TRANSMISIÓN DE INFORMACIÓN USANDO LA MODULACIÓN (DSSS) ESPECTRO ENSANCHADO POR SECUENCIA DIRECTA

JOHN ALEXIS COLORADO AGUIRRE
ALBERTO LUIS RAMÍREZ HURTADO

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS
INGENIERÍA ELECTRÓNICA
PEREIRA
2011

TRANSMISIÓN DE INFORMACIÓN USANDO LA MODULACIÓN (DSSS) ESPECTRO ENSANCHADO POR SECUENCIA DIRECTA

JOHN ALEXIS COLORADO AGUIRRE
ALBERTO LUIS RAMÍREZ HURTADO

Trabajo de grado para optar por el Título de Ingeniero Electrónico

Director
Edwin Andrés Quintero Salazar
Ingeniero Electrónico
Profesor Auxiliar
Universidad Tecnológica de Pereira

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS
INGENIERÍA ELECTRÓNICA
PEREIRA
2011

CONTENIDO

LISTA DE TABLAS

LISTA DE ANEXOS

LISTA DE FIGURAS

INTRODUCCIÓN

Dada la creciente evolución en el mundo de la electrónica y en específico del campo enfatizado en esta investigación, las telecomunicaciones, es importante que la formación como ingenieros se encuentre en un proceso de actualización y de aprendizaje constantes, con miras hacia el desarrollo de nuevas tecnologías, en este caso Spread Spectrum.

Esta técnica en sus inicios era una tecnología que por su complejidad y altos costos era de uso exclusivo para proyectos militares, siendo difícil acceder a ella para utilizarla en aplicaciones comerciales; la integración a gran escala de circuitos digitales permitió colocar el costo de la tecnología de "Spread Spectrum" en un nivel realista para desarrollar aplicaciones comerciales.

En este proyecto se hace uso de la técnica antes mencionada para implementarla en un circuito integrado de gran auge comercial como lo es la FPGA[1], que por su naturaleza y funcionamiento es la ideal para emular circuitos físicos tales como compuertas lógicas, multiplexores, moduladores y demoduladores, etc.

La finalidad de este proyecto es desarrollar un sistema de comunicación aplicando los principios de la técnica Spread Spectrum, para así comprobar y verificar su correcto desempeño sobre un dispositivo de hardware reconfigurable (FPGA), dicha comunicación entre modulador y demodulador se realizará mediante un par de cobre, así se determinará la viabilidad de utilizar dicho principio en el intercambio de información.

[1] FPGA: Field Programmable Gate Array

1. DESCRIPCIÓN DEL PROYECTO

1.1 DESCRIPCIÓN DEL PROBLEMA

Cuando se desea diseñar un sistema de comunicación, uno de los principales inconvenientes es lograr la optimización en el uso del ancho de banda del canal, al igual que la energía con la que opera el sistema, siendo estos algunos de los asuntos más importantes a la hora de implementar procesos de transmisión de información. Sin embargo, en ciertos casos existen situaciones en las que es necesario, que el sistema resista a las interferencias externas y que opere sobre una banda muy ancha para que la potencia en cualquier frecuencia sea baja. Además debe proporcionar capacidad de acceso múltiple sin control externo y un canal seguro e inaccesible para oyentes no autorizados [1].

En el proceso de comunicación es la seguridad lo que genera mayor dificultad ya que esta tiene 3 criterios: integridad, confidencialidad y disponibilidad, los cuales se ven violados por las interferencias que se puedan presentar en el sistema de comunicación, algunas ocasionadas por el mismo sistema, pero en otros casos son interferencias intencionales o “jamming”, las cuales consisten en señales de alta potencia con un ancho de banda limitado que son dirigidos directamente al receptor para sabotear la comunicación [2].

Adicionalmente, el aprovechamiento del ancho de banda del canal de comunicaciones también presenta dificultades, pues por un canal solo se permite un usuario enviando y recibiendo información, siendo notoria la ineficiencia en el uso del ancho de banda presente en él.

1.2 JUSTIFICACIÓN

Spread Spectrum (espectro disperso) es una técnica de comunicación nacida durante la segunda guerra mundial que por los altos costos que acarreaba, se aplicó casi exclusivamente para objetivos militares, hasta comienzos de los años noventa [1].

La expectativa general es que comercialmente, la técnica Spread Spectrum sea cada vez más usada para la transmisión de datos. Como la potencia de emisión se difunde sobre un ancho de banda amplio, puede ser usada por encima de bandas de frecuencia existentes, sin interferir la recepción de banda angosta; razón por la cual es posible admitir más usuarios en una banda de frecuencia y se facilita la adaptación de esta técnica en la difusión de datos por la red eléctrica.

Otra de las ventajas es la seguridad de la comunicación, al fin y al cabo, la información se envía cifrada, por lo cual el ruido implícito en el sistema eléctrico no debería influir o afectar el proceso de comunicación, ni tampoco las interferencias intencionales o "jamming". En un sistema RLAN[2] con 100 usuarios que utilizan Spread Spectrum es suficiente con 1 frecuencia emisora y 100 señales-codificadoras diferentes. La información se codifica, entonces, directamente [1].

Actualmente existen demasiadas bandas para transmitir haciéndose muy complejo diseñar nuevos sistemas de comunicación sin modificar o quitar los existentes. Esta técnica permite transmitir en las bandas ya usadas por otros operadores u otro tipo de comunicaciones sin interferir la comunicación entre ellos, en otras palabras, en un mismo canal se puede enviar la información de un medio a otro e implícitamente habría una información adicional totalmente ajena a esta, la cual sería la proporcionada por la técnica Spread Spectrum.

El objetivo general de este trabajo es llevar a cabo el diseño de un transmisor y un receptor sobre una FPGA con los fundamentos de la técnica Spread Spectrum para así implementar un sistema de comunicaciones basado en este principio; concluyendo si es posible implementarlo en algunas aplicaciones, como en redes domóticas, transmisión y recepción de datos por la red eléctrica (PLC)[3], entre otras.

[2] RLAN: Radio Local Area Network.
[3] PLC: Power Line Communications.

1.3 OBJETIVOS

1.3.1 Objetivo general

Desarrollar un sistema de comunicación usando la modulación Spread Spectrum por Secuencia Directa (DSSS)[4] sobre FPGA.

1.3.2 Objetivos específicos

- Elaborar un algoritmo que permita, mediante los principios de la modulación Spread Spectrum por secuencia directa, la transmisión de datos usando una FPGA.

- Elaborar un algoritmo que permita, mediante los principios de la modulación Spread Spectrum por secuencia directa, la recepción de datos usando una FPGA.

- Establecer mecanismos de sincronismo entre transmisor y receptor a fin de conservar la coherencia de la comunicación.

- Realizar pruebas al sistema de transmisión para verificar su correcto desempeño.

[4] DSSS: Direct Sequence Spread Spectrum.

1.4 ESTADO DEL ARTE

En alguna ocasión, la tecnología de Espectro Ensanchado llegó a ser de uso exclusivo para proyectos militares, para asegurar comunicaciones confiables inmunes a la interferencia premeditada de las señales de radio así como a otros tipos de interferencia. Debido a la complejidad de la tecnología de "Spread Spectrum" en aquel tiempo era difícil acceder a ella para utilizarla en aplicaciones comerciales.

Sin embargo, a finales de los 80's, la integración a gran escala de circuitos digitales permitió colocar el costo de la tecnología de "Spread Spectrum" en un nivel realista para desarrollar aplicaciones comerciales. Al mismo tiempo la FCC[5] en los Estados Unidos autorizó la transmisión de señales de "Spread Spectrum" en la banda de 902-928 MHz sin licencia o permiso.

En la actualidad, muchos sistemas orientados a voz y datos, tanto civiles como militares emplean sistemas de espectro ensanchado, y cada vez se encuentran más aplicaciones. Una prueba de ello es que entre 1995 y 1997 se patentaron más de 1200 ideas relacionadas con el espectro ensanchado [3].

Al Hablar de transmisión de datos y de Spread Spectrum, se hace necesario mencionar el estándar IEEE[6] 802.11, ya que este define el uso de dos tipos de arquitecturas (capas física y de enlace de datos), especificando así sus normas de funcionamiento y los protocolos necesarios para la transmisión de información entre redes locales (LAN) y redes de área metropolitana (MAN) [4].

El estándar IEEE 802.11, a pesar de su reciente aparición, está penetrando en el mercado rápidamente. El secreto del éxito de esta técnica se basa principalmente en que trabaja en bandas de frecuencia que no necesitan de licencia para su utilización: ISM (Industrial, Scientific and Medical; 2,4GHz) y U-NII (Unlicensed National Information Infrastructure; 5GHz).

Inicialmente el 802.11 se pensó para redes locales inalámbricas (WLAN) de corto alcance, pensadas para entornos SOHO (Small Office – Home Office), pero la necesidad de comunicar dispositivos portátiles a velocidad de transmisión elevada, ha llevado a plantear e incluso llevar a la práctica la creación de redes inalámbricas de mayor capacidad [5].
El estándar IEEE 802.11 se divide en dos capas principales: la capa MAC (Media Access Control) y la capa física o PHY. Estas dos capas permiten hacer una

[5] FCC: Comisión Federal de Comunicaciones.
[6] IEEE: Institute of Electrical and Electronics Engineers.

separación funcional del estándar y, lo que es más importante, permite que un único protocolo de datos pueda usarse con distintos métodos de transmisión [5].
La capa física del estándar 802.11 define diferentes técnicas de transmisión. En concreto estas tres:

• FHSS (Frequency Hopping Spread Spectrum)
• DSSS (Direct Sequence Spread Spectrum)
• Infrarrojo Difuso

Por lo anterior, se encuentra entonces, que la técnica de modulación Spread Spectrum pertenece a una de las 2 capas principales del estándar IEEE 802.11, a la capa física.

1.4.1 Productos Comerciales Existentes

Los productos se dividen en el tipo de capa física que utilizan, la cual puede ser DSSS (Direct Sequence Spread Spectrum – Espectro Ensanchado por Secuencia Directa) o FHSS (Frecuency Hopping Spread Spectrum - Espectro Ensanchado por Salto en Frecuencia). Hace tiempo la mayoría de productos eran propietarios y con velocidades de 1,5Mbps y estaban pensados para aplicaciones concretas (inventarios…) y también eran bastante caros. Pero desde que apareció el estándar 802.11b asegurando la compatibilidad entre dispositivos de diferentes fabricantes se pone la tecnología Spread Spectrum en un nivel más realista. Todo esto ha hecho que muchos fabricantes proporcionen sus soluciones y se acojan a este estándar. Llevando ello a la competitividad en este tipo de productos y la consiguiente disminución de precios. Existen un sin número de dispositivos compatibles con este estándar. Se comentarán algunos dispositivos utilizados más comúnmente en el mercado actual, para proporcionar la última milla [6].

1.4.2 Ultima Milla

Dentro del ámbito de las telecomunicaciones se conoce al termino “última milla” como el tramo final, que relativamente son distancias cortas, para establecer la intercomunicación entre el cliente y su proveedor de servicios de telecomunicación. El medio para establecer la última milla puede ser guiado (Fibra Óptica) o no guiado (Inalámbrico).

Existen infinidad de posibilidades en equipos para optar y definir la última milla, a continuación algunas características de los equipos más comunes utilizados para implementar la última milla inalámbrica [6]:

- **Apple Airport**

Los dispositivos de Apple son los más baratos del mercado debido a que funcionan solo con Macintosh y que todos los equipos nuevos incluyen la antena necesaria para la comunicación (figura 1).

Figura 1. Producto Apple para transmitir inalámbricamente.

Algunas características son:
Certificado Wi-Fi
Frecuencia de funcionamiento: 2.4Ghz
Distancia máxima entre Punto Acceso y dispositivo: 150 pies (304mm) -> 45.6 metros [varía por la construcción del edificio]
IEEE 802.11HR Direct Sequence Spread Spectrum (DSSS) 11 Mbps and 5.5 Mbps
Standard IEEE 802.11 DSSS 1 and 2 Mbps standard noncondensing
Recomendado para 10 usuarios.

- **Puntos de acceso a internet**

Figura 2. Puntos para la conexiona internet.

En la figura 2.A, el teletronics 11 Mbps tiene las siguientes características:

Este equipo es compatible con la especificación 802.11b y provee la misma conectividad para redes Ethernet cableada y redes cableadas e inalámbricas.
Protocolo: IEEE 802.11b compliant Media Access
Modulación: Direct Sequence Spread Spectrum
Banda de frecuencia: 2.4 GHz.
Ancho de banda: 2.412 GHz - 2.462 GHz

En la figura 2.B, el TrendNet Bridge Inalámbrico 11/22Mbps tiene las siguientes características:

Modulación: Direct Sequence Spread Spectrum (DSSS)
Canales: 11 Canales
Rango de Transmisión: 22Mbps, 11Mbps, 5.5Mbps, 2Mbps, y 1Mbps
Frecuencia: 2.4 ~ 2.4835 GHz.

En la figura 2.C, el D-Link DI-624M tiene las siguientes características:

Protocolos: IEEE 802.11g, IEEE 802.11b, IEEE 802.3, IEEE 802.3u
Modulación: Direct Sequence Spread Spectrum (DSSS)
Rango de Transmisión: 108Mbps, 54Mbps, 48Mbps, 36Mbps, 24Mbps, 18Mbps, 12Mbps, 11Mbps, 9Mbps, 6Mbps, 5.5Mbps, 2Mbps, 1Mbps.
Frecuencia: 2.4GHz ~ 2.462GHz.

Los anteriores dispositivos son los que se ven hoy en día, como se puede apreciar, son diseños en los que se hace énfasis en la seguridad y en la velocidad de la comunicación. Todos estos dispositivos hacen uso de la técnica Spread Spectrum en Secuencia Directa, observando que una de las ventajas de esta técnica es que garantiza la seguridad en la transmisión de la información.

En el campo de las FPGA se han desarrollado muchas aplicaciones de spread spectrum, tales como transceptores inalámbricos (wifi)[7] usando FHSS (spread spectrum por saltos de frecuencia) [7], también se han hecho gran variedad de emisores y receptores RF. Adicional a esto también se han aprovechado las ventajas que tiene el DSP ya que mejora notablemente la versatilidad del sistema en aplicaciones diversas. Una de ellas es *un enlace por microondas de bajo costo utilizando spread spectrum y las técnicas del DSP*.

Los diseños hechos en hardware reconfigurable tienen muchas ventajas, como lo son la robustez del sistema, la gran variedad de aplicaciones, el ahorro de tiempo diseñando circuitos dispendiosos, costosos y con una alta probabilidad del error humano, además el uso de elementos discretos que pueden introducir datos

[7] WIFI: Wireless Fidelity.

erróneos en el sistema. Por otra parte, acarrean un costo que se hace notable al producirlos en masa, también se debe tener en cuenta que los dispositivos reprogramables pueden ser usados una cantidad innumerable de veces sin mayor esfuerzo idealizándolos cada vez más para aplicaciones de control, comunicaciones y demás labores que serian tareas arduas para diseños análogos.

2. SPREAD SPECTRUM

Un atributo importante de la técnica Spread Spectrum es que ésta puede ofrecer protección contra señales de interferencia (jamming) con potencia finita generadas externamente. La señal jamming puede consistir en ruido de ancho de banda amplio bastante potente o en una forma de onda multitonos que se dirige al receptor con el fin de interrumpir las comunicaciones. La protección contra formas de onda perturbadoras se proporciona al hacer intencionalmente que la señal que contiene la información ocupe un ancho de banda bastante mayor que el mínimo necesario para transmitirla. Esto tiene el efecto de provocar que la señal transmitida asuma una apariencia similar al ruido, de manera que se mezcle con el ruido de fondo. De esa manera se permite que la señal trasmitida se propague por el canal sin que la detecte alguien que quizás se encuentre escuchando. Por tanto, es posible pensar en Spread Spectrum como un método de “camuflaje” de la señal que contiene la información [3].

La principal ventaja del sistema de comunicación de Spread Spectrum es su capacidad para resistir a las interferencias ya sean las interferencias no intencionales generadas por otro usuario que trata al mismo tiempo de transmitir a través del canal, o una interferencia intencional o “jamming” dirigida directamente al receptor para sabotear la comunicación.

La técnica Spread Spectrum se puede definir en dos partes:

1. Spread Spectrum es una técnica de comunicación en la que la secuencia de datos ocupa un ancho de banda por encima del mínimo necesario para enviarlo.

2. La modulación se lleva a cabo antes de la transmisión a través de la utilización de un código que es independiente de la secuencia de datos, el mismo código se utiliza en el receptor (que opera en sincronismo con el transmisor) para demodular la señal en el receptor de manera que la secuencia de datos original puede ser recuperada [3].

2.1 SPREAD SPECTRUM CON SALTOS DE FRECUENCIA FHSS

En un sistema FHSS, un transmisor da "saltos" entre las frecuencias disponibles (figura 3) de acuerdo a un algoritmo determinado, que puede ser aleatorio o pre-planeado. El transmisor funciona en sincronización con un receptor, que sigue siendo sintonizado en el mismo centro de frecuencia que el transmisor. Una breve ráfaga de datos se transmite en una banda estrecha.

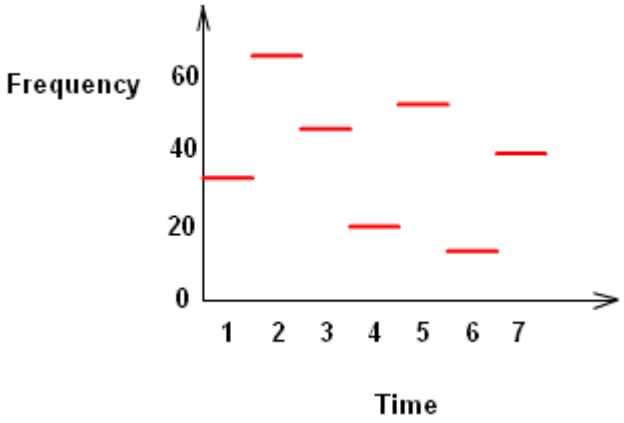


Figura 3. Canales de trasmisión en FHSS utilizados en diferentes intervalos de tiempo [8].

En este momento, el transmisor sintoniza a otra frecuencia y transmite de nuevo. El receptor por lo tanto es capaz de saltar sobre su frecuencia en un determinado ancho de banda varias veces por segundo, se transmite en una frecuencia durante un cierto período de tiempo, entonces salta a otra frecuencia y transmite de nuevo [8].

Las frecuencias utilizadas para los saltos y el orden de utilización se denominan modelo de Hopping (Hopping Pattern). El tiempo de permanencia en cada frecuencia es lo que se conoce como Dwell Time; tanto el Dwell Time como el Hopping están sujetos a restricciones por parte de los organismos de regulación [6]. El esquema básico de construcción de un transmisor en FHSS se ve en la Figura 4.

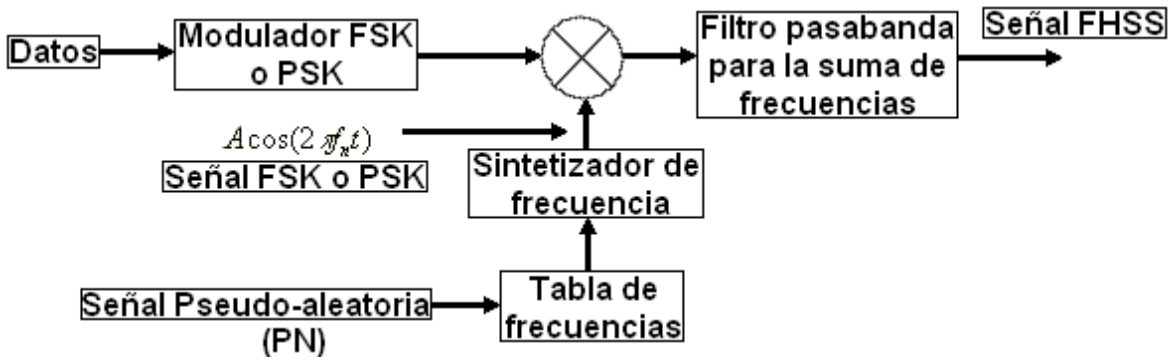


Figura 4. Transmisión de una señal con salto de frecuencia.

La secuencia de frecuencias tiene que ser idéntica tanto para el Transmisor como para el receptor. Si el receptor sigue la secuencia correcta la salida del detector

sincrónico producirá una señal coherente, similar a lo que se hubiese recibido por un receptor perfectamente sintonizado. En cambio si el receptor no sigue la secuencia correcta la salida del detector sincrónico NO podrá producir una señal coherente, lo cual le va a impedir distinguir el dato del ruido de fondo. Si alguien intenta interceptar la transmisión tendría que observar toda la banda amplia. Lo que observaría sería similar a ruido [6]. El esquema básico de construcción de un receptor en FHSS se ve en la Figura 5.

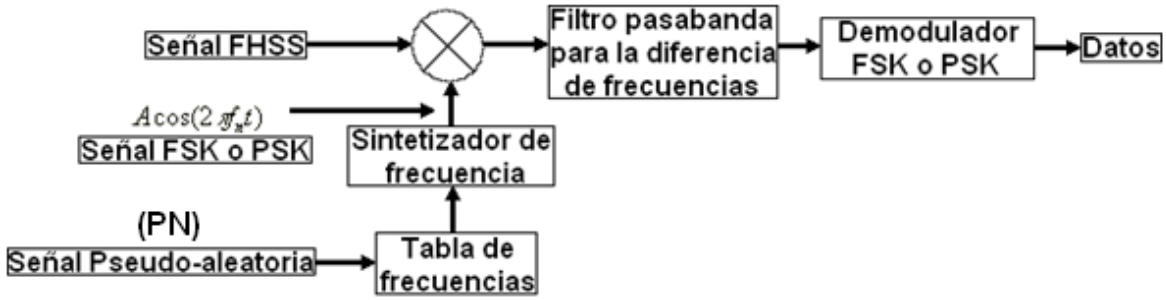


Figura 5. Recepción de una señal con salto de frecuencia.

2.2 SPREAD SPECTRUM POR SECUENCIA DIRECTA DSSS

Esta técnica consiste en la generación de un patrón de bits redundante llamado señal de chip para cada uno de los bits que componen la señal de información y la posterior modulación de la señal resultante. Cuanto mayor sea esta señal, mayor será la resistencia de la señal a las interferencias. El estándar IEEE 802.11 recomienda un tamaño de 11 bits, pero el óptimo es de 100. En la recepción es necesario realizar el proceso inverso para obtener la señal de información original. La secuencia de bits utilizada para modular cada uno de los bits de información es la llamada secuencia pseudo-aleatoria o código de dispersión [6].

Un método para aumentar el ancho de banda de la secuencia (datos) que contiene la información implica el uso de modulación. Sea {bk} una secuencia binaria y sea {ck} una secuencia pseudo-aleatoria. Sean las formas de onda b(t) y c(t) sus respectivas representaciones polares sin retorno a cero en términos de dos niveles de igual amplitud y polaridad opuesta, esto es, +/- 1. Se refiere a b(t) como la señal que contiene a la información (datos) y a c(t) como la señal pseudo-aleatoria. La modulación deseada se logra aplicando la señal de datos de b(t) y la señal pseudo-aleatoria c(t) en un modulador balanceado, como en la figura 6a. Se sabe de la teoría de la transformada de Fourier que la multiplicación de dos señales produce una señal cuyo espectro es igual a la convolución de los espectros de las dos señales que la componen. En consecuencia, si la señal del

mensaje b(t) es de banda angosta y la señal pseudo-aleatoria c(t) es de banda amplia, la señal producto (modulada) m(t) tendrá un espectro que es casi el mismo que la señal pseudo-aleatoria de banda amplia. En otras palabras, la secuencia pseudo-aleatoria desempeña el papel de un código de dispersión [3].

Al multiplicar la señal b(t) que contiene información por la señal pseudo-aleatoria c(t), cada bit de información se "recorta" en varios incrementos de tiempo que suelen conocerse como divisiones.

En la transmisión de banda base, la señal del producto m(t) representa a la señal transmitida.

Es posible entonces expresar la señal transmitida como

$$m(t) = c(t)b(t) \tag{1}$$

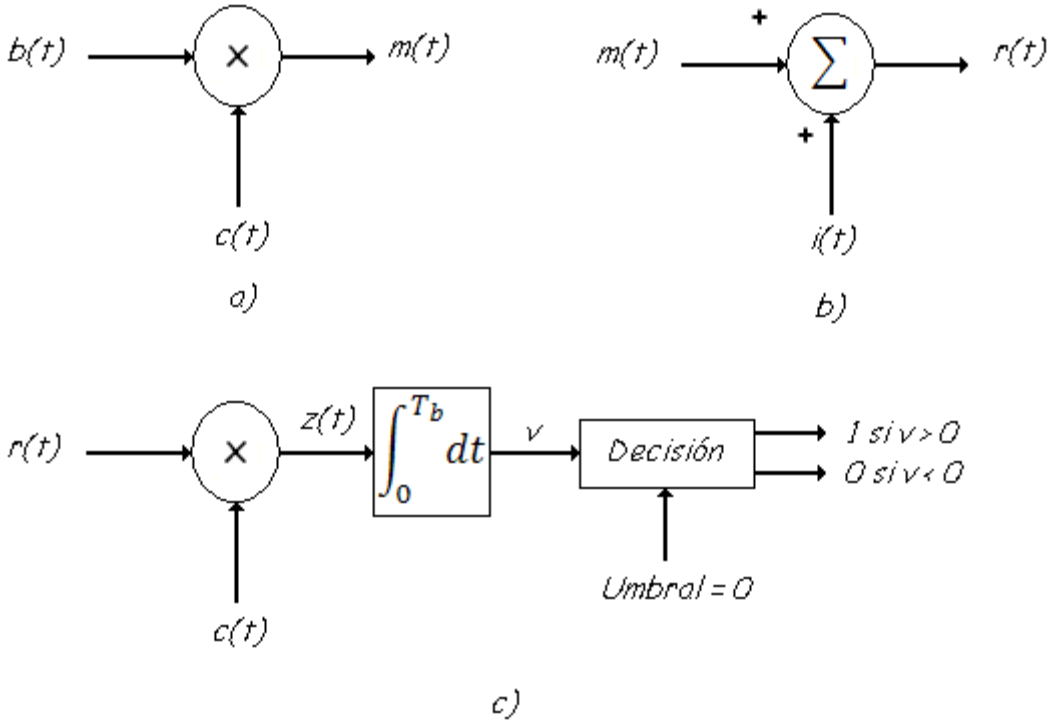


Figura 6. Modelo idealizado del sistema Spread Spectrum en Secuencia Directa. a) Transmisor. b) Canal. c) Receptor [3].

La señal recibida r(t) consiste en la señal transmitida m(t) más una interferencia aditiva denotada por i(t), como se indica en el modo de canal de la figura 6b. Por tanto,

$$r(t) = m(t) + i(t)$$
$$r(t) = c(t)b(t) + i(t) \tag{2}$$

Para recuperar la señal de mensaje original b(t), la señal recibida r(t) se aplica a un demodulador que está compuesto por un multiplicador seguido de un integrador y un dispositivo de decisión, como en la figura 6c. El multiplicador se alimenta con una secuencia pseudo-aleatoria generada localmente que es una réplica exacta del que se usa en el transmisor. Además, suponemos que el receptor opera en sincronismo perfecto con el transmisor, lo que significa que la secuencia pseudo-aleatoria en el receptor está alineada exactamente con la correspondiente al transmisor [3]. La salida del multiplicador en el receptor está dada entonces por:

$$z(t) = c(t)r(t)$$
$$z(t) = c^2(t)b(t) + c(t)i(t) \quad (3)$$

la ecuación (3) muestra que la señal de datos b(t) se multiplica dos veces por la señal de ruido c(t), en tanto que la señal indeseable i(t) se multiplica solamente una vez. La señal pseudo-aleatoria c(t) se alterna entre los niveles -1 y +1, y la alternancia se destruye cuando ésta se eleva al cuadrado. Por consiguiente,

$$c^2(t) = 1 \quad para\ todo\ t \quad (4)$$

Por tanto, es posible simplificar la ecuación (3) como

$$z(t) = b(t) + c(t)i(t) \quad (5)$$

De ese modo se observa, de acuerdo con la ecuación (5), que la señal de datos b(t) se reproduce a la salida del multiplicador en el receptor, excepto por el efecto de la interferencia representada por el término aditivo c(t)i(t). La multiplicación de la interferencia i(t) por la señal pseudo-aleatoria c(t) generada localmente equivale a que el código de dispersión afectará a la interferencia exactamente como lo hizo la señal original en el transmisor. Después de esto se observa que la componente de datos b(t) es de banda angosta, en tanto que la componente adulterada c(t)i(t) es de banda amplia. Esto se soluciona con un filtro pasabajas que permita el paso solo de la componente de datos b(t) (de banda angosta) y elimine el paso de la componente adulterada c(t)i(t) la cual es de banda mayor, reduciéndose significativamente el efecto de interferencia [3].

El bloque básico de construcción del transmisor de Spread Spectrum en Secuencia Directa (DSSS) se muestra en la figura 7. El dato a transmitir es multiplicado con la secuencia Pseudo-aleatoria (PN) para generar el dato "ensanchado". Los datos son convertidos a la forma analógica y son transmitidos después de la modulación [9].

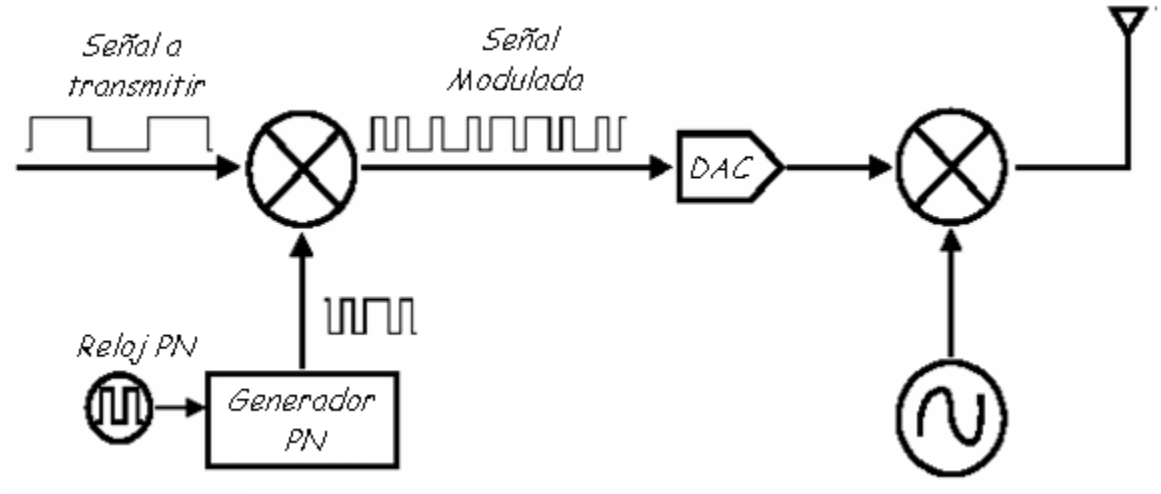


Figura 7. Transmisor DSSS básico [9].

La multiplicación del dato con la secuencia PN provoca el incremento del ancho de banda por un factor conocido como la ganancia de procesamiento (PG). PG se expresa en [dB] y se calcula mediante la razón de cambio del período del dato, Ts, para el período del circuito de la secuencia PN, Tc: donde el Ts >> Tc.

$$PG = 10.log_{10}(\frac{T_s}{T_c}) \quad (6)$$

Esto da como resultado la disminución de la potencia de pico de densidad espectral de la ganancia de procesamiento (figura 8) [9].

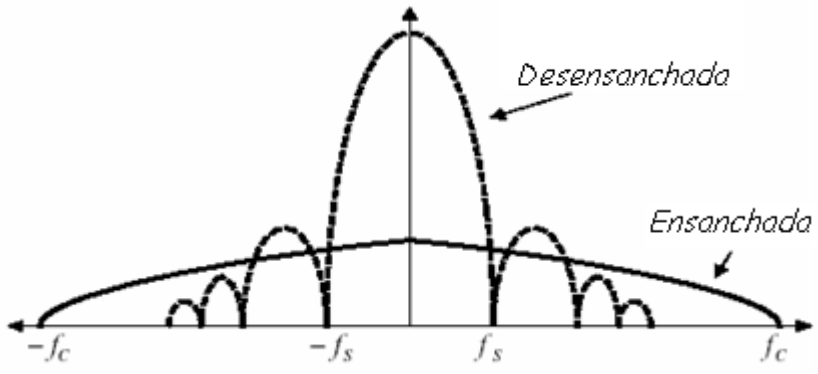


Figura 8. Densidad espectral de potencia de la señal transmitida [9].

La señal de Spread Spectrum de transmisión es una señal de banda amplia que apenas se diferencia del ruido del canal si la ganancia de procesamiento es lo suficientemente alta.

La cantidad de degradación que enfrenta la señal en el canal de transmisión depende del grado y las características de las fuentes de interferencia presentes en el canal. Una de las ventajas de Spread Spectrum es su resistencia al ruido de banda estrecha.

Como se puede ver en la figura 9, el espectro de la señal es mucho más amplio que el Dispositivo de interferencia de banda estrecha en donde la mayoría de la potencia de la señal aún se puede recibir.

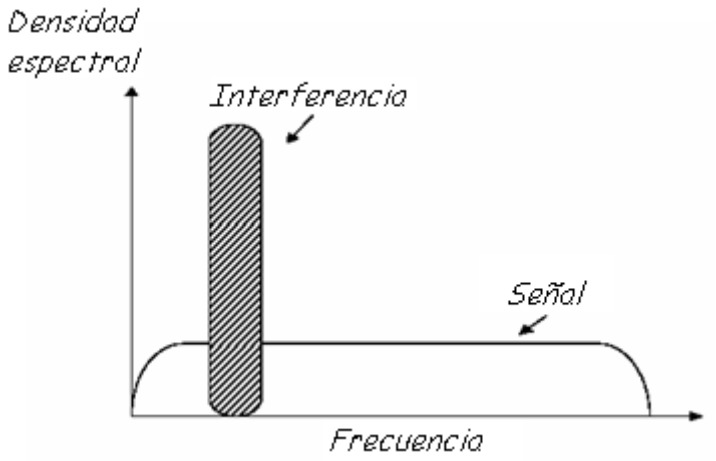


Figura 9. Interferencia de banda estrecha en la transmisión de la señal [9].

El bloque básico de construcción del receptor se muestra en la figura 10. La señal recibida se multiplica por una réplica local de la secuencia PN del transmisor para "desensanchar" la señal original. El oscilador local en el receptor se supone que está en sincronización con el oscilador en el transmisor.

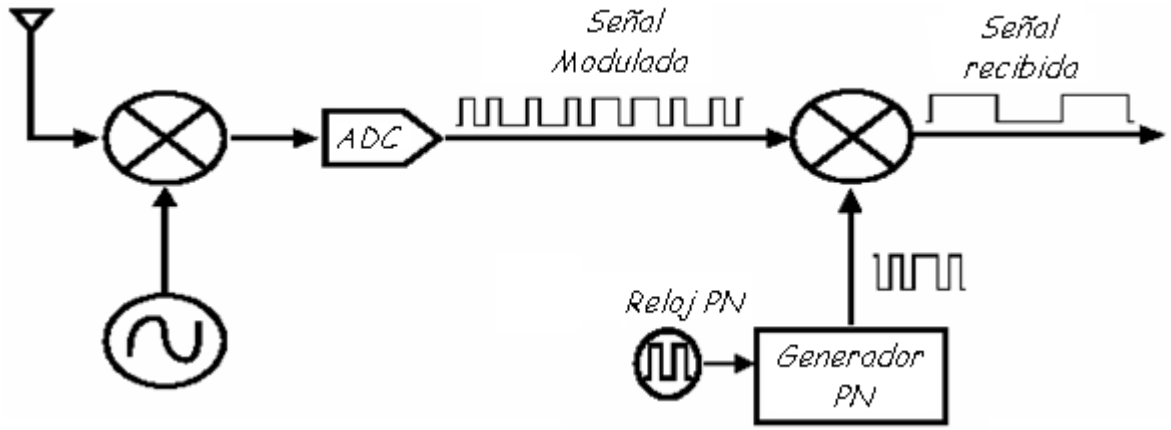


Figura 10. Receptor DSSS básico [9].

El espectro ensanchado de los datos recuperados por el receptor, junto con el ruido adicional durante la transmisión se muestra en la figura 11 [9].

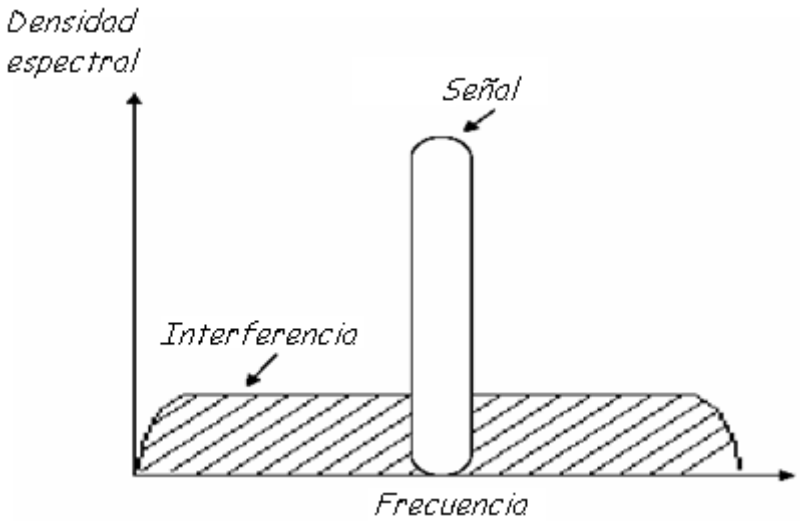


Figure 11. Densidad espectral de potencia en la señal recibida [9].

Como se puede observar la interferencia es ensanchada por el receptor mientras los datos están siendo desensanchados. El diagrama de tiempos de todo el proceso se muestra en la figura 12.

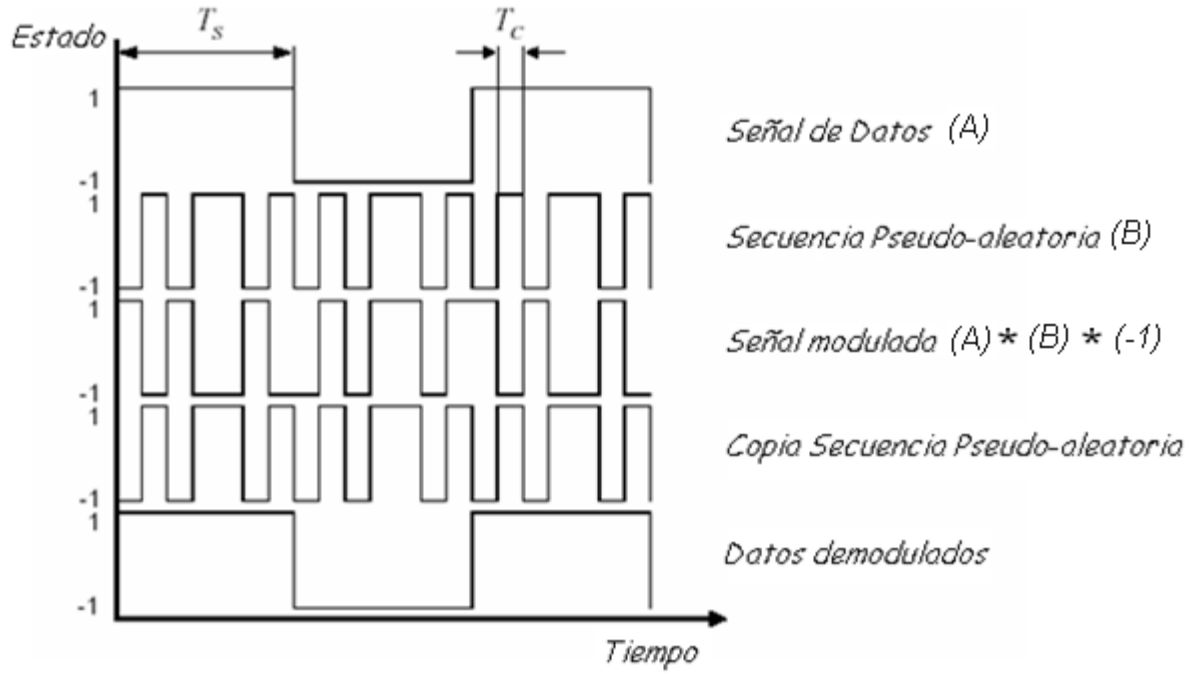


Figura 12. Secuencia de tiempos de la señal codificada y decodificada [9].

Hay muchos tipos de secuencias ensanchadas y métodos para producirlas. El tipo de secuencia que se utilice depende del canal de comunicación usado.

El aspecto más sensible de un sistema de secuencia directa es la sincronización de la secuencia del transmisor PN a la del receptor, donde un desplazamiento de hasta un ciclo de reloj de PN puede resultar en ruido en lugar de una secuencia de datos desensanchados. La sincronización se compone de dos elementos a saber, adquisición y seguimiento.

Estos pueden ser vistos como la alineación de las secuencias PN, y el mantenimiento de este estado alineado. Los sistemas de sincronización hacen uso de factores que determinan la correlación de la señal recibida para la réplica local de la transmisión de secuencias PN. Cuando un valor de correlación alto se detecta, la adquisición se ha logrado. La rapidez con que se reciben los datos depende de la rapidez de la correlación entre las secuencias PN logradas [9].

2.3 SECUENCIAS PSEUDOALEATORIAS

Los sistemas Spread Spectrum, además de ser prácticamente inmunes a las interferencias tanto de carácter intencional como casual (entre ellas las interferencias debido a la propagación de una onda por varios caminos diferentes "multitrayecto"), pueden ser utilizados en sistemas de múltiple acceso por división de código (CDMA) [10].

Una señal de Spread Spectrum es generada usando una señal pseudo-aleatoria (PN), la cual varía según el tipo de Spread Spectrum utilizado (FHSS o DSSS). Para Spread Spectrum en Secuencia Directa ésta coincide con un código pseudo-aleatorio. En el caso de Spread Spectrum en Saltos de Frecuencia cambia conforme al código PN utilizado.

Una señal PN es una función del dominio del tiempo generada determinísticamente, que cumple con ciertas propiedades matemáticas interesantes que facilitan las funciones de ensanchamiento del espectro de la señal de información y detección por un receptor autorizado (el cual conoce las propiedades con las que se generó la señal PN), dificultando a su vez la intercepción de la misma por parte de un receptor no autorizado, ya que la señal transmitida queda camuflada dando la impresión de ser ruido. La función de autocorrelación debería parecerse, tanto como sea posible, a la función de autocorrelación del ruido blanco Gaussiano, por esto puede llamarse secuencia PN o pseudo-aleatoria. Todos los códigos utilizados son secuencias periódicas de unos y ceros de período N. Un código está compuesto por la división de un bit en pequeños intervalos llamados chips a los que se le asignan términos binarios como + ó - ,0° ó 180°, 1 ó 0, etc. [10].

Existen diversos tipos de secuencias PN, pero la más importante es la secuencia de máxima longitud o Secuencia M, la cual, posee un valor de autocorrelación que es muy beneficioso para las funciones que debe desempeñar un sistema Spread Spectrum, entre otras características. La Secuencia M se puede obtener usando registros de desplazamiento realimentados adecuadamente, asociados a una lógica digital conformada por compuertas XOR.

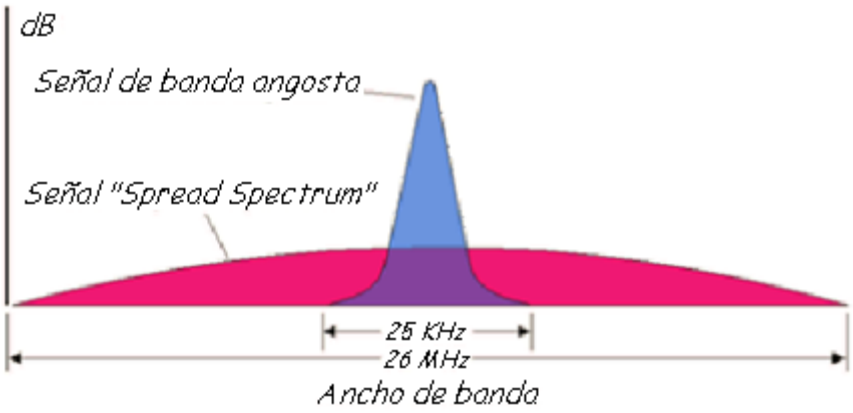


Figura 13. Espectro ensanchado [10].

En la figura 13 se observa que la señal Spread Spectrum es de banda amplia, la cual es producto de una secuencia pseudo-aleatoria por una señal de datos de banda angosta. Para que se diera el efecto de ensanchar el ancho de banda, se debió haber multiplicado la señal de datos de banda angosta por una señal de banda ancha, concluyendo que el espectro de la secuencia PN es de banda ancha, muy similar al espectro de la señal Spread Spectrum.

Es necesario diseñar códigos para secuencia directa que conserven un alto parecido con lo que sería una secuencia perfectamente aleatoria. Estos son los llamados códigos pseudo-aleatorios, ya que aunque proceden de una función booleana determinística, la probabilidad de ocurrencia de cualquiera de los dos símbolos sigue una ley de distribución Gaussiana.

Tres de los códigos pseudo-aleatorios más populares utilizados en Spread Spectrum son los códigos Barker, Gold y de Máxima Longitud [11].

2.3.1 Gold

Los códigos de Gold tienen las siguientes propiedades:

- El generador Gold está en capacidad de originar distintos códigos, dependiendo de las condiciones iniciales del generador.

- Todas las secuencias generadas a partir de un mismo generador de Gold poseen el mismo período y tienen idénticas propiedades de correlación.

- La multiplicación de cualquiera de dos secuencias a y b de una determinada familia de códigos Gold, resulta en otra secuencia perteneciente a la misma familia.

- La correlación cruzada entre dos secuencias cualquiera a y b de la familia de Gold cumple con la siguiente desigualdad [11]:

$$|R_{ab}(k)| \leq \begin{cases} \frac{1}{L}\left[2^{(n+1)/2}+1\right] \; si\ n\ es\ impar \\ \frac{1}{L}\left[2^{(n-1)/2}+1\right] \; si\ n\ es\ par \end{cases} \tag{7}$$

2.3.2 Barker

Los códigos Barker son subconjuntos de secuencias Pseudo-aleatorias, comúnmente usados para sincronizar los datos en sistemas de comunicación digital. Tienen una longitud de hasta 13 bits usados en Spread Spectrum en secuencia directa y sistemas de compresión de pulsos de radar por su baja autocorrelación [12]. En la tabla 1 se observan los códigos Barker conocidos.

Length	Codes	
2	+1 −1	+1 +1
3	+1 +1 −1	
4	+1 +1 −1 +1	+1 +1 +1 −1
5	+1 +1 +1 −1 +1	
7	+1 +1 +1 −1 −1 +1 −1	
11	+1 +1 +1 −1 −1 −1 +1 −1 −1 +1 −1	
13	+1 +1 +1 +1 +1 −1 −1 +1 +1 −1 +1 −1 +1	

Tabla 1. Códigos Barker conocidos [12].

Los códigos Barker deben cumplir las siguientes propiedades:

1. **Propiedad de balance:** en cada periodo del código, el número de 1's es siempre uno más que el número de 0's.

2. **Propiedad Run:** Un Run está definido como una secuencia consecutiva de 1's y 0's. En una secuencia pseudo aleatoria, la mitad de los Runs tiene longitud 1. El total de los Runs de una secuencia pseudo aleatoria es:

$$R = (N + 1)/2 \text{ [9]}$$

Donde R es la cantidad total de Runs de la secuencia pseudo aleatoria y N es el número de bits.

3. **Propiedad de correlación:** La función de auto correlación de una secuencia debe ser periódica y de valor binario.

2.3.3 Secuencia de Máxima Longitud

Las secuencias de longitud máxima cuentan con muchas de las propiedades que posee una verdadera secuencia binaria aleatoria. Esta última constituye una secuencia en la cual la presencia del número binario 1 o 0 es igualmente probable. Algunas de las propiedades de longitud máxima son las siguientes [10]:

- El desplazamiento cíclico de una secuencia PN es también una secuencia PN.

- Una secuencia de máxima longitud, satisface la recurrencia:

$$C_{i+m} = g_{m-1}C_{i+m-1} + g_{m-2}C_{i+m-2} \cdots + g_1C_{i+1} + C_1 \quad (8)$$

- Si una ventana de ancho no es desplazada a lo largo de una secuencia de registros de un estado, cada una de las m duplas binarias que se generan es vista una sola vez.

- En cualquier secuencia m se tiene:

$$2^{m-1}\ unos\ y\ 2^{m-1} - 1\ ceros \quad (9)$$

- La suma binaria de una secuencia-M es también otra secuencia-M.

- La suma de una secuencia-M con el desplazamiento cíclico de la misma es otra secuencia-M.

- Un Run es una secuencia consecutiva de unos y ceros. En una secuencia-M la mitad de los Runs tiene longitud uno, un cuarto tiene longitud dos, un octavo tiene longitud tres y así sucesivamente. El total de Runs de una secuencia-M es (N+1)/2 donde N=2m-1[10].

3. TECNOLOGÍA FPGA (FIELD PROGRAMMABLE GATE ARRAY)

Una FPGA es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad se puede programar. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip. Se puede configurar estos chips para implementar funcionalidades de hardware personalizadas sin tener que recurrir a una placa o soldadura de hierro [13].

Las FPGAs tienen las ventajas de ser reprogramables (lo que añade una enorme flexibilidad al flujo de diseño), el tiempo de desarrollo es menor y sus costes de desarrollo y adquisición son también mucho menores para pequeñas cantidades de dispositivos. El tamaño, estructura, número de bloques y la cantidad y conectividad de las conexiones varían en las distintas arquitecturas.

Es un circuito integrado que contiene celdas lógicas idénticas (64 hasta 8'000.000) que se puede ver como componentes estándar. Las celdas lógicas se interconectan por medio de una matriz de cables y switches programables; se programa por la carga de celdas de memoria de configuración, que controlan la lógica e interconexiones.

Hay software especial para definir las conexiones de los switches y las funciones de las celdas lógicas. El software traduce diagramas esquemáticos del usuario o código de lenguaje de descripción de hardware, y luego valida el diseño traducido; el software permite al usuario influenciar una implementación y validarla para obtener mayor eficiencia y utilización del dispositivo [14].

3.1 LENGUAJE VHDL

El lenguaje VHDL[8] fue creado con el propósito de especificar y documentar circuitos y sistemas digitales utilizando un lenguaje formal. Las principales características del lenguaje VHDL se explican en los siguientes puntos:

- Descripción textual normalizada: El lenguaje VHDL es un lenguaje de descripción que especifica los circuitos electrónicos en un formato adecuado para ser interpretado tanto por máquinas como por personas. Está, como ya se ha dicho, normalizado, o sea, existe un único modelo para el lenguaje, cuya utilización está abierta a cualquier grupo que quiera desarrollar

[8] VHDL: VHSIC Hardware Description Language
- VHSIC: Very High Speed Integrated Circuits.

herramientas basadas en dicho modelo, garantizando su compatibilidad con cualquier otra herramienta que respete las indicaciones especificadas en la norma oficial. Es un lenguaje ejecutable, lo que permite que la descripción textual del hardware se materialice en una representación del mismo utilizable por herramientas auxiliares tales como simuladores y sintetizadores lógicos, compiladores de silicio, simuladores de tiempo, de cobertura de fallos, herramientas de diseño físico, etc.

- Amplio rango de capacidad descriptiva: El lenguaje VHDL posibilita la descripción del hardware con distintos niveles de abstracción, pudiendo adaptarse a distintos propósitos y utilizarse en las sucesivas fases que se dan en el desarrollo de los diseños.

- Otras ventajas: Además de las ventajas ya reseñadas también es destacable la capacidad del lenguaje para el manejo de proyectos de grandes dimensiones, las garantías que comporta su uso cuando, durante el ciclo de mantenimiento del proyecto, hay que sustituir componentes o realizar modificaciones en los circuitos, y el hecho de que, para muchas organizaciones contratantes, sea parte indispensable de la documentación de los sistemas [15].

3.1.1 Principios básicos al modelado en VHDL

La realización del modelo de hardware de un dispositivo en VHDL consiste en la elaboración de dos unidades de código VHDL: una Declaración de Entidad y un Cuerpo de Arquitectura.

La Declaración de Entidad es la unidad de diseño VHDL que sirve para especificar el interfaz de los dispositivos. Cumple, por tanto, funciones equivalentes a las de los símbolos en las representaciones gráficas.

El Cuerpo de Arquitectura es la unidad de diseño VHDL que sirve para especificar el funcionamiento de un dispositivo identificado por una determinada declaración de Entidad, por lo que se puede considerar el equivalente a las tablas de verdad o a los cronogramas [15]. En la figura 14 se muestra la representación gráfica del concepto de entidad y arquitectura.

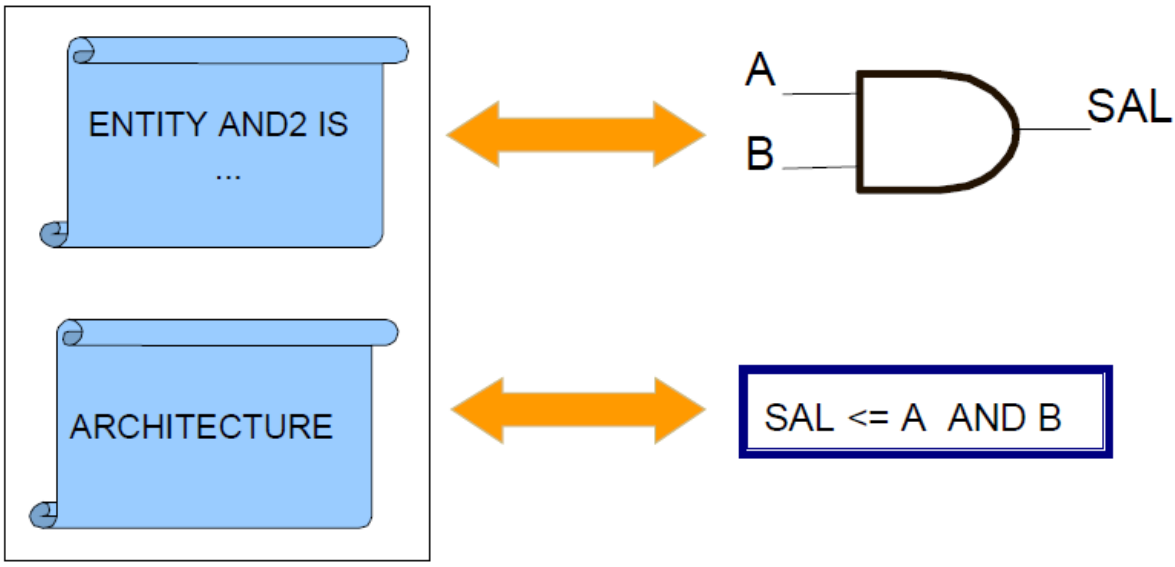


Figura 14. La entidad representa el dispositivo físico y la arquitectura representa el comportamiento lógico [15].

3.1.2 Tipos de datos predefinidos y operadores

Los siguientes son los tipos de datos con los cuales se trabaja en VHDL, cada tipo de dato es asignado a una variable y es utilizado dependiendo del tipo de programa que se esté realizando:

a. Enumerados:
CHARACTER: Formado por los 128 caracteres **ASCII**.
BOOLEAN: Definido sobre los valores (**FALSE**, **TRUE**).
BIT: Formado por los caracteres **'0'** y **'1'**
SEVERITY LEVEL: Formado por los valores **(NOTE, WARNING, ERROR, FAILURE)**.
STD_LOGIC: Formado por los caracteres **'0', '1', 'Z', 'U'**

b. Escalares:
INTEGER y **REAL**: Con un rango que depende de cada herramienta.

c. Arrays:
STRING: Definido como un *array* de caracteres.
BIT_VECTOR: Definido como un *array* de **BIT.**
STD_LOGIC VECTOR: Definido como un *array* de **STD_LOGIC.**

d. Físicos:
TIME: Definido para especificar unidades de tiempo.

e. Tipos para operaciones de entrada/salida sobre ficheros:
LINE: Puntero a **STRING**.
TEXT: Fichero de **STRINGS**
SIDE: Enumerado sobre los valores **(RIGHT, LEFT)**

También existen subtipos predefinidos por restricción de rango:
NATURAL: desde **0** al máximo valor **INTEGER**.
POSITIVE: desde **1** al máximo valor **INTEGER**.
WIDTH: desde **0** al máximo valor **INTEGER [13]**.

En la tabla 2 se observan algunos operadores básicos en VHDL.

TIPO DE OPERACIÓN	SIMBOLO	FUNCION
Aritméticas de dos operandos	+ - * / mod rem **	suma resta producto división módulo resto potencia
Aritméticas de un operando	+ - abs	Incremento Decremento Valor absoluto
Relacionales	= /= < > <= =>	Igual a distinto a menor que mayor que menor o igual que mayor o igual que
Lógicas de dos operandos	and or nand nor xor xnor	And lógico or lógico nand lógico nor lógico or exclusiva xor negada
Lógicas de un operando	not	Complementación
Encadenamiento	&	Encadenamiento

Tabla 2. Operadores básicos en VHDL [15]

La sintaxis básica para declarar la entidad de un modelo a realizar en VHDL es la siguiente, en la cual se especifica el nombre de cada pin y su respectivo funcionamiento como entrada, salida, buffer o bidireccionales:

```
ENTITY {nombre del dospositivo} IS

PORT(

        {lista de puertos de entrada}        : IN            {tipo de dato};
```

```
        {lista de puertos bidireccionales} : INOUT    {tipo de dato};
        {lista de puertos de salida}       : OUT      {tipo de dato};
        {lista de puertos de entrada}      : BUFFER   {tipo de dato});
END ENTITY;
```

La sintaxis básica para declarar una arquitectura de un modelo a realizar en VHDL es la siguiente, en la cual se especifica el nombre de la arquitectura y que entidad representa, dentro de esta se dice que funciones va a realizar la entidad.

```
ARCHITECTURE {nombre_de_arquitectura} OF {nombre_de_entidad} IS

        {zona de declaración}

        BEGIN

        {zona de descripción}

        END {nombre_de_arquitectura};
```

3.1.3 Sentencias de descripción

La programación en VHDL es semejante a la programación en cualquier otro lenguaje de alto nivel, en el cual también se usan ciclos, sentencias condicionales, selección de casos, contadores, instrucciones de manejo de tiempo, algunas sentencias básicas son las siguientes:

Sentencia IF:

```
IF X = '0' THEN                 -- Si x es igual a '0' entonces
        nuevo_estado <= S0;     -- Se asigna el valor de S0 a nuevo_estado
ELSE                            -- Sino
        nuevo_estado <= S2;     -- Se asigna el valor de S2 a nuevo_estado
END IF;
```

Sentencia FOR:

```
FOR i IN 15 DOWNTO 0 LOOP       -- Para I desde 15 hasta 0
        IF data_in(i) = '0' THEN    -- Si data_in en la posicion i es igual a '0'
                Pb_temp <= '0';     -- Reinicie Pb_temp
```

```
        ELSE                    - - Sino
            Pb_temp <= '1';     - - Se asigna 1 a Pb_temp
        END IF;
END LOOP;
```

Sentencia CASE:

```
CASE current_state IS
        WHEN RESET =>
            Count_end <= '0';
            . . .
        WHEN RAMPA1 =>          - - Si current_state tiene el valor de rampa 1
            Count_clk_en <= NOT end_counter;
        WHEN others =>          - - Si current_state toma valores diferentes
            Count_end <= '-';
            Op_down   <= '-';
END CASE;
```

Lo que hace que el funcionamiento de una FPGA sea óptimo comparado con otros dispositivos es que basa su funcionamiento en la concurrencia, esto quiere decir que si en un programa se tienen diferentes procesos, estos se ejecutan al mismo tiempo (en paralelo) optimizando el tiempo de procesamiento.

3.2 COMPILADOR XILINX ISE 9.2i

El diseño del modulador y demodulador Spread Spectrum construido en este trabajo se llevó a cabo usando el compilador xilinx ise 9.2i, a continuación se mostrarán los pasos a seguir cuando se desea crear un proyecto:

3.2.1 Creación de un proyecto.

Para arrancar el programa se da click en el ícono de Xilinx o bien, desde:

Inicio → Todos los programas → Xilinx ISE 9.2i → Project Navigator (figura 15a)

Puede aparecer una ventana con el "Tip of the day" que son indicaciones que hace la herramientas cada vez que se arranca. Si se leen habitualmente se puede ir aprendiendo poco a poco. Click en Ok, con lo que se cierra dicha ventana.

Normalmente la herramienta abre el último proyecto que se ha realizado. Así que se cierra haciendo click en:

File → Close Project.

Para empezar a crear un nuevo proyecto, click en File → New Project... y saldrá la ventana New Project Wizard – Create New Project. En ella se pone el nombre del proyecto, en este caso se llamará "Ejemplo", se indica la ruta donde se guardará el proyecto (D:/Ejemplo). Respecto al nombre y a la ruta, no es conveniente trabajar desde un dispositivo de memoria USB, ni tampoco incluir en la ruta o el nombre ni acentos ni eñes, ni caracteres extraños, lo más conveniente es limitarse a caracteres alfanuméricos, y usar el guión bajo en vez del espacio, además se recomienda que la ruta donde se guarda el proyecto no esté muy anidada, ya que genera errores a la hora de sintetizar (figura 15b).

Para el último recuadro de la ventana, donde pone Top-Level Source Type se selecciona HDL, ya que el diseño se realizará mediante código VHDL.

Luego de realizar el proceso anterior se da click en "Next". Ahora aparece la ventana de selección del dispositivo (figura 15c). En Product Category se pone All. En la familia se pone Spartan3E, que es la FPGA presente en el Starter Kit. Los siguientes datos se observan en el texto del encapsulado de la FPGA.

Al finalizar lo anterior, se da click en "Next" y en la ventana siguiente click en "New Source" esto hará saltar a una nueva ventana que pedirá seleccionar el tipo de fuente a crear. Como se está trabajando con código de VHDL, se selecciona "VHDL Module", y se nombra al fichero Ejemplo. Este paso se muestra en la figura 16. Posteriormente click en "Next" y en la siguiente ventana en "Finish". La siguiente ventana es el asistente para crear la entidad del diseño en la cual se seleccionan las entradas y salidas del dispositivo a modelar, este proceso se observa en la figura 17.

Luego de establecer los pines de entrada y salida, se da click en "Next", luego en "Finish" y en la ventana que aparece se da click en "Yes", en las siguientes ventanas se confirma si la configuración previa es correcta, si así es se da click en "Next", "Next" y luego "Finish". Luego del proceso anterior se puede empezar a programar el código fuente, notando que el ayudante para crear la entidad ahorró trabajo haciendo la entidad automáticamente y solo restando la elaboración de la arquitectura.

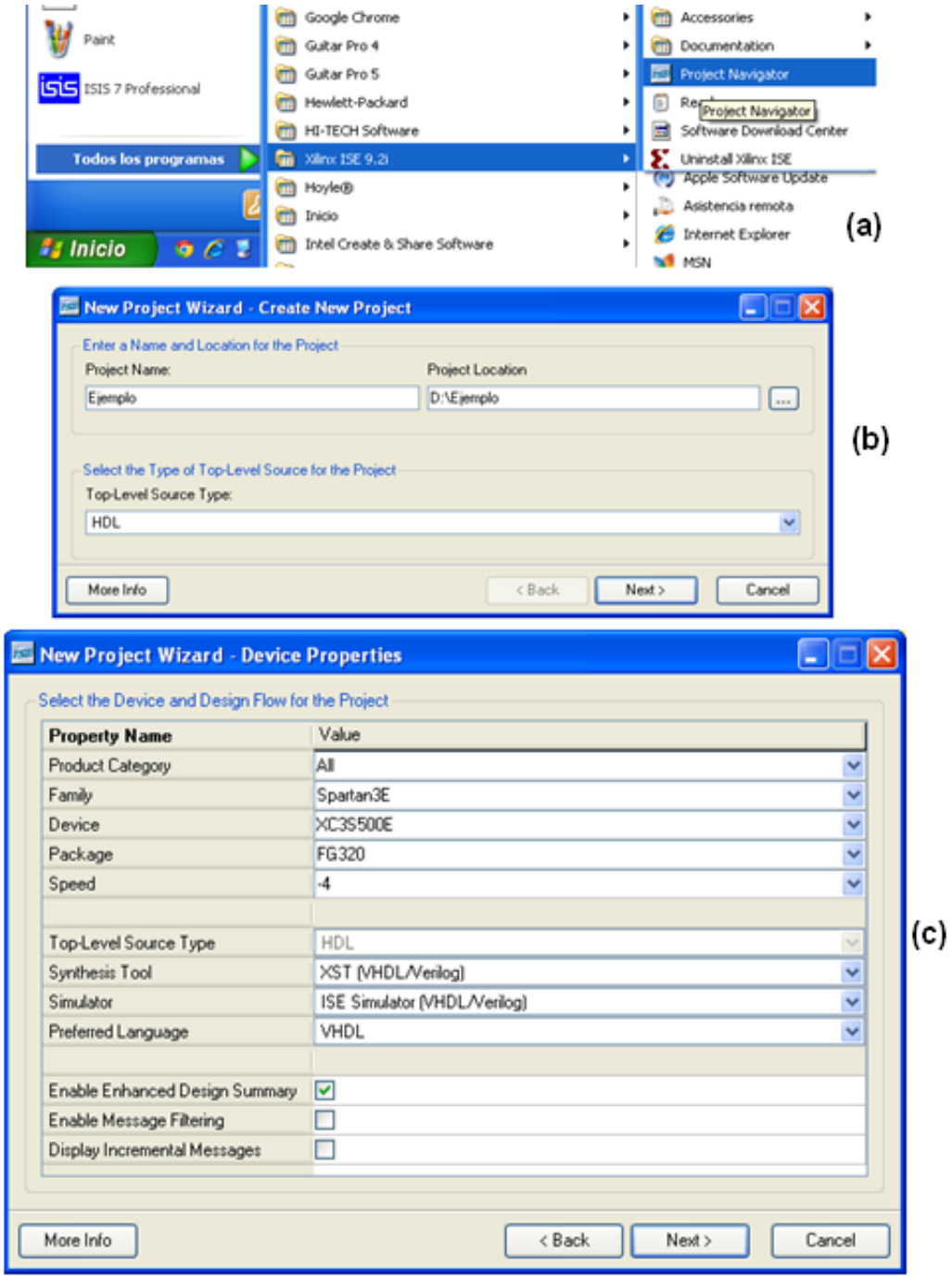


Figura 15. a. Abriendo Xilinx. b. Creación de un nuevo proyecto c. Ventana de selección del dispositivo.

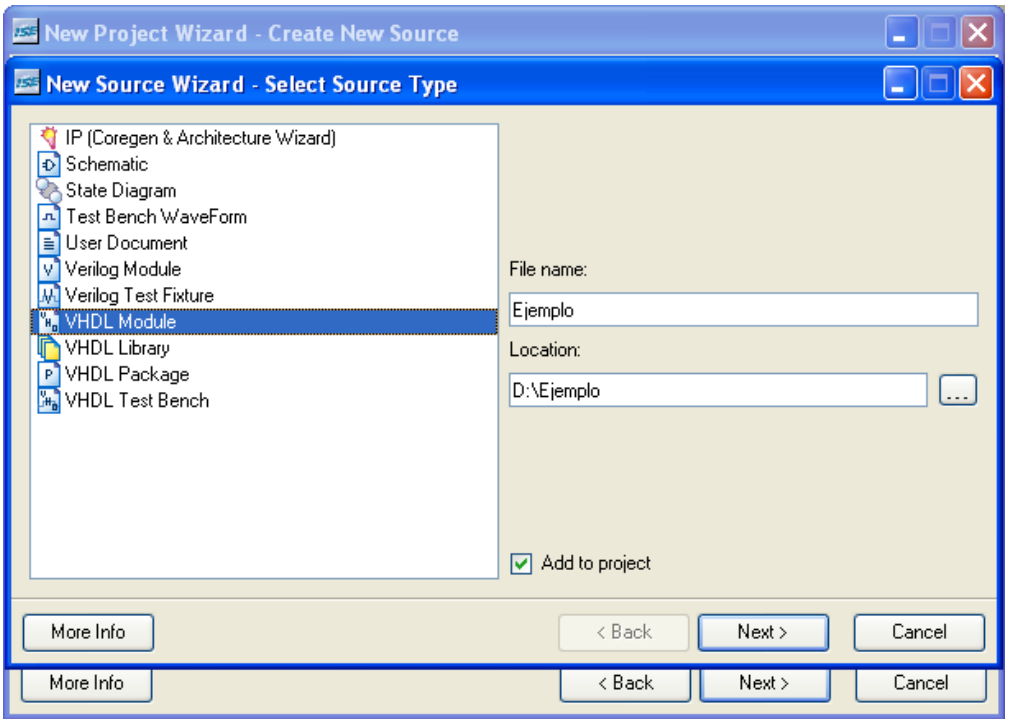


Figura 16. Selección del tipo de fuente a crear.

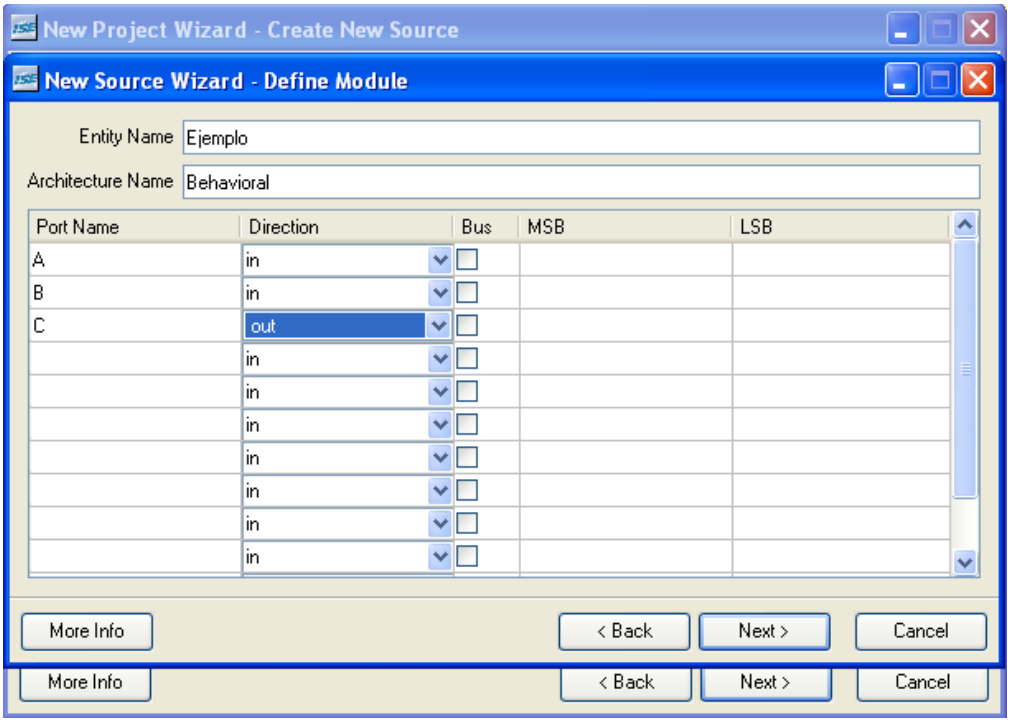


Figura 17. Asistente para crear la entidad.

3.2.2 Síntesis e implementación del diseño

Para este ejemplo se hizo el diseño de una compuerta lógica AND, con dos entradas de datos A y B y una salida C. Figura 18

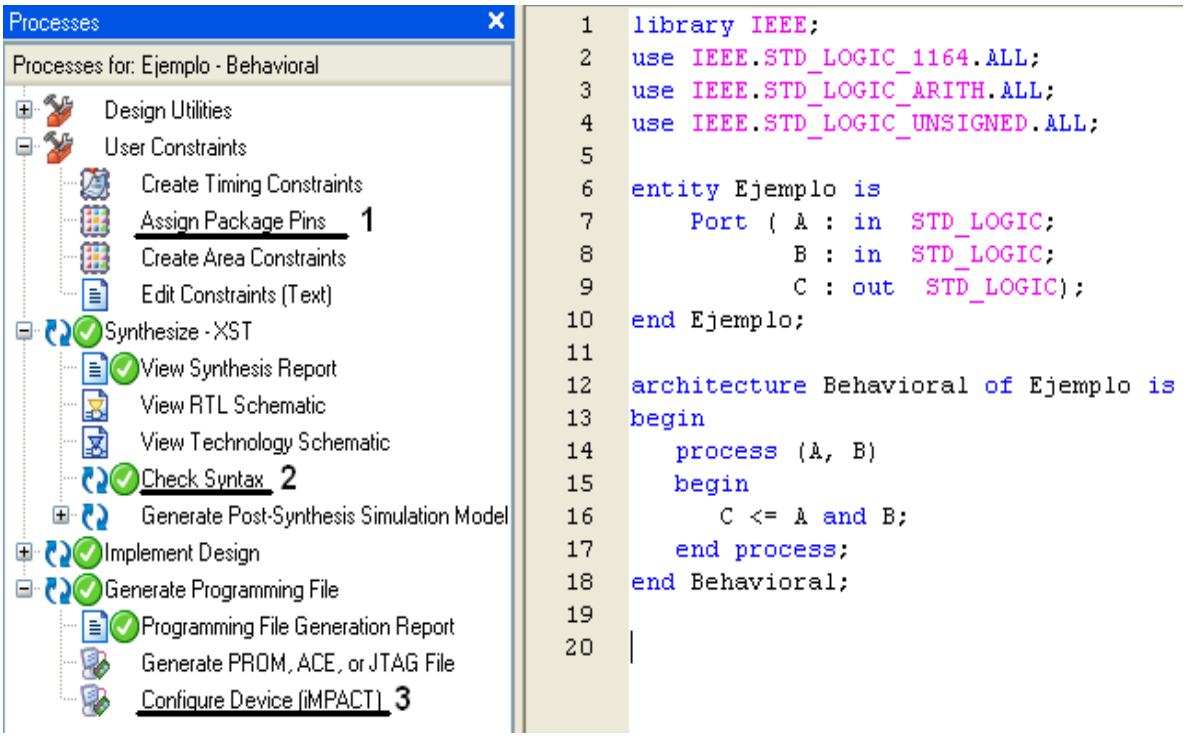


Figura 18. Código debidamente sintetizado e implementado.

Como se ve en la figura 18, el asistente para crear la entidad deja el código de la entidad ya estructurado, lo único que se hace es describir en la arquitectura el comportamiento lógico de la compuerta AND, al terminar de escribir el programa, se verifican errores, esto se hace en el numeral 2 de la figura 18 en "Check Syntax", luego se asignan los pines que representarán la compuerta AND físicamente, esto se hace en el numeral 1 de la figura 18 "Assign Package Pins", luego se procede a generar el archivo de extensión ".bit" para proceder a programarlo en la FPGA, esto se hace dando doble click en el numeral 3 de la figura 18 "Configure Device (iMPACT)". Cabe mencionar que esto debe hacerse siempre y cuando la sintaxis y la síntesis sean correctas, que no tengan ningún error.

Al generar el archivo .bit aparece un esquema que representa la FPGA y los dispositivos asociados a ella, esto se observa en la figura 19, se asigna el archivo .bit a la FPGA correspondiente (XC3S500E), los demás dispositivos se ignoran

(BYPASS). Luego se da click derecho → Program en el gráfico que representa la FPGA, con lo cual, el diseño queda implementado físicamente en la FPGA.

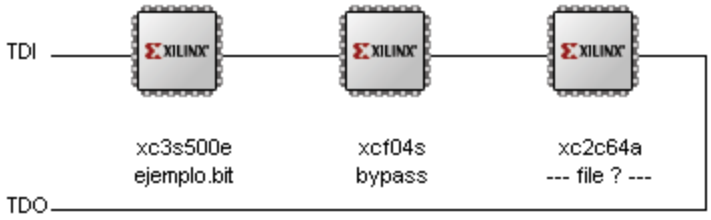


Figura 19. Asignación del archivo .bit a la FPGA.

4. DISEÑO

El diseño de este sistema de comunicaciones fue desarrollado e implementado en una FPGA Spartan 3E, programando los respectivos algoritmos usando el lenguaje VHDL y con la ayuda del compilador Xilinx ISE 9.2i. Además de esto se usó un amplificador operacional con un ancho de banda elevado (OPA860) para equilibrar las impedancias de las FPGA's.

El algoritmo general de diseño es el siguiente:

Se hace la conexión directa del generador de señales a la FPGA para obtener la señal cuadrada, esta es la señal de entrada en la FPGA 1 (Emisor) para su posterior tratamiento, haciendo la modulación mediante la técnica Spread Spectrum en secuencia directa. A continuación los datos debidamente modulados se envían a la FPGA 2 (Receptor) por par de cobre, en el receptor se demodula la información con la misma técnica obteniendo los datos inicialmente capturados, terminando de manera efectiva el proceso de comunicación. (Figura 20).

Figura 20. Diseño sistema de comunicación.

Dado que se busca que entre emisor y receptor haya máxima transferencia de potencia se buscó acoplar las impedancias en la entrada y la salida del canal. Un amplificador operacional (Op. Amp.) ideal debe tener una impedancia de entrada infinita y una impedancia de salida nula, por eso se usó un amplificador operacional en esta conexión, porque para el emisor él refleja una impedancia tendiendo a infinito y para el receptor una nula. Además de lo anterior fue necesario que el amplificador operacional tuviera un ancho de banda elevado, ya que una de las características de Spread Spectrum es incrementar el ancho de banda de la señal para su posterior envío. Estas características se encontraron en el OPA860 el cual tiene un Buffer con ancho de banda aproximado de 1600 MHz (ver hoja de datos en el Anexo 1)

4.1 DISEÑO DEL EMISOR

El proceso de Modulación usando la técnica Spread Spectrum es relativamente sencillo, consta de hacer una operación lógica XOR de cada bit de datos con una secuencia Pseudo-aleatoria Barker de 11 bits. Todas las secuencias pseudo-aleatorias usadas en la transmisión de información mediante Spread Spectrum deben cumplir las propiedades mencionadas capitulo 2.

Para realizar el envío de la información ya modulada se utilizó la lógica de sincronización de bit de arranque, dicho bit tiene un cuarto del tiempo de duración de un bit de la pseudo-aleatoria.

Con base en las propiedades del código Barker se escogió el siguiente código 10110111000. En la figura 21 se ve esta secuencia PN, sola (figura 21B) y luego con el bit de sincronismo añadido (figura 21C). En la figura 22 se ve la señal de datos modulada con la secuencia Barker de 11 bits.

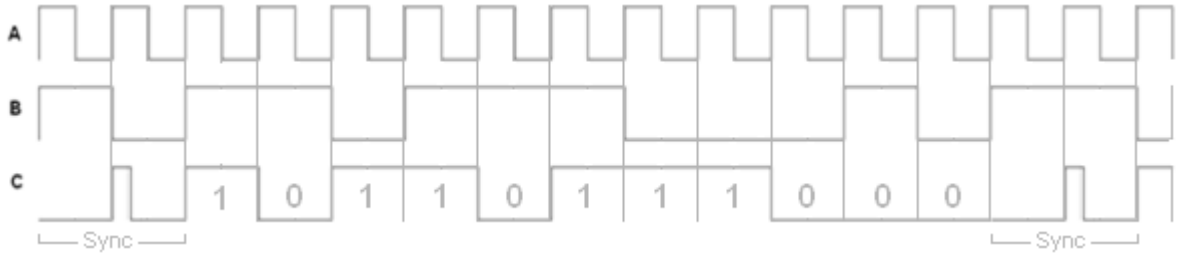


Figura 21. A. señal de reloj B. secuencia pseudo-aleatoria C. secuencia pseudo-aleatoria con bit de sincronismo.

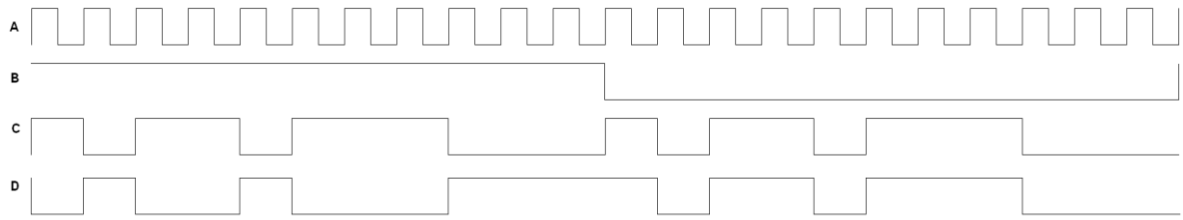


Figura 22. A. señal de reloj B. señal de datos C. secuencia pseudo-aleatoria D. operación lógica xor con la trama de datos.

En la figura 23 se observa la lógica utilizada en el bloque del emisor mediante un diagrama de flujo.

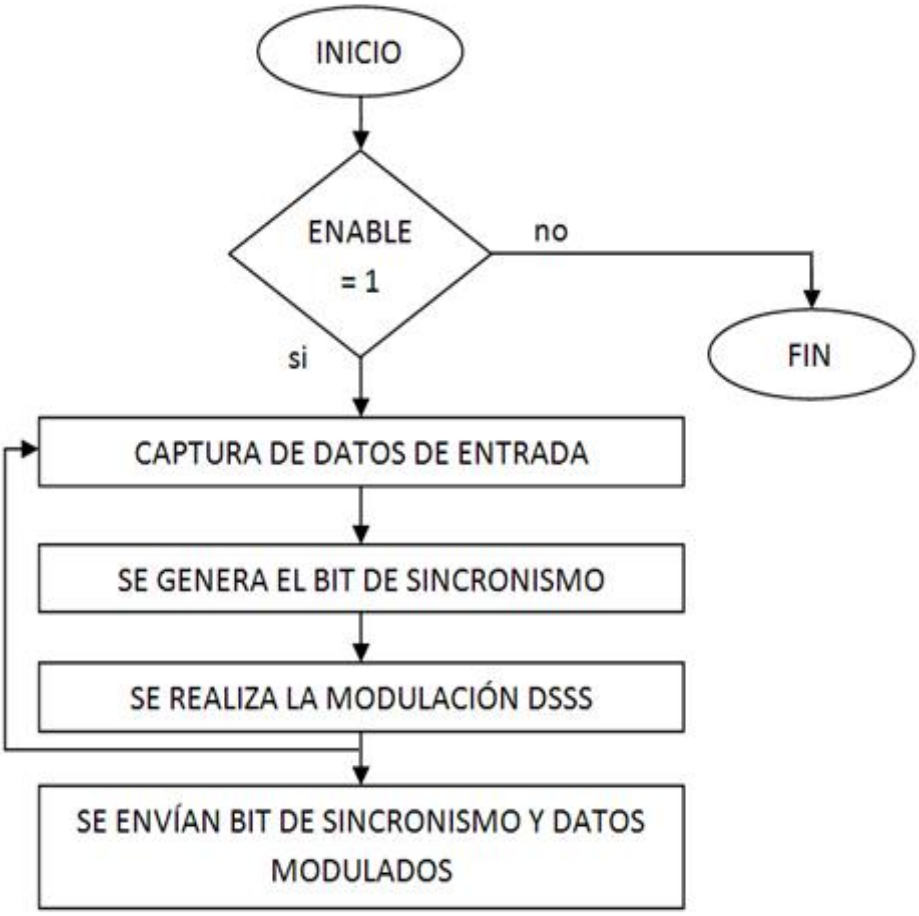


Figura 23. Diagrama de flujo Emisor.

4.1.1 Código fuente del emisor

```
-- Company:       Universidad Tecnológica de Pereira
-- Engineer:      John Alexis Colorado Aguirre
--                Alberto Luis Ramírez Hurtado
-- Create Date:    09:41:57 09/20/2010
-- Design Name:   Emisor Spread Spectrum
-- Module Name:    Emisor - Behavioral
-- Project Name:  Transmisión de información con la técnica Spread Spectrum
----------------------------------------------------------------------------------


---------------------------------------
--------Declaración de librerías---------
---------------------------------------
library IEEE;                          --
use IEEE.STD_LOGIC_1164.ALL;           --          A
use IEEE.STD_LOGIC_ARITH.ALL;          --
use IEEE.STD_LOGIC_UNSIGNED.ALL;       --
---------------------------------------


-------------------------------------------------------------------------------------------
----------------------Declaración de la entidad y asignación de puertos----------------------
-------------------------------------------------------------------------------------------
entity modulador is
       generic(n: integer := 416);
    Port ( Clk  : in STD_LOGIC;           -- Reloj
           clk2 : inout  std_logic;       -- Reloj de proceso de modulación y envío
           Datos : in  std_logic;         -- Datos de entrada                          B
           enable : in  std_logic;        -- Señal de habilitación
           Mod_out : out  std_logic);     -- Datos Modulados
end modulador;
-------------------------------------------------------------------------------------------
architecture Behavioral of modulador is
signal flag3 : std_logic := '0';          -- Señal para habilitar el envío de datos
signal flag2 : std_logic := '0';          -- Señal para habilitar el proceso de modulación                C
signal flag1 : std_logic := '1';          -- Señal para habilitar la captura de la señal de entrada
signal Aux_Mod_out : std_logic_vector(0 to 51); -- Variable auxiliar donde se cargan los datos modulados para su posterior tratamiento
signal Mask : std_logic_vector(0 to 51) := "000100000111111111111111111111111111111111111111111111"; -- Mascara para evitar datos parásitos.
signal Mod_out_vec : std_logic_vector(0 to 51); -- Buffer donde se almacenan los datos modulados para su posterior envío
shared variable AuxDatos : std_logic;          -- Variable donde se capturan los datos de entrada
shared variable AuxDatosOut : std_logic;       -- Variable auxiliar donde se carga el resultado de la xor de cada bit
shared variable i: integer := 0;               -- Contador para ordenar los datos en el vector Aux_Mod_out
```

Figura 24. Código emisor parte 1.

En la figura 24. **A**. Se declaran las librerías necesarias para la elaboración del código con sus respectivas instrucciones. En la parte **B** se declara la entidad la cual define las entradas y salidas del dispositivo a implementar así como también se define una variable genérica la cual es usada para la división de reloj, este dispositivo tiene como entradas la señal de reloj los datos de entrada y un pin de habilitación (encendido) y como salida, la modulación de salida. En la parte **C** está la declaración de las variables globales las cuales serán usadas en diferentes procesos.

En la figura 25 se ve la representación gráfica de la entidad del emisor.

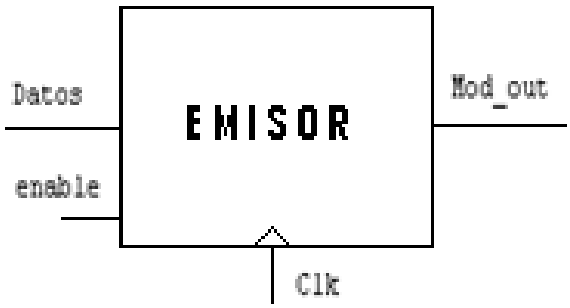


Figura 25. Diseño básico del emisor.

En la figura 26. A. se describe la división del reloj mediante un contador que cuenta de 0 a n/104 cambiando de un estado lógico alto a un estado lógico bajo y viceversa, este proceso se ejecuta continuamente durante todo el programa. En la parte B, se declara la secuencia pseudo-aleatoria (10110111000) también se declaran variables locales utilizadas en el proceso de modulación y envió de datos, en la parte c, se activa la bandera de captura de datos, luego de esto se activa la bandera de modulación para saltar a este proceso con los datos capturados.

La figura 27, es la parte del código más importante ya que en esta se realiza la sincronización, modulación y el envió de los datos al receptor. Primero se pone en un vector de salida la trama de bits “00010000”, esta cadena de bits representa el bit de sincronismo, luego se realiza la modulación de los datos haciendo la XOR bit a bit entre la Pseudo-aleatoria y cada dato de entrada, cargándolo así cada bit modulado en el vector de salida. En este punto el vector de salida ya tiene cargada la trama de sincronismo seguida de la señal ya modulada en Spread Spectrum, este vector de salida, antes de ser enviado al receptor pasa por una máscara la cual suprime posibles bits parásitos en la trama de sincronismo, de esta forma se asegura que el receptor entienda correctamente cuando iniciar el proceso de demodulación. Antes de enviar el vector de salida al receptor se activa el proceso de envió de datos y nuevamente se habilita la captura de datos, así, mientras se está enviando el vector de salida un nuevo vector de salida está siendo cargado.

```
begin
-------------------------------------------------------------------------------------------------


-------------------------------------------------------------------------------
--------------Proceso donde se hacen todas las divisiones del reloj-----------
-------------------------------------------------------------------------------
A:  process(clk)
    variable d: integer range 1 to n/104;     -- Contador para clk2
    begin
    if enable = '1' then                                    A
         if rising_edge (clk) then
            d := d + 1;
            if d = n/104 then                 -- Proceso de División de reloj
                d := 1;
                clk2 <= not clk2;
            end if;

         end if;
    else                                      -- Condición para enable desactivado
         d := 1;
    end if;
    end process A;
-------------------------------------------------------------------------------


-------------------------------------------------------------------------------------------------
---------------------Proceso donde se realiza la modulación y el envio de datos modulados-----
-------------------------------------------------------------------------------------------------
C:  process(clk2)
    variable Pni: std_logic_vector (0 to 10):= "10110111000";  -- Secuencia Pseudo-aleatoria
    variable count : integer := 2;      -- Contador el cual le da el ancho a cada bit de la pn
    variable j : integer;               -- Contador para el envío de datos
    variable k : integer;               -- Contador para llenar el vector Aux_Mod_out
    begin                                                                   B
    if enable = '1' then
         if rising_edge (clk2) then

-----------------------------------------------------------------------
--------------Se realiza la captura de datos de entrada----------------
-----------------------------------------------------------------------
            if flag1 = '1' then
               AuxDatos := Datos;
               flag1 <= '0';          -- Se desactiva la captura de datos   C
               flag2 <= '1';          -- Se activa la modulación
            end if;
-----------------------------------------------------------------------
```

Figura 26. Código emisor parte 2.

```
if flag2 = '1' then
   if i <= 18 then
      if i = 0 then
         Aux_Mod_out(i) <= '0';      --|
         i := i + 1;                 --|Esta es la trama
      elsif i = 1 then               --|de sincronismo
         Aux_Mod_out(i) <= '0';      --|en la cual se
         i := i + 1;                 --|agrega un bit
      elsif i = 2 then               --|de un tamaño
         Aux_Mod_out(i) <= '0';      --|menor a un bit
         i := i + 1;                 --|normal de la pn
      elsif i = 3 then               --|
         Aux_Mod_out(i) <= '1';      --|
         i := i + 1;                 --|el tamaño es de
      elsif i = 4 then               --|1/4 de bit de la pn
         Aux_Mod_out(i) <= '0';      --|
         i := i + 1;                 --|
      elsif i = 5 then               --|
         Aux_Mod_out(i) <= '0';      --|
         i := i + 1;                 --|
      elsif i = 6 then               --|
         Aux_Mod_out(i) <= '0';      --|
         i := i + 1;                 --|
      elsif i = 7 then               --|
         Aux_Mod_out(i) <= '0';      --|
         i := i + 1;
         k := 8;
      else
         if count = 3 then    -- Tamaño de cada bit = 4
            AuxDatosOut := AuxDatos xor Pni(i - 8); -- Se realiza la modulación Spread Spectrum
            i := i + 1;
            count := 0;
         end if;
         Aux_Mod_out(k) <= AuxDatosOut;
         k := k + 1;
         count := count + 1;
      end if;
      flag1 <= '0';  -- Se desactiva la captura de datos
   else
      Mod_out_vec <= Aux_Mod_out and Mask;
      flag3 <= '1';  -- Se activa el envío de los datos
      k := 8;
      i := 0;
      j := 0;
      flag1 <= '1';  -- Se activa la captura de datos
      flag2 <= '0';  -- Se desactiva la modulación de datos
      count := 2;
   end if;
end if;
```

Figura 27. Código emisor parte 3.

En la figura 28 se muestra el proceso de envió de datos el cual manda los datos de forma serial hacia el receptor, después de enviado se desactiva esta función y se deja a la espera de un nuevo vector de salida.

```
-------------------------------------------------------------------------
------------Se realiza el envío de los datos modulados----------------
-------------------------------------------------------------------------
            if flag3 = '1' then
               if j <= 51 then
               Mod_out <= Mod_out_vec(j);
               j := j + 1;
               flag3 <= '1';
               elsif j = 52 then
                  j := 0;
                  flag3 <= '0';  -- Se desactiva el envío de los datos
               end if;
            end if;
-------------------------------------------------------------------------

         end if;
      else
-------------------------------------------------------------
------------Se resetean todas las variables----------------
-------------------------------------------------------------
         i := 0;
         j := 0;
         count := 2;
         flag1 <= '0';
         flag2 <= '1';
         flag3 <= '0';
    end if;
   end process C;
end behavioral;
```

Figura 28. Código emisor parte 4.

4.2 DISEÑO DEL RECEPTOR

El proceso de demodulación en el receptor es el mismo proceso que se hace en el emisor cuando se modula la información. Basta con recoger los bits entrantes y hacer la misma operación lógica XOR con cada bit y la secuencia pseudo-aleatoria, cabe mencionar que el emisor y receptor deben tener la misma secuencia pseudo-aleatoria para que se lleve a cabo correctamente la comunicación entre ellos, (10110111000).

En la figura 29 se observa la lógica utilizada en el bloque del receptor mediante un diagrama de flujo.

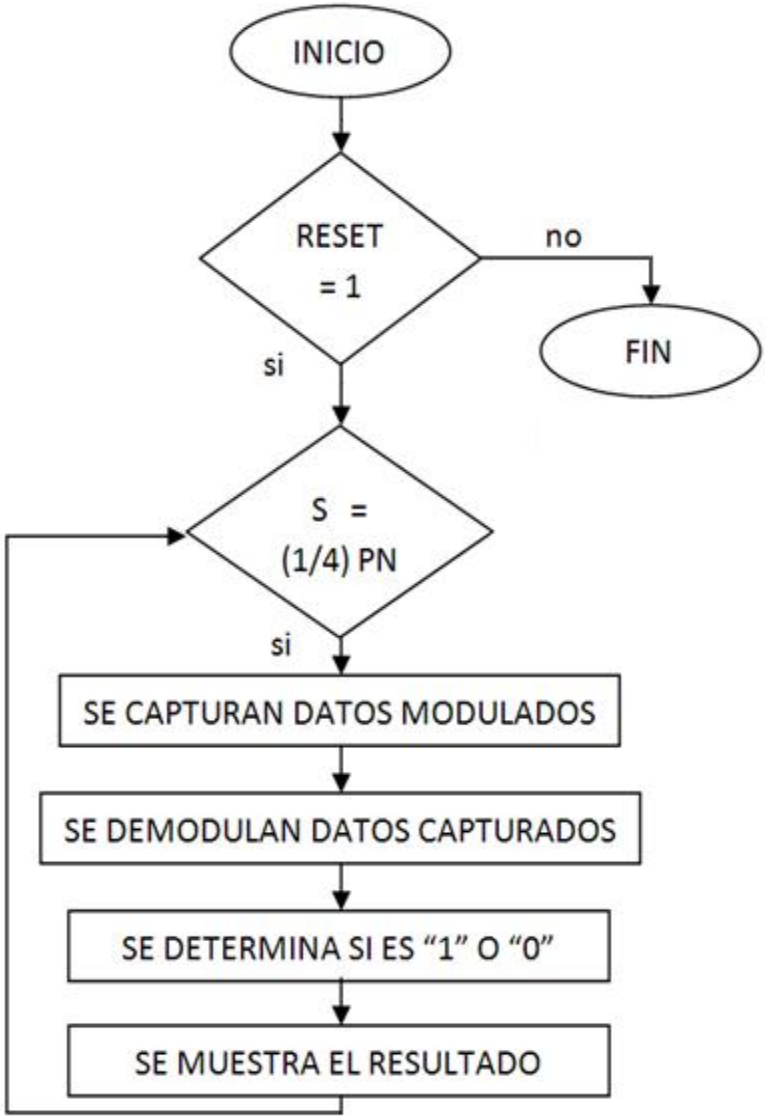


Figura 29. Diagrama de flujo Receptor.

```
-- Company:       Universidad Tecnológica de Pereira
-- Engineer:      John Alexis Colorado Aguirre
--                Alberto Luis Ramírez Hurtado
-- Create Date:    09:41:57 09/20/2010
-- Design Name:   Receptor Spread Spectrum
-- Module Name:   Demodulador - Behavioral
-- Project Name:  Transmisión de información con la técnica Spread Spectrum
----------------------------------------------------------------------------------


----------------------------------------
--------Declaración de librerías---------
----------------------------------------
library IEEE;                           --
use IEEE.STD_LOGIC_1164.ALL;            --        A
use IEEE.STD_LOGIC_ARITH.ALL;           --
use IEEE.STD_LOGIC_UNSIGNED.ALL;        --
----------------------------------------


--------------------------------------------------------------------------------------
-------------------Declaración de la entidad y asignación de puertos------------------
--------------------------------------------------------------------------------------
entity Demodulator is
        generic(n: integer := 416);       -- Reloj
    Port ( Mod_in : in  bit;              -- Entrada señal modulada                    B
           clk    : in std_logic;         -- Reloj interno
           pn_clk : inout std_logic;      -- Reloj de recepción y demodulación
           rst    : in bit;               -- Señal de reset
           control_mod : out  bit;        -- Señal de control de captura datos entrada
           Datos_out : out  bit);         -- Salida de datos demodulados
end Demodulator;
--------------------------------------------------------------------------------------


architecture Behavioral of Demodulator is
signal Y : bit_vector (10 downto 0);      -- Variable donde se carga la señal modulada de entrada
signal Aux: bit := '0';                   -- Contador de 1's de la señal demodulada          C
signal Mod_in_vec : bit_vector (10 downto 0);   -- Auxiliar donde se carga la variable Y
signal Aux_Data_out : bit_vector (10 downto 0); -- Variable donde se carga la señal luego de su demod
shared variable d: integer range 1 to n/104 := n/104;    -- Contador para la división de frecuencia.
```

.

Figura 30. Código receptor parte 1.

En la figura 30. **A**. se declaran las librerías necesarias para la elaboración del código con sus respectivas instrucciones. En la parte **B** se declara la entidad la cual define las entradas y salidas del dispositivo a implementar así como también se define una variable genérica la cual es usada para la división de reloj. Este dispositivo tiene como entradas la señal de reloj, los datos de entrada y un pin de habilitación (encendido) y como salida la modulación de salida. En la parte **C** está la declaración de las variables globales las cuales serán usadas en diferentes procesos. En la figura 31 se ve la representación gráfica de la entidad del receptor.

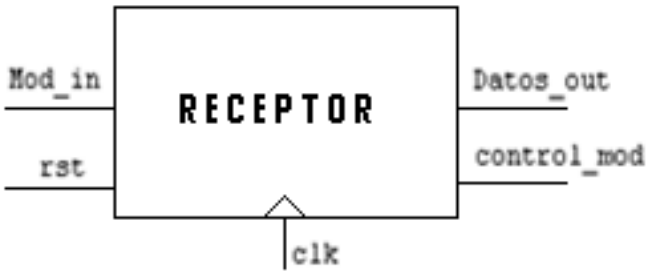


Figura 31. Diseño básico del receptor.

```
begin

-------------------------------------------------------
------Proceso donde se hace la división del reloj------
-------------------------------------------------------
A:  process(rst, clk)
    begin
    if rst = '1' then
       if rising_edge (clk) then
         d := d + 1;
         if d = n/104 then
            d := 1;
            pn_clk <= not pn_clk;
         end if;
       end if;
    else
        d := n/104;
    end if;
    end process A;
-------------------------------------------------------
```

Figura 32. Código receptor parte 2.

En la figura 32. Se describe la división del reloj mediante un contador que cuenta de 0 a n/104 cambiando de un estado lógico alto a un estado lógico bajo y viceversa, este proceso se ejecuta continuamente durante todo el programa.

En la figura 33. Se declaran variables locales utilizadas en el proceso de demodulación así como la secuencia pseudo-aleatoria la cual se le asigna un valor fijo “10110111000” esta secuencia PN tiene que ser la misma con la que se realizó el proceso de modulación para poder desencriptar los datos de manera correcta. Si hay un flanco de subida entonces empieza el proceso de detección de la señal de sincronismo el cual es llevado a cabo, contando la duración de cada “1” presente en el pin de recepción, si la duración es de 1 ciclo entonces ese es el bit de sincronismo activando posteriormente la captura de datos y demodulación.

```
-------------------------------------------------------------------------------------------
-------------Proceso donde se realiza la recepción de datos y la demodulación----------------
-------------------------------------------------------------------------------------------
B: process(rst, pn_clk)
    variable Pni: bit_vector (0 to 10):= "10110111000";  -- Secuencia Pseudoaleatoria
    variable i: integer range 0 to 11 := 0; -- Contador para llenar el vector Y
    variable j: integer;         -- Contador para hacer la demodulación
    variable cont: integer := 0; -- Contador para detctar el ancho de la señal de sincronismo
    variable flag2: bit := '1';  -- Bandera para activar el reconocimiento de la señal de sincronismo
    variable flag: bit := '0';   -- Bandera para activar la demodulación
    variable t: integer := 0;    -- Contador para leer por completo cada chip de la Pseudoaleatoria recibida
    variable h: integer := 0;    -- Contador de la señal demodulada bit a bit
    variable hi: integer := 0;   -- Contador de 1's de la señal demodulada
    begin
      if rst = '1' then
         if rising_edge (pn_clk) then

           if flag2 = '1' then
              if mod_in = '1' then    -- Medición del tamaño de cada bit recibido
                 cont := cont + 1;
                 flag := '0';
                 t := 0;
              elsif mod_in = '0' then
                 if cont = 1 or cont = 2 then  -- Reconocimiento de la señal de sincronismo
                    flag := '1';               --  | Activacion para la captura de
                    flag2 := '0';              --  |    datos y demodulación
                    cont := 0;
                    t := 0;
                 else
                    cont := 0;
                 end if;
              end if;
           end if;
```

Figura 33. Código receptor parte 3.

En la figura 34. está presente el proceso de captura y demodulación, capturando cada dato presente en el pin de recepción y cargándolo bit a bit en un vector de entrada, luego de cargar todo el vector de entrada se realiza la operación lógica XOR entre el vector de entrada y la secuencia Pseudo-aleatoria de la misma forma con la que se hizo en el emisor. Luego se evalúa el resultado de la demodulación para decir si la respuesta de la misma es un “1” o un “0”, para mostrarlo en el pin de salida. Posteriormente se desactiva este proceso y se activa nuevamente el proceso de detección del bit de sincronismo figura 33.

```
   if flag = '1' then                  -- Se activa la demodulación

      if t = 2 then
         if (i < 11) then
            Y(i) <= Mod_in;           -- Se llena el vector con cada bit modulado
            control_mod <= Y(i);      -- Se observa lo que se está recibiendo
            i := i + 1;
         elsif i = 11 then
            Mod_in_vec <= Y;
            for j in 0 to 10 loop
               Aux_Data_out(j) <= Mod_in_vec(j) xor Pni(j);--Se realiza la demodulación
            end loop;
            for h in 0 to 10 loop
               Aux <= Aux_Data_out(h);
               if Aux = '1' then -- Se cuentan cuantos 1´s tiene la señal demodulada
                  hi := hi + 1;
               end if;
            end loop;
            if hi > 7 then
               Datos_out(k) <= '1';   --  | Dependiendo de cuantos
            elsif hi < 3 then         -- -|       1´s tenga
               Datos_out(k) <= '0';   --  |  se decide si es 1 o 0
            end if;
            i := 0;
            flag := '0';              --  | Activacion del reconocimiento del
            flag2 := '1';             --  |       bit de sincronismo
            cont := 0;
            hi := 0;
            h := 0;

         end if;
         t := 0;
      else
         t := t + 1;
      end if;

   end if;
end if;
```

Figura 34. Código receptor parte 4.

En la figura 35. Cuando el pin RST está en “0” se reinician todas las variables a sus estados por defecto.

```
      else
-------------------------------------------------------------------------------
---Se resetean todas las variables y se dejan a la espera de nuevos datos---
-------------------------------------------------------------------------------
         i := 0;
         Mod_in_vec <= "00000000000";
         Y <= "00000000000";
         Aux_Data_out <= "00000000000";
         Datos_out <= '0';
         control_mod <= '0';
         t := 0;
         flag := '0';
         flag2 := '1';
         cont := 0;
         h := 0;
         hi := 0;
      end if;
    end process B;

end Behavioral;
-------------------------------------------------------------------------------
```

Figura 35. Código receptor parte 5.

4.3 SINCRONIZACIÓN

Una de las problemáticas más grandes que se tienen en el área de comunicaciones es la sincronización entre emisor y receptor, dado que cuando no hay una correcta sincronización en los datos enviados, el receptor los interpreta de manera errónea con, lo cual el proceso de comunicación puede verse afectado ya que las tramas se demodulan sin tener un orden o sin saber que bit es el que debe ser procesado inicialmente, generando datos inconsistentes y provocando el fallo en el sistema.

Para solucionar este problema y hacer que los datos enviados por el emisor se entendieran de manera ordenada por el receptor y lograr que el mismo entendiera en qué momento debía empezar con la demodulación, se agregó un bit de sincronismo el cual tiene una duración de 1/4 de un bit de la pseudo-aleatoria (figura 36).

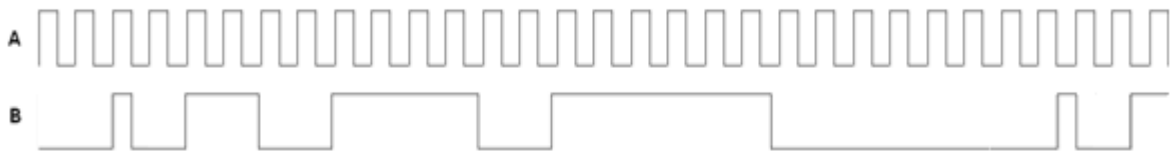


Figura 36. A. señal de reloj B. secuencia pseudo-aleatoria con bit de sincronismo (t = 1/4).

4.4 IMÁGENES ENTORNO DE DESARROLLO Y PROTOTIPO

En las figuras 37 y 38 se muestra el modulador y demodulador implementados en las tarjetas FPGA Spartan 3E, en la figura 37 se observa el proceso de comunicación donde el modulador y el demodulador están conectados mediante cable UTP de 20 cm, y en la figura 38 están conectados mediante par de cobre de 3 m.

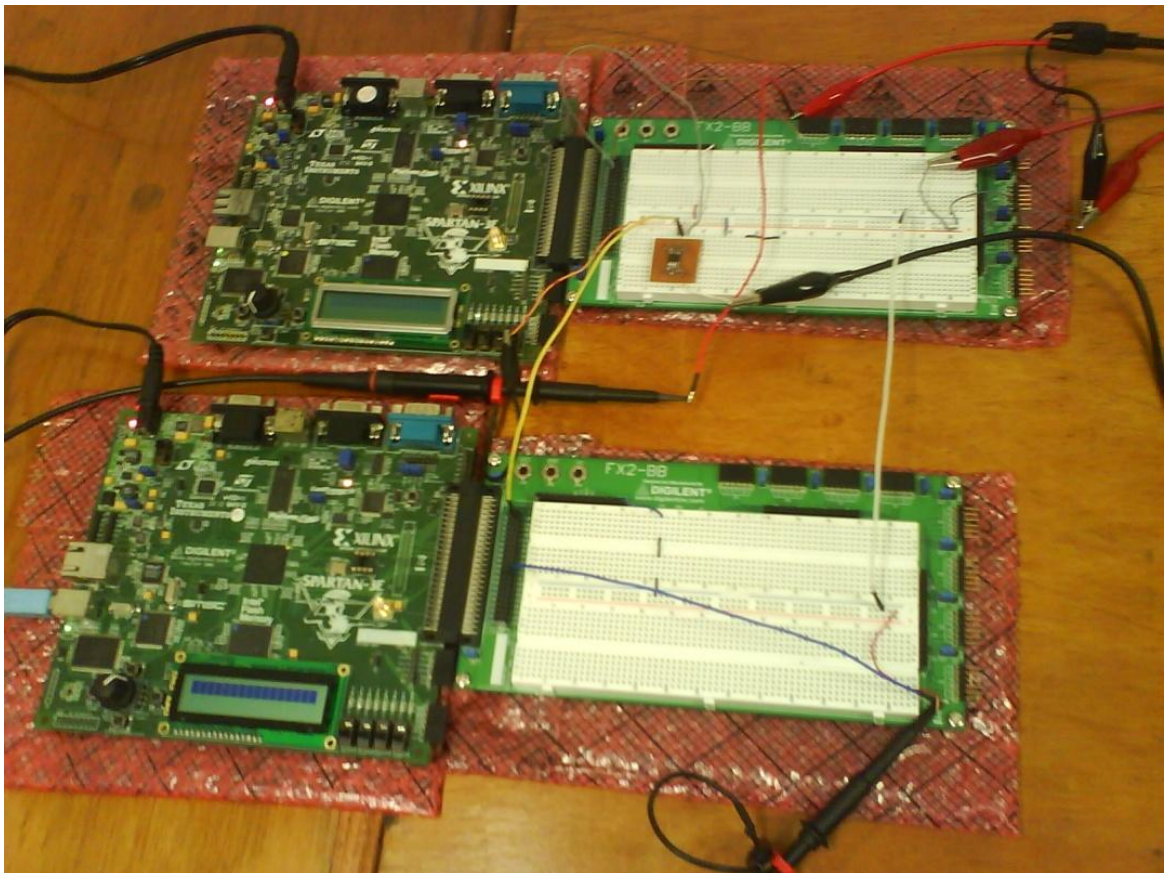


Figura 37. Modulador y demodulador implementados en la tarjeta FPGA Spartan 3E conectados mediante cable UTP de 20 cm.

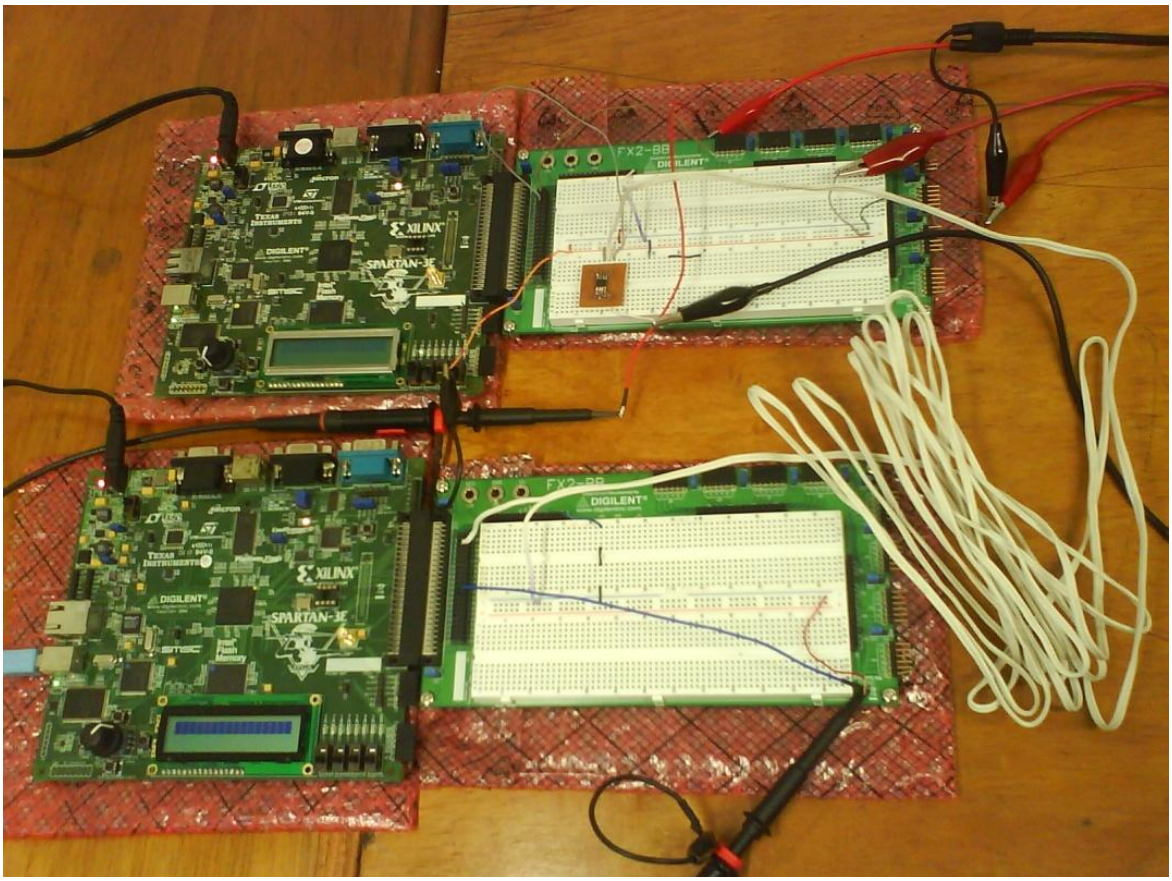


Figura 38. Modulador y demodulador implementados en la tarjeta FPGA Spartan 3E conectados mediante par de cobre de 3 m.

En las figuras 39 y 40 se muestra el entorno de desarrollo en el cual se llevó a cabo el desarrollo de este proyecto, en estas imágenes se observan los diferentes equipos necesarios para la realización de las pruebas de funcionamiento, entre ellos está: Osciloscopio, generador de señales, fuente DC y computador de escritorio con el software Xilinx Ise 9.2i.

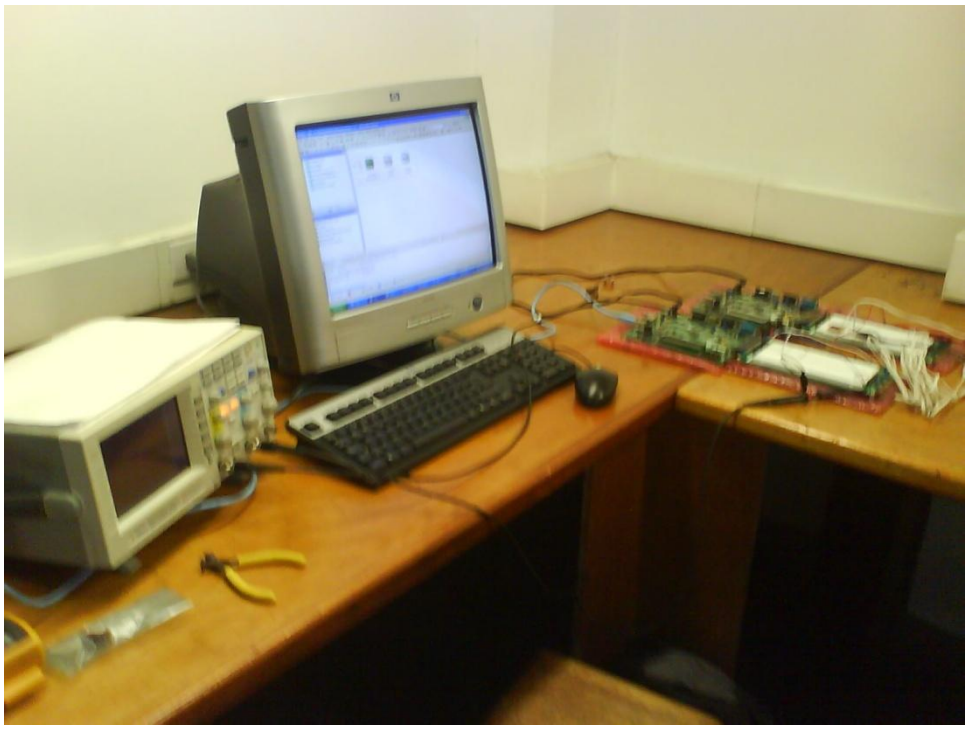


Figura 39. Entorno de desarrollo 1.

Figura 40. Entorno de desarrollo 2.

5. RESULTADOS

El objetivo general de este proyecto fue el diseño de un sistema de comunicaciones usando la modulación spread Spectrum en secuencia directa sobre FPGA, los resultados obtenidos se muestran a continuación.

Se logró la transmisión de una onda cuadrada a una frecuencia máxima de 100 KHz (Figuras 69, 70, 91 y 92). Se realizaron pruebas en dos etapas las cuales difieren entre sí con el tipo de cable por el cual se hace la comunicación, la etapa 1 es con cable UTP categoría No. 3 de 20 centímetros de longitud y la segunda etapa es con par de cobre de 3 metros, iniciando por 1 KHz como frecuencia mínima hasta llegar al límite. A continuación las pruebas con el cable UTP de 20 centímetros.

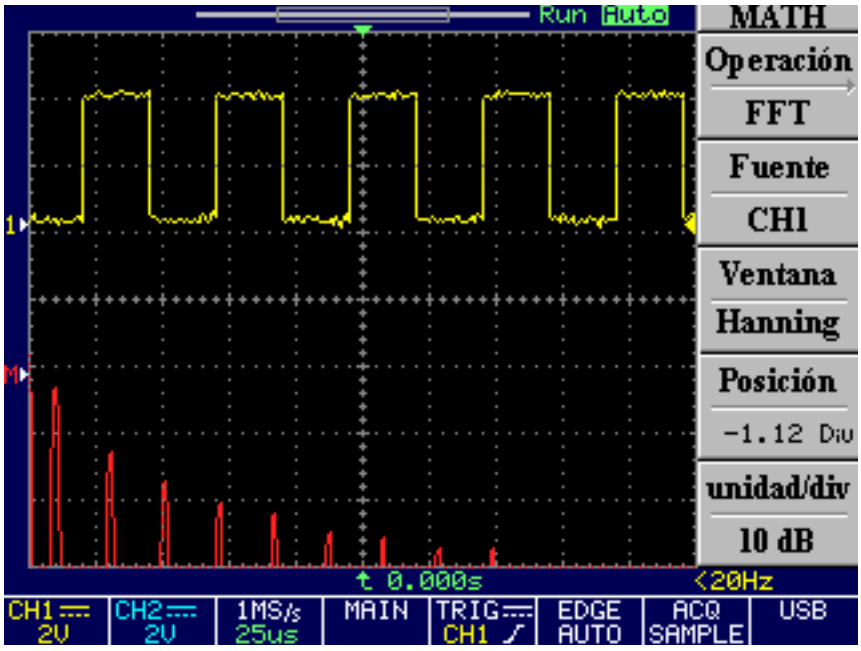


Figura 41. Señal de entrada (Onda cuadrada de 20 KHz - amarilla). Espectro (rojo).

En la figura 41 se observa la señal de entrada en color amarillo y en color rojo se observa su respectivo espectro, de la misma manera en la figura 43 en color amarillo la señal modulada en el canal de comunicaciones y en color rojo, se aprecia el espectro de la señal modulada notando que se cumple la teoría que plantea la técnica Spread Spectrum, ensanchando las componentes en frecuencia y reduciendo la amplitud de los picos.

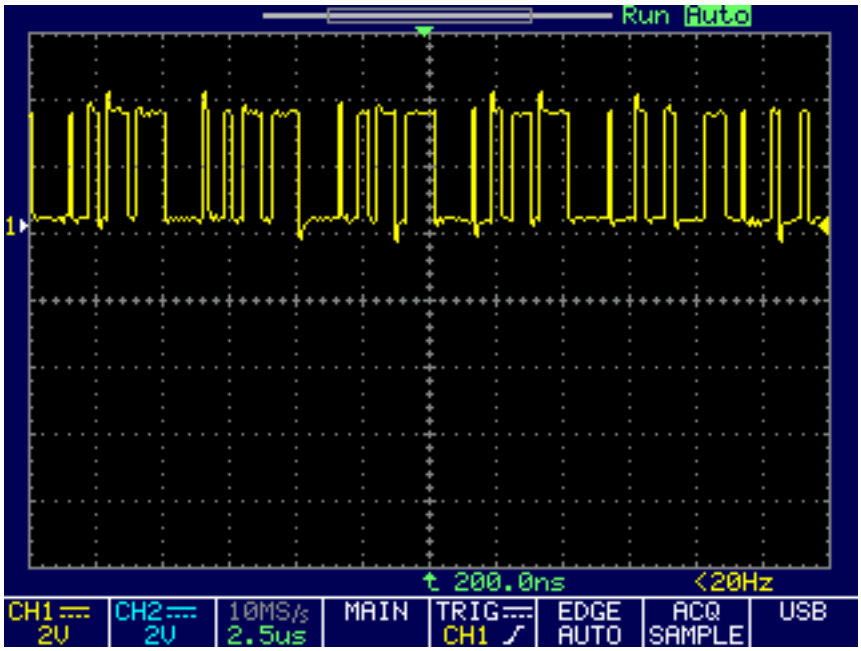


Figura 42. Señal modulada con bit de sincronismo a una frecuencia de 20 KHz.

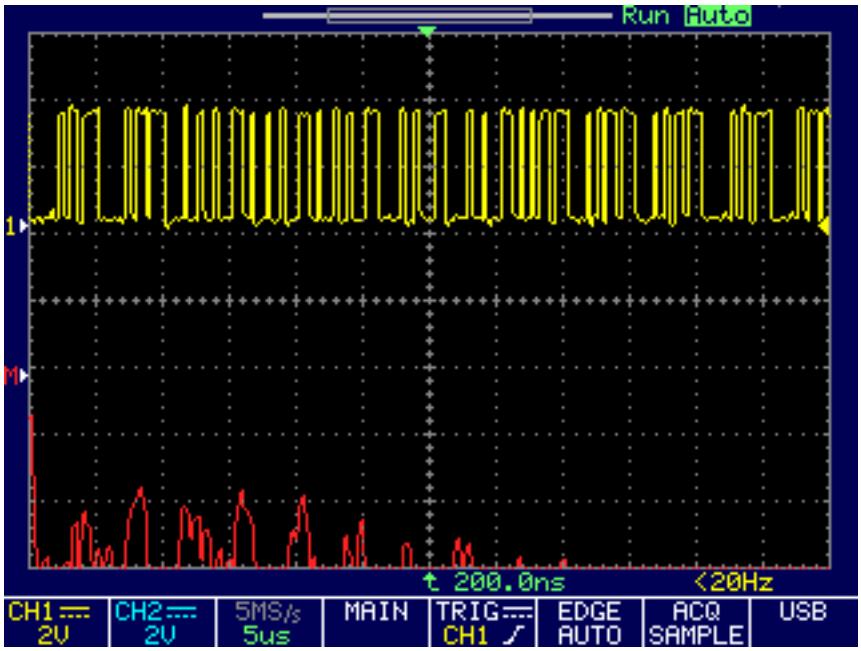


Figura 43. Señal modulada con bit de sincronismo a una frecuencia de 20 KHz (amarillo). Espectro (rojo).

En la figura 42 se puede apreciar la secuencia psudo-aleatoria (PN) con el bit de sincronismo presente en el canal de comunicaciones, se sabe que la PN es

10110111000, observandose esta secuencia 4 veces consecutivas y luego seguida por una secuencia parecida a lo siguiente 01001000111 la cual es la PN negada producto de la operación lógica XOR realizada entre la PN y un estado lógico alto (“1”).

Como se dijo anteriormente se realizaron pruebas con una onda cuadrada de entrada con frecuencias desde 1 KHz hasta 100 KHz observando la señal de entrada (antes del proceso), la señal de salida (luego de pasar por el proceso de comunicación) y el espectro de ambas, (Figura 44 hasta la Figura 67) observando que para frecuencias bajas los espectros de las señales de entrada y salida son iguales y que a partir mas o menos de 30 KHz (Figura 56 y Figura 57) se empieza a notar diferencia entre ambos, pero igualmente la diferencia no es mucha, solo unos pocos armónicos de baja amplitud, notando que las señales de entrada y salida no tienen mucha diferencia, diferencia que se empieza a percibir a partir de los 45 KHz (Figura 60 y Figura 61) cuando se observan algunos bits con un periodo más o menos largo que otros en la señal de salida y con un espectro de salida que difiere bastante de la señal de entrada, este efecto se ve también en las Figuras 64 y 65.

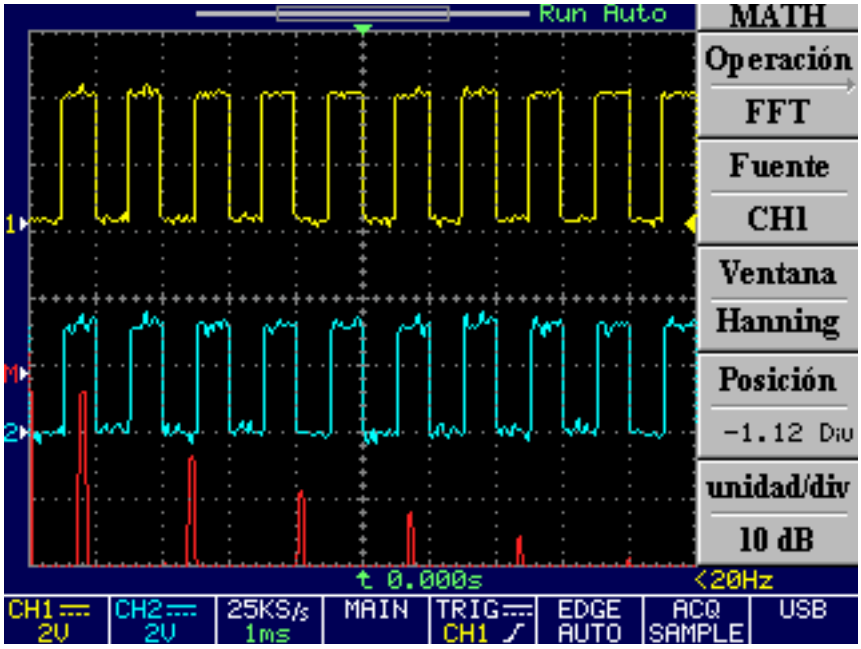


Figura 44. Señal de entrada (Onda cuadrada de 1 KHz – amarilla). Señal de salida (Onda cuadrada de 1 KHz – azul). Espectro de la señal de entrada (rojo).

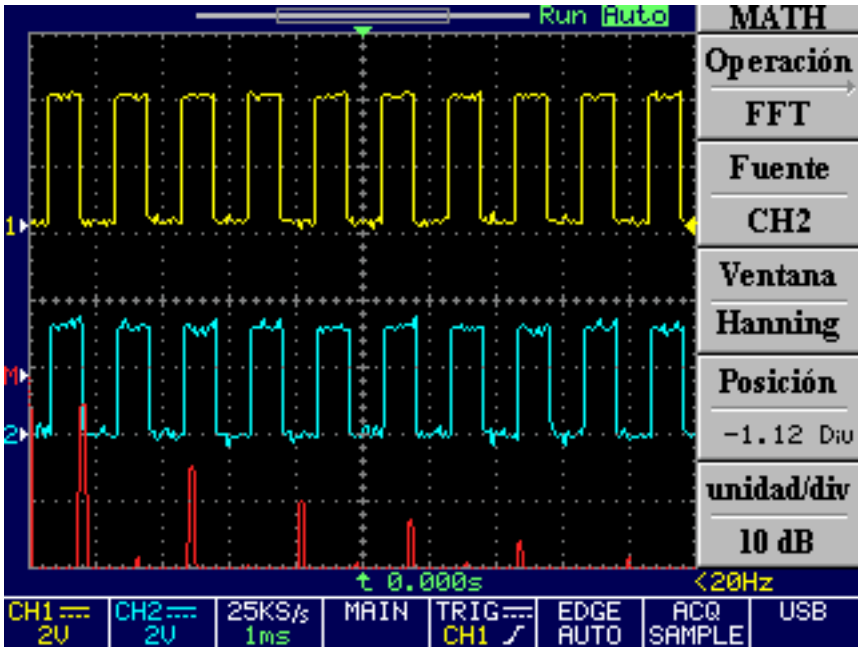


Figura 45. Señal de entrada (Onda cuadrada de 1 KHz – amarilla). Señal de salida (Onda cuadrada de 1 KHz – azul). Espectro de la señal de salida (rojo).

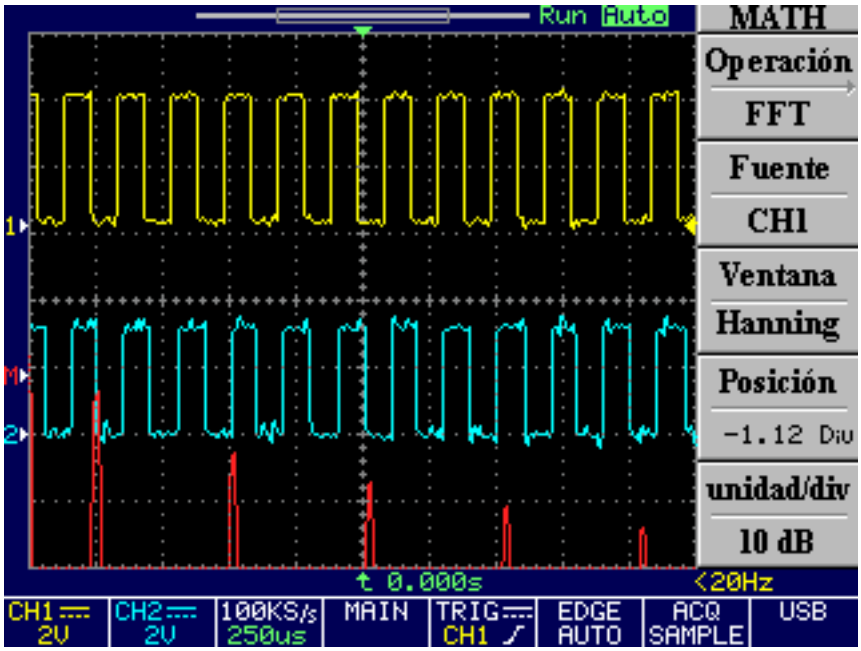


Figura 46. Señal de entrada (Onda cuadrada de 5 KHz – amarilla). Señal de salida (Onda cuadrada de 5 KHz – azul). Espectro de la señal de entrada (rojo).

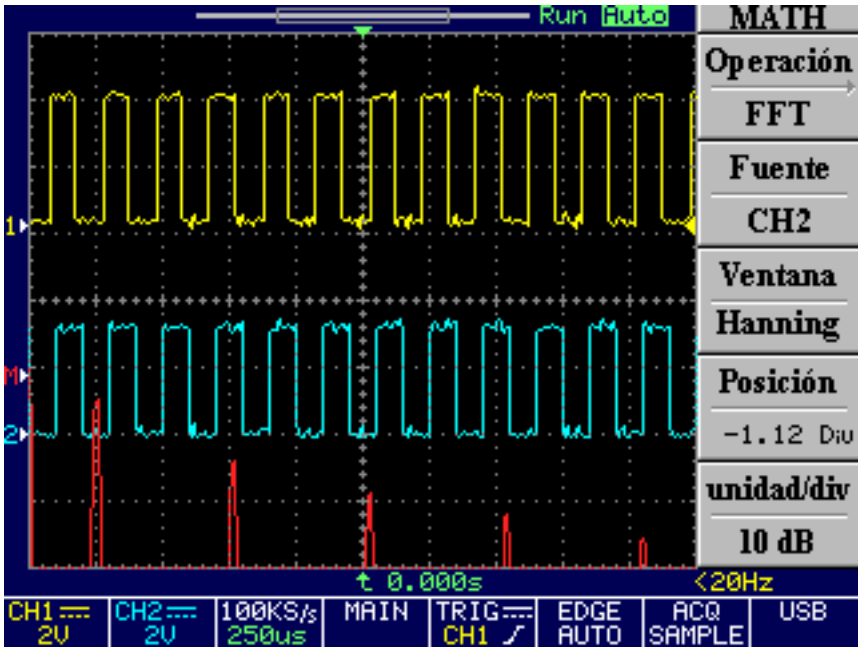


Figura 47. Señal de entrada (Onda cuadrada de 5 KHz – amarilla). Señal de salida (Onda cuadrada de 5 KHz – azul). Espectro de la señal de salida (rojo).

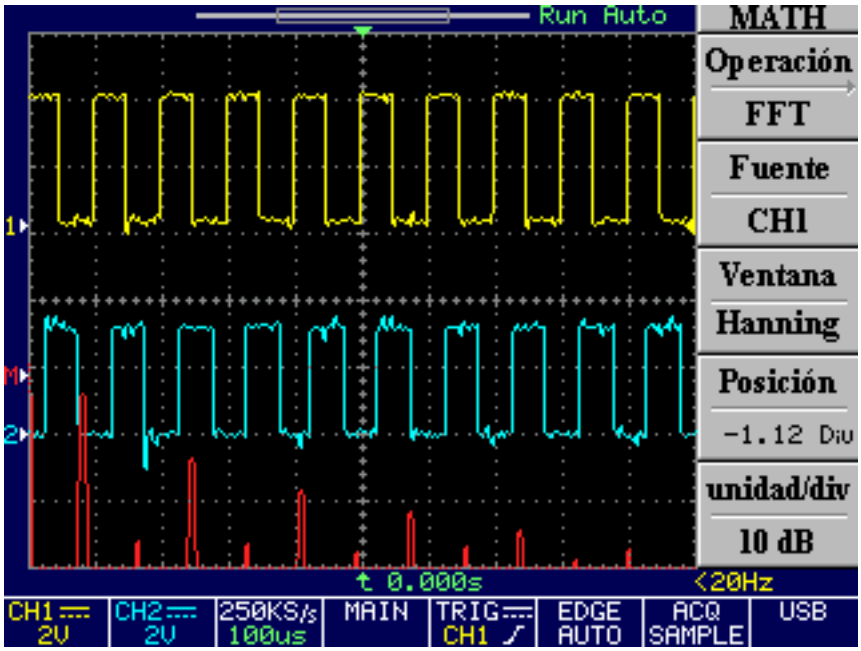


Figura 48. Señal de entrada (Onda cuadrada de 10 KHz – amarilla). Señal de salida (Onda cuadrada de 10 KHz – azul). Espectro de la señal de entrada (rojo).

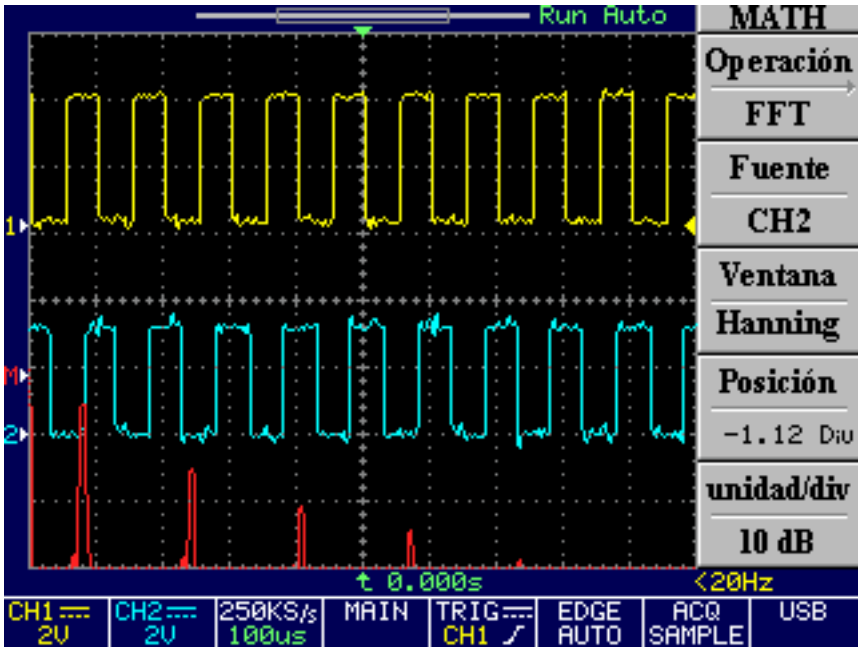


Figura 49. Señal de entrada (Onda cuadrada de 10 KHz – amarilla). Señal de salida (Onda cuadrada de 10 KHz – azul). Espectro de la señal de salida (rojo).

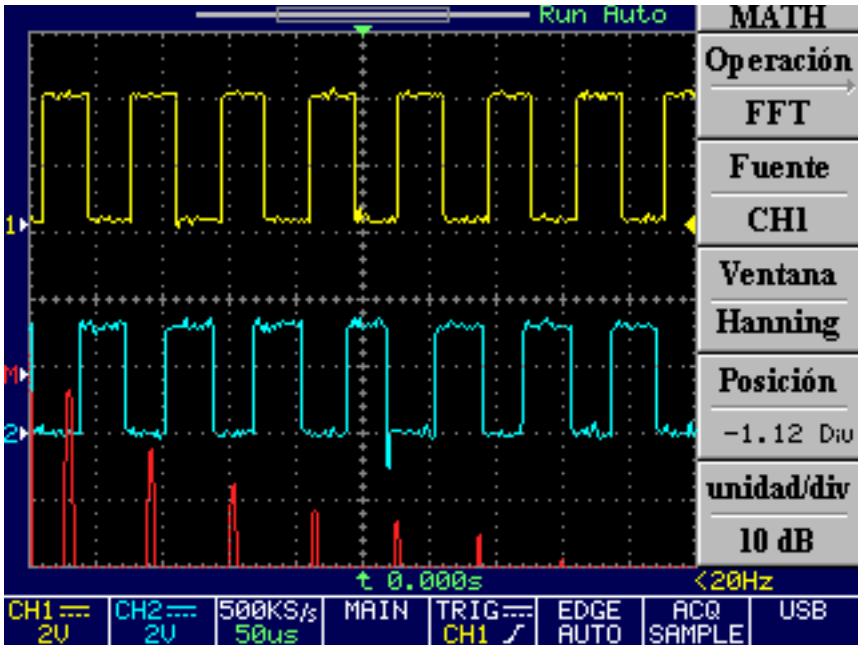


Figura 50. Señal de entrada (Onda cuadrada de 15 KHz – amarilla). Señal de salida (Onda cuadrada de 15 KHz – azul). Espectro de la señal de entrada (rojo).

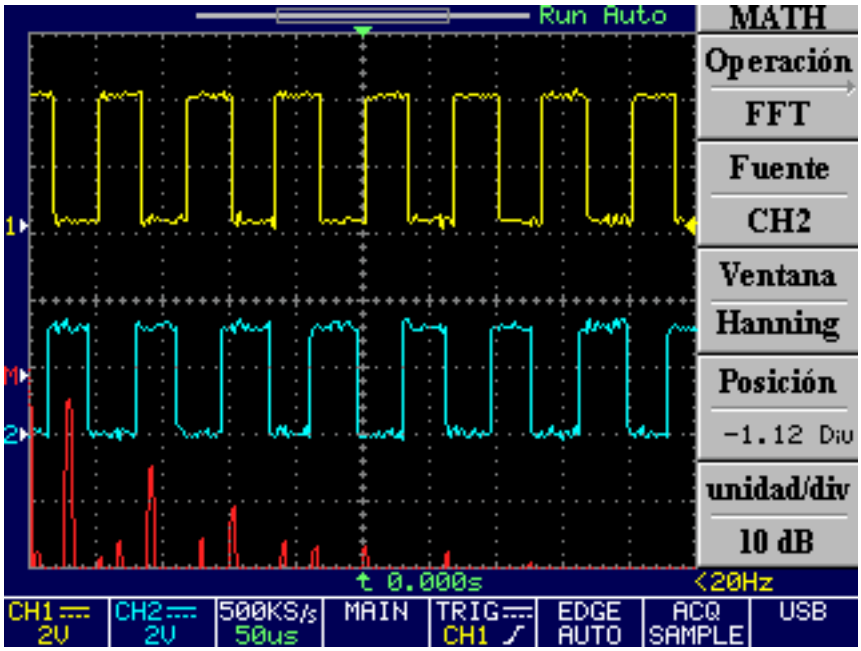


Figura 51. Señal de entrada (Onda cuadrada de 15 KHz – amarilla). Señal de salida (Onda cuadrada de 15 KHz – azul). Espectro de la señal de salida (rojo).

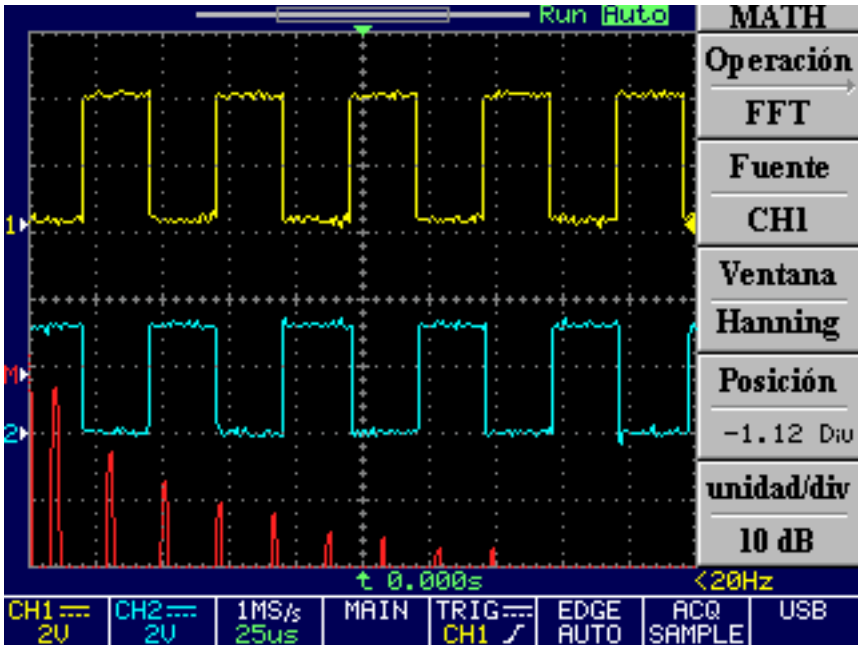


Figura 52. Señal de entrada (Onda cuadrada de 20 KHz – amarilla). Señal de salida (Onda cuadrada de 20 KHz – azul). Espectro de la señal de entrada (rojo).

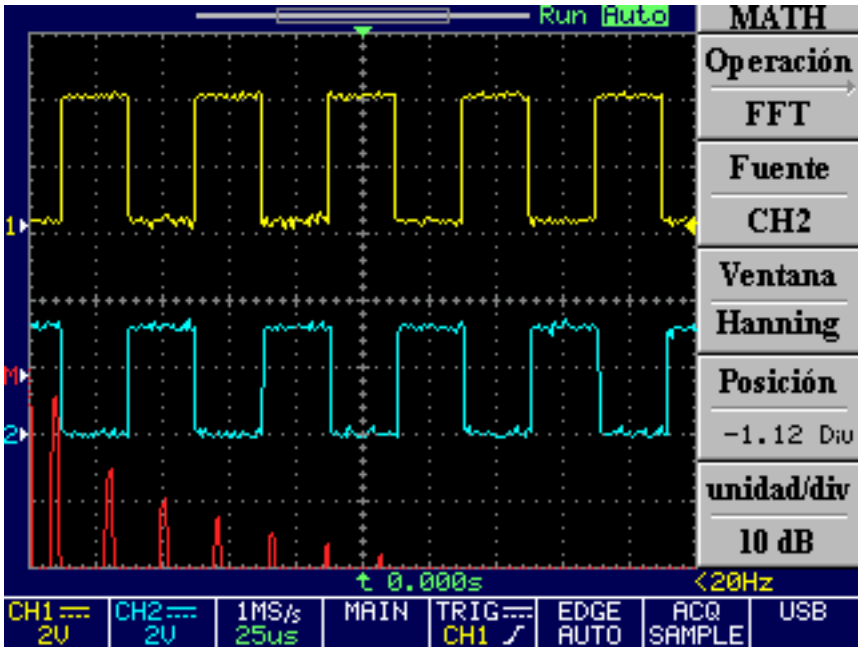


Figura 53. Señal de entrada (Onda cuadrada de 20 KHz – amarilla). Señal de salida (Onda cuadrada de 20 KHz – azul). Espectro de la señal de salida (rojo).

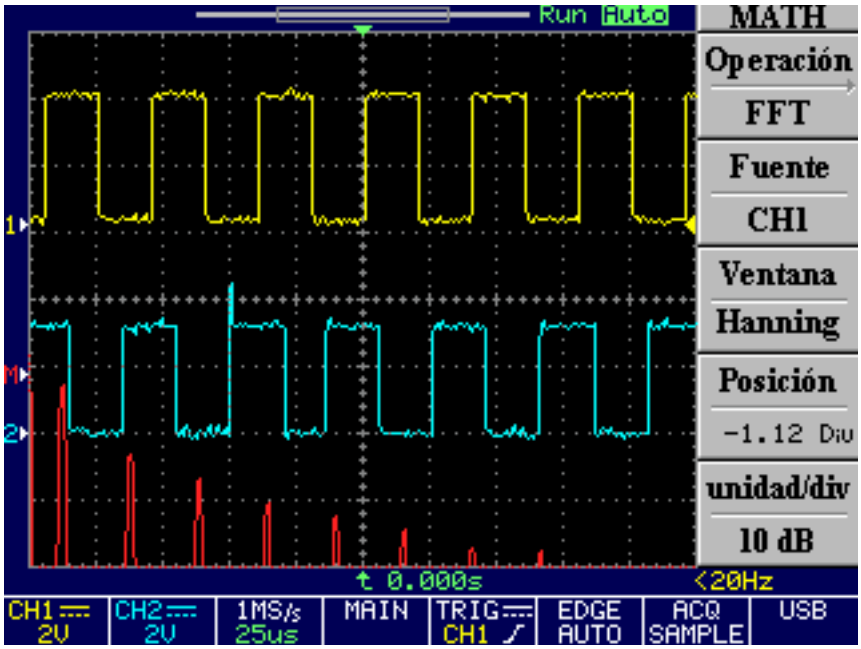


Figura 54. Señal de entrada (Onda cuadrada de 25 KHz – amarilla). Señal de salida (Onda cuadrada de 25 KHz – azul). Espectro de la señal de entrada (rojo).

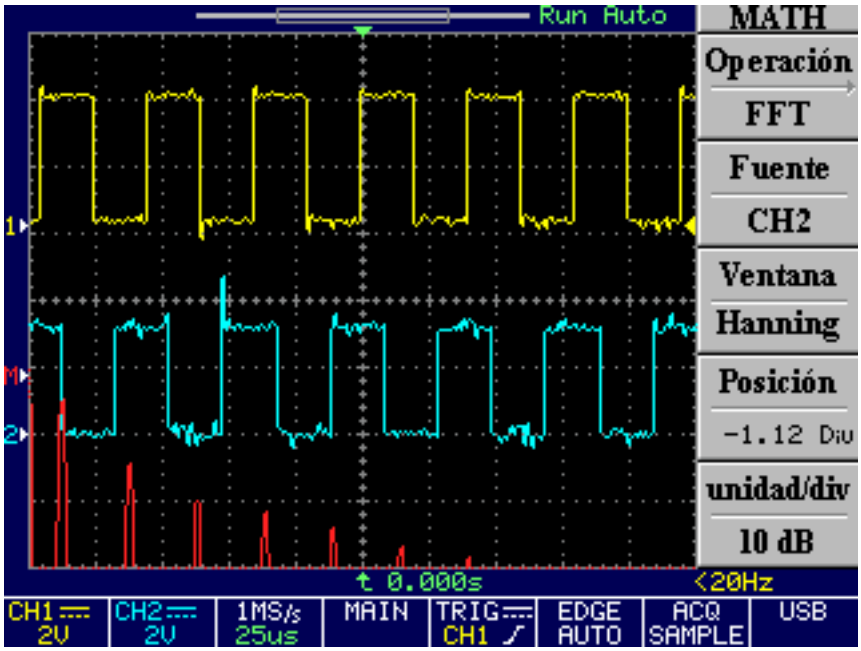


Figura 55. Señal de entrada (Onda cuadrada de 25 KHz – amarilla). Señal de salida (Onda cuadrada de 25 KHz – azul). Espectro de la señal de salida (rojo).

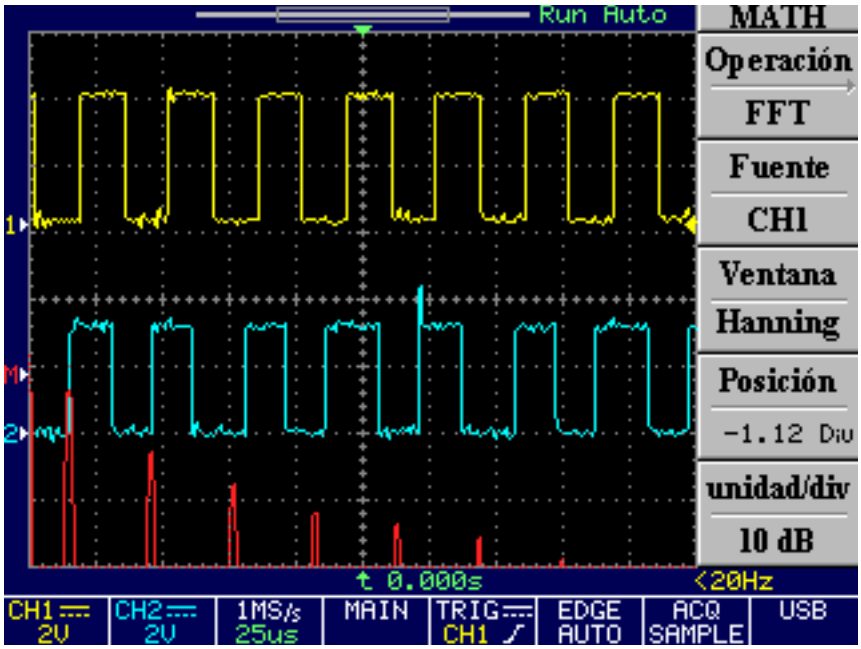


Figura 56. Señal de entrada (Onda cuadrada de 30 KHz – amarilla). Señal de salida (Onda cuadrada de 30 KHz – azul). Espectro de la señal de entrada (rojo).

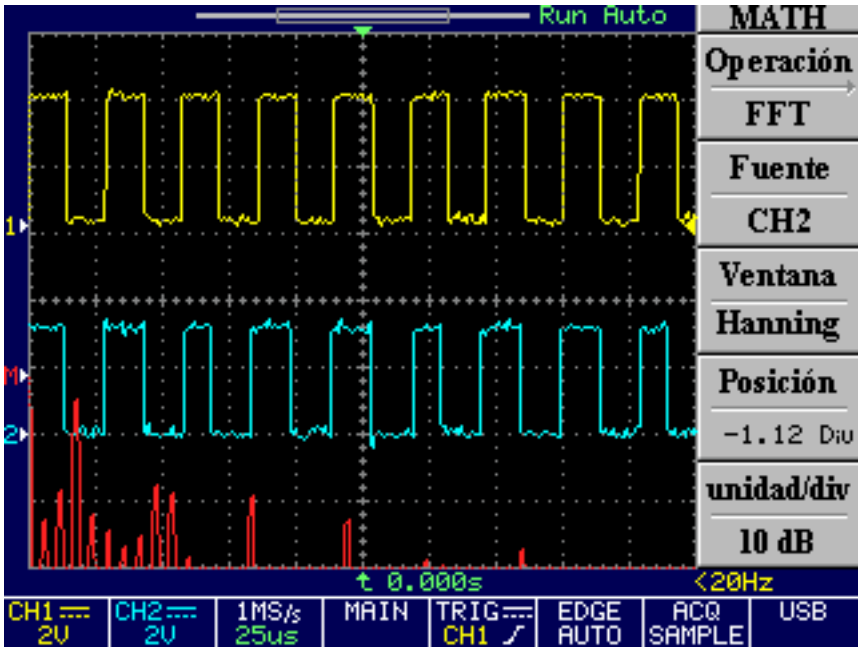


Figura 57. Señal de entrada (Onda cuadrada de 35 KHz – amarilla). Señal de salida (Onda cuadrada de 35 KHz – azul). Espectro de la señal de salida (rojo).

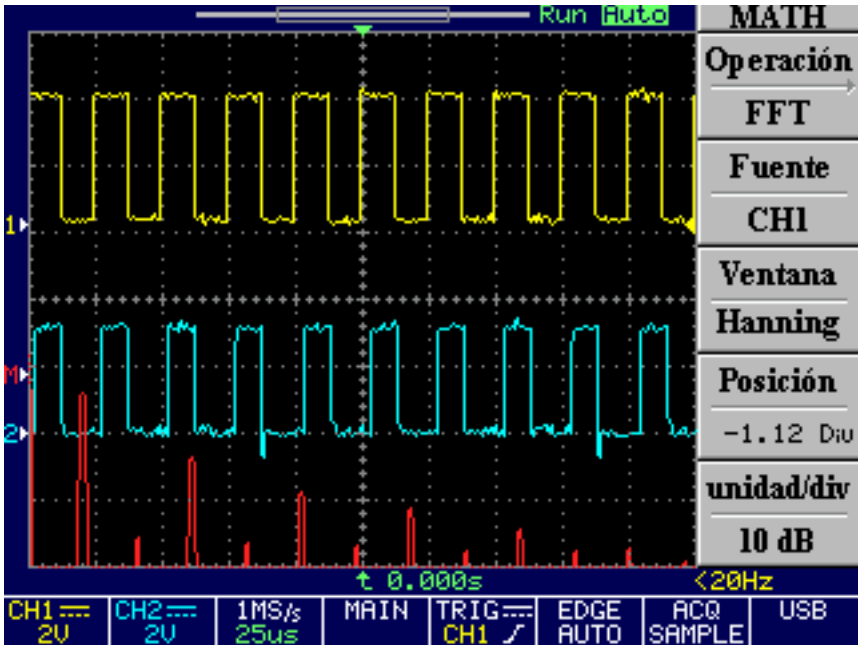


Figura 58. Señal de entrada (Onda cuadrada de 40 KHz – amarilla). Señal de salida (Onda cuadrada de 40 KHz – azul). Espectro de la señal de entrada (rojo).

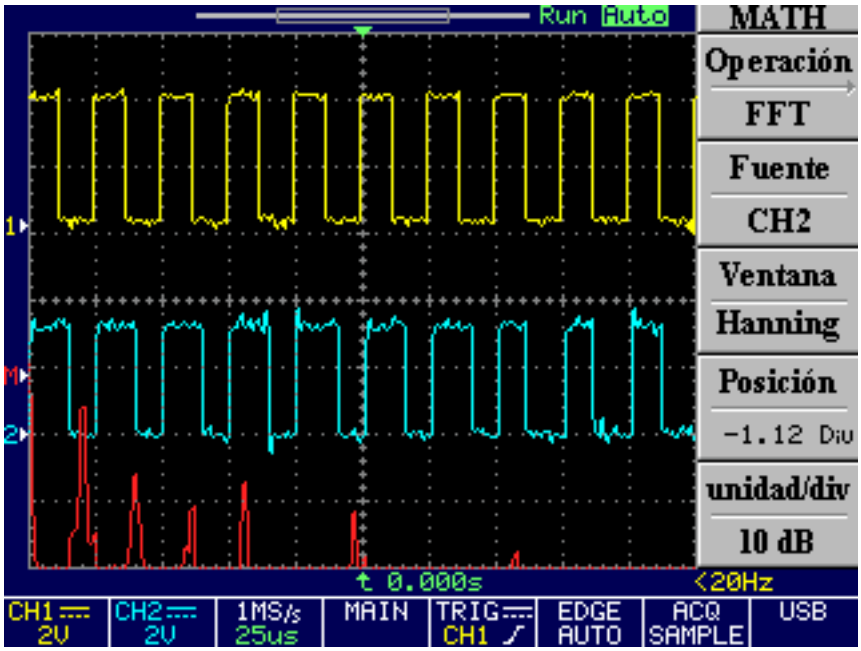


Figura 59. Señal de entrada (Onda cuadrada de 40 KHz – amarilla). Señal de salida (Onda cuadrada de 40 KHz – azul). Espectro de la señal de salida (rojo).

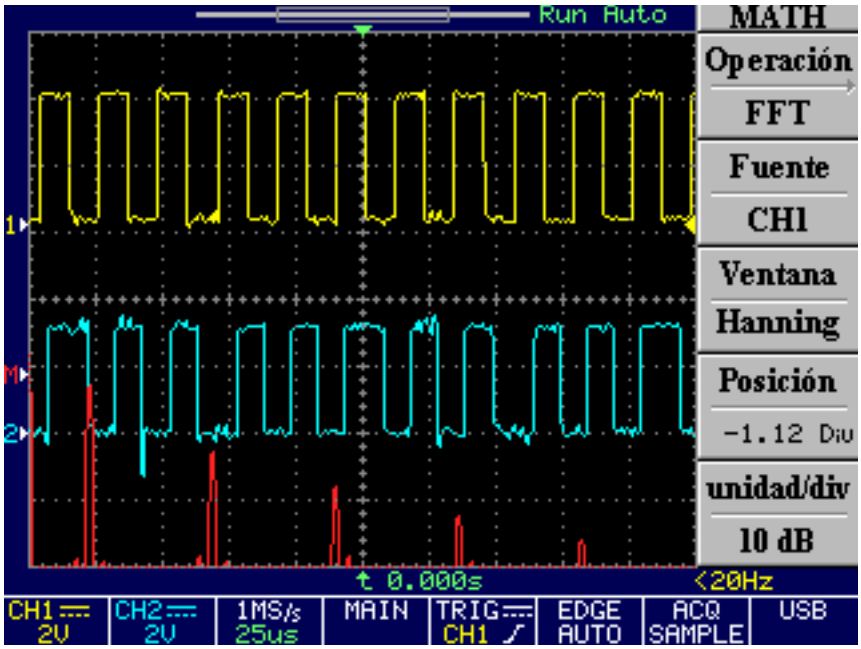


Figura 60. Señal de entrada (Onda cuadrada de 45 KHz – amarilla). Señal de salida (Onda cuadrada de 45 KHz – azul). Espectro de la señal de entrada (rojo).

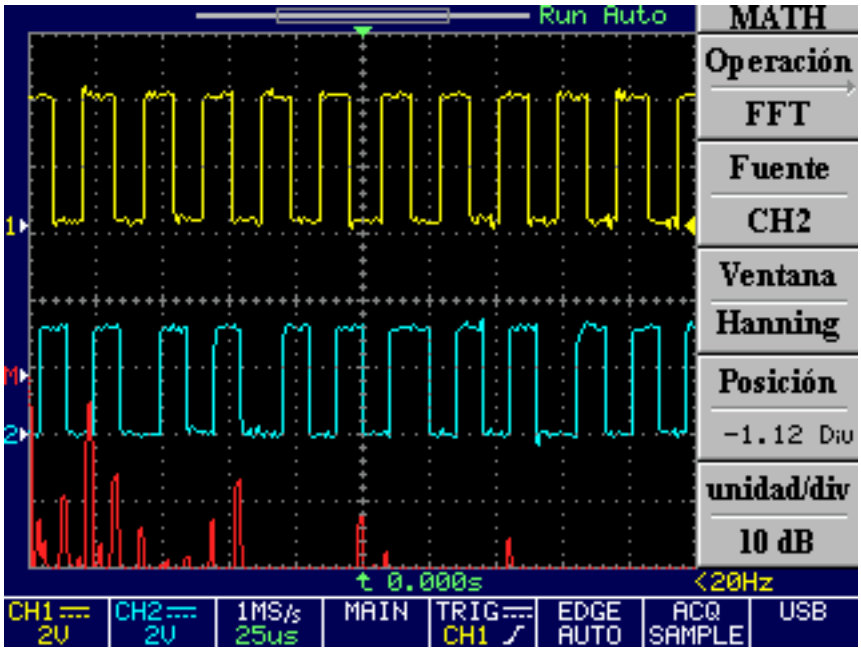


Figura 61. Señal de entrada (Onda cuadrada de 45 KHz – amarilla). Señal de salida (Onda cuadrada de 45 KHz – azul). Espectro de la señal de salida (rojo).

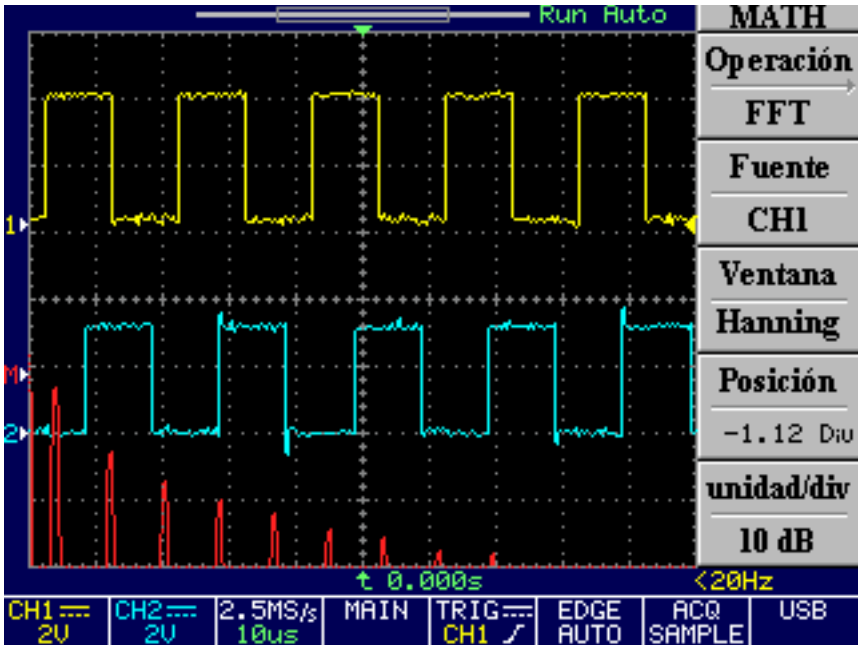


Figura 62. Señal de entrada (Onda cuadrada de 50 KHz – amarilla). Señal de salida (Onda cuadrada de 50 KHz – azul). Espectro de la señal de entrada (rojo).

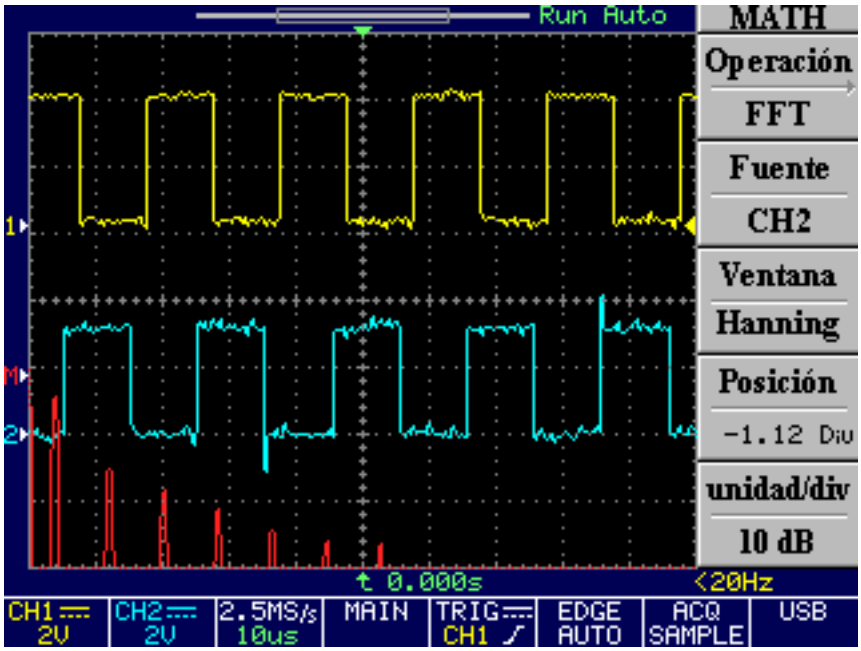


Figura 63. Señal de entrada (Onda cuadrada de 50 KHz – amarilla). Señal de salida (Onda cuadrada de 50 KHz – azul). Espectro de la señal de salida (rojo).

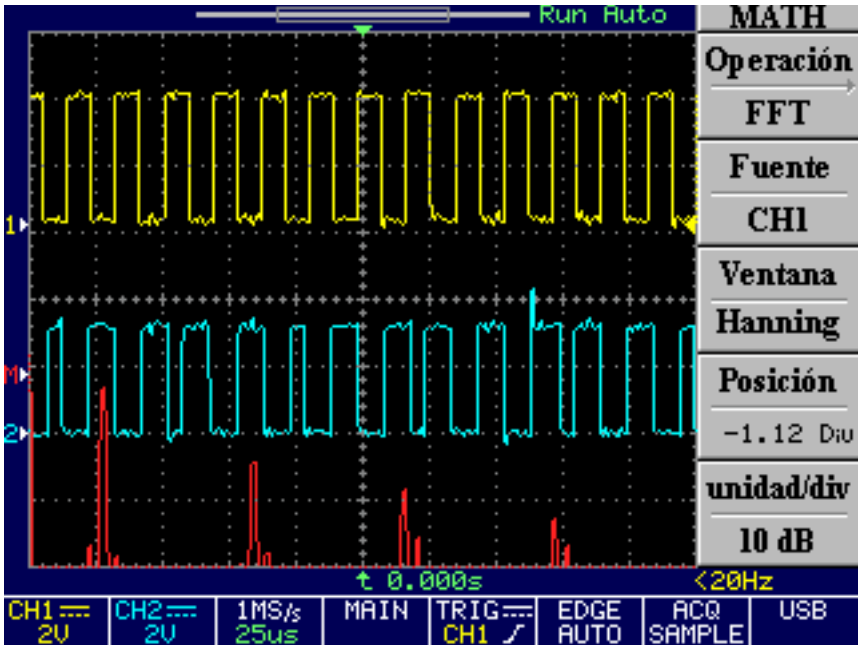


Figura 64. Señal de entrada (Onda cuadrada de 55 KHz – amarilla). Señal de salida (Onda cuadrada de 55 KHz – azul). Espectro de la señal de entrada (rojo).

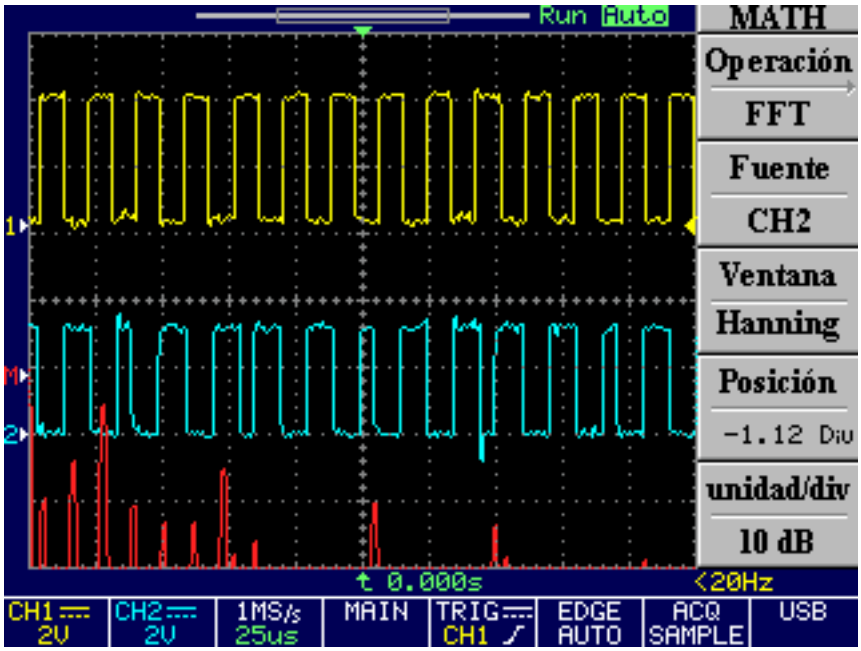


Figura 65. Señal de entrada (Onda cuadrada de 55 KHz – amarilla). Señal de salida (Onda cuadrada de 55 KHz – azul). Espectro de la señal de salida (rojo).

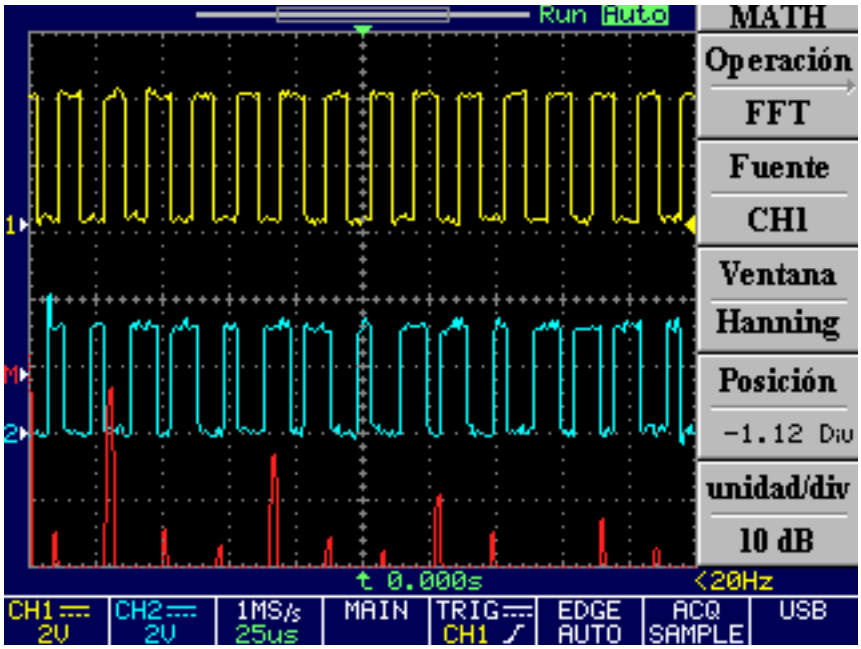


Figura 66. Señal de entrada (Onda cuadrada de 60 KHz – amarilla). Señal de salida (Onda cuadrada de 60 KHz – azul). Espectro de la señal de entrada (rojo).

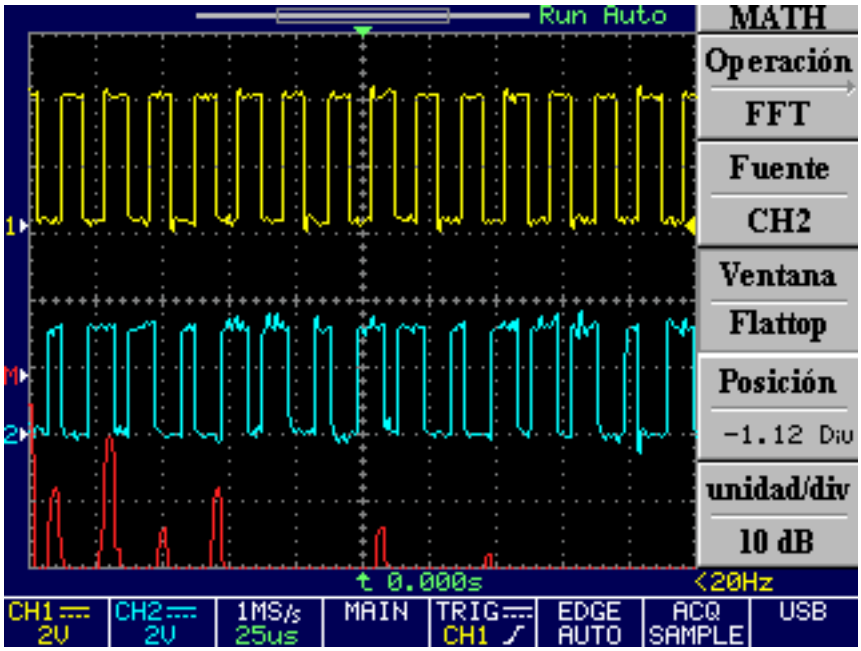


Figura 67. Señal de entrada (Onda cuadrada de 60 KHz – amarilla). Señal de salida (Onda cuadrada de 60 KHz – azul). Espectro de la señal de salida (rojo).

Según el teorema del muestreo, la reconstrucción aproximada de una señal podría ser matemáticamente posible si la tasa de muestreo es superior al doble de su frecuencia.

Por tal principio, el algoritmo de modulación lo que básicamente hace es tomar dos o más muestras de cada bit entrante, para poder tener una buena reconstrucción de la señal, evitando así que al perderse una muestra se pierda todo un bit de información.

Como se muestra en la figura 68, se están tomando dos muestras por cada bit de datos, teniendo en cuenta que cada muestra está precedida por un bit de sincronismo, a medida que la frecuencia de la señal disminuye, el número de muestras aumenta, por lo cual, cuando la frecuencia de la señal entrante supera los los 90 KHz aproximadamente, la señal de salida presentará errores, ya que si se pierde un bit en el proceso de comunicación es un bit completo el que se está perdiendo de la señal de entrada.

Se observa en la figura 69, que por cada bit de datos se está tomando una muestra ya que la frecuencia de entrada es de 100 KHz.

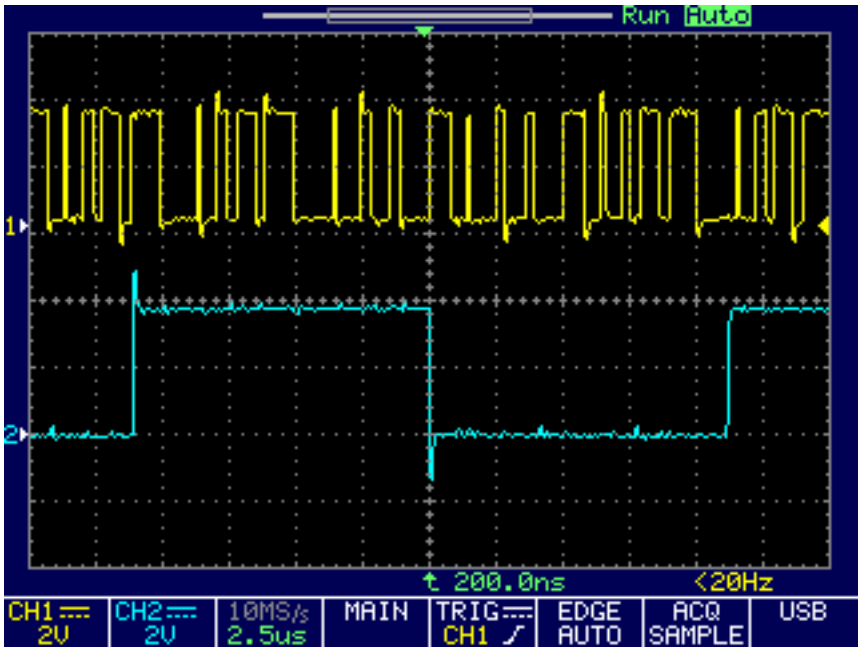


Figura 68. Señal de entrada (Onda cuadrada de 45 KHz - azul). Muestras por bit (amarilla)

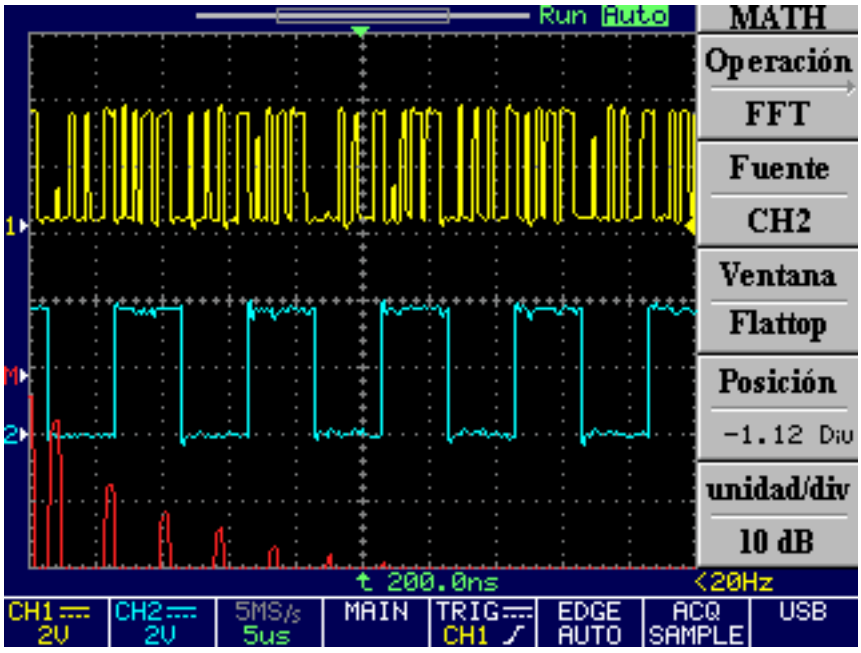


Figura 69. Señal de entrada (Onda cuadrada de 100 KHz - azul). Muestras por bit (amarilla). Espectro de la señal cuadrada (rojo).

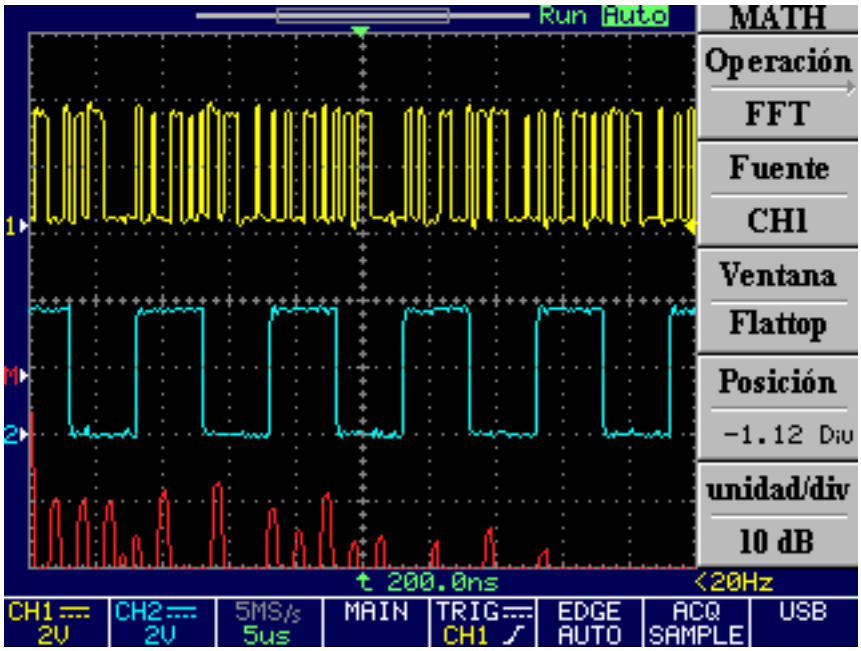


Figura 70. Señal de entrada (Onda cuadrada de 100 KHz - azul). Muestras por bit (amarilla). Espectro de la señal del canal (rojo).

Las figuras 41 a la 70, representan la etapa 1 (Recordando que la etapa 1 es el proceso de comunicación conectado con cable UTP categoría 3 de 20 centímetros), las siguientes figuras (Figura 71 - Figura 94) representan la etapa 2 (proceso de comunicación conectado con par de cobre de 3 metros).

Se observan resultados similares en ambas etapas, con la pequeña diferencia que en la señal de salida (señal azul para todas las imágenes) se nota una adición de ruido producido por el cambio de cable. Se nota nuevamente que a partir de 30 KHz (Figura 83 y Figura 84) en el espectro de la señal de salida (Figura 84, señal en color rojo) hay una adición de armónicos de baja frecuencia y amplitud, que a frecuencias menores no se presenta, pero que la duración de los periodos de ambas señales (entrada y salida) son muy parecidos y casi perfectamente reconstruidos.

A partir de 60 KHz (Figura 87 y Figura 88) se aprecian bits más largos que otros, y a 90 KHz (Figura 89 y Figura 90) se observan pérdidas de bits.

Frecuencias	Nivel de señal de entrada en dB	Nivel de señal de salida en dB	Diferencia dB
1Khz	27.12	25.62	1.5
5Khz	27.12	26.62	0.5
10Khz	27.12	25.62	1.5
15Khz	27.62	26.62	1
20Khz	28.12	26.62	1.5
25Khz	28.62	26.12	2.5
40Khz	27.12	25.12	2
45Khz	28.12	26.12	2
50Khz	28.12	26.62	1.5
55Khz	28.12	25.62	2.5
60Khz	28.12	21.12	7

Tabla 3. Relación en dB de las potencias de entrada y salida con cable UTP de 20cm.

La Tabla 3 se elaboró mediante la medición con el osciloscopio usando la función FFT que muestra los espectros de las señales de entrada y salida, los resultados son coherentes con las observaciones de las señales, notando que a partir de 25 KHz se empieza a presentar una atenuación de 2 dB o más, teniendo en cuenta que a los 3 dB se pierde el 50 % de la potencia de entrada.

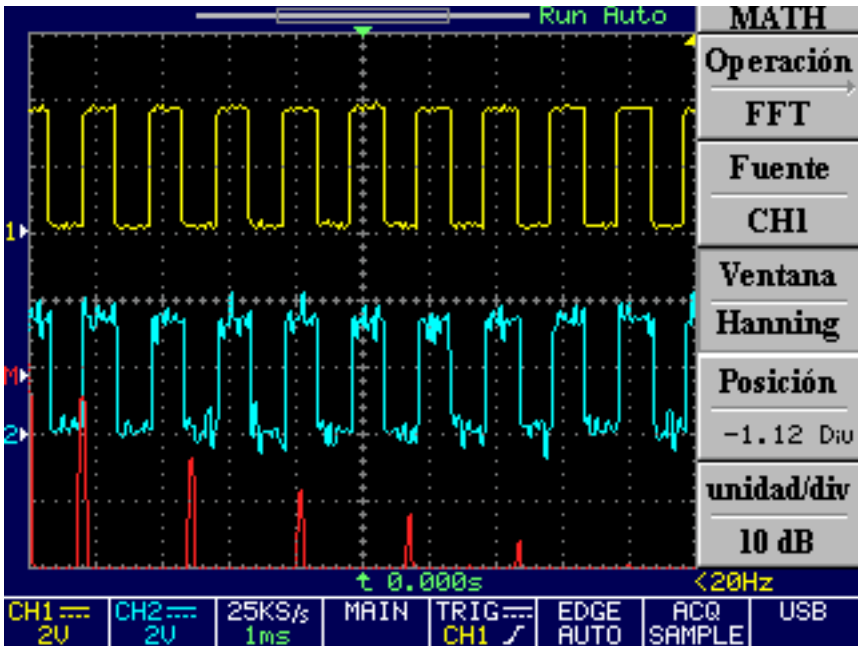


Figura 71. Señal de entrada (Onda cuadrada de 1 KHz – amarilla). Señal de salida (Onda cuadrada de 1 KHz – azul). Espectro de la señal de entrada (rojo).

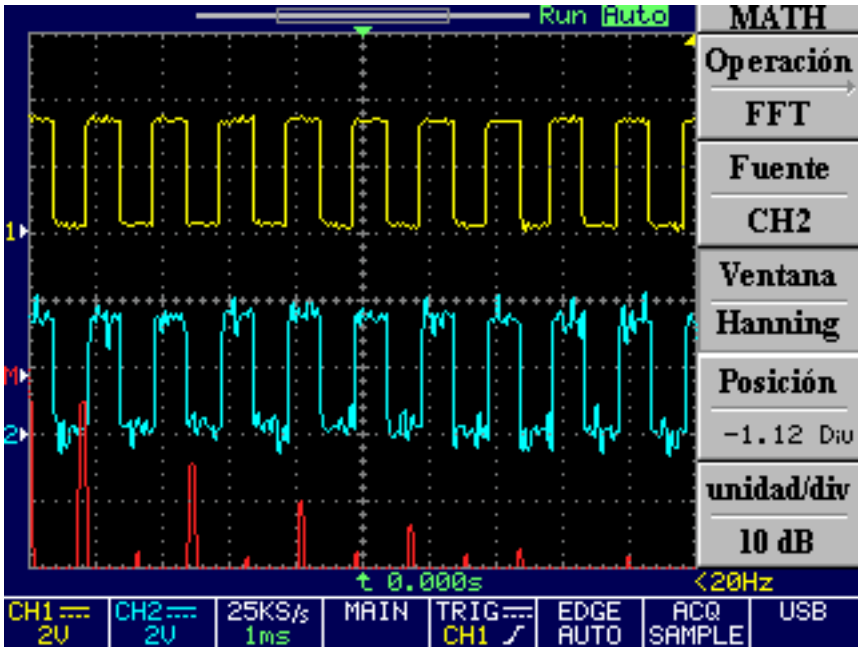


Figura 72. Señal de entrada (Onda cuadrada de 1 KHz – amarilla). Señal de salida (Onda cuadrada de 1 KHz – azul). Espectro de la señal de salida (rojo).

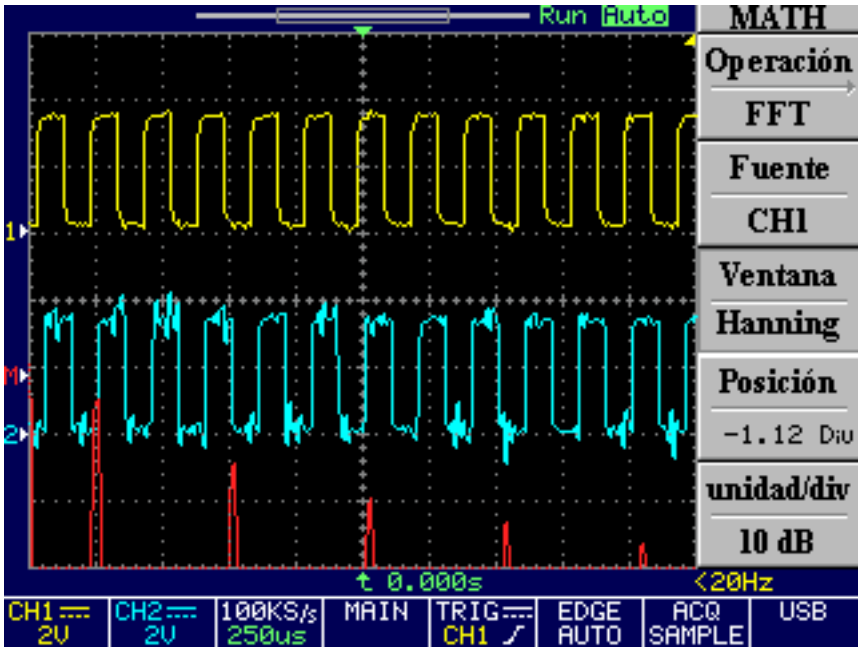


Figura 73. Señal de entrada (Onda cuadrada de 5 KHz – amarilla). Señal de salida (Onda cuadrada de 5 KHz – azul). Espectro de la señal de entrada (rojo).

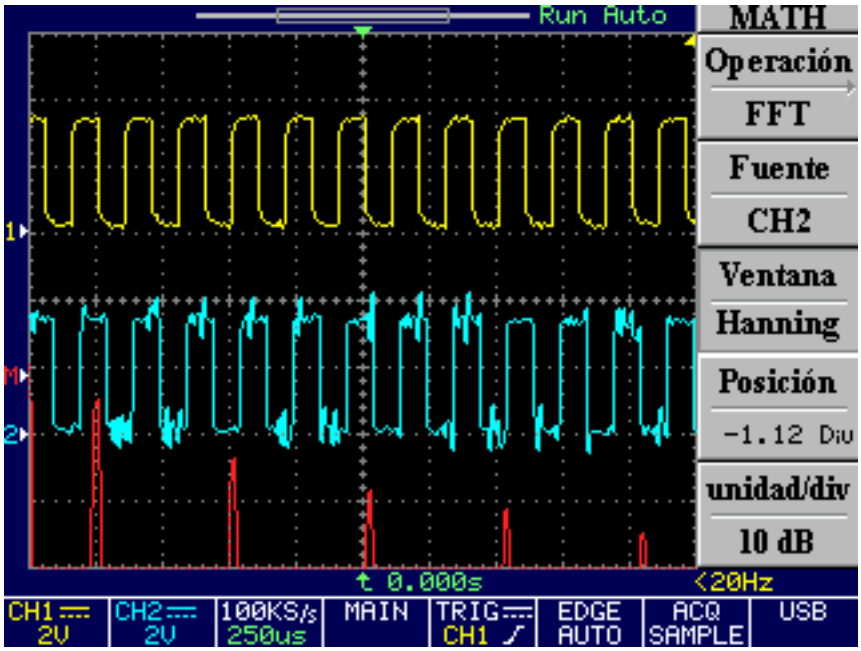


Figura 74. Señal de entrada (Onda cuadrada de 5 KHz – amarilla). Señal de salida (Onda cuadrada de 5 KHz – azul). Espectro de la señal de salida (rojo).

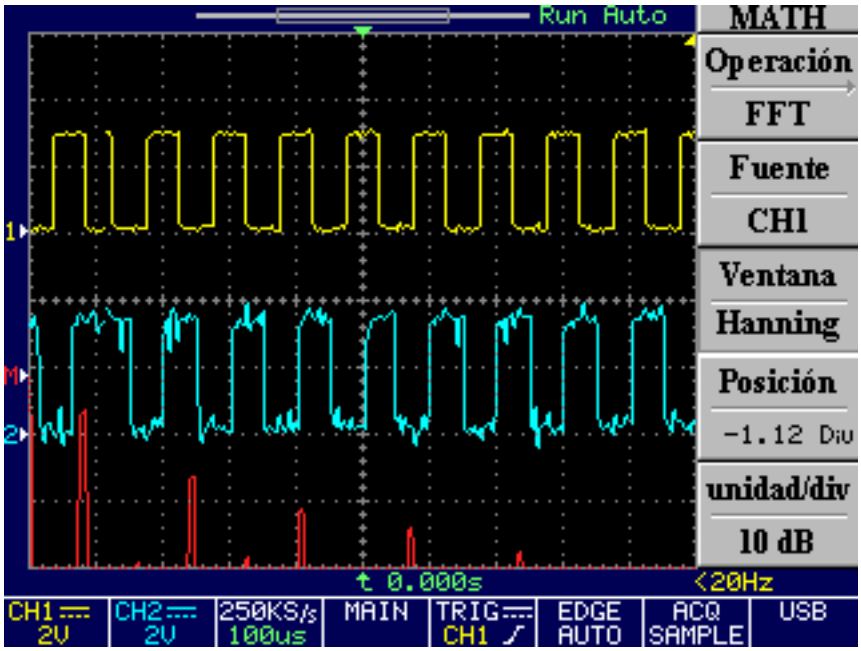


Figura 75. Señal de entrada (Onda cuadrada de 10 KHz – amarilla). Señal de salida (Onda cuadrada de 10 KHz – azul). Espectro de la señal de entrada (rojo).

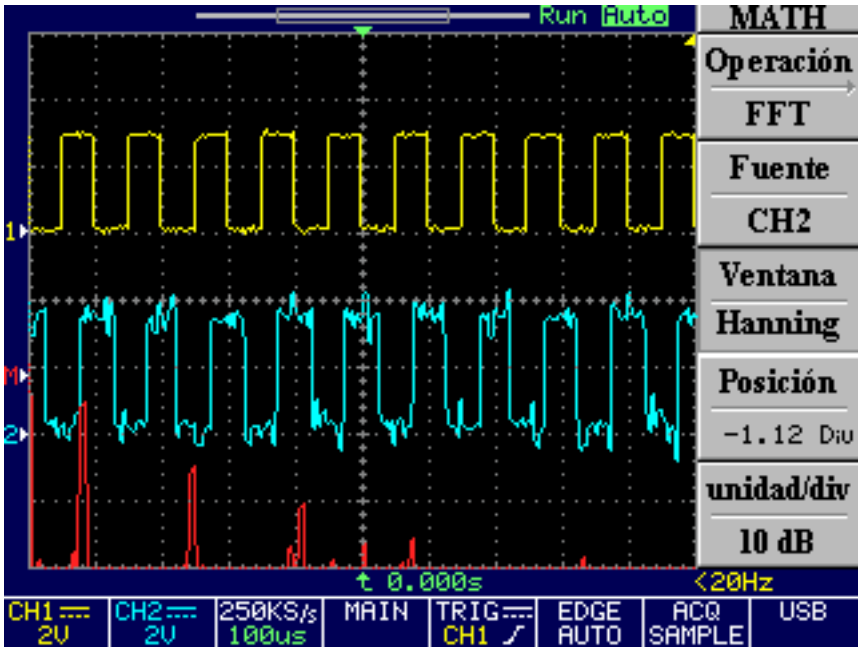


Figura 76. Señal de entrada (Onda cuadrada de 10 KHz – amarilla). Señal de salida (Onda cuadrada de 10 KHz – azul). Espectro de la señal de salida (rojo).

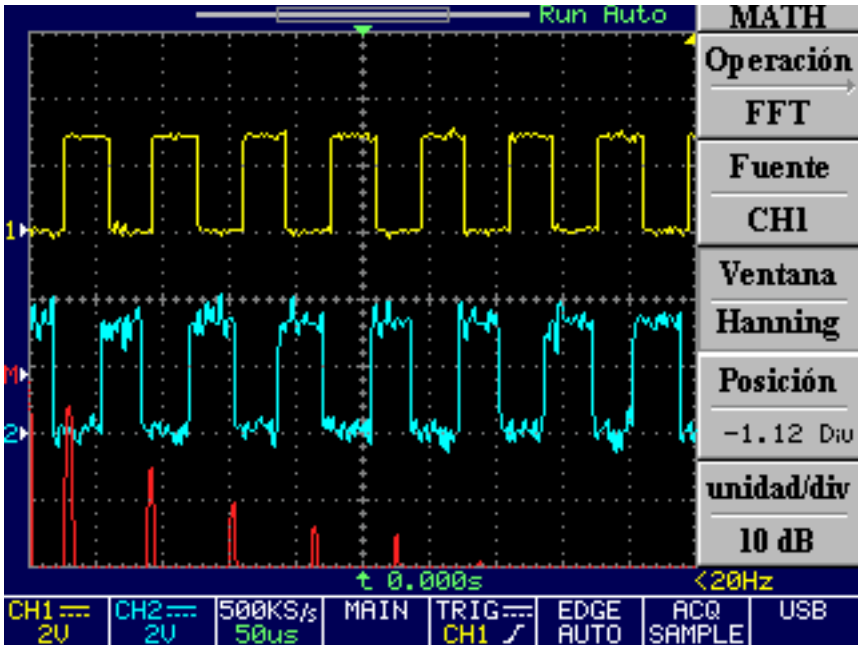


Figura 77. Señal de entrada (Onda cuadrada de 15 KHz – amarilla). Señal de salida (Onda cuadrada de 15 KHz – azul). Espectro de la señal de entrada (rojo).

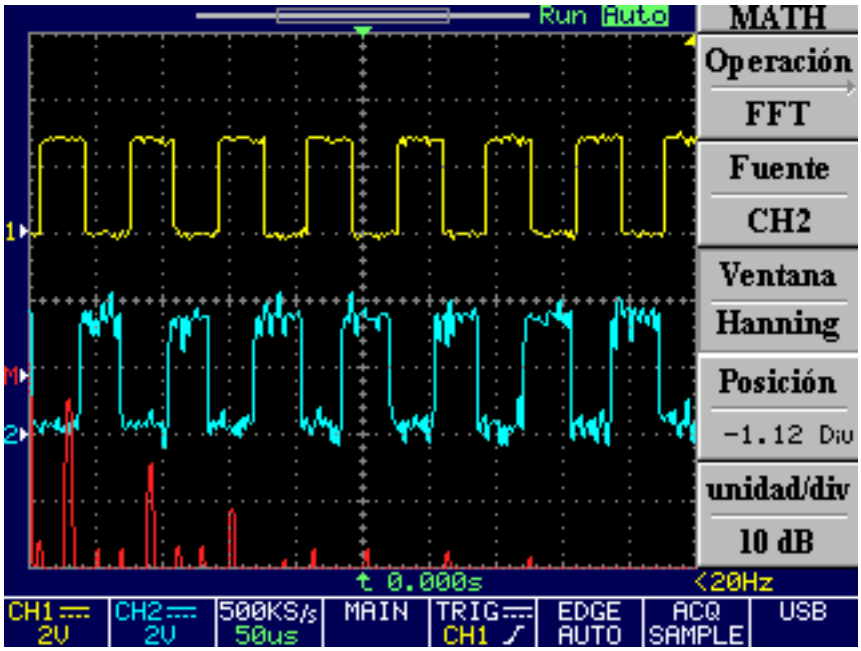


Figura 78. Señal de entrada (Onda cuadrada de 15 KHz – amarilla). Señal de salida (Onda cuadrada de 15 KHz – azul). Espectro de la señal de salida (rojo).

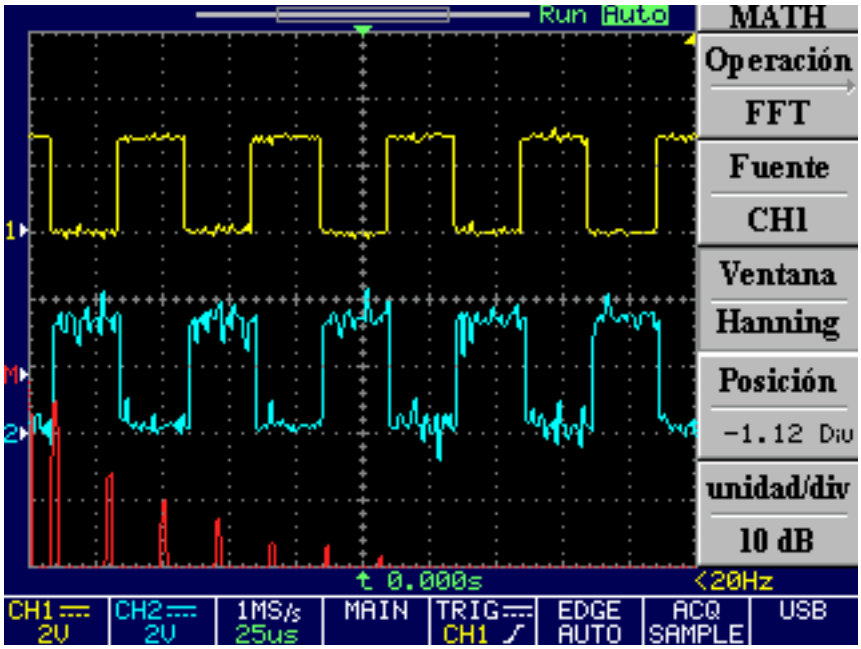


Figura 79. Señal de entrada (Onda cuadrada de 20 KHz – amarilla). Señal de salida (Onda cuadrada de 20 KHz – azul). Espectro de la señal de entrada (rojo).

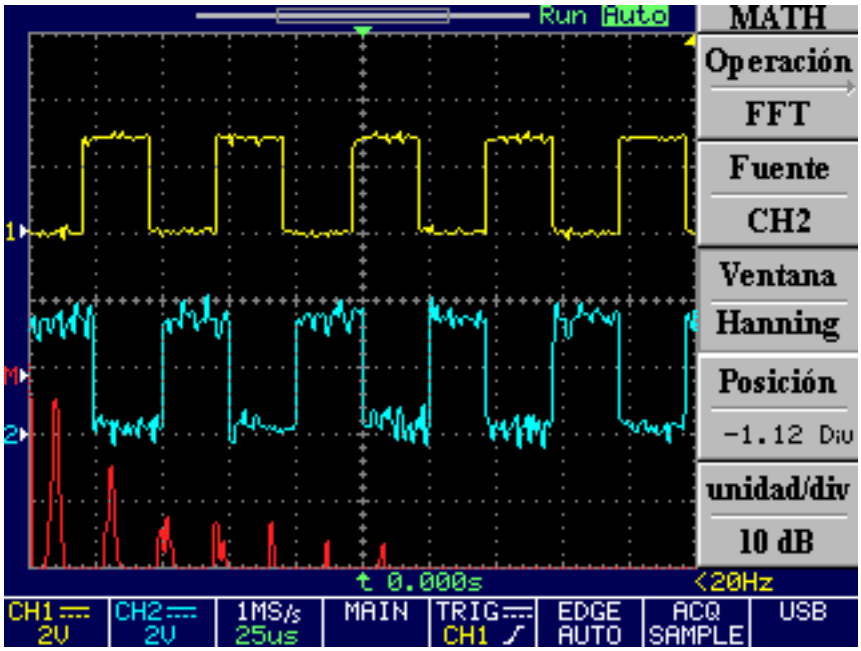


Figura 80. Señal de entrada (Onda cuadrada de 20 KHz – amarilla). Señal de salida (Onda cuadrada de 20 KHz – azul). Espectro de la señal de salida (rojo).

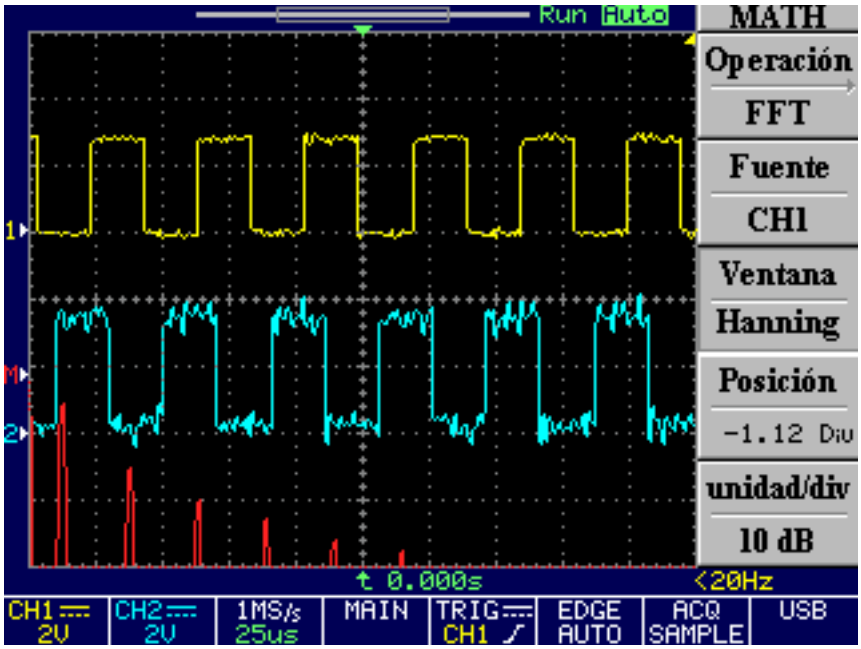


Figura 81. Señal de entrada (Onda cuadrada de 25 KHz – amarilla). Señal de salida (Onda cuadrada de 25 KHz – azul). Espectro de la señal de entrada (rojo).

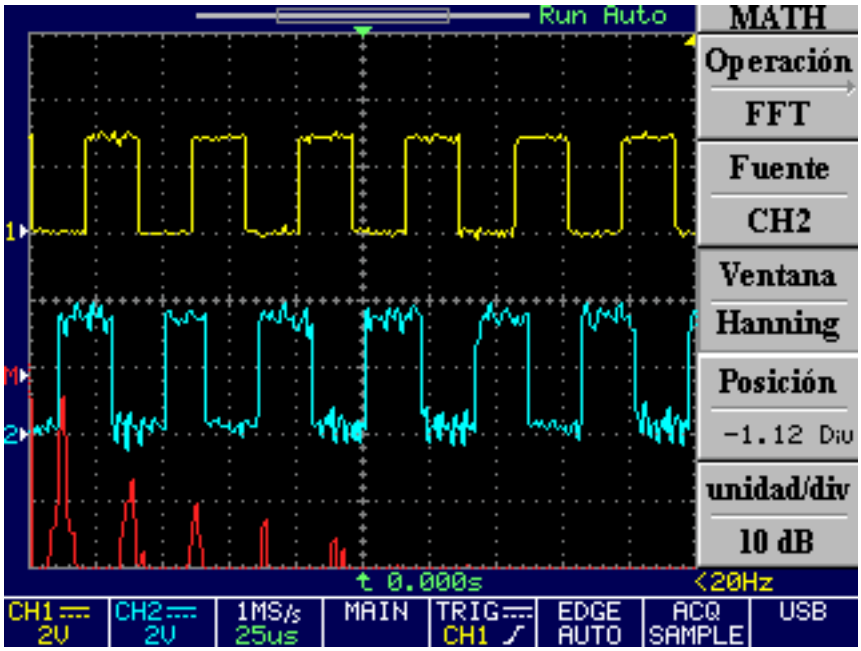


Figura 82. Señal de entrada (Onda cuadrada de 25 KHz – amarilla). Señal de salida (Onda cuadrada de 25 KHz – azul). Espectro de la señal de salida (rojo).

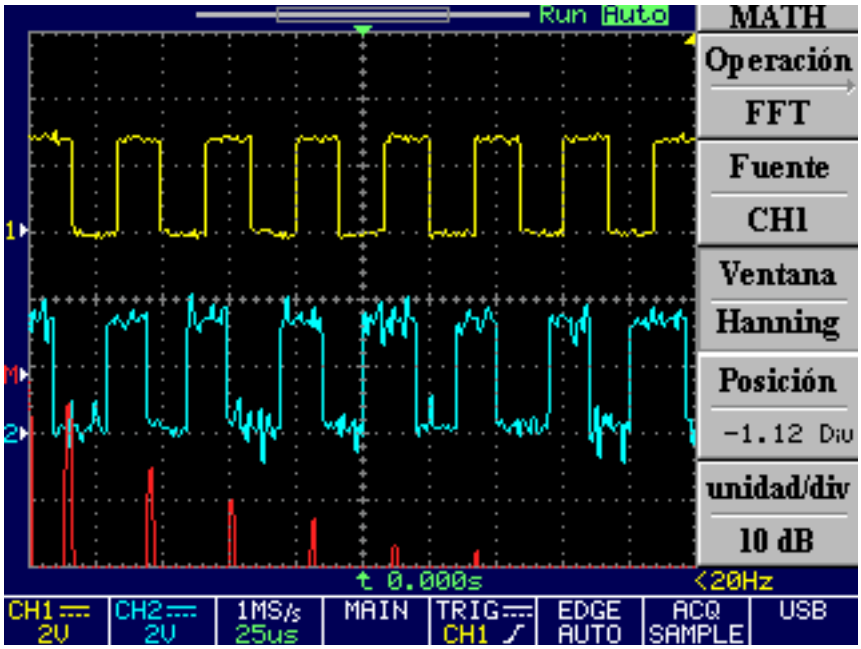


Figura 83. Señal de entrada (Onda cuadrada de 30 KHz – amarilla). Señal de salida (Onda cuadrada de 30 KHz – azul). Espectro de la señal de entrada (rojo).

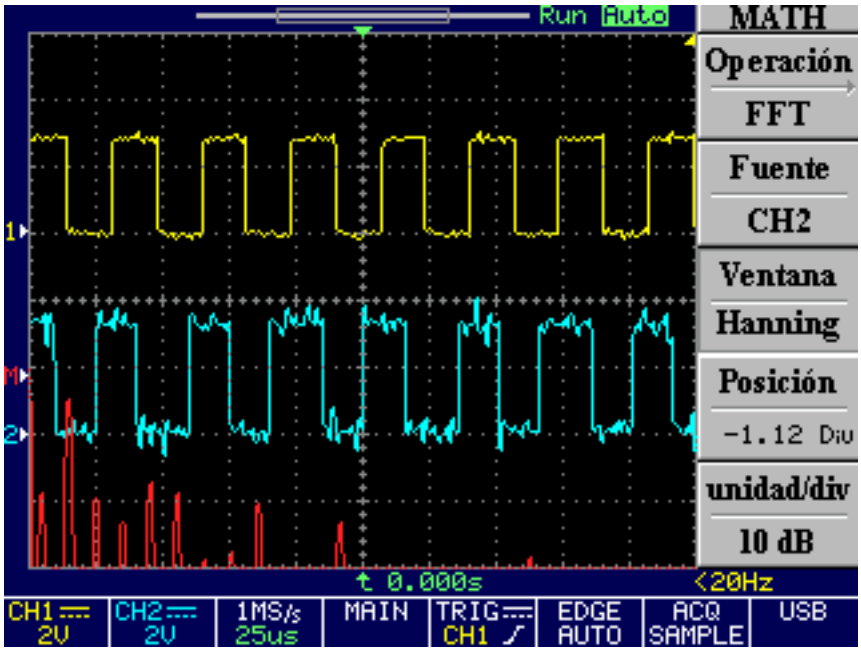


Figura 84. Señal de entrada (Onda cuadrada de 30 KHz – amarilla). Señal de salida (Onda cuadrada de 30 KHz – azul). Espectro de la señal de salida (rojo).

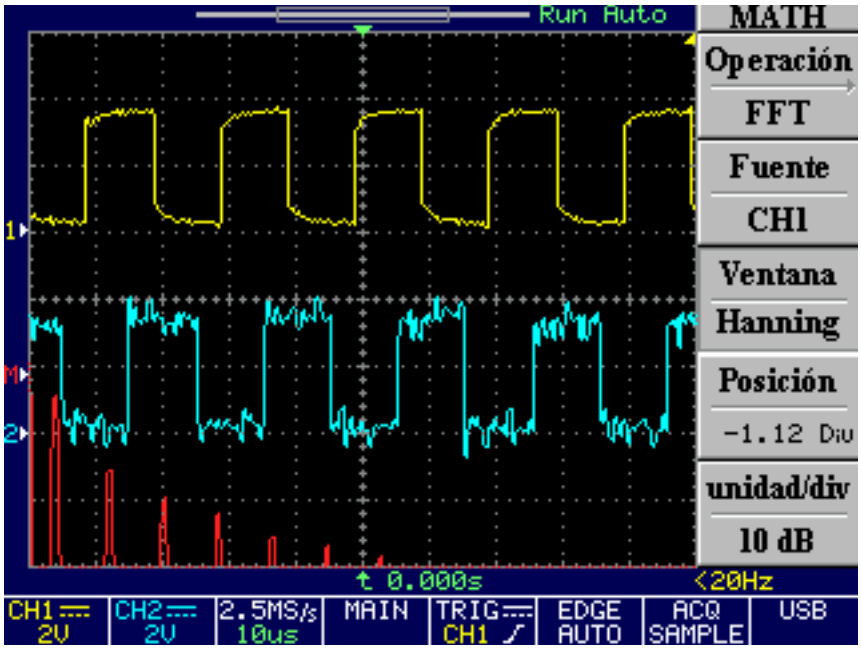


Figura 85. Señal de entrada (Onda cuadrada de 50 KHz – amarilla). Señal de salida (Onda cuadrada de 50 KHz – azul). Espectro de la señal de entrada (rojo).

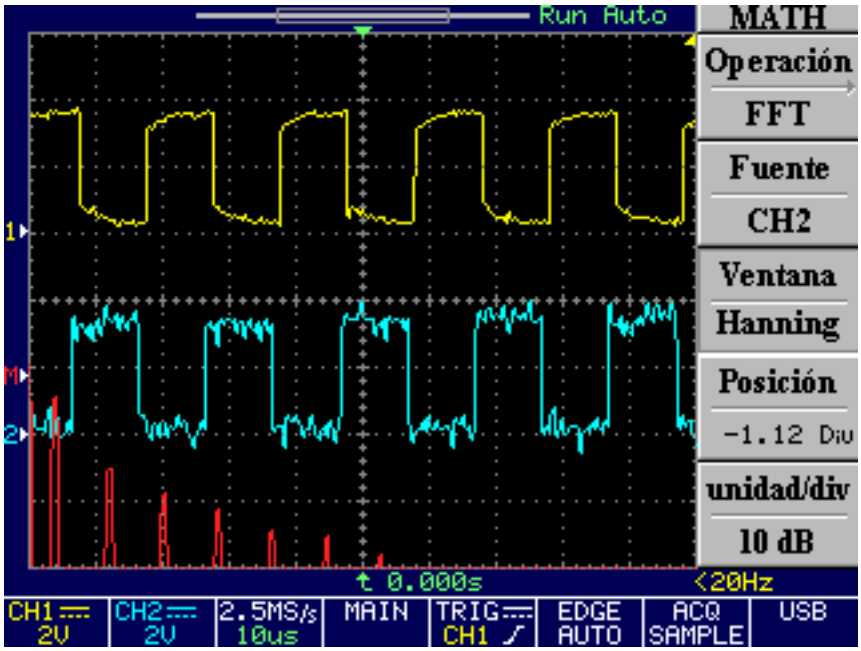


Figura 86. Señal de entrada (Onda cuadrada de 50 KHz – amarilla). Señal de salida (Onda cuadrada de 50 KHz – azul). Espectro de la señal de salida (rojo).

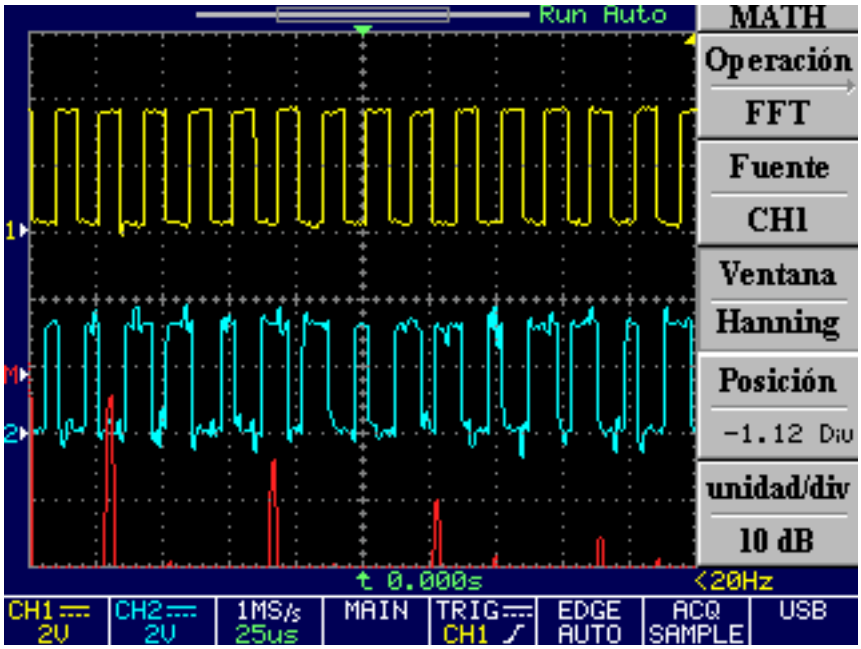


Figura 87. Señal de entrada (Onda cuadrada de 60 KHz – amarilla). Señal de salida (Onda cuadrada de 60 KHz – azul). Espectro de la señal de entrada (rojo).

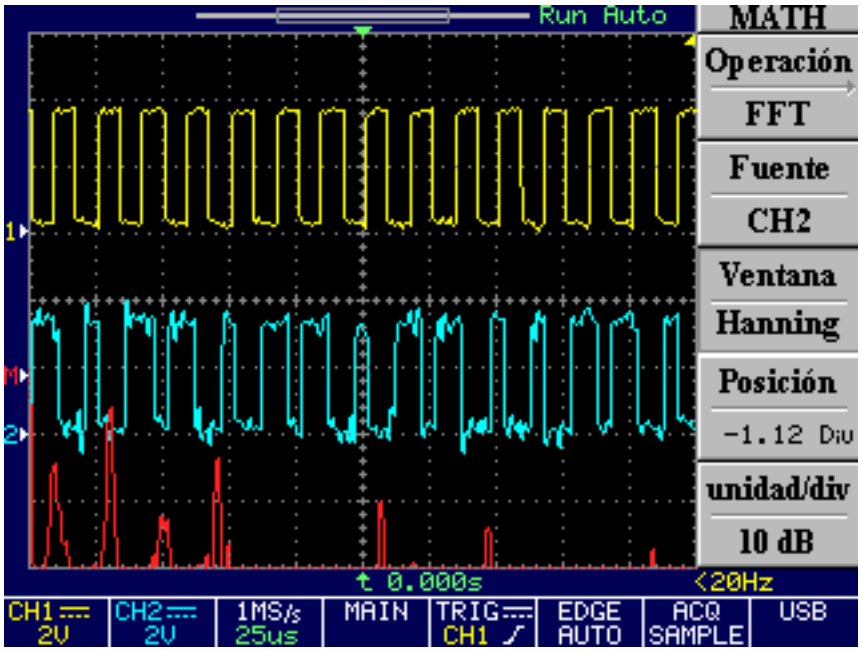


Figura 88. Señal de entrada (Onda cuadrada de 60 KHz – amarilla). Señal de salida (Onda cuadrada de 60 KHz – azul). Espectro de la señal de salida (rojo).

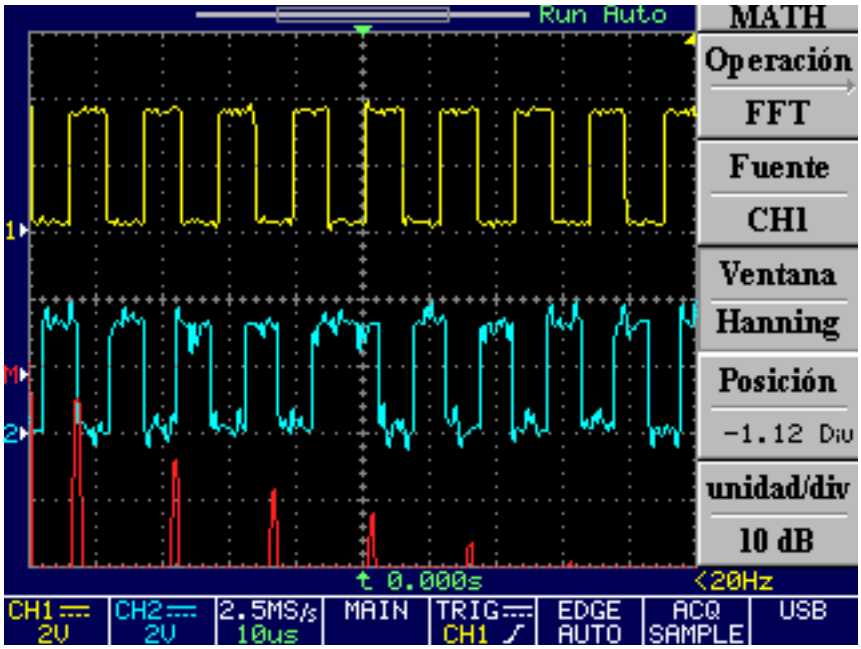


Figura 89. Señal de entrada (Onda cuadrada de 90 KHz – amarilla). Señal de salida (Onda cuadrada de 90 KHz – azul). Espectro de la señal de entrada (rojo).

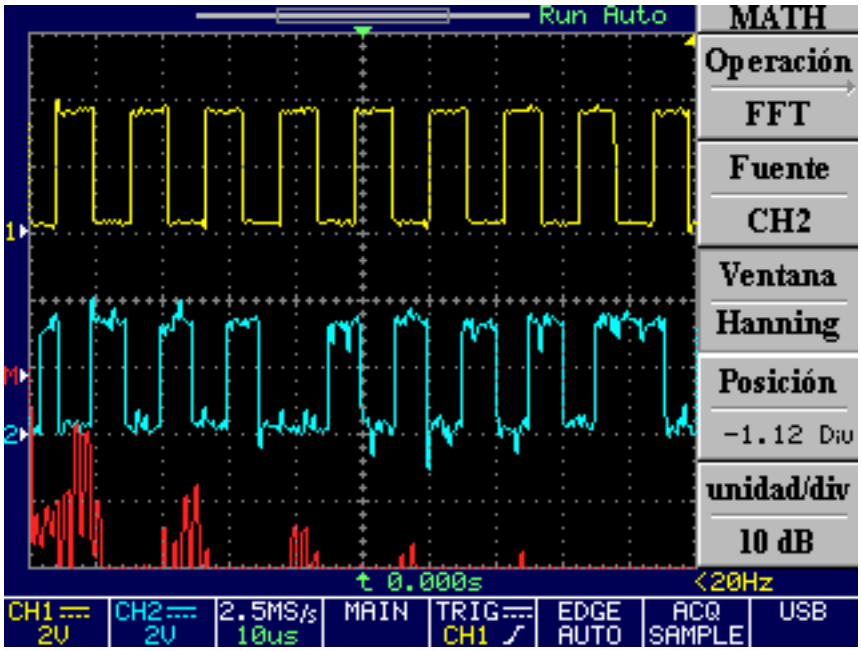


Figura 90. Señal de entrada (Onda cuadrada de 90 KHz – amarilla). Señal de salida (Onda cuadrada de 90 KHz – azul). Espectro de la señal de salida (rojo).

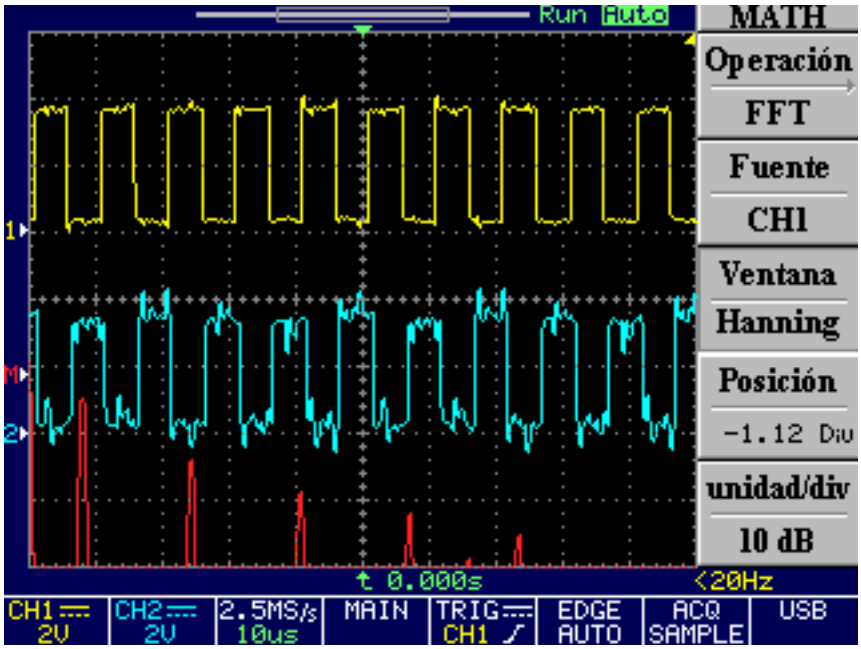


Figura 91. Señal de entrada (Onda cuadrada de 100 KHz – amarilla). Señal de salida (Onda cuadrada de 100 KHz – azul). Espectro de la señal de entrada (rojo).

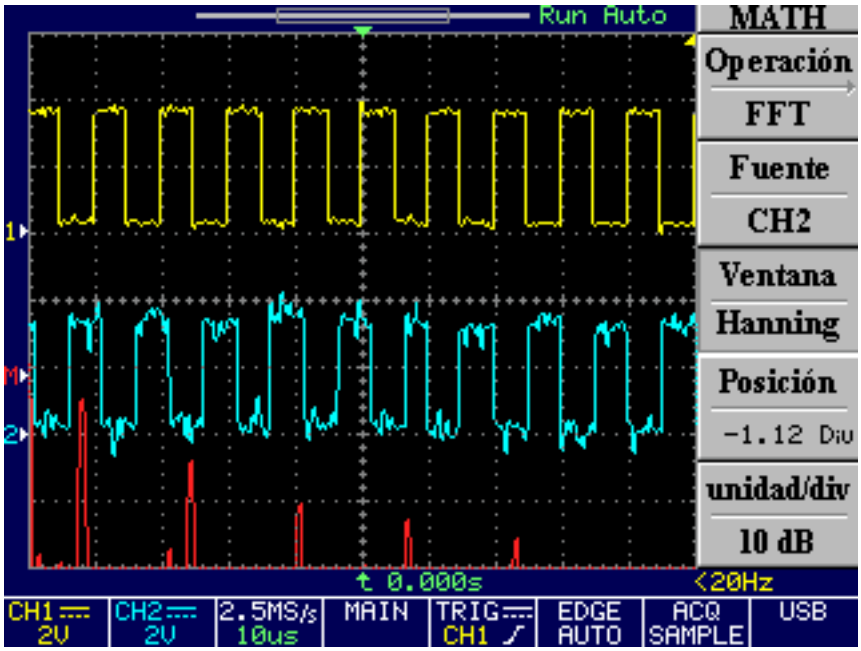


Figura 92. Señal de entrada (Onda cuadrada de 100 KHz – amarilla). Señal de salida (Onda cuadrada de 100 KHz – azul). Espectro de la señal de salida (rojo).

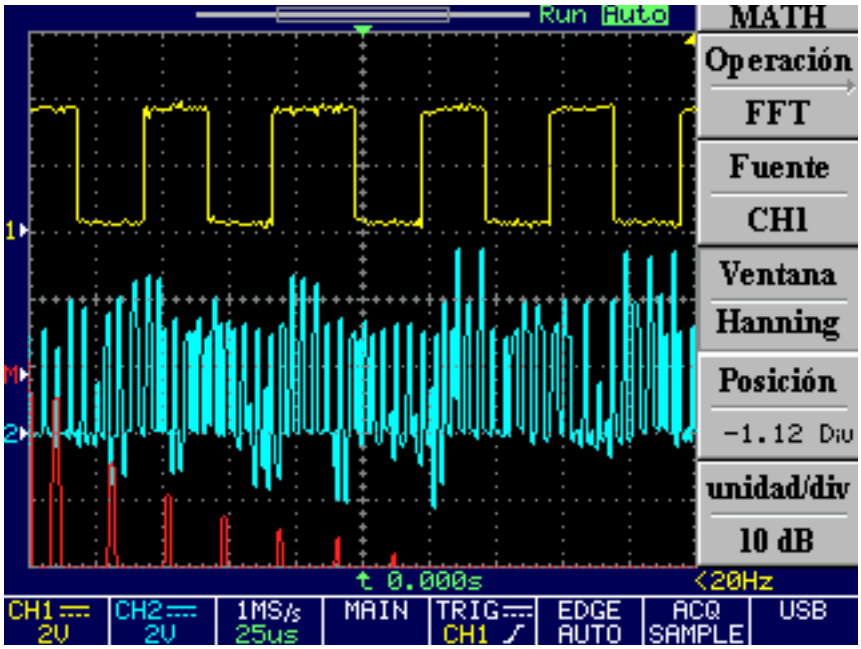


Figura 93. Señal de entrada (Onda cuadrada de 20 KHz – amarilla). Señal presente en el canal de comunicación (Azul). Espectro de la señal de entrada (rojo).

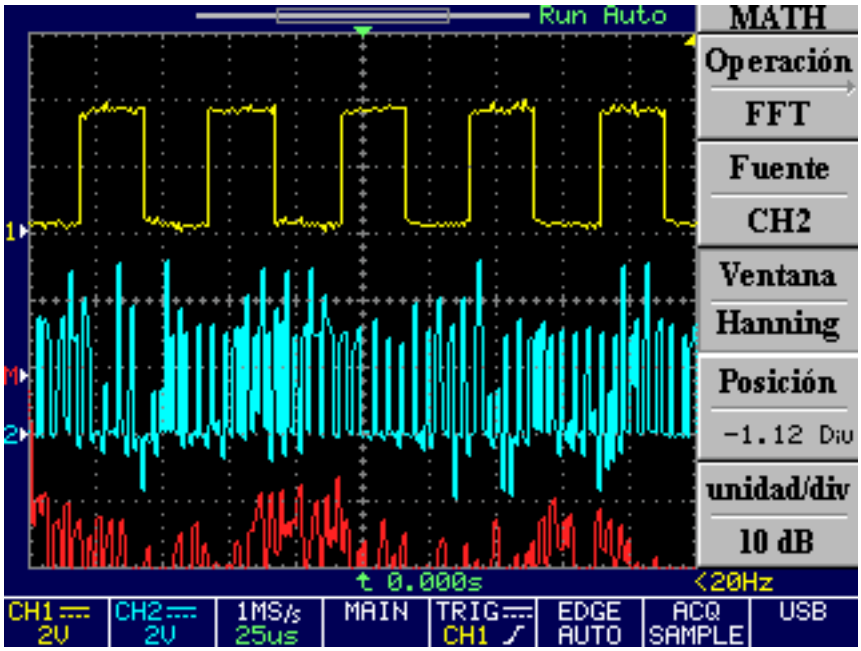


Figura 94. Señal de entrada (Onda cuadrada de 20 KHz – amarilla). Señal presente en el canal de comunicación (Azul). Espectro del canal (rojo).

En las figuras 93 y 94 se hace una comparación de los espectros de la señal de entrada (figura 93 en color rojo) y espectro del canal de comunicación (figura 94 en color rojo), apreciándose como el espectro es ensanchado en frecuencia y reducido en amplitud a razón de 11 dB, siendo semejante al espectro de una señal de ruido.

En la Tabla 4 se puede apreciar cómo el ruido generado por el cambio del cable del canal de comunicaciones afecta la señal de salida aumentando la amplitud de las frecuencias fundamentales, efecto observado en las mediciones de las frecuencias desde los 10KHz hasta los 30KHz donde la atenuación es negativa significando esto una amplificación de la potencia en la señal de salida. En los 90KHz como se había mencionado antes se toma una muestra por bit generando una pérdida de datos, lo cual se puede ver en la Tabla 4, en donde la atenuación a esta frecuencia es de 3.5 dB dando como pérdida más del 50% de la potencia de entrada.

Frecuencias	Nivel de señal de entrada en dB	Nivel de señal de salida en dB	Atenuación dB
1KHz	26.62	26.12	0.5
5KHz	26.12	26.12	0
10KHz	25.12	26.12	- 1
15KHz	25.12	26.12	- 1
20KHz	26.12	26.12	0
25KHz	25.62	27.12	- 1.5
30KHz	25.62	26.62	- 1
50KHz	27.12	26.62	0.5
60KHz	26.62	25.12	1.5
90KHz	26.62	22.12	3.5

Tabla 4. Relación en dB de las potencias de entrada y salida con par de cobre de 3 metros.

5.1 CONCLUSIONES

- ✓ Uno de los efectos secundarios de la técnica de Spread Spectrum en Secuencia Directa, es el incremento en el ancho de banda en el proceso de transmisión, dado que al realizar la modulación spread spectrum en secuencia directa se multiplica cada bit de datos por una secuencia pseudo-aleatoria, incrementando el tamaño de cada bit de datos. Este efecto se observa en la figura 69, por cada bit de datos se notan: un bit de sincronismo mas 11 bits de la pseudo-aleatoria, esto causa que el espectro sea ensanchado en frecuencia y disminuido en amplitud como se observa en la figura 94 en color rojo.

- ✓ Al interceptar una señal modulada con Spread Spectrum sin conocer la secuencia Pseudo-aleatoria (PN) con la que fue modulada se obtiene una señal de ruido Pseudo-aleatorio irreconocible, por lo cual es importante que el receptor esté debidamente sincronizado y conozca perfectamente la secuencia PN para que la comunicación tenga éxito; se concluye entonces que sin importar la modalidad del Spread Spectrum el directamente responsable del funcionamiento de un sistema de comunicaciones con esta técnica es la secuencia PN. Por lo anteriormente mencionado se puede garantizar la seguridad en los datos cifrados de esta manera, ya que es inmune a interferencias intencionadas y que la correcta recepción de los datos solo puede traducirse por un receptor autorizado.

- ✓ Al hacer una comparación de los resultados cuando se cambia el tipo de cable con que se conectan emisor y receptor, se hace notorio un efecto producido por este cambio y es que con el par de cobre se añade ruido a la salida, efecto perfectamente observable al comparar las figuras 45 y 72 las cuales representan la señal de entrada, salida y espectro de salida a una frecuencia de 1KHz, apreciando en la señal de salida (señal en color azul de la figura 72) un ruido añadido. Aún así con la adición de ruido, se sigue manteniendo la coherencia en la comunicación.

5.2 RECOMENDACIONES

- ✓ Si se desea diseñar un sistema de comunicación con la técnica de espectro ensanchado por secuencia directa, se hace necesario el uso de dispositivos de hardware de alta velocidad y de gran ancho de banda. También cabe mencionar que los sistemas de comunicación son muy susceptibles al ruido y las interferencias externas, como es el caso de las FPGA's las cuales son propensas a sufrir daños y mal funcionamiento por la estática. Por tales razones es recomendable usar los dispositivos adecuados con las características requeridas por esta técnica, además de usar las protecciones necesarias para manipular dichos elementos.

- ✓ A partir del desarrollo de este trabajo, se dejan varios temas para posteriores investigaciones y desarrollos, por ejemplo: la generación automática y aleatoria de tipos de secuencias Pseudo-aleatorias (Barker, Gold, Máxima longitud) esto con el fin de hacer la comunicación más segura. También se podría realizar la comunicación mediante esta técnica, pero en lugar de usar señales cuadradas (digitales), hacer uso de señales análogas con el fin de transmitir voz y datos.

6. GLOSARIO

- ✓ **Ancho de banda:** es la cantidad de información o de datos que se puede enviar a través de una conexión de red en un período de tiempo dado. El ancho de banda se indica generalmente en bites por segundo (BPS), kilobits por segundo (Kbps), o megabits por segundo (Mbps).
- ✓ **Codificar:** Es el proceso por el cual la información de una fuente es convertida en símbolos para ser comunicada.
- ✓ **Canal de comunicación:** es el medio de transmisión por el que viajan las señales portadoras de la información que pretenden intercambiar emisor y receptor. Es frecuente referenciarlo también como canal de datos.
- ✓ **Correlación:** es una medida sobre el grado de relación entre dos variables, sin importar cuál es la causa y cuál es el efecto. La dependencia de la que se habla en este sentido es la dependencia entre la varianza de las variables.
- ✓ **Decodificar:** Aplicar las reglas adecuadas a un mensaje que ha sido emitido en un sistema de signos determinado para entenderlo.
- ✓ **Densidad espectral:** la Densidad Espectral (Spectral Density) de una señal es una función matemática que nos informa de cómo está distribuida la potencia o la energía (según el caso) de dicha señal sobre las distintas frecuencias de las que está formada, es decir, su espectro.
- ✓ **Espectro:** en términos generales, el espectro es toda la gama de radiaciones electromagnéticas, que va desde los rayos gamma a las ondas radio.
- ✓ **Ganancia:** relación de transferencia entre la salida y la entrada de un sistema.
- ✓ **Oscilación:** variación, perturbación o fluctuación en el tiempo de un medio o sistema. Si el fenómeno se repite, se habla de oscilación periódica.
- ✓ **Radiocomunicación:** es un sistema de telecomunicación que se realiza a través de ondas de radio u ondas hertzianas, y que a su vez está caracterizado por el movimiento de los campos eléctricos y campos magnéticos. La comunicación vía radio se realiza a través del espectro radioeléctrico cuyas propiedades son diversas a lo largo de su gama así cómo baja frecuencia, media frecuencia, alta frecuencia, muy alta frecuencia, ultra alta frecuencia, etc. En cada una de ellas, el comportamiento de las ondas es diferente.
- ✓ **Red LAN:** Una **red de área local**, **red local** o **LAN** (del inglés *local area network*) es la interconexión de varias computadoras y periféricos. Su extensión está limitada físicamente a un edificio o a un entorno de 200 metros, o con repetidores podría llegar a la distancia de un campo de 1 kilómetro.
- ✓ **Relación señal a ruido:** se define como el margen que hay entre la potencia de la señal que se transmite y la potencia del ruido que la corrompe. Este margen es medido en decibelios.
- ✓ **Secuencia pseudoaleatoria:** cualquier grupo de secuencias binarias que presentan propiedades aleatorias parecidas a las del ruido. Las secuencias de pseudorruido se distinguen de las secuencias aleatorias de verdad en que muestran una periodicidad.
- ✓ **Sincronizar:** es hacer que coincidan en el tiempo dos o más fenómenos. En otras palabras, sincronizar se refiere a que dos o más elementos, eventos u

operaciones sean programadas para que ocurran en un momento predefinido de tiempo o lugar.

- ✓ **Sistema de comunicación:** conjunto de dispositivos interconectados que realizan acciones las cuales permiten que las personas puedan comunicarse o conectarse entre sí.

7. REFERENCIAS BIBLIOGRAFICAS

[1] IMPLEMENTACIÓN DE UNA RED. Referencias técnicas [en linea]<http://www.pdfdownload.org/pdf2html/view_online.php?url=http%3A%2F%2Fitzamna.bnct.ipn.mx%3A8080%2Fdspace%2Fbitstream%2F123456789%2F2963%2F1%2FIMPLEMENTACIONDEUNARED.pdf>[citado el 4 de Abril de 2010]

[2] QUINTERO, EDWIN. OROZCO, HOOVER. GALLEGO, HUGO. Generación de Spread Spectrum usando microcontrolador. Scientia et Technica Año XV, No 41, Mayo de 2009. Universidad Tecnológica de Pereira. ISSN 0122-1701

[3] HAYKIN, Simon. Communication Systems. 4th Edition. John Wiley & sons, Inc. 2001. ISBN 0-471-17869-1

[4] DATA COMMUNICATIONS. The IEEE 802.11b Standard <http://www.pulsewan.com/data101/802_11_b_basics.htm> [Citado el 20 de Mayo de 2010]

[5] HERNANZ, DANIEL. Estudio de la capa física del 802.1. 8 de Abril de 2002

[6] Documento Informativo, Capitulo 1[en línea] <http://bibdigital.epn.edu.ec/bitstream/15000/83/1/CD-0055.pdf> [Citado el 18 de Enero de 2011]

[7] SANTIAGO T. PÉREZ, JESÚS B. ALONSO, CARLOS M. TRAVIESO, MIGUEL A. FERRER, JOSÉ F. CRUZ Implementation of a Fast Frequency Hopping Spread Spectrum modulator with System Generator on a FPGA. Universidad de las palmas, S/N, 35017, Las Palmas de Gran Canaria.

[8] DIGITAL MODULATION TECHNIQUES. Frequency Hopping Spread Spectrum <http://digitalmodulation.net/fhss2.html> [citado el 5 de Mayo de 2010]

[9] RAJAGOPAL, HARISH. NAWANI, VARUN. Design of Optimized Engine for Direct Sequence Spread Spectrum Transceiver. Project Report Año 2004

[10] Practica de laboratorio [en línea] <http://neutron.ing.ucv.ve/comunicaciones/Asignaturas/TxDatos/SpreadSpectrum.pdf> [Citado el 18 de Enero de 2011]

[11] ALÍ AGIB, BASHAR. GARCÍA SIMÓN, AMADO IGNACIO. CABARROUY FERNÁNDEZ FONTECHA, SERGIO LÁZARO. Influencia del balance de los códigos de GOLD y MLS en la calidad de la recepción en sistemas DS CDMA. Volumen 10, Número 34, junio 2005. pp 103-107. Universidad Ciencia y Tecnología.

[12] MATHWORKS. Documentation, Communications Blockset [en línea] <http://www.mathworks.com/access/helpdesk/help/toolbox/commblks/ref/barkercodegenerator.html> [Citado el 7 de Junio de 2010]

[13] NATIONAL INSTRUMENTS. Introduction to FPGA Technology [en línea] <http://zone.ni.com/devzone/cda/tut/p/id/6984> [Citado el 15 de Junio de 2010]

[14] UNIVERSIDAD FRANCISCO DE PAULA SANTANDER. Microelectrónica <www.ufps.edu.co/materias/uelectro/htdocs/pdf/fpga.pdf> [Citado el 15 de Julio de 2010]

[15] FREIRE RUBIO, MIGUEL ANGEL. Introducción al lenguaje VHDL, Universidad Politécnica de Madrid.

8. ANEXOS

8.1 ANEXO 1: DATASHEET OPA860

Wide Bandwidth OPERATIONAL TRANSCONDUCTANCE AMPLIFIER (OTA) and BUFFER

OTA FEATURES

- Wide Bandwidth (80MHz, Open-Loop, G = +5)
- High Slew Rate (900V/µs)
- High Transconductance (95mA/V)
- External I_Q-Control

BUFFER FEATURES

- Closed-Loop Buffer
- Wide Bandwidth (1600MHz, V_O = $1V_{PP}$)
- High Slew Rate (4000V/µs)
- 60mA Output Current

OPA860 FEATURES

- Low Quiescent Current (11.2mA)
- Versatile Circuit Function

APPLICATIONS

- Baseline Restore Circuits
- Video/Broadcast Equipment
- Communications Equipment
- High-Speed Data Acquisition
- Wideband LED Driver
- AGC-Multiplier
- ns-Pulse Integrator
- Control Loop Amplifier
- OPA660 Upgrade

DESCRIPTION

The OPA860 is a versatile monolithic component designed for wide-bandwidth systems, including high performance video, RF and IF circuitry. It includes a wideband, bipolar operational transconductance amplifier (OTA), and voltage buffer amplifier.

The OTA or voltage-controlled current source can be viewed as an *ideal transistor*. Like a transistor, it has three terminals—a high impedance input (base), a low-impedance input/output (emitter), and the current output (collector). The OTA, however, is self-biased and bipolar. The output collector current is zero for a zero base-emitter voltage. AC inputs centered about zero produce an output current, which is bipolar and centered about zero. The transconductance of the OTA can be adjusted with an external resistor, allowing bandwidth, quiescent current, and gain trade-offs to be optimized.

Also included in the OPA860 is an uncommited closed-loop, unity-gain buffer. This provides 1600MHz bandwidth and 4000V/µs slew rate.

Used as a basic building block, the OPA860 simplifies the design of AGC amplifiers, LED driver circuits for fiber optic transmission, integrators for fast pulses, fast control loop amplifiers and control amplifiers for capacitive sensors and active filters. The OPA860 is available in an SO-8 surface-mount package.

This integrated circuit can be damaged by ESD. Texas Instruments recommends that all integrated circuits be handled with appropriate precautions. Failure to observe proper handling and installation procedures can cause damage.

ESD damage can range from subtle performance degradation to complete device failure. Precision integrated circuits may be more susceptible to damage because very small parametric changes could cause the device not to meet its published specifications.

ORDERING INFORMATION(1)

PRODUCT	PACKAGE	PACKAGE DESIGNATOR	SPECIFIED TEMPERATURE RANGE	PACKAGE MARKING	ORDERING NUMBER	TRANSPORT MEDIA, QUANTITY
OPA860	SO-8	D	–45°C to +85°C	OPA860	OPA860ID	Rails, 75
					OPA860IDR	Tape and Reel, 2500

(1) For the most current package and ordering information see the Package Option Addendum at the end of this document, or see the TI web site at www.ti.com.

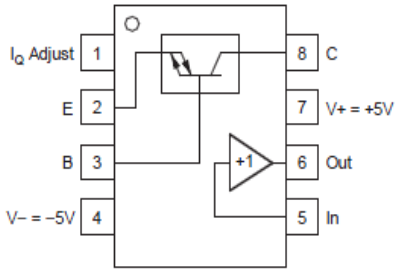
ABSOLUTE MAXIMUM RATINGS(1)

Power Supply	$\pm 6.5 V_{DC}$
Internal Power Dissipation	See Thermal Information
Differential Input Voltage	±1.2V
Input Common-Mode Voltage Range	$\pm V_S$
Storage Temperature Range: D	–65°C to +125°C
Lead Temperature (soldering, 10s)	+300°C
Junction Temperature (T_J)	+150°C
ESD Rating:	
Human Body Model (HBM)(2)	1500V
Charge Device Model (CDM)	1000V

(1) Stresses above these ratings may cause permanent damage. Exposure to absolute maximum conditions for extended periods may degrade device reliability. These are stress Ratings only, and functional operations of the device at these and any other conditions beyond those specified is not supported.
(2) Pin 2 > 500V HBM.

PIN CONFIGURATION

Top View SO



ELECTRICAL CHARACTERISTICS: V_S = ±5V

R_L = 500Ω and R_{ADJ} = 250Ω, unless otherwise noted.

PARAMETER	CONDITIONS	OPA860ID				UNITS	MIN/ MAX	TEST LEVEL(1)
		TYP	MIN/MAX OVER TEMPERATURE					
		+25°C	+25°C(2)	0°C to 70°C(3)	–40°C to +85°C(3)			
Closed Loop OTA + BUFFER (see Figure 53)								
AC PERFORMANCE	G = +2, See Figure 53							
Bandwidth	V_O = 200mV$_{PP}$	470	380	375	370	MHz	min	B
	V_O = 1V$_{PP}$	470				MHz	typ	C
	V_O = 5V$_{PP}$	350				MHz	typ	C
Bandwidth for 0.1dB Gain Flatness	V_O = 200mV$_{PP}$	42				MHz	typ	C
Slew Rate	V_O = 5V Step	3500	3000	2800	2700	V/µs	typ	C
Rise Time and Fall Time	V_O = 1V Step	0.7				ns	typ	C
Harmonic Distortion	G = +2, V_O = 2V$_{PP}$, 5MHz							
2nd-Harmonic	R_L = 100Ω	–54				dBc	typ	C
	R_L = 500Ω	–77				dBc	typ	C
3rd-Harmonic	R_L = 100Ω	–66				dBc	typ	C
	R_L = 500Ω	–79				dBc	typ	C
OTA - Open-Loop (see Figure 48)								
AC PERFORMANCE								
Bandwidth	G = +5, V_O = 200mV$_{PP}$, R_L = 500Ω	80	77	75	74	MHz	min	B
	G = +5, V_O = 1V$_{PP}$	80				MHz	typ	C
	G = +5, V_O = 5V$_{PP}$	80				MHz	typ	C
Slew Rate	G = +5, V_O = 5V Step	900	860	850	840	V/µs	min	B
Rise Time and Fall Time	V_O = 1V Step	4.4				ns	typ	C
Harmonic Distortion	G = +5, V_O = 2V$_{PP}$, 5MHz							
2nd-Harmonic	R_L = 500Ω	–68	–55	–54	–53	dB	max	B
3rd-Harmonic	R_L = 500Ω	–57	–52	–51	–49	dB	max	B
Base Input Voltage Noise	f > 100kHz	2.4	3.0	3.3	3.4	nV/√Hz	max	B
Base Input Current Noise	f > 100kHz	1.65	2.4	2.45	2.5	pA/√Hz	max	B
Emitter Input Current Noise	f > 100kHz	5.2	15.3	16.6	17.5	pA/√Hz	max	B
OTA DC PERFORMANCE(4) (see Figure 48)								
Min OTA Transconductance	V_O = ±10mV, R_C = 0Ω, R_E = 0Ω	95	**80**	77	75	mA/V	min	A
Max OTA Transconductance	V_O = ±10mV, R_C = 0Ω, R_E = 0Ω	95	**150**	155	160	mA/V	min	A
B-Input Offset Voltage	V_B = 0V, R_C = 0Ω, R_E = 100Ω	±3	**±12**	±15	±20	mV	max	A
Average B-Input Offset Voltage Drift	V_B = 0V, R_C = 0Ω, R_E = 100Ω	±3		±67	±120	µV/°C	max	B
B-Input Bias Current	V_B = 0V, R_C = 0Ω, R_E = 100Ω	±1	**±5**	±6	±6.6	µA	max	A
Average B-Input Bias Current Drift	V_B = 0V, R_C = 0Ω, R_E = 100Ω			±20	±25	nA/°C	max	B
E-Input Bias Current	V_B = 0V, V_C = 0V	±30	**±100**	±125	±140	µA	max	A
Average E-Input Bias Current Drift	V_B = 0V, V_C = 0V			±500	±600	nA/°C	max	B
C-Output Bias Current	V_B = 0V, V_C = 0V	±5	**±18**	±30	±38	µA	max	A
Average C-Output Bias Current Drift	V_B = 0V, V_C = 0V			±250	±300	nA/°C	max	B
OTA INPUT (see Figure 48)								
B-Input Voltage Range		±4.2	±3.7	±3.6	±3.6	V	min	B
B-Input Impedance		455 || 2.1				kΩ || pF	typ	C
Min E-Input Input Resistance		10.5	12.5	13.0	13.3	Ω	min	B
Max E-Input Input Resistance		10.5	6.7	6.5	6.3	Ω	max	B

(1) Test levels: (A) 100% tested at +25°C. Over temperature limits set by characterization and simulation. (B) Limits set by characterization and simulation. (C) Typical value only for information.
(2) Junction temperature = ambient for +25°C specifications.
(3) Junction temperature = ambient at low temperature limit; junction temperature = ambient + 8°C at high temperature limit for over temperature specifications.
(4) Current is considered positive out of node. V_{CM} is the input common-mode voltage.

ELECTRICAL CHARACTERISTICS: V_S = ±5V (continued)

R_L = 500Ω and R_{ADJ} = 250Ω, unless otherwise noted.

PARAMETER	CONDITIONS	OPA860ID TYP +25°C	OPA860ID MIN/MAX OVER TEMPERATURE +25°C[2]	0°C to 70°C[3]	–40°C to +85°C[3]	UNITS	MIN/ MAX	TEST LEVEL[1]
OTA OUTPUT								
E-Output Voltage Compliance	I_E = ±1mA	±4.2	**±3.7**	±3.6	±3.6	V	min	A
E-Output Current, Sinking/Sourcing	V_E = 0	±15	**±10**	±9	±9	mA	min	A
C-Output Voltage Compliance	I_C = ±1mA	±4.7	**±4.0**	±3.9	±3.9	V	min	A
C-Output Current, Sinking/Sourcing	V_C = 0	±15	**±10**	±9	±9	mA	min	A
C-Output Impedance		54 || 2				kΩ || pF	typ	C
BUFFER (see Figure Figure 45)								
AC PERFORMANCE								
Bandwidth	V_O = 200mV$_{PP}$	1200	750	720	700	MHz	min	B
	V_O = 1V$_{PP}$	1600				MHz	typ	C
	V_O = 5V$_{PP}$	1000				MHz	typ	C
Slew Rate	V_O = 5V Step	4000	3500	3200	3000	V/µs	min	B
Rise Time and Fall Time	V_O = 1V Step	0.4				ns	typ	C
Settling Time to 0.05%	V_O = 1V Step	6				ns	typ	C
Harmonic Distortion	V_O = 2V$_{PP}$, 5MHz							
2nd-Harmonic	R_L = 100Ω	–52	–47	–46	–44	dBc	max	B
	R_L ≥ 500Ω	–72	–65	–63	–61	dBc	max	B
3rd-Harmonic	R_L = 100Ω	–67	–63	–63	–62	dBc	max	B
	R_L ≥ 500Ω	–96	–86	–85	–83	dBc	max	B
Input Voltage Noise	f > 100kHz	4.8	5.1	5.6	6.0	nV/√Hz	max	B
Input Current Noise	f > 100kHz	2.1	2.6	2.7	2.8	pA/√Hz	max	B
Differential Gain	NTSC, PAL	0.06				%	typ	C
Differential Phase	NTSC, PAL	0.02				Degrees	typ	C
BUFFER DC PERFORMANCE								
Gain	R_L = 500Ω	1	**0.98**	0.98	0.98	V/V	min	A
	R_L = 500Ω	1	**1**	1	1	V/V	max	A
Input Offset Voltage		±16	**±30**	±36	±38	mV	max	A
Average Input Offset Voltage Drift				±125	±125	µV/°C	max	B
Input Bias Current		±3	**±7**	±8	±8.5	µA	max	A
Average Input Bias Current Drift				±20	±20	nA/°C	max	B
BUFFER INPUT								
Input Impedance		1.0 || 2.1				MΩ || pF	typ	C
BUFFER OUTPUT								
Output Voltage Swing	R_L = 500Ω	±4.0	**±3.8**	±3.8	±3.8	V	min	A
Output Current	V_O = 0	±60	**±50**	±49	±48	mA	min	A
Closed-Loop Output Impedance	f ≤ 100kHz	1.4				Ω	typ	C
POWER SUPPLY (OTA + BUFFER)								
Specified Operating Voltage		±5				V	typ	C
Maximum Operating Voltage			**±6.5**	±6.5	±6.5	V	max	A
Minimum Operating Voltage			**±2.5**	±2.5	±2.5	V	min	B
Maximum Quiescent Current	R_{ADJ} = 250Ω	11.2	**12**	13.5	14.5	mA	max	A
Minimum Quiescent Current	R_{ADJ} = 250Ω	11.2	**10.5**	9.5	7.9	mA	min	A
OTA Power-Supply Rejection Ratio (+PSRR)	$\Delta I_O/\Delta V_S$	±20	**±50**	±60	±65	µA/V	max	A
Buffer Power-Supply Rejection Ratio (–PSRR)	$\Delta V_O/\Delta V_S$	54	**48**	46	45	dB	min	A
THERMAL CHARACTERISTICS								
Specification: ID		–40 to +85				°C	typ	C
Thermal Resistance θ_{JA}								
D SO-8	Junction-to-Ambient	125				°C/W	typ	C

TYPICAL CHARACTERISTICS: V_S = ±5V

At T_A = +25°C, I_Q = 11.2mA, and R_L = 500Ω, unless otherwise noted. (See Figure 53.)

OTA + BUF Performance

SMALL-SIGNAL FREQUENCY RESPONSE

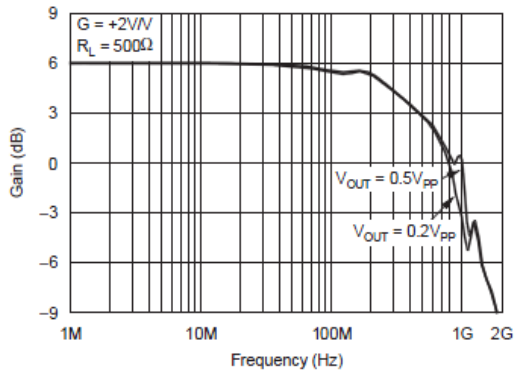


Figure 1.

LARGE-SIGNAL FREQUENCY RESPONSE

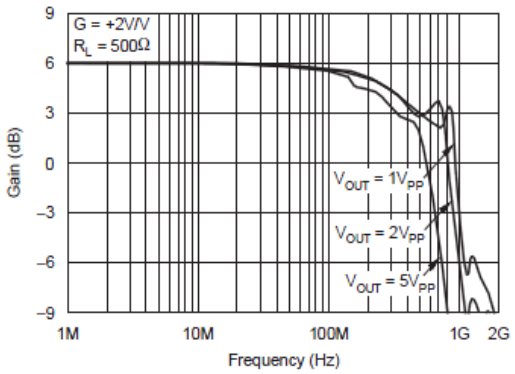


Figure 2.

SMALL-SIGNAL FREQUENCY RESPONSE vs QUIESCENT CURRENT

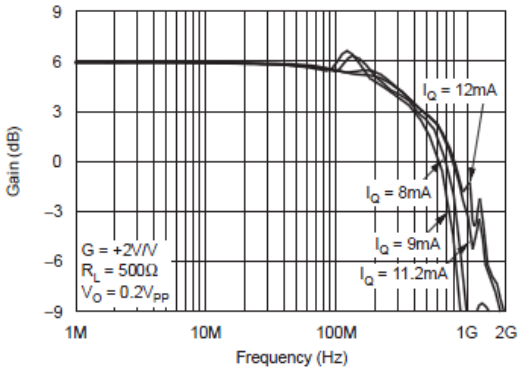


Figure 3.

GAIN FLATNESS vs QUIESCENT CURRENT

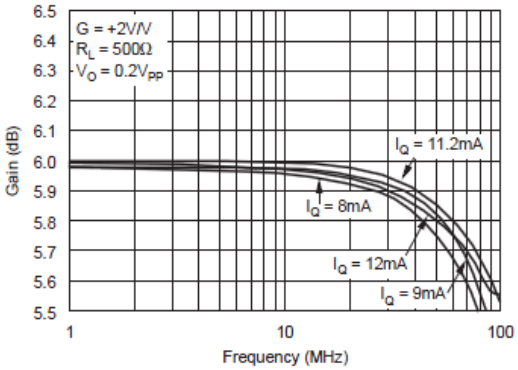


Figure 4.

SMALL-SIGNAL PULSE RESPONSE

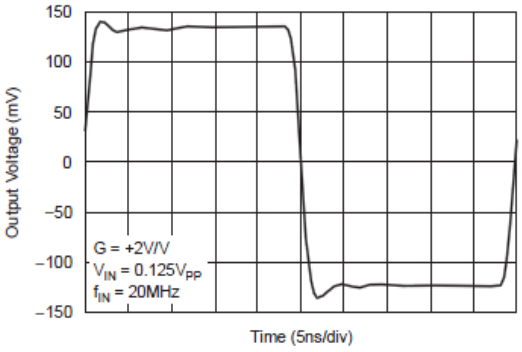


Figure 5.

LARGE-SIGNAL PULSE RESPONSE

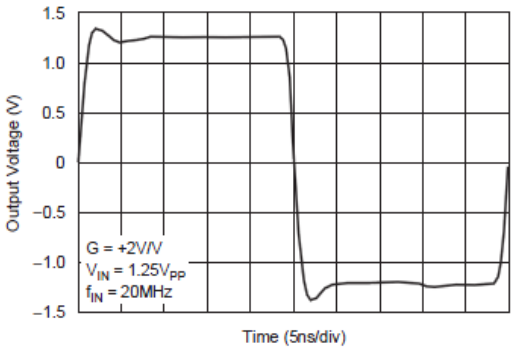


Figure 6.

TYPICAL CHARACTERISTICS: V_S = ±5V (continued)

At T_A = +25°C, I_Q = 11.2mA, and R_L = 500Ω, unless otherwise noted. (See Figure 53.)

HARMONIC DISTORTION vs FREQUENCY

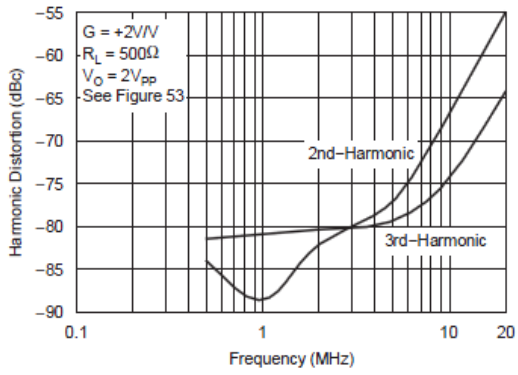


Figure 7.

HARMONIC DISTORTION vs OUTPUT RESISTANCE

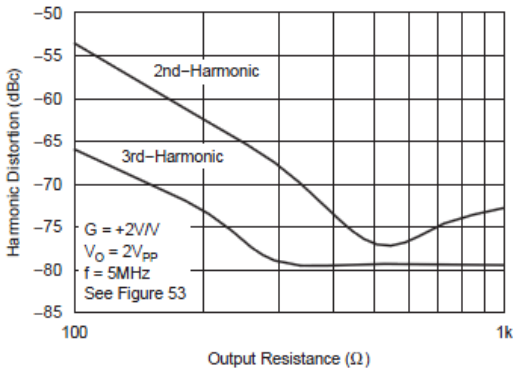


Figure 8.

HARMONIC DISTORTION vs OUTPUT VOLTAGE

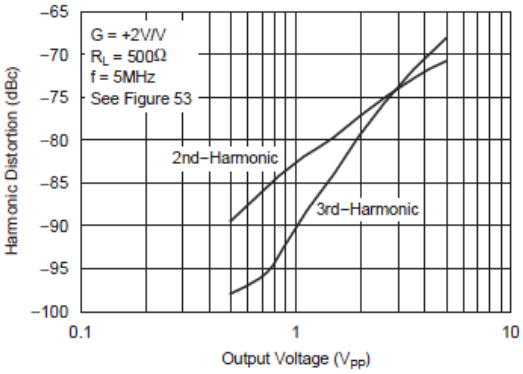


Figure 9.

HARMONIC DISTORTION vs SUPPLY VOLTAGE

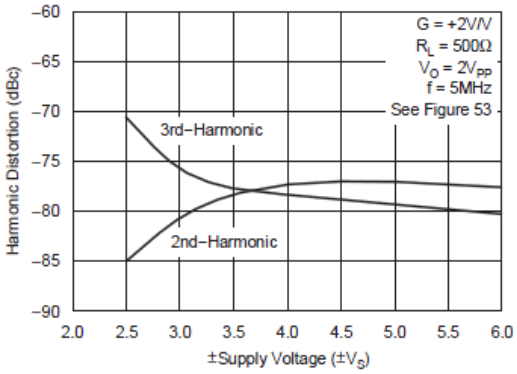


Figure 10.

HARMONIC DISTORTION vs QUIESCENT CURRENT

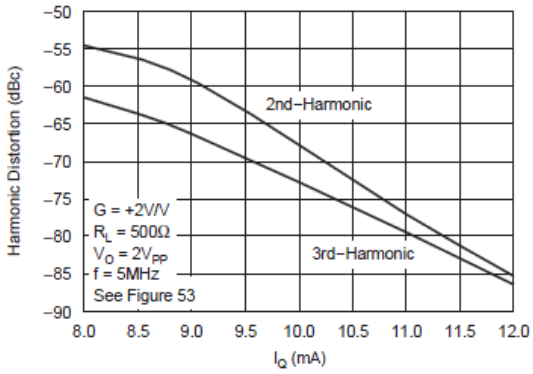


Figure 11.

QUIESCENT CURRENT vs R_{ADJ}

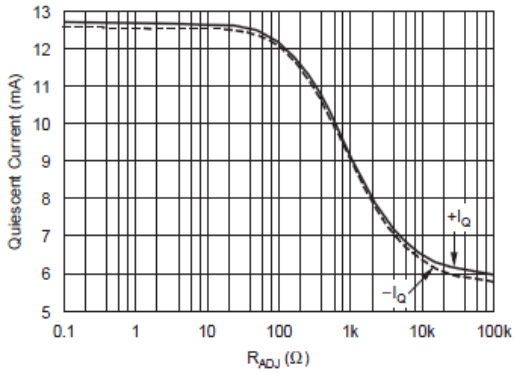


Figure 12.

TYPICAL CHARACTERISTICS: V_S = ±5V

At T_A = +25°C, I_Q = 11.2mA, and R_L = 500Ω, unless otherwise noted.

OTA Performance

OTA TRANSCONDUCTANCE vs FREQUENCY

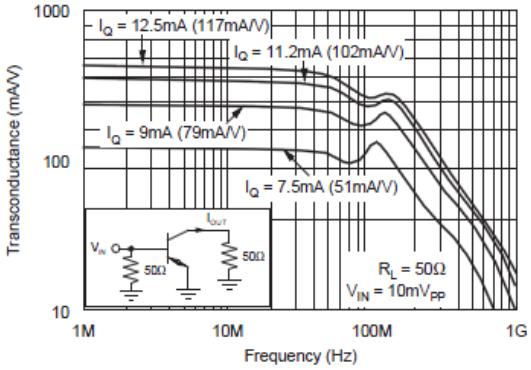


Figure 13.

OTA TRANSCONDUCTANCE vs QUIESCENT CURRENT

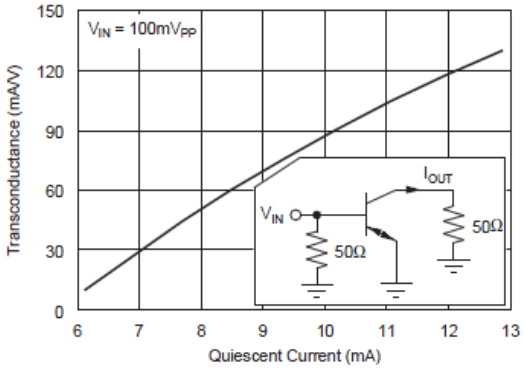


Figure 14.

OTA TRANSCONDUCTANCE vs INPUT VOLTAGE

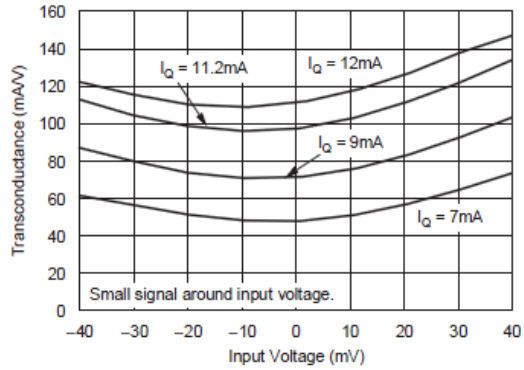


Figure 15.

OTA TRANSFER CHARACTERISTICS

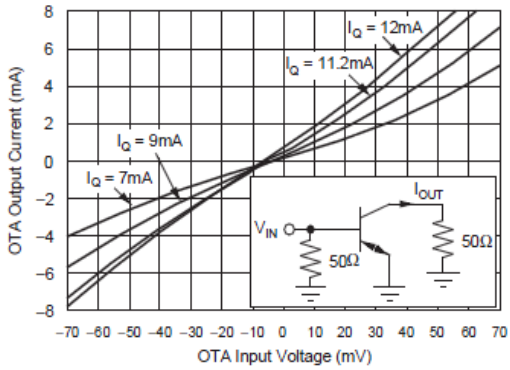


Figure 16.

OTA SMALL-SIGNAL PULSE RESPONSE

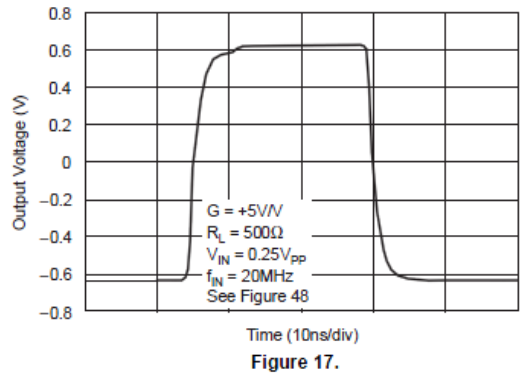


Figure 17.

OTA LARGE-SIGNAL PULSE RESPONSE

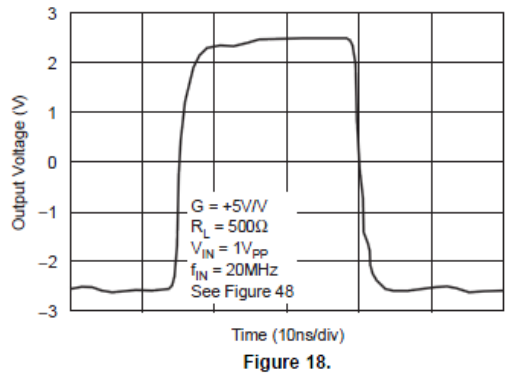


Figure 18.

TYPICAL CHARACTERISTICS: V_S = ±5V (continued)

At T_A = +25°C, I_Q = 11.2mA, and R_L = 500Ω, unless otherwise noted.

B-INPUT RESISTANCE vs QUIESCENT CURRENT

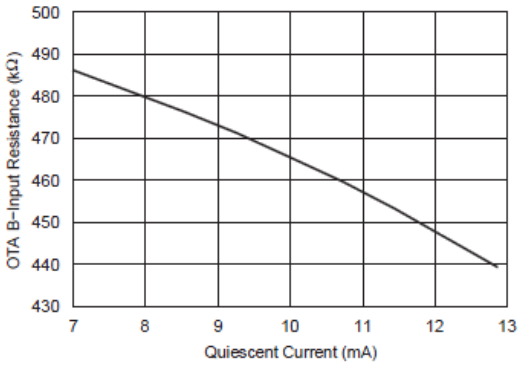


Figure 19.

C-OUTPUT RESISTANCE vs QUIESCENT CURRENT

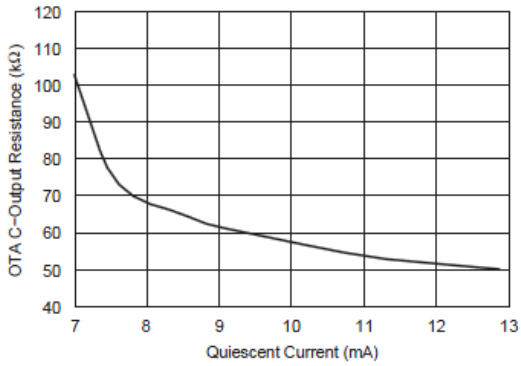


Figure 20.

E-OUTPUT RESISTANCE vs QUIESCENT CURRENT

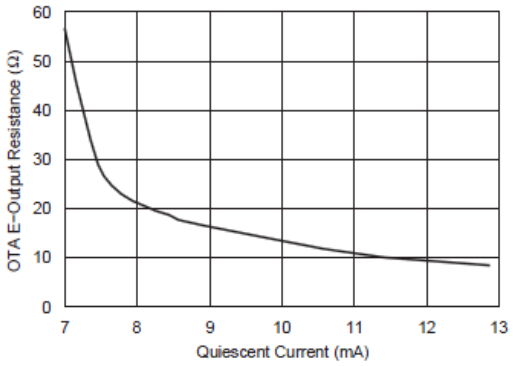


Figure 21.

INPUT VOLTAGE AND CURRENT NOISE DENSITY

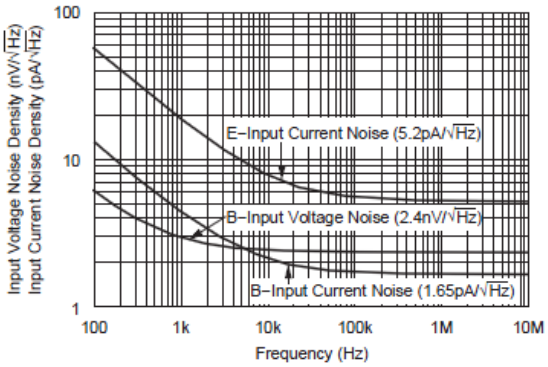


Figure 22.

1MHz OTA VOLTAGE AND CURRENT NOISE DENSITY vs QUIESCENT CURRENT ADJUST RESISTOR

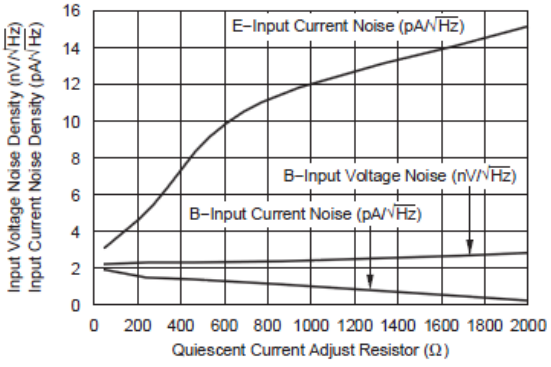


Figure 23.

TYPICAL CHARACTERISTICS: V_S = ±5V

At T_A = +25°C, I_Q = 11.2mA, and R_L = 500Ω, unless otherwise noted.

BUF Performance

BUFFER BANDWIDTH vs OUTPUT VOLTAGE

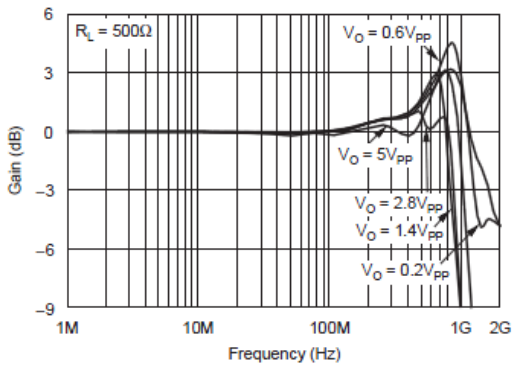


Figure 24.

BUFFER BANDWIDTH vs LOAD RESISTANCE

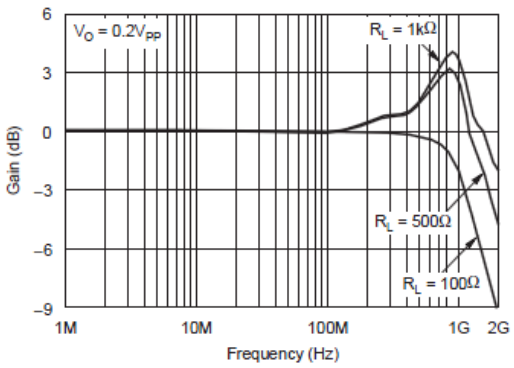


Figure 25.

BUFFER GAIN FLATNESS

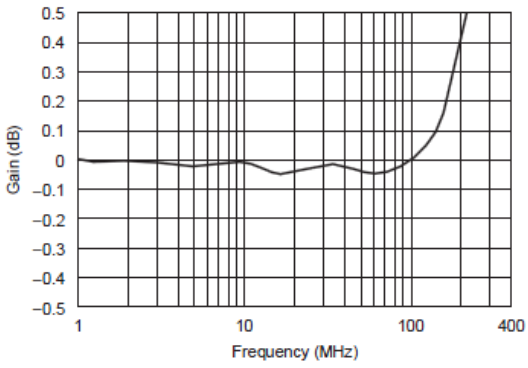


Figure 26.

BUFFER SMALL-SIGNAL PULSE RESPONSE

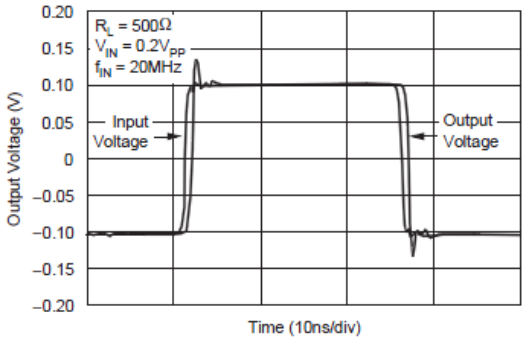


Figure 27.

BUFFER LARGE-SIGNAL PULSE RESPONSE

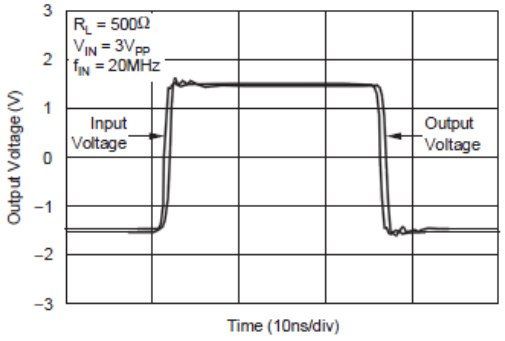


Figure 28.

HARMONIC DISTORTION vs FREQUENCY

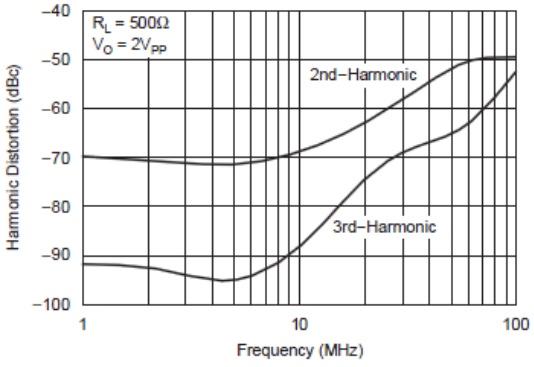


Figure 29.

TYPICAL CHARACTERISTICS: V_S = ±5V (continued)

At T_A = +25°C, I_Q = 11.2mA, and R_L = 500Ω, unless otherwise noted.

5MHz HARMONIC DISTORTION vs LOAD RESISTANCE

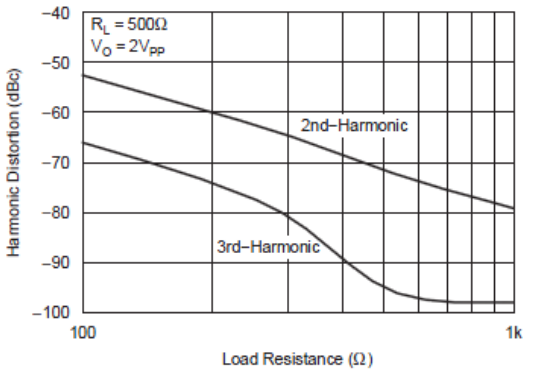


Figure 30.

HARMONIC DISTORTION vs OUTPUT VOLTAGE

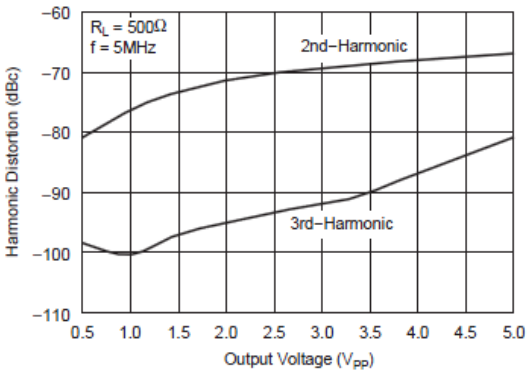


Figure 31.

5MHz HARMONIC DISTORTION vs SUPPLY VOLTAGE

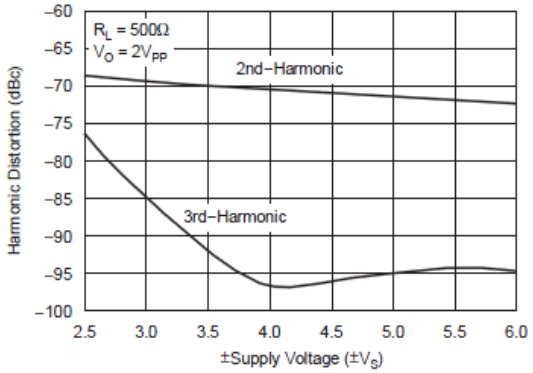


Figure 32.

BUFFER TRANSFER FUNCTION

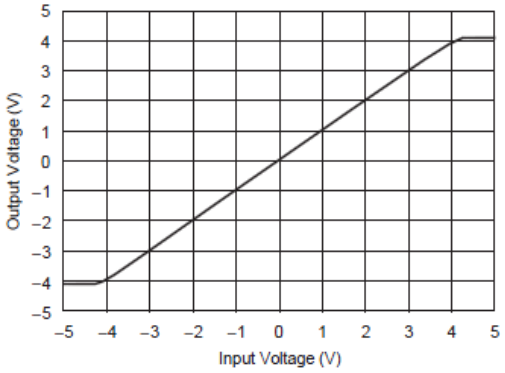


Figure 33.

INPUT VOLTAGE AND CURRENT NOISE DENSITY

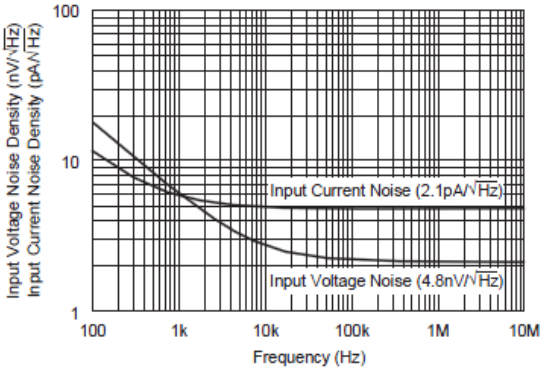


Figure 34.

BUFFER OUTPUT IMPEDANCE

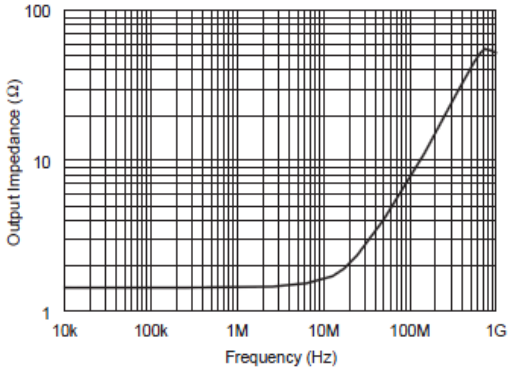


Figure 35.

TYPICAL CHARACTERISTICS: V_S = ±5V (continued)

At T_A = +25°C, I_Q = 11.2mA, and R_L = 500Ω, unless otherwise noted.

BUFFER GROUP DELAY TIME vs FREQUENCY

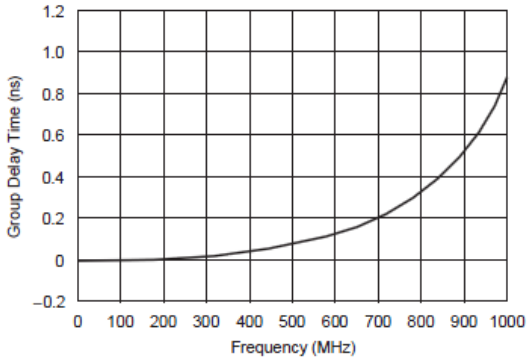


Figure 36.

BUFFER OUTPUT VOLTAGE AND CURRENT LIMITATIONS

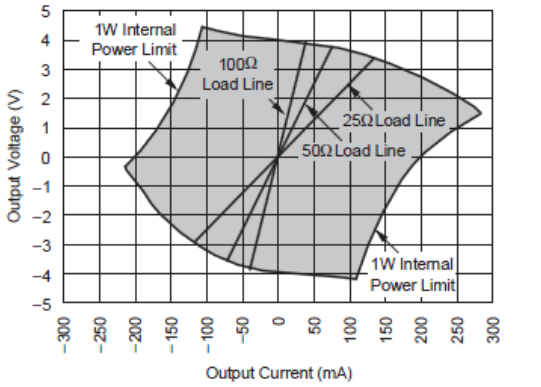


Figure 37.

QUIESCENT CURRENT vs TEMPERATURE

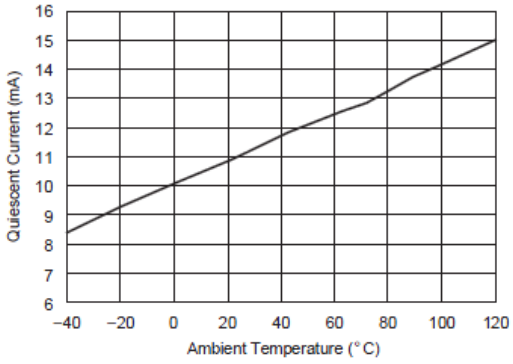


Figure 38.

POWER-SUPPLY REJECTION RATIO vs FREQUENCY

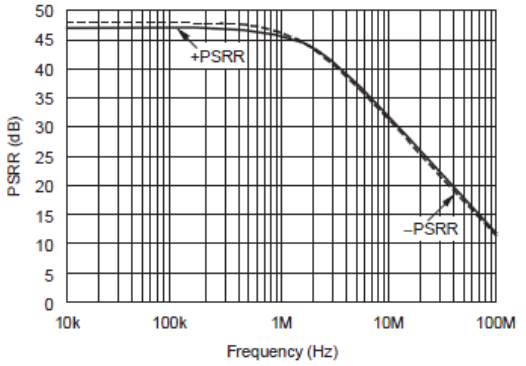


Figure 39.

VOLTAGE RANGE vs TEMPERATURE

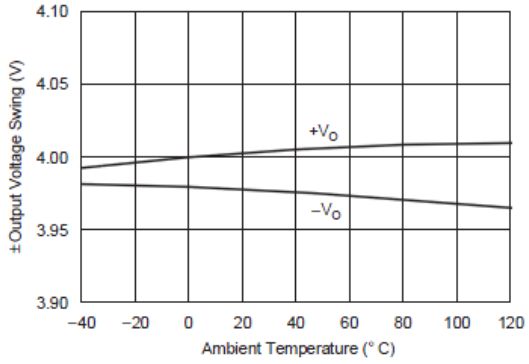


Figure 40.

OUTPUT CURRENT vs TEMPERATURE

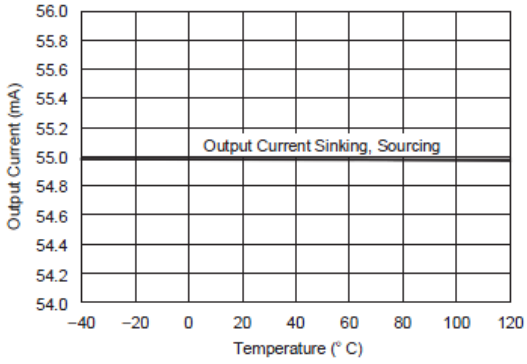


Figure 41.

TYPICAL CHARACTERISTICS: V_S = ±5V (continued)

At T_A = +25°C, I_Q = 11.2mA, and R_L = 500Ω, unless otherwise noted.

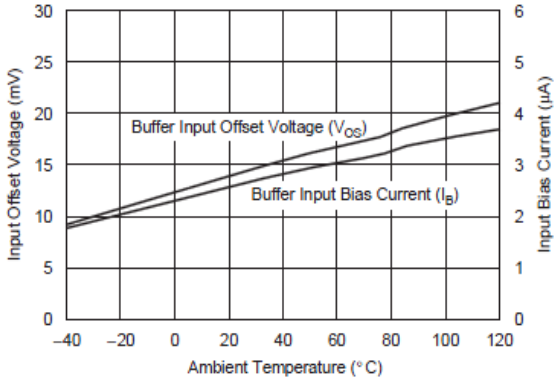


Figure 42.

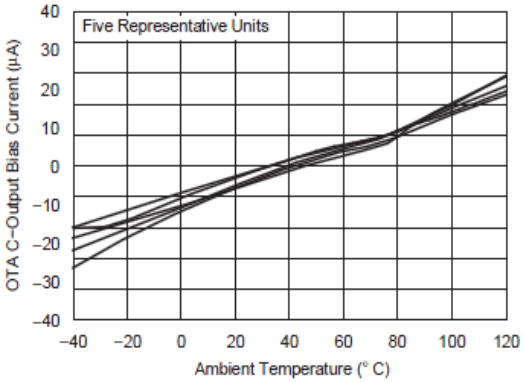


Figure 43.

8.2 ANEXO 2: DATASHEET FPGA SPARTAN 3E XC3S500E

XILINX®

Spartan-3E FPGA Family: Data Sheet

DS312 (v3.8) August 26, 2009

Product Specification

Module 1: Spartan-3E FPGA Family: Introduction and Ordering Information

DS312-1 (v3.8) August 26, 2009

- Introduction
- Features
- Architectural Overview
- Package Marking
- Ordering Information

Module 2: Functional Description

DS312-2 (v3.8) August 26, 2009

- Input/Output Blocks (IOBs)
  - Overview
  - SelectIO™ Signal Standards
- Configurable Logic Block (CLB)
- Block RAM
- Dedicated Multipliers
- Digital Clock Manager (DCM)
- Clock Network
- Configuration
- Powering Spartan®-3E FPGAs
- Production Stepping

Module 3: DC and Switching Characteristics

DS312-3 (v3.8) August 26, 2009

- DC Electrical Characteristics
  - Absolute Maximum Ratings
  - Supply Voltage Specifications
  - Recommended Operating Conditions
  - DC Characteristics
- Switching Characteristics
  - I/O Timing
  - SLICE Timing
  - DCM Timing
  - Block RAM Timing
  - Multiplier Timing
  - Configuration and JTAG Timing

Module 4: Pinout Descriptions

DS312-4 (v3.8) August 26, 2009

- Pin Descriptions
- Package Overview
- Pinout Tables
- Footprint Diagrams

Spartan-3E FPGA Family: Introduction and Ordering Information

DS312-1 (v3.8) August 26, 2009

Product Specification

Introduction

The Spartan®-3E family of Field-Programmable Gate Arrays (FPGAs) is specifically designed to meet the needs of high volume, cost-sensitive consumer electronic applications. The five-member family offers densities ranging from 100,000 to 1.6 million system gates, as shown in Table 1.

The Spartan-3E family builds on the success of the earlier Spartan-3 family by increasing the amount of logic per I/O, significantly reducing the cost per logic cell. New features improve system performance and reduce the cost of configuration. These Spartan-3E FPGA enhancements, combined with advanced 90 nm process technology, deliver more functionality and bandwidth per dollar than was previously possible, setting new standards in the programmable logic industry.

Because of their exceptionally low cost, Spartan-3E FPGAs are ideally suited to a wide range of consumer electronics applications, including broadband access, home networking, display/projection, and digital television equipment.

The Spartan-3E family is a superior alternative to mask programmed ASICs. FPGAs avoid the high initial cost, the lengthy development cycles, and the inherent inflexibility of conventional ASICs. Also, FPGA programmability permits design upgrades in the field with no hardware replacement necessary, an impossibility with ASICs.

Features

- Very low cost, high-performance logic solution for high-volume, consumer-oriented applications
- Proven advanced 90-nanometer process technology
- Multi-voltage, multi-standard SelectIO™ interface pins
  - Up to 376 I/O pins or 156 differential signal pairs
  - LVCMOS, LVTTL, HSTL, and SSTL single-ended signal standards
  - 3.3V, 2.5V, 1.8V, 1.5V, and 1.2V signaling
  - 622+ Mb/s data transfer rate per I/O
  - True LVDS, RSDS, mini-LVDS, differential HSTL/SSTL differential I/O
  - Enhanced Double Data Rate (DDR) support
  - DDR SDRAM support up to 333 Mb/s
- Abundant, flexible logic resources
  - Densities up to 33,192 logic cells, including optional shift register or distributed RAM support
  - Efficient wide multiplexers, wide logic
  - Fast look-ahead carry logic
  - Enhanced 18 x 18 multipliers with optional pipeline
  - IEEE 1149.1/1532 JTAG programming/debug port
- Hierarchical SelectRAM™ memory architecture
  - Up to 648 Kbits of fast block RAM
  - Up to 231 Kbits of efficient distributed RAM
- Up to eight Digital Clock Managers (DCMs)
  - Clock skew elimination (delay locked loop)
  - Frequency synthesis, multiplication, division
  - High-resolution phase shifting
  - Wide frequency range (5 MHz to over 300 MHz)
- Eight global clocks plus eight additional clocks per each half of device, plus abundant low-skew routing
- Configuration interface to industry-standard PROMs
  - Low-cost, space-saving SPI serial Flash PROM
  - x8 or x8/x16 parallel NOR Flash PROM
  - Low-cost Xilinx® Platform Flash with JTAG
- Complete Xilinx ISE® and WebPACK™ software
- MicroBlaze™ and PicoBlaze™ embedded processor cores
- Fully compliant 32-/64-bit 33 MHz PCI support (66 MHz in some devices)
- Low-cost QFP and BGA packaging options
  - Common footprints support easy density migration
  - Pb-free packaging options
- XA Automotive version available

Table 1: **Summary of Spartan-3E FPGA Attributes**

Device	System Gates	Equivalent Logic Cells	CLB Array (One CLB = Four Slices)				Distributed RAM bits(1)	Block RAM bits(1)	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
			Rows	Columns	Total CLBs	Total Slices						
XC3S100E	100K	2,160	22	16	240	960	15K	72K	4	2	108	40
XC3S250E	250K	5,508	34	26	612	2,448	38K	216K	12	4	172	68
XC3S500E	500K	10,476	46	34	1,164	4,656	73K	360K	20	4	232	92
XC3S1200E	1200K	19,512	60	46	2,168	8,672	136K	504K	28	8	304	124
XC3S1600E	1600K	33,192	76	58	3,688	14,752	231K	648K	36	8	376	156

Notes:

1. By convention, one Kb is equivalent to 1,024 bits.

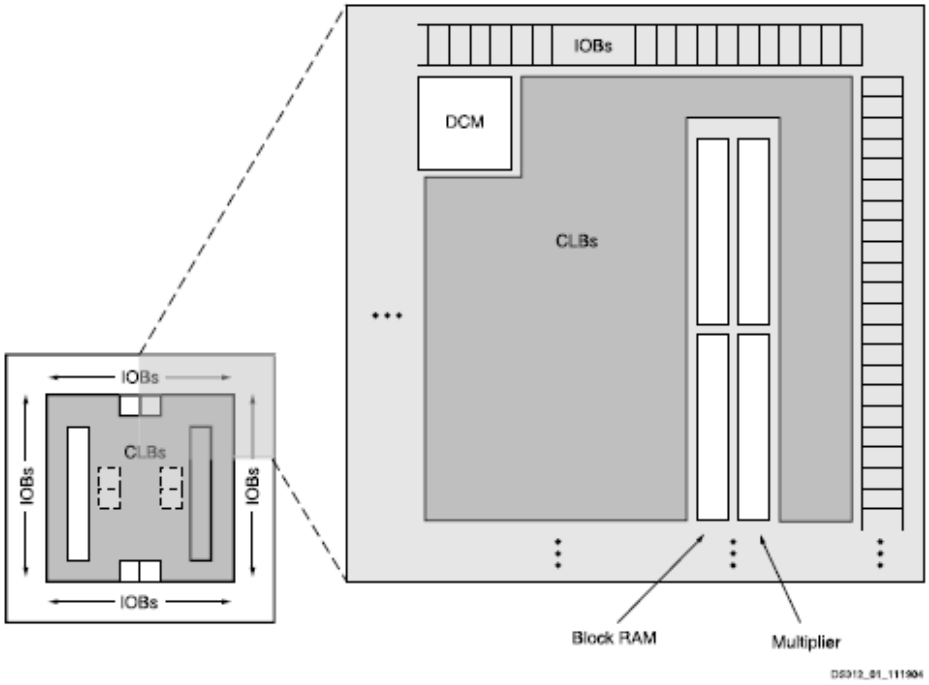
Architectural Overview

The Spartan-3E family architecture consists of five fundamental programmable functional elements:

- **Configurable Logic Blocks (CLBs)** contain flexible Look-Up Tables (LUTs) that implement logic plus storage elements used as flip-flops or latches. CLBs perform a wide variety of logical functions as well as store data.
- **Input/Output Blocks (IOBs)** control the flow of data between the I/O pins and the internal logic of the device. Each IOB supports bidirectional data flow plus 3-state operation. Supports a variety of signal standards, including four high-performance differential standards. Double Data-Rate (DDR) registers are included.
- **Block RAM** provides data storage in the form of 18-Kbit dual-port blocks.
- **Multiplier Blocks** accept two 18-bit binary numbers as inputs and calculate the product.
- **Digital Clock Manager (DCM) Blocks** provide self-calibrating, fully digital solutions for distributing, delaying, multiplying, dividing, and phase-shifting clock signals.

These elements are organized as shown in Figure 1. A ring of IOBs surrounds a regular array of CLBs. Each device has two columns of block RAM except for the XC3S100E, which has one column. Each RAM column consists of several 18-Kbit RAM blocks. Each block RAM is associated with a dedicated multiplier. The DCMs are positioned in the center with two at the top and two at the bottom of the device. The XC3S100E has only one DCM at the top and bottom, while the XC3S1200E and XC3S1600E add two DCMs in the middle of the left and right sides.

The Spartan-3E family features a rich network of traces that interconnect all five functional elements, transmitting signals among them. Each functional element has an associated switch matrix that permits multiple connections to the routing.



Notes:

1. The XC3S1200E and XC3S1600E have two additional DCMs on both the left and right sides as indicated by the dashed lines. The XC3S100E has only one DCM at the top and one at the bottom.

Figure 1: **Spartan-3E Family Architecture**

Configuration

Spartan-3E FPGAs are programmed by loading configuration data into robust, reprogrammable, static CMOS configuration latches (CCLs) that collectively control all functional elements and routing resources. The FPGA's configuration data is stored externally in a PROM or some other non-volatile medium, either on or off the board. After applying power, the configuration data is written to the FPGA using any of seven different modes:

- Master Serial from a Xilinx Platform Flash PROM
- Serial Peripheral Interface (SPI) from an industry-standard SPI serial Flash
- Byte Peripheral Interface (BPI) Up or Down from an industry-standard x8 or x8/x16 parallel NOR Flash
- Slave Serial, typically downloaded from a processor
- Slave Parallel, typically downloaded from a processor
- Boundary Scan (JTAG), typically downloaded from a processor or system tester.

Furthermore, Spartan-3E FPGAs support MultiBoot configuration, allowing two or more FPGA configuration bitstreams to be stored in a single parallel NOR Flash. The FPGA application controls which configuration to load next and when to load it.

I/O Capabilities

The Spartan-3E FPGA SelectIO interface supports many popular single-ended and differential standards. Table 2 shows the number of user I/Os as well as the number of differential I/O pairs available for each device/package combination.

Spartan-3E FPGAs support the following single-ended standards:

- 3.3V low-voltage TTL (LVTTL)
- Low-voltage CMOS (LVCMOS) at 3.3V, 2.5V, 1.8V, 1.5V, or 1.2V
- 3V PCI at 33 MHz, and in some devices, 66 MHz
- HSTL I and III at 1.8V, commonly used in memory applications
- SSTL I at 1.8V and 2.5V, commonly used for memory applications

Spartan-3E FPGAs support the following differential standards:

- LVDS
- Bus LVDS
- mini-LVDS
- RSDS
- Differential HSTL (1.8V, Types I and III)
- Differential SSTL (2.5V and 1.8V, Type I)
- 2.5V LVPECL inputs

Table 2: **Available User I/Os and Differential (Diff) I/O Pairs**

Package	VQ100 VQG100		CP132 CPG132		TQ144 TQG144		PQ208 PQG208		FT256 FTG256		FG320 FGG320		FG400 FGG400		FG484 FGG484	
Size (mm)	16 x 16		8 x 8		22 x 22		28 x 28		17 x 17		19 x 19		21 x 21		23 x 23	
Device	User	Diff	User	Diff	User	Diff	User	Diff	User	Diff	User	Diff	User	Diff	User	Diff
XC3S100E	**66** *(7)*	**30** *(2)*	**83** *(11)*	**35** *(2)*	**108** *(28)*	**40** *(4)*	-	-	-	-	-	-	-	-	-	-
XC3S250E	**66** *(7)*	**30** *(2)*	**92** *(7)*	**41** *(2)*	**108** *(28)*	**40** *(4)*	**158** *(32)*	**65** *(5)*	**172** *(40)*	**68** *(8)*	-	-	-	-	-	-
XC3S500E	**66**[3] *(7)*	**30** *(2)*	**92** *(7)*	**41** *(2)*	-	-	**158** *(32)*	**65** *(5)*	**190** *(41)*	**77** *(8)*	**232** *(56)*	**92** *(12)*	-	-	-	-
XC3S1200E	-	-	-	-	-	-	-	-	**190** *(40)*	**77** *(8)*	**250** *(56)*	**99** *(12)*	**304** *(72)*	**124** *(20)*	-	-
XC3S1600E	-	-	-	-	-	-	-	-	-	-	**250** *(56)*	**99** *(12)*	**304** *(72)*	**124** *(20)*	**376** *(82)*	**156** *(21)*

Notes:

1. All Spartan-3E devices provided in the same package are pin-compatible as further described in Module 4: **Pinout Descriptions**.
2. The number shown in **bold** indicates the maximum number of I/O and input-only pins. The number shown in (*italics*) indicates the number of input-only pins.
3. The XC3S500E is available in the VQG100 Pb-free package and not the standard VQ100. The VQG100 and VQ100 pin-outs are identical and general references to the VQ100 will apply to the XC3S500E.

Package Marking

Figure 2 provides a top marking example for Spartan-3E FPGAs in the quad-flat packages. Figure 3 shows the top marking for Spartan-3E FPGAs in BGA packages except the 132-ball chip-scale package (CP132 and CPG132). The markings for the BGA packages are nearly identical to those for the quad-flat packages, except that the marking is rotated with respect to the ball A1 indicator. Figure 4 shows the top marking for Spartan-3E FPGAs in the CP132 and CPG132 packages.

On the QFP and BGA packages, the optional numerical Stepping Code follows the Lot Code.

The "5C" and "4I" part combinations can have a dual mark of "5C/4I". Devices with a single mark are only guaranteed for the marked speed grade and temperature range. All "5C" and "4I" part combinations use the Stepping 1 production silicon.

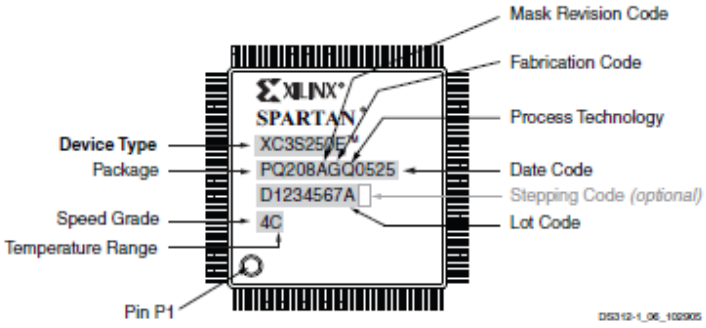


Figure 2: **Spartan-3E QFP Package Marking Example**

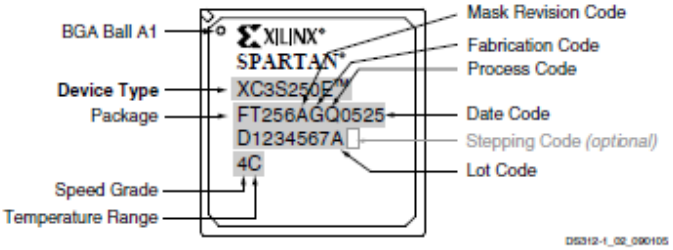


Figure 3: **Spartan-3E BGA Package Marking Example**

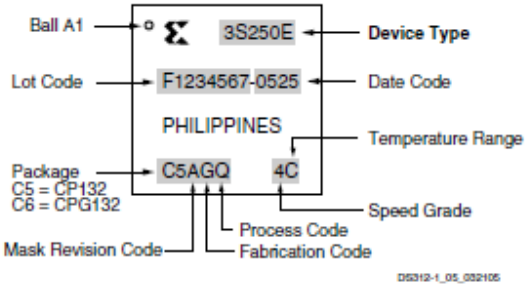
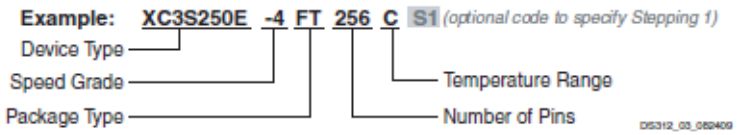


Figure 4: **Spartan-3E CP132 and CPG132 Package Marking Example**

Ordering Information

Spartan-3E FPGAs are available in both standard and Pb-free packaging options for all device/package combinations. All devices are available in Pb-free packages, which adds a 'G' character to the ordering code. All devices are available in either Commercial (C) or Industrial (I) temperature ranges. Both the standard –4 and faster –5 speed grades are available for the Commercial temperature range. However, only the –4 speed grade is available for the Industrial temperature range. See Table 2 for valid device/package combinations.



Device	Speed Grade		Package Type / Number of Pins		Temperature Range (T_J)	
XC3S100E	–4	Standard Performance	VQ100 VQG100	100-pin Very Thin Quad Flat Pack (VQFP)	C	Commercial (0°C to 85°C)
XC3S250E	–5	High Performance	CP132 CPG132	132-ball Chip-Scale Package (CSP)	I	Industrial (–40°C to 100°C)
XC3S500E			TQ144 TQG144	144-pin Thin Quad Flat Pack (TQFP)		
XC3S1200E			PQ208 PQG208	208-pin Plastic Quad Flat Pack (PQFP)		
XC3S1600E			FT256 FTG256	256-ball Fine-Pitch Thin Ball Grid Array (FTBGA)		
			FG320 FGG320	320-ball Fine-Pitch Ball Grid Array (FBGA)		
			FG400 FGG400	400-ball Fine-Pitch Ball Grid Array (FBGA)		
			FG484 FGG484	484-ball Fine-Pitch Ball Grid Array (FBGA)		

Notes:

1. The –5 speed grade is exclusively available in the Commercial temperature range.
2. The XC3S500E VQG100 is available only in the -4 Speed Grade.
3. See DS635 for the XA Automotive Spartan-3E FPGAs.

Production Stepping

The Spartan-3E FPGA family uses production stepping to indicate improved capabilities or enhanced features.

Stepping 1 is, by definition, a functional superset of Stepping 0. Furthermore, configuration bitstreams generated for any stepping are forward compatible. See Table 72 for additional details.

Xilinx has shipped both Stepping 0 and Stepping 1. Designs operating on the Stepping 0 devices perform similarly on a Stepping 1 device. Stepping 1 devices have been shipping since 2006. The faster speed grade (-5), Industrial (I grade), Automotive devices, and -4C devices with date codes 0901 (2009) and later, are always Stepping 1 devices. Only -4C devices have shipped as Stepping 0 devices.

To specify only the later stepping for the -4C, append an *S#* suffix to the standard ordering code, where *#* is the stepping number, as indicated in Table 3.

Table 3: **Spartan-3E Optional Stepping Level Ordering**

Stepping Number	Suffix Code	Status
0	None or S0	Production
1	S1	Production

The stepping level is optionally marked on the device using a single number character, as shown in Figure 2, Figure 3, and Figure 4.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/01/05	1.0	Initial Xilinx release.
03/21/05	1.1	Added XC3S250E in CP132 package to Table 2. Corrected number of differential I/O pairs for CP132 package. Added package markings for QFP packages (Figure 2) and CP132/CPG132 packages (Figure 4).
11/23/05	2.0	Added differential HSTL and SSTL I/O standards. Updated Table 2 to indicate number of input-only pins. Added **Production Stepping** information, including example top marking diagrams.
03/22/06	3.0	Upgraded data sheet status to Preliminary. Added XC3S100E in CP132 package and updated I/O counts for the XC3S1600E in FG320 package (Table 2). Added information about dual markings for –5C and –4I product combinations to **Package Marking**.
11/09/06	3.4	Added 66 MHz PCI support and links to the Xilinx PCI LogiCORE data sheet. Indicated that Stepping 1 parts are Production status. Promoted Module 1 to Production status. Synchronized all modules to v3.4.
04/18/08	3.7	Added XC3S500E VQG100 package. Added reference to XA Automotive version. Updated links.
08/26/09	3.8	Added paragraph to **Configuration** indicating the device supports MultiBoot configuration. Added package sizes to Table 2. Described the speed grade and temperature range guarantee for devices having a single mark in paragraph 3 under **Package Marking**. Deleted Pb-Free Packaging example under **Ordering Information**. Revised information under **Production Stepping**. Revised description of Table 3.