltage applied to pin 5.

If V_{cc} is constant, proper values of R_{10}, C_5 and V_{in} are determined to generate the LM566 output frequencies f_o of 1072 Hz and 1272 Hz. In practice, the limitations of using LM566 VCO are as follows:

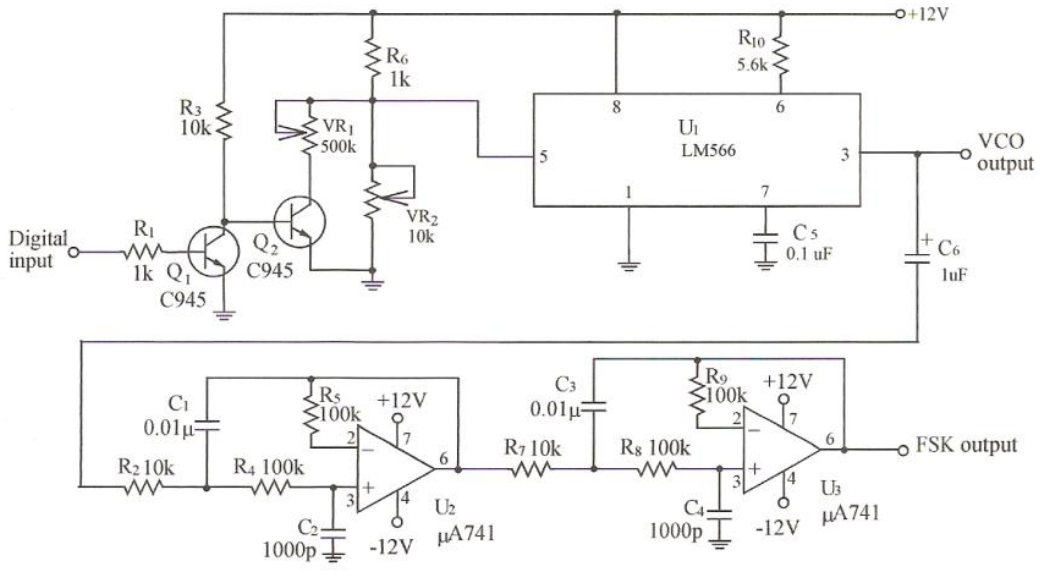


Fig.13-2 FSK modulator circuit.

and the output frequency is f_2. Therefore the output frequencies f_1 = 1270 Hz and f_2 = 1070 Hz can be obtained by carefully adjusting the VR_1 and VR_2 values. Both U_2 and U_3 are the second-order low-pass filters. The fourth-order low-pass filter formed by cascading these filters is used to filter the high-frequency harmonic components on the output of LM566 and therefore the FSK modulated signal is obtained.

If the FSK modulated signal mentioned above is desired to transmit by an antenna, a mixer is required to modulate the signal to the frequency range in RF band.

13.3 EQUIPMENT REQUIRED

1 - Module KL-92001
2 - Module KL-94003
3 - Oscilloscope

13.4 EXPERIMENTS AND RECORDS

Experiment 13-1 FSK Modulator

☐1. Locate the FSK modulator circuit on Module KL-94003.

☐2. Connect 5Vdc to digital signal input (I/P). Using the oscilloscope, observe the LM566 output frequency (pin 3) and adjust VR_2 to obtain the frequency of 1070Hz, and then record the result in Table 13-1.

☐3. Using the oscilloscope, observe and record the FSK output signal in Table 13-1.

☐4. Connect digital signal input (I/P) to ground (0V). Using the oscilloscope, observe the LM566 output frequency (pin 3) and adjust VR_1 to obtain the frequency of 1270Hz, and record the result in Table 13-1.

☐5. Using the oscilloscope, observe and record the FSK output signal in Table 13-1.

☐6. Set the output of signal generator to TTL level and the frequency of 200 Hz and then connect the output to the digital signal input (I/P). Using the oscilloscope, observe and record the input, LM566 output (pin 3), and FSK output signals in Table 13-2.

☐7. Change the output frequency of signal generator to 5kHz and repeat step 6.

Chapter 18

ASK SYSTEM

18.1 OBJECTIVES

1. To study the principles of ASK modulation and demodulation.
2. To implement an ASK modulator.
3. To implement coherent and noncoherent ASK demodulators.

18.2 DISCUSSION OF FUNDAMENTALS

When it is required to transmit digital data over a bandpass channel, it is necessary to modulate the incoming data onto a carrier wave with fixed frequency limits imposed by the channel. The data may represent digital computer output or PCM waves generated by digitizing audio or video signals. The channel may be a telephone channel, microwave radio link, or a satellite channel.

Modulation is defined as the process by which some characteristic of a carrier is varied in accordance with a modulating wave. In digital communications, the modulating wave consists of binary data or an M-ary encoded version of it. For the carrier, it is customary to use a sinusoidal wave. With a sinusoidal carrier, the feature that is used by the modulator to distinguish one signal from another is a step change in the amplitude, frequency, or phase of the carrier. The result of this modulation process is amplitude-shift keying (ASK), frequency-shift keying (FSK), or phase-shift keying (PSK), which may be viewed as special cases of amplitude modulation, frequency modulation, and phase modulation, respectively.

ASK Modulator

An ASK modulated signal can be expressed as

$$X_{ASK}(t)=Ai \cos (\omega_c t+\Phi_0) \qquad 0 \le t \le T, \quad i=1, 2,\ldots, M,$$

where the amplitude Ai has M possible values, the angular frequency ω_c and the phase Φ_0 are constant. If M=2 (A_1=0 and A_2 =A, where A is an

arbitrary constant), $X_{ASK}(t)$ will be a binary ASK modulated signal as shown in Figure 18-1. The ASK signal transmits a binary message which is on when the modulating data is a logic high and off when the modulating signal is a logic low. It is also called On-Off Keying (OOK) modulation.

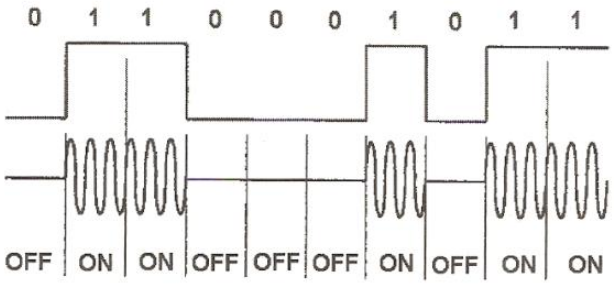


Figure 18-1 ASK modulated signal

Figure 18-2 shows an ASK modulator. The A represents a dc bias, the sinusoidal carrier $V_C(t)=A_C \cos 2\pi f_C t$, and the modulating signal $V_D(t)$ is a binary data. The modulated signal $V_T(t)$ can be expressed as

$$V_T(t) = [V_D(t) + A\,]A_C \cos 2\pi f_C t$$

The waveforms of $V_D(t)$, $[V_D(t) + A]$ and $V_T(t)$ are shown in Figure 18-3. Clearly the ASK modulated signal $V_T(t)$ consists of two levels $[V_D(t)+V_L]A_C$ and $[V_D(t)+V_H]A_C$ corresponding to V_L and V_H of the modulating signal $V_D(t)$, respectively.

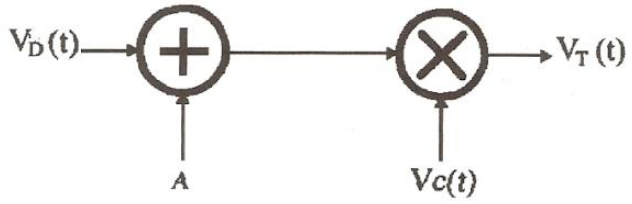
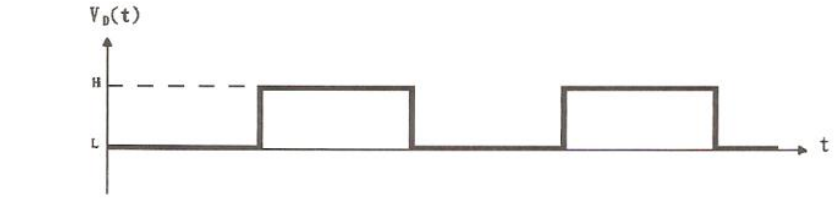
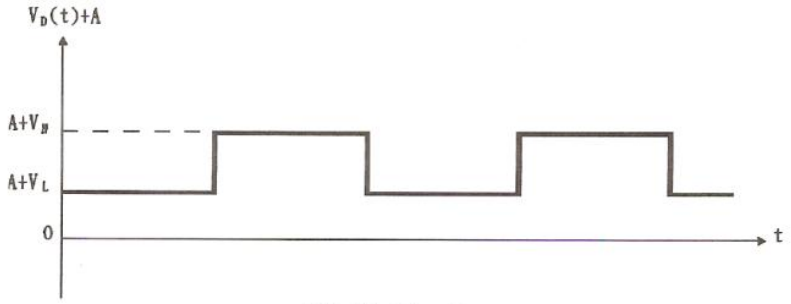


Figure 18-2 Block diagram of ASK modulator

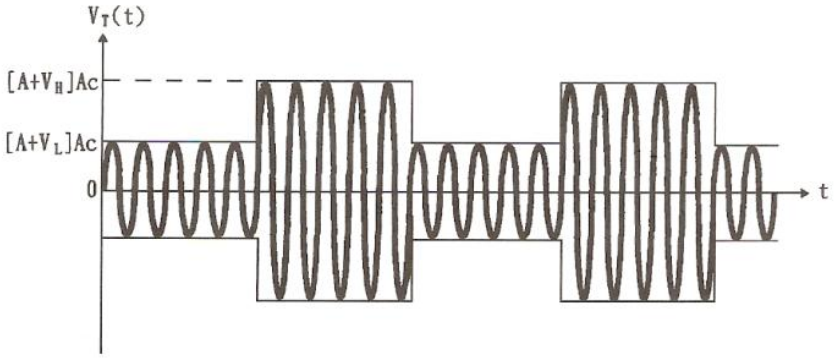
A digital communication system is referred to as coherent if a local reference is available for demodulation that is in phase with the transmitted carrier (with fixed phase shifts due to transmission delays accounted for). Otherwise, it is referred to as noncoherent. Likewise, if a periodic signal is available at the receiver that is synchronous with the transmitted sequence of digital signals (referred to as a clock), the system is referred to as synchronous; if a signaling technique is employed in which such a clock is unnecessary, the system is called asynchronous.



(a) $V_D(t)$



(b) $V_D(t)+A$



(c) $V_T(t)$

Figure 18-3 ASK modulator waveforms

ASK Demodulator

ASK demodulation is a process that restores the digital modulating signal from the ASK signal received. Figure 18-4 shows the operation of ASK demodulation.The electronic circuit that perform ASK demodulation is called ASK demodulator. ASK demodulators can be categorized into two types: noncoherent and coherent demodulators.

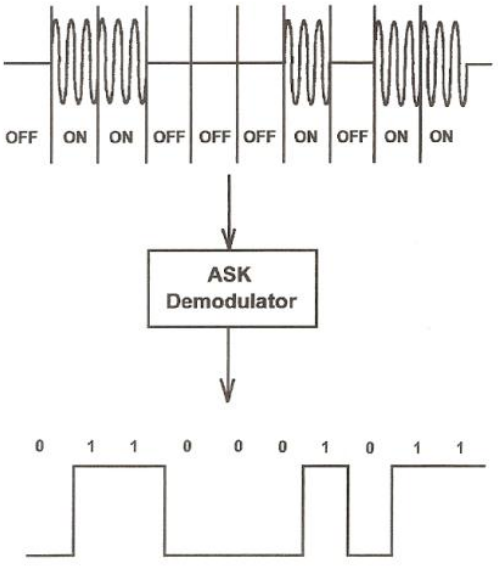


Figure 18-4 ASK demodulation

A. Noncoherent ASK Demodulator

Figure 18-5 shows the functional blocks and waveforms of a noncoherent ASK demodulator. The envelope detector removes the high-frequency carrier and blocks the negative half of the received ASK signal V_R. The output of the envelope detector, V_E, is therefore the positive envelope plus dc and sawtooth components. The dc component is blocked by ac-coupling and the high-frequency sawtooth component is rejected by the lowpass filter.

The voltage comparator compares the LPF output $V_{LP}(t)$ with a fixed threshold voltage and produces the digital output signal V_O equal to the original modulating signal.

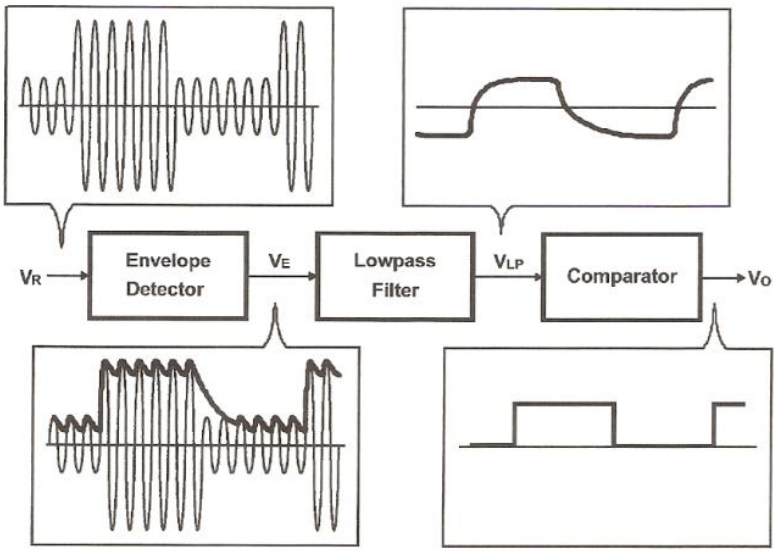


Figure 18-5 Block diagram of noncoherent ASK demodulator

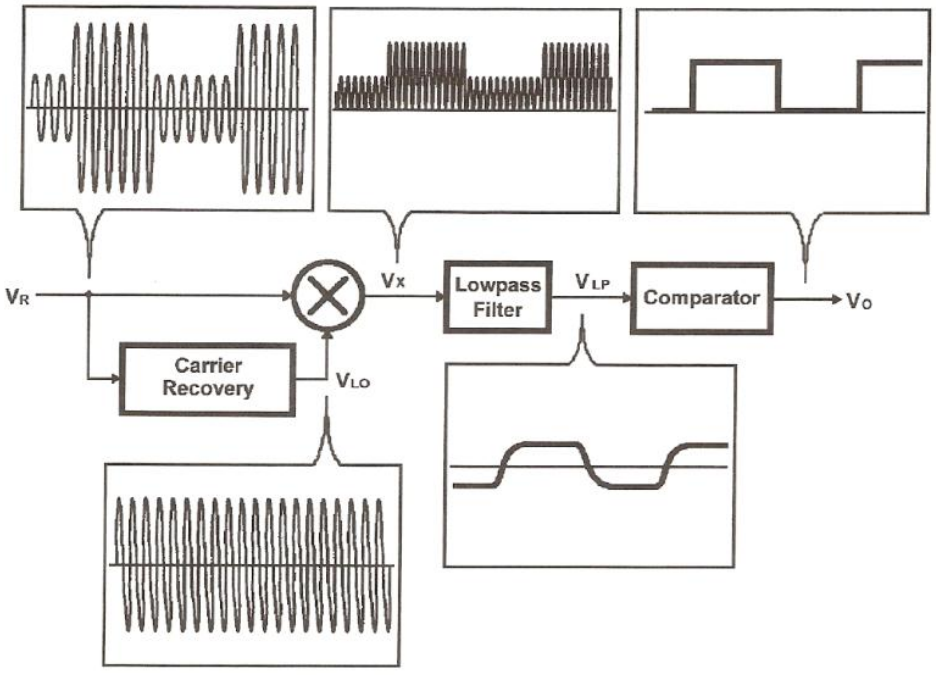


Figure 18-6 Block diagram of coherent ASK demodulator

B. Coherent ASK Demodulator

Figure 18-6 shows the block diagram of coherent ASK demodulator. The received ASK signal $V_R(t)$ is equal to the transmitted ASK signal $V_T(t)$.

$$V_R(t) = V_T(t) = [V_D(t) + A\,]A_R \cos 2\pi f_C t$$

The carrier signal $V_{LO}(t)$ is recovered by the carrier recovery circuit from the $V_R(t)$.

$$V_{LO}(t) = A_{LO} \cos(2\pi f_C t + \Phi)$$

When the received ASK signal $V_R(t)$, and the reconstructed carrier signal $V_{LO}(t)$ are connected to the inputs of the multiplier, the multiplier output becomes

$$\begin{aligned} V_X(t) &= [V_D(t) + A]A_R A_{LO} \cos 2\pi f_C t \cos(2\pi f_C t + \Phi) \\ &= [(A A_R A_{LO})/2] \cdot \cos\Phi + [(A_R A_{LO})/2] \cdot \cos\Phi V_D(t) + \\ &\quad [V_D(t) + A] \cdot [(A_R A_{LO})/2] \cdot \cos(2\pi 2f_C t + \Phi) \end{aligned}$$

The first term of the equation is a dc component, the second term the modulating digital signal, and the third term is the ASK signal with a frequency of 2fc, twice the carrier frequency. The dc component is blocked by ac-coupling and the high-frequency sawtooth component is rejected by the lowpass filter.

The voltage comparator compares the LPF output $V_{LP}(t)$ with a fixed threshold voltage and produces the digital output signal V_O equal to the original modulating signal.

Practical Circuit Description

1. ASK Modulator

The practical ASK modulator is shown in Figure 18-7. The multiplier(1) performs the function of ASK modulation.

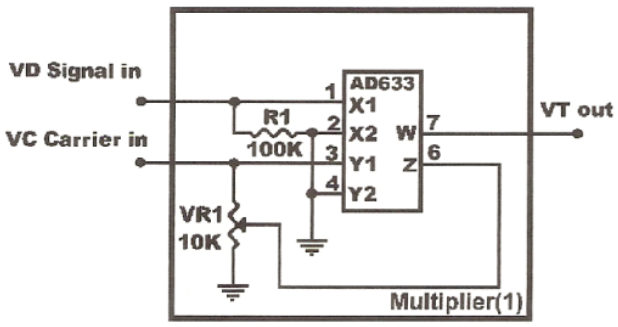


Figure 18-7 ASK modulator

The multiplier output $V_T(t)$ is expressed as

$$V_T(t) = \frac{V_D(t)V_C(t)}{10} + \alpha\, V_C(t)$$

The α value is divided by VR1 potentiometer. If the carrier $V_C(t)=A_C \cos 2\pi f_C t$, $V_T(t)$ becomes

$$V_T(t) = [\frac{1}{10} V_D(t) + \alpha\,]A_C \cos 2\pi f_C t$$

(1) The digital modulating signal $V_D(t)$ has two voltage levels: V_H = 5V and V_L = 0V.
If $V_D(t)= V_H = 5V$, then $V_T(t) \fallingdotseq (0.5+\alpha)\, A_C \cos 2\pi f_C t$
If $V_D(t)= V_L = 0V$, then $V_T(t) \fallingdotseq \alpha\, A_C \cos 2\pi f_C t$

(2) ASK modulated signal $V_T(t)$ has two discrete voltages: the $(0.5+\alpha)A_C$ represents a high and the other (αA_C) represents a low.

(3) If α =0, two voltage levels of $V_T(t)$ are $0.5A_C$ and 0. This is also called an On-Off Keying (OOK) modulation.

2. ASK Demodulator

A. Noncoherent ASK demodulator is shown in Figure 18-8.

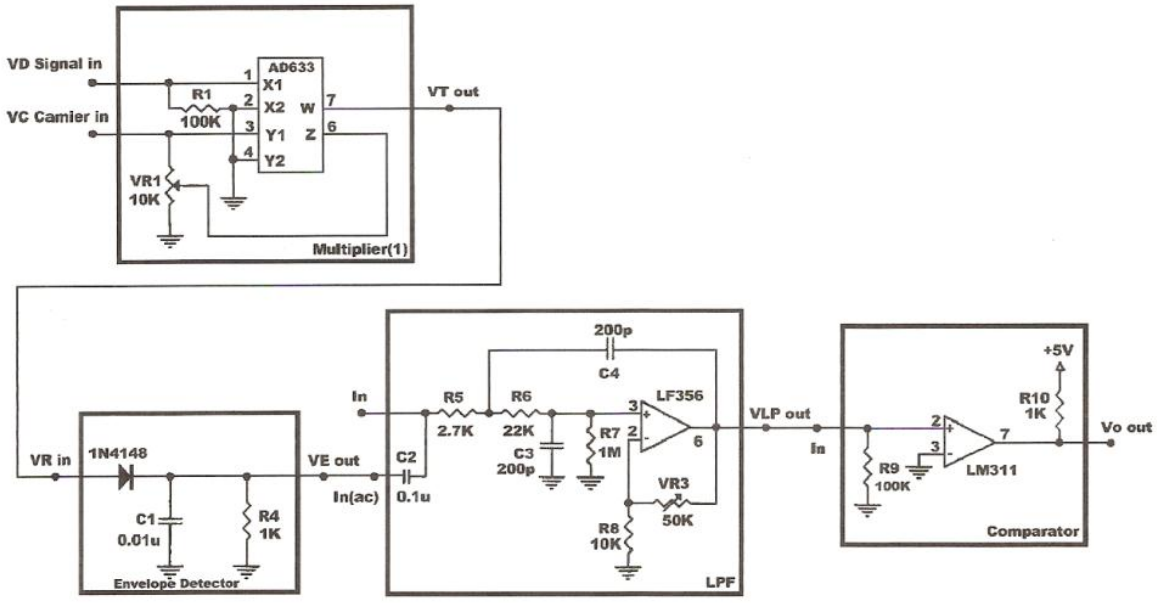


Figure 18-8 Noncoherent ASK demodulator

(1) Multiplier(1) operates as an ASK modulator.

(2) Envelope detector blocks the negative half of V_R in signal and detects the positive half.

(3) Lowpass filter (LPF) rejects the sawtooth component of V_E out signal. The dc component of V_E out signal is blocked by the coupling capacitor C2 when the signal is connected to the In(ac) terminal.

(4) Comparator shapes the LPF output signal (V_{LP} out) to a digital signal with two voltage levels 0V and 5V.

B. Coherent ASK demodulator is shown in Figure 18-9.

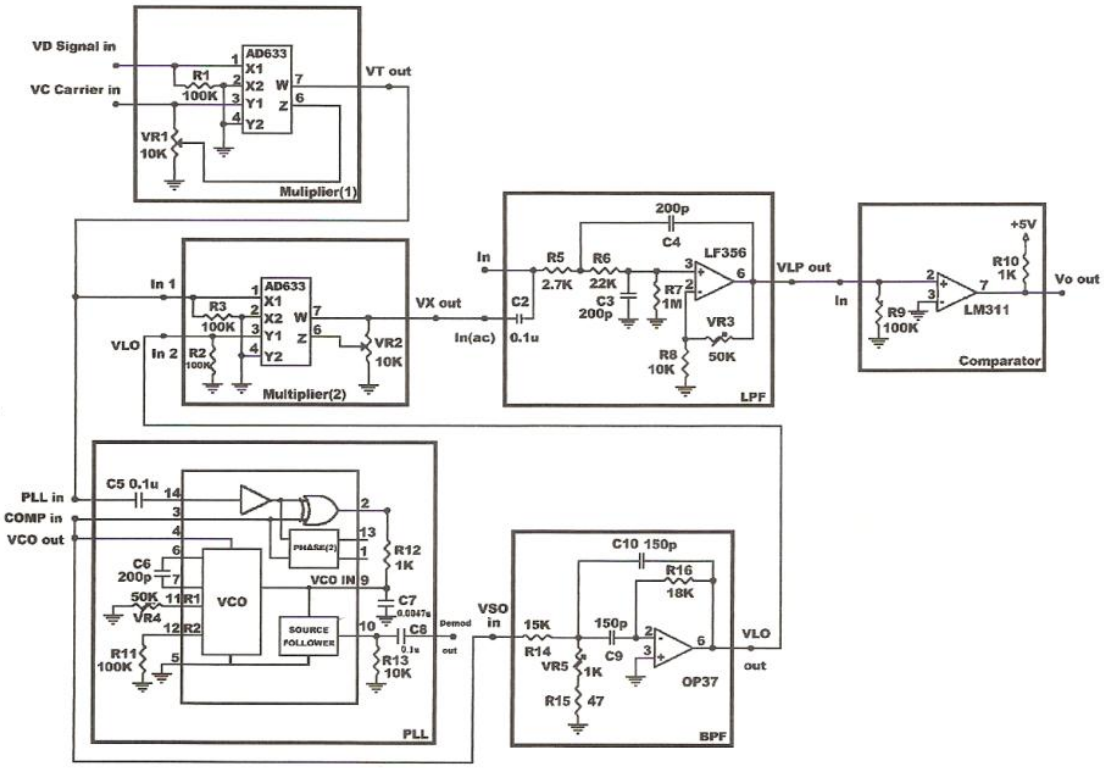


Figure 18-9 Coherent ASK demodulator

(1) Multiplier(1) operates as the ASK modulator that converts the digital modulating signal into an ASK modulated signal.

(2) Phase-Locked Loop (PLL) and bandpass filter (BPF) construct a carrier recovery circuit which reconstructs the carrier signal. The frequency of the recovered carrier signal on V_{LO} out terminal is equal to the original carrier in transmitter. The phase can be synchronized to the original carrier by turning the potentiometer VR5.

(3) Multiplier(2) performs the multiplication of the received ASK signal and the recovered carrier signal.

(4) Lowpass filter (LPF) is used to reject the high-frequency components of Multiplier(2) output signal (Vx out). The dc component is blocked by the ac-coupling capacitor C2.

(5) Comparator compares the V_{LP} out signal with ground potential and recovers the original modulating signal.

The KL-94005 module shown in Figure 18-10 consists of ASK modulator, noncoherent ASK demodulator, and coherent ASK demodulator.

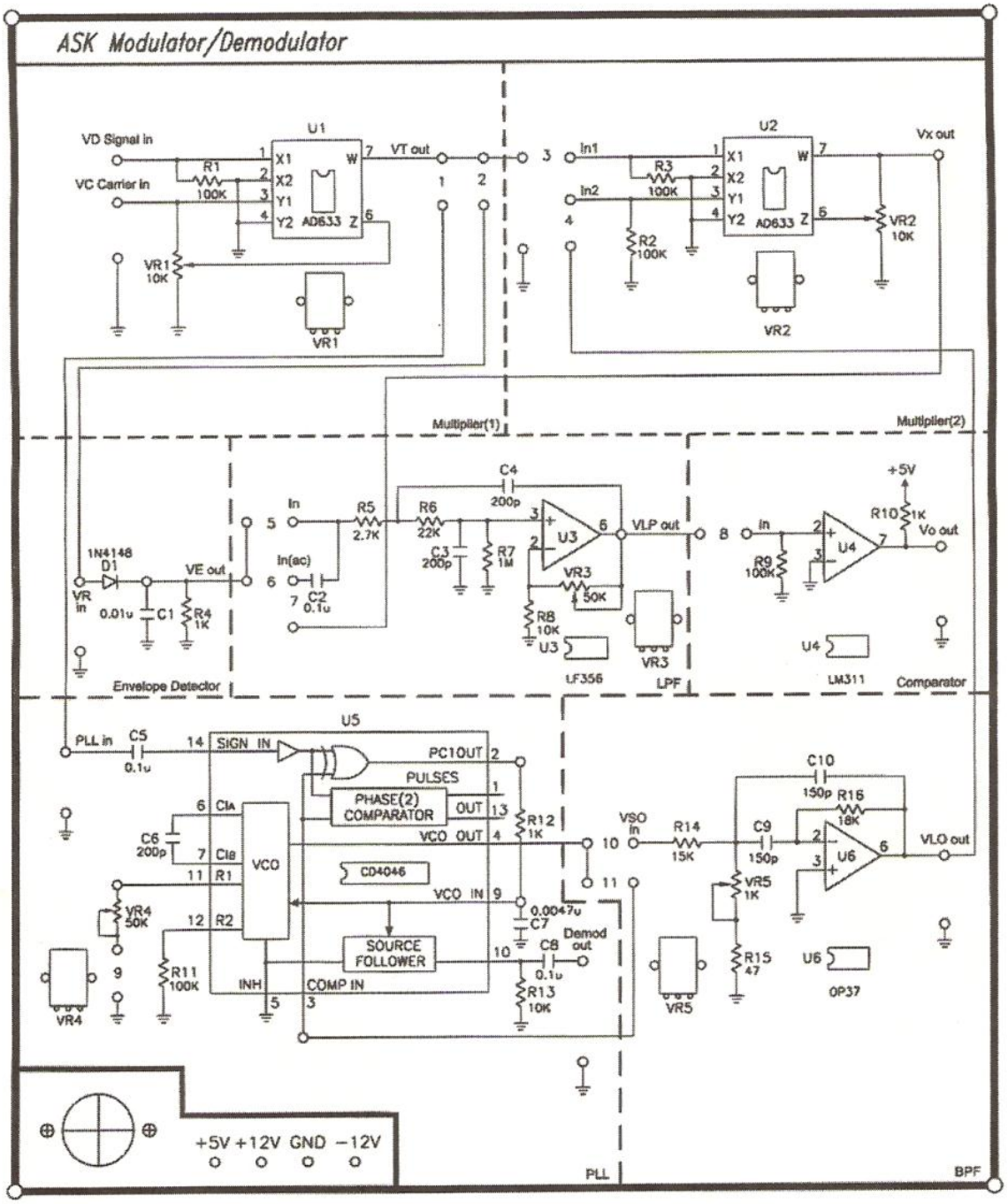


Figure 18-10 KL-94005 module

18.3 EQUIPMENT REQUIRED

1 - Module KL-92001
2 - Module KL-94005
3 - Oscilloscope

18.4 EXPERIMENTS AND RECORDS

Experiment 18-1 ASK Modulator

1. Locate the ASK modulator circuit shown in Figure 18-7 on the KL-94005 module.

2. Connect a 500KHz, 4Vpp sinewave to the VC Carrier in terminal.

3. Connect a 20KHz, TTL-level square wave from Function Generator TTL/CMOS out to the VD Signal in terminal.

4. Turn the VR1 fully CW to obtain a maximum amplitude of ASK modulated signal on VT out. Measure and record the ASK signal waveform in Table 18-1.

5. Turn the VR1 fully CCW to obtain a minimum amplitude of ASK modulated signal on the VT out. Measure and record the ASK signal waveform in Table 18-1.

6. Connect a 1KHz, TTL-level square wave from Function Generator TTL/CMOS out to the VD Signal in terminal.

7. Repeat steps 4 and 5.

8. Connect a 10KHz, TTL-level square wave from Function Generator TTL/CMOS out to the VD Signal in terminal.

9. Repeat steps 4 and 5.

10. Connect a 50KHz, TTL-level square wave from Function Generator TTL/CMOS out to the VD Signal in terminal.

11. Repeat steps 4 and 5.

Chapter 19

PSK/QPSK SYSTEM

19.1 OBJECTIVES

1. To study the principle of PSK/QPSK modulation.
2. To study the principle of PSK/QPSK demodulation.
3. To implement PSK/QPSK modulator.
4. To implement PSK/QPSK demodulator.

19.2 DISCUSSION OF FUNDAMENTALS

PSK/QPSK Modulator

As mentioned in Chapter 18, phase-shift keying (PSK) modulation process may be viewed as the special case of phase modulation (PM). The PSK modulation is shown in Figure 19-1.

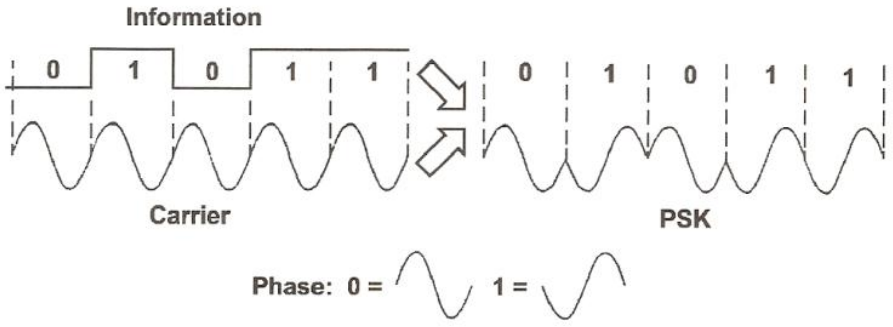


Figure 19-1 PSK modulation

In Figure 19-1, the carrier signal is a sinusoidal wave with fixed frequency and amplitude, the modulating signal is binary information. If input information is a low (0), the carrier signal maintains its phase. If input information is a high (1), the carrier reverses its phase by 180 degrees. The pair of sinusoidal waves that differ only in a relative phase-shift of 180 degrees are referred to as antipodal signals. This type of phase-shift keying is called binary PSK (BPSK) or phase-reverse keying (PRK).

As with BPSK, this modulation scheme is characterized by the fact that the information carried by the transmitted wave is contained in the phase. In particular, in quadriphase-shift keying (QPSK), the phase of the carrier takes on one of four equally spaced values, such as 0°, 90°, 180°, and 270°. Each possible value of the phase corresponds to a unique pair of bits called a dibit. For example, we may choose the forgoing set of phase values to represent the Gray-coded set of dibits: 00, 01, 11, and 10. The typical waveforms of QPSK modulation are shown in Figure 19-2.

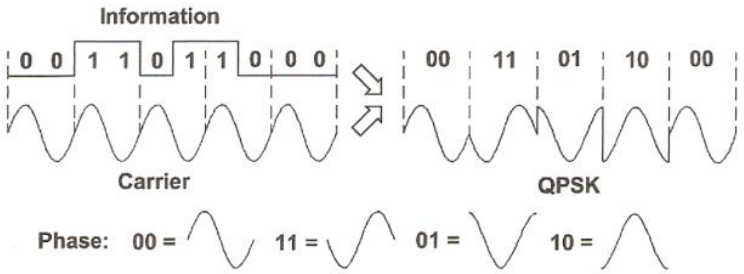


Figure 19-2 QPSK modulation

Note that the phase set of PSK and QPSK mentioned above is only a possible choice. Other possible phase-shifts of PSK and QPSK signals are shown in Table 19-1.

Table 19-1 Possible phase-shifts of PSK and QPSK

System	Information	Phase (degrees)		
		Learned	#1	#2
PSK	0	0	180	45
	1	180	0	225
QPSK	00	0	180	45
	11	180	0	225
	01	90	270	135
	10	270	90	315

Figure 19-3 shows a PSK/QPSK communication system. The modulator modulates the carrier signal by input information and produces a PSK or QPSK modulated signal. The modulated signal is transmitted through transmission medium, such as air, cable, and optical fiber, to the

input of demodulator. The demodulator receives PSK or QPSK transmitted signal and then reconstructs the original information data.

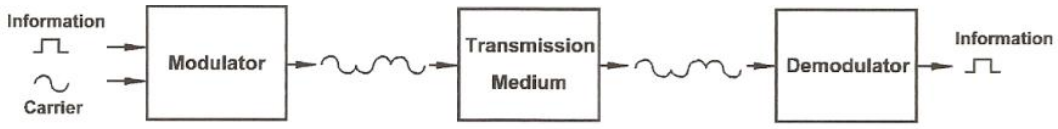


Figure 19-3 Block diagram of PSK/QPSK system

Figure 19-4 shows the functional blocks of a PSK/QPSK modulator. The carrier signal generator provides a carrier signal (sinusoidal wave) to the phase switching network and a square wave to the timing circuit. Phase-switching network provides four outputs (0°, 90°, 180°, 270°) to the inputs of data selector. The output X of data selector is determined by the select inputs A and B. There are four following cases:

1. If BA=00 (Q1=Q0=low), X=X0, the signal with phase shift 0°.
2. If BA=11 (Q1=Q0=high), X=X3, the signal with phase shift 180°.
3. If BA=01 (Q1=low, Q0=high), X=X1, the signal with phase shift 90°.
4. If BA=10 (Q1=high, Q0=low), X=X2, the signal with phase shift 70°.

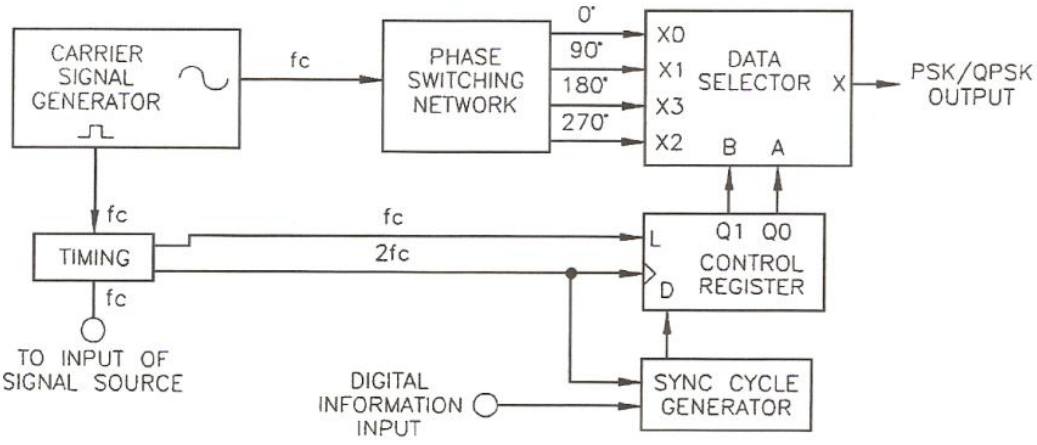


Figure 19-4 Block diagram of PSK/QPSK modulator

The timing circuit receives the square wave (fc) from the carrier signal generator output and produces two outputs: fc to the load control input and signal 2fc (twice the carrier frequency) to the clock input of control register as well as the sync cycle generator. These two signals of fc and 2fc and the data rate (measured in bits per second, bps) of input digital

information are used to determine whether the modulator operates in binary PSK or QPSK mode. There are three possible cases:

1. Bit rate = fc and no sync cycle generated

   In this case, the data rate is equal to the carrier frequency fc and the clock frequency is twice the carrier frequency 2fc. One bit of digital data stream is loaded into the control register two times. The outputs Q0-Q1 of control register are therefore the same, 00 or 11. The output X of data selector is either X0 or X3 input signal. This system operates in PSK mode.

2. Bit rate=2fc and no sync cycle generated

   In this case, data rate and clock frequency are equal to twice the carrier frequency, 2fc. Two bits of data stream are loaded to the control register each carrier cycle. The control register outputs Q0-Q1 may be 00, 01, 11, or 10. This system therefore operates in QPSK mode.

3. Bit rate = fc or 2fc, and sync cycle generated

   If a sync cycle is required, sync cycle control circuit will produce a control signal to control the output data of control shift register, and then a sync cycle signal will present at modulator output. The format of sync cycle shown in Figure 19-5 is used in our experiments. This sync cycle signal is different from the PSK/QPSK modulated signals shown in Figures 19-1 and 19-2. The sync cycle signal can be identified by the sync cycle detector in PSK/QPSK demodulator and can be viewed as an identification word.

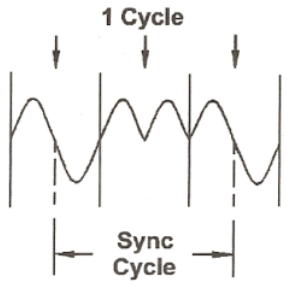


Figure 19-5 Synchronization cycle

PSK/QPSK Demodulator

There are many methods and circuits used to reconstruct the information (modulating signal) from PSK/QPSK modulated signal. In typical PSK/QPSK demodulator, PLL circuit is required for reconstructing clock signal that is used in modulator.

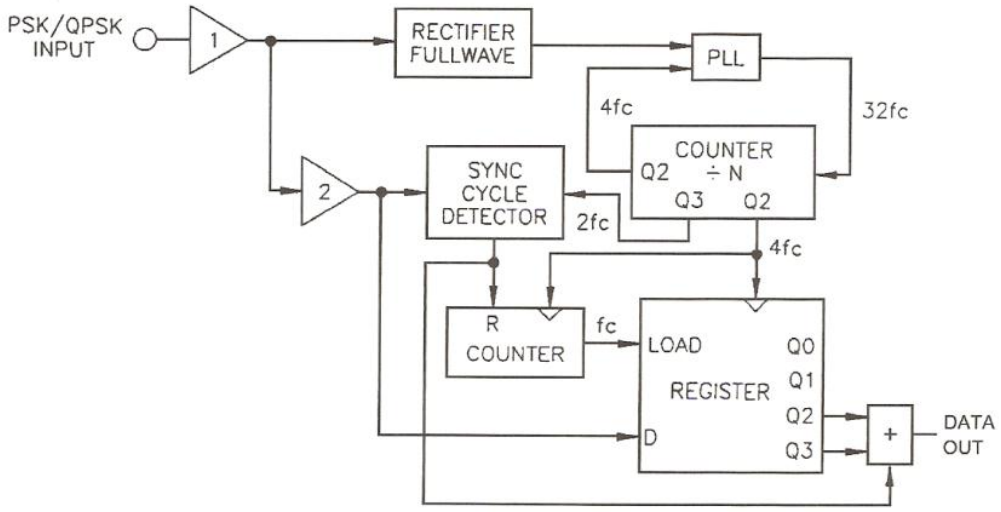


Figure 19-6 Block diagram of PSK/QPSK demodulator

Figure 19-6 shows the block diagram of PSK/QPSK demodulator. PSK/QPSK input signal is amplified by amplifier 1 and then rectified by full-wave rectifier, then the rectified pulse is connected to the input of the phase detector in PLL. This signal is used to reconstruct the clock signal.

Figure 19-7 shows the demodulator output data that is recovered from the received PSK/QPSK signal. From Figure 19-6, we see that various clock frequencies are produced by the PLL and divide-by-N counter. These clock signals are used to reconstruct the information data and to convert sync cycle signal to sync cycle data.

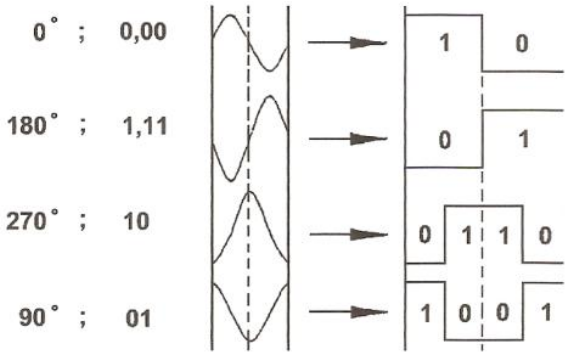


Figure 19-7 Demodulator output data

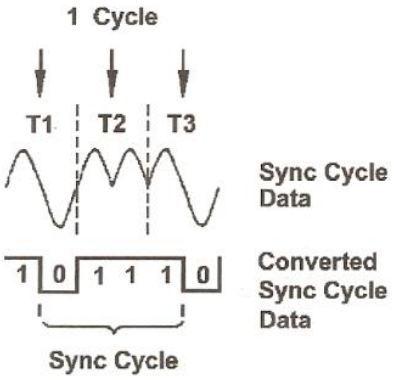


Figure 19-8 Sync cycle conversion

The sync cycle conversion is shown in Figure 19-8. The converted sync cycle data word is 0111. When the sync cycle detector receives this data word, a low is presented to indicate that a sync cycle is detected.

Practical Circuit Description

1. PSK/QPSK modulator

Figure 19-9 shows the schematic diagram of PSK/QPSK modulator. The precision waveform generator chip ICL8038 serves as the carrier signal generator that produces sinusoidal and square waves. The

frequency of the carrier generator is determined by the external timing resistors R2-R3 and capacitor C2 and is 7.1KHz approximately. Pins 7 and 8 are connected together to characterize the generator operating in VCO mode. The generated sinusoidal signal is connected to the inputs of phase switching network consisting of two noninverting amplifiers (U2a and U2d) and two inverting amplifiers (U2b and U2c). This phase-shift network provides four phase shifts 0°, 90°, 180°, and 270° to data inputs X0, X1, X2, and X3 of the data selector (U3), respectively. The output of data selector is determined by the state of select inputs A and B. Once the output is selected, the PSK/QPSK modulated signal is amplified by the non-inverting amplifier U8. The potentiometer VR5 is used to control the output amplitude of PSK/QPSK modulated signal.

The square wave present at U1 pin9 is connected to the input of timing circuit to generate a signal with the frequency 2fc twice the carrier frequency by the frequency doubling network constructed by U4b, U4c, and U5a and associated components R21, R22, C6, C7. The 2fc signal is connected to the clock inputs of the shift register U7 and the 4-bit binary counter U6a. The signal of the counter output Q0 is connected to inverters U4f and U4d and U7 pin1 (Load input). The frequency of this signal is fc. The modulating signal (digital information) is connected to DATA input (pin 2) of the control shift register U7. The outputs Q0-Q1 of the shift register are XORed with the signal on TP6, and then connected to the select inputs A and B of data selector.

Binary counters U6a and U6b are used to determine when to generate a sync cycle. The clock frequency of the binary counter U6a is 2fc. The output Q1 of U6a is connected to clock input of U6b, so that the clock frequency is fc/2 and the frequency of U6b Q3 is fc/32. The sync cycle is generated only in one-half duration of Q3 output signal or fc/16.

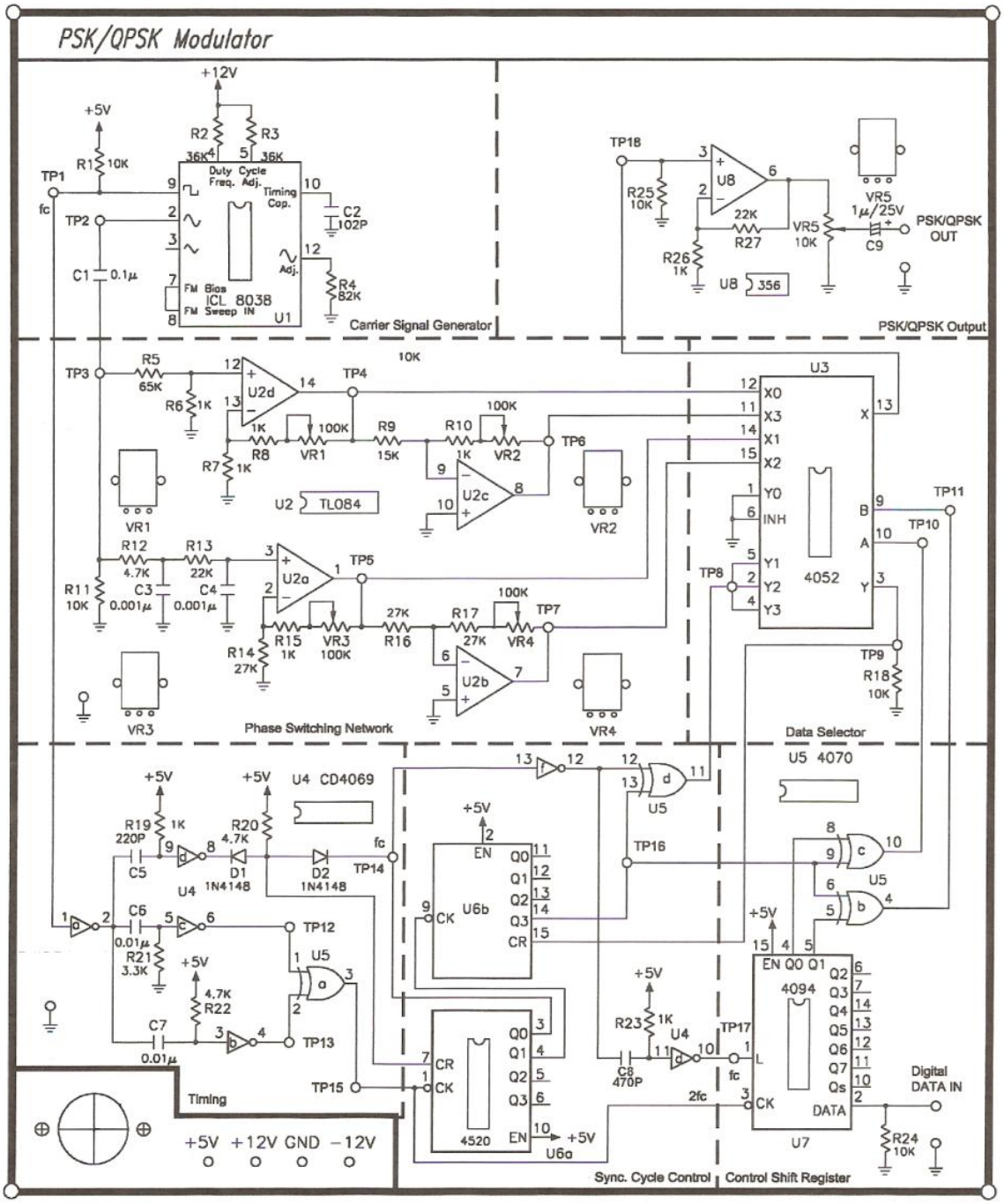


Figure 19-9 KL-94006 module

2. PSK/QPSK demodulator

Figure 19-10 shows the schematic diagram of PSK/QPSK demodulator. The amplifier U1d receives and amplifies the PSK/QPSK modulated signal for compensating the losses and improving the distortion caused by the transmission line. The full-wave rectifier, constructed by op amps U1c-U1b and diodes D1-D4, shapes the received PSK/QPSK signal to a positive-cycle signal, which is connected to the input of the phase detector in PLL (U2). VCO out signal (32fc) is the clock pulse of the divide-by-N counter. The counter produces two frequencies 4fc and 2fc on the outputs Q2 and Q3, respectively.

The amplified PSK/QPSK signal on U1d output terminal is connected to the input of amplifier U1a. U1a converts the PSK/QPSK signal to digital pulse signal as shown in Figure 19-7. This digital signal on TP5 is buffered by inverters U3e and U3f and then connected to the inputs J and K* of U4.

Sync cycle detector contains 4-stage register (U4) and 4-input NAND gate (U5b). The clock frequency of the register is 2fc. The reset pulse generated by the network (R21, C9, U3d) is used to clear the register outputs Q0-Q3. The digital data on TP5 is connected to the inputs J and K*. When a sync cycle is received, the register outputs Q3-Q0=0111, the output (TP13) of 4-input NAND gate U5b presents a low to indicate that a sync cycle is detected. During TP13 is low, the output of U8c NAND gate is high, therefore the demodulator output is inhibited. For other output sets of Q0-Q3, TP13 presents a high.

The digital data on TP5 is also sent to the DATA input of the shift register U7. The clock frequency of U7 and U6b is 4fc, while bit rate of digital input data is equal to fc or 2fc. The frequencies of the counter (U6b) outputs Q1 and Q2 are fc and fc/2, respectively. The Q1 output is connected to the load input of the register U9 and to the RX CLK OUT terminal. The counter is reset either a sync cycle detected (TP13=0, CR=1) or Q2 output is high (Q2=1, CR=1).

The output of the demodulated data on U7 outputs Q2-Q3 is controlled by the control logic (NAND gates U8a, b, c, d). If no sync cycle is detected (TP13=1), the demodulated data can be sent to DATA OUT terminal. If a sync cycle is detected (TP13=0), the data path is blocked by the control logic.

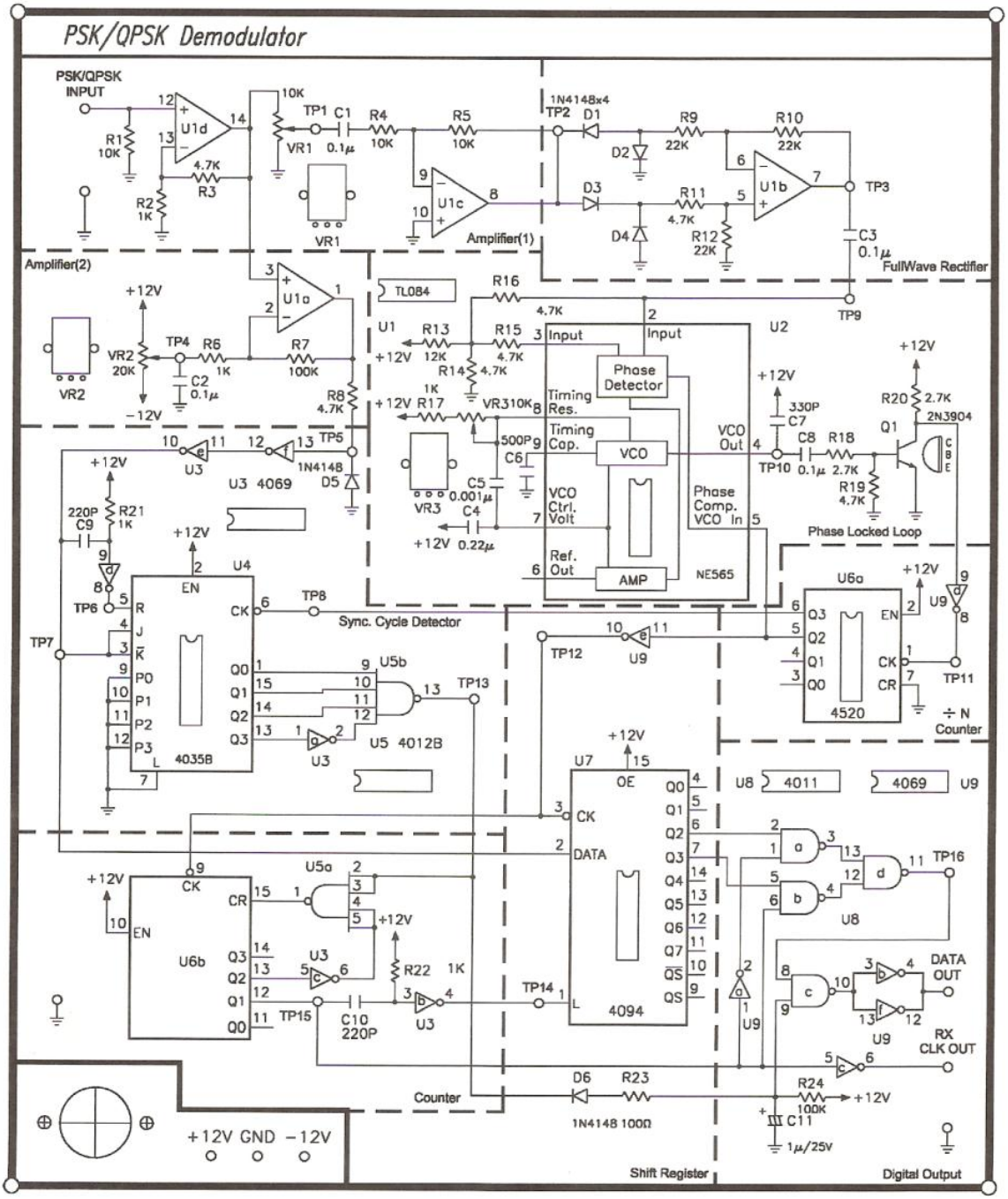


Figure 19-10 KL-94007 module

19.3 EQUIPMENT REQUIRED

1 - Module KL-92001
2 - Module KL-94006,KL-94007
3 - Oscilloscope

19.4 EXPERIMENTS AND RECORDS

Experiment 19-1 Measurement and Adjustment

A. KL-94006 Measurement and Adjustment

1.Apply the required power supply voltages +12V, -12V, and +5V to PSK/QPSK Modulator module KL-94006 shown in Figure 19-9.

2.Using the oscilloscope, measure and record the waveforms and frequencies on test points TP1, TP2, and TP3 in Table 19-2.

3.Connect scope CH1 IN to TP3 and CH2 IN to TP4. Measure and record the waveforms and frequencies in Table 19-3. Set the amplitude of the signal on TP4 to 1Vpp by adjusting the VR1 and note the phase difference between these two waveforms.

4.Connect scope CH1 IN to TP3 and CH2 IN to TP6. Measure and record the waveforms and frequencies in Table 19-3. Set the amplitude of the signal on TP6 to 2Vpp by adjusting the VR2 and note the phase difference between these two waveforms.

5.Connect scope CH1 IN to TP3 and CH2 IN to TP5. Measure and record the waveforms and frequencies in Table 19-3. Set the amplitude of the signal on TP5 to 3Vpp by adjusting the VR3 and note the phase difference between these two waveforms.

6.Connect scope CH1 IN to TP3 and CH2 IN to TP7. Measure and record the waveforms and frequencies in Table 19-3. Set the amplitude of the signal on TP7 to 3Vpp by adjusting the VR4 and note the phase difference between these two waveforms.

7. Connect a 500Hz, TTL-level square wave to the Digital DATA IN terminal.

8. Connect scope CH1 IN to the PSK/QPSK OUT. Measure the waveform and set the output amplitude to 10Vpp by adjusting the VR5 and record the result in Table 19-4. (Note: The frequency can not be measured in this case.)

9. Turn off the power.

10. Connect the PSK/QPSK OUT of module KL-94006 to the PSK/QPSK INPUT of module KL-94007.

11. Connect the power supply voltages required to both KL-94006 and KL-94007 modules.

B. KL-94007 Measurement and Adjustment

12. Connect scope CH1 IN to TP4. Measure and set the dc voltage to -5Vdc exactly by adjusting the VR2. To do this well, an extra DVM is a good choice.

13. Connect scope CH1 IN to TP1. Measure the waveform and frequency and set the amplitude to 5Vpp by adjusting the VR1. Record the result in Table 19-5.

14. Connect scope CH1 IN to TP11. Measure the waveform and frequency and set the frequency to 32fc by adjusting the VR3. Record the result in Table 19-6. If the carrier frequency is 8KHz, the signal frequency on TP11 should be 256KHz.

15. Connect scope CH1 IN to DATA OUT. Measure and record the waveform and frequency in Table 19-7. The waveform on DATA OUT should be the demodulated digital signal, 500Hz. If not, you can slightly turn the VR1 or turn off and then on the power.

16. Connect scope CH1 IN to RX CLK OUT terminal. Measure and record the waveform and frequency in Table 19-7. The waveform on RX CLK OUT should be recovered carrier signal. If not, you can slightly turn the VR1 or turn off and then on the power.

17. Turn off the power.

Experiment 19-2 PSK/QPSK Modulator

A. 2fc Measurement (KL-94006)

1. Connect a 500Hz, TTL-level digital signal to Digital DATA IN.

2. Connect scope CH1 IN to TP12 and CH2 IN to TP13. Measure and record the waveforms and frequencies in Table 19-8. Compare the phase difference between these two waveforms.

3. Connect scope CH1 IN to TP15. Measure and record the waveform and frequency in Table 19-8. The frequency should be twice the carrier frequency, 2fc.

B. Sync Cycle Measurement

4. Measure and record the waveforms and frequencies on the test points listed in Table 19-9.

C. Control Shift Register Measurement

5. Connect scope CH1 IN to TP10 and CH2 IN to TP11. Measure and record the waveforms and frequencies in Table 19-10.

6. Repeat step 5 for digital signal frequencies 100Hz and 1KHz to Digital DATA IN.

7. Recover the digital signal frequency to 500Hz.

D. PSK/QPSK modulated Signal Measurement

8. Connect scope CH1 IN to PSK/QPSK OUT. Measure and record the waveform with the various TIME/DIV settings in Table 19-11.

BIBLIOGRAFÍA

1. **BROWN., S Y OTROS.,** Fundamentos de electrónica digital con diseño VHDL., 2a.ed., Santa Fe – México., Mc Graw Hill Interamericana., 2006., Pp. 4-5-6-7-8-9-55-56-57-94-95-96-97-98-99-107-108-109.

2. **TOMASI., W.,** Sistemas de comunicaciones eléctricas., 4a.ed., Phoenix – Arizona., Pearson Educación., 2003., Pp. 467-468-469-471-478-482-496-498.

3. **RODRÍGUEZ., J Y OTROS.,** Modulación de señales digitales., 1a.ed., Sevilla – España., Servicio de Publicaciones de la Universidad de Sevilla., 1995., Pp. 17-18-30-31-45-46-47-51-52-53-56-57-58-59-60-61-62-63-64-65-66-67-68.
   E-book.
   http://personal.us.es/jluque/Libros%20y%20apuntes/1995%20Modulacion%20digital.pdf

4. **KUON I., Y OTROS.,** FPGA Architecture Survey and Challenges., 2a.ed., Massachusetts – USA., NOW the essence of knowledge., 2008., Pp. 143-144-146-147.
   E-book.
   http://www.eecg.toronto.edu/~jayar/pubs/kuon/foundtrend08.pdf

5. **ROJO., R.,** Modulación ASK, FSK & PSK., 1a.ed., Ciudad de México – México., Instituto Politécnico Nacional (IPN)., Pp. 3-4-6.
   E-book.

http://es.scribd.com/doc/84778536/Modulacion-ASK-FSK-PSK

6. **SISTENA., C.,** Field Programmable Gate Arrays (FPGAs)., 1a.ed., San Juan – Argentina., Sistemas Digitales II – Universidad Nacional de San Juan., Pp. 4-5. E-book. http://dea.unsj.edu.ar/sisdig2/Field%20Programmable%20Gate%20Array_A.pdf

7. **ALVERCA., Y.,** Síntesis De Circuitos Digitales Utilizando VHDL (VHSIC hardware description language) y FPGA (field programmable gate arrays)., Ingeniería Eléctrica y Electrónica., Ingeniería en Electrónica y Telecomunicaciones., Escuela Politécnica Nacional., Quito – Ecuador., TESIS., 2008., Pp. 12-13-14 E-book. http://bibdigital.epn.edu.ec/bitstream/15000/764/1/CD-1250.pdf

8. **LUMERTZ., F.,** Implementação de um Codificador LDPC para um Sistema de TV Digital usando Ferramentas de Prototipagem Rápida., Ingeniería Eléctrica y Computación., Ingeniería en Telecomunicaciones y Telemática., Universidad Estatal de Campinas., Campinas – Brasil., TESIS., 2006., Pp 11-12. E-boook. http://www.bibliotecadigital.unicamp.br/document/?code=vtls000406415

9. **PÉREZ., J Y OTROS.,** Entorno Educativo Para El Aprendizaje De Sistemas Digitales., Ingeniero técnico de telecomunicación., Ingeniería en Sonido e Imagen., Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicaciones., Pamplona – España., TESIS.,

2010., Pp. 17-20.
E-book.
http://www.google.com.ec/url?sa=t&rct=j&q=Entorno+Educativo+Para+El+AprendizajeDe+Sistemas+Digitales.,+Ingeniero+t%C3%A9cnico+detelecomunicaci%C3%B3n.,+Ingenier%C3%ADa+en+Sonido+e+Imagen&source=web&cd=1&ved=0CCkQFjAA&url=http://academica-e.unavarra.es/bitstream/handle/2454/1794/577023.pdf%3Fsequence%3D1&ei=iBUrU6ivL_Pp0QH8zYDgAw&usg=AFQjCNH3KtnPX_wl8e96vr4y1KcLtqJV2g

10. **ALTERA**
http://www.altera.com/corporate/about_us/abt-index.html
http://www.altera.com/corporate/about_us/history/abt-history.html
2013 -07 -05

11. **ALTERA DE0 BOARD**
http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=165&No=364&PartNo=1
2013 - 07 - 11.

12. **ALTERA DE1 BOARD**
http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=53&No=83#1732
2013 - 07 - 12.

13. **ALTERA DE2 BOARD**
http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=39&No=30
2013 - 07 - 13.

14. **ALTERA DE2-115 DEVELOPMENT AND EDUCATIONAL BOARD**

http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=502

2013 - 07 - 14.

15. **ANVYL SPARTAN-6 FPGA DEVELOPMENT BOARD**

http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,1175&Prod=ANVYL

2013 - 07 - 11.

16. **APLICACIONES GENERALES**

http://www.generatecnologias.es/aplicaciones_fpga.html

2013 - 06 - 23.

17. **ATLYS SPARTAN-6 FPGA DEVELOPMENT KIT**

http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,1028&Prod=ZEDBOARD

2013 - 07 - 10.

18. **BASYS 2 SPARTAN-3E FPGA BOARD**

http://www.digilentinc.com/Products/Detail.cfm?Prod=BASYS2

2013 -07 - 08.

19. **CONCEPTOS FUNDAMENTALES**

http://www.generatecnologias.es/ingenieria_fpga.html

http://www.marisolcollazos.es/noticias-informatica/?tag=fpga

2013 - 06 - 17.

20. **DE0-NANO DEVELOPMENT BOARD**

http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=165&No=593&PartNo=1

2013 - 07 - 11.

21. **HERRAMIENTAS PARA LA PROGRAMACIÓN DE FPGAS.**

http://www.ni.com/white-paper/6983/es/

2013 - 06 - 20.

22. **INTRODUCCION**

http://fpga.com.ar/4.html

2013 - 06 - 02.

23. **ORÍGENES DE LA TECNOLOGÍA**

http://mcielectronics.wordpress.com/2010/03/14/un-poco-de-historia-dispositivos-logicos-programables-plds/

2013 - 06 - 18.

24. **SEÑAL SINUSOIDAL**

http://www.ciudadoscura.com/electronica/1/generador-de-senal-sinusoidal-en-vhdl.html

2013 - 09 - 20.

25. **SOFTWARE ISE**

http://webs.uvigo.es/mdgomez/SED/Guia_Inicio_ISE.pdf

2013 - 07 - 07.

26. **SOFTWARE QUARTUS II**

http://www.element14.com/community/docs/DOC-40098/l/altera-introduction-to-the-quartus-ii-software

2013 - 07 - 05.

27. **SPARTAN-3A DSP 1800A EDITION**

http://www.xilinx.com/products/boards-and-kits/HW-SD1800A-DSP-SB-UNI-G.htm

2013 - 07 - 09.

28. **SPARTAN-3A STARTER KIT**

http://www.xilinx.com/products/boards-and-kits/HW-SPAR3A-SK-UNI-G.htm

2013 - 07 - 09.

29. **SPARTAN-3AN STARTER KIT**

http://www.xilinx.com/products/boards-and-kits/HW-SPAR3AN-SK-UNI-G.htm

2013 - 07 - 09.

30. **SPARTAN-3E STARTER KIT**

http://www.xilinx.com/products/boards-and-kits/HW-SPAR3E-SK-US-G.htm

2013 - 07 - 10.

31. **XILINX**

http://centrodeartigos.com/articulos-informativos/article_69419.html

http://www.xilinx.com/about/company-overview/index.htm

2013 - 07 - 07.