

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

**“ESTUDIO DE UN RADIO ENLACE ENTRE LOS DISPENSARIOS DE
PULINGÚ Y NITILUISA DEL SEGURO SOCIAL CAMPESINO EN LA
PROVINCIA DEL CHIMBORAZO Y EL DISEÑO DE UNA INTERFAZ DE
DATOS”**

TESIS DE GRADO

Previa a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Neiser Stalin Ortiz Mosquera

Ximena Fabiola Trujillo Borja

Guayaquil - Ecuador

2008

AGRADECIMIENTO

A la Escuela Superior Politécnica del Litoral, génesis del conocimiento, noble institución universitaria al servicio de Guayaquil y de la Patria.

A las autoridades, personal docente y administrativo por su mística de trabajo y responsabilidad.

Al Sr. Ing. César Martín Moreno por su valiosa e inteligente asesoría en la realización de esta tesis.

DEDICATORIA

A nuestros padres, pilar fundamental en nuestra existencia por su infinito amor,
apoyo incondicional y por ser faros permanentes de sabiduría y bondad guiando
nuestras vidas

Los Autores


DECLARACIÓN EXPRESA

“La responsabilidad del contenido de esta Tesis de Grado, nos corresponden exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL”

(Reglamento de Graduación de la ESPOL)



Neiser Ortiz Mosquera



Ximena Trujillo Borja

TRIBUNAL DE GRADO



Ing. Holger Cevallos

PRESIDENTE



MSc. César Martín Moreno

DIRECTOR DE TESIS



MSc. Juan Carlos Aviles

VOCAL PRINCIPAL



MSc. María Antonienta Alvarez

VOCAL PRINCIPAL

ÍNDICE GENERAL

ÍNDICE GENERAL	VII
ÍNDICE DE FIGURAS	XI
ÍNDICE DE TABLAS	XIV
ABREVIATURAS	XV
INTRODUCCIÓN	1

CAPÍTULO 1

1. ANÁLISIS DE LA SITUACIÓN ACTUAL DE LAS COMUNICACIONES EN LAS POBLACIONES DE NITILUISA Y PULINGUÍ

1.1 Situación Geográfica	3
1.2 Situación Económica	7
1.3 Recursos Humanos	8
1.4 Recursos Tecnológicos	11

CAPÍTULO 2

2. ANÁLISIS DE LOS DISPOSITIVOS DISPONIBLES EN EL MERCADO ECUATORIANO PARA EL DISEÑO DEL RADIO ENLACE

2.1 Dispositivos para el radio enlace	12
2.1.1 Antenas	12
2.1.2 Cables	14

2.1.3 Conectores	15
2.1.4 Radio digital	17

CAPÍTULO 3

3. DISEÑO

3.1 Diseño del radio software	21
3.1.1 Diagrama del radio software	23
3.1.2. Diagrama del modulador	24
3. 1.3. Conversión Digital/Analógica	24
3.1.4. Modulación QPSK	26
3.1.5. Oscilador	27
3.1.5.1. Método de la acumulación de fase	28
3.1.5.2. Frecuencia de salida	30
3.1.5.3. Control de fase	30
3.1.5.4. Pureza espectral	31
3.1.5.5. NCO de cuadratura	32
3.1.5.6. Compresión de la LUT	33
3.1.6. Generación de la tabla senoide	34
3.6.1. Conversión complemento 2 a unipolar sin signo	35
3.1.7. Diagrama del demodulador	35
3.1.8. Conversión Analógica/Digital	36
3.1.9. Demodulación QPSK	38

3.1.10. Desplazamiento en frecuencia y Frecuencia Intermedia	38
3.1.11. Mezclador Complejo	39
3.1.12. Filtrado	40
3.1.12.1. Filtro CIC	42
3.1.12.2. Filtro FIR	45
3.1.13. Decisor QPSK	48
3.2 Diseño del Enlace	49
3.2.1 Mapa del terreno	49
3.2.2 Cálculo del margen de ganancia	51
3.2.3 Indicaciones sobre implementación	57
3.2.4 Tabla de costos	60

CAPÍTULO 4

4. IMPLEMENTACIÓN PRÁCTICA

4.1. Radio software	62
4.1.1 Hardware	63
4.1.1.2. Especificación de los dispositivos utilizados	63
4.1.2. Software	64
4.1.3. Algoritmos	65
4.1.3.1 Acumulador	65
4.1.3 .2. NCO	67
4.1.3 .3. Modulador QPSK	72

4.1.3.4. Mezclador Complejo	73
4.1.3 .5. Filtro CIC	77
4.1.3 .6. Filtro FIR	81
4.1.3.7. Decisor QPSK	84
4.1.3.8. Demodulador QPSK	86

CAPÍTULO 5

5. PRUEBAS DE LA IMPLEMENTACIÓN

5.1. Radio software	90
5.1.1. Prueba de la implementación de la modulación	91
5.1.2. Resultados obtenidos de la modulación	91
5.1.3. Prueba de la implementación de la demodulación	92
5.1.4. Resultados obtenidos de la demodulación	92

CONCLUSIONES Y RECOMENDACIONES

BIBLIOGRAFÍA

ANEXOS

ÍNDICE DE FIGURAS

1.1 Comunidad Nitiluisa	3
1.2 Dispensario Nitiluisa	5
1.3 Comunidad Pulinguí	5
1.4 Dispensario Pulinguí	7
2.1 Esquema de radio digital	17
2.2 Radio modem conexión serie RS232	19
2.3 Radio modem usb 19.2Kb easy radio 433Mhz	20
3.1 Diagrama de un radio software	23
3.2 Diagrama de un modulador QPSK	24
3.3 Diagrama de conexión	25
3.4 Relación de la fase de salida contra tiempo para un modulador QPSK	26
3.5 Diagrama fasorial del QPSK	27
3.6 Oscilador de acumulación de fase	28
3.7 Diagrama de bloques de un NCO con LUT	29
3.8 Diagrama de bloques del NCO con controlador de fase	31
3.9 La senoide dividida en cuadrantes	33
3.9 Diagrama de un demodulador QPSK	36
3.10 Diagrama de conexión	37
3.11 Espectro de la señal después de pasar por Desplazamiento en frecuencia y Frecuencia Intermedia	39
3.12 Diagrama de bloques del mezclador complejo	40

3.13 Espectro de la señal después de pasar por el filtro CIC y el filtro FIR	41
3.14 Estructura básica de un filtro CIC diezrador	43
3.15 Esquema de bloque del filtro FIR	46
3.16 Diagrama de bloque de un filtro FIR que aprovecha la simetría de los coeficientes	47
3.17 Constelación QPSK	48
3.18 Mapa general	49
3.19 Mapa específico	50
3.20 Zona de Fresnel	55
3.21 Para el Montaje	59
4.1 Tarjeta FPGA EPF10K10LC84-4	63
4.2 Diagrama de bloques del acumulador	66
4.3 Bloque del acumulador en el editor gráfico	67
4.4 Diagrama de bloques del NCO	68
4.5 Bloque del NCO en el editor gráfico	71
4.6 Bloque del modulador QPSK en el editor gráfico	73
4.7 Diagrama en bloque de un mezclador	75
4.8 Bloque del mezclador del canal I en el editor gráfico	77
4.9 Bloque del mezclador del canal Q en el editor gráfico	77
4.10 Diagrama en bloque del filtro CIC	78
4.11 Espectro de salida del filtro CIC para $N=4$, $R=7$ y $M=1$	79
4.12 Bloque del filtro CIC en el editor gráfico	80

4.13 Diagrama en bloque del filtro FIR	82
4.14 Bloque del filtro FIR en el editor gráfico	84
4.15 Bloque del decisor QPSK en el editor gráfico	85
4.16 Diagrama en bloque del demodulador QPSK del canal I	86
4.17 Bloque del demodulador QPSK del canal I en el editor gráfico	88
4.18 Diagrama en bloque del demodulador QPSK del canal Q	88
4.19 Bloque del demodulador QPSK del canal Q en el editor gráfico	89
5.1 Modulador QPSK implementado	91
5.2 Simulación del modulador implementado QPSK	91
5.3 Demodulador QPSK implementado	92
5.4 Simulación del demodulador QPSK implementado	93
5.5 Simulación del sistema radio software	94
5.6 Hyper Terminal	94
5.7 Conexión del MAX 232	95
5.8 Sistema de comunicación completo	96
5.9 Esquema eléctrico de la implementación	96
5.10 Circuito impreso	97

ÍNDICE DE TABLAS

1.1 Muestra mensual de pacientes atendidos Nitiluisa	9
1.2 Muestra mensual de pacientes atendidos Pulinguí	10
2.1 Comparación entre antenas	13
2.2 Especificaciones de cables	15
2.3 Comparación de conectores	16
3.1 Tabla de verdad del Modulador QPSK	26
3.2 Ganancia de antenas	54
3.3 Pérdidas	54
3.4 Cálculo del margen de ganancia radio 1	54
3.5 Cálculo del margen de ganancia radio 2	54
3.6 Tabla de costos	61
4.1 Variación de la onda portadora	70
4.2 Variación de la fase de la portadora del modulador QPSK por datain	73

ABREVIATURAS

ADC	Convertidor Analógico – Digital
ASIC	Circuitos Integrados los de Aplicación Específica
BB	Banda Base
CIC	Cascaded integrator-comb o cascada integrador-peine
CPLD	Complex Programmable Logic Device
DAC	Convertidor Digital – Analógico
DSP	Digital Signal Processor
EIA	Asociación de las Industrias Electrónicas
f_c	carrier frequency
f_{clk}	frecuencia de reloj
FI	Frecuencia Intermedia
FIR	Finite Impulse Response o Respuesta finita al impulso
FPGA	Field Programmable Gate Array
f_s	frecuencia de muestreo
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
LPM	Bibliotecas de Módulos Parametrizados
LUT	Método de la Tabla
mW	milivatios
NCO	Oscilador Controlado Numéricamente
PLD	Dispositivos Lógicos Programables
PVC	Policloruro de Vinilo

QPSK	Quadrature Phase Shift Keying
RF	Radio Frecuencia
RSL	Mínimo Nivel de Señal Recibida
SFDR	Rango Dinámico Libre de Espurios
SNR	Relación Señal a Ruido
USART	Universal Synchronous Asynchronous Receiver Transmitter
UTP	Cable de Par Trenzado no Blindado
VCO	Oscilador Controlado por Tensión
VHDL	Very high speed integrated circuit Hardware Description Language

RESUMEN

Los dispensarios de Nitiluisa y Pulinguí ubicados en la zona rural de la provincia de Chimborazo al norte de la ciudad de Riobamba, tienen a su disposición un personal operativo formado por una auxiliar de enfermería y un médico tratante el cual asiste dos días por semana a cada uno de los dispensarios y por tanto se hace necesario en el resto de días la comunicación con el mismo, aún más cuando se necesita atención médica, es por esta razón que el presente estudio pretende dar una solución parcial con nuevas tendencias tecnológicas a dicho problema.

Se ha dividido en dos partes principales el estudio, en primera instancia se encuentra el diseño e implementación de una interfaz de datos (radio digital) y como segunda instancia se encuentra el estudio del radio enlace Nitiluisa – Pulinguí.

El sistema de radio digital o también llamado radio software se basa en los dispositivos lógicos programables. Específicamente se utilizaron FPGA's, que se emplean cada vez más en las telecomunicaciones por su versatilidad.

El análisis y diseño del radio enlace Nitiluisa-Pulinguí muestra la factibilidad técnica y económica, además los beneficios que las poblaciones y personal operativo de los dispensarios obtendrían si fuera implementado.

INTRODUCCIÓN

El trabajo se lo realizó pensando en la utilización de nuevas tecnologías de transmisión y recepción de datos actualmente disponibles, donde las mismas pueden ser implementadas a través de software y no solo de hardware, siendo parte esencial de una solución en el problema expuesto por los dispensarios de Nitiluisa y Pulinguí, en este trabajo se dice que son una parte de la solución puesto que no se ha podido poner en práctica el radio enlace debido a las limitaciones de los dispositivos existentes en el Laboratorio de Sistemas Digitales (recursos de las FPGAs) y los costos de los demás dispositivos para completar el enlace bidireccional.

El radio-digital está descrito en forma de software, también llamado radio software, utilizando algoritmos esenciales para la modulación y demodulación en código VHDL, las mismas que pueden ser: BPSK, QPSK, FSK, QAM y otras analógicas como FM y AM, en el presente caso se ha optado por la modulación y demodulación QPSK [4] por ser un sistema robusto. Las herramientas utilizadas para el diseño e implementación son: como hardware los Dispositivos Lógicos Programables (tarjeta FPGA EPF10K10LC84-4) y como software Altera Max Plus II los cuales están disponibles en el Laboratorio de Sistemas Digitales.

Previo al estudio del radio enlace y el diseño del radio software se revisó la situación geográfica, la situación económica, los recursos humanos y los recursos tecnológicos

con los que cuentan los dispensarios y las poblaciones de Nitiluisa y Pulinguí. Este análisis permite explicar la situación actual de estas dos poblaciones rurales y mostrar la factibilidad del radio enlace a realizarse.

Se utilizó un estudio topográfico de la zona la cual ha sido monitoreada por varias semanas para saber sus características. En cuanto a las herramientas de apoyo que se utilizaron se encuentra simuladores virtuales que permite simular el enlace de punto a punto que necesitamos, permitiendo calcular los factores importantes como la primera zona de fresnel y el margen de ganancia del enlace.

CAPÍTULO 1

1. ANÁLISIS DE LA SITUACIÓN ACTUAL DE LAS COMUNICACIONES EN LAS POBLACIONES DE NITILUISA Y PULINGUÍ

1.1 Situación Geográfica

Los dispensarios de Nitiluisa y Pulinguí se encuentran en la zona rural de la provincia de Chimborazo, al norte de la ciudad de Riobamba la cual se toma como referencia.

Nitiluisa



Figura 1.1 Comunidad Nitiluisa

La población de Nitiluisa pertenece al cantón Riobamba y se encuentra a 30Km de la ciudad de Riobamba tomando la vía Panamericana Sur como vía principal para más tarde ingresar en un camino secundario y llegar al dispensario localizado junto

al colegio “Técnico Experimental Autachi”. Esta población cuenta con transporte público proporcionado por la cooperativa “Unidos”, los cuales salen de Riobamba en dos turnos por la mañana 6:30 y 7:30, e ingresan en la población recorriendo poblaciones aledañas y regresando inmediatamente.

El dispensario se encuentra a una altura de 3228mts, localizado en un valle rodeado por el oeste de una zona montañosa.

Coordenadas Geográficas:

- Latitud: 1° 35' 27''
- Longitud: 78° 45' 35''

El clima es frío entre 15° y 8° celcius, despejado por la mañana y nublado por la tarde, poca neblina y vientos fuertes por las tardes.



Figura 1.2 Dispensario Nitiluisa

Pulinguí



Figura 1.3 Comunidad Pulinguí

La población de Pulinguí pertenece al cantón San Andrés y se encuentra a 35Km de

la ciudad de Riobamba tomando la vía Panamericana Norte como vía principal para luego ingresar en un camino secundario empedrado para llegar al dispensario localizado en la plaza principal de Pulinguí junto al Colegio Nacional “Once de Noviembre” . Pulinguí cuenta con un sistema de transporte proporcionado por la cooperativa “Cóndor” quienes cuentan con turnos cada cuatro horas empezando a las 6:00am hasta las 6:00pm en la ruta Riobamba - Pulinguí y Pulinguí - Riobamba.

Este dispensario se encuentra a 3344mts, en una zona montañosa al oeste del dispensario de Nitiluisa.

Coordenadas Geográficas:

- Latitud: $1^{\circ} 33' 55''$
- Longitud: $78^{\circ} 45' 05''$

Su clima es frío al igual que en la población de Nitiluisa, despejado por las mañanas y nublado por las tardes, las precipitaciones ocurren generalmente por las noches.



Figura 1.4 Dispensario Pulinguí

Los datos aquí expuestos fueron tomados con dispositivos GPS Magallan.

1.2 Situación Económica

Los pobladores de Nitiluisa y Pulinguí se dedican en su mayoría a la agricultura, los mismos que son propietarios de pequeñas parcelas y en promedio de unos 10 o 15 animales entre ganado ovino, vacuno y porcino que son el sustento de sus hogares, por lo que estas poblaciones no cuentan con mayores recursos económicos. En promedio los agricultores logran un ingreso mensual de alrededor de \$100 mientras que los albañiles de la zona ganan entre \$250 y \$300.

Poseen guardería, escuela y colegio proporcionados por el estado ecuatoriano. En el caso de la población de Pulinguí cuentan con un centro de enseñanza de tareas

dirigidas proporcionado por la Organización Evangélica “Verbo”.

En el Dispensario de Nitiluisa cada jefe de familia debe realizar el pago de \$1,13 mensualmente por concepto del Seguro Social Campesino y de \$ 1,44 anualmente por concepto de luz, agua del dispensario y transporte de la directiva. Las medicinas, materiales de oficina, limpieza son proporcionados por el Seguro Social Campesino.

En el Dispensario de Pulinguí cada jefe de familia debe realizar el aporte de \$13,56 anualmente por concepto del Seguro Social Campesino y de \$1,44 por gastos internos del dispensario y al igual que en el dispensario de Nitiluisa la institución asume el resto de gastos.

1.3 Recursos Humanos

El personal operativo de cada dispensario consta de un médico tratante que acude a la consulta dos o tres días por semana de 8:00am a 2:00pm, una auxiliar de enfermería que permanece en el dispensario de lunes a viernes de 8:00am a 4:00pm y un odontólogo que visita el dispensario uno o dos meses por año en el mismo horario que el médico tratante.

El Dispensario de Nitiluisa atiende a 375 jefes de familia y se encuentra conformado

por 3 organizaciones:

- Nitiluisa.- Con 200 jefes de familia, consta de Nitiluisa Central, Corona Real, Rumipamba, La Moya y Rumicruz.
- San Vicente.- Con 40 jefes de familia, consta de La Turun, Gaushi Chico , que se encuentran aproximadamente a 3Km de distancia del dispensario
- Cunduana.- Con 135 jefes de familia, divide en barrios como: La Central, Carmelitas, Merced, que se encuentran aproximadamente a 2Km de distancia del dispensario.

Entre el 90% y 95% de la población total de esta comunidad se encuentra afiliado al Seguro Social Campesino por lo que la población total se acerca a 2200 personas.

En la Tabla 1.1 existe una muestra mensual de la atención médica:

Mes	Diciembre	Enero	Febrero
Afiliados del SSC	59	45	88
No afiliados	13	13	35
Transferencias	3	6	6

Tabla 1.1 Muestra mensual de pacientes atendidos Nitiluisa

El personal operativo permanente de este dispensario está conformado por: Dr.

Héctor Lazo como médico tratante quien da consulta médica los días martes y jueves, la auxiliar de enfermería Sra. Carmita Benítez.

El Dispensario de Pulinguí atiende a 311 jefes de familia con un total de 1600 afiliados y una población entre afiliados y no afiliados de aproximadamente 2000 habitantes, consta de 2 organizaciones:

- Pulinguí.- Samjapama, Tohuolog, Tunsalao y Cuatro Esquinas
- Rumicruz

En la Tabla 1.2 existe una muestra mensual de la atención médica:

Mes	Diciembre	Enero	Febrero
Afiliados del SSC	94	100	78
No afiliados	14	21	22
Transferencias	4	5	6

Tabla 1.2 Muestra mensual de pacientes atendidos Pulinguí

El personal operativo permanente de este dispensario está conformado por: Dra. María Fabiola Borja como médico tratante quien da consulta médica los días lunes, miércoles y viernes, la auxiliar de enfermería Lcda. María Elena Cazco.

1.4 Recursos Tecnológicos

En cuanto a los recursos tecnológicos con los que cuentan estas poblaciones cabe resaltar que en cada población existe una línea telefónica proporcionada por la compañía Andinatel, aún no se dispone del servicio de telefonía móvil e internet

También cuentan con el servicio de luz eléctrica y agua proveniente del canal de riego sin potabilizar.

Los dispensarios poseen una computadora cada uno, así como un equipo de sonido y en el caso del dispensario de Pulinguí cuentan con un televisor y un refrigerador utilizado para mantener las vacunas.

Todos los datos aquí expuestos se encuentran en los registros de cada uno de los dispensarios, actualizados hasta abril del 2008, el personal operativo de cada uno de los dispensarios se encuentra en la sección del anexo A.

CAPÍTULO 2

2. ANÁLISIS DE LOS DISPOSITIVOS DISPONIBLES EN EL MERCADO ECUATORIANO PARA EL DISEÑO DEL RADIO ENLACE

2.1 DISPOSITIVOS PARA EL ENLACE

El enlace a realizarse es punto a punto, los dispositivos deben ser resistentes al viento y la lluvia. En la frecuencia de 2.4GHz [31] por ser frecuencia libre de licencia en el Ecuador.

2.1.1 ANTENA

Una vez que se sabe que el enlace debe estar en la frecuencia de los 2.4Ghz, se procede a la elección de las antenas. En este caso es necesario aclarar que el enlace es punto a punto por lo que la antena elegida debe ser adecuada para este tipo de transmisión con el fin de no desperdiciar recursos, al mismo tiempo debe ser resistente (no corrosivo), con una ganancia alta que permita la mejor propagación de la señal.

La antena escogida puede ser direccional o sectorial, en el presente enlace se ha escogido una direccional de la marca Hiperlink [26] con una ganancia de 24dBi que presenta las características antes descritas. Además la antena escogida representa un menor costo lo cual es un factor importante para la elección.

Los precios que se hallan en la Tabla 2.1 se los han obtenido de la siguiente

referencia [27].

Se presenta una tabla comparativa entre varias antenas disponibles en el mercado.




	ANTENA YAGUI 	ANTENA SECTORIAL 	ANTENA DIRECCIONAL EXTERNA 
Marca	Teletronics [18]	Teletronics[19]	Hiperlink [20]
Frecuencia	2.4 – 2.5GHz	2.3 – 2.5GHz	2.4 – 2.5GHz
Ganancia	15 dBi	15 dBi	24 dBi
Máxima potencia de entrada	100W	100W	50W
Polarización	Vert/Hori	Vertical	26dBi cruzada
Ancho de haz	28°	14.5°/60-160°	8°
VSWR 50Ohm	<1.5	1.5:1 máx	1.5:1 máx
Velocidad del viento	210Km/h	200Km/h	
Peso	3lbs	3lbs	4.8lbs
Conector	N-tipo hembra	N-tipo hembra	N-tipo hembra
Temperatura de operación	-40 °C a +60 °C	-40 °C a +60 °C	-40 °C a + 85 °C
Material	PVC	Aluminio	Aluminio
Características adicionales	UV estabilizado	UV estabilizado	UV estabilizado
Precio	\$150	\$336	\$90

Tabla 2.1 Comparación entre antenas

2.1.2 CABLES

Existe una amplia gama de cables disponibles para realizar las conexiones en el enlace, dependiendo de las características del radio, antena, distancias de los cables, pérdidas de los mismos. Para este caso en especial se ha escogido el cable LMR 400 por ser robusto, resistente al viento, su costo y su baja pérdida que pueden ser calculados en la siguiente referencia [24], la distancia aproximada de cable a utilizarse es de 20m. En esta parte del capítulo se presentan algunos de los cables disponibles.

Los cables TIMES LMR [20] de banda ancha de alto rendimiento, flexibles, de baja pérdida. Cables coaxiales de comunicación diseñados para su uso en aplicaciones inalámbricas tales como:

- Red local wireless
- Wireless internet
- IEEE 802.11a y 802.11b

Por otro lado tenemos los TCOM [20] cables que ofrecen lo último en rendimiento en un cable flexible. Para ser utilizados en UHF, en aplicaciones de microondas y todos los demás sistemas comerciales y militares de interconexión.

Los cálculos de la pérdida en la señal se han realizado a una frecuencia de 2.4GHz como se muestra en Tabla 2.2.

CABLE	PÉRDIDA EN (dB/100m)
LMR 200	54.2
LMR 240	41.5
LMR 400	21.7
LMR 600	14.2
TCOM-200	54.17
TCOM-240	41.5
TCOM-300	33.44
TCOM-400	21.07

Tabla 2.2 Especificaciones de cables

2.1.3 CONECTORES

Los conectores [21] son importantes en el funcionamiento de un enlace puesto que una mala instalación de los mismos provocaría pérdidas en la señal, por lo tanto estos deben ser resistentes, soldados adecuadamente. Se los escoge de acuerdo a los conectores que posee la antena escogida y los de los demás equipos por lo tanto podemos tener de diferentes tipos y formas.

Ya que se ha escogido la direccional de grilla los conectores escogidos son de tipo N macho. En la Tabla 2.3 se presentan varios de ellos.







TIPO DE CONECTOR		Especificación para cable LMR 400
N Macho		Crimp: CNML400 Times EZ: EZ400NM Times LLPL: EZ400NMLLPL Amphenol Premium: SPECNM400
N Hembra		Crimp: CNFM400 Bulkhead: EZ400NFB Times EZ: EZ400NF Times LLPL: EZ400NFLLPL
RP TNC Macho		Crimp: CRTNC400 Times EZ: EZ400RTNCM
RP TNC Hembra		Crimp: CRTNCF400 Times EZ: EZ400RTNCF
RP SMA Macho		Crimp: CRSMAM400
RP SMA Hembra		N/A

Tabla 2.3 Comparación de conectores

2.1.4 RADIO DIGITAL

En la actualidad hay una tendencia para la sustitución de circuitos analógicos por circuitos digitales que desempeñan una función equivalente. La utilización de circuitos digitales presenta diversas ventajas ante sus predecesores analógicos, expresamente una mayor inmunidad a las variaciones de temperatura y al envejecimiento, una mayor integración permitiendo entre otras, una mayor miniaturización.

La tecnología del radio software fue iniciada por las fuerzas armadas de Estados Unidos para conseguir comunicaciones permanentes entre distintas bandas con un solo equipo. Básicamente se trata de trasladar a software las funciones que normalmente se realizan hasta ahora en hardware, este radio software también es conocido como radio digital.

Básicamente un radio digital está compuesto por:

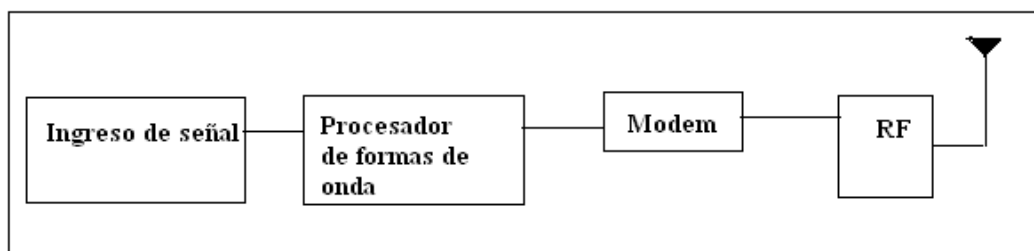


Figura 2.1 Esquema de radio digital

Donde RF se refiere a los convertidores análogo-digital y digital-análogo así como

también un conmutador Tx/Rx para la transmisión y recepción de las señales.

Los demás bloques se encuentran descritos en software. El bloque de ingreso de señal, se refiere a la preparación de la señal para ser procesada dependiendo si se esta recibiendo o enviando, así por ejemplo: en este bloque se puede encontrar filtros, en el caso de recibir la señal, el procesador de formas de onda es capaz de procesar la señal en tiempo real tanto en la transmisión como en la recepción creando formas de ondas a las frecuencias esperadas, el sistema sabe que forma de onda crear puesto que posee una tabla con código y frecuencia para crearla y enviarla luego al modem que no es más que un modulador/demodulador.

Como se ha visto los radios digitales se crean de acuerdo a las necesidades de los clientes y por ello no se encuentran normalmente en el mercado común, así lo más cercano a un radio software es un transmisor/receptor que no precisamente ofrece las misma ventajas que un radio software.

A continuación se presenta algunos transmisores/receptores del mercado:

1.- Radio modem conexión serie RS232

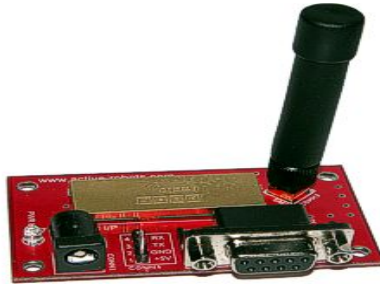


Figura 2.2 Radio modem conexión serie RS232

Radio módem con conexión serie RS232, permite transmitir datos de forma transparente para el usuario. El radio módem se entrega completo para su uso, incluyendo una pequeña antena helicoidal, un cable de conexión para puerto serie RS232 y un transformador de alimentación de 12V. Este radio módem esta basado en módulo de radio de Easy Step S350170 que tiene unas excelentes características técnicas entre las que destaca la posibilidad de tener 10 canales diferentes, selección del nivel de potencia entre 1 y 10 mW y posibilidad de ajustar la velocidad de conexión entre 2400 y 38400 baudios, presentando un nivel optimo de transmisión a 19200 baudios. Todas las opciones se ajustan fácilmente con el software de configuración que se puede bajar libremente. Una vez realizados los ajustes, estos quedan almacenados en la memoria interna del módulo de radio. Alimentación: 7 a 20 VDC. Dimensiones: 70 x 40 mm. Alcance: 250 M

2.- Radio Modem usb 19.2Kb easy radio 433Mhz S350173



Figura 2.6 Radio modem usb 19.2Kb easy radio 433Mhz

Completo radiomodem con conexión USB, permite transmitir y recibir datos desde cualquier PC a otro PC con un módulo similar, o bien a otro radiomodem como el S350200 o el S350205. El circuito incluye un convertidor de usb a puerto serie que una vez instalado en el PC crea un puerto serie virtual sobre el puerto USB. De esta forma, aparece un puerto serie en Windows que es totalmente compatible y transparente para las aplicaciones de Windows. Por otra parte el circuito incluye un radiomodem ER400TRS conectado a la salida serie del convertidor USB a serie y por lo tanto se tienen las mismas características técnicas que este. Un detalle que hay que tener en cuenta es que el bufer del circuito convertidor de usb a serie es 128 bytes, mientras que el del módulo transmisor es de 196, lo que implica que la longitud máxima del paquete es este caso de 128 bytes. La conexión USB proporciona varias ventajas como la compatibilidad plug and play y la alimentación desde el propio bus USB.

CAPÍTULO 3

3. DISEÑO

3.1 Diseño del radio software

A los sistemas de telecomunicaciones que utilizan procesamiento digital en las etapas de radio-frecuencia (RF), frecuencia intermedia (FI) o banda base (BB), damos el nombre de radios digitales.

A partir del momento en que la señal está en la forma digital las operaciones que sobre ella se efectúan pasan para el dominio del procesamiento digital de señal. La calidad de la implementación depende sólo de los algoritmos utilizados que obviamente están sujetos a las limitaciones de velocidad y capacidad impuestas por el procesador digital escogido. Este procesador puede ser un microprocesador, pero más frecuentemente es utilizado un procesador optimizado para el procesamiento digital de señal, conocido como DSP (Digital Signal Processor) [11]. Otra hipótesis, que ha ganado interés, es la utilización de circuitos integrados los de aplicación específica (ASIC) [12] o, en alternativa, la utilización de dispositivos lógicos programables (PLD) [13] tales como FPGA's y CPLD's. La producción de un ASIC sólo se hace económicamente viable si este fuera fabricado en grandes cantidades. Para productos fabricados en pequeñas cantidades, o durante la fase de desarrollo de un producto, la utilización de PLD's es la solución preferida. Una característica presente en los

PLD's es la posibilidad de que sean programados repetidas veces sin ser necesario retirarlos del circuito. Esto facilita mucho en la fase de desarrollo del producto y hace posible, por ejemplo, el envío de un fichero para los clientes conteniendo nuevas funcionalidades y/o correcciones de defectos. Recientemente a los radios digitales se les llama radio software por su funcionalidad de programar cuantas veces sean necesarias sus ficheros.

Cuando comparamos los PLD's con los DSP's poseen una mayor integración, una menor disipación de potencia y un mayor desempeño lo que los convierte en la primera elección para sistemas portátiles y que utilizan frecuencias de trabajo elevadas y grandes anchuras de banda.

Una tendencia reciente ha sido combinar FPGA con microprocesadores y periféricos relacionados como los DSP's para formar un "Sistema programable en un chip" [14]. Estos sistemas programables tienen un costo adicional por la adquisición de los derechos de propiedad intelectual del software, Matlab [10], y no es conveniente en el desarrollo o producción en pequeña escala por su alto costo.

Por las razones presentadas, y teniendo en cuenta el aumento constante de la capacidad de procesamiento y progresiva reducción del precio, será de esperar que cada vez más sistemas de radio digital utilicen PLD's para efectuar el procesamiento de la señal digital.

En este capítulo se presenta el diseño de un radio software con sus diferentes bloques y una descripción de cada una. El tipo de modulación y demodulación que se

va a detallar para el radio software es QPSK.

3.1.1 Diagrama del radio software

El radio software es un sistema de telecomunicaciones que utiliza procesamiento de señales digitales. Sin embargo el canal o el medio de transmisión es un inherente analógico por lo que es importante abordar las etapas que efectúan la conversión del dominio analógico para el digital y viceversa. Las operaciones efectuadas en la transmisión son semejantes a las efectuadas en la recepción. En tanto la complejidad del receptor es normalmente mayor que la del transmisor una vez que necesita extraer una señal débil del medio del ruido.

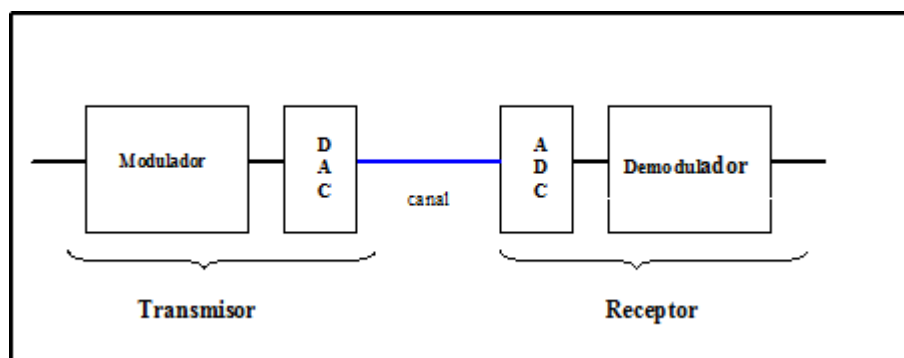


Figura 3.1 Diagrama de un radio software

Como se presenta en la Figura 3.1, el transmisor está compuesto de un modulador y un DAC (Convertidor Digital - Analógico). El canal puede ser un cable coaxial o un UTP (Cable de Par Trenzado no Blindado), una fibra óptica o el espacio libre. El

receptor esta compuesto de un ADC (Convertidor Analógico - Digital) y un demodulador.

3.1.2. Diagrama del modulador

El diagrama de un modulador QPSK, de un radio software, se presenta en la Figura 3.2.

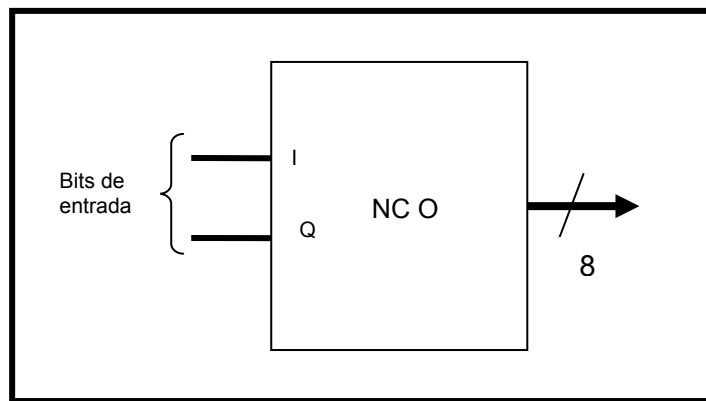


Figura 3.2 Diagrama de un modulador QPSK

Como se observa en la Figura 3.2, el componente principal de un modulador QPSK es un NCO (Oscilador Controlado Numéricamente) que se detallara más adelante en este capítulo.

3.1.3. Conversión Digital/Analógica

Para la conversión de los datos del modulador QPSK se utilizó un convertidor

digital–analógico, este dispositivo es un DAC 0808 de la National Semiconductor [16]. Este es un dispositivo de 8 bits de resolución que permite una tasa de conversión de 150 ns. La facilidad de obtención en el mercado local y el encapsulamiento DIP fueron las razones principales que condujeron a la elección de este componente. En la Figura 3.3 se observa el diagrama de conexión del dispositivo.

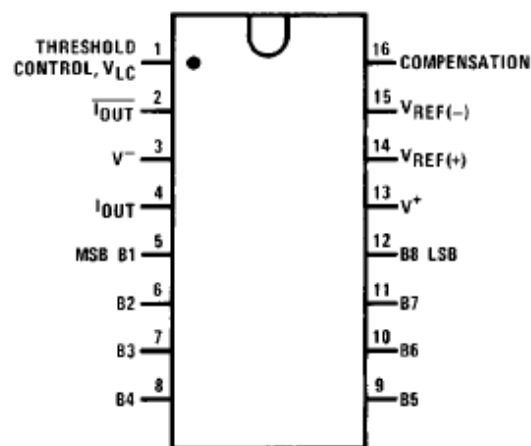


Figura 3.3 Diagrama de conexión

Este procedimiento de conversión es necesario ya que el canal de transmisión es un medio analógico.

La señal analógica así obtenida no es una señal continua, sino que se obtiene un número discreto de escalones a consecuencia de la discretización de las entradas. Si el DAC tiene más bits en la entrada, la salida será una aproximación de una onda analógica continua.

3.1.4. Modulación QPSK

La modulación por cuadratura de desplazamiento de fase o QPSK (Quadrature Phase Shift Keying) [4] es una forma de modulación angular que consiste en hacer variar la fase de la portadora entre un número de valores discretos. Estos valores discretos son las entradas I (canal en fase) y Q (canal de cuadratura).

Entrada binaria		Fase de salida QPSK
Q	I	
0	0	-135°
0	1	-45°
1	0	+135°
1	1	+45°

Tabla 3.1 La tabla de verdad del Modulador QPSK

En la Figura 3.4 puede verse que cada una de las cuatro posibles fases de salida tiene, exactamente, la misma amplitud. En consecuencia, la información binaria tiene que ser codificada por completo en la fase de la señal de salida.



Figura 3.4 Relación de la fase de salida contra tiempo para un modulador

QPSK

En la Figura 3.5 muestra el diagrama fasorial de una modulación QPSK.

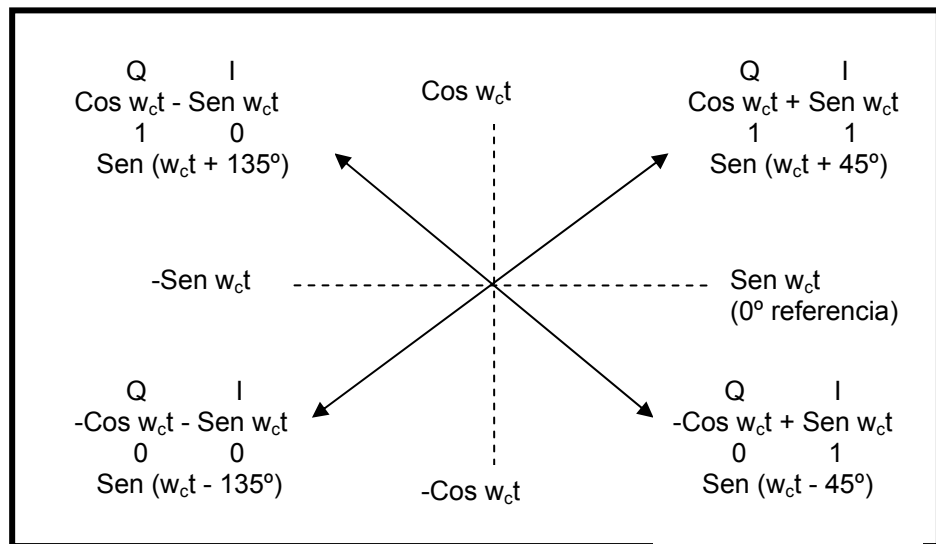


Figura 3.5 diagrama fasorial del QPSK

3.1.5. Oscilador

En los sistemas de telecomunicaciones los osciladores tienen presencia obligatoria normalmente con la función de generar la frecuencia de los osciladores locales.

El método más popular para implementar osciladores digitales es el método de la acumulación de fase [1].

3.1.5.1. Método de la acumulación de fase

Este oscilador es también llamado oscilador controlado numéricamente (NCO) por su semejanza al oscilador controlado por tensión (VCO) analógico. La Figura 3.6 ilustra su funcionamiento.

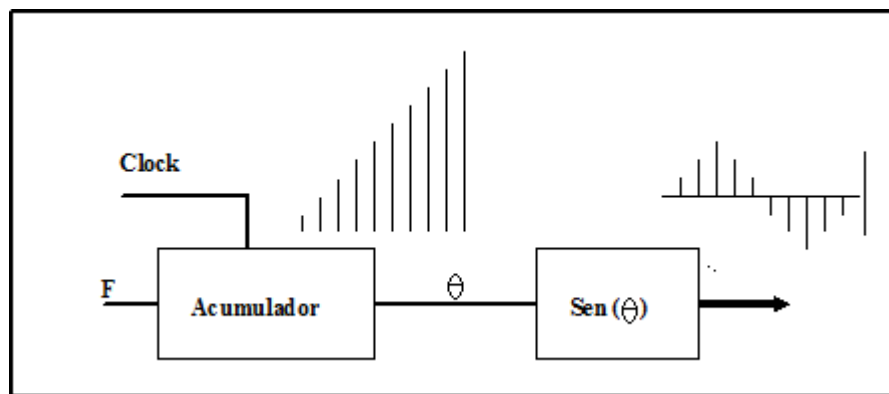


Figura 3.6 Oscilador de acumulación de fase

Vamos a explicar el funcionamiento del NCO. En cada impulso de reloj el acumulador coloca en su salida la suma de la entrada con el valor de la salida anterior. Como el acumulador tiene un número finito de bits llegará una altura en que el valor máximo será desbordado y el acumulador continuará a contar a partir de cero. La señal en su salida será entonces una onda en diente de sierra. Esta señal es después entregada a un bloque que calcula el seno o coseno. La salida del acumulador es por lo tanto la fase de la senoide.

Cuanto mayor sea el valor F presente en la entrada del acumulador, mayores serán los saltos de la fase y en consecuencia más elevada será la frecuencia de la senoide. Existen varios métodos para calcular el coseno o seno. Vamos sólo a mencionar el método de la tabla (LUT) que consiste en guardar en una memoria (ROM por ejemplo) el valor del coseno o seno en varios instantes y usar el valor presente en la salida del acumulador como un índice en esta memoria.

Un diagrama de bloques de un NCO que utiliza una LUT está representado en la figura 3.7. Este NCO utiliza un acumulador con N bits. De estos N bits, los M (con $M = N$) más significativos son entregadas a la LUT. La resolución de salida de la LUT (resolución en amplitud) es de D bits.

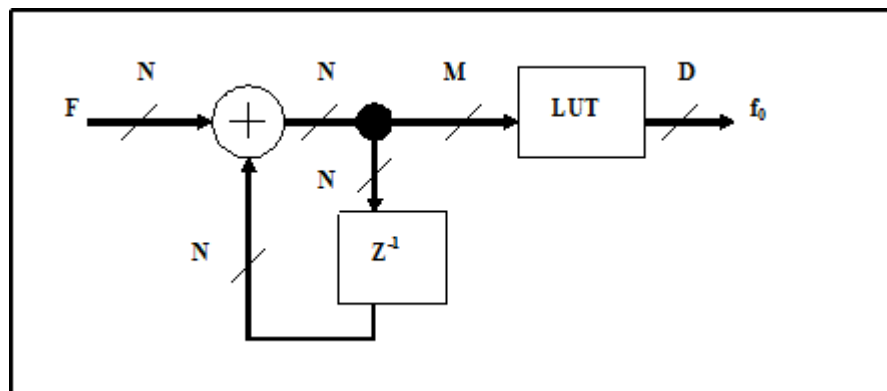


Figura 3.7 Diagrama de bloques de un NCO con LUT

Como se comentó antes, la señal presente en la salida del acumulador es la fase instantánea de la senoide. Esta señal varía entre 0 a $2^N - 1$ que corresponde la variación de la fase entre 0 a 2π o 360° .

3.1.5.2. Frecuencia de salida

La frecuencia de salida de la señal generada por el NCO esta dado por

$$f_0 = f_{\text{clk}} (F / 2^N) \quad [\text{Hz}] \quad 3.1$$

Donde f_{clk} es la frecuencia de reloj del sistema

F es el incremento de fase

N es el número de bits en la entrada del acumulador

Como se observa en la formula 3.1, la entrada F del acumulador define la frecuencia de salida del NCO. La frecuencia de salida es también conocida como la frecuencia de portadora f_c (carrier frequency).

3.1.5.3. Control de fase

Nosotros podemos controlar la fase de la señal generada por el NCO haciendo una pequeña modificación, esbozado en la figura 3.8. El valor P es sumada al valor presente en la salida del acumulador antes de ser entregada a la LUT. De este modo, el bit más significativo de P representa 180° , el siguiente representa 90° , el próximo 45° , y así sucesivamente. La resolución de fase, o sea la diferencia de fase más

pequeña que el NCO consigue generar es dada por $360/2^N$ grados.

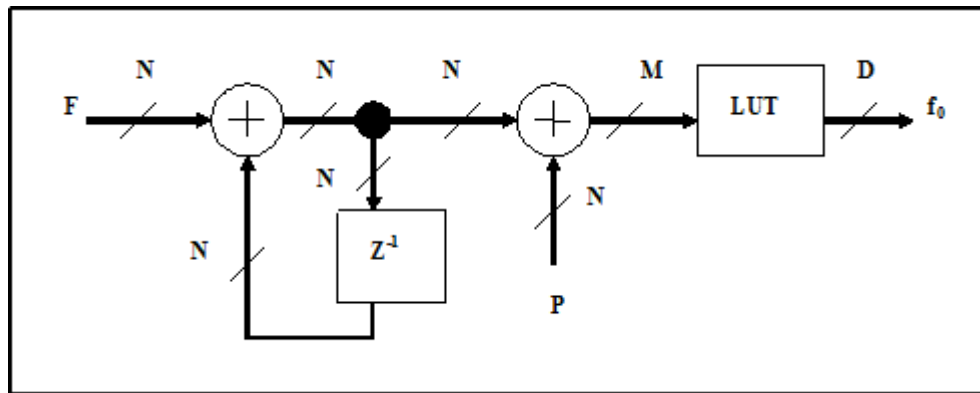


Figura 3.8 Diagrama de bloques del NCO con controlador de fase.

3.1.5.4. Pureza espectral

El ruido es un problema que enfrenta el NCO al momento de generar la señal de salida. El ruido introducido al NCO es causado por los errores de amplitud y de fase.

Los errores de fase se manifiestan mediante la reducción del rango dinámico libre de espurios (SFDR). Por lo tanto, hay que tener en cuenta que cada bit adicional de fase que se añade producirá una mejora del SFDR de aproximadamente 6dB.

La supresión de espurios que deberá tener la señal de salida del NCO está dada por

$$S = N * 6 \text{ [dB]} \quad 3.2$$

Donde N es el número de bits en la entrada del acumulador del NCO y nos permite determinar la fase de la señal de salida (seno o coseno).

Los errores de amplitud se manifiestan mediante la reducción de la relación señal a ruido (SNR). Por lo tanto, hay que tener en cuenta que cada bit extra de amplitud que se añada producirá una mejora de la SNR de aproximadamente 6dB.

La relación señal a ruido que deberá tener la señal de salida del NCO está dada por

$$\text{SNR} = \mathbf{D} * 6 \text{ [dB]} \qquad \mathbf{3.3}$$

Donde **D** es el número de bits en la salida de la LUT del NCO y nos permite determinar la amplitud de la señal de salida.

3.1.5.5. NCO de cuadratura

En ciertas aplicaciones es necesaria la utilización de dos senoides en cuadratura, o sea un seno y un coseno. En el caso de la demodulación de cuadratura QPSK.

El NCO puede ser fácilmente adaptado para generar ambas componentes. Uno de los métodos más simples consiste en utilizar dos LUT's, una para el seno y otra para el coseno. Otro método utiliza sólo una LUT y hace dos consultas en la tabla por cada muestra. La primera consulta es hecha en la dirección que viene directamente del acumulador. La segunda consulta deberá estar desfasada 90° lo que equivale a sumar

2^{N-2} a la dirección del acumulador. Esta técnica necesita de un reloj que funcione al doble de la frecuencia de muestreo.

3.1.5.6. Compresión de la LUT

La función de la LUT es efectuar la conversión de la fase actual para el valor instantáneo de la amplitud, o sea, efectuar la transformación $\theta \rightarrow \sin(\theta)$.

Para la compresión de la LUT se aprovecha las simetrías que existen entre los cuatro cuadrantes de la senoide. Como se observa en la Figura 3.9, cada cuadrante de la senoide está dividido en 90° . Por lo cual se almacena solamente el primer cuadrante en la LUT para generar un coseno o seno.

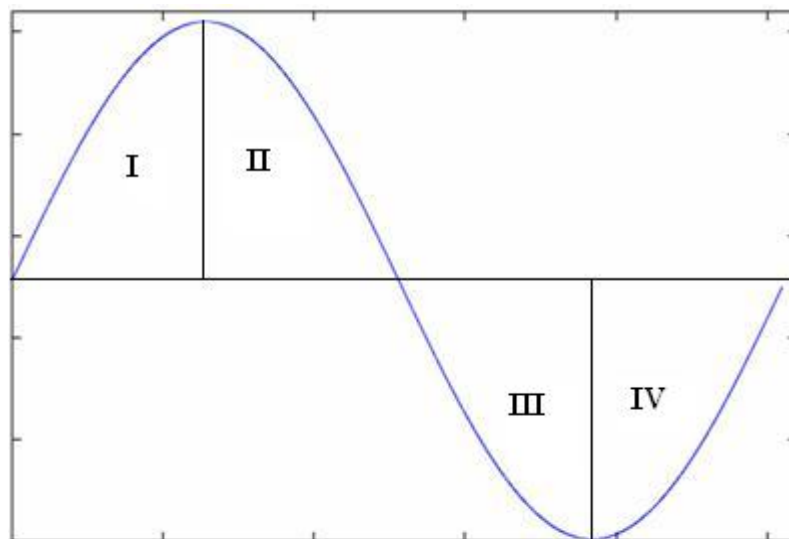


Figura 3.9 La senoide dividida en cuadrantes

La compresión de la LUT se utiliza cuando los valores de la tabla son demasiado grandes y engorrosos de hacer, ya que estos valores uno a uno tiene que ser introducidos en una memoria, y nos permite también reducir la memoria utilizada en la FPGA.

3.1.6. Generación de la tabla senoide

Para la generación de la tabla seno [3] se considera un parámetro muy importante que es el número de bits de salida del NCO. Cuando utilizamos 8 bits de salida del NCO, implica generar 256 muestras 2^8 .

Con la siguiente ecuación generaremos una tabla de un seno

$$Y = 127 * \sin(2\pi * x / 256) \quad 0 \leq x \leq 255 \quad 3.4$$

Como se observa en la formula 3.4, x es el número de muestras 2^D y obtenemos un rango Y que va desde -127 a +127 apropiado cuando se utiliza números complemento 2 de 8 bits.

Estos valores Y de complemento 2 son guardados en la LUT del NCO y son enviados directamente a un convertidor digital-analógico para generar la onda senoide.

3.1.6.1. Conversión complemento 2 a unipolar sin signo

Como hardware de salida del NCO normalmente se utiliza un convertidor digital-analógico el cual utiliza valores unipolares sin signo también. Para convertir números complemento 2 a formato unipolar sin signo, le sumamos 128.

De la siguiente manera se genera la tabla seno

$$Y = 128 + 127 * \sin(2\pi * x / 256) \quad 0 \leq x \leq 255 \quad 3.5$$

Estos valores Y convertidos en formato unipolar sin signo son guardados en la LUT del NCO y son enviados directamente a un convertidor digital-analógico para generar la onda senoide.

3.1.7. Diagrama del demodulador

El diagrama de un demodulador QPSK, de un radio software, se presenta en la Figura 3.9.

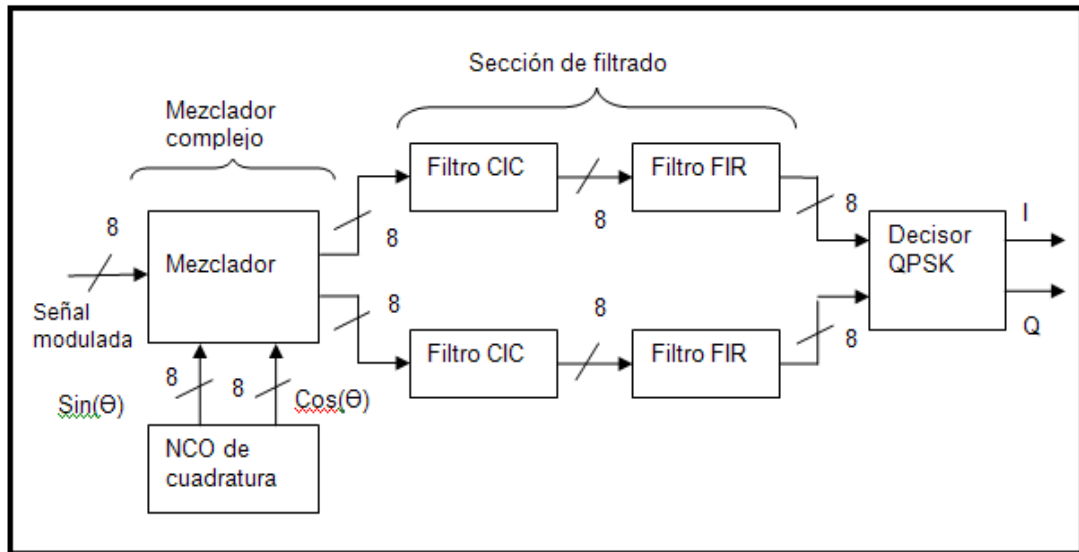


Figura 3.9 Diagrama de un demodulador QPSK

Un demodulador QPSK [2] está compuesto por tres bloques que son: un mezclador complejo, la sección de filtrado y un decisor QPSK. A continuación se detalla los diferentes bloques que conforman el demodulador.

3.1.8. Conversión Analógica/Digital

Como se dijo anteriormente en este capítulo el medio de transmisión es un inherente analógico así que antes de llegar la señal modulada al demodulador QPSK se necesita un convertidor analógico-digital. La misión de un convertidor analógico-digital es obtener una representación digital de una magnitud analógica. El convertidor usado para este proceso es el ADC 1175 de la National Semiconductor [16]. Este es un dispositivo de 8 bits de resolución de salida que permite una tasa de conversión máxima de 20 Mhz y trabaja con una señal analógica unipolar o con signo

(complemento 2). Las razones principales que condujeron a la elección de este componente fueron la velocidad de conversión de los datos y el encapsulamiento DIP. En la figura 3.10 se representa el diagrama de conexión.

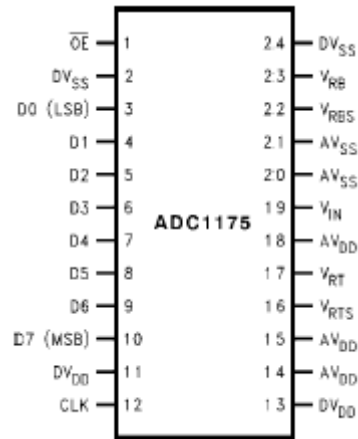


Figura 3.10 Diagrama de conexión

El ADC 1175 trabaja en una frecuencia de muestreo f_s . La frecuencia de muestreo debe ser de un valor tal, que en el proceso de digitalización, se asegure la completa reconstrucción de la señal original que proviene del transmisor. Esta frecuencia de muestreo tiene muy en cuenta el Teorema de Nyquist que dice: “La frecuencia de muestreo f_s debe ser por lo menos el doble que la frecuencia máxima f_{\max} que se desee generar”.

$$f_s \geq 2f_{\max}$$

3.1.9. Demodulación QPSK

La demodulación QPSK [4] es una forma de demodulación angular que consiste en recuperar los valores discretos, I y Q, a través de la entrada que tiene una variación de la fase de la portadora. Dependiendo de la variación de la fase de la portadora la información binaria tiene que ser decodificada por completo en la salida.

3.1.10. Desplazamiento en frecuencia y Frecuencia Intermedia

La señal modulada entregada, que está en frecuencia intermedia, debe convertirse en formato digital. El convertidor analógico-digital hace el muestreo de la señal de banda limitada centrada en una frecuencia intermedia FI, lo que se conoce como IF-sampling. Este muestreo genera unas imágenes de la banda de interés que quedan centradas en nf_s , donde $n = 1, 2, 3, 4, \text{etc.}$, siendo la frecuencia imagen más baja la que interesa.

El desplazamiento en frecuencia, también conocido como Down-conversion se consigue mezclando, o multiplicando, las muestras generadas por el convertidor analógico-digital por la salida de un generador de señales senoides (NCO de cuadratura).

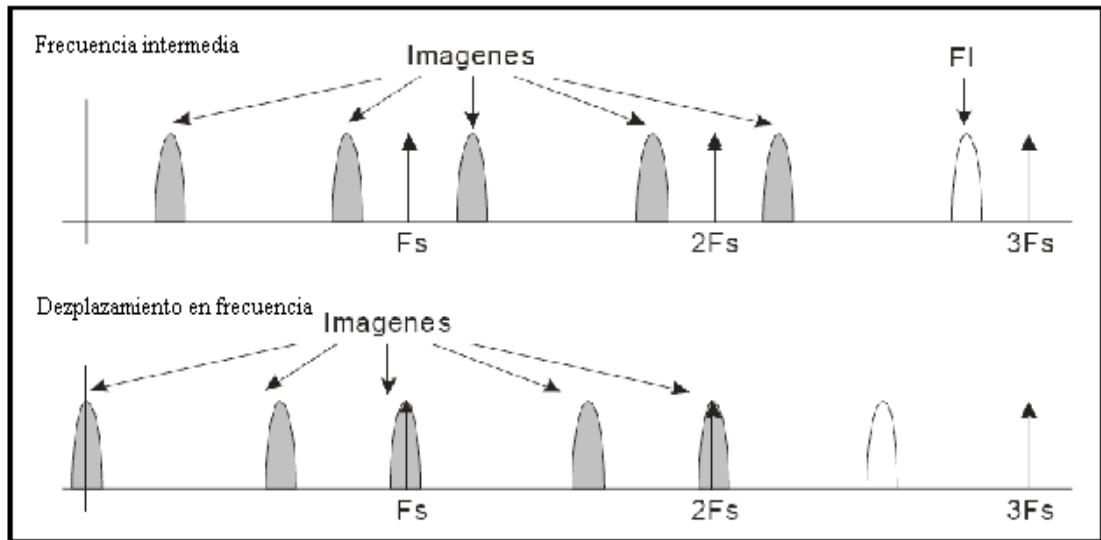


Figura 3.11 Espectro de la señal después de pasar por Desplazamiento en frecuencia y Frecuencia Intermedia

De esta manera, mediante el Desplazamiento en frecuencia y Frecuencia Intermedia Se trasladará la señal de interés situada en FI a banda base para así poder demodularla posteriormente.

3.1.11. Mezclador Complejo

El mezclador complejo se encarga de acondicionar la señal de entrada de tal manera que se consiga dos cosas: la primera es desplazar la frecuencia de interés a banda base y la segunda descomponerla en señales I y Q. Estos dos requisitos se logran multiplicando la señal de entrada por dos senoides a frecuencia de portadora f_c (carrier frequency), con 90° de desfase relativo entre ellas.

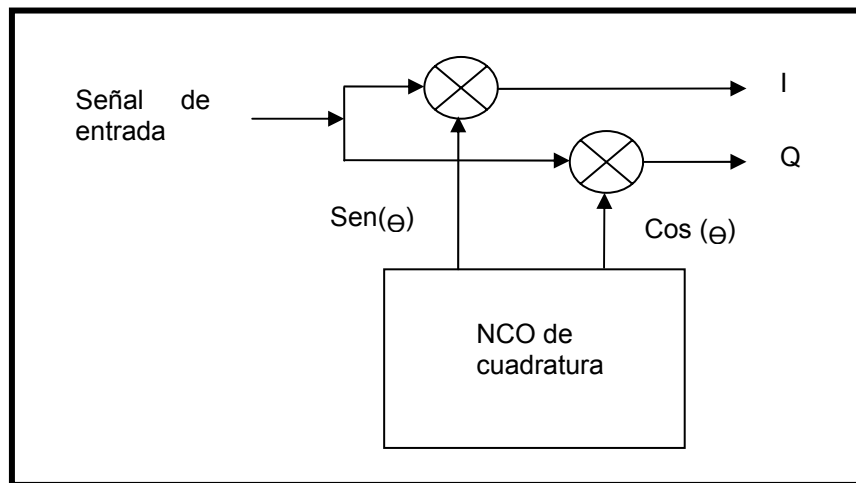


Figura 3.12 Diagrama de bloques del mezclador complejo

3.1.12. Filtrado

En la entrada del bloque de filtrado se tiene la banda de interés en banda base. La señal que está en banda base tendrá que ser filtrada para eliminar las interferencias, provenientes del medio de transmisión. El bloque de filtrado está compuesto por dos filtros:

- Filtro CIC
- Filtro FIR

El filtro CIC realiza un prefiltrado de la señal y reduce la frecuencia de muestreo mediante un diezmado de las muestras. Este diezmado se hace para reducir la carga computacional en el siguiente filtro.

El **filtro FIR** es un filtro paso-bajo que deja pasar solamente la banda de interés.

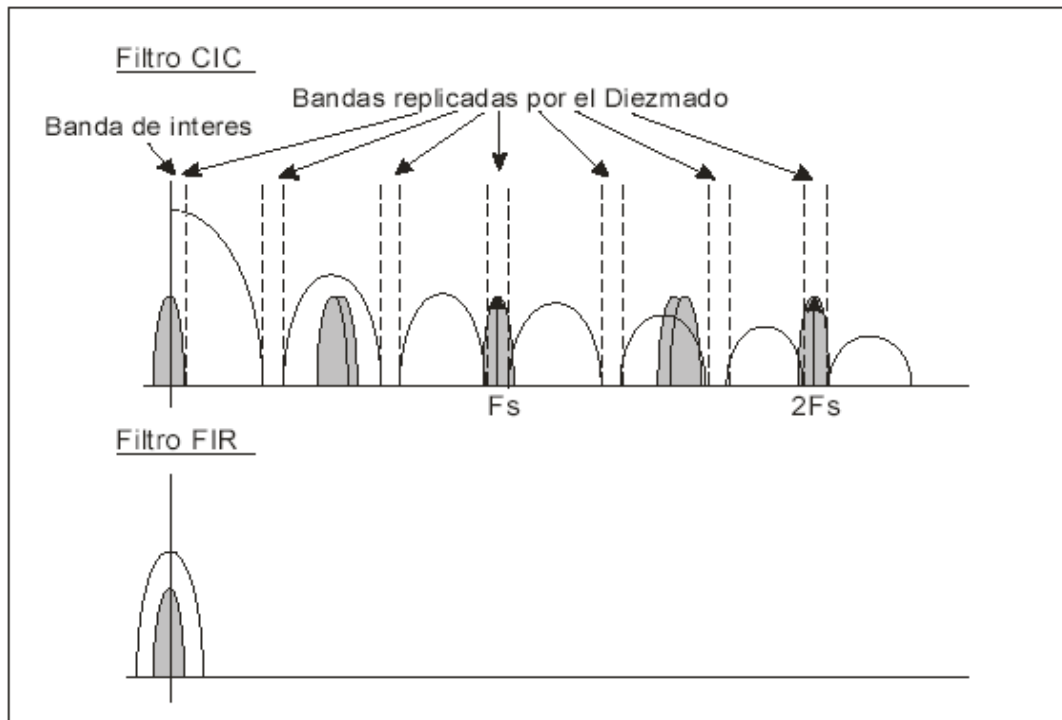


Figura 3.13 Espectro de la señal después de pasar por el filtro CIC y el filtro FIR

En la Figura 3.13 se muestra el espectro de la señal después de pasar por el filtro CIC y el filtro FIR. El filtro CIC inserta más imágenes, a parte de las ya añadidas por el medio de transmisión. Estas imágenes insertan interferencias a la señal de interés. El filtro CIC inserta imágenes constructivas que aumenta la ganancia de la banda de interés. El filtro FIR tendrá una respuesta paso-bajos que atenúa todo lo que esté fuera de la banda de interés.

3.1.12.1. Filtro CIC

El filtro CIC (Cascaded integrator-comb o cascada integrador-peine) o filtro de Hogenau [2] [5] [6] [7], es un filtro multitasa (multirate) en la cual diezma la señal a una tasa de muestreo mucho más baja provocando así una menor carga computacional y añade imágenes constructivas, provocando ganancia a la banda de interés. Este tipo de filtro está dividido en filtro CIC diezmador y filtro CIC interpolador. Para el diseño se aplicó el filtro CIC diezmador y si desea información sobre el filtro CIC interpolador consulte las referencias [5] [6].

La Figura 3.14 muestra la estructura básica de un filtro CIC diezmador. La sección de integradores o acumuladores consiste de N etapas de integradores ideales operando a la velocidad de muestreo f_s . Cada etapa es implementada como un filtro de un polo con coeficiente de retroalimentación unitario. La función de transferencia para un integrador es

$$H_1(z) = 1 / (1 - z^{-1}) \quad 3.6$$

La sección de filtro peine (comb filter) opera a una baja velocidad de muestreo f_s/R . Donde R es un número entero y representa el factor de cambio de la velocidad. Esta

sección consiste de N etapas de peine con un retardo diferencial de M muestras por etapa. El retardo diferencial es un parámetro de diseño de filtro utilizado para el control de la respuesta de frecuencia del filtro. M se restringe a los valores 1 ó 2. Un filtro peine coloca en su salida la resta de la entrada actual con el valor de la entrada anterior. La función de transferencia de una sola etapa de peine a una f_s de entrada es

$$H_C(z) = 1 - z^{-RM} \quad 3.7$$

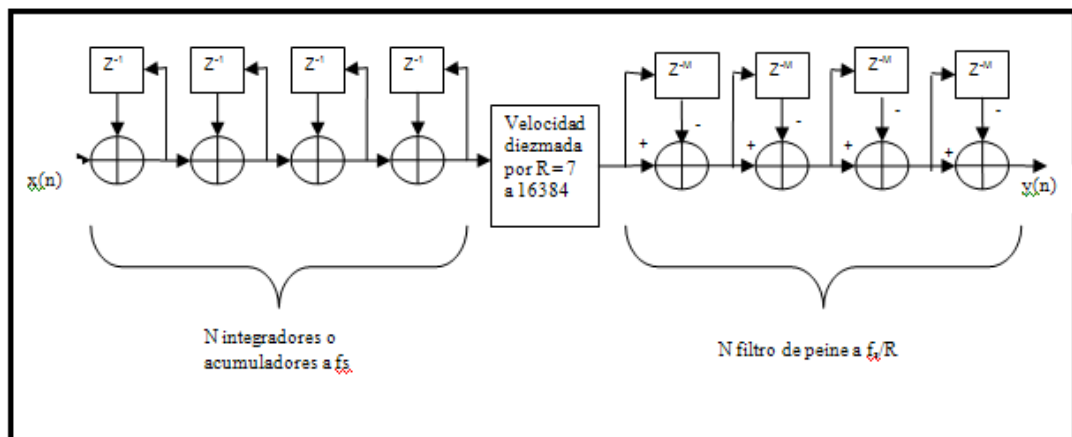


Figura 3.14 Estructura básica de un filtro CIC diezmador

Resumiendo, un filtro CIC diezmador debería tener N integradores en cascada sincronizados a una frecuencia f_s , seguido por un cambio de frecuencia de factor R, y, por ultimo, N filtros de peine en cascada sincronizados a una frecuencia f_s/R .

La función de transferencia para un filtro CIC a una frecuencia f es:

$$\mathbf{H}(z) = \mathbf{H}_I^N(z) \mathbf{H}_C^N(z) = (1 - z^{-RM})^N / (1 - z^{-1})^N = \left(\sum_{K=0}^{RM-1} z^{-K} \right)^N \quad \mathbf{3.8}$$

Esta ecuación muestra que un filtro CIC diezmador es equivalente a una cascada de N filtros FIR, teniendo cada uno una respuesta impulsional rectangular. Como todos los coeficientes de estos filtros FIR son unitarios, y por lo tanto simétricos, un filtro CIC también tiene una fase lineal y un retardo de grupo constante.

Para los CIC diezmadores, la ganancia G a la salida del final de la sección de filtros de peine es

$$\mathbf{G} = (\mathbf{RM})^N \quad \mathbf{3.9}$$

Teniendo en cuenta que se trabaja en el formato unipolar sin signo, se puede utilizar este resultado para calcular el número de bits necesarios para el último filtro de peine debido al aumento de bits. Si B_{in} es el número de bits de entrada, entonces el número de bits de salida, B_{out} , es

$$\mathbf{B}_{out} = [\mathbf{N} \log_2 \mathbf{RM} + \mathbf{B}_{in}] \quad \mathbf{3.10}$$

También puede ser que se necesiten B_{out} bits para cada integrador y filtro de peine. La entrada necesita ser extendida a B_{out} , pero los LSB puede ser truncado o redondeado en etapas posteriores.

3.1.12.2. Filtro FIR

Una vez que se ha diezmado la señal, el siguiente paso será eliminar las interferencias. Para esto bastará con utilizar un filtrado paso-bajo que atenué todas las frecuencias que estén fuera de la banda de interés.

Esto se consigue mediante la utilización de un filtro FIR (Finite Impulse Response) o Respuesta finita al impulso [2] [7] [8]. Se trata de un filtro en el que, como su nombre indica, si la entrada es un impulso, la salida tendrá un número finito de términos no nulos que son los coeficientes que configuran el tipo de filtrado. Por lo tanto, si variamos estos coeficientes, los cuales pueden cambiarse en cualquier momento, puede generarse cualquier tipo de filtrado.

Para obtener la salida sólo se utilizan entradas actuales y anteriores, lo que hará que estos filtros tengan todos los polos en el origen, por lo que su comportamiento será estable. Su expresión en el dominio n es:

$$y_n = \sum_{K=0}^{N-1} h_K x(n - k)$$

3.11

En la expresión anterior N es el orden del filtro, que también coincide con el número

de términos $x(n)$ y con el número de coeficientes del filtro h_k .

Aplicando la transformada z a la expresión anterior:

$$H(z) = \sum_{k=0}^{N-1} h_k z^{-k} \quad 3.12$$

Por lo tanto, para generar el filtrado basta con almacenar, mediante retardos, tantas entradas sucesivas como coeficientes (orden) tenga el filtro, de manera que en cada ciclo se pueda realizar la convolución de estas entradas con la respuesta impulsional de filtro, mediante la multiplicación de cada una de las entradas retardadas por su correspondiente coeficiente. El filtro tendrá entonces la siguiente estructura:

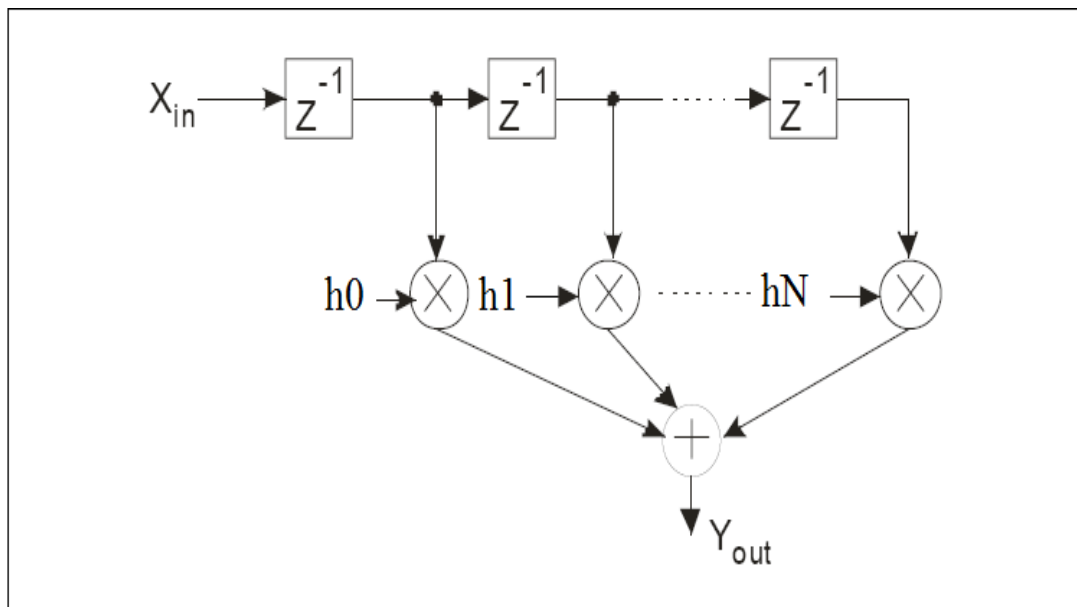


Figura 3.15 Esquema de bloque del filtro FIR

Para el diseño se utilizó filtros FIR de fase lineal lo cual permite ciertas propiedades en la simetría de los coeficientes. Esta simetría permite reducir a la mitad el número de multiplicadores necesarios, realizando la suma de las muestras pasadas y luego multiplicando por el coeficiente simétrico, tal y como se ilustra en la Figura 3.16. La consideración de simetría logra un ahorro considerable en el dispositivo FPGA.

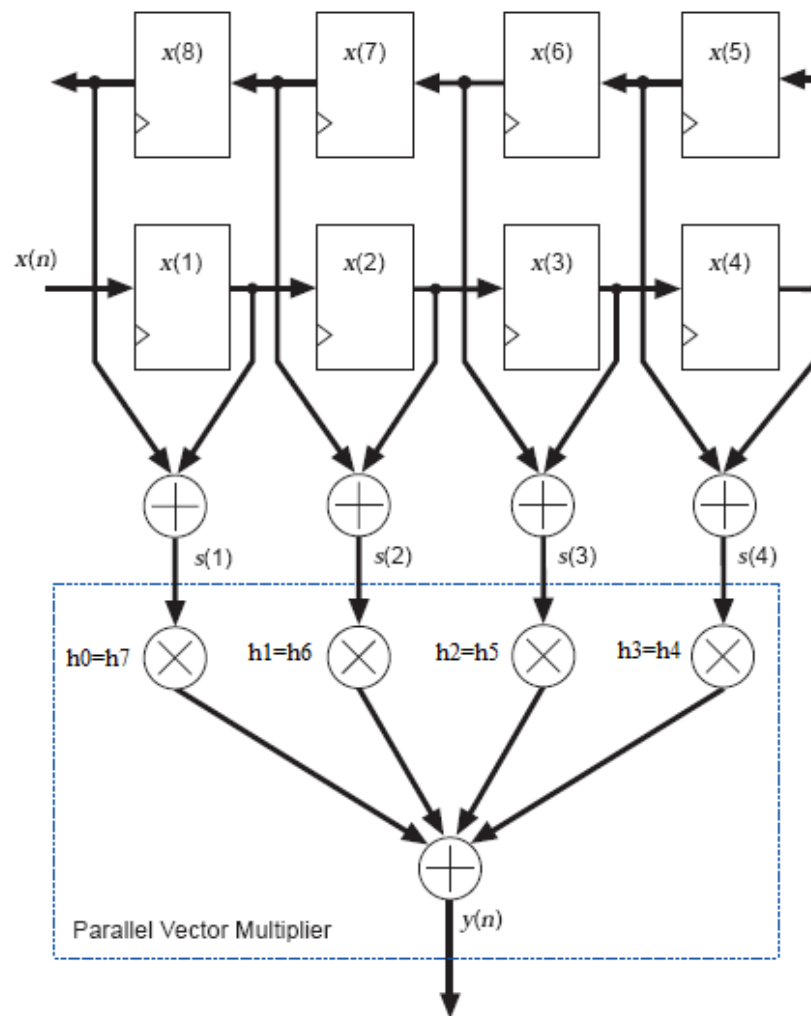


Figura 3.16 Diagrama de bloque de un filtro FIR que aprovecha la simetría de los coeficiente

3.1.13. Decisor QPSK

Una vez filtradas las componentes en fase y cuadratura se tendrá que interpretar la información que se recibe, la cual depende de la modulación QPSK. El bloque encargado de decodificar los valores provenientes de las componentes I y Q, ya filtrados, es un decisor [2] QPSK. La salida de este bloque depende de la constelación QPSK que se presenta en la Figura 3.17.

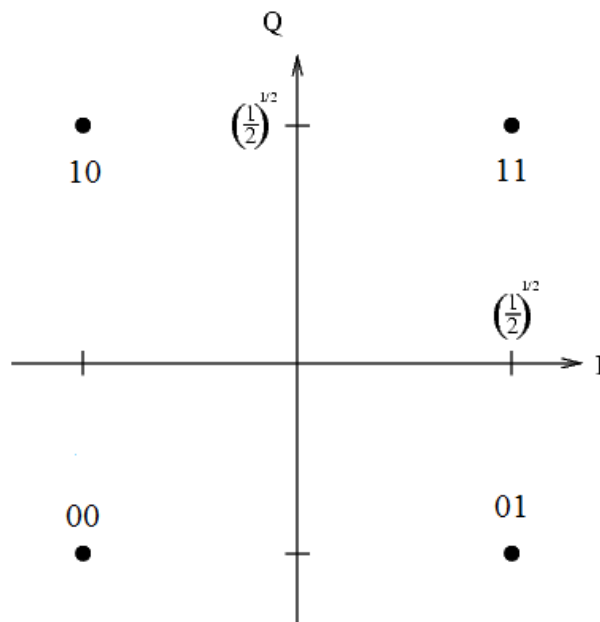


Figura 3.17 Constelación QPSK

Como se presenta en la figura anterior, a cada combinación de valores IQ se le asigna un código, hasta un total de 4 combinaciones, las cuales se mostrará a la salida del decisor en relación a las entradas que estén presentes en ese momento.

3.2 Diseño del Enlace

3.2.1 Mapa del terreno

El siguiente mapa (Figura 3.18) fue obtenido de la carta topográfica del Instituto Militar Geográfico donde se puede apreciar las curvas de nivel del terreno y las alturas exactas de los puntos donde se desea realizar el enlace.



Figura 3.18 Mapa general

En el siguiente mapa (Figura 3.19) podemos observar de cerca los lugares donde se desea realizar el enlace Punguín – Nítiluisa, encontrándose Nítiluisa a una altura menor que Punguín. El enlace atravesará las poblaciones de: La Moya, Rumicruz, Santa Lucía Chica entre las más importantes

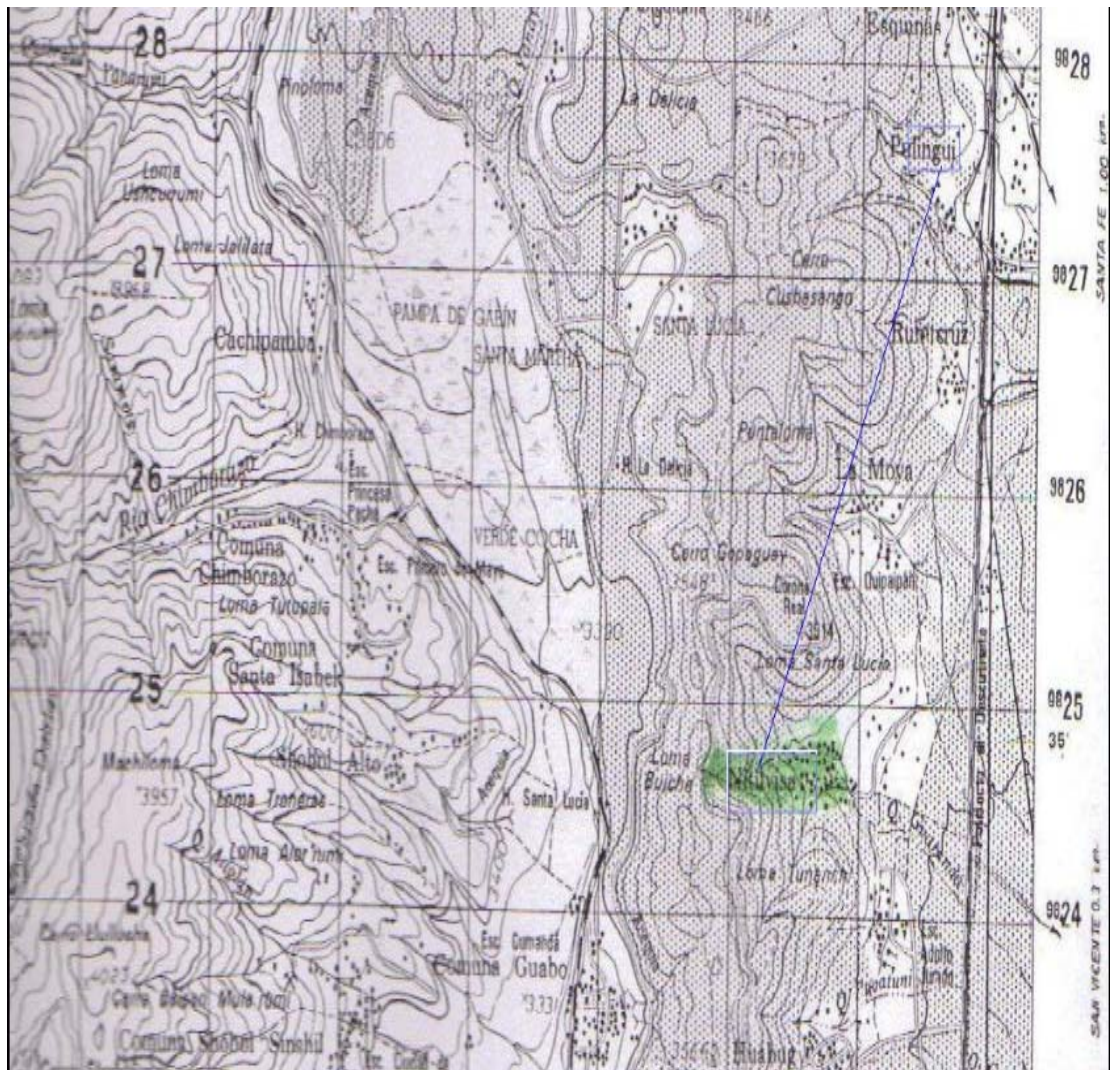


Figura 3.19 Mapa específico

3.2.2 Cálculo del margen de ganancia

Para realizar el cálculo de un enlace es importante tomar en cuenta que tipo de información se va transmitir en el caso de este sistema de comunicaciones la información son exclusivamente datos.

Un sistema de comunicación consta básicamente de dos radios, dos antenas y una trayectoria entre los mismos. Para asegurarse de que si existirá una comunicación se debe realizar un cálculo del margen de ganancia, también llamado presupuesto de potencia, este dependerá de la potencia de los equipos y también de las pérdidas de potencia tanto en la trayectoria como en las conexiones.

Así para el cálculo del margen de ganancia vamos a necesitar de los siguientes conceptos:

Potencia de Transmisión. Es la potencia emitida por el radio que se encuentra en el lado del transmisor y que debe estar en los rangos permitidos por nuestro país. Se expresa en milivatios o en dBm. La potencia de transmisión tiene un rango de 30mW a 200mW o más.

La potencia TX a menudo depende de la tasa de transmisión. La potencia TX de un dispositivo dado debe ser especificada en los manuales provistos por el fabricante, pero algunas veces puede ser difícil de encontrar.

Ganancia de las Antenas. La ganancia depende del tipo de antena sea esta omnidireccional, direccional, parabólica u otra. Hay que tener en cuenta que para lograr obtener la mayor ganancia especificada de la antena hay que realizar una instalación correcta cuidando de la orientación.

Las antenas son dispositivos pasivos que crean el efecto de amplificación debido a su forma física. Las antenas tienen las mismas características cuando reciben que cuando transmiten. Las antenas parabólicas tienen una ganancia de 19-24 dBi, las antenas omnidireccionales de 5-12 dBi, y las antenas sectoriales, de 12-15 dBi.

El Mínimo Nivel de Señal Recibida, o simplemente, la sensibilidad del receptor. El RSL (por su sigla en inglés) mínimo es expresado siempre como dBm negativos y es el nivel más bajo de señal que la red inalámbrica puede distinguir. El mínimo va a ser generalmente en el rango de -75 a -95 dBm. Al igual que la potencia TX, las especificaciones RSL deben ser provistas por el fabricante del equipo.

Pérdidas en los Cables. Este se refiere a la pérdida que experimenta la señal al atravesar el cable entre la antena y el radio, esta pérdida depende del grosor del cable y de la frecuencia de la señal, en general un cable más grueso y más rígido presenta una menor atenuación. La pérdida de señal para cables coaxiales cortos incluyendo los conectores es bastante baja, del rango de 2-3 dB.

Pérdida en la trayectoria, La potencia de la señal se ve disminuida por la dispersión geométrica del frente de onda, conocida comúnmente como pérdida en el espacio libre. Ignorando todo lo demás, cuanto más lejanos los dos radios, más pequeña la

señal recibida debido a la pérdida en el espacio libre. Esto es independiente del medio ambiente, se debe solamente a la distancia.

$$L_{\text{espacio libre}} = 40 + 20 \cdot \log(r) \quad \mathbf{3.13}$$

Donde r es la distancia entre los radios en (m).

Sensibilidad del receptor: La sensibilidad del receptor es un factor a tener muy en cuenta puesto que nos da el valor de potencia necesario para poder recibir la señal, es necesario que el margen de potencia sea mayor que el valor de la sensibilidad del receptor para asegurar una buena recepción y la tasa de bits que deseamos.

Cálculo del margen de ganancia:

Margen = Potencia de Transmisión [dBm] – Pérdidas en el cable TX [dB] + Ganancia de AntenaTX [dBi] - pérdida en la trayectoria del Espacio Abierto [dB] + Ganancia de Antena RX [dBi]-Pérdida de Cable RX [dB] - Sensibilidad del receptor [dBm]

Ganancia de Antenas

Antena Radio 1 (dBi)	+ Antena Radio 2 (dBi)	= Ganancia Total de la Antena
24	24	48

Tabla 3.2 Ganancia de antenas

Pérdidas

Pérdida en los cables (dB) Radio 1	Pérdida en los cables (dB) Radio 2	Pérdida en el espacio libre (dB)	= Pérdida Total (dB)
2	2	112	116

Tabla 3.3 Pérdidas

Cálculo del Margen de Ganancia de la Señal en el Radio 1

Potencia TX Radio 1	+ Ganancia de la Antena	- Pérdida Total	= Señal	➤ Sensibilidad del Radio 2
18	48	-116	-69	-85

Tabla 3.4 Cálculo del margen de ganancia radio 1

Cálculo del Margen de Ganancia de la Señal en el Radio 2

Potencia TX Radio 1	+ Ganancia de la Antena	- Pérdida Total	= Señal	➤ Sensibilidad del Radio 2
18	48	-116	-69	-85

Tabla 3.5 Cálculo del margen de ganancia radio 2

Por lo tanto el margen de ganancia es igual a:

Margen= $18-2+24-112+24-2+85 = 11\text{dB}$

El cual es suficiente para realizar el enlace.

El cálculo del margen de ganancia fue realizado escogiendo los siguientes equipos: antena direccional de grilla de 24dBi, un transmisor/receptor (Linksys WRT54GL) con potencia de 18dB y con sensibilidad de -85dB, estos dispositivos son básicos y con fines de demostrar que el enlace propuesto es factible. (Referirse al anexo B)

ZONAS DE FRESNEL [23]: Este concepto se refiere principalmente a que una onda electromagnética que viaja de un punto A a un punto B en línea recta tiene un figura de ovalo y se debe tener en cuenta los obstáculos que se encuentran en medio de dicho viaje, es una ley general que “por lo menos el 60% de la primera zona de Fresnel se encuentre despejada para asegurar la factibilidad del enlace”.

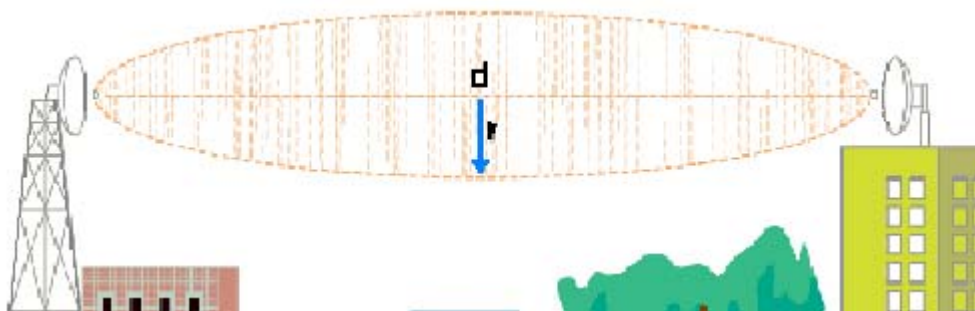


Figura 3.20 Zona de Fresnel

El radio de esta primera zona de fresnel se la puede calcular de la siguiente forma;

$$r = 17,32 * \sqrt{(d/4f)}$$

3.14

r: radio en metros

f: frecuencia en GHz

d: distancia en Km

Cálculo:

d=4Km

f=2.4Ghz

$$r = 17,32 * \sqrt{3/4 * 2.4} = 9.68m$$

(Referirse al anexo C)

LINEA DE VISTA: Es la línea que une al transmisor con el receptor y que a nuestra vista puede o no ser visible dependiendo de la distancia de los puntos, en ocasiones es necesario utilizar instrumentos como binoculares de alta potencia para lograr observarla.

TRAYECTORIA MÚLTIPLE: Este efecto se refiere a los posibles obstáculos que se pueden encontrar en la trayectoria de propagación y que pueden ocasionar que la onda electromagnética se disperse de diferentes formas, en ocasiones puede mejorar la recepción puesto que la señal al chocar con un obstáculo puede reflectarse y mejorar su llegada dependiendo de la potencia de la señal.

3.2.3 Indicaciones sobre implementación

Cuando se instala una red, se está construyendo una infraestructura de la cual la gente dependerá y por lo tanto, la red debe ser confiable. En muchas instalaciones, puede ser que la gente no sepa qué tipo de equipamiento se ha instalado. Es posible que cambien de posición al radio porque les estorba, un enchufe puede ser desconectado de una regleta porque alguien más necesita esa conexión. Asegurar la seguridad física de la instalación es un asunto prioritario. Colocar el equipo fuera del camino, y limitar el acceso al mismo es el mejor medio para asegurarse de que no ocurran accidentes o se manipule el equipamiento.

Se puede incrustar tubería de PVC en las paredes de cemento para pasar el cable de una habitación a otra, evitando hacer perforaciones cada vez que tenemos que pasar un cable. Para el aislamiento, se pueden rellenar los conductos alrededor del cable con bolsas de plástico.

Los hubs, puntos de acceso o radios interiores pueden atornillarse directamente a la pared. Lo mejor es poner el equipo lo más alto posible para reducir las posibilidades de que alguien toque los dispositivos o sus cables.

Los **cables** [25] deben esconderse y atarse. Los conductos deben enterrarse aproximadamente a 30cm de profundidad (o más abajo si el suelo se congela a mayor profundidad en climas extremos). También es recomendable comprar conductos de

un calibre superior al mínimo necesario para que en el futuro otros cables que se requieran puedan pasarse por la misma tubería.

Proteja el equipo **del agua y de la humedad [25]**. En todos los casos asegúrese de que el equipo, incluida su PC, estén al menos a 30cm. del piso para evitar daños por posibles inundaciones. También intente tener una cubierta sobre el equipo, para que de esta forma el agua y la humedad no caigan sobre él.

El equipo instalado en **un mástil o torre [25]** a menudo está a salvo de los ladrones. No obstante, para disuadirlos y mantener su equipo a salvo del viento es bueno sobreestructurar estos montajes. Los equipos que se monten sobre la torre o mástil deben pintarse de colores apagados, blanco o gris mate para reflejar el sol, así como para desviar la atención, haciéndolo lucir poco interesante. También evite las antenas que se parezcan a las de televisión, porque esas pueden atraer el interés de los ladrones, mientras que una antena WiFi no va a ser de utilidad para la mayoría de ellos.

PARA EL MONTAJE



Figura 3.21 Para el Montaje

Cuando se colocan antenas en torres, es muy importante utilizar soportes separadores. Los soportes ayudan en muchas funciones incluyendo separación, alineación y protección de la antena.

Los **soportes [25]** deben ser lo suficientemente fuertes para aguantar el peso de la antena, y también mantenerla en su lugar en los días ventosos. Las antenas pueden actuar como pequeñas velas y cuando hay vientos fuertes pueden hacer mucha fuerza sobre sus montajes. Cuando se estima la resistencia al viento [29] (en las poblaciones que se va a realizar el enlace la velocidad del viento es de 13km/h [30]), se debe considerar la superficie total de la antena, así como la distancia desde el centro de la antena al punto en el que está pegada al edificio. Las antenas grandes como los platos o los paneles sectoriales de gran ganancia pueden tener una considerable carga de viento. Si se utiliza una parabólica grillada o en malla, en lugar de un plato sólido, se ayuda a reducir la carga del viento sin afectar mucho la ganancia de la antena. Los

soportes de montaje y la estructura de soporte en general deben ser sólidos, de otra forma la antena se va a desalinear con el tiempo.

La clave para lograr una alineación exitosa de las antenas en un enlace a larga distancia es la comunicación. Si se modifican muchas variables al mismo tiempo (es decir, un equipo comienza a mover la antena mientras el otro intenta tomar una lectura de la intensidad de la señal), el proceso tomará todo el día y probablemente va a terminar con las antenas desalineadas.

Una parte importante del montaje es realizar una instalación de tierra adecuada. Se persiguen dos objetivos: proveer un cortocircuito a tierra en caso de que caiga un rayo, y proveer un circuito para que la energía estática excesiva sea disipada.

3.2.4 Tabla de costos

La siguiente tabla muestra un costo aproximado de un modelo de enlace, en este caso se ha escogido los siguientes dispositivos:

- Antena direccional de grilla de 24dBi (Hiperlink)
- Conectores tipo N
- Cable LM400 (coaxial)
- Radio.- El costo de este dispositivo no puede ser comparado con el dispositivo diseñado en este proyecto por las diferencias antes ya expuestas, pero se puede integrar un dispositivo solo con fines de dar un costo global

aproximado del enlace propuesto. Por lo tanto se ha escogido una radio Linksys WRT54GL cuyas características se encuentran en la referencia [28].

Dispositivo	Cantidad	Valor Unitario	Valor Total
Antena direccional	4	\$ 90	\$ 360
Conectores tipo N	8	\$ 2,99	\$ 23,92
Cable LM400	15	\$ 1,96	\$ 29,40
Radio Linksys WRT54GL	3	\$ 123,20	\$ 369
Caja de protección	3	\$ 18	\$ 54
Mástil de 6 mts	3	\$ 60	\$ 180
Total			\$ 1.016,32

Tabla 3.6 Tabla de costos

Por razones de costos se ha optado por colocar una repetidora para así evitar utilizar un mástil demasiado alto (20mts a un valor de \$1600) que elevaría innecesariamente los costos del radio enlace.

CAPÍTULO 4

4. IMPLEMENTACIÓN PRÁCTICA

4.1. Radio software

En este capítulo será descrita la implementación del radio software y sus componentes. Los componentes que ayudaron en la implementación del sistema del radio software en términos generales son:

- El hardware que abarca los circuitos electrónicos que contiene los dispositivos lógicos programables.
- El software que abarca el programa que corre en la pc y que permite la simulación y comunicación del pc con la placa que contiene el dispositivo lógico programable.
- Y los algoritmos digitales que abarca la implementación de los diversos bloques necesarios para la realización del radio software, expresamente los abordados en el capítulo anterior. El lenguaje de programación para la realización de los algoritmos que se uso es el VHDL (Very high speed integrated circuit Hardware Description Language).

Estos componentes serán detallados más adelante en este capítulo.

4.1.1 Hardware

En esta sección se describe los dispositivos electrónicos que se usaron en el desarrollo del radio software.

4.1.2. Especificación de los dispositivos utilizados

El principal elemento de la interfaz de datos (radio software) son las tarjetas de desarrollo con FPGA EPF10K10LC84-4 fabricada por ALTERA [15]. Este dispositivo contiene el equivalente a 10000 puertas lógicas siendo parte de ellas utilizadas para implementar la memoria RAM que son 6144 bits. La opción por la utilización de un dispositivo de ALTERA fue condicionada por el hecho de que este dispositivo está disponible en el “Laboratorio de Sistemas Digitales”. La familia FLEX 10K que pertenece al FPGA EPF10K10LC84-4 tiene dentro de sus funciones memoria RAM que se necesita para implementar la LUT. A pesar de sus modestas capacidades, este dispositivo se adecuó para las necesidades de este trabajo. En la Figura. 4.1 se muestra la tarjeta FPGA.



Figura 4.1 Tarjeta FPGA EPF10K10LC84-4

4.1.2. Software

En esta sección se describe el software que se usó para la realización del trabajo de la

presente tesis. Todo el proceso de la implementación se ha realizado con el programa MAX+PLUS II proporcionado por la misma ALTERA. Este programa permite la utilización de un lenguaje de descripción de circuitos, más específicamente el lenguaje VHDL [17]. Este lenguaje fue adoptado como una norma de industria por el IEEE en 1987 y actualizado en 1993. El lenguaje VHDL tiene la posibilidad de describir circuitos lógicos complejos a través de códigos simples, y de la organización del código en niveles jerárquicos.

Una de las características más poderosas del lenguaje VHDL es la posibilidad de implementación de componentes genéricos es decir si se quiere usar un decodificador de 8 bits de entrada y un bit de salida y si más tarde se necesita de un decodificador de 16 bits de entrada y 2 bits de salida será necesario implementar otro componente que es en todo semejante al primero excepto en el número de bits. Utilizando las potencialidades del lenguaje VHDL podemos escribir un único componente genérico en el cual el número de bits es pasado como parámetro.

En 1993 fue creada una biblioteca de componentes genéricos básicos, la que fue dado el nombre de Bibliotecas de Módulos Parametrizados (LPM), que fue como una norma por la Asociación de las Industrias Electrónicas (EIA) en complemento de las bibliotecas descritas anteriormente en la IEEE en 1987 [17]. Esta biblioteca pretende ser independiente de la arquitectura y del fabricante y es compuesta por componentes de uso general tal como sumadores, memorias RAM y ROM, multiplicadores y

demás elementos útiles para el diseñador. En algunos algoritmos, que se desarrolló en el ámbito de este trabajo fueron utilizados los componentes genéricos de la biblioteca LPM.

4.1.3. Algoritmos

En esta sección del capítulo de implementación se describe uno por uno los algoritmos utilizados en código VHDL. Cada uno de los algoritmos representa un bloque del sistema del Radio Software. El código VHDL y la simulación de todos estos algoritmos se encuentra en el anexo D.

4.1.3.1 Acumulador

Fue necesario implementar un acumulador pues este es un elemento esencial para el funcionamiento del NCO. Podría haberse implementado el acumulador como parte integrante del NCO pero se optó por desarrollar un bloque independiente ya que el acumulador es un componente de uso común.

El diagrama de bloques de este algoritmo se ilustra en la Figura 4.2. Su funcionamiento como se observa en la figura depende de la salida actual que contiene la suma de la entrada actual con la salida anterior. Así el valor presente en la salida va aumentando o acumulando progresivamente en función del valor en la entrada. El código en VHDL y la simulación, se encuentra en la sección de anexo D.1.

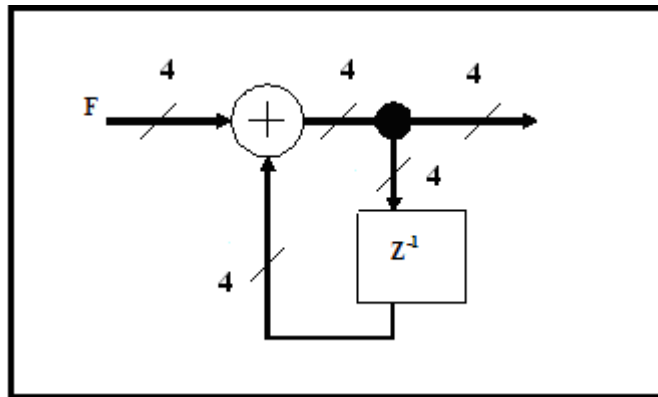


Figura 4.2 Diagrama de bloques del acumulador

La entity o la identidad de este algoritmo es decir la descripción de la interfaz que el acumulador presenta para el exterior, es la siguiente:

```

ENTITY acumul IS
    GENERIC (acc_size: natural := 4);
    PORT(acc_clk: IN std_logic;
    acc_input: IN std_logic_vector(acc_size-1 downto 0);
    acc_output: OUT std_logic_vector(acc_size-1 downto 0));
END acumul;

```

Este algoritmo permite la potencialidad de crear componentes genéricos en el lenguaje VHDL. El parámetro `acc_size` que toma un valor de 4 también puede optar cualquier otro valor determinado por el acumulador. En este caso concreto este parámetro define el número de bits de entrada y salida y por supuesto del sumador interno. La entity de este módulo declara tres puertos dos de entrada y uno de salida.

El puerto acc_input es la entrada del acumulador. Si la suma sobrepasa el valor máximo que se pueda representar con acc_size bits, el resultado será truncado con el puerto de salida acc_output. El puerto acc_clk es la entrada de la señal de reloj del acumulador. En cada transición ascendente de esta señal es alterado el valor presente en el puerto de salida acc_output que pasa a contener la suma de sí mismo con el valor presente en el puerto de entrada acc_input. En la Figura 4.3 se observa el bloque del acumulador en el editor gráfico.

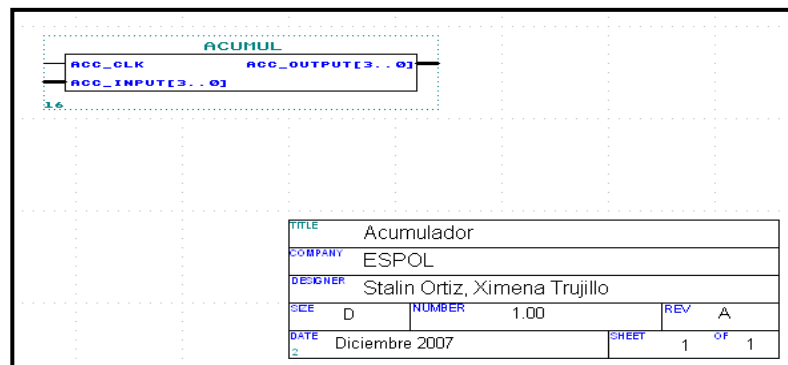


Figura 4.3 Bloque del acumulador en el editor gráfico

4.1.3.2. NCO

El método utilizado para implementar el Oscilador Controlado Numéricamente (NCO) fue el método de la acumulación de fase especificado en el capítulo 3. Para poder explicar mejor como funciona este algoritmo repetimos el diagrama de bloques del NCO en la Figura 4.4. Como se puede observar, el NCO es constituido por un acumulador seguido de un sumador y de una LUT. El código y la simulación de este algoritmo se encuentran descritos en la sección de anexo D.2.

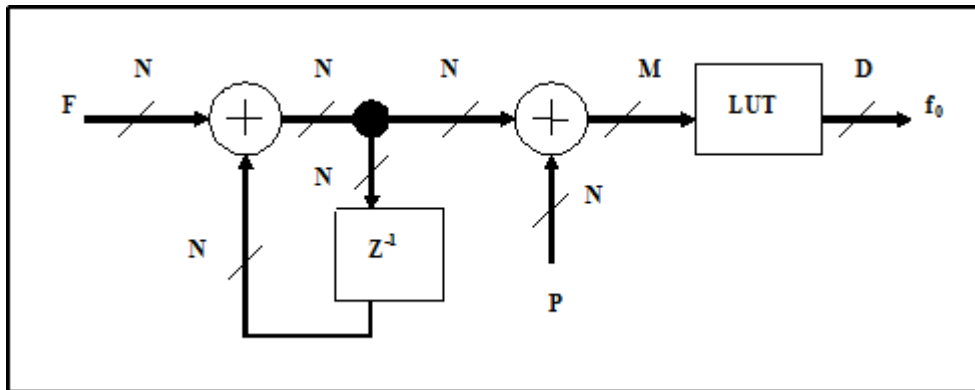


Figura 4.4 Diagrama de bloques del NCO

Este algoritmo también fue implementado como un componente genérico, como se puede confirmar en la entity siguiente:

```

ENTITY nco IS
    GENERIC (acc_size: positive := 24;
            lut_addr_size: positive := 8;
            lut_out_size: positive := 8;
            phase_size: positive := 3);
    PORT(nco_clk: IN std_logic;
         freq: IN std_logic_vector(acc_size-1 downto 0);
         phase: IN std_logic_vector(phase_size-1 downto 0);
         nco_output: OUT std_logic_vector(lut_out_size-1 downto 0));
END nco;

```

En la entity se observa que el parámetro `acc_size` define el número de bits del acumulador, como fue referido en la sección anterior. La resolución de frecuencia del

NCO está dependiendo de este parámetro. El número de bits de la línea de direcciones de la LUT es dado por `lut_addr_size` y el número de bits de salida (que define la resolución en amplitud) es dado por el parámetro `lut_out_size`. En este algoritmo no fue implementado ningún mecanismo de compresión de la LUT. La tabla de la senoide guardada en la LUT será por lo tanto dividida en $2^{\text{lut_addr_size}}$ muestras, cada una pudiendo tomar una de las $2^{\text{lut_out_size}}$ amplitudes posibles. El ultimo parámetro, `phase_size`, especifica cuantos bits serán utilizados en el control de la fase. Los bits del control de la fase son sumadas a los bits más significativos de la dirección proveniente del acumulador. El parámetro `phase_size` es igual a 3 para provocar desfases de 45° .

La entity de este módulo declara tres puertos de entrada y un puerto de salida. El puerto `nco_clock` es la entrada de la señal de reloj. El puerto `freq` es la entrada que internamente está conectado a la entrada del acumulador, permite la elección de la frecuencia de la señal de salida o portadora f_0 . Esta frecuencia, f_0 es dada por $f_0 = f_{\text{clk}} (\text{freq} / 2^{\text{acc_size}})$ en donde f_{clk} es la frecuencia de la señal de reloj es decir `nco_clock`. El puerto de entrada `phase` es el encargado de controlar la fase del NCO y tiene como parámetro a `phase_size`. Como se observa en la Tabla 4.1, el control de fase, `phase`, produce un desfase de 45° en la fase de la portadora del NCO cada vez que varía sus bits. El puerto de salida del NCO es `nco_ouput` que define la resolución en amplitud. La onda portadora que genera el NCO es un seno.

Phase	Onda portadora
000	$\text{Sin}(\Theta)$
001	$\text{Sin}(\Theta + 45^\circ)$
010	$\text{Cos}(\Theta)$
011	$\text{Sin}(\Theta + 135^\circ)$
100	$-\text{Sin}(\Theta)$
101	$\text{Sin}(\Theta - 135^\circ)$
110	$-\text{Cos}(\Theta)$
111	$\text{Sin}(\Theta - 45^\circ)$

Tabla 4.1 Variación de la onda portadora

El sumador que se implementa para sumar la salida del acumulador con el control de fase utiliza el componente `lpm_add_sub` y la LUT utiliza el componente `lpm_rom` de la biblioteca LPM. La utilización de este último componente implica que el NCO solo puede ser utilizado en dispositivos que poseen memoria interna, como es el caso de la FPGA de la familia FLEX 10K10 que se utiliza. Los contenidos de esta memoria, o sea la generación de la tabla de la onda portadora. Para generar la tabla se utilizó el método numérico unipolar sin signo. Esta tabla tiene que estar presente en un fichero llamado `seno.hex` en el formato INTEL-HEX. Para crear este fichero se tomó un pequeño programa que ya estaba descrito en la referencia [1]. Este programa llamado `intelhex.m`, corre en OCTAVE [9] o MATLAB [10], tiene como entrada un vector cuyos elementos son las muestras de la onda seno que tiene que estar comprendidos entre 0 y 255. El código de este programa está descrito en la sección de anexos D.2.

Durante la ejecución del programa es solicitado el nombre del fichero de salida donde será generado el formato INTEL-HEX correspondiente al vector de entrada. El fichero de salida deberá ser copiado para el directorio de trabajo. En la Figura 4.5 se observa el bloque del NCO en el editor gráfico.

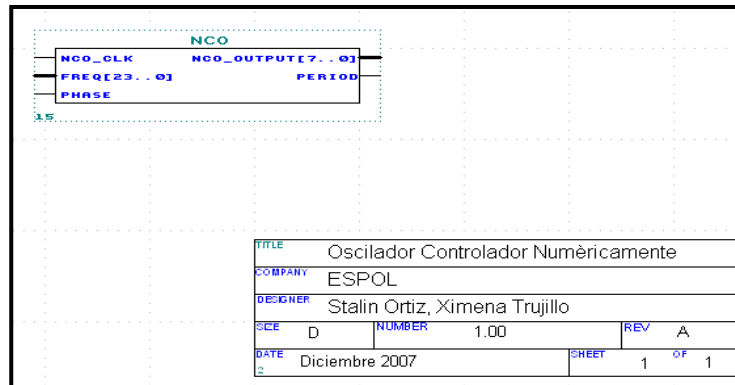


Figura 4.5 Bloque del NCO en el editor gráfico

Por último se tomó en cuenta el ruido que se puede generar a la señal de salida del NCO. El ruido introducido en el NCO es causado por errores de fase y de amplitud. Para minimizar los errores de la fase se debe reducir el rango dinámico libre de espurios (SFDR) o simplemente supresión de espurios S. La supresión de espurio está dada por $S = acc_size * 6$ en donde acc_size es el número de bits de entrada del NCO. Para reducir los errores de la amplitud del NCO se debe minimizar la relación señal al ruido (SNR). La relación señal al ruido esta dada por $SNR = nco_ouput * 6$ en donde nco_ouput es el número de bits de salida del NCO. Cada bit adicional en la fase/amplitud se produce en una mejora de la SFDR/ SNR de aproximadamente 6dB.

4.1.3 .3. Modulador QPSK

El algoritmo del modulador QPSK aquí presentado tiene como elemento principal el NCO. La modulación QPSK es una modulación digital donde la fase de la portadora es desfasada siempre que existe un cambio en la entrada (controlador de fase). El código VHDL y la simulación de este algoritmo se encuentra en la sección de anexo D.3. La entity de este algoritmo es la siguiente:

```
PORT(clk: IN std_logic;  
      datain: IN std_logic_vector(1 downto 0);  
      qpsk_output: OUT std_logic_vector(7 downto 0));  
END qpsk_mod;
```

La entity de este algoritmo describe 2 puertos de entrada y 1 de salida. El puerto clk es la entrada de la señal de reloj. El puerto de entrada datain es el controlador de fase de este algoritmo y está compuesto por 2 bits. Internamente este algoritmo utilizó la condicional WHEN que permitió controlar con datain el controlador de fase del NCO que permite el desfasamiento en la salida del modulador como se observa en la Tabla 4.2. El puerto de salida del algoritmo es qpsk_output. El puerto de salida qpsk_output define la resolución en amplitud como el puerto de salida del NCO es decir que el puerto de salida del modulador es la misma que el puerto de salida del NCO.

Datain	Controlador de fase del NCO	Desfase de la onda portadora
--------	-----------------------------	------------------------------

00	101	$\text{Sin}(\Theta - 135^\circ)$
01	111	$\text{Sin}(\Theta - 45^\circ)$
10	011	$\text{Sin}(\Theta + 135^\circ)$
11	001	$\text{Sin}(\Theta + 45^\circ)$

Tabla 4.2 Variación de la fase de la portadora del modulador QPSK por datain

En la Figura 4.6 se observa el bloque del modulador QPSK en el editor gráfico.

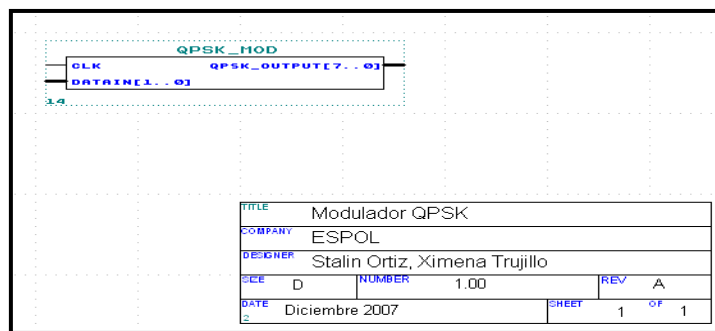


Figura 4.6 Bloque del modulador QPSK en el editor gráfico

4.1.3.4. Mezclador Complejo

El mezclador complejo es el bloque de inicio del demodulador QPSK. Este bloque fue dividido en dos partes porque el demodulador requiere mayor cantidad de recursos de la tarjeta FPGA. La tarjeta de desarrollo FPGA FLEX EPF10K10LC84-4 no tiene suficientes recursos para soportar el demodulador QPSK. Por estas razones se tuvo que utilizar 2 tarjetas FPGA FLEX EPF10K10LC84-4 para implementar el demodulador. El demodulador fue dividido en dos componentes, un componente por tarjeta, el componente en fase/canal I y en el componente en cuadratura/canal Q. No se implementó el NCO de cuadratura en el mezclador complejo por razones ya explicadas. El demodulador que se encuentra en el canal I tenemos en la entrada un

mezclador que se encarga de multiplicar la entrada FI, señal modulada, por la señal generada por un NCO como se observa el diagrama en bloque en el Figura 4.7. El diagrama en bloque del mezclador está compuesto por un multiplicador digital y un NCO. La salida del multiplicador digital tiene un número de bits igual a la suma de los bits de las 2 señales de entrada del bloque. Como se observa en el diagrama en bloque, en la salida del multiplicador digital se trunca los bits menos significativos, LSB. El motivo del truncamiento de bits es evitar la carga computacional y el consumo de recursos en los bloques posteriores del demodulador. Los bits restantes es decir los bits más significativos, MSB, tienen la misma cantidad de bits que la señal de entrada. Estos bits son la salida del bloque. El NCO se lo utiliza solamente en el mezclador para generar la onda portadora, seno. El mezclador que se encuentra en el canal Q tiene las mismas características que el mezclador que se encuentra en el canal I pero el NCO de este mezclador genera una onda desfasada 90° con respecto a la portadora, coseno, y esta señal es multiplicada a la señal de entrada FI. Los códigos en VHDL y las simulaciones de estos algoritmos se encuentran en la sección de anexos D.4.

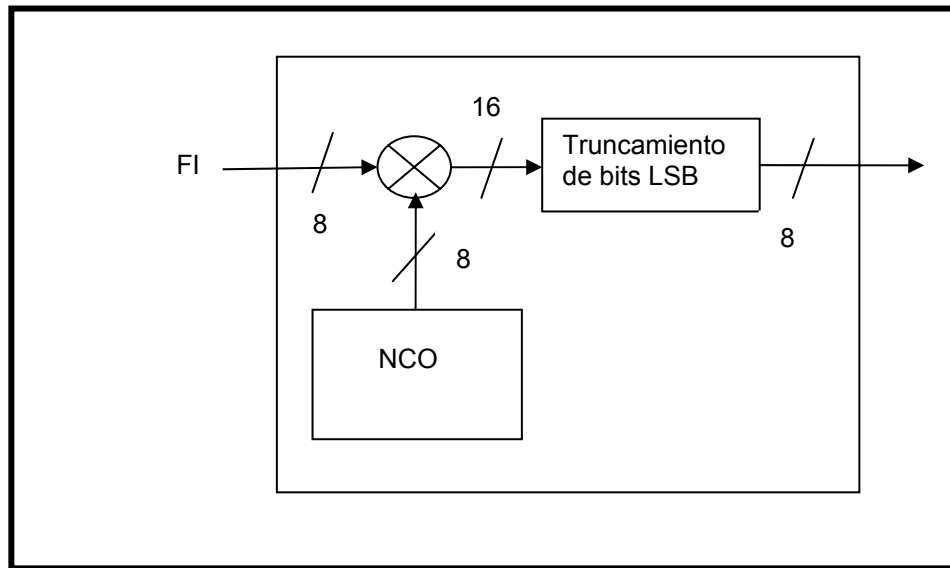


Figura 4.7 Diagrama en bloque de un mezclador

Los algoritmos descritos en esta sección también fueron implementados como componentes genéricos, como se puede confirmar en la entity de cada uno. La entity del mezclador del canal I es la siguiente:

```

ENTITY mezclador_i IS
    GENERIC ( mez_size: natural := 8;
              mez_trunc: natural := 16);
    PORT(clk: IN std_logic;
          mod_input: IN std_logic_vector(mez_size-1 downto 0);
          mez_output: OUT std_logic_vector(mez_size-1 downto 0));
END mezclador_i;

```

Y también se muestra la entity del mezclador del canal Q:

```
ENTITY mezclador_q IS
    GENERIC ( mez_size: natural := 8;
              mez_trunc: natural := 16);
    PORT(clk: IN std_logic;
          mod_input: IN std_logic_vector(mez_size-1 downto 0);
          mez_output: OUT std_logic_vector(mez_size-1 downto 0));
END mezclador_q;
```

En la entity del mezclador del canal I se observa 2 parámetros. El parámetro `mez_trunc` es la cantidad de bits en la salida del multiplicador digital. El parámetro `mez_size` define el número de bits de la señal de entrada, del NCO y de salida. En la entity también se observa 2 puertos de entrada y 1 de salida. El puerto `clk` es la entrada de la señal de reloj del mezclador. El puerto de entrada `mod_input` es donde entra la señal de IF y tiene como parámetro `mez_size`. Esta señal IF que entra en el mezclador es multiplicada por la señal que internamente se genera en este bloque por el NCO, onda portadora. En la salida de esta multiplicación digital se truncan los bits menos significativos y solo deja los bits más significativos en el puerto de salida del mezclador. El puerto de salida del mezclador es el `mez_output_i` y tiene la misma cantidad de bits que el puerto de entrada `mod_input`. En la Figura 4.8 se presenta el bloque del mezclador del canal I en el editor gráfico.

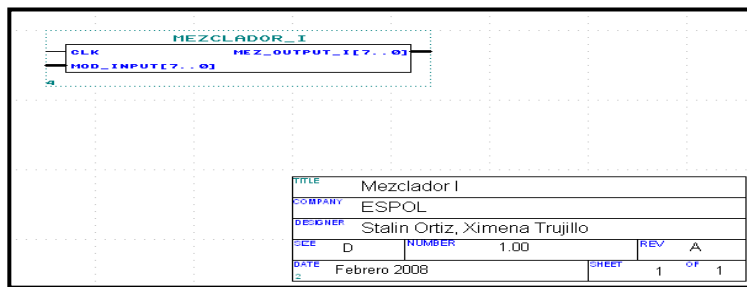


Figura 4.8 Bloque del mezclador del canal I en el editor gráfico

En la entity del mezclador del canal Q tiene la misma cantidad de parámetros y puertos que la entity del mezclador del canal I e incluso tiene los mismos nombres, los puertos y parámetros. También internamente comparte las mismas características y funciones excepto el NCO. El NCO, de esta entity del canal Q, genera una señal desfasada con 90° con respecto a la señal portadora es decir una señal coseno. En la Figura 4.9 se observa el bloque del mezclador del Q en el editor gráfico.

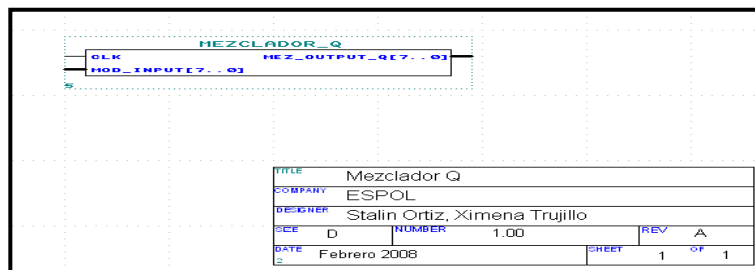


Figura 4.9 Bloque del mezclador del canal Q en el editor gráfico

4.1.3 .5. Filtro CIC

El filtro CIC(Cascaded integrator-comb o cascada integrador-peine) es un bloque que tiene la función de realizar un prefiltrado de la señal y también realiza un diezmado a la frecuencia de muestreo, provocando así menor carga computacional en el siguiente filtro. Se presenta de nuevo el diagrama en bloque del filtro CIC en la Figura 4.10. Como se puede observar, el filtro CIC está compuesto de un bloque de

acumuladores/integradores en cascada, un diezmador de velocidad (frecuencia de muestreo) R y un bloque de filtros peine/comb en cascada.

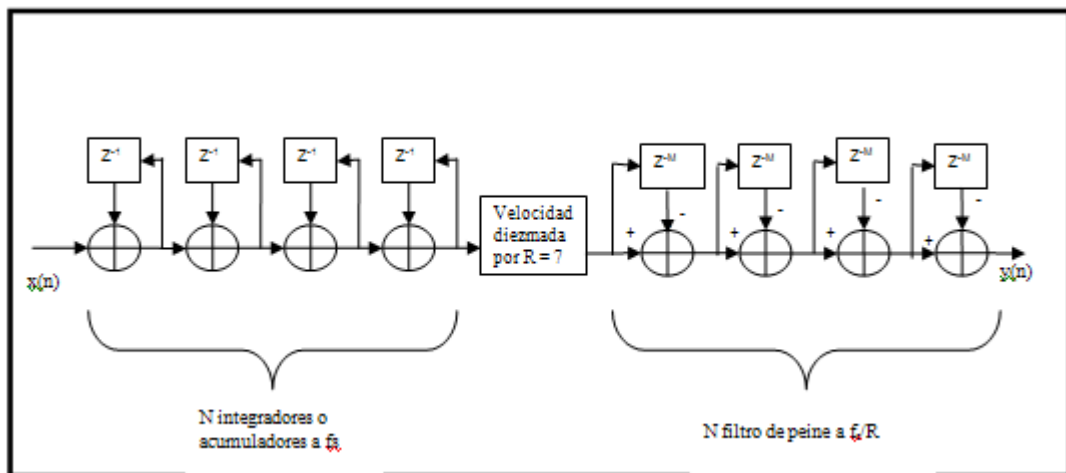


Figura 4.10 Diagrama en bloque del filtro CIC

Los parámetros de diseño de un filtro CIC son: R , N y M . R es el diezmador de velocidad, N es el tamaño que tiene el bloque de cascada de acumuladores y el bloque de cascada de filtros peine y el último parámetro M es el retardo diferencial. La función de transferencia del filtro CIC es $H(z) = (1 - z^{-RM})^N / (1 - z^{-1})^N$. Estos parámetros de diseño permiten mejorar la banda de paso de la señal, que está en banda base, y disminuir la velocidad de muestreo para la siguiente etapa. En la siguiente Figura 4.11 se observa el espectro de salida del filtro CIC para $N = 4$, $R = 7$ y $M = 1$. Estos valores de los parámetros de diseño se tomaron de la referencia [6] por tener una respuesta positiva en la banda de paso en el espectro de salida de la señal en el filtro CIC. El código y la simulación de este algoritmo se encuentran en la sección de anexo D.5.

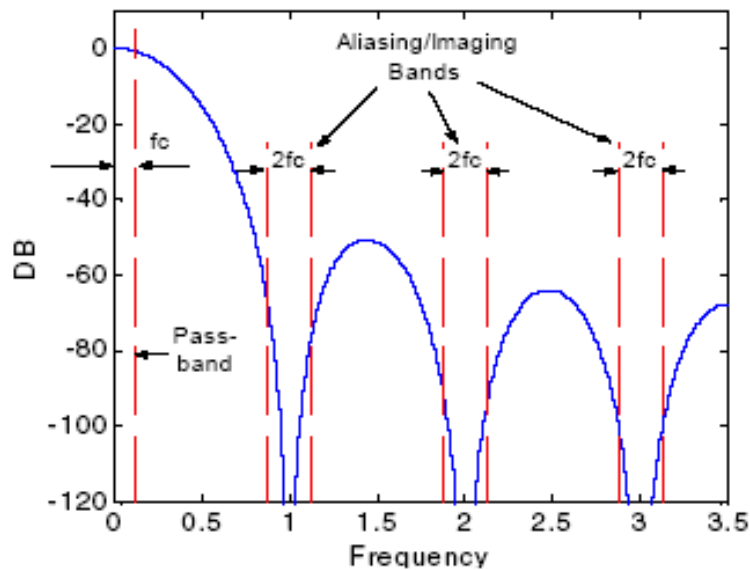


Figura 4.11 Espectro de salida del filtro CIC para $N=4$, $R=7$ y $M=1$

El algoritmo del filtro CIC también fue implementado como un componente genérico, como se puede confirmar en la entity siguiente:

```

ENTITY cic IS
    GENERIC (size: natural := 8;
             i: natural := 3);
    PORT (clk: IN std_logic;
          input: IN std_logic_vector(size-1 downto 0);
          output: OUT std_logic_vector(size-1 downto 0));
END cic;

```

En la entity se observa el parámetro size que define el número de bits del filtro CIC.

El parámetro i define el número de bits de R, de la suma recursiva y de la variable

para realizar dicha suma. R, la suma recursiva y la variable de la suma son variables internas. En la entity también se observa 2 puertos de entrada y 1 puerto de salida. El puerto clk es la entrada de la señal de reloj del sistema. El puerto input es la entrada de los datos que provienen del mezclador. El puerto input internamente está conectado a la entrada del bloque de acumuladores en cascada. Este bloque de acumuladores se halla descrito en un bloque independiente llamado integrad.vhd y se encuentra detallado en la sección de anexo D.5. El tamaño, N, que tiene este bloque es de 4 es decir 4 acumuladores conectados en cascada. Este bloque trabaja a la frecuencia de muestreo f_s . El puerto de salida del filtro CIC es output. Este puerto internamente esta conectado a la salida del bloque de filtros peine en cascada que también está descrito en un bloque independiente llamado comb.vhd y se halla en la misma sección de anexo D.5. El tamaño de este bloque también es de $N = 4$. Los filtros peine están conectados en cascada y trabaja a una frecuencia más baja f_s/R . Para la realización del bloque, comb, se utilizó 2 veces la condicional IF, en la primera si la suma recursiva es igual a cero entra en funcionamiento el bloque de filtros peine en cascada y en la segunda si el valor de la suma recursiva es igual al valor de R se encera la suma recursiva. En la Figura 4.12 se observa el bloque del filtro CIC en el editor gráfico.

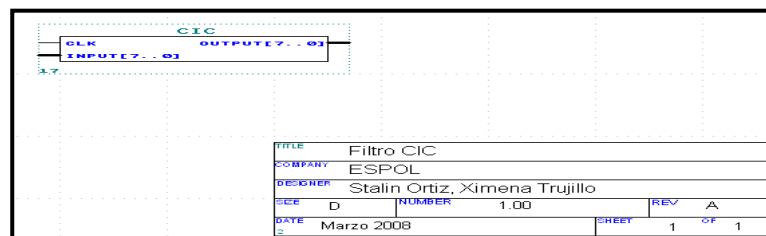


Figura 4.12 Bloque del filtro CIC en el editor gráfico

4.1.3 .6. Filtro FIR

El filtro FIR (Finite Impulse Response o Respuesta Finita al impulso) es un filtro paso-bajo que deja pasar solamente la banda de interés, banda base, y atenúa todo lo que este fuera de dicha banda, interferencias. El filtro FIR es el último bloque de la etapa de filtrado que tiene el demodulador. La función de transferencia del filtro FIR

$$\text{es } H(z) = \sum_{K=0}^{N-1} h_K z^{-k}$$

en donde la salida es la multiplicación de cada una de las entradas anteriores, z^{-k} , por su correspondiente coeficiente, h_k , y N (es igual a 8) representa el orden del filtro. El filtro FIR, de fase lineal, tiene coeficientes simétricos. Esta simetría permite reducir a la mitad el número de multiplicadores necesarios, realizando la suma de las muestras pasadas y luego multiplicando por el coeficiente simétrico, tal y como se ilustra en la Figura 4.13. La consideración de simetría logra un ahorro considerable en el dispositivo FPGA. Para el cálculo de los coeficientes del filtro FIR pasa-bajo se utilizó el método de ventana Hamming utilizando el programa MATLAB/OCTAVE. El código y la simulación de este algoritmo se encuentran descritos en la sección de anexo D.6.

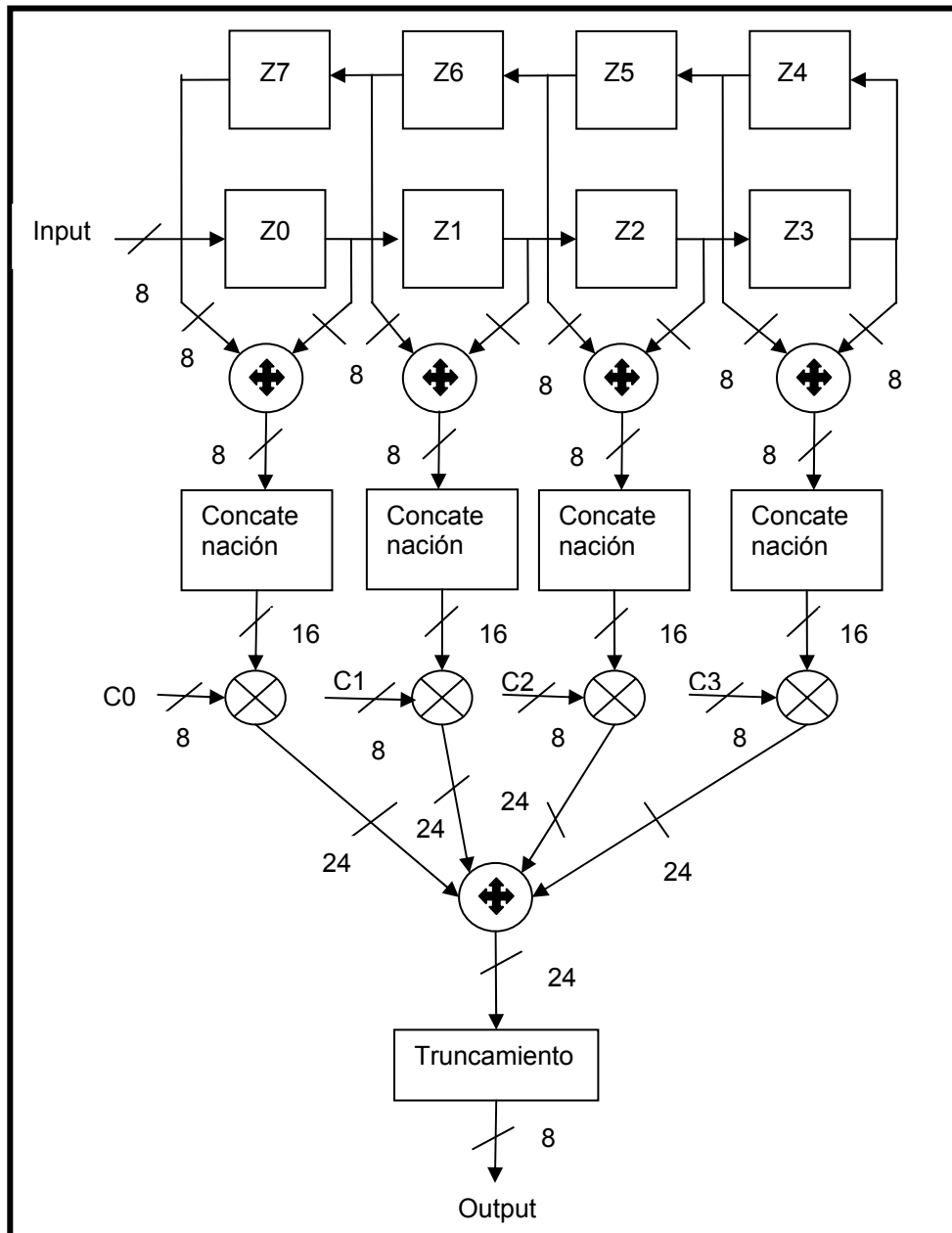


Figura 4.13 Diagrama en bloque del filtro FIR

Este algoritmo también fue implementado como un componente genérico, como se puede confirmar en la entity siguiente:

```

ENTITY fir IS
    GENERIC (fir_size: natural:= 8;
             fir_prod: natural:= 16;
             fir_sum_prod: natural:= 24);
    PORT(fir_input: IN std_logic_vector(fir_size-1 downto 0);
         fir_output: OUT std_logic_vector(fir_size-1 downto 0));
END fir;

```

En la entity se observa que el parámetro `fir_size` define el número de bits del puerto de entrada, del puerto de salida, de las entradas anteriores, de los coeficientes simétricos y de la primera suma que se realiza. El parámetro `fir_prod` define el número de bits del multiplicador. El último parámetro `fir_sum_prod` es la cantidad de bits de la segunda suma que se realiza en el filtro FIR. En la entity también se presenta el puerto de entrada `fir_input` donde entra la señal que proviene del filtro CIC. Este puerto está conectado internamente con la primera entrada anterior, el número total de entradas anteriores es 8 y se suman como se observa en la Figura 4.13. Las sumas se concatenan antes de multiplicarse con los coeficientes simétricos correspondientes que son parámetros internos del algoritmo. El resultado de los productos se suman entre sí y después se realiza un truncamiento de los bits menos significativos en la salida de la suma total. Los bits restantes se colocan en el puerto de salida `fir_output`. Para la implementación de los sumadores y de los multiplicadores se utilizó los componentes `lpm_add_sub` y `lpm_mult` de la biblioteca

LPM. Como se sumaban varias veces y se repetían el mismo procedimiento una y otra vez se decidió realizar un solo bloque sumador independiente del algoritmo, para usarlo cuantas veces sea necesario y de la misma manera se procedió para la multiplicación, estos bloques están descritos en la misma sección del anexo D.6. En la Figura 4.14 se observa el bloque del filtro FIR en el editor gráfico.

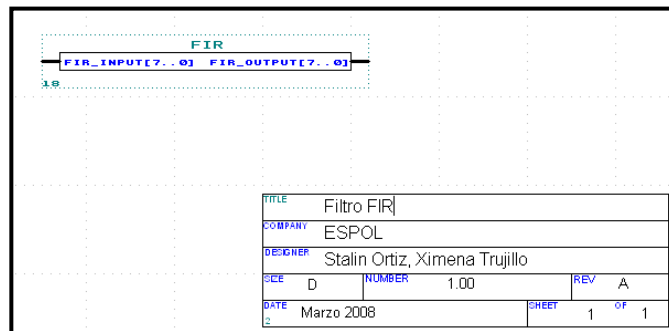


Figura 4.14 Bloque del filtro FIR en el editor gráfico

4.1.3.7. Decisor QPSK

El decisor QPSK es un simple decodificador. Este bloque es la última etapa del demodulador QPSK y se encarga de recuperar los datos que se modularon. Como se explicó anteriormente se decidió dividir el demodulador en 2 canales, I y Q, entonces para la recuperación de datos se implementó un solo bloque decisor 2 veces para cada canal. Para realizar la recuperación de los datos se utilizó la condicional IF en el algoritmo. El código y la simulación de este algoritmo se encuentran en la sección del anexo D.7.

El algoritmo del decisor QPSK también fue implementado como un componente genérico, como se puede confirmar en la entity siguiente:

```

ENTITY dec_qpsk IS
    GENERIC (size: natural:= 8);
    PORT( clk: IN std_logic;
          input: IN std_logic_vector(size-1 downto 0);
          output: OUT std_logic);
END dec_qpsk;

```

Como se observa en la entity, el parámetro size define el número de bits de la entrada del decisor o decodificador. También se presenta en la entity el puerto clk que es la entrada de la señal de reloj del bloque. El puerto de entrada input es donde entra la señal proveniente del filtro FIR. La señal proveniente del filtro FIR es un grupo de valores. Este grupo de valores varía dependiendo del dato, los que proviene del modulador. Este grupo de valores que están presente en el puerto de entrada input se decodifican utilizando el condicional IF para el respectivo dato que se desea decodificar y el dato decodificado se coloca en la salida del bloque. La salida del bloque es el puerto output. En la Figura 4.15 se observa el bloque del decisor QPSK en el editor gráfico.

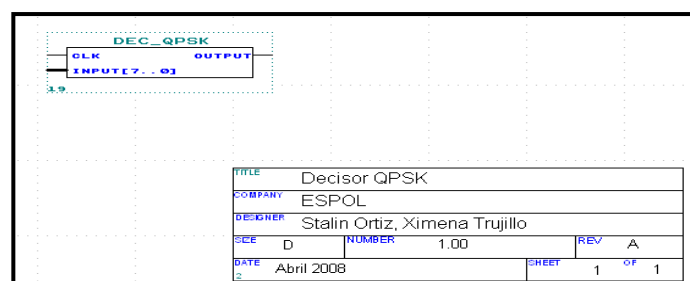


Figura 4.15 Bloque del decisor QPSK en el editor gráfico

4.1.3.8. Demodulador QPSK

En el demodulador QPSK se realiza la interconexión de todos los bloques. Como se explicó anteriormente el demodulador QPSK es más complejo en la implementación que el modulador QPSK y por tal motivo requiere mayor cantidad de recursos de la tarjeta FPGA. Para la implementación del demodulador se utilizó 2 tarjetas FPGA, una por cada canal.

En la Figura 4.16 se presenta el diagrama en bloque del demodulador QPSK del canal I. Este bloque está compuesto del mezclador I, del filtro CIC, del filtro FIR y por último el decisor QPSK. El código y la simulación de este algoritmo se encuentran en la sección del anexo D.8.

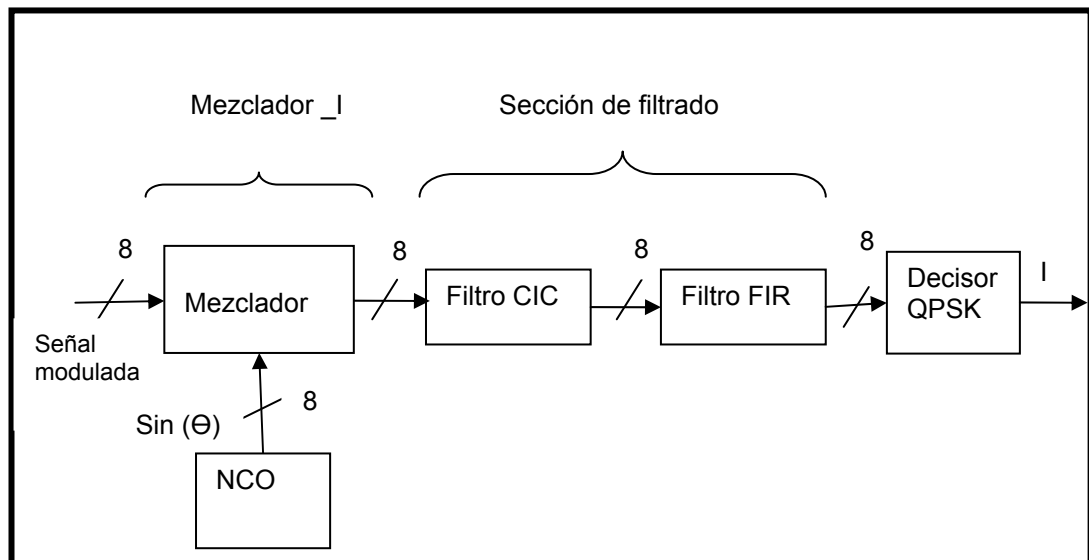


Figura 4.16 Diagrama en bloque del demodulador QPSK del canal I

Este algoritmo del demodulador QPSK del canal I también fue implementado como un componente genérico, como se puede confirmar en la entity siguiente:

```
ENTITY demod_i IS
  GENERIC (size: natural := 8);
  PORT(clk: IN std_logic;
        input: IN std_logic_vector(size-1 downto 0);
        output: OUT std_logic);
END demod_i;
```

En la entity se presenta el parámetro `size` que define el número de bits del demodulador. También se observa el puerto `clk` que es la entrada de la señal de reloj del sistema. El puerto de entrada `input` es donde entra la señal modulada, que envía el modulador QPSK a través del medio de transmisión. Este puerto está conectado internamente en la entrada del mezclador_I, como se observa en la Figura 4.18. La señal de salida en el mezclador es la señal banda base con interferencias. Esta señal va a la sección de filtrado para eliminar las interferencias y luego al bloque del decisor QPSK que es un decodificador. En la salida del decisor es un dato de un solo bit que va al puerto de salida `output` del demodulador QPSK. Este dato que se recupera en el proceso total del demodulador es el mismo dato que está presente en la entrada I del modulador QPSK. En la Figura 4.17 se presenta el bloque del

demodulador del canal I en el editor gráfico.

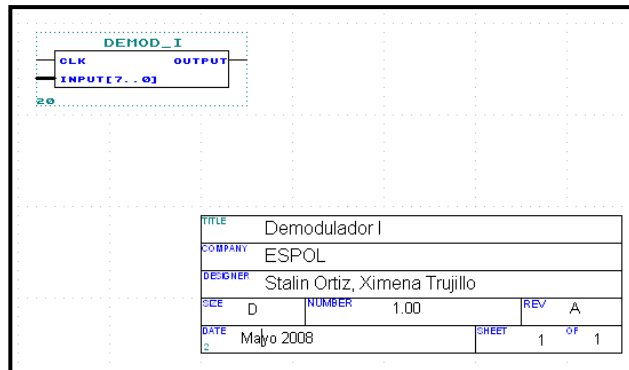


Figura 4.17 Bloque del demodulador QPSK del canal I en el editor gráfico

En la Figura 4.18 se presenta el diagrama en bloque del demodulador QPSK del canal Q. Este bloque está compuesto por los mismos elementos anteriormente explicados en el demodulador QPSK del canal I excepto por el mezclador Q. El código y la simulación de este algoritmo está también presente en la sección de anexo D.8.

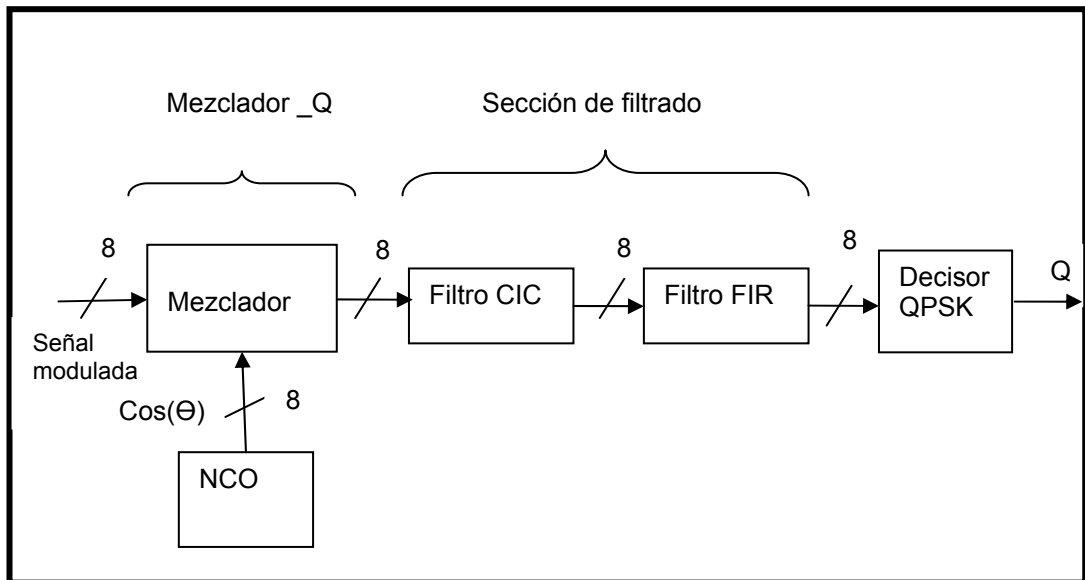


Figura 4.18 Diagrama en bloque del demodulador QPSK del canal Q

El algoritmo del demodulador QPSK del canal Q también fue implementado como un

componente genérico, como se puede confirmar en la entity siguiente:

```
ENTITY demod_q IS
    GENERIC (size: natural := 8);
    PORT(clk: IN std_logic;
          input: IN std_logic_vector(size-1 downto 0);
          output: OUT std_logic);
END demod_q;
```

La entity de este algoritmo tiene las mismas características que el anterior. El dato que se recupera en el proceso total del demodulador QPSK del canal Q es el mismo dato que está presente en la entrada Q del modulador QPSK. En la Figura 4.19 se presenta el bloque del demodulador del canal Q en el editor gráfico.

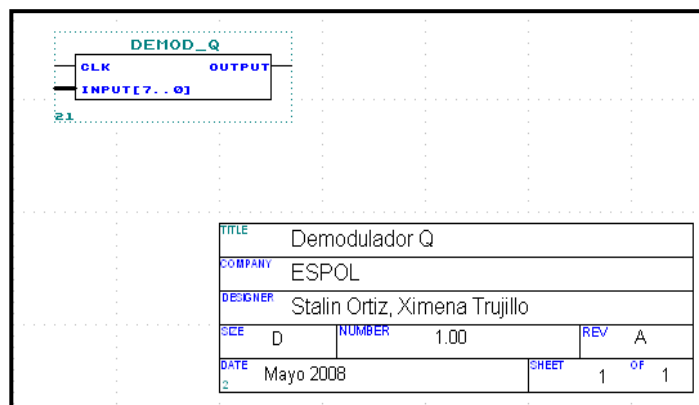


Figura 4.19 Bloque del demodulador QPSK del canal Q en el editor gráfico

CAPÍTULO 5

5. PRUEBAS DE LA IMPLEMENTACIÓN

Las pruebas de implementación se han realizado con el objetivo de demostrar que el radio software implementado funciona. Por lo tanto las pruebas de la implementación consisten en comunicar dos computadoras de forma serial (usando cables DB9), utilizando para ello la interfaz serial USART junto con las tarjetas FPGAs ya programadas y el circuito integrado MAX 232 descritos más adelante. La idea es que una de las computadoras envíe datos (utilizando el teclado) a través de la interfaz serial USART que utiliza el programa Hyperterminal para poder comunicarse con la otra computadora que recibe los datos enviados (las palabras son visibles en el monitor). La comunicación es unidireccional, los datos salientes del modulador QPSK están conectados directamente a los datos entrantes del demodulador QPSK a través de conectores de bus de datos. Una de las tarjetas contiene el modulador QPSK y uno de los componentes del demodulador (demod_i) y la otra tarjeta contiene el otro componente del demodulador (demod_q).

5.1. Radio software

El radio software que se ha realizado consta básicamente de un modulador demodulador descrito en VHDL cuyas implementaciones se presentan en este capítulo.

5.1.1. Prueba de la implementación de la modulación

Además del módulo del modulador QPSK se ha utilizado un registro de corrimiento de dos bits [4] implementado con flip-flop J K, conocido como separador de bits. En la Figura 5.1 se observa el modulador con el separador de bits en el editor gráfico.

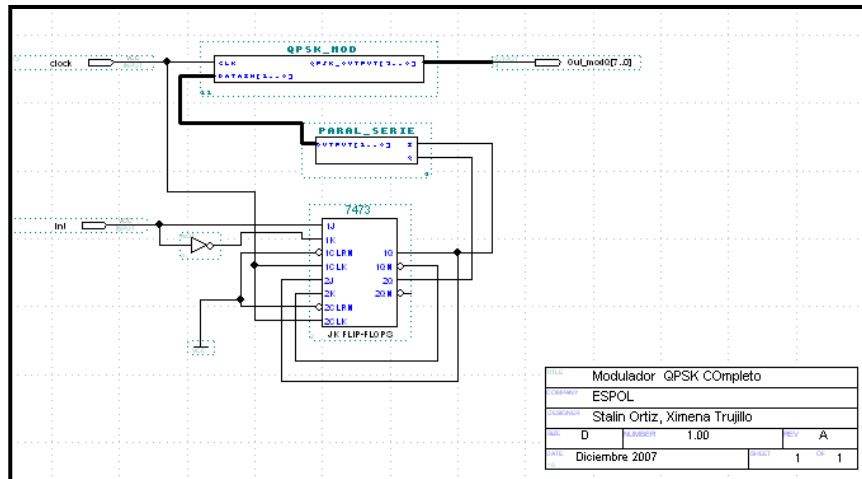


Figura 5.1 Modulador QPSK implementado

5.1.2. Resultados obtenidos de la modulación

En la Figura 5.2 se muestra la simulación del modulador QPSK implementado en la tarjeta FPGA EPF10K10LC84-4.

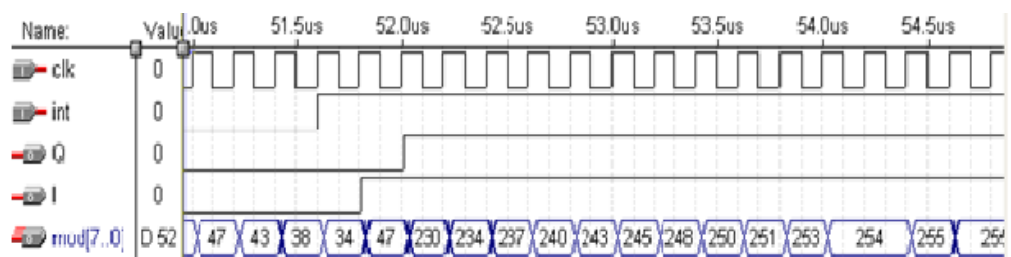


Figura 5.2 Simulación del modulador implementado QPSK

Como se observa en la Figura 5.2, el puerto de entrada clk es la señal de reloj del sistema. La señal serial que se desea modular es la entrada int y con el separador de bits se divide esta señal en las entradas I y Q del modulador QPSK. El puerto de salida del modulador es mod. Este puerto esta compuesto de 8 bits y forma la resolución en amplitud de la onda sinusoidal. La fase de salida de la señal modulada depende de las entradas (I y Q).

5.1.3. Prueba de la implementación de la demodulación

En la Figura 5.3 se observa la implementación del demodulador QPSK en el editor gráfico. Este demodulador esta compuesto por los bloques demod_i y demod_q. Cada bloque fue implementado en una tarjeta FPGA.

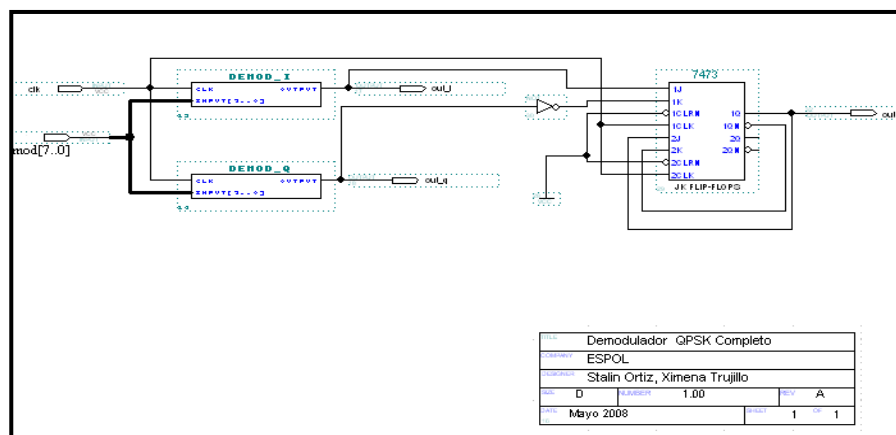


Figura 5.3 Demodulador QPSK implementado

5.1.4. Resultados obtenidos de la demodulación

En la Figura 5.4 se muestra la simulación del demodulador QPSK implementado en las tarjetas FPGAs EPF10K10LC84-4.

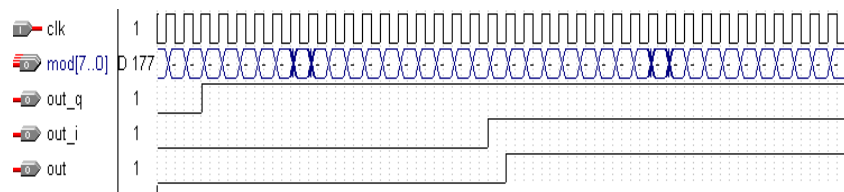


Figura 5.4 Simulación del demodulador QPSK implementado

Como se observa en la Figura 5.4, el puerto clk es la entrada de la señal de reloj del demodulador. La señal modulada que proviene del modulador ingresa en el puerto de entrada mod, de cada bloque demodulador. Esta señal es procesada en cada uno de los bloques demoduladores obteniendo una salida out_i en el bloque demod_i y en el bloque demod_q se tendrá la salida out_q. Las señales de salidas de cada bloque demodulador son colocadas en la entrada de un registro de corrimiento de dos bits para recuperar la señal serial que se moduló en un inicio y esta señal es colocada en el puerto de salida out.

En la Figura 5.5 se muestra el proceso total de la simulación del sistema radio software. Como se puede observar en el gráfico las señales del modulador QPSK y del demodulador QPSK son iguales. Hay un tiempo de retardo que tienen las señales procesadas en el demodulador que normalmente se llama tiempo de estabilización del sistema. Con esta simulación se quiso demostrar que el sistema implementado funcionó bien en las pruebas realizadas en el laboratorio de Sistema Digitales.

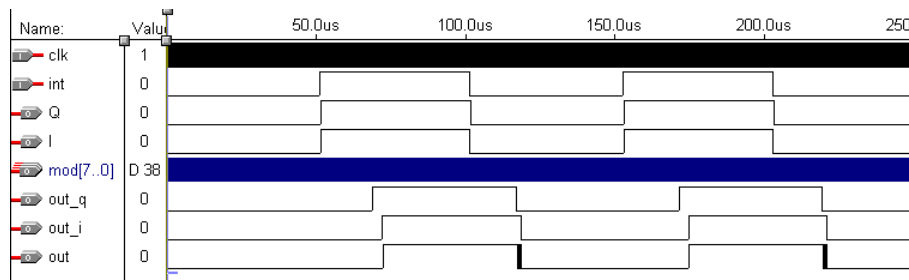


Figura 5.5 Simulación del sistema radio software

Pruebas Realizadas

Con fines de demostrar que el radio software propuesto funciona correctamente se ha decidido utilizar el programa Hyperterminal como se muestra en la Figura 5.6. Este programa utiliza el USART (Universal Synchronous Asynchronous Receiver Transmitter), que funciona en este caso en modo asíncrono. Esta interfaz utiliza el protocolo de comunicación serial RS 232 cuyos voltajes de entrada y salida oscilan entre + 12 y – 12 voltios.

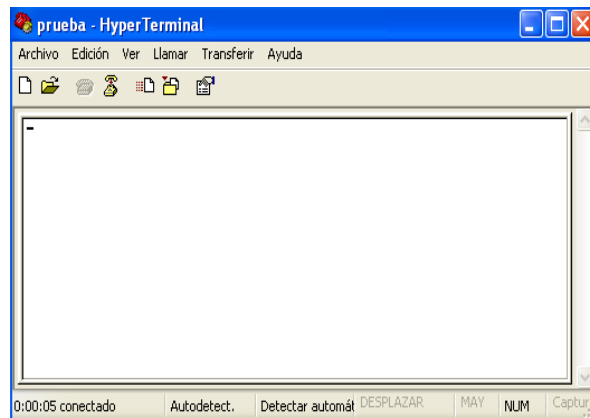


Figura 5.6 Hyper Terminal

Para poder realizar la conversión de esta interfaz a voltajes TTL que maneja la tarjeta FPGA se ha utilizado el dispositivo MAX 232. En la Figura 5.7 se observa el dispositivo y su respectiva conexión.

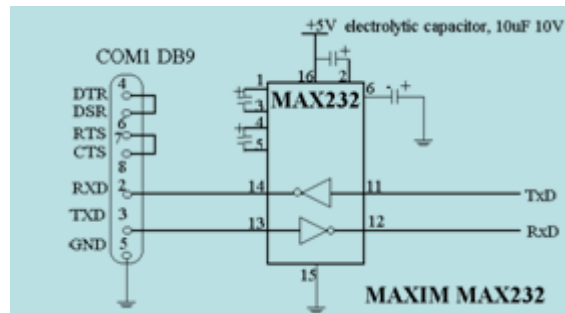


Figura 5.7 Conexión del MAX 232

Con todos estos elementos incluyendo el radio software implementado en las tarjetas FPGAs, como se indica en la Figura 5.8, se ha podido realizar la comunicación entre dos computadoras teniendo un sistema de comunicación completo. En la sección del anexo E se detalla los elementos utilizados en esta prueba.

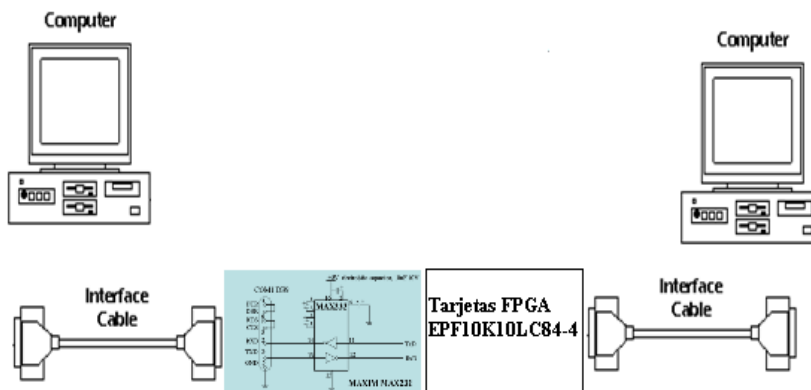


Figura 5.8 Sistema de comunicación completo

En la Figura 5.9 se muestra el esquema eléctrico de la implementación. Como se observa en el gráfico se encuentran todas las conexiones de los dispositivos que se usaron en la implementación para realizar la prueba entre las dos computadoras.

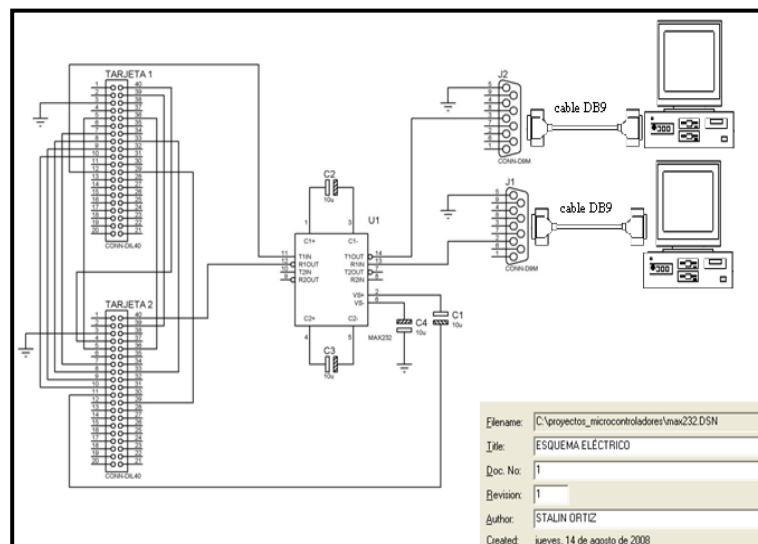


Figura 5.9 Esquema eléctrico de la implementación

El radio software tiene una señal de reloj de 8 Mhz y con los valores presentes en el código VHDL la frecuencia de portadora es 128 veces menor que la frecuencia del reloj, o sea 62500 Hz. El sistema esta sincronizado porque no se usa ningún tipo de protocolo. El sistema trabaja entre 110 hasta 4800 Baudios sin tener complicaciones en la trasmisión.

En la figura 5.10 se muestra el circuito impreso de la implementación, con dos conectores de bus de datos de 40 pines que se utilizan para conectar el circuito impreso con las tarjetas FPGA, con dos conectores DB9 que se utilizan para conectar las computadoras con el dispositivo MAX 232, finalmente se tiene el dispositivo MAX 232 con sus respectivos capacitores.

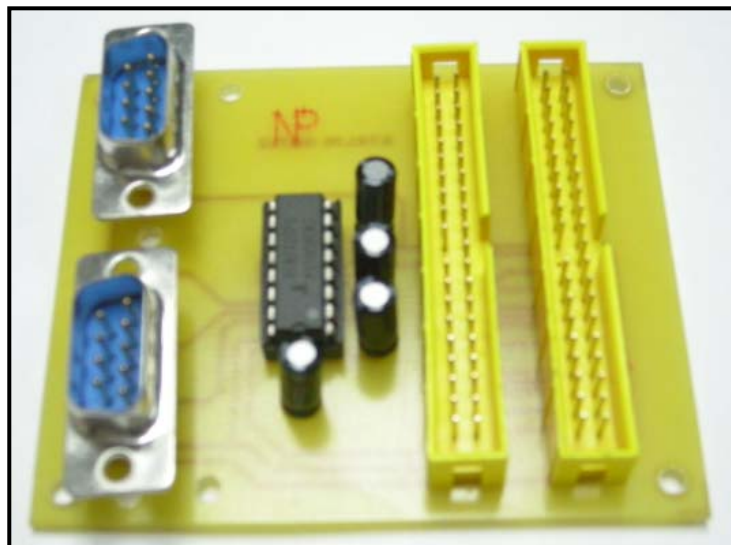


Figura 5.10 Circuito impreso

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

El presente trabajo se ejecutó con la finalidad de iniciar una solución para las necesidades de comunicación de dos poblaciones rurales y el diseño de una interfaz de datos para la comunicación.

- Después de realizar los estudios necesarios se puede concluir que es viable un sistema de comunicación entre dichas poblaciones, así pues se ha demostrado que el enlace con los dispositivos escogidos puede ser implementado en el momento que las poblaciones cuenten con los recursos requeridos.
- Por otra parte la interfaz de datos propuesta es un radio software con nuevas tecnologías (FPGAs) que proporcionan versatilidad en el momento de la programación, permitiendo cambiar la programación del dispositivo cuantas veces sea necesaria sin cambiar el hardware. Esta propiedad proporciona una adaptación a los requerimientos de los usuarios ahorrando recursos ya que se han utilizado simples algoritmos para resolver problemas complejos en comunicaciones.
- Es necesario aclarar que el sistema ha sido probado con fines académicos, debido a las limitaciones que se han tenido con los equipos disponibles para

su desarrollo, por tal motivo no se ha podido terminar con la etapa de RF en el sistema.

- Con fines de de ahorrar recursos en la tarjeta FPGA EPF10K10LC84-4 se decidió implementar ciertos procedimientos como:
 - Unir los bloques que conforman el modulador y el demodulador QPSK con código VHDL en vez de utilizar el editor gráfico
 - Se utilizó un controlador de fase en el NCO permitiendo desfasar la onda portadora, seno, para generar las distintas fases de salida en el modulador y en el demodulador en el canal I y en el canal Q se utilizó en el bloque de mezclador complejo para generar la onda portadora en el canal I y la onda coseno en el canal Q. Gracias a este controlador de fase el oscilador controlado numéricamente (NCO) es un algoritmo muy versátil ya que fue utilizado en el demodulador y es el elemento principal del modulador.
 - Para la generación de la tabla de la LUT del NCO, se determinó generar una onda senoidal de 256 muestras con el fin de ahorrar recursos en las tarjetas FPGA.
 - Se determinó utilizar el truncamiento de bits.

- La frecuencia de salida del NCO depende de la señal de reloj del sistema y se comprobó que esta frecuencia es 128 veces menor que la frecuencia de reloj.

- Con los filtros digitales CIC y FIR se concluye que son más inmunes al ruido porque trabajan con señales digitales y son menos complejo que los filtros analógicos. Se verificó que el filtro CIC es un prefiltrador de la señal banda base y un diezmador de la velocidad de muestreo. Se verificó que el filtro FIR es un filtro pasa bajo y solo deja pasar la señal banda base.

RECOMENDACIONES

En el ámbito de las nuevas tecnologías la utilización de las FPGAs propone soluciones interesantes en el área de las telecomunicaciones como es el caso de este trabajo, que aún encontrándose en una etapa de desarrollo muestra una solución parcial para un sistema de comunicación.

- Este sistema puede ser mejorado utilizando otros algoritmos que permitan la implementación de un generador de secuencias pseudo-aleatorias que permitiría, en conjunto con el NCO, la creación de un modulador con desfasamiento espectral de secuencia directa o de salto en frecuencia sin mayor dificultad. También se puede implementarse algoritmos de codificación de canal tal como bloques de detección y corrección de errores, incluso se puede implementar algoritmos para otros tipos de modulación y demodulación tanto digitales como analógicas.
- Si se deseará implementar el sistema de comunicaciones propuesto en las poblaciones se hace necesaria la utilización de otros elementos como: una tarjeta FPGA con mayores recursos recomendaríamos una tarjeta que contenga el siguiente circuito integrado EPF10K70RC 240-4, convertidores análogo digital y viceversa, amplificadores de frecuencia y de potencia.
- Es importante continuar con estudios y proyectos en el área de las telecomunicaciones basados en FPGAs que permitan reducir costos en un futuro. Puesto que en estos momentos el proyecto propuesto representa un

mayor costo que los dispositivos existentes en el mercado.

- Existen instituciones a las cuales los médicos de las población han pensado en recurrir para poder obtener el financiamiento necesario para este proyecto, como por ejemplo: al mismo Seguro Social Campesino, a la prefectura de la provincia de Chimborazo, y organizaciones no gubernamentales como: PDA y Misión Mundial. Por nuestra parte hemos recomendado al CYCIT y VLIR como instituciones que podrían proporcionar recursos para la culminación total y futuro desarrollo del sistema.

BIBLIOGRAFÍA

- [1] Implementação de Sistemas de Telecomunicações Utilizando Dispositivos Lógicos Programáveis, Universidad de Aveiro, <http://www.nocturno.org/ruben/mestrado/mestrado.ps.gz>
- [2] Diseño e implementación con FPGA de un demodulador para comunicaciones Digitales, Universidad Politécnica de Cataluña, <http://upcommons.upc.edu/pfc/bitstream/2099.1/3592/1/54608-1.pdf>
- [3] Usando un PIC para la Generación de Tonos de Audio, Universidad de Oriente, <http://www.bolivar.udo.edu.ve/microinternet/articulos/tonos.pdf>
- [4] Wayne Tomasi, Sistemas de Comunicaciones Electrónicas, Pearson Educación, 2^{da} edición 1996.
- [5] CIC filter introduction, Matthew P. Donadio, <http://users.snip.net/~donadio/cic.pdf>
- [6] Cascaded Integrator-Comb (CIC) filter, <http://www.xilinx.com/ipcenter>
- [7] Diseño e implementación de un receptor digital de radio, Universidad Técnica Federico Santa María, <http://ftp.bib.utfsm.cl/Mem/M621381536B465.pdf>
- [8] Implementing FIR Filters in FLEX Devices, <http://www.altera.com/literature/an/an073.pdf>
- [9] Octave, <http://www.gnu.org/software/octave>
- [10] Matlab, <http://www.mathworks.com>
- [11] Procesador digital de señal (dsp), <http://es.wikipedia.org/wiki>

/Procesador_digital_de_se%C3%B1al

[12] Circuito Integrado para Aplicaciones Específicas (ASIC), <http://es.wikipedia.org/wiki/ASIC>

[13] Dispositivos lógicos programables (*PLD*), http://es.wikipedia.org/wiki/Dispositivo_l%C3%B3gico_programable

[14] Field Programmable Gate Array (FPGA), <http://es.wikipedia.org/wiki/FPGA>

[15] ALTERA, <http://www.altera.com>

[16] National Semiconductor, <http://www.national.com/analog>

[17] Stephen Brown y Zvonko Vranesic, Fundamentals of Digital Logic, Mc Graw Hill Higher Education, 2^{da} edición 2005

[18] TELETRONICS, <http://www.teletronics.com/tant24sub15dbiyagui.html>

[19] TELETRONICS, <http://www.teletronics.com/tant24sector15dbi.html>

[20] Mercado libre, http://articulo.mercadolibre.com.ec/MEC-3951119-antena-hyperlink-24-dbi-24-ghz-direccional-internet-wifi-_JM

[21] TIMESMICROWAVE, <http://www.timesmicrowave.com/spanish/wireless/index.shtml>

[22] Wireless Network Products, <http://www.wirelessnetworkproducts.com/index.asp?70=228pageAction.Custom>

[23] Wireless Network Link Analysis, <http://cgi1.gbppr.org/wireless.super.main.cgi>

[24] Times Microwave Systems, <http://www.timesmicrowave.com/cgi-bin/calculate.pl>

[25] Redes inalámbricas en los países en desarrollo, Creative Commons,

<http://wndw.net>

[26] Informe Final de Ejecución Proyecto: “Redes de Área Comunal”, ESPOL,

http://www.programafrida.net/docs/informes/redes_area_comunal_espol.pdf

[27] SISTELAUSTRO, <http://www.sistelaustro.com>

[28] LINKSYS, http://www.linksys.com/servlet/Satellite?c=L_Promotion_

[C2&childpagename=ES%2FLayout&cid=1175232408196&pagename=Linksys%2FCommon%2FVisitorWrapper&lid=0819627411L00](http://www.linksys.com/servlet/Satellite?c=L_Promotion_C2&childpagename=ES%2FLayout&cid=1175232408196&pagename=Linksys%2FCommon%2FVisitorWrapper&lid=0819627411L00)

[29] VENTDEPOT, <http://www.ventdepot.com/mexico/equiposyproductos/medidores/index.html>

[30] FREEMETEO, <http://www.freemeteo.com/default.asp?pid=15&la=4&gid=3658704>

[31] CONATEL, http://www.conatel.gov.ec/site_conatel/index.php?option=com_content&view=article&id=326:espectro&catid=159:contenidos-estaticos

ANEXOS

ANEXO A
PERSONAL OPERATIVO DE NITILUISA



Dr. Héctor Lazo

Médico tratante



Sra. Carmita Benítez

Auxiliar de enfermería

PERSONAL OPERATIVO DE PULINGUI



Dra. María Fabiola Borja

Médico tratante



Lcda. María Elena Cazco

Auxiliar de enfermería

ANEXO B

Análisis de una red wireless

<http://www.gbppr.org>

Frecuencia de transmisión: 2.467000 GHz
Longitud de onda : 0.1215 metros
Potencia del transmisor : 20.000 dBm (100.000 milliwatts)
Tipo de línea del transmisor : Times Microwave LMR-400
Longitud del cable del transmisor : 10.00 metros
Pérdida en la línea del transmisor : 22.03 dB/100-metros
Cálculo de la pérdida de la línea del transmisor : 2.20 dB (0.22 dB/metro) Eficiencia de la línea del transmisor : 60.21 %
(aceptable pérdida de la línea)
Pérdida total de los conectores en el transmisor : 0.12 dB con 2 conectores
Pérdidas adicionales en la línea: 1.00 dB
Pérdidas totales en la línea : 3.33 dB
Ganancia en la antena del transmisor : 24.00 dBi
Ancho de haz en la antena transmisora: 29.16 °
Potencia total RF de la antena: 26.67 dBm
Altura de la antena transmisora : 5.00 metros
Altura de la tierra en el punto transmisor : 3344.00 meters
Altura total del transmisor: 3349.00 meters

Tipo de línea del receptor : Times Microwave LMR-400
Longitud del cable del receptor: 10.00 metros (32.81 pies)
Pérdida en la línea del receptor : 22.03 dB/100-metros
Cálculo de la pérdida de la línea del receptor : 2.20 dB (0.22 dB/metro)
Eficiencia de la línea del receptor : 60.21 % (aceptable pérdida de la línea)
Pérdida en la línea receptora: 2.20 dB
Ganancia de la antena receptora: 24.00 dBi
Ancho de haz en la antena transmisora : 29.16 °
Altura de la antena : 5.00 metros
Altura de la tierra en el punto receptor: 3228.00 metros
Altura total del receptor: 3233.00 metros

Nombre del lugar de transmisión : Pulingui
Latitud del transmisor: 1.565278 (01° 33' 55.00")
Longitud del transmisor : 78.751389 (078° 45' 5.00")
Nombre del lugar de recepción : Nitiluisa
Latitud del receptor : 1.590833 (01° 35' 27.00")
Longitud del receptor : 78.759722 (078° 45' 35.00")
Orientación del transmisor al receptor: 18.05 ° Este al verdadero norte
Orientación del receptor al transmisor: 198.05 ° Este al verdadero

norte

Distancia Total: 2.98 kilometros

Pérdidas en el espacio libre : 109.78 dB

Estimado de la pérdida en el área: 144.42 dB

Pérdidas en caso de precipitación : 0.021 dB

Pérdidas en caso de humedad: 0.000 dB

Pérdidas por oxígeno : 0.018 dB

Pérdidas totales en el espacio libre : 109.82 dB

Pérdidas totales en el área urbana: 144.46 dB

Potencia de radiación isotrópica efectiva(EIRP) : 32.673 dBm

Thermal Noise Free Space Fade Margin : -65.35 dB

Ideal margen en espacio libre para este clima: 7.98 dB

Margen del espacio libre: -65.35 dB (Urban : -99.99 dB)

Radio de la tierra(K Factor) : 4/3

Factor climático : 1.02

Temperatura promedio anual : 15.00 ° C

Máxima distancia de comunicación: 36.05 kilometros

ANEXO C

<http://www.laserwifi.com/caculadora.htm>

Calculadora de WIFI

Calcule la potencia de salida de 802,11

Utilice esta calculadora para caminar a través de todos los factores que componen su potencia total de salida. La potencia total de salida es la potencia total de salida de su sistema inalámbrico y es la suma de:

- Poder del Transmisor del Radio
- -Menos - cables y perdidas de conectores
- Ganancia de la Antena

Introduzca un parámetro en cada columna (o dejar en blanco), en función de lo que sabe y la calculadora suministra las conversiones adecuada automáticamente. Hasta 2 decimales pueden introducirse como 0.xx o .xx.

Nota: Pérdida de conector es generalmente pequeña a no ser que tengo un montón de empates en ese caso usted está probablemente en problemas de todos modos, o el cable es muy corto. Deje la línea en blanco si usted se siente perezoso.

Radio y Antena		
Poder Transmisor <i>(Esto puede ser ubicado en la hoja de especificaciones de su sistema inalámbrico)</i>	Ganancia Antena <i>(Seleccione una antena de nuestra página de productos y entrar en la antena de ganancia)</i>	Poder
<input type="text"/> mW O <input type="text"/> dBm	<input type="text" value="24"/> dB (i)	<input type="text" value="42"/> dB
Perdida de Cable		
Propiedades del Cable (por 100 Pies o m)	Larga del Cable	
<input type="text"/> dB (100 pies) O <input type="text" value="21.7"/> dB (100 m)	<input type="text"/> Pies O <input type="text" value="15"/> metros	<input type="text" value="3.26"/> dB
Perdida Conector		
Frecuencia in MHz	No. de Conectores	
<input type="text" value="2400"/> MHz.	<input type="text" value="2"/> No.	<input type="text" value="0.31"/> dB
Budget		

dB

Pérdida de Espacio Libre

La Pérdida de Poder sobre distancia es un cálculo aproximado (asumiendo que no hay efecto FRESNEL y no hay nada en línea con la antena). Recuerde que puede haber interferencia por otro medio entre la distancia.

Ingrese la frecuencia en MHz. y la distancia en **Kilómetros (Km)** ó **Millas** y clic en el botón "Calcular" para obtener el resultado.

1 GHz = 1000 MHz

2400 MHz = 2.4 GHz.

Frecuencia **Distancia**

Resultados

Mhz.

Km.

OR

Millas

dB

Zona Fresnel

Defines cuanta distancia necesitas (Si necesitas más que una línea de vista simple) y para links de más de 3 Km (2 millas).

Ingrese la distancia total del link (en Millas o Kilómetros). Si usted tiene un **Obstáculo en la distancia** ingrésela (en Millas o Kilómetros) y la calculadora usará el punto medio para todos los cálculos. Luego ingrese la frecuencia del sistema en MHz y finalice con clic en el botón "Calcular".

Nota: La calculadora asume que las antenas se encuentran a la misma altura.

1 GHz = 1000 MHz

2400 MHz = 2.4 GHz.

La calculadora va a generar solamente el radius de la primera Zona Fresnel (al punto del obstáculo o al punto medio). El 60% (no obstáculo) radius y la altura de la curvatura de la tierra en el punto medio de la **total distancia del link**

**Total
distancia del
Link**

Km.

OR

Millas

Frecuencia

**Distancia
de
Obstáculo**

Km.

OR

Millas

1st Zona Fresnel Radius

m

at

Km.

ft

at

Millas

**60% No Obstáculo
Radius**

MHz**m****ft****Altura de Tierra
(punto medio)****Mt.****Pies**

System Performance

Esta Calculadora le va a dar 3 respuestas:

1. Si usted deja la distancia en blanco automáticamente aplica un factor de SAD (o default de 30%) a el margen de Operación y dar el máximo de distancia (en Km y Millas) en cual el margen opera.
2. Si usted ingresa la distancia, automáticamente calcula el Margen de Operación y el factor SAD.
3. Si usted ingresa la distancia pero deja RX, TX (antena ganancia) o las dos en blanco automáticamente aplica el factor SAD escogido (o un default a 30% si no esta especificado) y genera el poder de antena requerido. Si los dos están en blanco calculará la ganancia simétrica de la antena.

Para RESETEAR cualquier parámetro solamente déjalo en blanco antes de Calcular.

Notas: RX Sensitividad es siempre expresada como negativo dBm (- dBm) y es el poder de señal mas bajo que el radio opera. Siempre lo encuentra en las especificaciones del radio y normalmente en rango de -80 a -110 dBm. No adivine, averigüe este numero.

Frecuencia	Distancia	Resultados	
<input type="text" value="2400"/> MHz	<input type="text" value="4"/> Km.	<input type="text" value="112.07"/> dB	
	<input type="text"/> O		
	<input type="text"/> Millas		
TX Power	TX Cable	TX Antena	
<input type="text" value="18"/> dBm	<input type="text" value="2"/> dBm.	<input type="text" value="24"/> dB.	<input type="text" value="40"/> dB
<input type="text"/> O			
<input type="text"/> mW			
RX Sensitividad	RX Cable	RX Antena	
<input type="text" value="-85"/> dBm	<input type="text" value="2"/> dB	<input type="text" value="24"/> dB	<input type="text" value="107"/> dB
Margen	RX Poder	SAD Factor	Margen Teórico
<input type="button" value="Calcular"/>	<input type="text" value="-50.07"/> dBm	<input type="text" value="87.33"/> %	<input type="text" value="34.93"/> dB

ANEXO D

ANEXO D.1

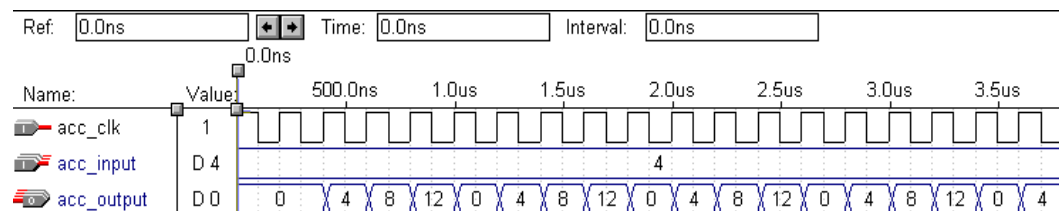
CÓDIGO VHDL DEL ACUMULADOR

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY acumul IS
GENERIC (acc_size: natural := 4);
PORT(acc_clk: IN std_logic;
      acc_input: IN std_logic_vector(acc_size-1 downto 0);
      acc_output: OUT std_logic_vector(acc_size-1 downto 0));
END acumul;

ARCHITECTURE archacumul OF acumul IS
signal acc: unsigned(acc_size-1 downto 0);
BEGIN
p1: process(acc_clk)
begin
    if (acc_clk'event and acc_clk='1') then
        acc <= acc + unsigned(acc_input);
        acc_output <= conv_std_logic_vector(acc, acc_size);
    end if;
end process;
END archacumul;
```

SIMULACIÓN



ANEXO D.2

CÓDIGO VHDL DEL NCO

```
-- Dependencias:
-- * acumul.vhd
-- * lpm_rom
-- * lpm_add_sub
--
LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY lpm;
USE lpm.lpm_components.lpm_rom;
USE lpm.lpm_components.lpm_add_sub;

ENTITY nco IS
  GENERIC (acc_size: positive := 24;
           lut_addr_size: positive := 8;
           lut_out_size: positive := 8;
           phase_size: positive := 3);
  PORT(nco_clk: IN std_logic;
        freq: IN std_logic_vector(acc_size-1 downto 0);
        phase: IN std_logic_vector(phase_size-1 downto 0);
        nco_output: OUT std_logic_vector(lut_out_size-1 downto 0));
END nco;

ARCHITECTURE archnco OF nco IS

  -- salida del acumulador
  signal addr1: std_logic_vector(acc_size-1 downto 0);
  -- salida de la dirección de fase
  signal addr2: std_logic_vector(acc_size-1 downto 0);

BEGIN

  acc: work.acumul
  GENERIC MAP (acc_size)
  PORT MAP (nco_clk, freq, addr1);
  add: lpm_add_sub
  GENERIC MAP (LPM_WIDTH => phase_size,
              LPM_DIRECTION => "ADD",
              LPM_REPRESENTATION => "UNSIGNED")
  PORT MAP (dataa => addr1(acc_size-1 downto acc_size-phase_size),
           datab => phase,
```

```

result => addr2(acc_size-1 downto acc_size-phase_size));
addr2(acc_size-phase_size-1 downto 0)
<= addr1(acc_size-phase_size-1 downto 0);

```

lut: lpm_rom

```

GENERIC MAP (lpm_width => lut_out_size,
             lpm_widthad => lut_addr_size,
             lpm_file => "seno.hex",
             lpm_address_control => "unregistered",
             lpm_outdata => "unregistered")

```

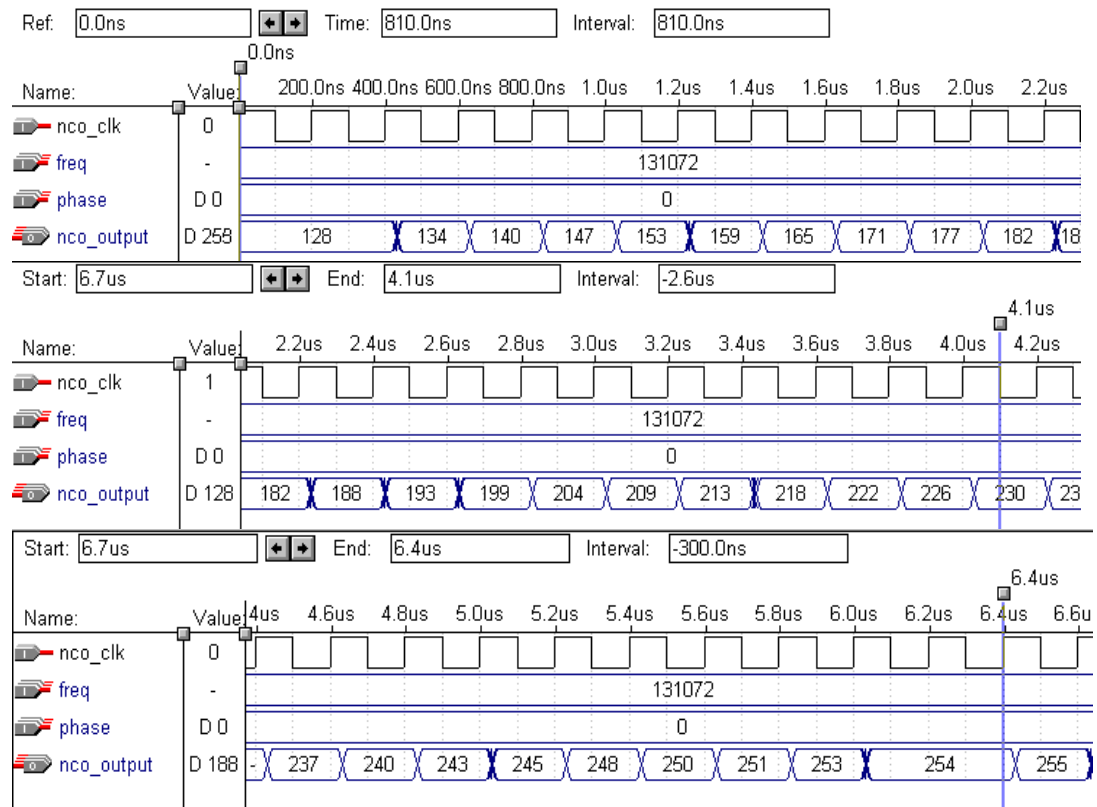
```

PORT MAP (address =>
          addr2(acc_size-1 downto (acc_size-lut_addr_size)),
          q => nco_output);

```

END archnco;

SIMULACIÓN



CÓDIGO DEL PROGRAMA EN MATLAB/OCTAVE

```
%INTELHEX(onda)
%Escribe el vector "onda" para un fichero en formato INTEL-HEX.
%Los elementos de "onda" tiene que estar comprendidos entre 0 y 255.
function intelhex(onda)
NCOLUNAS=68;
if((min(onda)<0) || (max(onda)>255))
    error("El vector debe tener valores superiores a 0 e inferiores a 255");
endif

NOMBRE=input("Nombre del fichero INTEL-HEX a gravar ?","s");
fid = fopen (NOMBRE, "w");

tamano=2*length(onda);
indice=1;

for i=1:ceil(tamano/NCOLUNAS)
    checksum=0;
    if i==ceil(tamano/NCOLUNAS)
        nbytes=rem(tamano,NCOLUNAS)/2;
    else
        nbytes=NCOLUNAS/2;
    endif
    endereco = (i-1)*NCOLUNAS/2;
    checksum = checksum + nbytes + floor(endereco/256) + rem (endereco,256);

    fprintf(fid,":");
    fprintf(fid,"%2.2x",nbytes);
    fprintf(fid,"%4.4x",endereco);
    fprintf(fid,"00");
    for j=1:nbytes
        fprintf(fid,"%2.2x",onda(indice));
        checksum = checksum + onda(indice);
        indice=indice+1;
    endfor
    fprintf(fid,"%2.2x\r\n", rem(256-rem(checksum,256),256));
endfor
fprintf(fid,":00000001FF\n");
fclose(fid);
endfunction
% FIN intelhex.m
```

ANEXO D.3

CÓDIGO VHDL DEL MODULADOR QPSK

```
-- Dependencia:
-- * nco.vhd
--
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY qpsk_mod IS
PORT(clk: IN std_logic;
      datain: IN std_logic_vector(1 downto 0);
      qpsk_output: OUT std_logic_vector(7 downto 0));
END qpsk_mod;

ARCHITECTURE archqpsk OF qpsk_mod IS
signal f: std_logic_vector(23 downto 0);
signal fase: std_logic_vector(2 downto 0);

BEGIN
f <= "000000100000000000000000"; -- f = 131072

osc: work.nco
GENERIC MAP(24,8,8,3)
PORT MAP(clk,f,fase,qpsk_output);

fase <= "101" when datain = "00" else -- salida, qpsk_output, es una onda Sin(x-135)
        "111" when datain = "01" else -- salida es una onda Sin(x-45)
        "011" when datain = "10" else -- salida es una onda Sin(x+135)
        "001"; -- salida es una onda Sin(x+45)

END archqpsk;
```

ANEXO D.4

CÓDIGO VHDL DEL MEZCLADOR DEL CANAL I

```
-- Dependencia:
-- * nco.vhd
-- Con este nco generamos una onda senoide seno --
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY mezclador_i IS
  GENERIC (mez_size: natural := 8;
          mez_trunc: natural := 16);
  PORT(clk: IN std_logic;
        mod_input: IN std_logic_vector(mez_size-1 downto 0);
        mez_output_i: OUT std_logic_vector(mez_size-1 downto 0));
END mezclador_i;

ARCHITECTURE archmezcl_i OF mezclador_i IS
  signal fase:std_logic_vector(2 downto 0);
  signal output:std_logic_vector(mez_size-1 downto 0);
  signal f: std_logic_vector(23 downto 0);

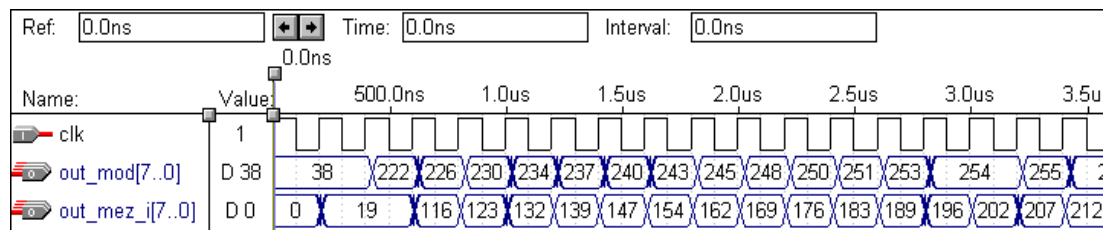
  BEGIN
  f <="000000100000000000000000"; -- f=131072
  fase <="000";

  osc: work.nco
  GENERIC MAP(24,8,8,3)
  PORT MAP(clk,f,fase,output);

  p1: process(clk)
  begin
    if (clk'event and clk='1') then
      mez_output_i <=
        conv_std_logic_vector(unsigned(mod_input)*unsigned(output),
        mez_trunc)(mez_trunc-1 downto mez_trunc-mez_size);
    end if;
  end process;

END archmezcl_i;
```


SIMULACIÓN



CÓDIGO VHDL DEL MEZCLADOR DEL Q

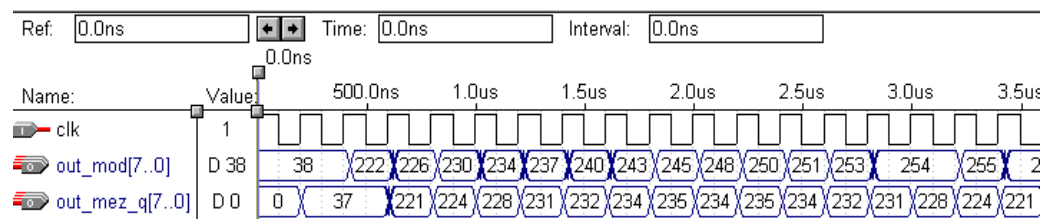
```
--  
-- Dependencia:  
-- * nco.vhd  
--  
-- Con este nco generamos una onda senoide coseno --  
--  
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
USE ieee.std_logic_arith.all;  
  
ENTITY mezclador_q IS  
  GENERIC (mez_size: natural := 8;  
          mez_trunc: natural := 16);  
  PORT (clk: IN std_logic;  
        mod_input: IN std_logic_vector(mez_size-1 downto 0);  
        mez_output_q: OUT std_logic_vector(mez_size-1 downto 0));  
END mezclador_q;  
  
ARCHITECTURE archmezcl_q OF mezclador_q IS  
  signal fase:std_logic_vector(2 downto 0);  
  signal output:std_logic_vector(mez_size-1 downto 0);  
  signal f: std_logic_vector(23 downto 0);  
  
  BEGIN  
    f <="00000010000000000000000000"; -- f=131072  
    fase <="010";  
  
    osc: work.nco  
    GENERIC MAP(24,8,8,3)  
    PORT MAP(clk,f,fase,output);  
  
    p1: process(clk)  
    begin
```

```

if (clk'event and clk='1') then
    mez_output_q <=
        conv_std_logic_vector(unsigned(mod_input)*unsigned(output),
            mez_trunc)(mez_trunc-1 downto mez_trunc-mez_size);
end if;
end process;
END archmezcl_q;

```

SIMULACIÓN



ANEXO D.5

CÓDIGO VHDL DEL FILTRO CIC

```

-- Dependencias:
-- * integrad.vhd
-- * comb.vhd
--
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY cic IS
    GENERIC (size: natural := 8;
            i: natural := 3);
    PORT(clk: IN std_logic;
         input: IN std_logic_vector(size-1 downto 0);
         output: OUT std_logic_vector(size-1 downto 0));
END cic;

ARCHITECTURE archcic OF cic IS
    signal int_output:std_logic_vector(size-1 downto 0);

BEGIN

```

```

-- bloque de acumuladores --
integrador:work.integrad
GENERIC MAP(size)
PORT MAP(clk,input,int_output);

```

```

-- bloque de filtros de peine --
peine:work.comb
GENERIC MAP(size,i)
PORT MAP(clk,int_output,output);

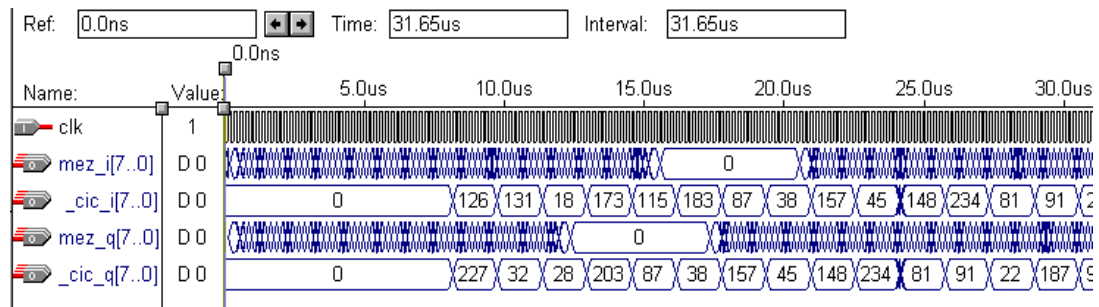
```

```

END archcic;

```

SIMULACIÓN



CÓDIGO VHDL DEL BLOQUE DE ACUMULADORES

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

```

```

ENTITY integrad IS
GENERIC (int_size: natural := 8);
PORT(int_clk: IN std_logic;
      int_input: IN std_logic_vector(int_size-1 downto 0);
      int_output: OUT std_logic_vector(int_size-1 downto 0));
END integrad;

```

```

ARCHITECTURE archintegrad OF integrad IS
signal int_sal_1: std_logic_vector(int_size-1 downto 0);
signal int_sal_2: std_logic_vector(int_size-1 downto 0);
signal int_sal_3: std_logic_vector(int_size-1 downto 0);
signal int_sal_4: std_logic_vector(int_size-1 downto 0);

```

```

BEGIN
p1: process(int_clk)
begin
if (int_clk'event and int_clk='1') then

    --primer integrador
    int_sal_1 <= conv_std_logic_vector(unsigned(int_input) +
    unsigned(int_sal_1) , int_size);

    --segundo integrador
    int_sal_2 <= conv_std_logic_vector(unsigned(int_sal_1) +
    unsigned(int_sal_2), int_size);

    --tercer integrador
    int_sal_3 <= conv_std_logic_vector(unsigned(int_sal_2) +
    unsigned(int_sal_3), int_size);

    --cuarto integrador
    int_sal_4 <= conv_std_logic_vector(unsigned(int_sal_3) +
    unsigned(int_sal_4),int_size);

    -- salida del bloque de acumuladores/integradores
    int_output <= int_sal_4 (int_size-1 downto 0);

end if;
end process;
END archintegrad;

```

CÓDIGO VHDL DEL BLOQUE DE FILTROS DE PEINE

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY comb IS
GENERIC (comb_size: natural := 8;
         comb_i: natural := 3);
PORT(comb_clk: IN std_logic;
      comb_input: IN std_logic_vector(comb_size-1 downto 0);
      comb_output: OUT std_logic_vector(comb_size-1 downto 0));
END comb;

ARCHITECTURE archcomb OF comb IS

```

```

-- variables que representan los retardos
signal z1: std_logic_vector(comb_size-1 downto 0);
signal z2: std_logic_vector(comb_size-1 downto 0);
signal z3: std_logic_vector(comb_size-1 downto 0);
signal z4: std_logic_vector(comb_size-1 downto 0);
-- variables que representan las salidas de cada filtro comb/peine
signal sal_1: std_logic_vector(comb_size-1 downto 0);
signal sal_2: std_logic_vector(comb_size-1 downto 0);
signal sal_3: std_logic_vector(comb_size-1 downto 0);
signal sal_4: std_logic_vector(comb_size-1 downto 0);
-- variable que representa el diezmador de velocidad
signal R: std_logic_vector(comb_i-1 downto 0);
-- variable que representa la suma recursiva
signal suma_i: std_logic_vector(comb_i-1 downto 0);
-- variable para la suma recursiva
signal i:std_logic_vector(comb_i-1 downto 0);

BEGIN
R <= "111";
i <= "001";
p1: process(comb_clk)
begin
if (comb_clk'event and comb_clk='1') then

    if ( suma_i="000" ) then
        --primer comb
        z1 <= comb_input(comb_size-1 downto 0);
        sal_1 <= conv_std_logic_vector(unsigned(comb_input) - unsigned(z1),
        comb_size);
        --segundo comb
        z2 <= sal_1(comb_size-1 downto 0);
        sal_2 <= conv_std_logic_vector(unsigned(sal_1) - unsigned(z2),
        comb_size);
        --tercer comb
        z3 <= sal_2(comb_size-1 downto 0);
        sal_3 <= conv_std_logic_vector(unsigned(sal_2) - unsigned(z3),
        comb_size);
        --cuarto comb
        z4 <= sal_3(comb_size-1 downto 0);
        sal_4 <= conv_std_logic_vector(unsigned(sal_3) - unsigned(z4),
        comb_size);
        -- salida del bloque de filtros de peine
        comb_output <= sal_4(comb_size-1 downto 0);
    end if;

```

```

suma_i <= conv_std_logic_vector(unsigned(suma_i) + unsigned(i),comb_i);

    if suma_i=R then
        suma_i<="000";
    end if;

end if;
end process;

END archcomb;

```

ANEXO D.6

CÓDIGO VHDL DEL FILTRO FIR

```

-- Dependencias:
-- * sum_sign.vhd
-- * prod_sign.vhd
--
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY fir IS
GENERIC (fir_size: natural:= 8;
         fir_prod: natural:= 16;
         fir_sum_prod: natural:= 24);
PORT(fir_input: IN std_logic_vector(fir_size-1 downto 0);
     fir_output: OUT std_logic_vector(fir_size-1 downto 0));
END fir;

ARCHITECTURE archfir OF fir IS
-- los coeficientes
signal c0: std_logic_vector(fir_size-1 downto 0);
signal c1: std_logic_vector(fir_size-1 downto 0);
signal c2: std_logic_vector(fir_size-1 downto 0);
signal c3: std_logic_vector(fir_size-1 downto 0);
-- los retardos
signal z0: std_logic_vector(fir_size-1 downto 0);
signal z1: std_logic_vector(fir_size-1 downto 0);
signal z2: std_logic_vector(fir_size-1 downto 0);
signal z3: std_logic_vector(fir_size-1 downto 0);
signal z4: std_logic_vector(fir_size-1 downto 0);

```

```

signal z5: std_logic_vector(fir_size-1 downto 0);
signal z6: std_logic_vector(fir_size-1 downto 0);
signal z7: std_logic_vector(fir_size-1 downto 0);
-- los productos
signal p0: std_logic_vector(fir_prod-1 downto 0);
signal p1: std_logic_vector(fir_prod-1 downto 0);
signal p2: std_logic_vector(fir_prod-1 downto 0);
signal p3: std_logic_vector(fir_prod-1 downto 0);
-- salida de las multiplicaciones
signal p0_sal: std_logic_vector(fir_sum_prod-1 downto 0);
signal p1_sal: std_logic_vector(fir_sum_prod-1 downto 0);
signal p2_sal: std_logic_vector(fir_sum_prod-1 downto 0);
signal p3_sal: std_logic_vector(fir_sum_prod-1 downto 0);
-- salida de las sumas de las muestras pasadas
signal s0: std_logic_vector(fir_size-1 downto 0);
signal s1: std_logic_vector(fir_size-1 downto 0);
signal s2: std_logic_vector(fir_size-1 downto 0);
signal s3: std_logic_vector(fir_size-1 downto 0);
-- salida de las sumas de los productos
signal sum0_sal: std_logic_vector(fir_sum_prod-1 downto 0);
signal sum1_sal: std_logic_vector(fir_sum_prod-1 downto 0);
signal sum2_sal: std_logic_vector(fir_sum_prod-1 downto 0);

```

```

BEGIN

```

```

c0 <= "00000010"; -- R=7 c0 = 0.007
c1 <= "00000110"; -- c1 = 0.024
c2 <= "00010001"; -- c2 = 0.067
c3 <= "00011011"; -- c3 = 0.104

```

```

z0 <= fir_input(fir_size-1 downto 0);
z1 <= z0(fir_size-1 downto 0);
z2 <= z1(fir_size-1 downto 0);
z3 <= z2(fir_size-1 downto 0);
z4 <= z3(fir_size-1 downto 0);
z5 <= z4(fir_size-1 downto 0);
z6 <= z5(fir_size-1 downto 0);
z7 <= z6(fir_size-1 downto 0);

```

```

-- SUMA DE LAS MUESTRAS PASADAS --
sum0: work.sum_sign
GENERIC MAP(fir_size)
PORT MAP(z0,z7,s0);
sum1: work.sum_sign

```

```

GENERIC MAP(fir_size)
PORT MAP(z1,z6,s1);
sum2: work.sum_sign
GENERIC MAP(fir_size)
PORT MAP(z2,z5,s2);
sum3: work.sum_sign
GENERIC MAP(fir_size)
PORT MAP(z3,z4,s3);

-- CONCATENACIÓN DE LAS SALIDAS DE LAS SUMAS--
p0 <= s0(fir_size-1 downto 0) & "00000000";
p1 <= s1(fir_size-1 downto 0) & "00000000";
p2 <= s2(fir_size-1 downto 0) & "00000000";
p3 <= s3(fir_size-1 downto 0) & "00000000";

-- MULTIPLICACIÓN --
prod0: work.prod_sign
GENERIC MAP(fir_prod,fir_size,fir_sum_prod)
PORT MAP(p0,c0,p0_sal);
prod1: work.prod_sign
GENERIC MAP(fir_prod,fir_size,fir_sum_prod)
PORT MAP(p1,c1,p1_sal);
prod2: work.prod_sign
GENERIC MAP(fir_prod,fir_size,fir_sum_prod)
PORT MAP(p2,c2,p2_sal);
prod3: work.prod_sign
GENERIC MAP(fir_prod,fir_size,fir_sum_prod)
PORT MAP(p3,c3,p3_sal);

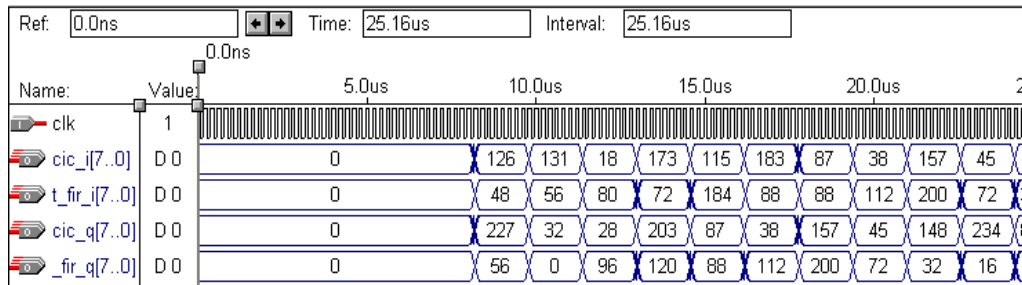
-- SUMA DE LAS SALIDAS DE LOS PRODUCTOS --
sum_prod0: work.sum_sign
GENERIC MAP(fir_sum_prod)
PORT MAP(p0_sal,p1_sal,sum0_sal);
sum_prod1: work.sum_sign
GENERIC MAP(fir_sum_prod)
PORT MAP(p2_sal,p3_sal,sum1_sal);
sum_prod2: work.sum_sign
GENERIC MAP(fir_sum_prod)
PORT MAP(sum0_sal,sum1_sal,sum2_sal);

-- SALIDA DEL FILTRO
fir_output<= sum2_sal(fir_sum_prod-1 downto fir_sum_prod-16);

END archfir;

```


SIMULACIÓN



CÓDIGO VHDL DEL SUMADOR CON SIGNO DE LA BIBLIOTECA LPM

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY lpm;
USE lpm.lpm_components.lpm_add_sub;

ENTITY sum_sign IS
    GENERIC (size: natural:= 8);
    PORT(entrada_a: IN std_logic_vector(size-1 downto 0);
          entrada_b: IN std_logic_vector(size-1 downto 0);
          salida: OUT std_logic_vector(size-1 downto 0));
END sum_sign;
ARCHITECTURE archsum_sign OF sum_sign IS

BEGIN
    sum: lpm_add_sub
    GENERIC MAP (LPM_DIRECTION => "ADD",
                 LPM_REPRESENTATION => "SIGNED",
                 LPM_WIDTH => size)
    PORT MAP (dataa => entrada_a,
              datab => entrada_b,
              result => salida);
END archsum_sign;

```

CÓDIGO VHDL DEL MULTIPLICADOR CON SIGNO DE LA BIBLIOTECA LPM

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY lpm;
USE lpm.lpm_components.lpm_mult;

```

```

ENTITY prod_sign IS
GENERIC (size_a: natural:= 8;
         size_b: natural:= 8;
         sal_size: positive := 16);
PORT(entrada_a: IN std_logic_vector(size_a-1 downto 0);
     entrada_b: IN std_logic_vector(size_b-1 downto 0);
     salida: OUT std_logic_vector(sal_size-1 downto 0));
END prod_sign;
ARCHITECTURE archprod_sign OF prod_sign IS

BEGIN
prod_sign: lpm_mult
GENERIC MAP (LPM_WIDTHA => size_a,
            LPM_WIDTHB => size_b,
            LPM_WIDTHP => sal_size,
            LPM_WIDTHS => size_a,
            LPM_REPRESENTATION => "SIGNED")
PORT MAP (dataa => entrada_a,
         datab => entrada_b,
         result => salida);
END archprod_sign;

```

ANEXO D.7

CÓDIGO VHDL DEL DECISOR QPSK

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY dec_qpsk IS
GENERIC (size: natural:= 8);
PORT( clk: IN std_logic;
     input: IN std_logic_vector(size-1 downto 0);
     output: OUT std_logic);
END dec_qpsk;

ARCHITECTURE arch OF dec_qpsk IS
signal temp: std_logic_vector(size-1 downto 0);
signal y: std_logic_vector(size-1 downto 0);

BEGIN
p1: process(input,clk)

```

```

begin
if (clk'event and clk='1') then
    if input="11111000" then -- 248
        temp <= input;
    elsif (input="11110000") then -- 240
    elsif (input="11101000") then -- 232
    elsif (input="01001000") then --72
        temp <= input;
    elsif (input="01011000") then --88
        temp <= input;
    elsif (input="01110000") then --112
        temp <= input;
    elsif (input="00010000") then --16
        output <= '1' ;
    elsif (input="11100000") then --224
    elsif (input="11010000") then --208
        temp <= input;
    elsif (input="10101000") then --168
    elsif (input="00011000") then --24
        temp <= input;
    elsif (input="01000000") then --64
    elsif (input="01100000" ) then --96
        temp <= input;
    elsif (input="01111000") then --120
        temp <= input;
    elsif (input="11000000") then --192
        output <= '0' ;
    elsif (input="11001000") then --200*
        y <= temp;
        if (y="01111000") then --120
            output <= '0' ;
        elsif y="11111000" then -- 248
        elsif (y="01110000") then --112
            output <= '1' ;
        end if;
        temp <= input;
    elsif (input="10111000") then --184*
        y <= temp;
        if (y="01011000") then --88
            output <= '1' ;
        elsif (y="11010000") then --208
        elsif (y="00011000") then --24
            output <= '0' ;
        end if;

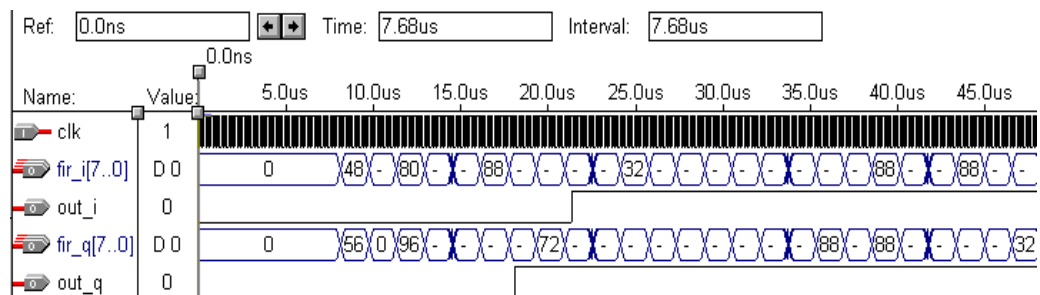
```

```

elsif (input="10011000") then --152*
    y <= temp;
    if (y="11001000") then --200
        output <= '1' ;
    elsif (y="00100000") then --32
        output <= '0' ;
    end if;
    temp <= input;
elsif (input="00100000") then --32*
    y <= temp;
    if (y="01001000") then --72
        output <= '1' ;
    elsif (y="01101000") then --104
        output <= '0' ;
    end if;
    temp <= input;
elsif (input="01101000") then --104*
    y <= temp;
    if (y="01100000" ) then --96
        output <= '0' ;
    elsif (y="10011000") then --152
        output <= '1' ;
    end if;
    temp <= input;
end if;
end if;
end process;
END arch;

```

SIMULACIÓN



ANEXO D.8

CÓDIGO VHDL DEL DEMODULADOR QPSK DEL CANAL I

```
-- Dependencias:
--
-- * mezclador_i.vhd
-- * cic.vhd
-- * fir.vhd
-- * dec_qpsk.vhd
--
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY demod_i IS
  GENERIC (size: natural := 8);
  PORT(clk: IN std_logic;
        input: IN std_logic_vector(size-1 downto 0);
        output: OUT std_logic);
END demod_i;

ARCHITECTURE arch OF demod_i IS
  signal mez_output:std_logic_vector(size-1 downto 0);
  signal cic_output:std_logic_vector(size-1 downto 0);
  signal fir_output:std_logic_vector(size-1 downto 0);

BEGIN

  -- Mezclador del canal I que tiene un oscilador,NCO, que genera
  -- la onda de la portadora seno.

  mez: work.mezclador_i
  GENERIC MAP(size,16)
  PORT MAP(clk,input,mez_output);

  -- El filtro CIC es el primer filtro de la sección de filtrado.
  -- Este filtro es el encargo del prefiltrado de la señal banda
  -- base y de reducir la frecuencia de muestreo a un factor R. Este
  -- bloque tambien reduce la carga computacional para el siguiente
  -- filtro.

  cic: work.cic
  GENERIC MAP(size,3)
  PORT MAP(clk,mez_output,cic_output);

  -- El filtro FIR es el último bloque de la sección de filtrado.
```

```
-- Este filtro es el encargado de atenuar todas las señales que
-- esten fuera de la señal banda base es decir las interferencias.
```

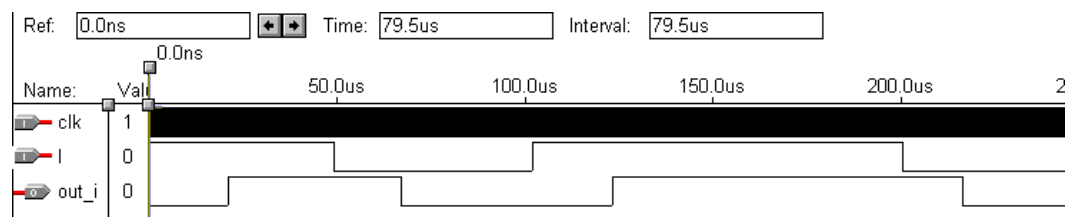
```
fir: work.fir
GENERIC MAP(size,16,24)
PORT MAP(cic_output,fir_output);
```

```
-- El decisor QPSK es un simple decodificador. Este bloque es el
-- encardo de recuperar el bit del canal I que fue anteriormente modulado.
```

```
decisor: work.dec_qpsk
GENERIC MAP(size)
PORT MAP(clk,fir_output,output);
```

```
END arch;
```

SIMULACIÓN



CÓDIGO VHDL DEL DEMODULADOR QPSK DEL CANAL Q

```
-- Dependencias:
```

```
--
```

```
-- * mezclador_q.vhd
```

```
-- * cic.vhd
```

```
-- * fir.vhd
```

```
-- * dec_qpsk.vhd
```

```
--
```

```
LIBRARY ieee;
```

```
USE ieee.std_logic_1164.all;
```

```
ENTITY demod_q IS
```

```
GENERIC (size: natural := 8);
```

```
PORT(clk: IN std_logic;
```

```
input: IN std_logic_vector(size-1 downto 0);
```

```
output: OUT std_logic);
```

```
END demod_q;
```

```

ARCHITECTURE arch OF demod_q IS
signal mez_output:std_logic_vector(size-1 downto 0);
signal cic_output:std_logic_vector(size-1 downto 0);
signal fir_output:std_logic_vector(size-1 downto 0);

BEGIN

-- Mezclador del canal Q que tiene un oscilador,NCO, que genera
-- la onda desfasada 90° con respecto a la portadora es decir un coseno.

mez: work.mezclador_q
GENERIC MAP(size,16)
PORT MAP(clk,input,mez_output);

-- El filtro CIC es el primer filtro de la sección de filtrado.
-- Este filtro es el encargo del prefiltrado de la señal banda
-- base y de reducir la frecuencia de muestreo a un factor R. Este
-- bloque tambien reduce la carga computacional para el siguiente
-- filtro.

cic: work.cic
GENERIC MAP(size,3)
PORT MAP(clk,mez_output,cic_output);

-- El filtro FIR es el ultimo bloque de la sección de filtrado.
-- Este filtro es el encargado de atenuar todas las señales que
-- esten fuera de la señal banda base es decir las interferencias.

fir: work.fir
GENERIC MAP(size,16,24)
PORT MAP(cic_output,fir_output);

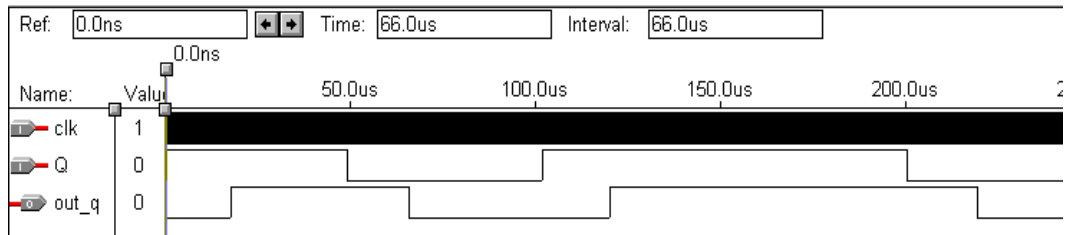
-- El decisor QPSK es un simple decodificador. Este bloque es el
-- encargo de recuperar el bit del canal Q que fue anteriormente modulado.

decisor: work.dec_qpsk
GENERIC MAP(size)
PORT MAP(clk,fir_output,output);

END arch;

```

SIMULACIÓN



ANEXO E

Elementos utilizados en la prueba:

- 1 MAX 232
- 2 conectores hembra/machos de bus de datos
- 2 conectores hembra/machos DB 9
- 3 filtros de 10 uf de 50 voltios
- 1 circuito impreso

