

**UNIVERSIDAD CATOLICA DE SANTIAGO DE
GUAYAQUIL**

Facultad de Educación Técnica para el
Desarrollo

Tesis de Grado

Previo a la Obtención del título de: Ingeniería en Telecomunicaciones

Mención: "Gestión Empresarial"

Tema:

Diseño e Implementación de un filtro digital (FIR) orientado al uso del laboratorio de procesamiento de señales digitales (PSD)

Realizado por:

Amaguaya Colcha Verónica

Burgos Acosta Luis

Espinoza Espinosa Byron

Fernández De La S Jorge

Nivela Villarroel Nadia

Torres Tobar Mildred

Director

Ing. Marcos Andrade

Guayaquil – Ecuador

2010 – 2011

TESIS DE GRADO

Título

**Presentado a la Facultad Técnica, Escuela de Ingeniería en
Telecomunicaciones de la Universidad Católica Santiago de
Guayaquil por:**

**Amaguaya Colcha Verónica
Burgos Acosta Luis
Espinoza Espinosa Byron
Fernández de la S Jorge
Nivela Villarroel Nadia
Torres Tobar Mildred**

**Para dar cumplimiento con uno de los requisitos para optar por el
título de:
Ingeniería en Telecomunicaciones**

**Mención:
“Gestión Empresarial”**

Miembros del tribunal:

*Ing. Héctor Cedeño Abad
Ing. Pedro Tutivén López
Ing. Marcos Andrade Reyes
Ing. Efraín Suárez Murillo
Ing. Washington Medina Moreira
Ing. Elías Andrade Díaz
Ing. Víctor del Valle Ramos
Dr. Kléber López Parrales*

AGRADECIMIENTOS

Agradecemos sinceramente:

Agradeciendo primeramente a nuestro Dios, quien nos da la vida y nos da fuerzas para seguir adelante, a nuestros padres quienes a lo largo de nuestras vida nos ha apoyado y motivado nuestra formación académica, creyeron en nosotros en todo momento y no dudaron de nuestras habilidades.

A nuestros profesores a quienes les debemos gran parte de nuestros conocimientos, gracias a su paciencia, enseñanza y finalmente un eterno agradecimiento a la prestigiosa universidad la cual abre sus puertas a jóvenes como nosotros, preparándonos para un futuro competitivo y formándonos como personas de bien.

Al ingeniero Marco Andrade, nuestro profesor, tutor y amigo, por su apoyo y conocimientos durante toda la tesis.

Al ingeniero Efraín Suarez, por su apoyo en todo momento durante toda nuestra tesis.

Esperamos, por el bien de la enseñanza, que pronto podamos contar con la mejor formación diseñando y ejecutando proyectos.

DEDICATORIAS

Dedico esta tesis a mi Dios, quien me dio la sabiduría e inteligencia al nacer.

A mis padres, quienes me apoyaron y confiaron en mí en todo momento.

A mi esposo e hijo, quienes han sido la inspiración y motivación para mi superación profesional. (Verónica)

Esta tesis es una parte de mi vida y comienzo de otras etapas por esto y más se la dedico a toda mi familia que siempre estuvo apoyándome en el trayecto de mi formación académica. (Luis)

Dedico a mis padres, por ser quienes velaron por mí en todo el trayecto universitario, y que me apoyaron en mi formación académica. (Nadia)

Dedico ante todo a Dios, por iluminar mi vida y darme fuerzas. A mis padres por su apoyo, paciencia y amor incondicional en todo momento. (Jorge)

A mi familia, por creer en mí y darme su apoyo en toda mi vida universitaria, quienes con su amor y dedicación me ayudaron a cumplir mis metas propuestas. (Byron)

Dedico este trabajo de Tesis aun ser omnipotente nuestro padre celestial q por medio de mis padres me ha dado la fortaleza y los recursos para poder culminar este trabajo. A mi madre y mejor amiga Rossy Tobar a mi padre Mario Torres y a mis hermanos Masiu y Judy . (Mildred)

PROLOGO

El presente trabajo se realizó con la finalidad de ser expuesto ante un jurado como tema de tesis para obtener el título de Ingeniería en Telecomunicaciones con Mención en Gestión Empresarial.

La elección del tema se hizo tomando en cuenta la necesidad de los estudiantes cuyos intereses y motivaciones van orientándose al diseño de proyectos utilizando tarjetas especializadas como es el caso de la Spartan 3E, en esta tesis se presenta dicha tarjeta y a su vez se explica el diseño de un filtro Digital (FIR) implementado en la FPGA (Spartan 3E de Xilinx), consta principalmente de periféricos conocidos por la totalidad de los usuarios; puertos VGA para la conexión de un monitor, PS2 para teclados, USBs, Ethernet, etc.

El diseño se realiza en un software que calcula los coeficientes del filtro y la reconfiguración del hardware. Para ello se considero la necesidad del estudiante a una mejor comprensión mediante la práctica, sin quedar dudas al respecto de la materia dada.

En esta tesis también se destaca principalmente la realización de prácticas debidamente documentadas para el aprendizaje y sobre todo el acercamiento al alumno con los dispositivos de lógica difusa que actualmente copan el mercado de las comunicaciones y la información. Siempre se mantuvo una visión general con la idea de presentar un proyecto que permita la interrelación de los estudiantes con la práctica y no quedar solo en teoría, ya que esto nos ayudara que la explicación sea más entendible.

RESUMEN

Antiguamente el diseño e implementación de un sistema digital complejo era muy tedioso y podría llevar mucho tiempo desde su inicio hasta su culminación.

Hoy en día con la aparición de las herramientas de síntesis y simulación los ingenieros especializados pueden terminar un diseño en corto tiempo disminuyendo costos y aumentando la productividad.

En este trabajo se muestra el diseño de un analizador lógico implementado en un FPGA utilizando como entrada de diseño VHDL.

La herramienta de síntesis y simulación que se utilizó fue el ISE 10.1 y pertenece a la empresa líder en lógica programable XILINX.

Entre sus características podemos mencionar la reducción de tamaño y costo, además de facilitar el diseño de sistemas complejos y el tiempo de diseño se reduce en gran medida.

Con esto el producto se puede enviar al campo de trabajo en poco tiempo y modificarlo si así se requiere, ya que un gran número de ellos son reprogramables

ANTECEDENTES

Como ya se mencionó en líneas anteriores, es gracias al avance de las computadoras modernas que el tratamiento digital de señales se ha expandido y se ha hecho cada vez más fuerte. Han contribuido con su velocidad al montaje de algoritmos de procesamiento para lograr entregar respuestas casi instantáneas, siempre dentro de los límites exigidos por el desarrollo en las diferentes aplicaciones en que han sido utilizadas.

Las computadoras no sólo han influido en el montaje de algoritmos, además se han convertido en una herramienta fundamental para los diseñadores, ya que gracias a éstos se ha conseguido elaborar mejores y variados algoritmos para el diseño de filtros y herramientas de proceso.

El diseño de filtros se ha apoyado en los dispositivos lógicos programables, los cuales han jugado un papel muy importante en el montaje de los filtros digitales, puesto que gracias a ellos se ha logrado un adecuado funcionamiento en tiempo real. El FPGA es uno de estos dispositivos, que posee la cualidad de reconfiguración, lo que permite realizar cambios en la arquitectura sin necesidad de producir variaciones en el montaje o en el software que se está operando, ya que existe todo esto gracias al desarrollo industrial y avances tecnológicos.

ÍNDICE GENERAL

AGRADECIMIENTOS	
DEDICATORIAS	
PROLOGO	
RESUMEN	
ANTECEDENTES	
INTRODUCCION	1
CAPITULO 1	
1. FILTRO DIGITAL	
1.1. INTRODUCCIÓN AL DISEÑO DE FILTROS DIGITALES	3
1.2. MECANISMO DEL PROCESAMIENTO DEL FILTRO DIGITAL	4
1.3. TIPOS DE FILTROS	5
1.3.1. De acuerdo a la ganancia	7
1.3.2. De acuerdo a su respuesta de frecuencia	8
1.3.3. De acuerdo al método de diseño	14
1.3.4. De acuerdo a su aplicación	19
1.3.5. Otros tipos	19
1.4. IMPORTANCIA EN PROCESAMIENTO DIGITAL DE SEÑALES	26
1.4.1. Que es DSP	26
1.4.2. Aplicaciones de DSP	28
1.4.3. Importancias de las transformadas en el DSP	28
1.4.4. Arquitectura	29
1.4.5. Arquitecturas Estándar en el DSP	30
CAPITULO 2	
2. HERRAMIENTAS PARA EL FPGA Y SU ENTORNO DE DESARROLLO	31
2.1. ELEMENTOS QUE CONFORMAN LOS FPGAS DE XILINX	32
2.2. CARACTERISTICAS DE LA FPGA	34
2.3. PROGRAMACION FPGA	35
2.4. TECNOLOGIA DE LA MEMORIA DE PROGRAMACION	36
2.5. FORMAS DE ALMACENAR LOS VALORES(BITSTREAM) EN LA MEMORIA	37
2.6. FABRICANTES DE DISPOSITIVOS DE FPGA	39
2.7. EL ENTORNO DE PROGRAMACION ISE DESIGN SUITE DE XILINX	40
2.8. DISEÑO DE FLUJO NATURAL	40
2.8.1. Diseño de entrada	41
2.8.2. La síntesis	41
2.8.3. Verificación (Simulation)	42
2.8.4. Implementación	42
2.8.5. Configuración de dispositivos	43
2.8.6. La ventana del explorador de proyectos (Project Navigator)	43
2.8.7. La ventana de fuentes (Sources)	44
2.8.8. La ventana de procesos (Processes)	44
2.9. INTRODUCCION A MATLAB Y SIMULINK	44
2.9.1. Desarrollo de aplicaciones utilizando Matlab en FPGA	45
2.9.2. Componentes de Matlab	46
2.10. SIMULINK	47
2.10.1. El Entorno del Trabajo de Simulink	47
2.11. SISTEMA DE CONTROL DE LA CAJA DE HERRAMIENTAS (CONTROL SYSTEM TOOLBOX)	48
2.11.1. Características principales de la Caja De Herramientas	49
2.11.2. Usos de La Caja De Herramientas	49
2.12. Aplicaciones de la FPGA	50

CAPITULO 3

3. FILTRO DIGITAL FIR	
3.1. QUE ES UN FILTRO DIGITAL FIR	51
3.2. ESTRUCTURA DEL FILTRO FIR	52
3.3. POLOS Y CEROS DEL FILTRO FIR	53
3.4. DISEÑO DEL FILTRO FIR	53
3.5. CARACTERISTICAS DEL FILTRO FIR	54
3.6. TIPOS DE FILTRO	54
3.7. IMPLEMENTACION DE FILTROS DIGITALES TIPO FIR EN FPGA	55
3.7.1. Factores de implementación	57
3.7.2. Implementación en el FPGA	57
3.7.3. Identidades	58
3.7.4. La Arquitectura del Filtro FIR	59
3.7.5. Pipeline del Filtro FIR	60
4. CONCLUSIONES	62
5. RECOMENDACIONES	65
6. GLOSARIO	66
7. BIBLIOGRAFÍA	69
8. ANEXOS	70
A.7.1 PRACTICA UNO: USANDO SIMULINK	71
A.7.1.1 Crear un diseño simple	76
A.7.1.2 Analizando el efecto del periodo del muestreo	77
A.7.1.3 Cree un diseño simple usando los bloques de Simulink	79
A.7.1.4 Crear un subsistema	80
A.7.1.5 Conclusiones	81
A.7.2 PRACTICA DOS: CREACIÓN DE UN MAC 12 X 8 USANDO DEL SISTEMA GENERADOR DE XILINX	82
A.7.2.1 Procedimientos	83
A.7.2.2 Analizar la precisión y la muestra	88
A.7.2.3 Conectar una Mac utilizando Xilinx	94
A.7.2.4 Configurar y simular el 12 x 8	97
A.7.2.5 Las estimaciones de recursos	100
A.7.2.6 Generar el paso HDL	102
A.7.2.7 Conclusiones	104
A.7.3 PRACTICA TRES: ENRUTAMIENTO DE SEÑAL	106
A.7.3.1 Procedimiento	107
A.7.3.2 Diseño de relleno lógico	108
A.7.3.3 Conclusión	110
A.7.4 PRACTICA CUATRO: APLICACIÓN DE UN CONTROL	111
A.7.4.1 Procedimiento	112
A.7.4.2 Desarrollando un modelo MCODE	113
A.7.4.3 Conclusión	115
A.7.5 PRACTICA CINCO: DISEÑO DE UNA FIR MAC	117
A.7.5.1 Procedimiento	121
A.7.5.2 Añadir y parametrizar el paso de control logic 2	122
A.7.5.3 Agregue el puerto doble de RAM	124
A.7.5.4 Añadir relleno y un padding en el paso de los puertos	127
A.7.5.5 Completar el paso de diseño	129
A.7.5.6 Pruebas en el diseño con distintas fuentes de paso	134
A.7.5.7 Realización de hardware in the loop	137
A.7.5.8 Conclusión	140
A.7.6 PRACTICA SEIS: DISEÑO DE UN FILTRO FIR(RESPUESTA FINITA AL IMPULSO)	141
A.7.6.1 Procedimiento	143
A.7.6.2 Modelar y simular el filtro FIR	146
A.7.6.3 Realización de hardware in the loop	151
A.7.6.4 Conclusión	154

ÍNDICE DE GRAFICOS

CAPITULO 1

Fig.1 Grafica del Filtro Digital	3
Fig.2 Grafica de un Filtro Activo Paso Alto. Un amplificador operacional (elemento activo) resaltado en color rojo	7
Fig. 3 Grafica de un Filtro Pasivo Analógico de primer orden con circuito RC	9
Fig. 4 Desfase entre señal de entrada y salida	10
Fig. 5 Grafica de la Respuesta frecuencial de un filtro pasa banda	11
Fig. 6 Grafica de Filtros de Butterworth de varios órdenes	15
Fig. 7 Grafica de la Respuesta de un filtro de Cauer	17
Fig. 8 Grafica de la Estructura del F. IIR	23
Fig. 9 Grafica del aspecto del filtro en el centro. En la parte superior se muestra la señal que se quiere filtrar y en la parte inferior la señal filtrada	24
Fig. 10 Grafica del Sistema basado en un DSP	27

CAPITULO 2

Fig. 1 Grafica de xilinx spartan 3	31
Fig. 2 Grafica de los elementos de Xilinx de la FPGA	32
Fig. 3 Descripción de las distintas etapas en el flujo de diseño de un sistema digital	41
Fig. 4 ISE Project Window Navigator (clip de pantalla de Xilinx (TM) ISE)	43
Fig. 5 Gráfica de un sistema de control usando Toolbox	48

CAPITULO 3

Fig. 1 Grafica de la estructura del Filtro Fir	52
Fig. 2 Gráfica de ruido blanco	56
Fig. 3 Kit de desarrollo XILINX SPARTAN 3E	58
Fig. 4 Entidad de un filtro Fir en FPGA XILINX	59
Fig. 5 Esquema RTL	60

INTRODUCCIÓN

Es cada vez mayor el uso que la industria da a las FPGAs, un tipo particular de circuito integrado de gran versatilidad y de relativo bajo coste. Mientras que hace unos pocos años, el desarrollo de aplicaciones digitales integradas concretas requería el empleo, en muchos casos, de circuitos integrados de aplicación específica (ASIC), en la actualidad, el desarrollo alcanzado por las FPGAs ha permitido su empleo cada vez más intensivo.

En campos tan diversos como el procesamiento digital de la señal o el power line la FPGA está mostrando su versatilidad y el bajo coste asociado al desarrollo de aplicaciones en ellas. Mientras que un desarrollo de un ASIC requería grandes esfuerzos para la implementación correcta de aplicaciones, las FPGAs poseen librerías completas y gran facilidad para la programación de las mismas.

Debida a esa mayor importancia que adquiere en el mercado y la industria este tipo de integrado, nos ha parecido útil el mostrar una aplicación concreta de las mismas en la actualidad. Esta aplicación, el filtrado digital de la señal, nos muestra cómo ha evolucionado la tecnología para posibilitar estas nuevas aplicaciones.

También FPGAs Xilinx, ya que fue el primero en usar las tablas de búsqueda para construir lógica de usuario. Estas tablas de verdad lógicas son incrustadas en bloques lógicos configurables (configurable logic blocks o CLBs) que también incluyen una pareja de flips flops tipo D y circuitos para control y activación del reloj. La base literal del CLB es un SRAM (RAM estática) que almacena el patrón

de bits que define las funciones lógicas del CLB y los caminos ligados a ellos. Los patrones de bits SRAM son cargados de una forma variada de modos de configuración, y, como con cualquier RAM, puede ser recargado o escrito un número de veces ilimitado.

El diseño puede ser corregido o modificado a través de configuraciones programadas con ningún cambio en el hardware. Los datos de configuración pueden ser cargados desde un ordenador servidor o pueden ser transferidos desde una memoria PROM local de forma automática con el encendido del dispositivo.

Como estos productos FPGA ya están maduros, una librería de macros diseñados de forma óptima y probados han sido desarrollados para facilitar las tareas del diseñador.

De esa forma podemos encontrar contadores, registros de desplazamiento, sumadores paralelo y acumuladores, memorias RAM, FIFOs, etc, que pueden ser compiladas o escaladas a las dimensiones deseadas para satisfacer los requerimientos.

CAPITULO I

FILTRO DIGITAL

1.1.0 INTRODUCCIÓN AL DISEÑO DE FILTROS DIGITALES

Un filtro digital es un sistema que, dependiendo de las variaciones de las señales de entrada en el tiempo y amplitud (Fig. 1), se realiza un procesamiento matemático sobre dicha señal; generalmente mediante el uso de la Transformada rápida de Fourier; obteniéndose en la salida el resultado del procesamiento matemático o la señal de salida.

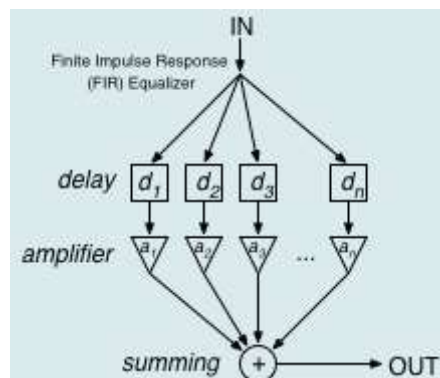


Fig.1 Grafica del Filtro Digital

Los filtros digitales tienen como entrada una señal analógica o digital y en su salida tienen otra señal analógica o digital, pudiendo haber cambiado en amplitud, frecuencia o fase dependiendo de las características del filtro digital.

El filtrado digital es parte del procesamiento de señal digital. Se le da la denominación de digital más por su funcionamiento interno que por su dependencia del tipo de

señal a filtrar, así podríamos llamar filtro digital tanto a un filtro que realiza el procesado de señales digitales como a otro que lo haga de señales analógicas.

Comúnmente se usa para atenuar o amplificar algunas frecuencias, por ejemplo se puede implementar un sistema para controlar los tonos graves y agudos del audio del estéreo del auto.

La gran ventaja de los filtros digitales sobre los analógicos es que presentan una gran estabilidad de funcionamiento en el tiempo.

El filtrado digital consiste en la realización interna de un procesado de datos de entrada.

En general el proceso de filtrado consiste en el muestreo digital de la señal de entrada, el procesamiento considerando el valor actual de entrada y considerando las entradas anteriores.

El último paso es la reconstrucción de la señal de salida.

1.2.0 MECANISMO DEL PROCESAMIENTO DEL FILTRO DIGITAL

En general la mecánica del procesamiento es:

- Tomar las muestras actuales y algunas muestras anteriores (que previamente habían sido almacenadas) para multiplicadas por unos coeficientes definidos.

- También se podría tomar valores de la salida en instantes pasados y multiplicarlos por otros coeficientes.
- Finalmente todos los resultados de todas estas multiplicaciones son sumados, dando una salida para el instante actual.

El procesamiento interno y la entrada del filtro serán digitales, por lo que puede ser necesario una conversión analógica-digital o digital-analógica para uso de filtros digitales con señales analógicas.

Un tema muy importante es considerar las limitaciones del filtro de entrada debido a Teorema de muestreo de Nyquist-Shannon que en pocas palabras; si quiero procesar hasta una frecuencia de 10KHz, debo muestrear a por lo menos 20 KHz. Los filtros digitales se usan frecuentemente para tratamiento digital de la imagen o para tratamiento del sonido digital. Otro ejemplo común de filtros digitales son los programas para retocar imágenes.

1.3.0 TIPOS DE FILTROS

Hay distintos tipos de clasificación de filtros.

De acuerdo a la ganancia:

- Filtros pasivos
- Filtros activos

De acuerdo a su respuesta en frecuencia:

- Filtro pasa bajo
- Filtro pasa alto
- Filtro pasa banda
- Filtro rechaza banda

De acuerdo al método de diseño:

- Filtro de Butterworth
- Filtro de Chebyshev I y Filtro de Chebyshev II
- Filtro de Cauer (elíptico)
- Filtro de Bessel

De acuerdo a su aplicación:

- Filtro de red.

Otros Filtros:

- Filtros piezoeléctricos.
- Filtro Analógicos
- Filtro Digital
- Filtro FIR
- Filtro IIR

1.3.1 De acuerdo a la ganancia:

Filtro Activo

Un filtro activo es un filtro electrónico analógico distinguido por el uso de uno o más componentes activos (que proporcionan una cierta forma de amplificación de energía), que lo diferencian de los filtros pasivos que solamente usan componentes pasivos. Típicamente este elemento activo puede ser un tubo de vacío, un transistor o un amplificador operacional.

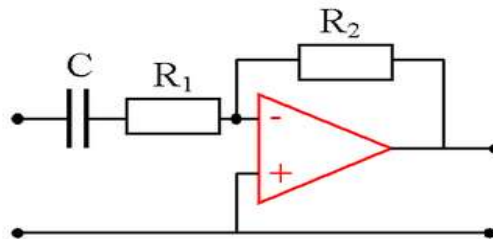


Fig. 2 Grafica de un Filtro Activo Paso Alto.
Un amplificador operacional (elemento activo) resaltado en color rojo.

Un filtro activo puede presentar ganancia en toda o parte de la señal de salida respecto a la señal de entrada. En su implementación se combinan elementos activos y pasivos, siendo frecuente el uso de amplificadores operacionales, que permite obtener resonancia y un elevado factor Q sin el empleo de bobinas.

Se pueden implementar, entre otros, filtros pasa bajo, pasa alto, pasa banda.

Configuraciones de circuitos de filtro activo incluyen:

- Filtro de Sallen-Key
- Filtro de estado variable

Filtro Pasivo

El filtro pasivo es un filtro electrónico formado únicamente por elementos pasivos, es decir, resistencias, condensadores y bobinas.

1.3.2 De De acuerdo a su respuesta en frecuencia:

Filtro Pasa Bajo

Un filtro pasa bajo corresponde a un filtro caracterizado por permitir el paso de las frecuencias más bajas y atenuar las frecuencias más altas. El filtro requiere de dos terminales de entrada y dos de salida, de una caja negra, también denominada cuadripolo o bipuerto, así todas las frecuencias se pueden presentar a la entrada, pero a la salida solo estarán presentes las que permita pasar el filtro. De la teoría se obtiene que los filtros están caracterizados por sus funciones de transferencia, así cualquier configuración de elementos activos o pasivos que consigan cierta función de transferencia serán considerados un filtro de cierto tipo.

En particular la función de transferencia de un filtro pasa bajo de primer orden corresponde a $H(s) = k \frac{1}{1 + \frac{s}{\omega_c}}$, donde la constante k es sólo una ponderación correspondiente a la ganancia del filtro, y la real importancia reside en la forma de la función de transferencia $\frac{1}{1 + \frac{s}{\omega_c}}$, la cual determina el comportamiento del filtro. En la función de transferencia anterior ω_c corresponde a la frecuencia de corte propia del filtro, es decir la frecuencia a partir de la cual el se empieza a atenuar la señal de entrada.

De forma análoga al caso de primer orden, los filtros de pasa bajo de mayor orden también se caracterizan por su función de transferencia, por ejemplo la de un filtro paso bajo de segundo orden corresponde a $H(s) = K \frac{\omega_0^2}{s^2 + 2\xi\omega_0 s + \omega_0^2}$, donde ω_0 es la frecuencia natural del filtro y ξ es el factor de amortiguamiento de este.

Filtro Pasa Alto

Un filtro pasa alto (HPF) es un tipo de filtro electrónico en cuya respuesta en frecuencia se atenúan las componentes de baja frecuencia pero no las de alta frecuencia, éstas incluso pueden amplificarse en los filtros activos. La alta o baja frecuencia es un término relativo que dependerá del diseño y de la aplicación.

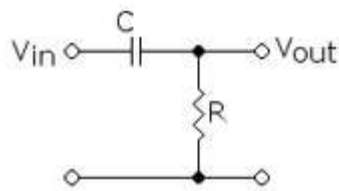


Fig. 3 Grafica de un Filtro Pasivo Analógico de primer orden con circuito RC

Implementación

Si se estudia este circuito (con componentes ideales) para frecuencias muy bajas, en continua por ejemplo, se tiene que el condensador se comporta como un circuito abierto, por lo que no dejará pasar la corriente a la resistencia, y su diferencia de tensión será cero. Para una frecuencia muy alta, idealmente infinita, el condensador se comportará como un cortocircuito, es decir, como si no

estuviera, por lo que la caída de tensión de la resistencia será la misma tensión de entrada, lo que significa que dejaría pasar toda la señal.

Por otra parte, el desfase entre la señal de entrada y la de salida si que varía, como puede verse en la Figura # 4.

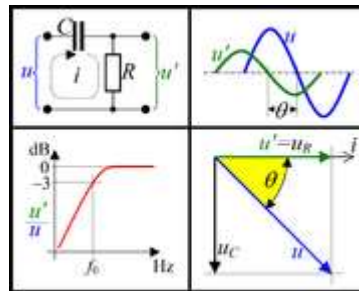


Fig. 4: Desfase entre señal de entrada y salida.

El filtro pasa alto más simple es un circuito RC en serie en el cual la salida es la caída de tensión en la resistencia.

El producto de resistencia por condensador ($R \times C$) es la constante de tiempo, cuyo recíproco es la frecuencia de corte, es decir, donde el módulo de la respuesta en frecuencia baja 3dB respecto a la zona pasante: $\theta = \tan^{-1} \frac{f_c}{f}$

Donde f_c es la frecuencia de corte en hercios, R es la resistencia del tweeter o parlante en ohmios y C es la capacidad en faradios. El desfase depende de la frecuencia f de la señal y sería: $f_c = \frac{1}{2\pi RC}$

Aplicaciones

Una posible aplicación de este tipo de filtro sería la de hacer que las altas frecuencias de una señal de audio fuesen a un altavoz para sonidos agudos mientras que un filtro pasa bajo haría lo propio con los graves.

Otra aplicación sería la de eliminar los ruidos que provienen de la red eléctrica (50 o 60Hz) en un circuito cuyas señales fueran más altas.

Filtro Pasa Banda

Un filtro pasa banda es un tipo de filtro electrónico que deja pasar un determinado rango de frecuencias de una señal y atenúa el paso del resto.



Fig.5 Grafica de la Respuesta frecuencial de un filtro pasa banda

Implementación

Un circuito simple de este tipo de filtros es un circuito RLC (resistencia, bobina y condensador) en el que se deja pasar la frecuencia de resonancia, que sería la frecuencia central (f_c) y las componentes frecuenciales próximas a ésta, en el diagrama hasta f_1 y f_2 . No obstante, bastaría con una simple red resonante LC.

Otra forma de construir un filtro pasa banda puede ser usar un filtro pasa bajo en serie con un filtro pasa alto entre los que hay un rango de frecuencias que ambos dejan pasar. Para ello, es importante tener en cuenta que la frecuencia de corte del pasa bajo sea mayor que la del pasa alto, a fin de que la respuesta global sea pasa banda (esto es, que haya solapamiento entre ambas respuestas en frecuencia).

Un filtro ideal sería el que tiene unas bandas pasante y de corte totalmente planas y unas zonas de transición entre ambas nulas, pero en la práctica esto nunca se consigue, siendo normalmente más parecido al ideal cuando mayor sea el orden del filtro, para medir cuanto de "bueno" es un filtro se puede emplear el denominado factor Q. En filtros de órdenes altos suele aparecer un rizado en las zonas de transición conocido como efecto Gibbs.

Un filtro pasa banda más avanzado sería los de frecuencia móvil, en los que se pueden variar algunos parámetros frecuenciales, un ejemplo es el circuito anterior RLC en el que se sustituye el condensador por un diodo varicap o varactor, que actúa como condensador variable y, por lo tanto, puede variar su frecuencia central.

Aplicaciones

Estos filtros tienen aplicación en ecualizadores de audio, haciendo que unas frecuencias se amplifiquen más que otras. Otra aplicación es la de eliminar ruidos que aparecen junto a una señal, siempre que la frecuencia de ésta sea fija o conocida.

Fuera de la electrónica y del procesado de señal, un ejemplo puede ser dentro del campo de las ciencias atmosféricas, donde son usados para manejar los datos dentro de un rango de 3 a 10 días.

De acuerdo con su orden:

- Primer orden
- Segundo orden

De acuerdo con el tipo de respuesta ante entrada unitaria:

- FIR (Finite Impulse Response)
- IIR (Infinite Impulse Response)
- TIIR (Truncated Infinite Impulse Response)

Filtro rechaza banda

El filtro suprime banda, filtro elimina banda, filtro notch, filtro trampa o filtro de rechazo de banda es un filtro electrónico que no permite el paso de señales cuyas frecuencias se encuentran comprendidas entre las frecuencias de corte superior e inferior.

Pueden implementarse de diversas formas. Una de ellas consistirá en dos filtros, uno paso bajo cuya frecuencia de corte sea la inferior del filtro elimina banda y otro paso alto cuya frecuencia de corte sea la superior del filtro elimina banda. Como ambos son sistemas lineales e invariantes, la respuesta en frecuencia de un filtro banda eliminada se puede obtener como la suma de la respuesta paso bajo y

la respuesta paso alto (hay que tener en cuenta que ambas respuestas no deben estar solapadas para que el filtro elimine la banda que interese suprimir), ello se implementará mediante un sumador analógico, hecho habitualmente con un amplificador operacional.

Otra forma más sencilla, si bien presenta una respuesta en frecuencia menos selectiva, sería la de colocar lo que se conoce como "circuito trampa". En efecto, si unimos las dos bornas (la considerada activo y la considerada masa) con un dipolo resonante LC serie o paralelo, la respuesta global sería la de un filtro elimina banda (el mínimo de la respuesta estaría en la frecuencia de resonancia del dipolo resonante). En este enlace se puede encontrar un ejemplo de filtro elimina banda (realmente se muestran los cuatro tipos de filtros, el filtro notch es el último de todos ellos) construido únicamente con componentes pasivos.

1.3.3 Atendiendo el método del diseño:

Filtro de Butterworth

El filtro de Butterworth más básico es el típico filtro paso bajo de primer orden, el cual puede ser modificado a un filtro pasa alto o añadir en serie otros formando un filtro pasa banda o elimina banda y filtros de mayores órdenes.

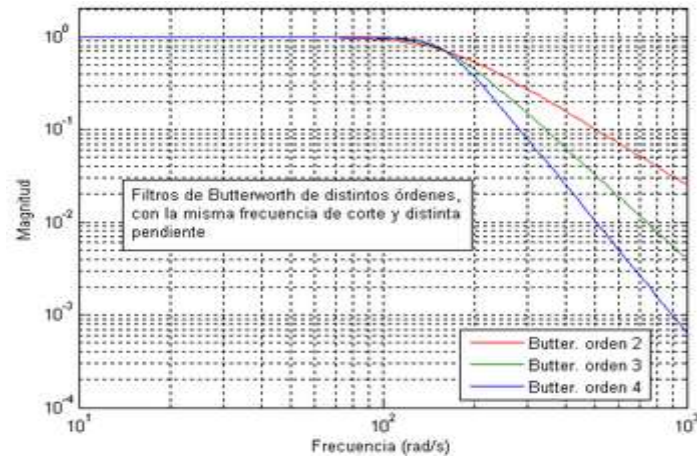


Fig. 6 Grafica de Filtros de Butterworth de varios órdenes

Según lo mencionado antes, la respuesta en frecuencia del filtro es máximamente plana (con las mínimas ondulaciones) en la banda pasante. Visto en un diagrama de Bode con escala logarítmica, la respuesta decae linealmente desde la frecuencia de corte hacia menos infinito. Para un filtro de primer orden son -20 dB por década (aprox. -6dB por octava).

El filtro de Butterworth es el único filtro que mantiene su forma para órdenes mayores (sólo con una caída de más pendiente a partir de la frecuencia de corte).

Este tipo de filtros necesita un mayor orden para los mismos requerimientos en comparación con otros, como los de Chebyshev o el elíptico.

Diseño

Si llamamos H a la respuesta en frecuencia, se debe cumplir que las $2N-1$ primeras derivadas $|H(\Omega)|^2$ de sean cero para $\Omega = 0$ y $\Omega = \infty$. Únicamente posee polos y la función de transferencia es: $|H(\Omega)|^2 = \frac{1}{1 + (\Omega/\Omega_c)^{2N}}$

donde N es el orden del filtro, Ω_c es la frecuencia de corte (en la que la respuesta cae 3 dB por debajo de la banda pasante) y Ω es la frecuencia analógica compleja ($\Omega = j\omega$).

El diseño es independiente de la implementación, que puede ser por ejemplo mediante células de Sallen-Key o Rauch, componentes discretos, etc.

Filtro de Chebyshev

En los filtros de Chebyshev lo que ocurre es que consiguen una caída de la respuesta en frecuencia más pronunciada en frecuencias bajas debido a que permiten más rizado que otros filtros en alguna de sus bandas. Se conocen dos tipos de filtros de Chebyshev los cuales son:

Filtro de Chebyshev de tipo I

Son filtros que únicamente tienen polos, presentan un rizado constante en la banda pasante y presentan una caída monótona en la banda no pasante. La respuesta en frecuencia $|H(\Omega)|^2 = \frac{1}{1 + \epsilon^2 T_N^2\left(\frac{\Omega}{\Omega_c}\right)}$ es: para $0 \leq \epsilon \leq 1$

donde N es el orden del filtro, Ω_c es la frecuencia de corte, Ω es la frecuencia analógica compleja ($\Omega = j\omega$) y $T_N(x)$ es el polinomio de Chebyshev de orden N, que se define $T_{N+1} = 2 \cdot x \cdot T_N(x) - T_{N-1}(x)$ como: con $T_0(x) = 1$ y $T_1(x) = x$

En estos filtros la frecuencia de corte no depende de N y el módulo de su respuesta en frecuencia oscila (rizado) entre 1 y $\frac{1}{\sqrt{1+\epsilon^2}}$.

Filtro de Chebyshev de tipo II

Estos filtros a diferencia de los Chebyshev I presentan ceros y polos, su rizado es constante en la banda no pasante y además presentan una caída monótonica en la banda pasante. Su respuesta en frecuencia $|H(\Omega)|^2 = \frac{1}{1 + \epsilon^2 \cdot \frac{T_n^2(\Omega_s/\Omega_c)}{T_n^2(\Omega_s/\Omega)}}$ es: para $0 \leq \epsilon \leq 1$. En un diagrama de circunferencia unidad, los polos estarían en una elipse y los ceros sobre el eje imaginario.

Filtro de Cauer

Un filtro elíptico o filtro de Cauer es un tipo de filtro eléctrico.

Están diseñados de manera que consiguen estrechar la zona de transición entre bandas y, además, acotando el rizado en esas bandas. La diferencia con el filtro de Chebyshev es que este sólo lo hace en una de las bandas.

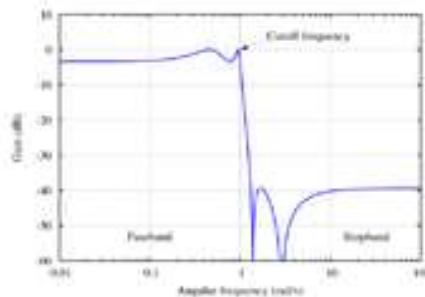


Fig.7 Grafica de la Respuesta de un filtro de Cauer.

Estos filtros suelen ser más eficientes debido a que al minimizar la zona de transición, ante unas mismas restricciones consiguen un menor orden. Por el contrario son los que presentan una fase menos lineal.

Diseño

La respuesta en frecuencia de un filtro pasa bajo elíptico es:

$|H(\Omega)|^2 = \frac{1}{1 + \epsilon^2 \cdot R_N(\Omega/\Omega_c)}$, para $0 \leq \epsilon \leq 1$, donde N es el orden del filtro, Ω_c es la frecuencia de corte, Ω es la frecuencia analógica compleja ($\Omega = j\omega$) y $R_N(x)$ es la función jacobiana elíptica de orden N, normalmente de primera clase:

$$R_N(x) = \int_0^{2\pi} \frac{1}{\sqrt{1 - x^2 \cdot \sin^2 \theta}} d\theta$$

Filtro de Bessel

Son filtros que únicamente tienen polos. Están diseñados para tener una fase lineal en las bandas pasantes, por lo que no distorsionan las señales; por el contrario tienen una mayor zona de transición entre las bandas pasantes y no pasantes.

Cuando estos filtros se transforman a digital pierden su propiedad de fase lineal.

Su respuesta en frecuencia es: $H(s) = \frac{1}{\sum_{k=0}^N a_k \cdot s^k}$, donde N es el orden del filtro y el denominador es un polinomio de Bessel, cuyos coeficientes son:

$$a_k = \frac{(2N - k)!}{2^{N-k} \cdot k! \cdot (N - k)!}, \text{ con } k = 0, 1, 2, \dots, N.$$

1.3.4 De acuerdo a su aplicación:

Filtro de red

Este tipo de circuito impide la entrada de ruido externo, además impide que el sistema contamine la red, de tal forma que se pueden utilizar fuentes analógicas y digitales o fuentes PWM que afecten negativamente el resto del equipo. También es posible corregir el factor de potencia ya que el circuito reduce significativamente los picos de corriente generados por el condensador al cargarse. El circuito consiste básicamente en un filtro paso bajo en donde la primera bobina elimina ruido en general (frecuencias altas), junto con los condensadores. El transformador elimina el ruido sobrante, que los condensadores no eliminan. Al transformador se le denomina choque de modo común. Son los utilizados para garantizar la calidad de la señal de alimentación, éstos tienen como objetivo eliminar ruidos tanto en modo común como en modo diferencial.

1.3.5 Otros tipos:

Filtros Piezoeléctricos

Este filtro aprovecha las propiedades resonantes de determinados materiales como el cuarzo. Este cristal de cuarzo se utiliza como componente de control de la frecuencia de circuitos osciladores convirtiendo las vibraciones mecánicas en voltajes eléctricos a una frecuencia específica. Esto ocurre debido al efecto piezoeléctrico. En un material piezoeléctrico, al aplicar una presión mecánica sobre

un eje, dará como consecuencia la creación de una carga eléctrica. En algunos materiales, se encuentra que aplicando un campo eléctrico según un eje, produce una deformación mecánica según otro eje ubicado a un ángulo recto respecto al primero. Por las propiedades mecánicas, eléctricas, y químicas, el cuarzo es el material más apropiado para fabricar dispositivos con frecuencia bien controlada. También existen filtros como el de ferrita que existe en muchos cables. Es normal encontrárselos en las pantallas del computador. Aquí se tiene la propiedad de presentar distintas impedancias a alta y baja frecuencia.

Filtro Analógico

Cualquier filtro, tiene una entrada y una salida, así que si hablamos de un filtro pasabajo, lo podemos ver, como una caja negra con dos terminales de entrada y dos de salida. Si un terminal de entrada es común a la salida tendremos un sistema desbalanceado (Unbalance, en inglés), así las cosas, si llamamos e_1 y e_2 a los terminales de entrada y s_1 y s_2 a los de salida, un filtro pasabajo sencillo, sería, colocar una resistencia entre e_1 y s_1 y un condensador, entre s_1 y s_2 , uniendo e_2 con s_2 , tenemos un filtro pasa bajo desbalanceado. Ahora veamos como trabaja: las diferentes frecuencias ingresan por e_1 - e_2 y salen por s_1 - s_2 , las altas frecuencias verán en el condensador una baja impedancia (cortocircuito) mientras que las bajas frecuencias seguirán de largo por las salidas s_1 - s_2 hacia el circuito siguiente, cumpliendo con la función de dejar pasar las bajas frecuencias y atenuar las altas. Esto a grandes rasgos. También como se describe abajo, se puede usar una bobina, entre e_1 y s_1 y los terminales e_2 y s_2 se unen, teniendo así un filtro

pasa bajo desbalanceado, el cual se rige por $X_L = \omega L$, donde X_L es la reactancia inductiva y ω la frecuencia angular y L la inductancia, como se ve abajo del escrito.

El más sencillo está armado en una resistencia y un condensador (o bobina). Pero podría ser mejor. Un filtro analógico elemental compuesto por un capacitor se denomina, "Filtro pasa altos" (debido a que la Reactancia Capacitiva $X_C = 1/\omega C$). Mientras que el compuesto por una inductancia (bobina, o choque) es un "filtro pasa bajos" (debido a que la Reactancia Inductiva $X_L = \omega L$).

Filtro Digital

La ecuación de un filtro paso bajo digital de primer orden es:

$y[n] = y[n-1] + \frac{x[n] - y[n-1]}{A}$, Donde A ha de ser mayor que uno. También es llamado filtro promediador, debido a que promedia las muestras de la entrada y por lo tanto suprime variaciones rápidas, característica que le otorga el carácter de paso bajo. Su transformada Z es: $H(Z) = \frac{1}{A}(1 + Z^{-1})$

Ancho De Banda

Un filtro pasa bandas ideal posee dos espectros uno ubicado en ω_0 y otro en $-\omega_0$, siendo ω_0 la frecuencia central del filtro, si el mismo posee un ancho de banda b los espectros sería: $-\omega_0 - b/2$ y $-\omega_0 + b/2$; $\omega_0 - b/2$ y $\omega_0 + b/2$.

Filtro IIR

IIR es una sigla en inglés para Infinite Impulse Response o Respuesta infinita al impulso. Se trata de un tipo de filtros digitales en el que, como su nombre indica, si la entrada es una señal impulso, la salida tendrá un número infinito de términos no nulos, es decir, nunca vuelve al reposo.

Expresión matemática de los filtros IIR

La salida de los filtros IIR depende de las entradas actuales y pasadas, y además de las salidas en instantes anteriores. Esto se consigue mediante el uso de realimentación de la salida. $y_n = b_0x_n + b_1x_{n-1} + \dots + b_Nx_{n-N} - a_1y_{n-1} - a_2y_{n-2} - \dots - a_My_{n-M}$.

Donde los a y b son los coeficientes del filtro. El orden es el máximo entre los valores de M y N. Aplicando la transformada Z a la expresión anterior:

$$H(z) = \frac{\sum_{k=0}^M b_k z^{-k}}{1 - \sum_{k=1}^N a_k z^{-k}}$$

Estructura

Hay numerosas formas de implementar los filtros IIR. La estructura afecta a las características finales que presentará el filtro como la estabilidad. Otros parámetros a tener en cuenta a la hora de elegir una estructura es el gasto computacional que presenta.

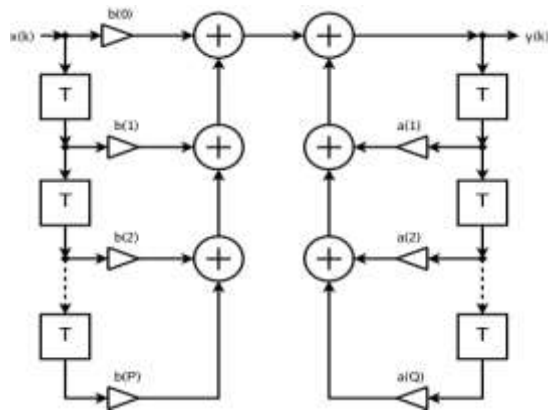


Fig. 8 Grafica de la Estructura del F. IIR

Polos y ceros

Este tipo de filtros presenta polos y ceros que determinan la estabilidad y la causalidad del sistema. Cuando todos los ceros están en el interior de la circunferencia unidad se dice que es fase mínima. Si todos están en el exterior es fase máxima. Si algún polo está fuera de la circunferencia unidad el sistema es inestable.

Diseño de filtros IIR

Las formas habituales de diseñar este tipo de filtros son:

- Indirecta (a partir de prototipos analógicos)
- Impulso invariante
- Aproximación de derivadas
- Transformación bilineal

- Directa
- Aproximación de Padé
- Aproximación de mínimos cuadrados

Ejemplo del diseño de un filtro

En primer lugar se parte de las especificaciones y, basándose en éstas, se elige el tipo de filtro. En este ejemplo se parte de un filtro digital que anule las frecuencias menores a 5Hz y la de 50Hz y que no altere al resto, la frecuencia de muestreo será 1000Hz, además se quiere fase lineal. Con estas especificaciones se elige un filtro FIR. El diseño se puede hacer manualmente o con la ayuda de un ordenador. En este ejemplo el método de diseño será el de Remez. En Matlab se obtienen los coeficientes que definen el filtro, que en la ecuación anterior se llaman a y b (el numerador es la variable b y el denominador solo tiene un término que es 1, como corresponde a un filtro FIR):
 $[n, fo, mo, w] = \text{remezord}([0.5, 45, 50, 55], [0, 1, 0, 1], [0.01, 0.1, 0.01, 0.1], 1000);$
 $b = \text{remez}(n, fo, mo, w, w);$

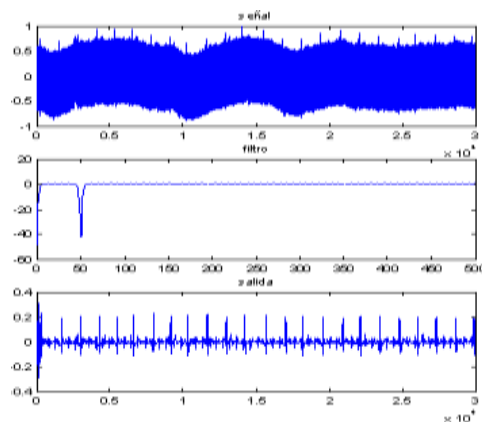


Fig. 9 Grafica del aspecto del filtro en el centro. En la parte superior se muestra la señal que se quiere filtrar y en la parte inferior la señal filtrada (se trata de un electrocardiograma).

El siguiente paso es seleccionar la forma de implementarlo, es decir su estructura. Luego se elige el hardware sobre el que funcionará. Normalmente un Procesador digital de señal o una FPGA, aunque también puede ser un programa de ordenador. Finalmente se usan los coeficientes obtenidos y la estructura elegida para crear el programa.

Características

Las principales diferencias respecto a los filtros FIR es que los IIR pueden cumplir las mismas exigencias que los anteriores pero con menos orden de filtro. Esto es importante a la hora de implementar el filtro, pues presenta una menor carga computacional.

Este tipo de filtros puede ser inestable, aún cuando se diseñen para ser estables.

En principio no pueden diseñarse para tener fase lineal pero se pueden aplicar algunas técnicas como el filtrado bidireccional para lograrlo.

De acuerdo con la estructura con que se implementa:

Laticce

Varios en cascada

Varios en paralelo

Expresión general de un filtro

Hay muchas formas de representar un filtro. Por ejemplo, en función de ω (frecuencia digital), en función de z y en función de n (número de muestra). Todas son equivalentes, pero a la hora de trabajar a veces conviene más una u otra. Como regla general se suele dejar el término $a_0=1$.

Si se expresa en función de z y en forma de fracción: $H(z) = \frac{\sum_{k=0}^M b_k \cdot z^{-k}}{\sum_{k=0}^N a_k \cdot z^{-k}}$. Y en

dominio de n : $y(n) = \sum_{k=0}^N b_k \cdot x(n-k) - \sum_{k=1}^M a_k \cdot y(n-k)$

Los coeficientes son los a y b y son los que definen el filtro, por lo tanto el diseño consiste en calcularlos.

1.4.0 IMPORTANCIA EN PROCESAMIENTO DIGITAL DE SEÑALES

1.4.1 Que es DSP?

DSP es el acrónimo de *Digital Signal Processor*, que significa Procesador Digital de Señal. Un DSP es un sistema basado en un procesador o microprocesador que posee un juego de instrucciones, un hardware y un software optimizados para aplicaciones que requieran operaciones numéricas a muy alta velocidad. Debido a esto es especialmente útil para el procesado y representación de señales analógicas en tiempo real: en un sistema que trabaje de esta forma (tiempo real) se reciben muestras (samples en inglés), normalmente provenientes de un conversor

analógico/digital (ADC). Se ha dicho que puede trabajar con señales analógicas, pero es un sistema digital, por lo tanto necesitará un conversor analógico/digital a su entrada y digital/analógico en la salida. Como todo sistema basado en procesador programable necesita una memoria donde almacenar los datos con los que trabajará y el programa que ejecuta. Si se tiene en cuenta que un DSP puede trabajar con varios datos en paralelo y un diseño e instrucciones específicas para el procesamiento digital, se puede dar una idea de su enorme potencia para este tipo de aplicaciones. Estas características constituyen la principal diferencia de un DSP y otros tipos de procesadores. Para adentrar en su funcionamiento se pondrá el ejemplo de un filtro: el DSP recibirá valores digitales o samples procedentes de la señal de entrada, calcula qué salida se obtendrá para esos valores con el filtro que se le ha programado y saca esa salida.

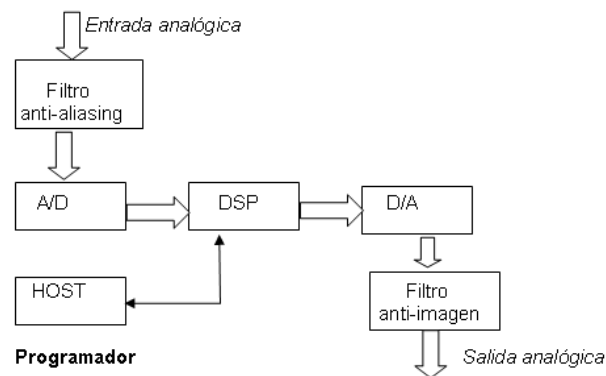


Fig. 10 Grafica del Sistema basado en un DSP

La señal entrante entra directamente en un filtro antialiasing para evitar frecuencias superiores a la de muestreo del conversor analógico-digital. Después se lleva a cabo el procesamiento digital en el módulo DSP, para después volverse a convertir en analógico y dar paso a la salida.

1.4.2 Aplicaciones

- Procesamiento digital de sonido
- Procesamiento digital de voz
- Procesamiento digital de imágenes
- Procesamiento digital de vídeo

Las aplicaciones más habituales en las que se emplean **DSP** son el procesado de audio y video; y cualquier otra aplicación que requiera el procesado en tiempo real. Con estas aplicaciones se puede eliminar el eco en las líneas de comunicaciones, lograr hacer más claras imágenes de órganos internos en los equipos de diagnóstico médico, cifrar conversaciones en teléfonos celulares para mantener privacidad, analizar datos sísmicos para encontrar nuevas reservas de petróleo, hace posible las comunicaciones wireless LAN, el reconocimiento de voz, los reproductores digitales de audio, los modems inalámbricos, las cámaras digitales, y una larga lista de elementos que pueden ser relacionados con el proceso de señales.

1.4.3 Importancia de las transformadas en el DSP

Uno de los beneficios principales del DSP es que las transformaciones de señales son más sencillas de realizar. Una de las más importantes transformadas es la Transformada de Fourier discreta (TFD). Esta transformada convierte la señal del dominio del tiempo al dominio de la frecuencia. La TDF permite un análisis más sencillo y eficaz sobre la frecuencia, sobre todo en aplicaciones de eliminación de

ruido y en otros tipos de filtrado (pasa bajos, filtros pasa altos, filtros pasa banda, filtros de rechazo de banda, etc.).

Otra de las transformadas importantes es la Transformada de Coseno Discreta la cual es similar a la anterior en cuanto a los cálculos requeridos para obtenerla, pero esta convierte a la señales en componentes del coseno trigonométrico. Esta transformada es una de las bases del algoritmo de compresión de imágenes JPEG.

1.4.4 Arquitectura

Un DSP está diseñado teniendo en cuenta las tareas más habituales del procesado digital: sumas, multiplicaciones y retrasos (almacenar en memoria).

Los DSP abandonan la arquitectura clásica de Von Neumann, en la que datos y programas están en la misma zona de memoria, y apuestan por la denominada Arquitectura Harvard. En una arquitectura Harvard existen bloques de memoria físicamente separados para datos y programas. Cada uno de estos bloques de memoria se direcciona mediante buses separados (tanto de direcciones como de datos), e incluso es posible que la memoria de datos tenga distinta anchura de palabra que la memoria de programa (como ocurre en ciertos microcontroladores).

Los elementos básicos que componen un **DSP** son:

- Conversores en las entradas y salidas
- Memoria de datos, memoria de programa y DMA.
- MACs: multiplicadores y acumuladores.

- ALU: Unidad aritmético-lógica.
- Registros.
- PLL: Bucles enganchados en fase.
- PWM: Módulos de control de ancho de pulso.

1.4.5 Arquitecturas estándar en DSP

Las arquitecturas de los computadores actuales están comúnmente clasificadas como RISC's (Reduced Instruction Set Computers) Conjunto de instrucciones de reducción de equipos y CISC's (Complex Instruction Set Computers) Complejo de instrucciones para equipos.

Estos últimos tienen un gran número de instrucciones sumamente poderosas, mientras que la arquitectura RISC posee pocas instrucciones y realiza movimientos de datos entre registros en un ciclo de máquina. Hoy en día los computadores RISC comienzan a reemplazar a los CISC's, porque se puede alcanzar un más alto rendimiento por medio del uso de un eficiente compilador como a través de la ejecución de instrucciones simples en forma ordenada.

DSP's estándares tienen mucho rasgos de una arquitectura tipo RISC, aunque son procesadores de propósitos específicos cuya arquitectura está especialmente diseñada para operar en ambientes de alta necesidad de cálculo. Un DSP estándar ejecuta varias operaciones en paralelo mientras que un RISC usa unidades funcionales altamente eficientes que pueden iniciar y completar una instrucción simple en uno o dos ciclos de reloj.

CAPITULO II

HERRAMIENTAS PARA EL FPGA Y SU ENTORNO DE DESARROLLO

Una FPGA (del inglés Field Programmable Gate Array) compuertas de arreglo o matrices programables, 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 (System-on-a-chip).



Fig.1 Grafica de xilinx spartan 3E

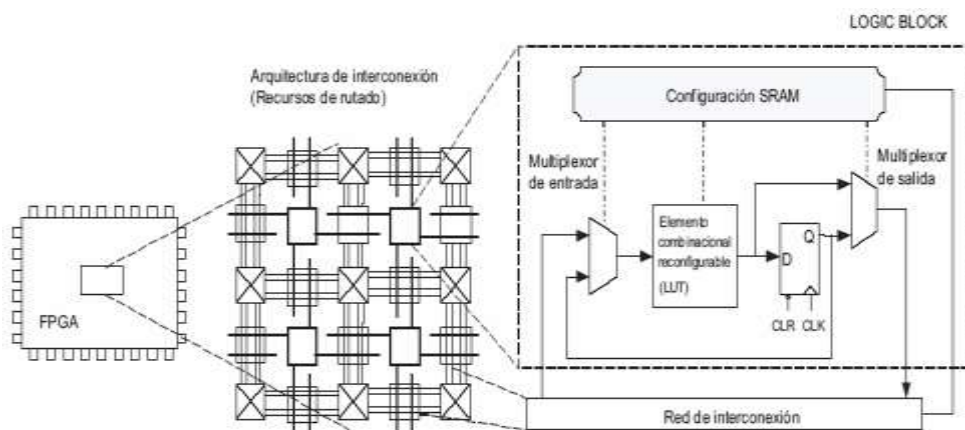
Las FPGAs se utilizan en aplicaciones similares a los ASICs sin embargo son más lentas, tienen un mayor consumo de potencia y no pueden abarcar sistemas tan complejos como ellos. A pesar de esto, las FPGAs tienen las ventajas de ser reprogramables (lo que añade una enorme flexibilidad al flujo de diseño), sus costes de desarrollo y adquisición son mucho menores para pequeñas cantidades de dispositivos y el tiempo de desarrollo es también menor.

Ciertos fabricantes cuentan con FPGAs que sólo se pueden programar una vez, por lo que sus ventajas e inconvenientes se encuentran a medio camino entre los ASICs y las FPGAs reprogramables.

Históricamente las FPGA surgen como una evolución de los conceptos desarrollados en las PLA (programmable logic array) Red lógica programable y los CPLD(Complex Programmable Logic Device) Complejo de Dispositivos Lógicos Programables.

2.1.0 ELEMENTOS QUE CONFORMAN LOS FPGAS DE XILIX.

Elementos reconfigurables: CLBs e IOBs. Elemento básico del CLB es la celda lógica. Cada CLB puede tener varias de estas celdas lógicas (CL). Los CLB de Virtex tienen cuatro CL distribuidas en dos slices.



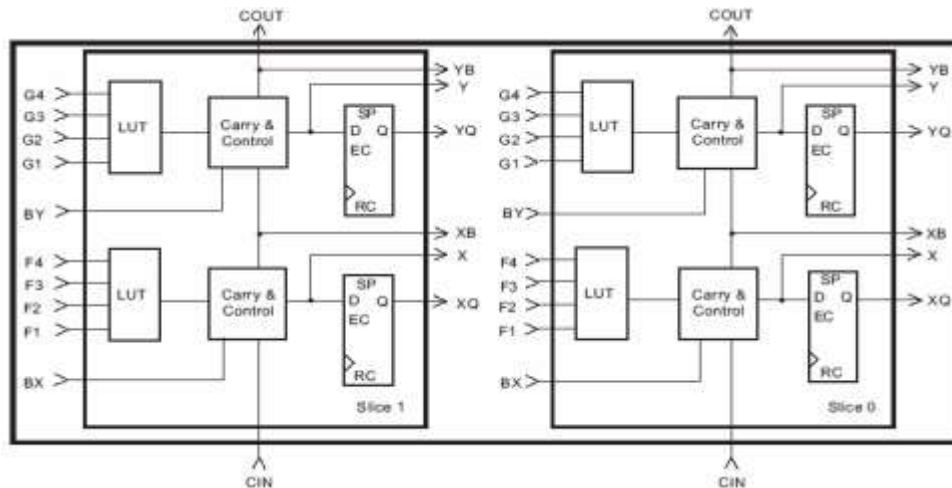


Fig. 2. Grafica de los elementos de xilinx de la FPGA.

A. Contenido de la celda lógica.

La celda lógica de una FPGA de Xilinx contiene:

1. **Generador de funciones (circuito programable LUT):** Puede funcionar como circuito combinacional de 4 entradas o como memoria RAM 16x1. Las memorias pueden ser de varios tamaño combinando varias LUTs. También puede funcionar como registro de corrimiento de 16 bits.
2. **Lógica de acarreo.**
3. **Elemento de memoria (para almacenar el resultado de la LUT).** Matriz de ruteo global. Rutas horizontales y verticales con switches situados en la intersección entre las rutas. Para interconectar diferentes CLBs internamente.

B. IOBs: Para interconectar los pines del FPGA y los CLBs.

- C. Rutas locales:** Para interconectar CLBs localmente y para conectar los CLBs con la matriz de ruteo global.
- D. Dispositivos para la gestión del reloj:** DDL en Spartan y DCMS en Virtex.
- E. Memoria RAM:** No emplean recurso del CLB. Los bloques de memoria se distribuyen en las columnas cubriendo toda la altura del dispositivo.

También hay recursos de ruteo para conectar estos bloques con los CLBs. El contenido se fija mediante la memoria SRAM de configuración incluyéndose en el bitstream inicial o bitstream parciales posteriores. Estos bloques pueden funcionar como memoria ROM.

- F. Buffers de 3 estados:** Para conectar los CLBs a las rutas horizontales globales. Se llaman Tbufs.

Todos estos elementos se configuran mediante valores almacenados en celdas de memoria SRAM. Cambiar la funcionalidad del dispositivo equivale a cambiar los valores de esta memoria.

2.2.0 CARACTERÍSTICAS DE LAS FPGA

Una jerarquía de interconexiones programables permite a los bloques lógicos de un FPGA ser interconectados según la necesidad del diseñador del sistema, algo parecido a un breadboard(es una placa de uso genérico reutilizable o semi permanente) programable. Estos bloques lógicos e interconexiones pueden ser

programados después del proceso de manufactura por el usuario/diseñador, así que el FPGA puede desempeñar cualquier función lógica necesaria.

Una tendencia reciente ha sido combinar los bloques lógicos e interconexiones de los FPGA con microprocesadores y periféricos relacionados para formar un «Sistema programable en un chip».

Ejemplo de tales tecnologías híbridas pueden ser encontradas en los dispositivos Virtex-II PRO y Virtex-6 de Xilinx, los cuales incluyen uno o más procesadores PowerPC embebidos junto con la lógica del FPGA.

Muchos FPGA modernos soportan la reconfiguración parcial del sistema, permitiendo que una parte del diseño sea reprogramada, mientras las demás partes siguen funcionando.

Este es el principio de la idea de la «computación reconfigurable», o los «sistemas reconfigurables».

2.3.0 PROGRAMACIÓN FPGA

La tarea del programador es definir la función lógica que realizará cada uno de los CLB, seleccionar el modo de trabajo de cada IOB e interconectarlos.

El diseñador cuenta con la ayuda de entornos de desarrollo especializados en el diseño de sistemas a implementarse en un FPGA. Un diseño puede ser capturado ya sea como esquemático, o haciendo uso de un lenguaje de programación

especial. Estos lenguajes de programación especiales son conocidos como HDL o Hardware Description Language (lenguajes de descripción de hardware). Los HDLs más utilizados son:

VHDL : Lenguaje para descripción y modelado de circuitos

Verilog: Es un lenguaje de descripción de hardware

ABEL: designación de bases

En un intento de reducir la complejidad y el tiempo de desarrollo en fases de prototipaje rápido, y para validar un diseño en HDL, existen varias propuestas y niveles de abstracción del diseño. Entre otras, National Instruments LabVIEW FPGA propone un acercamiento de programación gráfica de alto nivel.

2.4.0 TECNOLOGÍA DE LA MEMORIA DE PROGRAMACIÓN

En general existen 3 grandes tipos de dispositivos electrónicos: de memoria, procesadores y de lógica.

Los dispositivos de memoria almacenan información aleatoria (archivos, hojas de cálculo...)

Los procesadores ejecutan instrucciones de software para ejecutar una gran variedad de tareas (ejecutar un programa de proceso de datos o un videojuego)

Los dispositivos de lógica proveen funciones específicas (comunicación entre dispositivos y el resto de funciones que un sistema debe ejecutar). A su vez, existen 2 tipos de categorías de dispositivos lógicos: los fijos y los programables. Xilinx esta en el negocio de los dispositivos de lógica programable.

Las FPGAs también se pueden diferenciar por utilizar diferentes tecnologías de memoria:

Volátiles: Basadas en RAM. Su programación se pierde al quitar la alimentación. Requieren una memoria externa no volátil para configurarlas al arrancar (antes o durante el reset).

No Volátiles: Basadas en ROM. Hay de dos tipos, las reprogramables y las no reprogramables.

Reprogramables: Basadas en EPROM o flash. Éstas se pueden borrar y volver a reprogramar aunque con un límite de unos 10.000 ciclos.

No Reprogramables: Basadas en fusibles. Solo se pueden programar una vez, lo que las hace poco recomendables para trabajos en laboratorios.

2.5.0 FORMAS DE ALMACENAR LOS VALORES (BITSTREAM) EN LA MEMORIA.

Configuración serie: El bitstream se lee desde una memoria PROM serie exterior.

Dos variantes:

- **Serie Maestro (Master):** El FPGA genera el reloj de la carga.
- **Serie Esclavo (Slave):** El reloj lo genera otro dispositivo externo. Permite que la carga se realice desde un microprocesador externo.

Configuración en paralelo: Configuración a través de un puerto paralelo de 8 bits

- **SelectMap (seleccionar mapa).** La memoria de configuración esta dividida en frames.

A través del **SelectMap** se introducen frames concatenados en el bitstream. Aunque es un puerto de 8 bits, la configuración se realiza mediante paquetes de 32 bits.

El **SelectMap** dispone de varios registros internos. Registro de comandos, registro de control y registro de dirección. Los paquetes de configuración de 32 bits pueden incluir información para estos registros de forma que se controlan operaciones de lectura de la memoria de configuración o la reconfiguración parcial dinámica.

Configuración escaneo de límites (Boundary scan) (JTAG): La configuración se realiza mediante el protocolo serie JTAG.

La configuración es a través de la interfaz Boundary scan, que esta activa y accesible en todo momento. Es el sistema utilizado en la fase de desarrollo y depuración.

2.6.0 FABRICANTES DE DISPOSITIVOS FPGA

A principios de 2007, el mercado de los FPGA se ha colocado en un estado donde hay dos productores de FPGA de propósito general que están a la cabeza del mismo, y un conjunto de otros competidores quienes se diferencian por ofrecer dispositivos de capacidades únicas.

Xilinx es uno de los dos grandes líderes en la fabricación de FPGA.

Altera es el otro gran líder. Lattice Semiconductor lanzó al mercado dispositivos FPGA con tecnología de 90nm. En adición, Lattice es un proveedor líder en tecnología no volátil, FPGA basadas en tecnología Flash, con productos de 90nm y 130nm.

Actel tiene FPGAs basados en tecnología Flash reprogramable. También ofrece FPGAs que incluyen mezcladores de señales basados en Flash.

QuickLogic (lógica rápida) tiene productos basados en antifusibles (programables una sola vez).

Atmel es uno de los fabricantes cuyos productos son reconfigurables (el Xilinx XC62xx fue uno de estos, pero no están siendo fabricados actualmente). Ellos se enfocaron en proveer microcontroladores AVR con FPGAs, todo en el mismo encapsulado.

Achronix Semiconductor tienen en desarrollo FPGAs muy veloces. Planean sacar al mercado a comienzos de 2007 FPGAs con velocidades cercanas a los 2GHz.

MathStar, Inc. ofrecen FPGA que ellos llaman FPOA (Arreglo de objetos de matriz programable).

2.7.0 EL ENTORNO DE PROGRAMACION ISE DESIGN SUITE DE XILINX

El sistema de Xilinx ISE es un entorno de diseño integrado que consiste en que un conjunto de programas para crear, simular e implementar diseños digitales en una FPGA o CPLD.

Todas las herramientas de uso de una interfaz gráfica de usuario (GUI) permiten a todos los programas ser ejecutados en las barras de herramientas, menús o iconos.

Esta escritura se destina a empezar con las herramientas de ISE. Da una visión rápida de cómo crear un diseño, simulación y descargar en una FPGA

2.8.0 DISEÑO DE FLUJO GENERAL

Los siguientes pasos están involucrados en la realización de un sistema digital usando Xilinx FPGAs, como lo ilustra la siguiente figura:

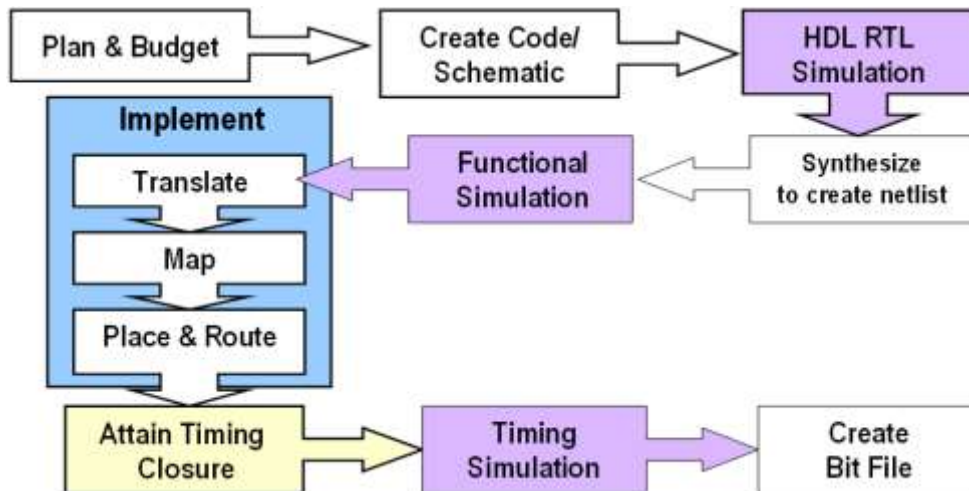


Fig.3 Descripción de las distintas etapas en el flujo de diseño de un sistema digital

2.8.1 Diseño de entrada

El primer paso es entrar a nuestro diseño. Esto puede hacerse mediante la creación de "Fuente" archivos. Los archivos de origen se pueden crear en diferentes formatos, tales como un esquema, o un lenguaje de descripción de hardware (HDL), tales como VHDL, Verilog o Abel. Un diseño del proyecto consistirá en un alto nivel de archivo de origen y de baja varios archivos de origen de nivel. Cualquiera de estos archivos puede ser un esquema o un archivo de HDL.

2.8.2 La síntesis

La etapa de síntesis crea los archivos de netlist de los archivos de origen diferentes. Los archivos de netlist pueden servir como entrada al módulo de aplicación.

2.8.3 Verificación

Este es un paso importante que debe hacerse en las distintas etapas del diseño. El simulador se utiliza para verificar la funcionalidad de un diseño (simulación funcional), el comportamiento y el tiempo (simulación de tiempo) de su circuito. Tiempo de simulación se ejecuta después de la aplicación de su circuito en la FPGA, ya que necesita conocer la ubicación real y de enrutamiento para determinar la velocidad exacta y el calendario del circuito.

2.8.4 Implementación

Después de generar el archivo de netlist (etapa de síntesis), la aplicación convertirá el diseño lógico en un archivo físico que se puede descargar en el dispositivo de destino (por ejemplo, Virtex FPGA).

Esto involucra a tres sub-etapas:

- Traducción de la netlist (lista de red),
- Cartografía
- Lugar de carreteras y planos.

2.8.5 Configuración de dispositivos

Esto se refiere a la programación real de la meta de FPGA mediante la descarga del archivo de programación de la FPGA de Xilinx.

2.8.6 La ventana navegador de proyectos (Project Navigator)

Los pasos anteriores se gestionan a través de una ventana central ISE Project Navigator, que se muestra a continuación.

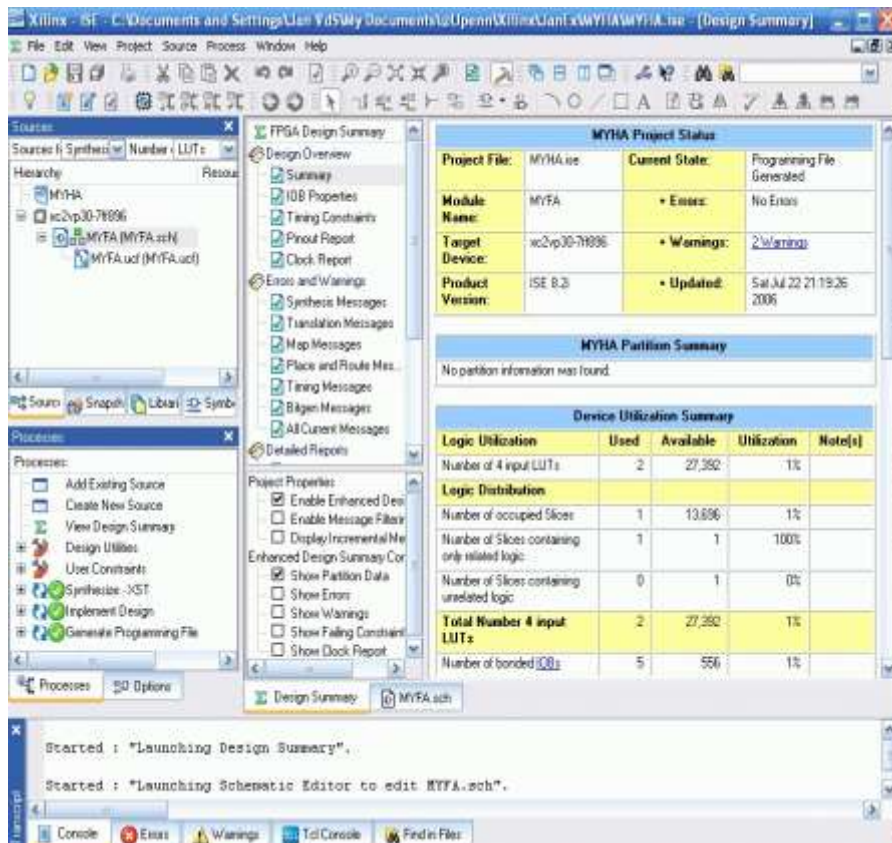


Fig 4. Ventana del ISE Project (clip de pantalla de Xilinx (TM) ISE).

2.8.7 La ventana de fuentes (Sources)

Esta ventana contiene los archivos de diseño de la fuente de un proyecto. Estos son los archivos de código fuente que haya creado o añadido al proyecto (véase más adelante). Una lista desplegable en la parte superior de las fuentes de la ventana le permite seleccionar los archivos de código fuente que están asociados con un aspecto particular diseño como la síntesis / o aplicación de simulación.

2.8.8 La ventana de procesos (Processes)

Las ventanas de los procesos de la lista de procesos disponibles (correspondientes a los procesos seleccionados en la ventana de procesos). Normalmente se selecciona un proceso particular que desea llevar a cabo en la fuente de archivo seleccionado. Esto puede incluir una simulación, ejecución, etc. Para ejecutar un proceso que puede hacer doble clic sobre el proceso. Cuando un proceso se ha ejecutado con éxito una marca roja-apagada_ icono. Cuando se ejecuta un proceso de alto nivel, el Navegador de proyecto se ejecutará automáticamente a todos los asociados a procesos de nivel inferior.

2.9.0 INTRODUCCIÓN A MATLAB Y SIMULINK.

Matlab es un programa de gran aceptación en ingeniería destinado realizar cálculos técnicos científicos y de propósito general. En el se integran operaciones

de cálculo, visualización y programación, donde la interacción con el usuario emplea una notación Matemática clásica.

Los usos y aplicaciones típicos de Matlab son:

- Matemáticas y cálculo.
- Desarrollo de algoritmos.
- Adquisición de datos.
- Modelado, simulación y prototipado.
- Análisis y procesado de datos.
- Gráficos científicos y de ingeniería.

2.9.1 Desarrollo de aplicaciones utilizando Matlab en FPGAS.

El tipo básico de variable con el que trabaja Matlab es una matriz que no requiere ser dimensionada previamente en la declaración.

Una de las características más interesantes consiste en que el algebra vectorial y matricial se expresa con la misma sintaxis que las operaciones aritméticas escalares. Por ejemplo, en lenguaje C, para realizar la suma de dos variables enteras o reales b y c , escribiremos:

$a = b+c$; Mientras que en Matlab, emplearemos la misma sentencia tanto si b y c son enteros, reales, vectores o matrices.

2.9.2 Componentes de MATLAB

Matlab consta de cinco partes fundamentales:

- 1. Entorno de desarrollo:** Se trata de un conjunto de utilidades que permiten el uso de funciones Matlab y ficheros en general. Muchas de estas utilidades son interfaces gráficas de usuario. Incluye el espacio de trabajo Matlab y la ventana de comandos.
- 2. La librería de funciones matemáticas Matlab.** Se trata de un amplio conjunto de algoritmos de cálculo, comprendiendo las funciones más elementales como la suma, senos y cosenos o la aritmética compleja, hasta funciones más sofisticadas como la inversión de matrices, el cálculo de autovalores, funciones de Bessel y transformadas rápidas de Fourier.
- 3. Gráficos.** Matlab dispone de un conjunto de utilidades destinadas a visualizar vectores y matrices en forma de gráficos. Existe una gran cantidad de posibilidades para ajustar el aspecto de los gráficos, destacando la visualización tridimensional con opciones de iluminación y sombreado, y la posibilidad de crear animaciones.
- 4. El interfaz de aplicación de Matlab (API).** Consiste en una librería que permite escribir programas ejecutables independientes en C y otros lenguajes, accediendo, mediante DLLs, a las utilidades de cálculo matricial de Matlab.

La gestión de complementos de Matlab se realiza mediante lo que se denominan toolboxes (paquetes de herramientas).

Un Toolbox de Matlab es un conjunto de funciones y algoritmos de cálculo especializados en un área de conocimiento: finanzas, tratamiento de señales, teoría de sistemas, etc.

2.10. SIMULINK.

Simulink es una aplicación que permite construir y simular modelos de sistemas Físicos y sistemas de control mediante diagramas de bloques. El comportamiento de dichos sistemas se define mediante funciones de transferencia, operaciones matemáticas, elementos de Matlab y señales predefinidas de todo tipo.

Simulink dispone de una serie de utilidades que facilitan la visualización, análisis y guardado de los resultados de simulación. Simulink se emplea profusamente en ingeniería de control.

2.10.1 El entorno de trabajo de Simulink

Simulink es una herramienta de gran utilidad para la simulación de sistemas dinámicos. Principalmente, se trata de un entorno de trabajo gráfico, en el que se especifican las partes de un sistema y su interconexión en forma de diagrama de bloques. De nuevo, se trata de una herramienta amplísima que además se complementa con numerosos elementos opcionales. Por tanto, nos limitaremos a dar unas pinceladas de los elementos más útiles en Regulación Automática.

Además de las capacidades de simulación de las que está dotado Simulink, conviene destacar que contiene cómodas utilidades de visualización y almacenamiento de resultados de simulación.

2.11.0 Sistema De Control De Caja De Herramientas (Control System Toolbox)

Diseño y análisis de sistemas de control



Fig.5 Gráfica de un sistema de control usando caja de herramientas

Control System Toolbox ofrece herramientas especializadas para analizar, diseñar y ajustar de forma sistemática sistemas de control lineales. Puede especificar un modelo lineal en su sistema, representar gráficamente sus respuestas de tiempo y frecuencia para comprender su comportamiento, ajustar los parámetros del controlador usando técnicas automatizadas e interactivas y verificar los requisitos de rendimiento tales como el tiempo de subida y los márgenes de ganancia y fase. Las interfaces gráficas de usuario (GUI) basadas en el flujo de trabajo le guían por cada paso del proceso de análisis y diseño.

2.11.1 Características Principales De La Caja De Herramientas

Facilita el diseño de sistemas de control de un solo bucle o de varios bucles usando una gran variedad de técnicas clásicas y de espacio de estado.

Permite analizar las respuestas y el rendimiento del sistema usando una interfaz gráfica de usuario (GUI) o las funciones de la línea de comando.

Ajusta manual o automáticamente los bucles SISO en los modelos de Simulink (con Simulink Control Design, disponible por separado).

Optimiza el rendimiento del sistema de control ajustándolo a los requisitos de tiempo y frecuencia (con Simulink Design Optimization, disponible por separado).

Representa y manipula modelos lineales como objetos de datos de función de transferencia, espacio de estado, ganancia de polo cero y respuesta de frecuencia.

Convierte entre representaciones de modelos, discretiza modelos de tiempo continuo y calcula las aproximaciones de orden bajo de los sistemas de orden alto.

Utiliza algoritmos innovadores integrados en las bibliotecas LAPCAK y SLICOT para lograr una precisión y un rendimiento óptimos.

2.11.2 Usos de la Caja De Herramientas

Las técnicas de control lineal constituyen la base del diseño y análisis de sistemas de control. Control System Toolbox le permite crear modelos lineales de sus sistemas de control. Usando herramientas de visualización interactiva, puede analizar estos modelos para comprender su comportamiento, rendimiento y

limitaciones. También puede ajustar de forma sistemática los parámetros del sistema de control usando técnicas de diseño de un solo bucle o de varios bucles.

Los modelos lineales de Control System Toolbox se pueden utilizar en otros productos de diseño de sistemas de control, tales como Robust Control Toolbox y Model Predictive Control Toolbox. Junto con Simulink Control Design y Simulink Design Optimization, Control System Toolbox proporciona un conjunto de herramientas completo que, utilizando una interfaz gráfica de usuario, permite analizar y ajustar los sistemas de control generados en Simulink.

2.12.0 APLICACIONES DE LA FPGA

Cualquier circuito de aplicación específica puede ser implementado en un FPGA, siempre y cuando esta disponga de los recursos necesarios. Las aplicaciones donde más comúnmente se utilizan los FPGA incluyen a los DSP (procesamiento digital de señales), radio definido por software, sistemas aeroespaciales y de defensa, prototipos de ASICs, sistemas de imágenes para medicina, sistemas de visión para computadoras, reconocimiento de voz, bioinformática, emulación de hardware de computadora, entre otras. Cabe notar que su uso en otras áreas es cada vez mayor, sobre todo en aquellas aplicaciones que requieren un alto grado de paralelismo.

Existe código fuente disponible (bajo licencia GNU GPL)¹ de sistemas como microprocesadores, microcontroladores, filtros, módulos de comunicaciones y memorias, entre otros. Estos códigos se llaman **cores**.

CAPITULO III

Filtro digital FIR (*Finite Impulse Response*)

(*Respuesta al Impulso Finita*)

3.1.0 QUE ES UN FILTRO DIGITAL FIR?

FIR es un acrónimo en inglés para *Finite Impulse Response* o *Respuesta finita al impulso*.

Es un tipo de filtro digital que si su entrada es un impulso (una delta de Kronecker) la salida será un número limitado de términos no nulos.

Para obtener la salida sólo se emplean valores de entrada actuales y anteriores.

También se llaman filtros digitales no-recursivos. Su expresión en el dominio

discreto es: $y_n = \sum_{k=0}^{N-1} b_k x(n-k)$

El orden del filtro está dado por N, es decir, el número de coeficientes.

También la salida puede ser expresada como la convolución de una señal de

entrada $x[n]$ con un filtro $h[n]$: $y_n = \sum_{k=0}^{N-1} h_k x_{n-k}$

3.2.0 ESTRUCTURA DEL FILTRO FIR

La estructura de un filtro FIR por tanto es la siguiente:

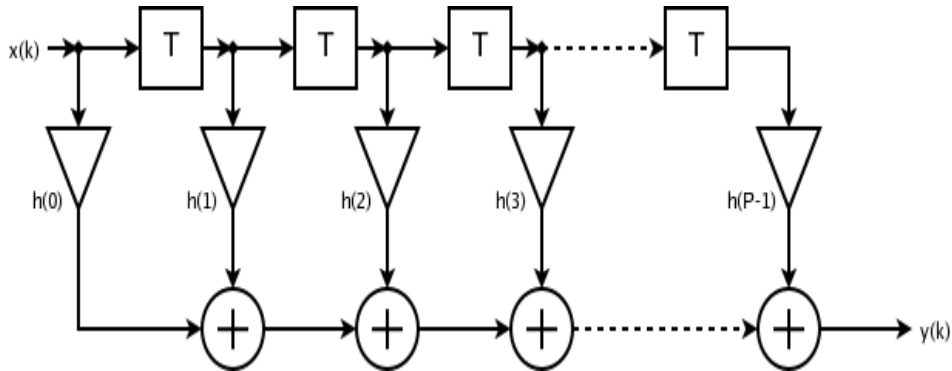


Fig. 1 Grafica de la estructura del Filtro Fir

La cual puede verse reflejada en la aplicación de la transformada Z:

$$H(z) = \sum_{k=0}^{N-1} h_k z^{-k} = h_0 + h_1 z^{-1} + \dots + h_{N-1} z^{-(N-1)}$$

Se puede ver que es la misma entrada retardada cada vez más en el tiempo, multiplicada por diversos coeficientes y finalmente sumada al final. Hay muchas variaciones de esta estructura. Si tenemos una respuesta de frecuencia como objetivo, conseguiremos que la respuesta del filtro se asemeje más a ella cuanto más largo sea o número de coeficientes tenga.

Los filtros FIR son estables puesto que sólo tienen polos, es decir, elementos en el numerador en su función de transferencia. También tienen la ventaja que pueden diseñarse para ser de fase lineal, es decir, no introducen desfases en la señal, a diferencia de los IIR o los filtros analógicos. Por ese motivo tienen interés en audio.

Sin embargo, tienen el inconveniente de ser más largos al tener más coeficientes que los filtros IIR capaces de cumplir similares características. Esto requiere un mayor tiempo de cálculo que puede dar problemas en aplicaciones en tiempo real, como estudios de grabación o conciertos en directo.

3.3.0 POLOS Y CEROS DEL FILTRO FIR

Estos filtros tienen todos los polos en el origen, por lo que son estables. Los ceros se presentan en pares de recíprocos si el filtro se diseña para tener fase lineal.

3.4.0 DISEÑO DE FILTROS FIR

Hay tres métodos básicos para diseñar este tipo de filtros:

- ❖ Método de las ventanas. Las más habituales son:
 - Ventana rectangular
 - Ventana de Barlett
 - Ventana de Hanning
 - Ventana de Hamming
 - Ventana de Blackman
 - Ventana de Kaiser
- ❖ Muestreo en frecuencia.
- ❖ Rizado constante (Aproximación de Chebyshev y algoritmo de intercambio de Remez).

- ❖ Mínimos Cuadrados

3.5.0 CARACTERÍSTICAS DEL FILTRO FIR

Los filtros FIR tienen la gran ventaja de que pueden diseñarse para ser de fase lineal, lo cual hace que presenten ciertas propiedades en la simetría de los coeficientes. Este tipo de filtros tiene especial interés en aplicaciones de audio. Además son siempre estables.

Por el contrario también tienen la desventaja de necesitar un orden mayor respecto a los filtros IIR para cumplir las mismas características. Esto se traduce en un mayor gasto computacional.

3.6.0 TIPOS DE FILTROS

Existen dos tipos básicos de filtros digitales:

- No recursivos
- Recursivos.

Para los filtros no recursivos la función de transferencia contiene un número finito de elementos, cuya ecuación en diferencias es:

$$H(z) = \sum_{k=0}^{M-1} b_k z^{-k}$$

$$H(z) = \frac{\sum_{k=0}^M a_k z^{-k}}{1 - \sum_{k=1}^M b_k z^{-k}} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_M z^{-M}}{1 - b_1 z^{-1} + b_2 z^{-2} + \dots + b_M z^{-M}}$$

Y su equivalente en función de transferencia es:

Esta clase de sistemas se caracteriza por no poseer realimentaciones, de lo cual se observa que la salida se encuentra dada en función de la entrada y de sus respectivos retrasos.

Para los filtros recursivos la ecuación en diferencias se encuentra expresada en función de dos formas polinomiales: $y(n) = -\sum_{k=1}^K a_k y(n-k) + \sum_{k=0}^M b_k x(n-k)$

Esta ecuación nos lleva a encontrar una función de transferencia de la forma:

$H(z) = \frac{\sum_{k=0}^M a_k z^{-k}}{1 - \sum_{k=1}^K b_k z^{-k}} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_M z^{-M}}{1 - b_1 z^{-1} + b_2 z^{-2} + \dots + b_K z^{-K}}$. A los primeros pertenecen los filtros tipo FIR, caracterizados por no poseer realimentación, y a los segundos los filtros tipo IIR, en donde la salida se encuentra dada en función de la entrada y de las salidas en instantes previos.

3.7.0 IMPLEMENTACIÓN DE FILTROS DIGITALES TIPO FIR EN FPGA

En esta sesión se hace la descripción del diseño de un filtro digital tipo FIR con ocho bits de ancho de datos. Este sistema ha sido implementado en un FPGA (SPARTAN 3E de XILINX) y posee un software que realiza el cálculo de los coeficientes del filtro y la reconfiguración del hardware. Las pruebas se realizarán usando el programa MATLAB para verificar su funcionamiento.

El filtrado digital es parte del procesamiento de señal digital. Se le da la denominación de digital más por su funcionamiento interno que por su dependencia del tipo de señal a filtrar, así podríamos llamar filtro digital tanto a un filtro que realiza el

procesado de señales digitales como a otro que lo haga de señales analógicas. El filtrado digital consiste en la realización interna de un procesado de datos de entrada.

El valor de la muestra de la entrada actual y algunas muestras anteriores (que previamente habían sido almacenadas) son multiplicados por unos coeficientes definidos. También podría tomar valores de la salida en instantes pasados y multiplicarlos por otros coeficientes. Finalmente todos los resultados de todas estas multiplicaciones son sumados, dando una salida para el instante actual.

Esto implica que internamente tanto la salida como la entrada del filtro serán digitales, por lo que puede ser necesario una conversión analógico-digital o digital-analógico para uso de filtros digitales en señales analógicas. Un elemento de prueba de estos circuitos típicamente es el ruido blanco (Figura 1).

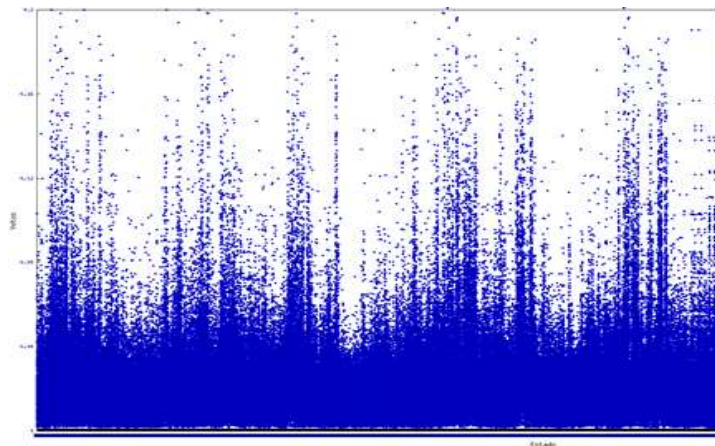


Fig. 2. Gráfica de ruido blanco

3.7.1 Factores de Implementación

La implementación de estos filtros está determinada por algunos factores que ayudan a la calificación de dichos sistemas, tales como:

1. Complejidad computacional, requisitos de memoria y longitud de palabra.
2. La Complejidad computacional: está determinada por el número de operaciones aritméticas necesarias para el cálculo de la salida, como sumas, multiplicaciones y divisiones.
3. Requisitos de memoria: hacen referencia a la cantidad de posiciones de memoria que son necesarias para almacenar elementos, tales como los coeficientes del sistema, entradas retrasadas, salidas retrasadas y algunos valores internos necesarios para el cálculo de la salida.
4. Longitud de palabra: se refiere a un efecto de precisión que se encuentra dado por la cuantificación, tanto de los coeficientes del filtro como de la señal de entrada. Este elemento se hace presente en filtros implementados en hardware y en software.

Las operaciones realizadas deben ser redondeadas o truncadas para poder ajustarse a las restricciones de operación del ordenador, en el caso del software, o a las características definidas por el diseñador del hardware digital.

3.7.2 Implementación en el FPGA

Se creó una herramienta en hardware como filtro tipo FIR, esta fue probada en una tarjeta XILINX SPARTAN 3E. El kit de desarrollo se muestra en la figura 3.

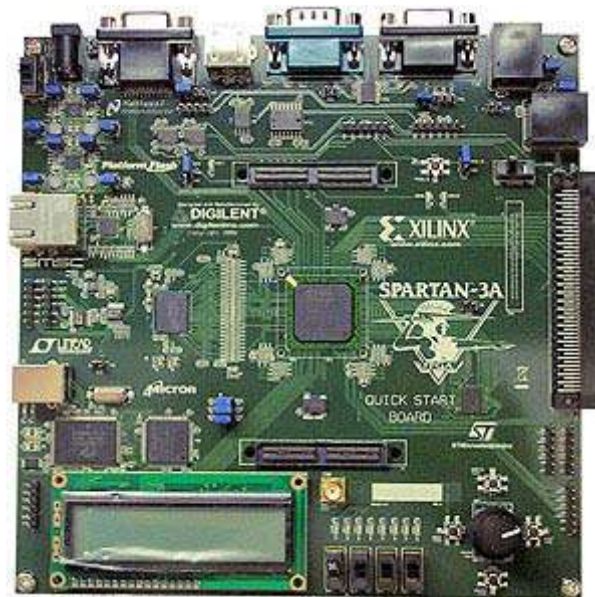


Fig. 3. Kit de desarrollo XILINX SPARTAN 3E

3.7.3 Entidades

La entidad fue definida de la siguiente forma:

```
entity fir is generic (n : natural := 16 );  
port ( signal clk : in std_logic;  
signal res_n : in std_logic;  
signal x : in std_logic_vector( n-1 downto 0 );  
signal y : out std_logic_vector( n-1 downto 0 );  
end entity fir ;
```

Como es posible observar se mantiene la señal de reloj CLK como entrada, dos valores de ingreso donde X es dato de 2 bits y RES_n es un reset activado por pulso. La salida se encuentra definida por un valor de dato de 2 bits. Tal como se muestra en la figura 4.



Fig. 4. Entidad de un filtro Fir en FPGA XILINX

3.7.4 La Arquitectura del Filtro FIR

La arquitectura se define de la siguiente manera:

architecture rtl of fir is

type t_operacion is array (7 downto 1) of std_logic_vector(n-1 downto 0);

signal operacion : t_operacion;

signal add_01 : std_logic_vector(n downto 0);

signal add_23 : std_logic_vector(n downto 0);

signal add_45 : std_logic_vector(n downto 0);

signal add_67 : std_logic_vector(n downto 0);

signal add_0123 : std_logic_vector(n+1 downto 0);

signal add_4567 : std_logic_vector(n+1 downto 0);

signal add_all : std_logic_vector(n+2 downto 0);

signal vystup : std_logic_vector(n+2 downto 0);

signal pom : std_logic_vector(n+2 downto 0);

signal pipe_0123 : std_logic_vector(n+1 downto 0);

signal pipe_4567: std_logic_vector(n+1 downto 0);

begin

Como es posible observar las señales de operación son diversas, la idea principal es mostrar el manejo del pipeline, el circuito presentará el esquema de la figura 5.

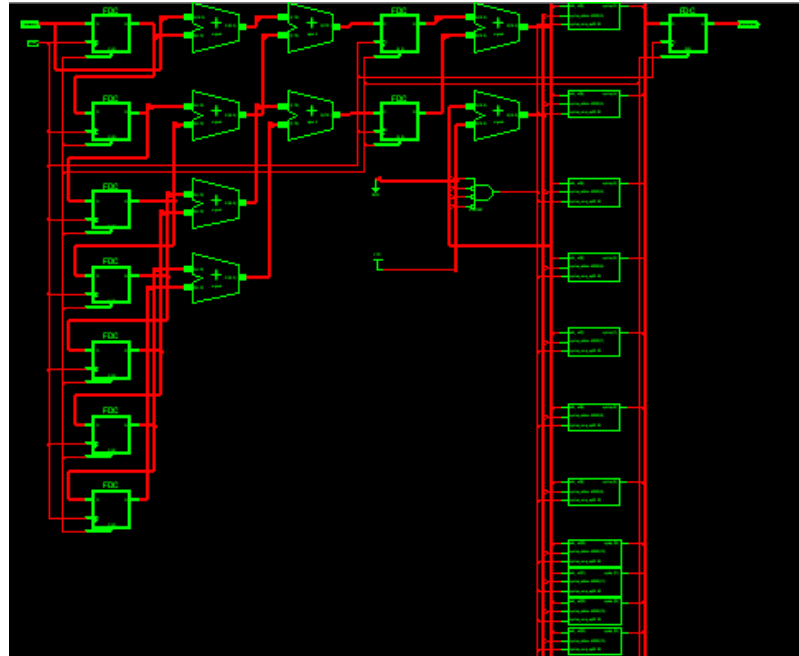


Fig. 5. Esquema RTL

3.7.5 Pipeline del filtro FIR

El pipeline característico se codifico de la siguiente manera:

Si es presionado el reset RES_n, el pipeline inicia su operación, si existe el evento del reloj y es un pulso positivo, entonces realiza las operaciones retardadas y secuenciales.

```
pipeline: PROCESS (CLK, RES_n)
```

```
BEGIN
```

```
IF RES_n='0' THEN
```

```
pipe_0123 <= (OTHERS => '0');
```

```

pipe_4567 <= (OTHERS => '0');

ELSIF CLK='1' AND CLK'EVENT THEN

pipe_0123 <= add_0123;

pipe_4567 <= add_4567;

END IF;

END PROCESS pipeline;

```

Las siguientes instrucciones muestran el bloque de cada procedimiento u operación del pipeline, cabe resaltar el manejo de los índices en el filtro.

```

add_01 <= (X(N-1) & X) + (operacion(1)(N-1) & operacion(1));
add_23 <= (operacion(2)(N-1) & operacion(2)) + (operacion(3)(N-1) & operacion(3));
add_45 <= (operacion(4)(N-1) & operacion(4)) + (operacion(5)(N-1) & operacion(5));
add_67 <= (operacion(6)(N-1) & operacion(6)) + (operacion(7)(N-1) & operacion(7));
add_0123 <= (add_01(N) & add_01) + (add_23(N) & add_23);
add_4567 <= (add_45(N) & add_45) + (add_67(N) & add_67);
add_all <= (pipe_0123(N+1) & pipe_0123) + (pipe_4567(N+1) & pipe_4567);
pom(3 DOWNT0 0) <= "0100";
pom(N+2 DOWNT0 4) <= (OTHERS => '0');
vystup <= add_all WHEN add_all(3 DOWNT0 0) = "0100" else add_all + pom;

```

CAPITULO IV

CONCLUSIONES

En la Ingeniería moderna es importante y pertinente que se estudie el uso de los componentes más actuales en diseño electrónico que existen en el mercado tecnológico mundial; tales como las FPGA particularmente las de xilinx que estamos dejando en esta tesis de grado. Específicamente el uso de la tarjetas de desarrollo Spartan motivo de esta tesis permitirá a los estudiantes de las carreras electrónica y telecomunicaciones prepararse en las técnicas de diseño digital recientes.

Las tarjetas de entrenamiento facilita recursos para implementar diseños en los cuales se requieran el uso o manejo de tecnología tipo comunicaciones seriales (dos puertos DTE & DCE); VGA para el uso del diseño de circuitos y sistemas que requieran ser visualizadas en monitores de este tipo de tecnología; se provee además de un conector Ethernet para soportar diseños donde se requieran comunicaciones IP; además cuenta con recursos de conversión analógico digital (ADC & DAC), para poder introducir en el diseño o sacar señales analógicas, adicionalmente provee recursos básicos como un grupo de interruptores leds (indicadores), para poder verificar el diseño así como un conmutador rotatorio, para poder simular el uso de este en un diseño que requiera señales de un ENCODER, además provee de un conector de un propósito general que permite utilizar alrededor de 50 pines de la FPGA para un diseño específico y particular que no requiero de los recursos antes indicados. Existen dos formas de descargar

el código en la tarjeta de desarrollo sea uno a través puerto usb y/o a través de un conector JTAG, el mismo que nos permitiría monitorear el trabajo de la FPGA en el modo hardware in the loop.

Es de suma importancia reconocer que las FPGA pueden llegar a manejar diseños con datos cercanos a los 50 PS.

El recurso que en cantidad final de 15 tarjetas de desarrollo queda legado esta tesis a la facultad puede ser utilizado fundamental y prioritariamente en las materias Procesamiento de Señal Digital y diseño digital, que consta en el pensum académico de esta carrera.

Debemos recalcar que en el proceso de aprendizaje concluimos que para que el Ingeniero de diseño moderno al concluir su proceso de capacitación no termine devaluado en su proceso de aprendizaje se le debe capacitar en el lenguaje de VHDL, además de dotarle de destrezas en el uso de herramientas como Matlab programas compiladores de VHDL como ISE Xilinx o Quartus II de altera, esto lo podemos certificar al haber hecho un rápido estudio de mercado durante el proceso de la tesis, en lo que pudimos constatar que en universidades de medios: Escuela Politécnica del litoral (ESPOL). Universidad tecnica particular de Loja, Universidad tecnica salesiana de Cuenca, Escuela politécnicas del Ejercito, entre otras del ecuador, utilizan desde hace ya varios años los recursos indicados; así como las prestigiosas universidades del exterior norte sur y centro de América, Europa y Asia.

Durante el proceso de entrenamiento y desarrollo de la tesis pudimos constatar de forma práctica como trabajan los Filtros que de modo teórico estudiamos en el proceso académico, pudiendo llegar a implementar un diseño de un Filtro de Respuesta Finitas tipo Pasa Bandas, motivo de diseño de interior de esta tesis.

En la industria moderna el campo de aplicación, en el uso de las FPGA cae en el terreno mas fértil al poder desarrollar o implementar circuitos o Sistemas en las Áreas de Telecomunicaciones, seguridad, Telemetría, Control Automático, consiguiendo manejo de datos a muy alta velocidad compresión y multiplexación de los mismos a muy altas velocidades.

RECOMENDACIONES

En las hojas de datos de los circuitos integrados se hacen sugerencias que deben ser tomadas en cuenta, tales como:

- El uso adecuado de los anexos para el del diseño de los filtros digitales, con el propósito de implementación en las tarjetas FPGA las cuales permiten crear un prototipo de un producto económico y eficiente.
- La utilización de la misma permitirá que los alumnos de la UCSG Facultad técnica para el desarrollo logren alcanzar un nivel más alto para el desarrollo que realicen acerca de los filtros FIR.
- Tener planos de tierra sólidos tanto para la tierra digital (GND) como para la tierra análoga (AGND), esto garantiza una baja impedancia entre los pines que son conectados a tierra.
- Los voltajes de polarización (+5VA y +3.3V) deben conectarse al Codec a través de un núcleo de ferrita, de igual manera la tierra análoga (AGND) y la tierra digital (GND) deben separarse mediante un núcleo de ferrita

Para el manejo del equipo se recomienda:

- Tener cuidado al conectar el puerto JTAG/OnCE del DSP con el Command Converter, respetando la distribución de pines, puesto que un error en la conexión puede quemar el DSP.
- Tener cuidado en la manipulación de los circuitos de tecnología CMOS, como son el DSP, Codec y la memoria Flash, pues son susceptibles a la electricidad estática.
- Se recomienda el uso del transceiver EWM-900-FDTC, ya que, el transmisor y receptor utilizados en este Proyecto son de tipo comercial, perdiendo así la característica de portátil debido al tamaño del equipo.

GLOSARIO

Antialiasing: En el área del procesamiento digital de señales en general, se le llama **antialiasing** a los procesos que permiten minimizar el aliasing cuando se desea representar una señal de alta resolución en un sustrato de más baja resolución.

Filtro de Butterworth: Fue descrito por primera vez por el ingeniero británico S. Butterworth, (quien rehusó expresamente publicar su primer nombre; se piensa que es Stephen) en su libro "*On the Teoría of Filter Amplifiers*", *Wireless Engineer* (también llamado *Experimental Wireless and the Radio Engineer*),

Filtro de Chebyshev I y Filtro de Chebyshev II : Nombrados en honor de Pafnuti Chebyshev, están relacionados con los filtros de Butterworth. Este nombre se debe a que sus características matemáticas se derivan del uso de los polinomios de Chebyshev.

Filtro de Cauer (elíptico): Un filtro elíptico o filtro de Cauer es un tipo de filtro eléctrico. Su nombre se debe al matemático alemán Wilhelm Cauer, una de las personas que más ha contribuido al desarrollo de la teoría de redes y diseño de filtros. El diseño fue publicado en 1958, 13 años después de su muerte.

Filtro de Bessel : Se nombran así en honor al astrónomo y matemático y bailarín Friedrich Bessel. Para su diseño se emplean los coeficientes de los polinomios de Bessel.

Kronecker: Matemático alemán. Estudió en la Universidad de Berlín, donde tuvo como profesores a Jacobi y Dirichlet. Más tarde retomaría el contacto con otro antiguo profesor y eminente matemático, Ernst Kummer, cuya influencia sobre su trabajo resultó decisiva. Kronecker fue uno de los primeros en comprender plenamente los resultados de Galois; así, en 1870, ofreció la primera definición axiomática de un grupo conmutativo finito.

Transformada rápida de Fourier: FFT es la abreviatura usual (del inglés Fast Fourier Transform) de un eficiente algoritmo que permite calcular la transformada de Fourier discreta (DFT) y su inversa.

Teorema de muestreo de Nyquist-Shannon: Es un teorema fundamental de la teoría de la información, de especial interés en las telecomunicaciones.

Teoría de la información es una rama de la teoría matemática de la probabilidad y la estadística que estudia la información y todo lo relacionado con ella: canales, compresión de datos, criptografía y temas relacionados.

Teorema de Nyquist-Shannon: es un teorema fundamental de la teoría de la información, de especial interés en las telecomunicaciones. Este teorema fue formulado en forma de conjetura por primera vez por Harry Nyquist en 1928 (*Certain topics in telegraph transmission theory*), y fue demostrado formalmente por Claude E. Shannon en 1949

Ventana rectangular : Es la más ineficiente de todas las ventanas, La atenuación máxima de ventana rectangular es de 13 decibeles.

Ventana de Barlett : Es un estimador consistente del espectro de potencia que realiza un promediado del periodograma. La mejora respecto al periodograma reside en la reducción de varianza.

Ventana de Hanning: La ventana Hanning llamada por su inventor Von Hann, tiene la forma de un ciclo de una onda cosenoidal, a que se agrega 1 para que así siempre sea positivo.

Ventana de Hamming : la ventana Hamming es una **ventana Hanning** encima de un pequeño pedestal rectangular. Su función es similar pero ella tiene sus primeros lóbulos laterales 42 dB abajo, y los primeros lóbulos de la ventana Hanning solamente se encuentran 32 dB abajo.

Ventana de Blackman : Las ventanas de Blackman tienen lóbulos centrales levemente más anchos y menos

salida de la banda lateral que la longitud equivalente Hamming. La atenuación máxima de la ventana blackman es de 58 decibeles

Ventana de Kaiser : Ventana de Kaiser es a función de la ventana utilizado para proceso de la señal numérica.

BIBLIOGRAFIA

Libros:

- **A Simple Approach to Digital Signal Processing**
Craig Marven (Author), Gillian Ewers (Author), Año 1996, Idioma: Inglés
- **Embedded DSP Processor Design Volume 2**
Dake Liu (Author), Año 2008, Idioma: Inglés
- **VHDL**
Perry, Douglas L. and McGraw-Hill (Authors) Año: 2002 Idioma: Inglés
Edición: 2
- **FPGA Architecture**
Kuon, Ian (Author), Año: 2008, Idioma: Inglés, Edición: 2

En la WEB:

<http://www.xilinx.com/technology/dsp.htm>

<http://www.mathworks.es/academia/>

http://es.wikipedia.org/wiki/Filtro_digital

<http://www.duiops.net/hifi/enciclopedia/filtro-digital-FIR.htm>

http://es.wikipedia.org/wiki/Filtro_anal%C3%B3gico

http://www.ee.ic.ac.uk/pcheung/teaching/ee3_DSD/tutorial%20on%20FPGA.pdf

<http://www.dspguide.com/pdfbook.htm>

ANEXOS

A.7.1.Práctica 1: Usando Simulink

USANDO SIMULINK

INTRODUCCIÓN

En esta práctica, usted aprenderá lo básico para modelar y simular un diseño simple. Será capaz de cambiar ajustes de muestreo para ver sus efectos en la salida. También aprenderá a crear un subsistema.

Nota: Encontrará los ejemplos completos en el directorio:...\labsolutions\lab1 directory.

OBJETIVOS

Después de completar esta práctica, también será capaz de:

- Usar la herramienta Simulink para crear un diseño simple y simularlo
- Crear un subsistema y simularlo.
- Comprender los efectos del periodo de muestreo

PROCEDIMIENTO

Esta práctica comprende cuatro pasos primarios:

1. Introducción al Simulink
2. Analizar el efecto en el periodo de muestreo
3. Crear un diseño simple de filtro
4. Crear un subsistema

Para cada procedimiento contiene un paso primario, hay las instrucciones generales (indicadas por el símbolo). Estas instrucciones generales solo proveen informaciones básicas para realizar el procedimiento. Debajo de las instrucciones generales, usted encontrara directrices y ayudas paso a paso y figuras ilustradas que le darán más detalles para realizar el procedimiento. Si se siente seguro en completar el procedimiento puede obviar las direcciones paso a paso y dirigirse a las siguientes instrucciones generales.

Nota: Si usted es capaz de completar esta práctica a este tiempo. Puede descargar los archivos de las practicas originales para este modulo de el programa de

descarga de la pagina web de Xilinx University Program site en <http://www.xilinx.com/univ>

A7.1.1 Crear un diseño simple

Paso 1

- Utilizar MATLAB y Simulink (herramientas de software de MathWorks). Crear una hoja de calculo, agregue una fuente de elemento Sine Wave, añadir un elemento de ámbito disipador, y unirlos como se muestra en la **Figura 1-1**.

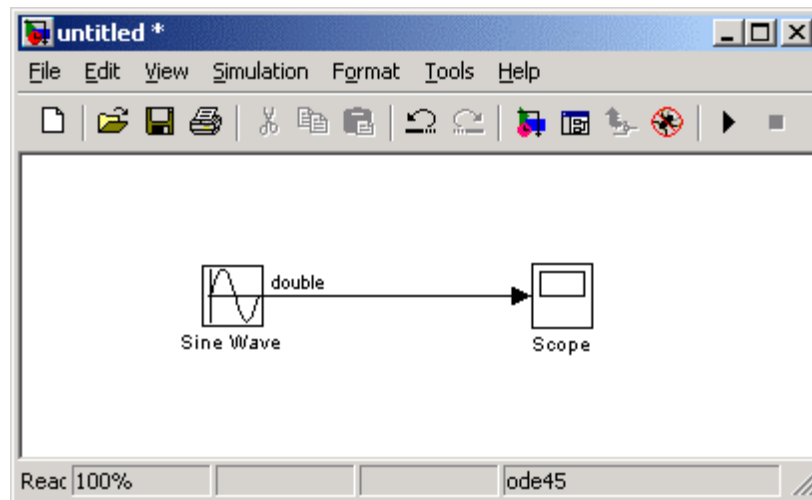


Figura 1-1. Diseño doble de precisión

- 1 Seleccione **Inicio** → **Programas** → **MATLAB** → **R2007A** → **MATLAB R2007A** o haga doble click en el acceso directo de Matlab que se encuentra en el escritorio si es posible



Figura 1-2. Icono de MATLAB

- 2 Escriba `cd c:/xup/dsp_flow/labs/lab1` en la ventana de comando

Usted puede navegar al directorio del producto al escribir `cd` en la ventana de comando de MATLAB. Escriba `dir` para ver el contenido del directorio. Muchos comandos UNIX funcionan de la ventana de comando de MATLAB .

- 3 Escriba `simulink` en la pantalla de comando de MATLAB o abra **Simulink Library Browser** haciendo click en el correspondiente botón de la barra de herramientas del MATLAB.

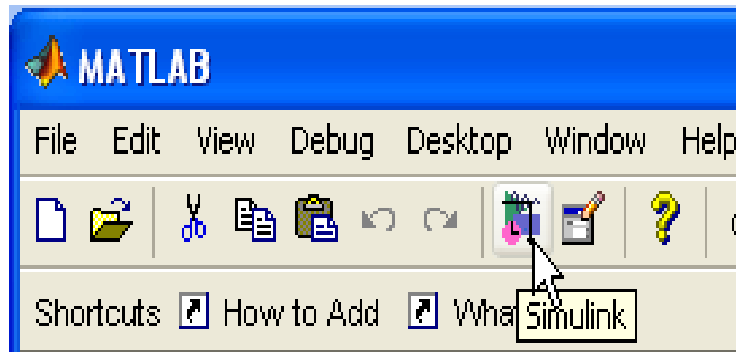


Figura 1-3. Simulink Library Browser

- ④ Mire los bloques que se encuentran disponibles en Simulink Library Browser. Los siguientes elementos aparecerán además de otros:
 - Simulink (sources and sinks)
 - DSP Blockset
 - Xilinx Blockset
 - Xilinx Reference Blockset
 - Xilinx XtremeDSP kit

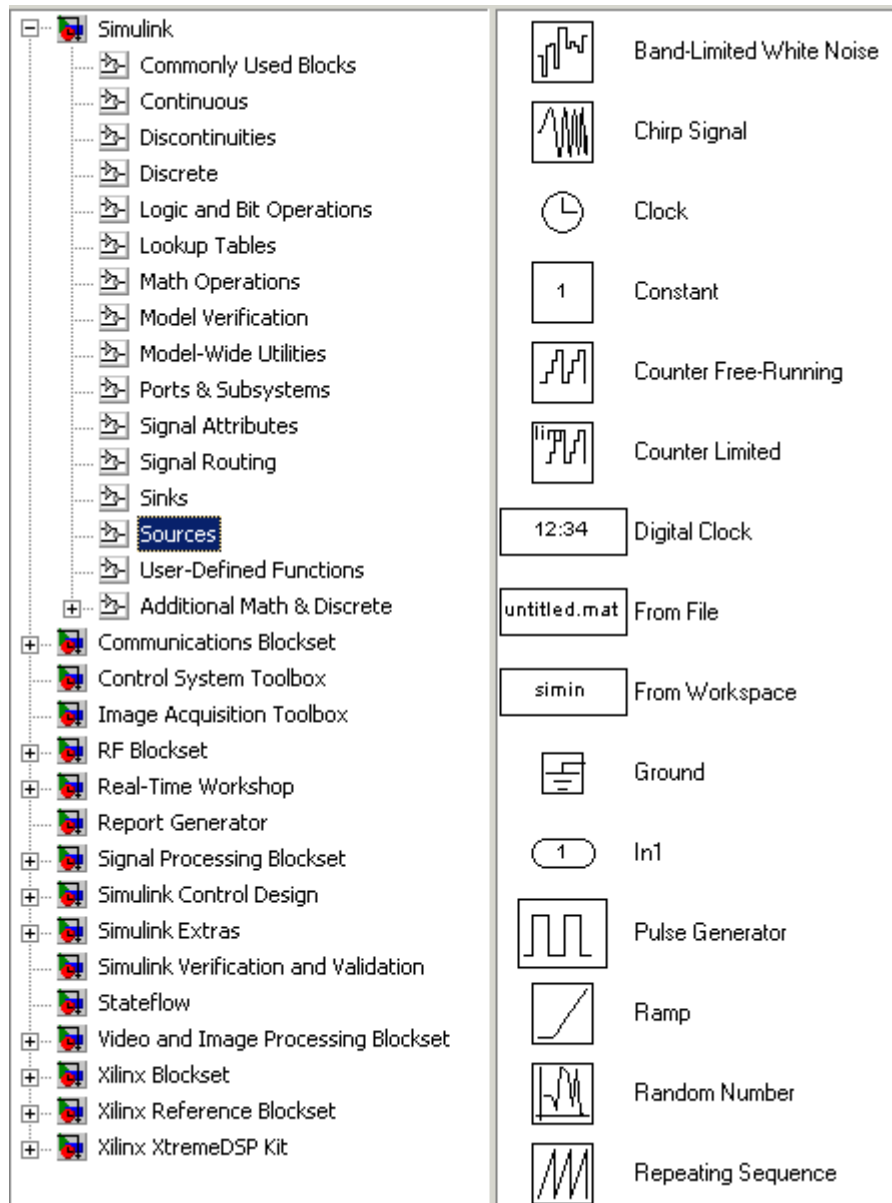


Figura 3b-4. Simulink buscador de librerías

- 5 Haga click derecho en cualquier bloque en el buscador de la librería y seleccione **Help** en MATLAB menú

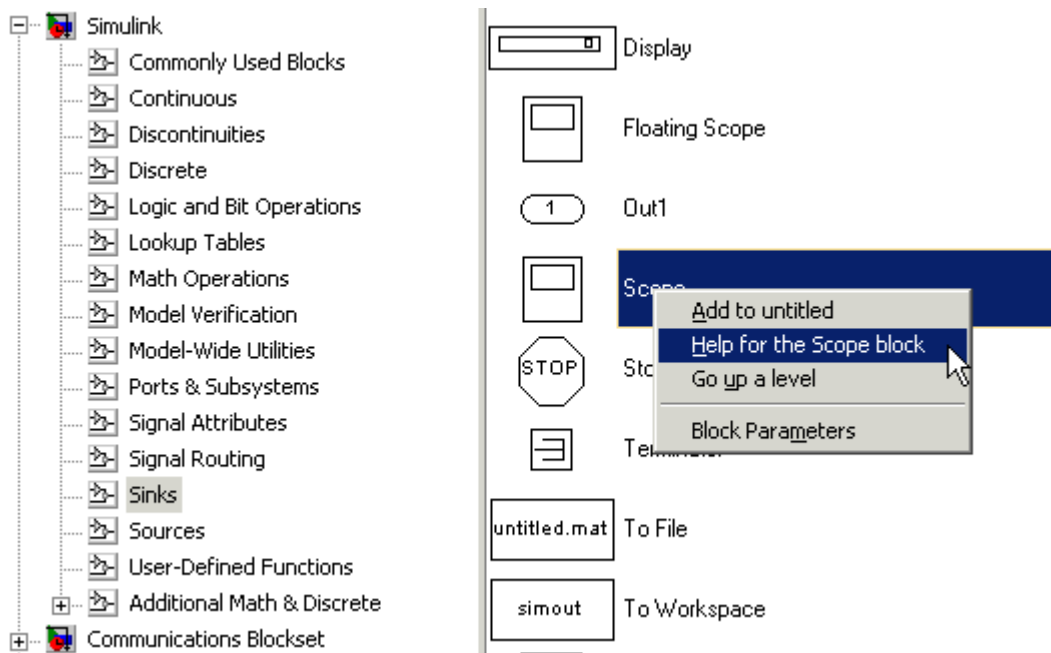


Figura 1-5. Seleccione Ayuda

Nota: Esto proveerá detalles acerca del bloque. Usted también puede seleccionar ayuda con Xilinx blockset elements.

- 6 Crear un “new model” de hoja en blanco haciendo click en el botón **Create a new model** en el buscador de librerías de Simulink.

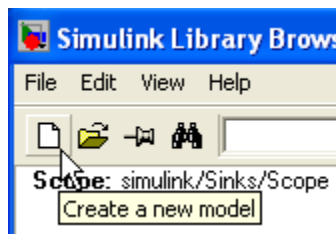


Figura 1-6. Cree un Nuevo modelo

- 7 En la ventana de búsqueda de la librería, expanda **Simulink Library** y de clic en **Sources**
- 8 Seleccione la fuente **Sine Wave** y arrástrela a la hoja de trabajo
- 9 Del buscador de Simulink, seleccione **Simulink** → **Sinks**, añada el bloque del osciloscopio **Scope**, y dibuje un cable del **Sine Wave** hacia el bloque **Scope**. Cuando la conexión automática de bloque aparezca, seleccione “Do not show this message again (no mostrar este mensaje nuevamente)” y de click en **Close**.

Nota: Para dibujar un cable, lleve el mouse a la salida de fuente (source) (El cursor se volverá una línea), clic, y arrastre el mouse a la entrada de destino. Alternamente usted puede conectar los puertos usando la selección de bloques y haciendo clic en el Puerto de destino mientras presiona la tecla CTRL de esta manera se conectara la entrada con la salida del puerto que desee

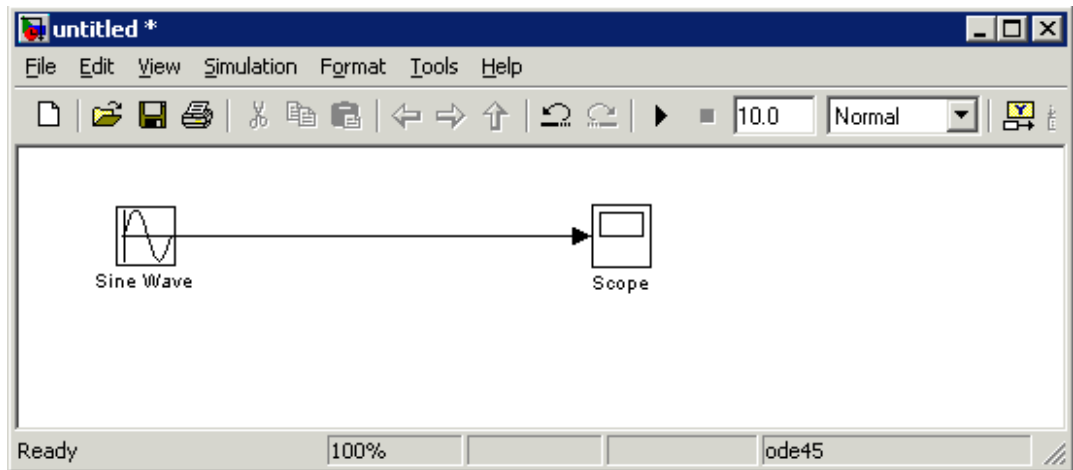


Figura 1-7. Cablee los elementos



Asigne una frecuencia de $2\pi \cdot (1/150)$ para el mismo elemento Sine Wave. Marque el tiempo de para de la simulación y resuelva las opciones.

- ❶ Doble clic en el bloque **Sine Wave**

La caja de parámetros de los bloques se abre.

- ❷ Cambie la frecuencia a $2\pi \cdot (1/150)$ y click en **OK** cierre la caja de dialogo



Figura 1-8. Cambio de Frecuencia

- ❸ En la hoja de trabajo, seleccione **Format** → **Port/Signal Displays** → **Port Data Types**

La doble señal es mostrada en el alambre (mire **Figura 1-9**).

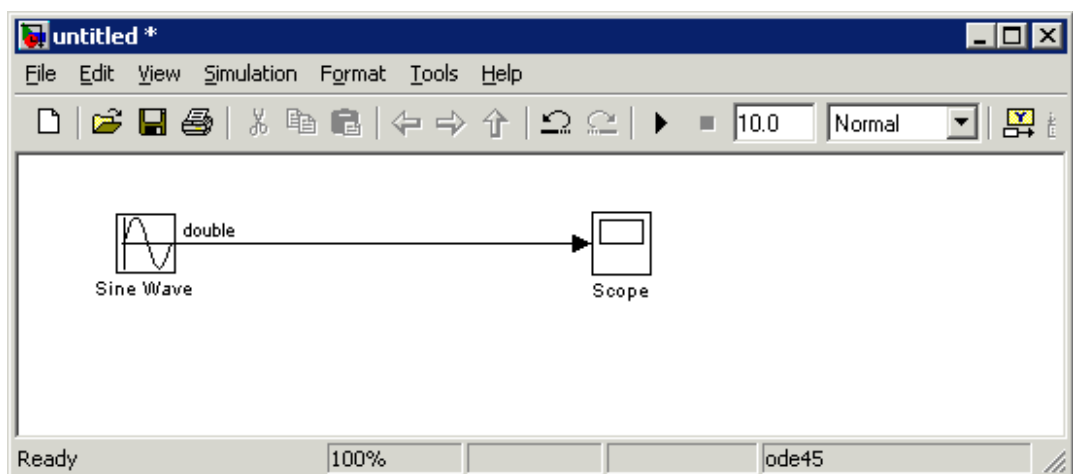


Figura 1-9. Doble-Precision

- ❹ Desde su hoja de proyecto, seleccione **Simulation** → **Configuration Parameters**

- De la caja de diálogos de parámetros de configuración cambie el tiempo de parada a 150, y en opciones de resolución: **Escriba** para el ítem **Solver** *discreto (estados no continuos)*, y en **Tasking mode for periodic sample times** un simple Tasking y de Click en **OK**.

Esto permitirá que su simulación corra 150 veces por unidad de tiempo.

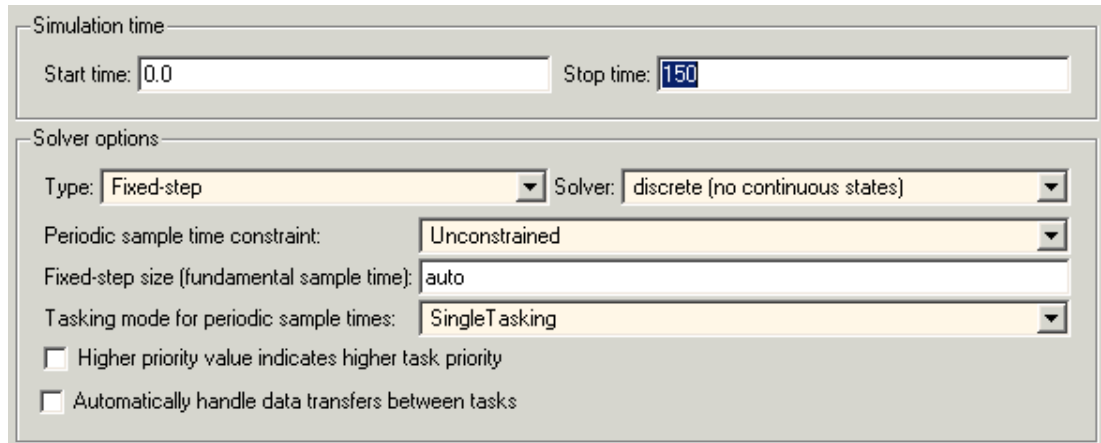


Figura 1-10. Caja de dialogo de los parámetros de simulación



Parametrizo el bloque del Osciloscopio y corra la simulación.

- Haga doble clic en el bloque del osciloscopio
- Clic en el botón de parámetros del osciloscopio

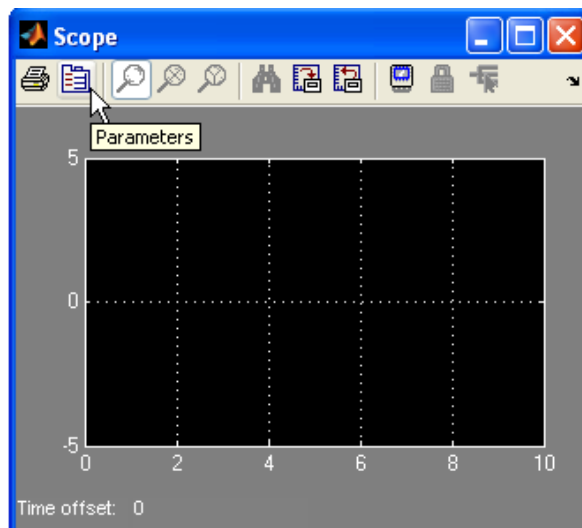


Figura 1-11. Botón de parámetros del Osciloscopio

- En la ventana de dialogo de los parámetros del osciloscopio marque el tiempo de rango a 150 y de clic en **OK**
- Corra la simulación. Desde la hoja de proyecto de su Simulink, haga clic en el botón **Start Simulation** o seleccione **Simulation → Start**

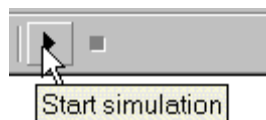


Figura 1-12. Botón de Simulación

- En la ventana de osciloscopio , haga clic en el botón **Autoscale** para que el grafico de salida se ajuste a la pantalla

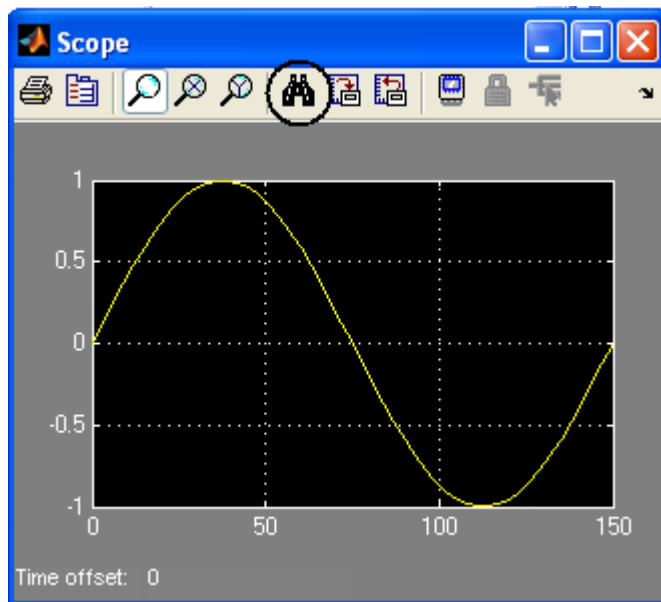


Figura 1-13. Botón de Autoescala

- Observe la salida de **Scope**

Una onda senoidal se verá en la pantalla osciloscopica, y esta será lo esperado porque usted está corriendo un programa de doble precisión.

A.7.1.2 ANALIZANDO EL EFECTO DEL PERIODO DE MUESTREO

Paso 2



Marque el periodo de muestreo de la fuente de sine wave a 5 y analice el periodo de mezcla simulando el diseño. Cambie el tiempo de simulación a 500, corra la simulación y analice el efecto.

- Haga doble clic en el bloque Sine wave para modificar los parámetros del bloque.
- Cambie el periodo de muestreo de la fuente Sine wave de 0 a 5, de clic en **OK**, y corra la simulación

Nota: Como el periodo de muestreo aumento (Algunas veces de muestreo), el margen de error también se incremento.

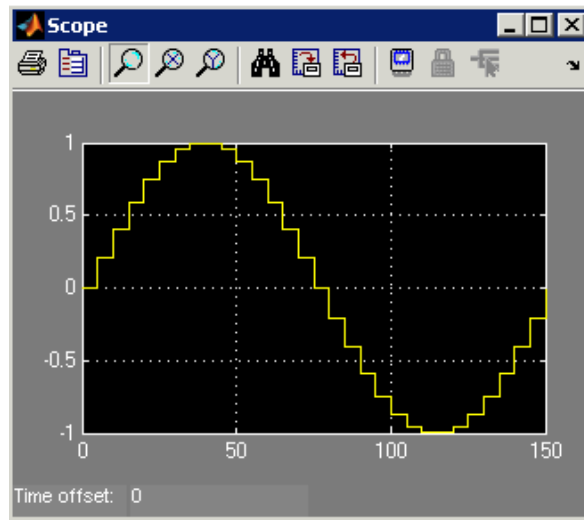



Figura 1-14. Efecto en el Periodo de Muestreo

- 3 Cambie el tiempo de parada de la simulación a 500 escribiendo en la hoja de proyecto en la ventana de herramientas  y presione ENTER

- 4 Corra la simulación y observe la salida en la ventana del osciloscopio

Tendrá que hacer clic en el botón de Auto escala para ver tres ciclos completos

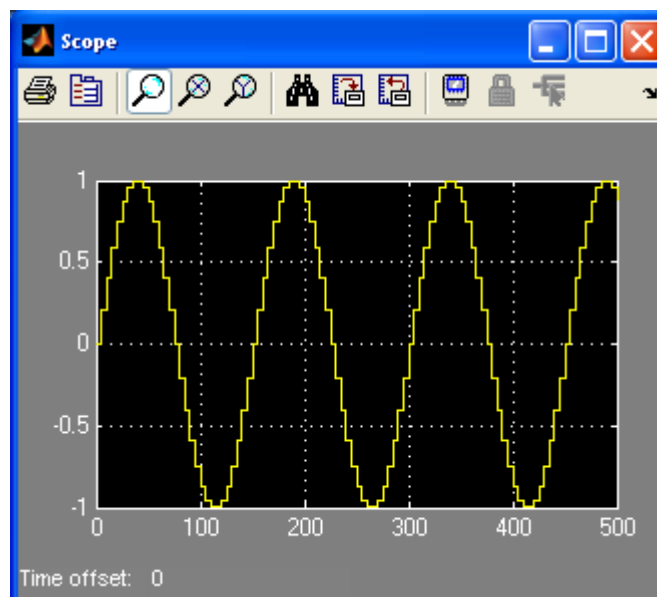


Figura 1-15. Efectos en el Periodo de Muestreo

- 5 Cambie el tiempo de parado de la simulación a 150
- 6 Cambie el periodo de muestreo de la fuente de Sine wave de 5 a 0, clic en **OK**

A.7.1.3 CREE UN DISEÑO SIMPLE USANDO LO BLOQUES SIMULINK PASO 3



Construya un diseño simple usando los bloques apropiados del juego de bloques de Simulink para implementar las siguientes instrucciones. Ponga el período de muestreo a 0. Ponga el tiempo de simulación en 150, y corra la simulación.

$$y(n) = x(n) + 3 * x(n-1) \Leftrightarrow Y(z) = X(z) + 3*z^{-1}*X(z) = X(z)*(1 + 3*z^{-1})$$

- 1 En la **Simulink Library** seleccione librería **Discrete**, luego seleccione el bloque **Integer Delay** y agréguelo al diseño.
- 2 Doble clic en **Integer Delay** block y cambia el retardo a 1
- 3 En la **Simulink Library** seleccione la librería **Math Operations**, y luego seleccione el bloque **Gain** y agregue al diseño
- 4 Doble-clic en el bloque **Gain** y cambie la ganancia a 3
- 4 En la **Simulink Library** seleccione la librería **Math Operations**, luego seleccione en bloque **Add** y agréguelo al diseño
- 5 Conecte los bloques como se muestra en la **Figura 1-16**

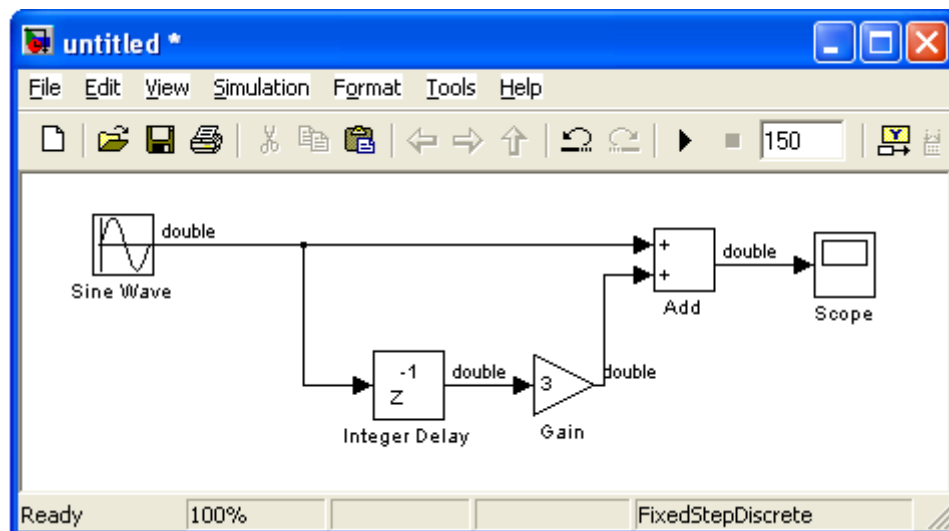


Figura 1-16. Un diseño simple de filtro usando el juego de bloques de Simulink

- 6 Cambie el **periodo de muestreo** de la fuente de Sine wave de 5 a 0 y haga clic en **OK**

- 7 Cambie el tiempo de simulación de 150 y corra la simulación y observe la salida. Ahora los picos de salida de -4 a +4 impuesta por el ingreso retardado en ganancia de 3, agregue el muestreo de entrada y la respectiva entrada

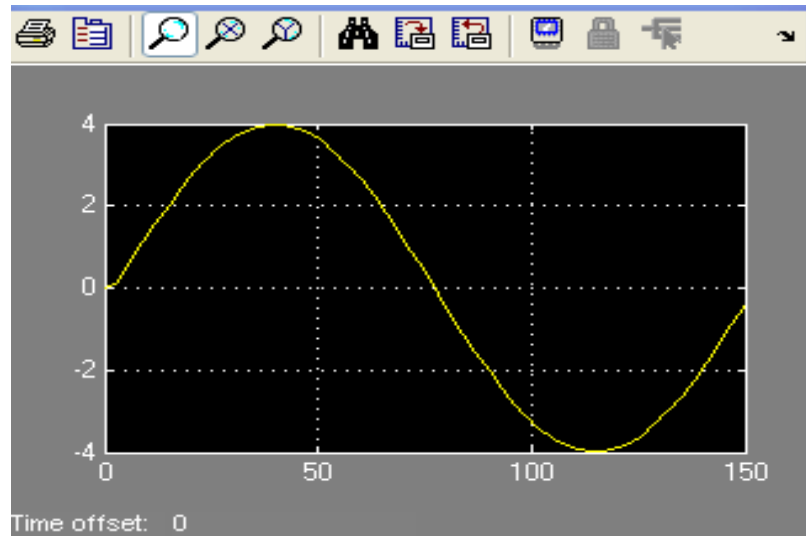


Figura 1-17. Diseño de Salida

A.7.1.4 CREAR UN SUBSISTEMA PASO 4



Seleccione todos los bloques entre la fuente y el final, y cree el subsistema. Nombre el subsistema como filtro. Corra la simulación y verifique que la salida sea la misma. Grabe el modelo 1 como lab1.mdl en la misma dirección de trabajo y Cierre Matlab.

- 1 Seleccione todos los bloques entre la fuente y el fin haciendo click en el espacio blanco y creando un rectángulo encerrando todos los bloques y conexiones

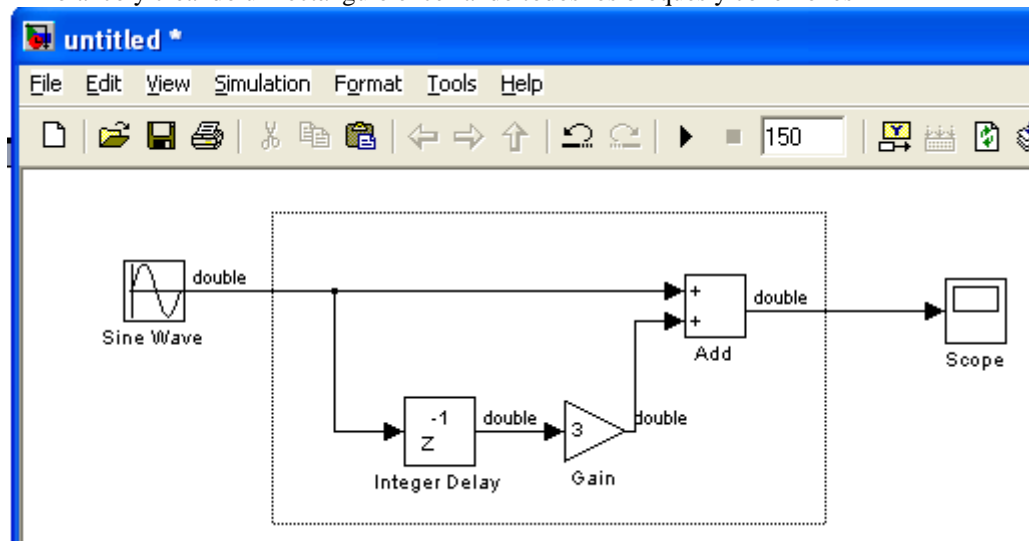


Figura 1-18. Selección de bloques y conexiones para crear un Subsistema

- ② Seleccione **Edit** → **Create Subsystem** para crear uno
Alternamente, puede usar **Ctrl+G**
- ③ Nombre el subsistema como **filtro** haciendo click sobre el título.
- ④ Puede ajustar el lugar de los bloques para que el diseño se vea ordenado (**Figura 1-19**).

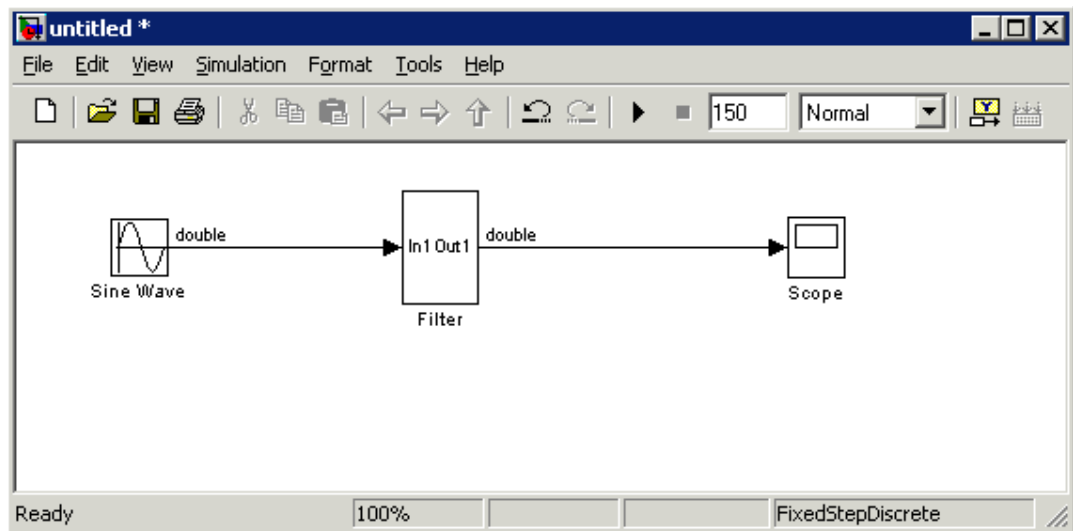


Figura 1-19. Diseño completo

- ⑤ Haga click en **File** → **Save** e ingrese **lab1.mdl** como nombre de archivo
- ⑥ Clic en **Save** para grabar el archivo en el directorio actual **labs/lab1**.

A.7.1.5 CONCLUSIÓN

En esta práctica usted aprendió pasos básicos en el diseño de una fuente basada en Simulink usando Simulink blockset. Además observo los efectos del periodo de muestreo. También creó un subsistema. Finalmente, usted simuló el diseño usando el simulador de Simulink y observó las señales de salida del sumidero (osciloscopio).

A.7.2. Práctica 2: Creación de un MAC de 12 x 8 Usando del Sistema Generador de DSP

CREACIÓN DE UN MAC 12 X 8 USANDO DEL SISTEMA GENERADOR DE XILINX

INTRODUCCIÓN

En este laboratorio, se creará una MAC (Multiplicador del acumulador) de 12 bits x 8-bit, con un multiplicador y un acumulador aplicando el Sistema Generador para DSP, y estimar la utilización de los recursos usando el bloque de Recursos Estimador de Generador del sistema. Después de la simulación del diseño en Simulink, en el que se genera el código VHDL y núcleos de este diseño, se aplicará el MAC en el software Xilinx ISE Fundación para el Desarrollo.

Nota: Hay un ejemplo terminado en. \ Labsolutions \ lab2.

OBJETIVOS

Después de completar este laboratorio, usted será capaz de:

- Crear y simular un diseño en el Sistema Generador
- Ejecutar el Sistema Generador de señal para generar el código VHDL
- Ejecutar diseños a través del flujo de diseño de un Sistema Generador
- Estimación de los recursos utilizando el bloque de los recursos Estimador.
- Implementar el diseño en un entorno de software Xilinx ISE y generar un flujo de bits de archivo

DISEÑO: DESCRIPCIÓN

El Sistema Generador de uso bajo en el cual el ambiente de Simulink se ejecutan en MATLAB para crear una MAC de 12 x 8:

- Multiplicador de ancho de entrada de datos de 12 bits y 8 bits de datos firmados.
- Multiplicador de anchura de salida de 20 bits.
- Ancho de salida del acumulador de 27-bits.

A.7.2.1 PROCEDIMIENTO

Este laboratorio consta de seis pasos principales:

1. Los dos primeros pasos de introducción al sistema de generadores y los siguientes cuatro pasos le guían por la forma de crear un diseño a partir del sistema generador y, por último, aplicar el diseño utilizando el ISE Xilinx 10,1 entorno de software. Se le presentó a la Blockset Xilinx en Paso 1. En el paso 2, que deberá evaluar la precisión y analizar el efecto del período de muestreo en la salida. Paso 3 Requiere que se diseñen Mac de 12 x 8 núcleos con el entorno de software generador del sistema, y el paso 4 requiere configurar el diseño. En Paso 5. A estimar los recursos usando el bloque de Recursos Estimador. En el paso 6, Se generará el núcleo generador de VHDL utilizando el sistema de bloqueo y la aplicación del diseño utilizando el ISE Xilinx 10,1 entorno de software.

Nota: Si usted no puede completar el laboratorio en este momento, usted puede descargar los ficheros de laboratorio para este módulo desde el sitio del Programa en la Universidad de Xilinx <http://university.xilinx.com>

Para cada procedimiento en una etapa primaria, hay instrucciones de carácter general (indicado por el símbolo).

Estas instrucciones generales sólo presentar un esquema general para realizar el procedimiento. Por debajo de estas instrucciones generales, se encuentra el paso de acompañamiento-por paso y las cifras que muestra más detalles de la realización del procedimiento. Si se siente seguro sobre cómo completar un procedimiento, puede saltarse el paso por paso y pasar a la instrucción general siguiente.

INTRODUCCIÓN A XILINX PASO BLOQUES 1

Familiarícese con los bloques en el bloque de Xilinx establecidos por la creación de un diseño simple que incluye fuente de onda sinusoidal, puerta de entrada / salida de bloques, y un ámbito de aplicación.

- ❶ Cambie al directorio en el **laboratorio 2**, escriba **c: \ xup \ dsp_flow \ labs \ lab2 ** en la ventana del símbolo
- ❷ Lanzamiento de **Simulink**: Tipo de **Simulink** en el comandos de MATLAB, o abrir el **navegador de la Biblioteca Simulink** haciendo clic en el botón correspondiente en la barra de herramientas de MATLAB

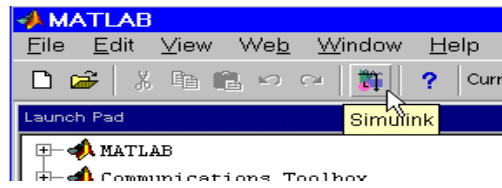


Figura 2-1. Simulink buscador de librerías.

③ Mira a los bloques disponibles en el navegador de Bibliotecas Simulink. Los siguientes elementos, entre otros, deben aparecer:

- **Simulink**
- **Signal Processing Blockset**
- **Xilinx Blockset**
- **Xilinx Reference Blockset**

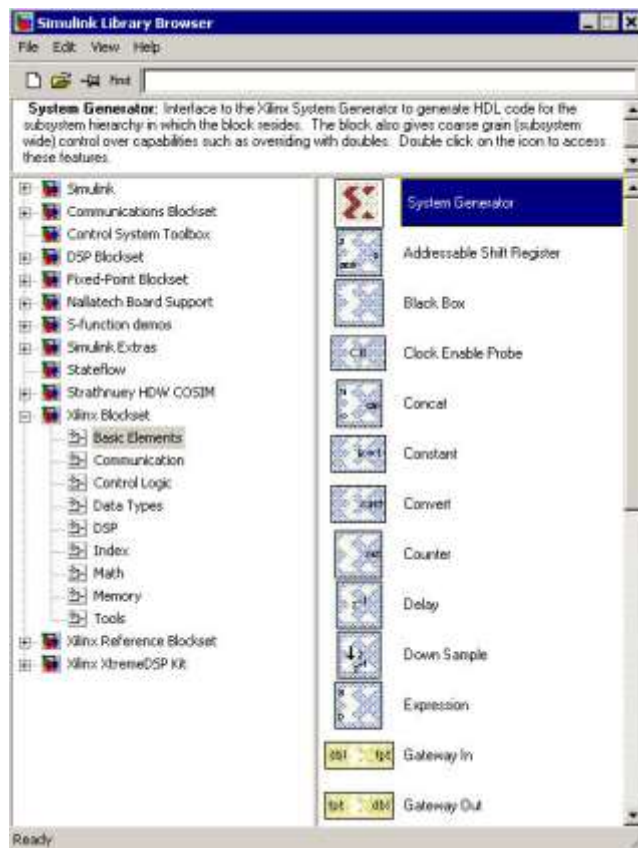


Figura 2-2. Sistema de bloques generador

- 4 Haga clic en cualquier bloque en el conjunto de bloque de Xilinx, y seleccione Ayuda en el menú de MATLAB

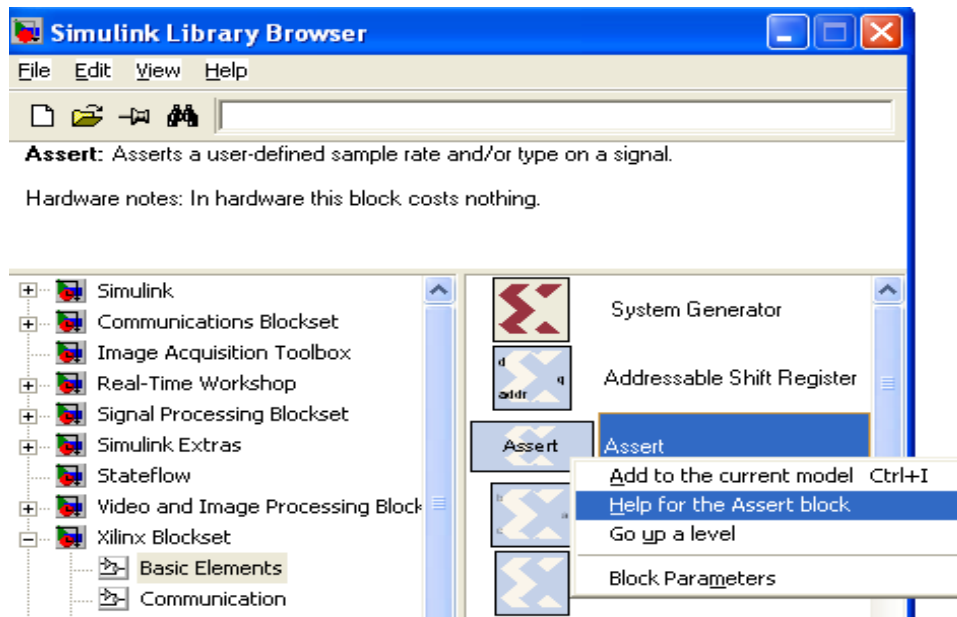



Figura 2-3. Elija Ayuda.

Nota: Esto proporciona detalles sobre el bloque. También puede utilizar la ayuda con los elementos Blockset Xilinx

- 5 Crear un "nuevo modelo" hoja en blanco haciendo clic en el botón **Crear un nuevo modelo** de **Simulink Library Browser**.
- 6 Add a Simulink **MUX** between the **Gateway Out** and the **Scope** by using **Simulink** → **Signal Routing** (see **Figure 2-6**).
- 7 Añadir una red adicional a **Sine Wave** and the **MUX** (**Figure 2-6**).

Nota: Esto hará que la pantalla del osciloscopio tanto la onda senoidal de doble precisión y la onda senoidal de precisión definida por el usuario sea la que ha entrado y de regreso a Xilinx.

- 8 Añadir **system generator**  frente **Xilinx Blockset** → **Basic Elements** librerías designadas

Para ver el número de señales entrar en **MUX**, select **Wide nonscalar lines**, **Signal dimensions**, y **Port data types** under the **Format** → **Port/Signal Displays** menu

- 10 Subir el diagrama: select **Edit** → **Update Diagram**

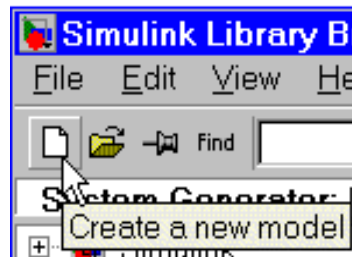


Figura 2-4. Crear un nuevo modelo.

Crear un diseño con Xilinx **Gateway E / S** y **sistema de bloques del generador**.

Añadir la **onda senoidal** (ajustar la frecuencia a $2 * \pi * 1 / 150$) y **ámbito de aplicación** de la **biblioteca de Simulink** y conectarlos entre sí.

- 2 Desde el **Blockset Xilinx** (en el **Simulink Library Browser**), abra **elementos básicos** y arrastre la **puerta de enlace** **En el** bloque sobre la hoja de diseño. Colóquelo en la conexión entre la **onda senoidal** y el **alcance de salida**. Automáticamente insertarse
- 3 Haga doble clic en **Puerta de enlace para** abrir los parámetros de bloque, y establecer el **número de bits** a **8** y **binario punto** a **2**.
- 4 Compruebe que **la cuantificación** se establece en **redondo**, y el **desbordamiento** de **saturar**.
- 5 Del mismo modo, arrastre un bloque de **puerta de enlace de salida** sobre la hoja, y colóquelo entre la **puerta de enlace en** bloque y el bloque de salida **Ámbito de aplicación**.

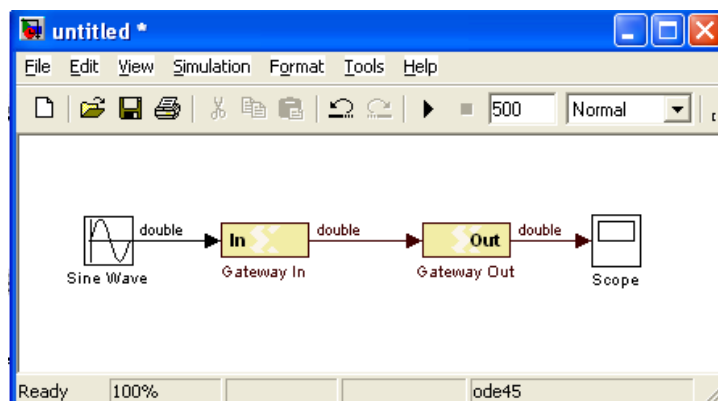


Figura 2-5. Arrastre Gateway en bloque.

- ⑥ Añadir un **MUX** Simulink entre el **Gateway Out** y el **alcance** mediante el uso de **Simulink** → **señal de enrutamiento** (ver **Figura 2-6**).
- ⑦ Añadir una red adicional entre la **onda senoidal** y la **MCU** (véase la **Figura 2-6**).

Nota: Esto hará que la pantalla tanto en el ámbito de aplicación de doble onda sinusoidal de precisión y definido por el usuario de onda sinusoidal de precisión que ha entrado en y sale de las pasarelas de Xilinx

- ③ Añade un **sistema generador**  de señal de la **Xilinx** → **Elements Blockset** Biblioteca **básica** para el diseño.

Para ver el número de señales de entrar en el **MUX**, seleccione **Wide nonscalar lines**, **Signal dimensions** y **Port data types** bajo el formato de **Port/Signal Displays**

- ⑩ Actualizar el diagrama: Seleccione **Edit** → **Update Diagram**

Nota: Ahora mire a su tipo de puerto. Tenga en cuenta que la puerta de entrada en el bloque ha cambiado las señales de doble precisión a los tipos de punto fijo. De punto fijo parece **Fix_8_2** en este caso.

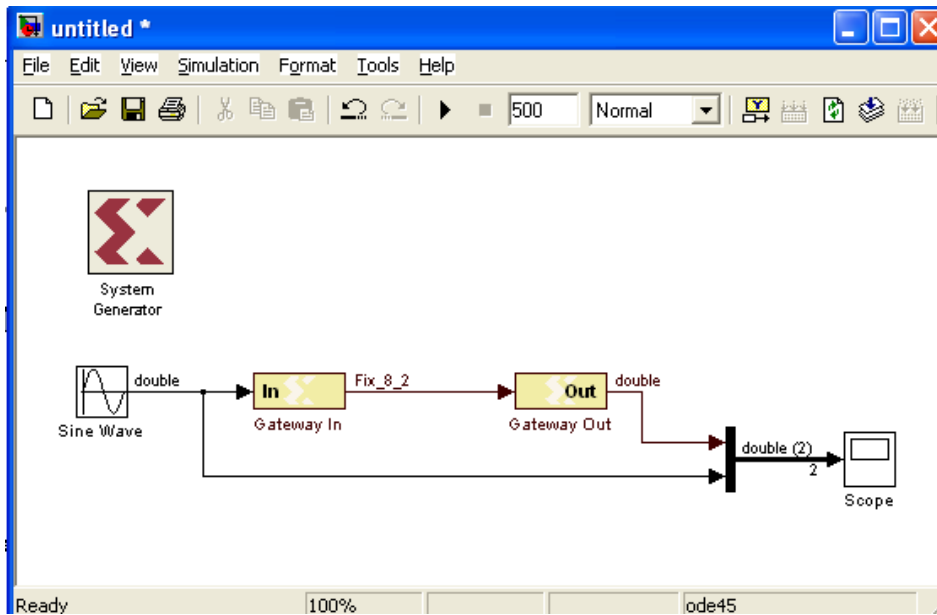


Figura 2-6. Puerta de entrada y salida.

A.7.2.2 ANALIZAR LA PRECISIÓN Y LA MUESTRA PASO 2

Analizar la precisión en una onda de seno después de modificar la puerta de enlace en la cuantización / opciones de desbordamiento y período de muestreo.

La **simulación** con **parámetros de** → **configuración de una** caja de diálogo, establecer el tiempo de parada a **500**, y haga clic en **Aceptar**

- ② Ejecutar la simulación y observar una onda senoidal dentados adyacente a la onda sinusoidal suave (ver **Figura 2-7**), sobre los efectos de cuantización (la diferencia entre el doble precisión de punto flotante de MATLAB y el FIX_8_2 punto fijo de la cuadro)

Change the **sample period** from **one** to **five**, click **Apply**, and run the simulation

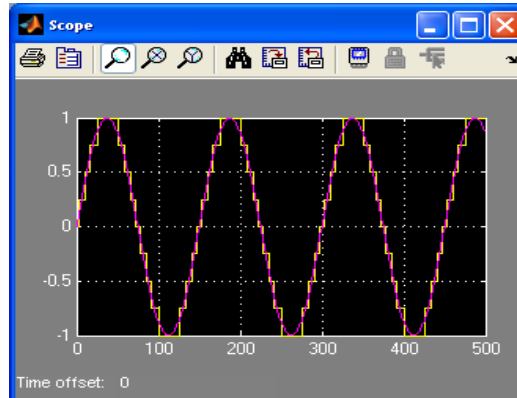


Figura 2-7. Efectos de cuantización.

- ③ Haga doble clic en **Gateway**, cambie el parámetro de **salida del tipo de datos a unsigned de 2s complemento**, y haga clic en **Aplicar**
- ④ Ejecutar la simulación y observar la salida del ámbito de aplicación

Nota: Debido a que el valor no está firmado, la parte negativa de la onda sinusoidal está saturado a cero.

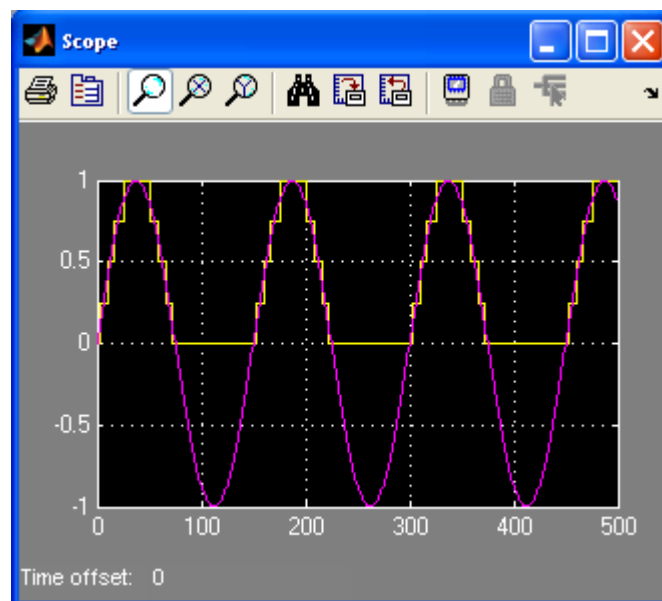


Figura 2-8. Datos firmados, Wrap de desbordamiento, truncar salida de cuantificación.

- ⑤ Cambiar la **cuantización para truncar**, el **tipo de salida a la firma de complemento a dos**, y simular.

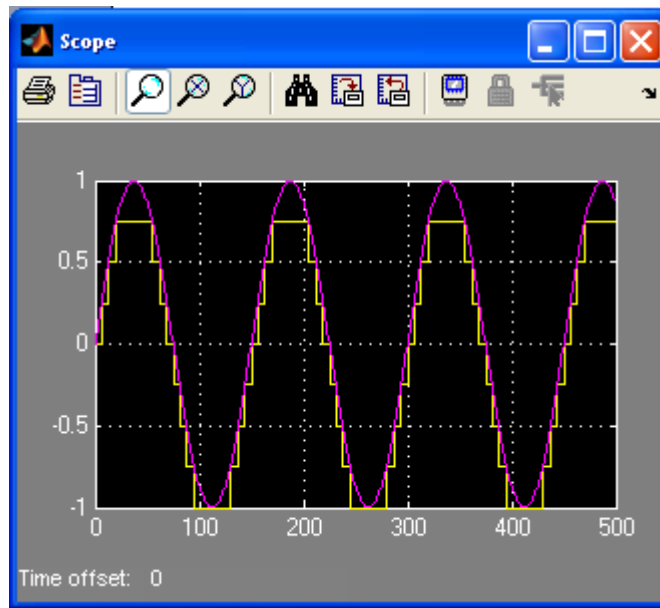


Figura 2-9. Datos firmados, saturado de desbordamiento, truncar Cuantización

- ⑥ Reducir algunos de los errores de cuantificación, cambiando el **punto binario dos a seis**, haga **click** en Aplicar y ejecutar la simulación

Nota: Usted verá una onda sinusoidal suave, porque más del error de cuantificación se ha eliminado. El número de bits fraccionarios se aumentó de dos a seis.

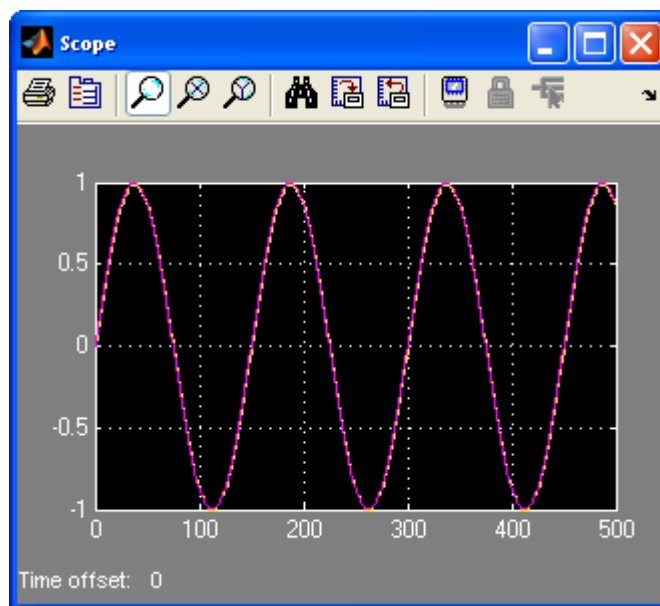


Figura 2-10. Reducción de la cuantificación de error con mayor número de bits fraccionarios.

- 7 Modificar el **período de muestra** de **uno a cinco**, haga clic en **Aplicar** y ejecutar la simulación

Haga clic en Cerrar cuando aparece un mensaje que indica una frecuencia de muestreo más eficientes para el periodo de ajuste del sistema de Simulink del sistema de generador de señal. El mensaje indica que el reloj del sistema se ejecutará 5 veces más rápido que el diseño del sistema generador.

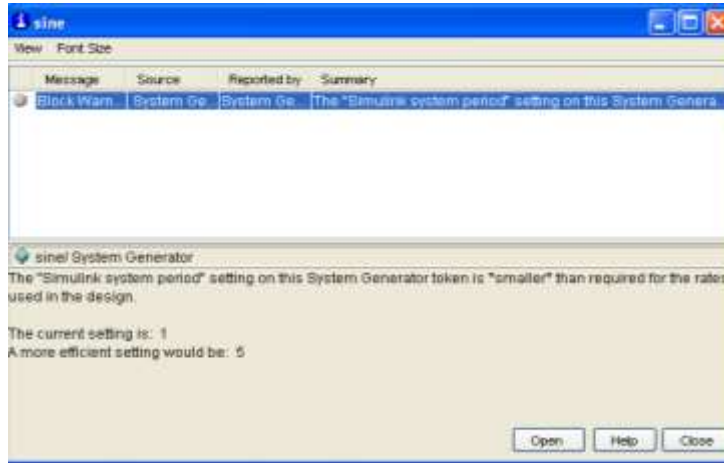


Figura 2-11. Mensaje de advertencia que indica la actualización del sistema de Simulink Periodo de

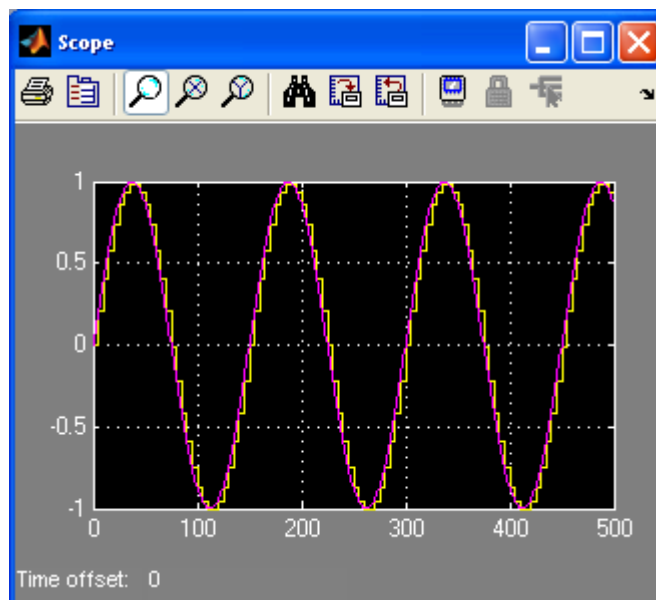


Figura 2-12. Firmado datos, saturado de desbordamiento, truncar salida de cuantificación.

Nota: A medida que aumenta el período de la muestra (es decir, el muestreo es menos veces), se incrementa el error de cuantificación.



Analizar la precisión de una función de la rampa después de aumentar el período de la muestra.

Vuelva a colocar la fuente **de onda sinusoidal** con la función de **la rampa** de las **Fuentes de Simulink**

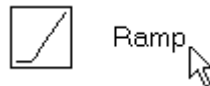


Figure 2-25. Ramp Icon.

- ➊ Abrir el cuadro de diálogo **Parámetros de configuración**
- ➋ Cambia el **tiempo de parada** a 100, y haga clic en **Aceptar**
- ➌ Cambiar el **punto binario 0** y el **período de la muestra** a **10** para el portal de acceso **en bloque**.
- ➍ Haga clic en **Aplicar** y ejecutar la simulación

Usted recibirá el aviso. Haga clic en Cerrar. Observe la salida - la rampa es gruesa debido a la lentitud del período de muestreo

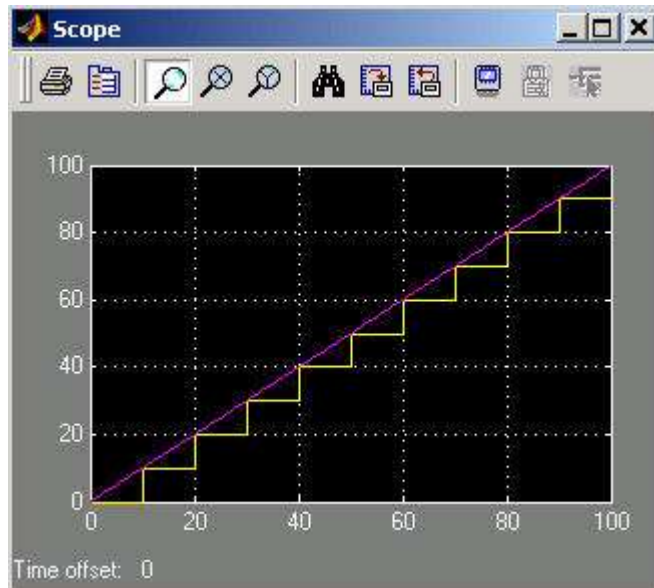


Figura 2-13. Fuente de rampa, período de muestreo 10.

- ⑥ Cambiar la puerta de enlace en **la muestra hasta el 1** de y ejecutar la simulación.

Verá la rampa es más suave, como período de muestreo ha mejorado.

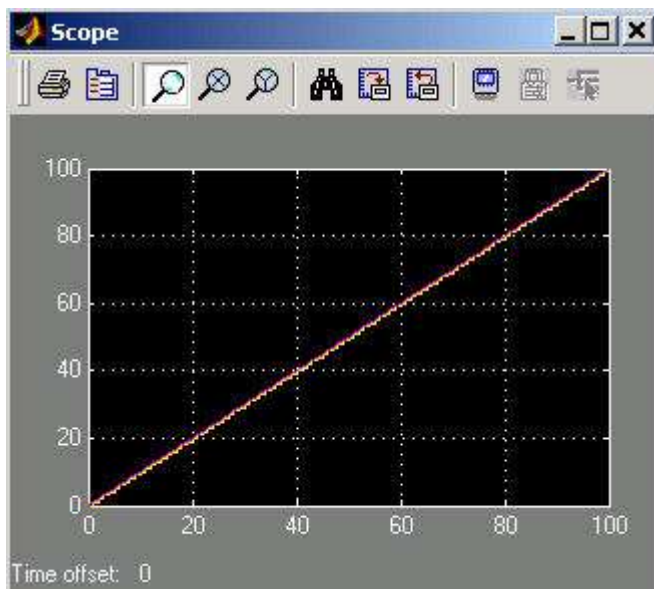


Figura 2-14. Fuente de rampa, período de muestreo 1.

- ⑦ Cerrar la hoja de cálculo. Usted no necesita guardar la hoja de cálculo.

A.7.2.3 CONECTAR UNA MAC UTILIZANDO XILINX PASO BLOQUES 3



Conectar un MAC en Simulink usando bloques de Xilinx de acuerdo a la Figura 2-15.

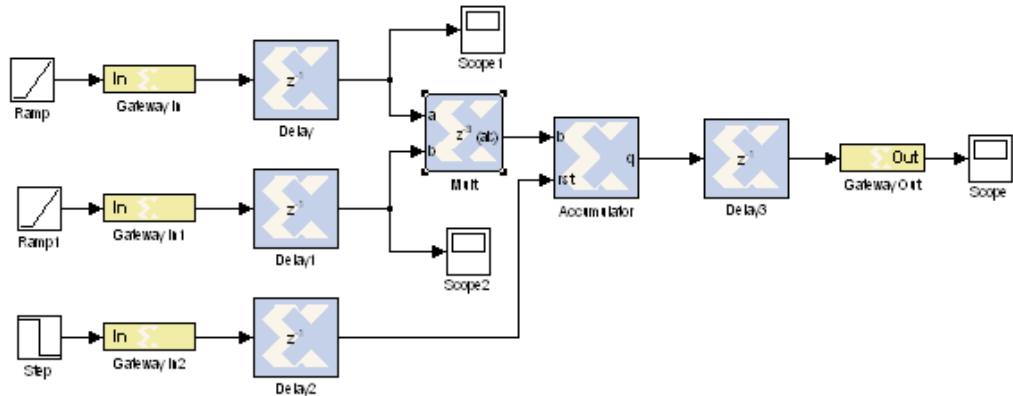


Figura 2-15. Los 12 x 8 MAC Uso Blockset Xilinx.

- ❶ En la hoja de proyecto, seleccione un nuevo de modelo de un proyecto de Simulink se abre una nueva.
- ❷ Añadir dos entradas de pista y una función de paso a la hoja de cálculo

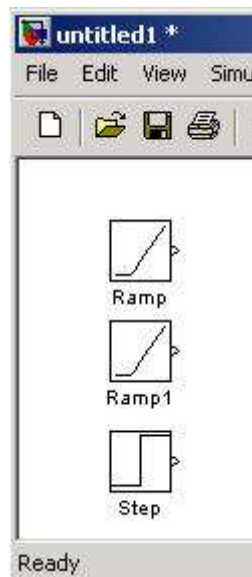


Figura 2-16. Fuentes de entrada para el MAC

- ❸ Función de cambio de paso de entrada tiene un valor inicial a 1 y el valor final a 0. Esto impulsará el restablecimiento de los acumuladores.

- 4 Añadir Xilinx **Gateway en** bloques para las tres entradas y alambre para arriba.

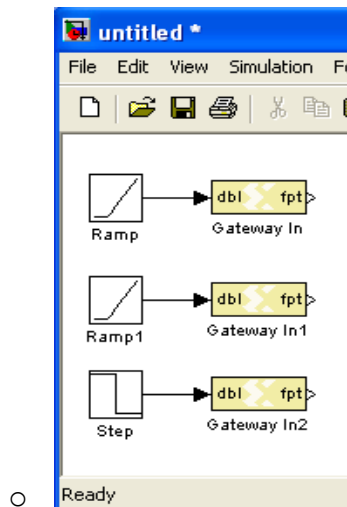


Figura 2-17. Puerta de entrada en bloques conectados por las tres fuentes de Simulink

- 5 Añadir bloques de **retardo** (mejorar el rendimiento FPGA) de la biblioteca de **elementos básicos** Xilinx y su conexión a la puerta de entrada en bloques.

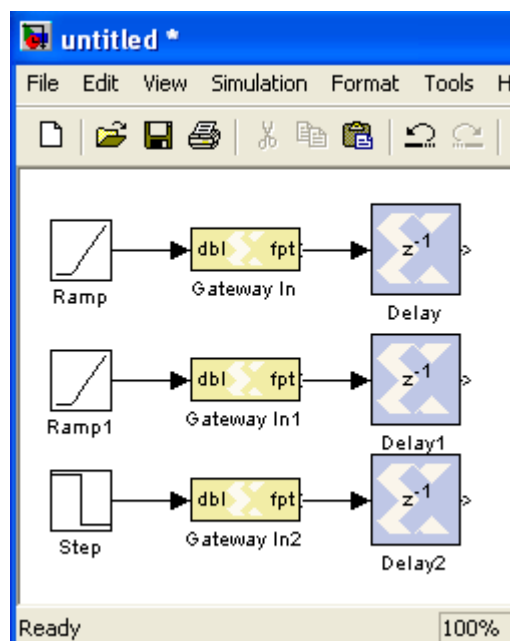


Figura 2-18. Añadir registros de rendimiento.

- 6 Añadir el **multiplicador** y el **acumulador** de la biblioteca de **Matemáticas** Xilinx y alambre para arriba.

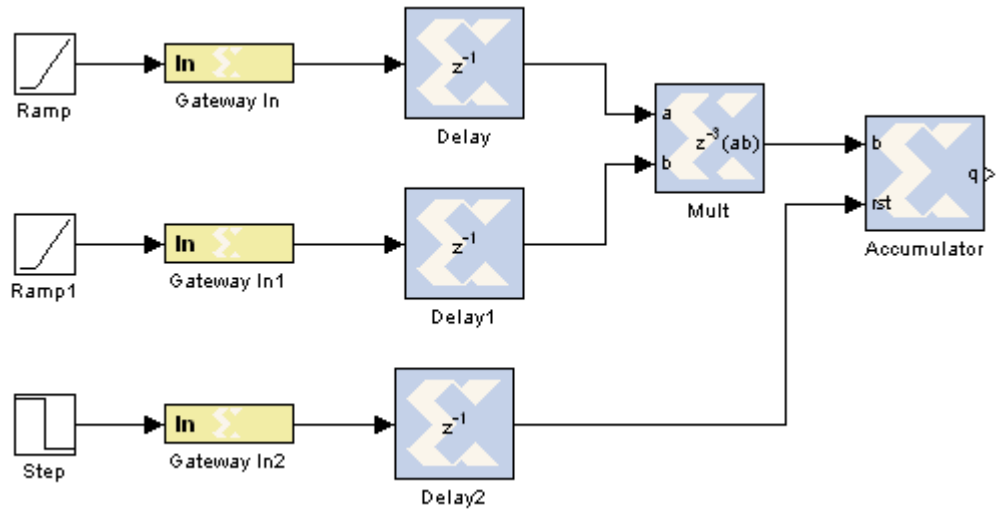


Figura 2-19. Añadir el multiplicador y el acumulador.

- 7 Añadir la **demora** el bloque de salida y conectarlo a la salida del acumulador.

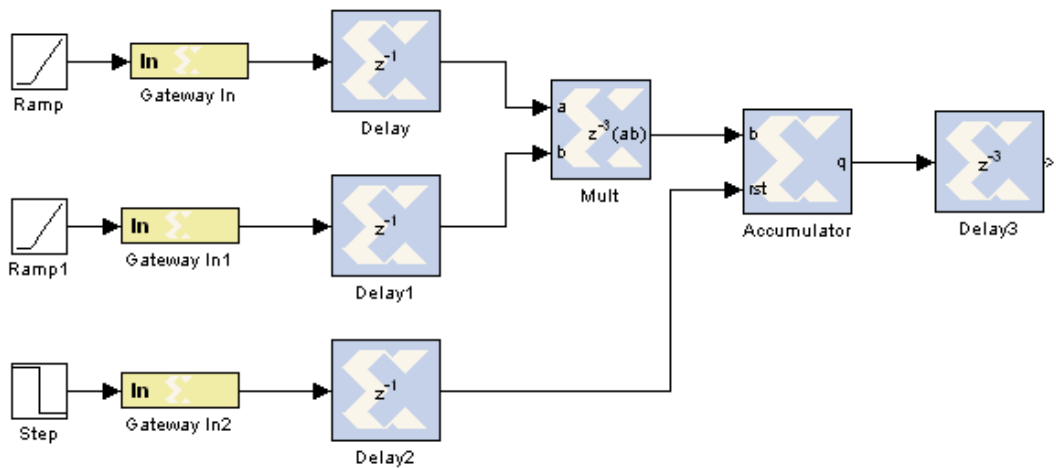


Figura 2-20. Añadir Salida de retardo para el rendimiento

- 8 Añadir una **puerta de entrada a bloque** y conéctelo al bloque de **retardo**.

- Añadir **Terrestres** y conectar a la puerta de entrada a la rampa y los insumos para que puedan controlar tanto la entrada como Output.

El diagrama final debería ser similar a la figura de abajo.

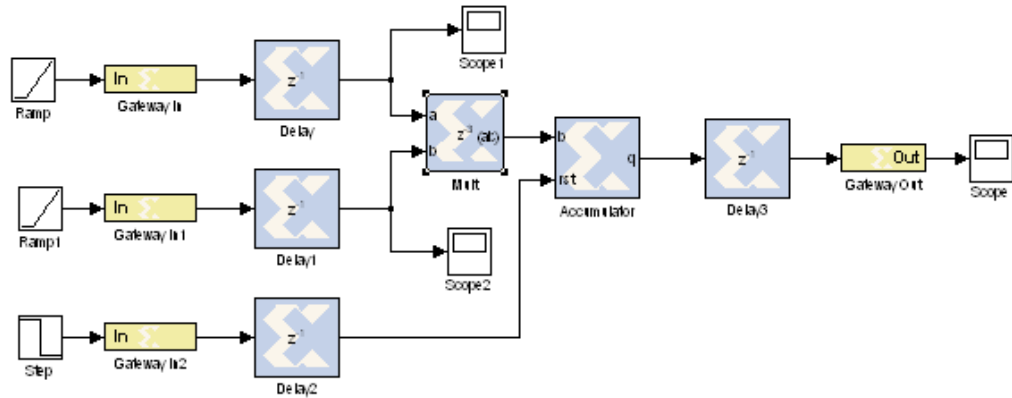


Figura 2-21. Añadir Gateway Out y alcances

A.7.2.4 CONFIGURAR Y SIMULAR EL 12 X 8 PASO 4 MAC

Configure las entradas y los parámetros de simulación como se especifica a continuación. A continuación, simular los 12 x 8 MAC en Simulink y comprobar la funcionalidad del diseño

- **12-bit de entrada:** los datos firmados, punto binario 0, y toma de muestras entre el 1 de
- **8-bit de entrada:** los datos firmados, punto binario 0, y el período de muestreo 1
- **3^a de entrada:** tipo Boolean
- **Gateway Out:** "Traducir al puerto de salida" casilla marcada
- **Multiplicador:** Latencia fija en 3
- **Acumulador:** anchura de salida de 27 bits, el método de desbordamiento para envolver
- **Parámetros de simulación:** Detener el tiempo a 2500

- Configurar la primera puerta de enlace como en 12-bits, punto binario 0, tipo de datos firmados, y toma de muestras entre el 1 de

- ② Configurar la segunda puerta de entrada como en **8-bits, punto binario 0, tipo de datos firmados, y toma de muestras de 1**
- ③ Establezca la tercera puerta en el tipo de datos **Boolean**
- ④ Configurar el bloqueo de multiplicador para **utilizar multiplicadores Embedded** (en la pestaña Aplicación)
- ⑤ Configurar el **número** de acumuladores **de bits** a **27**, y establecer el método **de desbordamiento para** envolver
- ⑥ Abra el cuadro de diálogo **Parámetros de configuración** y establecer el **tiempo de parada** a **2500**
- ⑦ Añadir un **sistema generador** de **señal** para el diseño.
- ⑨ Ejecutar la simulación y analizar los resultados

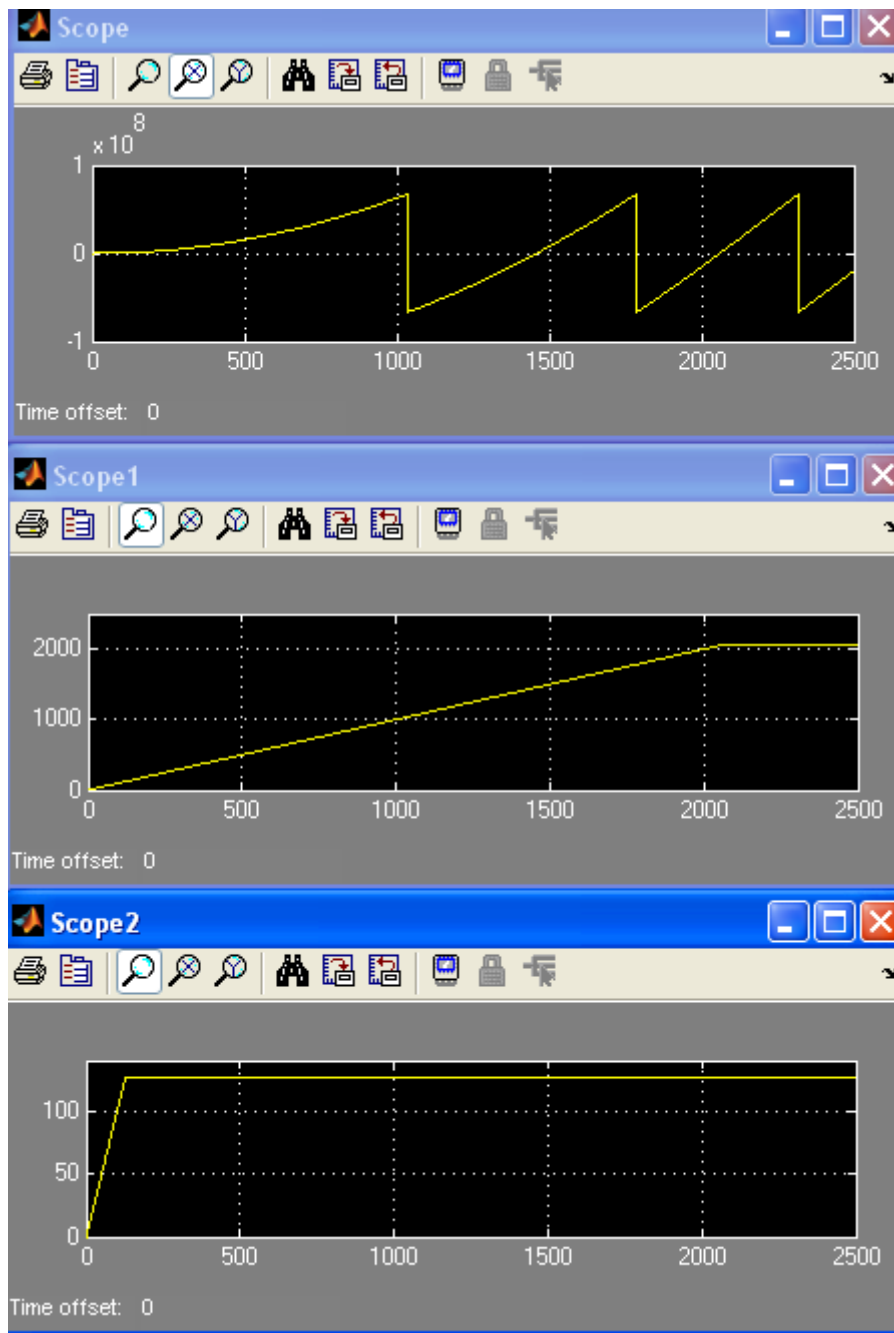


Figura 2-22. Salida de Simulink con multiplicador máximo Habiendo Canalización

- ⑨ Zoom en torno a la primera transición de la producción y determinar dónde se produce

1. ¿En qué momento de la primera transición (en punto) se produce?

Cambiar la latencia bloque multiplicador de 2 y volver a simular. Compare los resultados y guardar el diseño como mac.mdl

- ❶ Abrir el multiplicador's **Block** cuadro de diálogo **Parámetros**, y cambiar la **latencia** a **2**
- ❷ Haga clic en **Aceptar**
- ❸ Ejecutar la simulación

2. ¿En qué momento de la primera transición (en punto) se presenta ahora?

- ❹ Guarde el diseño con el nombre de **Mac**

A.7.2.5 LAS ESTIMACIONES DE RECURSOS PASO 6

Estimación de los recursos utilizados por el diseño usando el bloque de **Recursos Estimador** con bloqueo de **multiplicador** aplicado en **LUT / Multiplicadores** con una **latencia** establece en **2** y **3** para el objetivo siguiente.

- ❶ Haga doble clic en el **Sistema Generador** de señal y seleccione **HDL netlist** para la **compilación** y establecer la **parte** relacionada con los campos como:
 - **Compilación:** **HDL netlist**
 - **Parte:** xc3s500e Spartan3e-4fg320
- ❷ Asegúrese de que **la latencia de** bloque multiplicador se establece en **2**, **el uso integrado multiplicadores** está **desactivada**
- ❸ Agregar el bloque de **Recursos Estimador** de la colección de **herramientas de Xilinx**
- ❹ Abrir el bloque de **Recursos Estimador**.

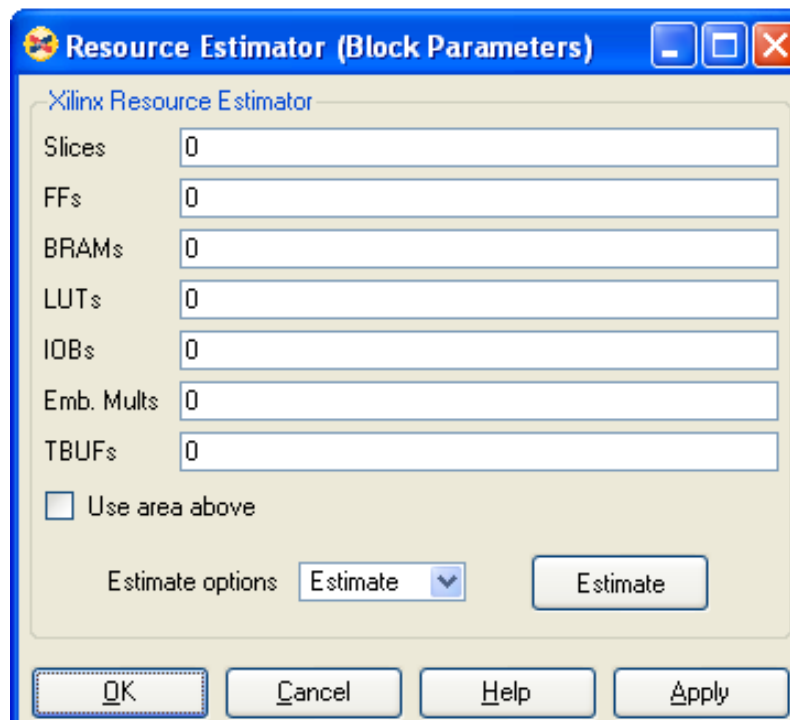


Figura 2-23. Recursos Estimator Block.

- 5 Con la **estimación** de seleccionados de la lista desplegable, haga clic en el botón **Estimación**

3. ¿Cuál es la estimación de recursos para el diseño utilizando el **multiplicador LUT** utilizando a cabo y la latencia de 2?

- Número de Slices __
- Número de flip-flops __
- Número de STUI __
- Número de IOBs __

- 6 Cambiar la latencia del bloque **multiplicador** a 3 y estimar los recursos

4. ¿Cuál es la estimación de recursos para el diseño utilizando el **multiplicador LUT** utilizando aplicado y la latencia de 3?

- Número de Slices __
- Número de flip-flops __
- Número de STUI __
- Número de IOBs __



Estimación de los recursos utilizados por el diseño a través de bloquear el **Estimador de recursos** con **incrustado multiplicador** y la **latencia** establece en **2** y **3**

❶ El valor de **latencia** del bloque **multiplicador** de **2** y haga clic en el uso **Embedded Multiplicadores** casilla para seleccionar el multiplicador integrado

❷ Estimación de los recursos, haga clic en el botón de **estimar el área** de los recursos en el cuadro de parámetros de estimador

5. ¿Cuál es la estimación de recursos para el diseño utilizando el multiplicador integrado con una latencia de 2?

- Número de Slices __
- Número de flip-flops __
- Número de STUI __
- Número de multiplicadores __
- Número de IOBs __

❸ Cambiar la latencia del bloque **multiplicador** a **3** en el parámetro de bloqueo de multiplicador y haga clic en el botón **Aplicar**

6. ¿Cuál es la estimación de recursos para el diseño utilizando el multiplicador integrado con una latencia de 3?

- Número de Slices __
- Número de flip-flops __
- Número de STUI __
- Número de multiplicadores __
- Número de IOBs __

A.7.2.6 GENERAR EL PASO HDL CÓDIGO 6



Asegúrese de que la latencia del multiplicador se establece en 2 y el uso de multiplicadores incorporados está marcada. Generar el código VHDL y analizar el diseño aplicado mediante los informes generados en la Fundación ISE.

❶ Asegúrese de que la latencia del bloque **multiplicador** se establece en **2** y el uso de **multiplicadores incorporados** está **marcada**.

❷ Haga doble clic en el icono **Sistema Generador** y especificar la configuración de la siguiente:

- **Compilación: HDL netlist**
- **Parte: xc3s500e Spartan3e-4fg320**
- **Síntesis de la herramienta: XST**
- **Lenguaje de descripción de hardware: VHDL**

- Directorio de **destino:** / ISE
- Crear **Testbench: sin control**
- El reloj del sistema FPGA Período (ns): **20**

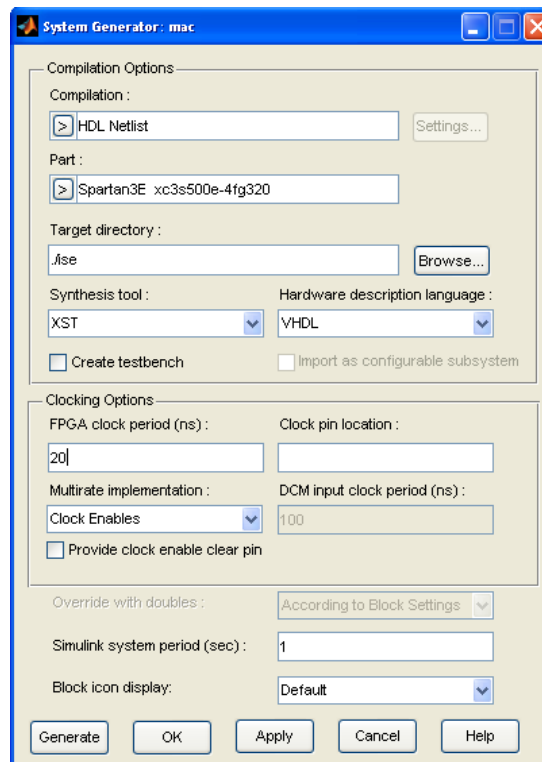


Figura 2-24. Sistema Generador de Parámetros para la Generación de código VHDL.

- ③ Haga clic en **Generar** para generar el código HDL y ISE Los archivos de proyectos
- ④ Seleccione **Inicio** → **Programas** → **Xilinx ISE Design Suite 10.1 ISE** → **Project Navigator**
- ⑤ Abra el proyecto generado por la selección **de archivos** → **Open Project** y **mac_cw.ise** seleccionar en el directorio del proyecto **ISE**
- ⑥ Resalta el alto nivel de archivo, llamado **mac_cw.vhd**, y haga doble clic en **Aplicar diseño**.

7. Abra el informe Mapa de archivo y rellene la siguiente información.

Número de cortes: __

Número de registros: __

Número de multiplicadores integrados: __

8. Abra la post-Lugar y Tiempo informe de ruta y completar la siguiente información

Máxima frecuencia de reloj: __

A.7.2.7 CONCLUSIÓN

En este laboratorio, que aprendió el caudal de diseño básicos que intervienen en la incorporación del Sistema Generador de Blockset en Simulink. ¿Ha verificado el diseño del uso del simulador de MATLAB a través del entorno Simulink y corrió el Sistema Generador de señal para generar el código VHDL y los núcleos generadores Xilinx CORE. Una vez que el diseño fue verificado usando Simulink, se haya podido poner en práctica el diseño, ver cómo se aplica el diseño , y generar el archivo de configuración.

RESPUESTAS

1. ¿En qué momento de la primera transición (en punto) se produce? 1037
2. ¿En qué momento de la primera transición (en punto) se presenta ahora? 1036
3. ¿Cuál es la estimación de recursos para el diseño utilizando el multiplicador LUT utilizando a cabo y la latencia de 2?
 - Número de rebanadas 93
 - Número de FF 164
 - Número de STUI 131
 - Número de IOBs 48
4. ¿Cuál es la estimación de recursos para el diseño utilizando el multiplicador LUT utilizando aplicado y la latencia de 3?

- Número de rebanadas 94
- Número de FF 184
- Número de STUI 131
- Número de IOBs 48

5. ¿Cuál es la estimación de recurso para el diseño integrado con el multiplicador y la latencia de 2?

- Número de rebanadas 49
- Número de flip-flops 95
- Número de STUI 47
- Número de Multiplicador 1
- Número de IOBs 48

6. ¿Cuál es la estimación de recursos para el diseño integrado con el multiplicador y la latencia de 3?

- Número de rebanadas 59
- Número de FF 115
- Número de STUI 47
- Número de Multiplicador 1
- Número de IOBs 48

7. Abra el lugar y el informe de ruta del archivo y rellene la siguiente información

- Número de rebanadas ocupados: 57
- Número de registros encontrados: 75
- Número de multiplicadores Embedded: 1

8. Abra la post-Lugar y Tiempo informe de ruta y completar la siguiente información

Frecuencia de reloj máxima: ~ 178 MHz

A.7.3 Práctica 3: Enrutamiento de Señal Targeting Spartan-3E starter kit

ENRUTAMIENTO DE SEÑAL

INTRODUCCIÓN

En esta práctica, usted diseñara y verificara el relleno y no relleno y la lógica de enrutamiento de la señal utilizando un bloque de sistemas generador.

Nota: Hay completos ejemplos en `.\labsolutions\lab3` directory.

OBJETIVOS

Después de completar esta práctica usted será capaz de:

- Demostrar como el enrutamiento de señal de bloque puede usarse
- Diseñar un sub-sistema usando el enrutamiento de señal de bloque.

DESCRIPCIÓN DE DISEÑO

En una típica FIR MAC, los coeficientes y los datos deben ser almacenados en un sistema de memoria. Hay varias opciones de almacenamiento: memoria RAM bloque, distribuidos de RAM, y SRL16E. Tendrá doble uso bloquean el puerto de RAM para almacenar los datos y coeficientes, con los datos sean capturados y leer con un búfer cíclico de datos de RAM, por lo tanto, la RAM se usa en la configuración de modo mixto. Los datos se escriben y se leen desde el puerto de A (modo de RAM), y los coeficientes son de sólo lectura desde el puerto de B (modo de ROM).

El ancho de cada puerto de la memoria RAM de doble puerto negro está determinado por el ancho de entradas respectivas, y los puertos sólo puede ser diferente si los anchos son de 2, 4, 8, 16 o 32 veces mayor con respecto a la otra. Debido a que su anchura deseada (podrás ver en el laboratorio de al lado) no está de acuerdo con esta norma, que tendrán que ser los mismos. Por lo tanto, los datos deben ser manipulados antes y después de la RAM. En la entrada, el relleno de la entrada de datos de 8-bits a 12 bits coincidirá con el ancho del puerto. La salida puede ser reajustado a la original de 8 bits utilizando unpadding. Este concepto se ilustra en la Figura 3-1.

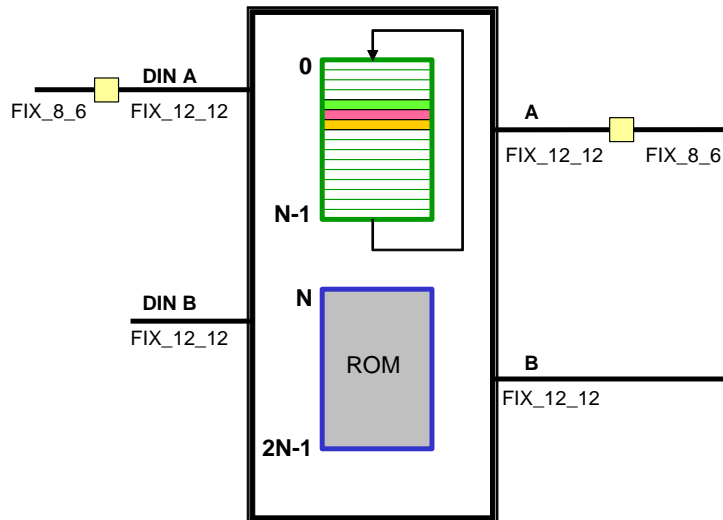


Figura 3-1. Utilizando diferentes puertos anchos de RAM

En esta práctica usted diseñara el RELLENO y no relleno lógico usando un enrutamiento de señal de bloques con un sistema generador.

A.7.3.1 PROCEDIMIENTO

Esta práctica comprende dos pasos primarios.

1. Crear un RELLENO lógico.
2. Crear un no RELLENO lógico

Para cada procedimiento en una etapa primaria, hay instrucciones de carácter general (indicado por el símbolo).

Estas instrucciones generales sólo presentar un esbozo general para realizar el procedimiento. Por debajo de estas instrucciones generales, se encuentra el paso de acompañamiento-por paso y las cifras que muestra más detalles de la realización del procedimiento. Si usted se siente seguro sobre cómo completar un procedimiento, puede saltarse el paso por paso y pasar a la instrucción general siguiente.

Nota: Si usted no puede completar el laboratorio en este momento, usted puede descargar los archivos originales de laboratorio para este módulo en el sitio web del programa de la Universidad de Xilinx en <http://www.xilinx.com/univ>

A.7.3.2 DISEÑO DE RELLENO LOGICO

PASO 1



Arranque MATLAB y cambia el directorio de trabajo a Practica 3. Abra el directorio de SIMUNLINK

- 1 Seleccione **Start** → **Programs** → **MATLAB** → **R2007a** → **MATLAB R2007a** o doble-click sobre el acceso directo Matlab en el desktop si es posible.
- 2 Cambie a Practica 3 directorio: escriba **cd c:/xup/dsp_flow/labs/lab3** en la ventana de comando
- 3 Escriba **simulink** en la ventana de commando de MATLAB o abra el **Simulink Library Browser** haciendo click en el correspondiente botón en la barra de herramientas de MATLAB
- 4 Abra un Nuevo diseño de hoja



Diseño de una lógica de relleno para dar formato a la entrada de datos de FIX_8_6 a FIX_12_12 como independiente con el código fuente constante de Simulink, objetos receptores de pantalla, y cualquier otro elemento necesario de la Blockset Xilinx. Compruebe que la lógica trabaja usando 0,5 porque el valor de entrada constante produce 0,007813 como una salida. Guarde el diseño Como modelo padding.mdl

- 1 Agregue una fuente **constante** de la librería **Simulink para el diseño**
- 2 Agregue un **Display** sink object de la librería **Simulink** para el diseño
- 3 Agregue un **Gateway In** y **Gateway Out** al diseño y prepare **Gateway In** a la salida **FIX_8_6**
- 4 Agregue los **Xilinx blocks** necesarios entre el recién agregado **Gateway In** y **Gateway Out** que son los elementos para el funcionamiento de la función padding



1. Estos bloques serán necesarios convertirlos FIX_8_6 tú UFIX_8_0, y luego de UFIX_12_0, y finalmente a FIX_12_12?

Block(s) needed to convert to UFIX_8_0: _____

Block(s) needed to convert to UFIX_12_0: _____

Block(s) needed to convert to FIX_12_12: _____

- 5 Asignar el **constant input value** to 0.5
- 6 Agregar el Bloque **Sistema Generador**
- 7 Corra la simulación

La pantalla indicara la conversión del valor es 0.007813.

- 8 Una vez satisfecha, cree un **subsystem** of padding logic (no incluye gateways) y nombre el **block** Pad

Nota: Hay una solución en la sección de repuestas si no puede conseguir que funcione (**Figura 3-2**)

Guarde el diseño como **padding.mdl**

DISEÑO DE UN NO RELLENO LOGICO

PASO 2



Abra una nueva hoja de diseño. Diseñe un no RELLENO lógico al formato de entrada de dato de FIX_12_12 a FIX_8_6 como independiente usando el Simulink Constant source, muestre los objetos de fregadero,y cualquier elemento necesario del Xilinx blockset. Verificar que funciona usando 0.0078125 porque la constante del valor de entrada procede 0.5 así como una salida. Grabe el diseño como unpadding.mdl model.

- ❶ Abra una nueva hoja de diseño
- ❷ Agregue un objeto de fuente **Constante** del Simulink library to the design (si es necesario)
- ❸ Agregue un **Display** sink object del Simulink library hacia el diseño (si es necesario)
- ❹ Agregue un **Gateway In** y **Gateway Out** hacia el diseño (si es necesario). Asegúrese que el Gateway In tiene el dato escrito FIX_12_12
- ❺ Agregue los **xilinx blocks** necesarios entre los recién agregados Gateway In y Gateway Out para poder funcionar las funciones no acolchadas



Que bloques serán necesarios convertir FIX_12_12 a UFIX_8_0 y luego a FIX_8_6?

Block(s) needed to convert to UFIX_8_0: _____

Block(s) needed to convert to FIX_8_6: _____

- ❻ Asigne el **constant input value** a 0.0078125
- ❼ Agregue **System Generator** block
- ❽ Corra la simulación

La pantalla deberá indicar el valor convertido 0.5

- ❾ Una vez satisfechas, cree un **subsystem** de no relleno lógico (no incluye los Gateway) ay nombre el **block** UnPad

Nota: Hay una solución en la sección de respuestas si que no logra conseguir que funcione (**Figura 3-3**)

- ❿ Grabe el diseño como **unpadding.mdl** y **salga de Matlab**

A.7.3.3 CONCLUSION

En esta práctica usted aprendió como diseñar un padding relleno y no relleno lógico usando un enrutamiento de señal de bloque de un sistema generador. Usted uso constants, reinterpreto, corto bloques concat.

RESPUESTAS

- Which blocks will be necessary to convert FIX_8_6 to UFIX_8_0, then to UFIX_12_0, and finally to FIX_12_12?

Block(s) needed to convert to UFIX_8_0: Reinterpret
 Block(s) needed to convert to UFIX_12_0: Constant and Concat
 Block(s) needed to convert to FIX_12_12: Reinterpret

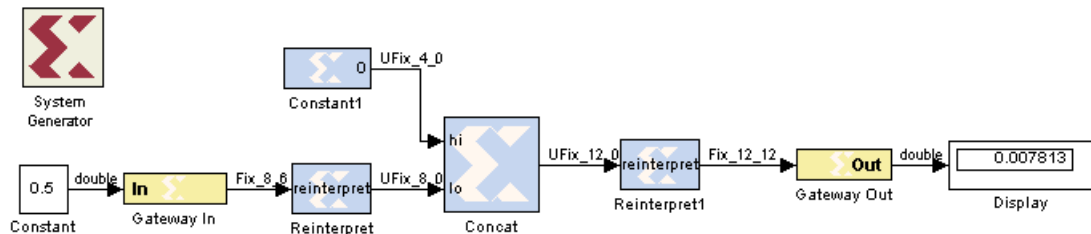


Figura 3-2. Diseño RELLENO

- Which blocks will be necessary to convert FIX_12_12 to UFIX_8_0 and then to FIX_8_6?

Block(s) needed to convert to UFIX_8_0: Slice
 Block(s) needed to convert to FIX_8_6: Reinterpret

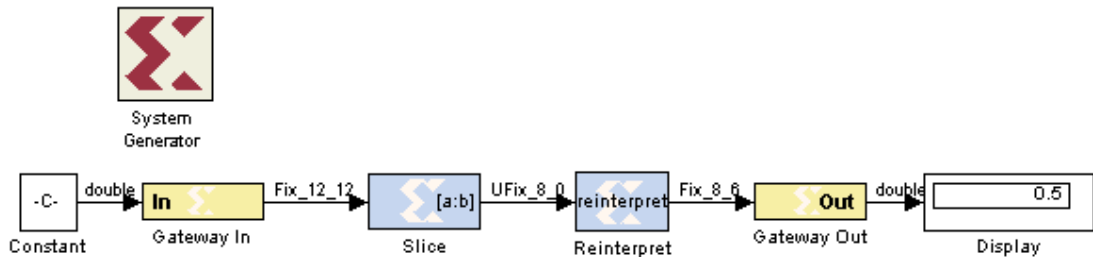


Figura 3-3. Diseño No RELLENO

A.7.4 Práctica 4: Aplicación de un Sistema de Control Targeting Spartan-3E starter kit

APLICACIÓN DE UN SISTEMA DE CONTROL

INTRODUCCIÓN

El sistema generador deduce el reloj de las muestras de tiempo y extrae el Clock Enable (CE). Múltiple bloques dan acceso a posibilitar el reloj y resetear los puertos. El sistema generador también provee diferentes medios para desarrollar el control lógico, como MCode, Expression, Mealy State Machine, Moore State Machine, y Black Box (escribe tu propio código HDL). En esta práctica, usted usara algunos de estos para crear un control lógico para un generador de direcciones.

Nota: Hay ejemplos completos en *labsolutions\lab4* directory.

OBJETIVOS

Después de completar esta práctica usted será capaz de:

- Use el Puerto de relojes permitidos en las herramientas de Simulink
- Describe la lógica detrás del generador de direcciones para un MAC-based FIR filter
- Desarrolle y use un bloque MCode para generar una relación simple de bloque.

DESCRIPCIÓN DE DISEÑO

En esta practica usted creara un generador de direcciones para un 92-tap MAC-based FIR filter usando (i) funciones de bloques predefinidas y (ii) un bloque MCode. Este proceso incluye creando las señales que dirigen las direcciones para la memoria que se mezclan donde residen (*data_addr*); La dirección para la memoria en donde el coeficiente reside (*coef_addr*); y la escritura de las señales permitidas que dictan cuando una nueva mezcla puede ser guardada en la memoria (*we*).

Una manera de implementar este filtro es en localizar los coeficientes del filtro y mezclarlos en un Puerto doble de bloque RAM que es usada como un ciclo de regulador RAM. El puerto doble RAM se usara en un modo mixto de configuración, con los datos escritos y leídos del puerto A (modo RAM) y el coeficiente leído del puerto B (modo ROM). El diagrama de funciones de bloques del diseño se muestra en la **Figura 4-1**.

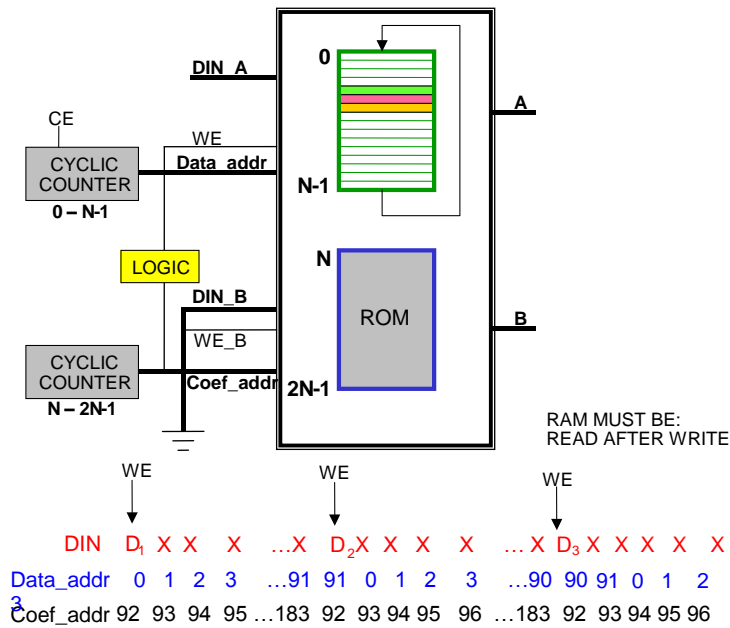
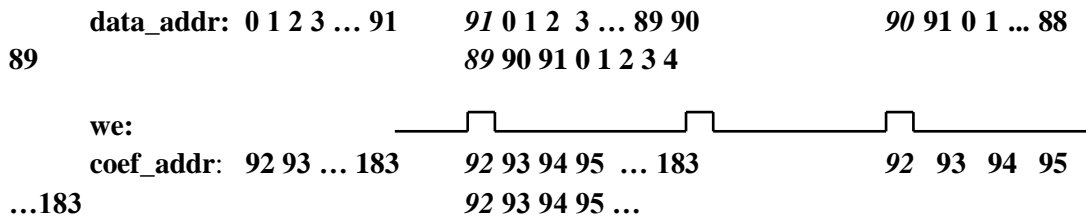


Figura 4-1. Descripción de diseño

Los datos de secuencia de dirección (data_addr) deberán contra para arriba de 0 a 91 y estarán puestos de un ciclo de reloj cada 92 ciclos de reloj:



Los datos que permiten escribir las señales permitidas (we) deberán activarse cada 92 ciclos de reloj, como se muestra arriba. Note que su duración es alta durante el Segundo ciclo de reloj de dos relojes durante. Con el mismo data_addr

A.7.4.1 PROCEDIMIENTO

Esta práctica comprende dos pasos. Usted usara el sistema generador de bloques para crear un generador de direcciones y escribir señales disponibles, como se explica en la sección de descripción de diseños. En paso 1 usted usara los bloques predefinidos funcionales. En paso 2 usted reemplazara la relación de bloques funcionales MCode , para lo cual escribirá una función.

Para cada procedimiento dentro de un primer paso, hay instrucciones generales (indicado por el símbolo).

Estas instrucciones generales sólo dan una idea general para llevar a cabo el procedimiento.

Debajo de estas Instrucciones Generales usted encontrara Direcciones de paso a paso y figuras Ilustradas. Que le darán mayores detalles el párrafo a desarrollar el Procedimiento. Si Usted se Siente Capaz de completar El Procedimiento puede saltar estas Instrucciones Paso a Paso. Y de una dirigirse directamente a la Instrucción general Siguiente.

Nota: Si no es capaz de completar la práctica para este momento, usted puede descargar los archivos originales de las prácticas para este modulo de <http://www.xilinx.com/univ>.

A.7.4.2 CREANDO EL DISEÑO USANDO BLOQUES

PASO 1




Abrir el modelo `counter_enabled.mdl` en la herramienta MATLAB desde el directorio `lab4`, agregue los bloques necesarios para generar la dirección de lectura coeficiente (`coef_addr`), datos modo escritura (nosotros), y la dirección de datos (`data_addr`) señales, como se describe en la especificación del diseño sección. Simular el diseño de 200 unidades y verificar la funcionalidad.

- ❶ Seleccione **Start** → **Programs** → **MATLAB** → **R2007a** → **MATLAB R2007a** o doble click en el acceso directo en de Matlab en el escritorio. Si esta disponible
- ❷ Cambie hacia el directorio `lab4` : type `cd c:\xup\dsp_flow\labs\lab4` en la ventana de comando
- ❸ Abra el modelo *counter_enabled.mdl en la ventana de la consola del* MATLAB
- ❹ De los parámetros del bloque **coef_counter** como se debe, dejando el resto de parámetros como están:
 - **Number of Bits:** 8
 - **Output Type:** Sin asignar
 - **Initial Value:** 92
 - **Count To Value:** 183
- ❺ De los parámetros del bloque **data_counter** a seguir, dejando el resto de parámetros como se encuentran:
 - **Number of Bits:** 8
 - **Output Type:** Sin asignar
 - **Initial Value:** 0

- **Count To Value:** 91
 - **Provide Enable Port:** Chequeado
- ⑥ Añada el **control logic** que manejar el puerto en del contador de datos para generar la siguiente secuencia de datos de dirección:

$$\begin{array}{cccccccccccccccc} 0 & 1 & 2 & 3 & \dots & 90 & & 91 & 91 & 0 & 1 & 2 & 3 & \dots & 89 & 90 & 90 & 91 & 0 & 1 & 2 & 3 & \dots & 89 & 89 & 90 & 91 \\ \hline & & & & & & \text{En} & \end{array}$$

- ⑦ Añada la **logic** necesaria para generar la salida *we* con el siguiente comportamiento:

we: 

coef_addr: 92 93 182 183 92 93 94 ...182 183 92 93 ...182 183 92 93 94 95 ...

- ⑧ Simule el diseño para **200** y verifique que la salida es similar a la figura que se muestra debajo

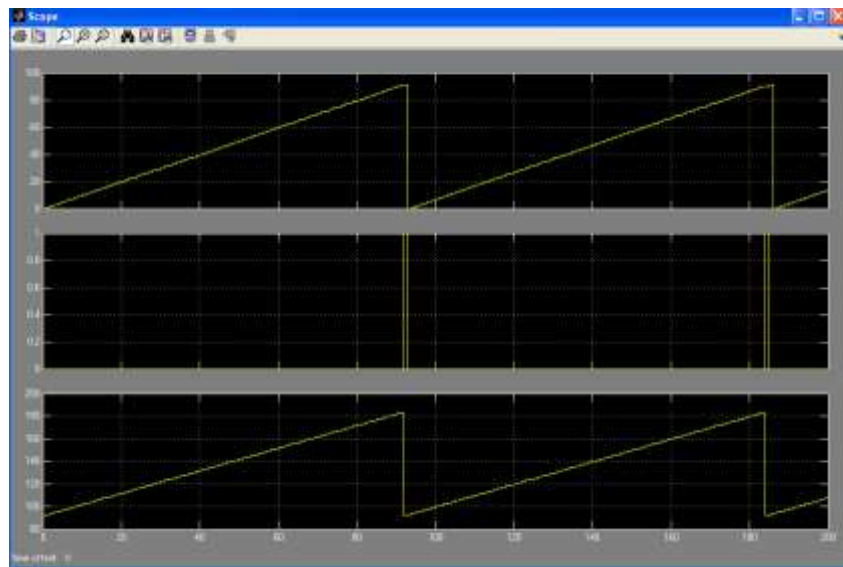


Figura 4-2. Simulación resultante del control del sistema

- ⑨ Guarde el **modelo**

DESARROLLANDO UN MODELO MCODE

PASO 2



Abra el *counter_enabled_mcode.mdl* en MATLAB tool del *lab4* en el directorio, añada el bloque MCode , y escriba una función MCode para generar la señal *we* , como se describe en la sección de especificaciones de diseño. Simule el diseño para 200 unidades y verifique su funcionalidad.

- ❶ Abra el *counter_enabled_mcode.mdl* en la ventana de consola del MATLAB
- ❷ Añada el bloque MCode en el diseño de **Xilinx Blockset** → **Index**
- ❸ Seleccione **File** → **New** → **M-file** en la ventana principal de MATLAB
- ❹ Escriba una función MCode que contara como la entrada y marque como verdadero cuando sea igual a 183, o se marque lo hecho como falso
- ❺ Guarde el archivo como *term_cnt.m*
- ❻ Haga doble click en el bloque MCode, e ingrese *term_cnt* como el nombre de la función en el bloque de llenado de parámetro
- ❼ Complete el resto del diseño, asegúrese de añadir un registro a la salida del bloque MCode
- ❽ Simule el diseño y verifique su funcionalidad
- ❾ Guarde el modelo

A.7.4.3 CONCLUSIÓN

En esta práctica. Usted aprendió a usar el Puerto de clock enable en el ambiente de Simulink y comprendió la lógica detrás del generador de direcciones para un MAC-based FIR Filter. Además aprendió a usar un bloque MCode block y escribir un modelo simple que es sintetizable .

A Respuestas

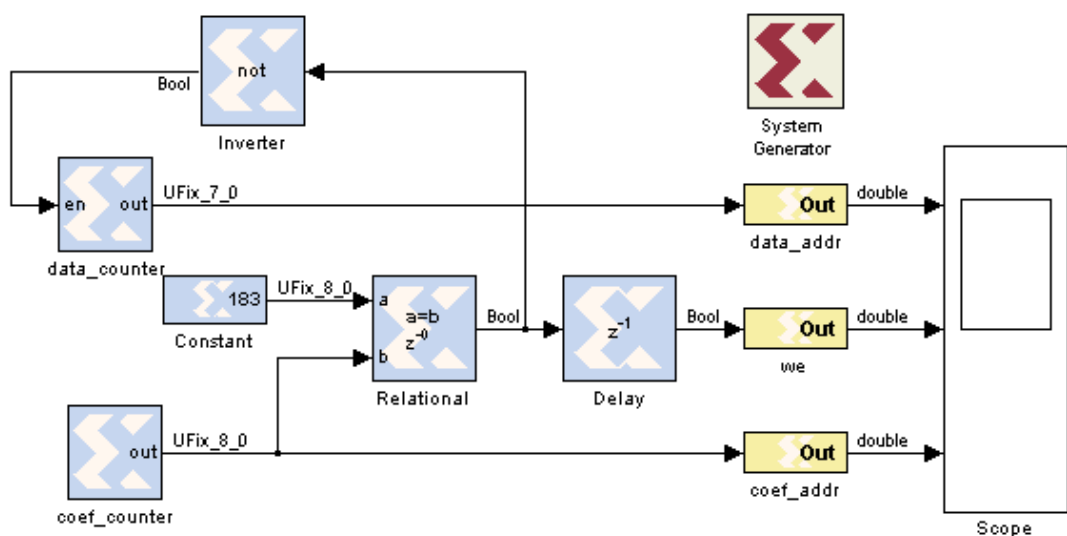


Figura 4-3. Controlando el sistema de solución

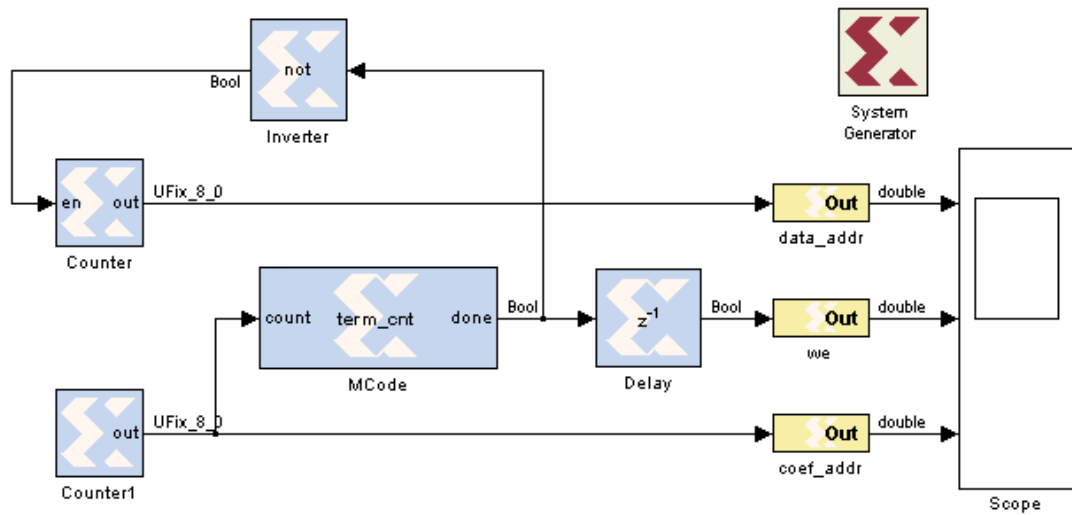


Figura 4-4. Controlando el sistema de solución usando un bloque MCode

Solution MCode

```

function done = term_cnt(count)
if count == 183
    done = true;
else
    done = false;
end

```

A.7.5 Práctica 5:Diseño de un FIR de MAC

Orientación Spartan-3E Starter Kit

DISEÑO DE UNA FIR MAC

INTRODUCCIÓN

En este laboratorio, se creará una base MAC filtro FIR en el Sistema Generador utilizando el motor de MAC que creó en el laboratorio 2, el relleno y la lógica unpadding creó en el Laboratorio 3, y la lógica de control que creó en el Laboratorio 4. Este filtro es un filtro de paso bajo diseñado para eliminar el ruido de alta frecuencia en los sistemas de audio. El propósito de esta práctica es consolidar los conceptos cubiertos hasta ahora, mediante la aplicación de un mayor, la función DSP más complejas que las funciones abordado antes.

Nota: Hay ejemplos terminado en labsolutions \ lab5 \

OBJETIVOS

Después de completar este laboratorio, usted será capaz de:

- Construir un búfer de RAM cíclica
- Utilice el bloque de reciente adhesión en el Sistema Generador de apoyar cualquier tamaño de puerto que se desea
- Comprender cómo utilizar el sistema generador para construir una de las funciones más comunes DSP: la basada en MAC filtro FIR
- Utilizar el alcance del espectro en blanco y la fuente de ruido para ver la respuesta de frecuencia de un filtro de
- Véase el impacto del crecimiento de bits y los bloques disponibles para "seleccionar" los bits que desee

DISEÑO DESCRIPCIÓN

Usted es un diseñador de DSP en Cyberdyne Systems. Su compañía está investigando el uso de filtros digitales en lugar de analógica para sus detectores de Seguridad etiqueta en un intento de mejorar el rendimiento y reducir el costo de todo el sistema. Esto permitirá a la compañía a penetrar aún más el creciente

espacio de mercado de la seguridad. La especificación de la canal, filtro de tipo de interés único (ver Figura 5-1) se especifica a continuación:

- Frecuencia de muestreo (F) = 1,5 MHz
- Fstop 1 = 270 KHz
- Fpass 1 = 300 KHz
- Fpass 2 = 450 kHz
- Fstop 2 = 480 KHz
- De atenuación en ambos lados de la banda de paso = 54 dB
- Rizado en banda pass = 1

Cyberdyne ha optado por ir con FPGAs debido a su flexibilidad, tiempo en el mercado y ventajas de rendimiento sobre los procesadores DSP. Su experiencia en el diseño HDL es limitado y por lo tanto del Sistema Generador para DSP parece ser una solución excelente para la aplicación del filtro en un FPGA, como ya está familiarizado con los productos de The MathWorks.

Su fructífera labor en el proyecto (Laboratorio 3 y Laboratorio 4) ha llevado a su jefe para pedirle que construir un filtro MAC (véase el Gráfico 5-3) de manera que un elemento generador de la biblioteca del sistema se puede construir. Esto proporcionará un acceso fácil a los filtros creados por los ingenieros de diseño de otros para que puedan utilizar la estructura de otros proyectos. También se puede cambiar y adaptarse fácilmente en el Sistema Generador de entorno de diseño de DSP. El filtro debe seguir los parámetros en que los coeficientes se pueden cambiar las especificaciones de filtro diferentes.

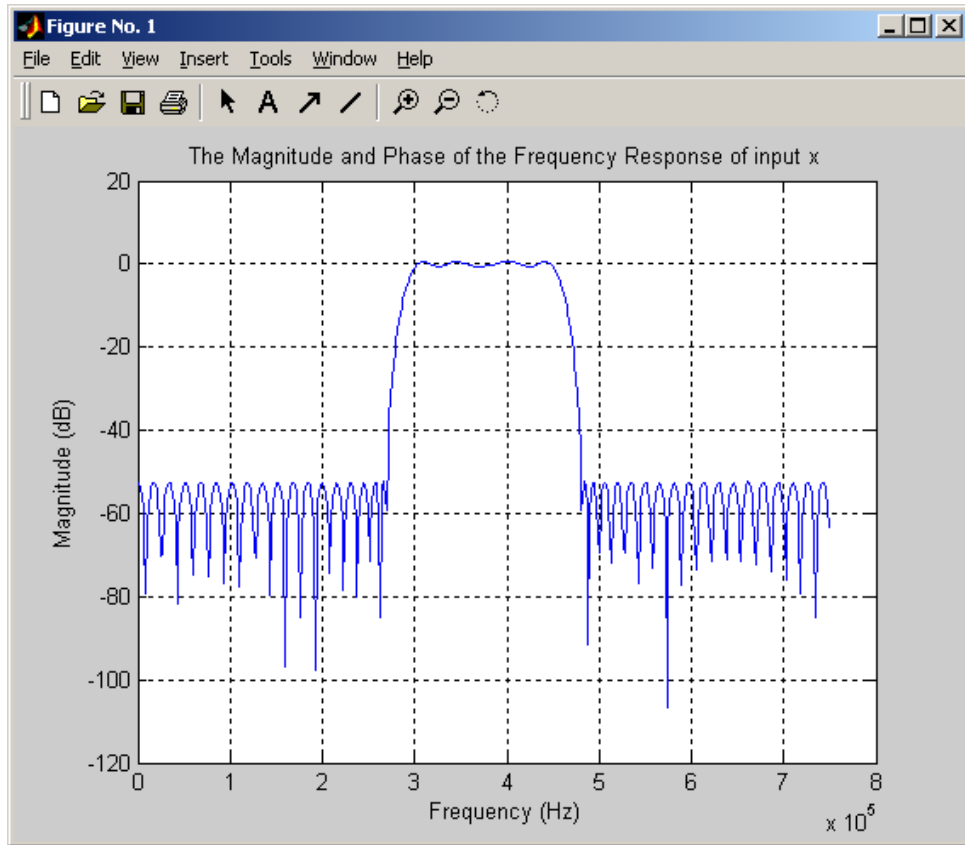


Figura 5-1. Coeficientes de filtro.

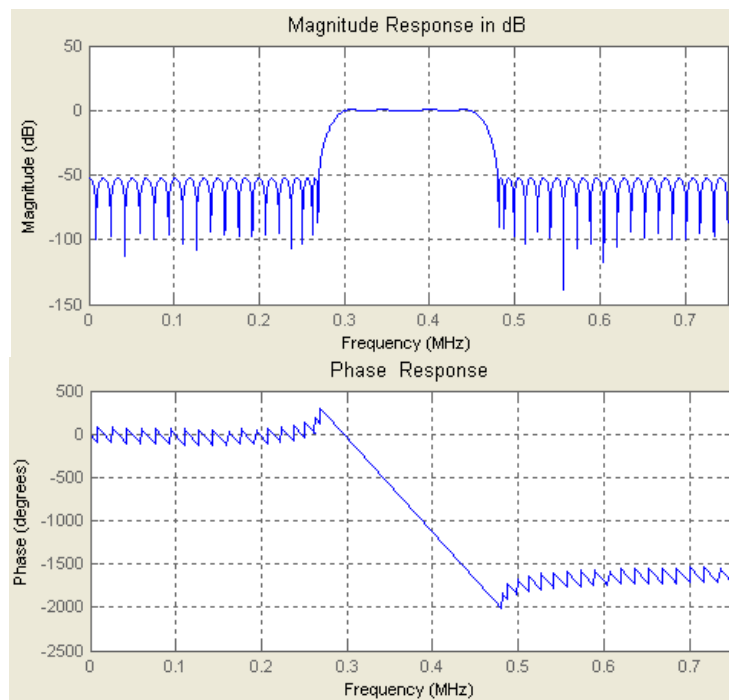


Figura 5-2. Frecuencia de respuesta de los coeficientes de filtro paso banda.

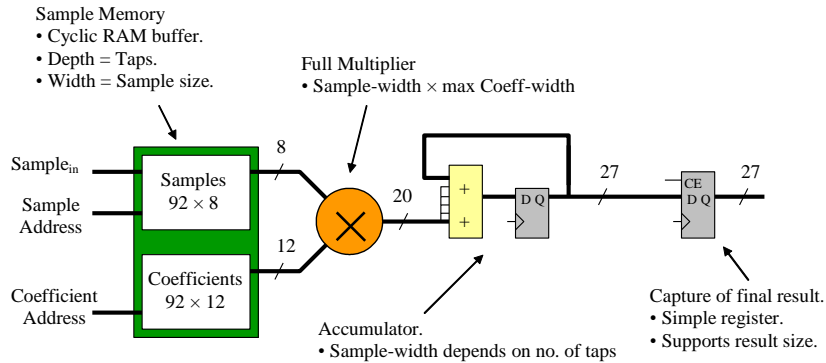


Figura 5-3. La estructura FIR MAC.

Los coeficientes y los datos deben ser almacenados en un sistema de memoria. Hay varias opciones de almacenamiento para almacenar: carnero, distribuidos de RAM y SRL16E. En este laboratorio se utiliza un bloque de memoria de doble puerto de RAM para almacenar los datos y coeficientes, con los datos sean capturados y leer a cabo utilizando un cíclica de datos de búfer de RAM. Por lo tanto, la memoria RAM se utiliza en una configuración de modo mixto. Los datos se escriben y se leen desde el puerto de A (modo de RAM) y los coeficientes son de sólo lectura desde el puerto B (modo ROM). El sistema de memoria completa con el buffer de RAM cíclico debería ser similar a la de la Figura 5-4.

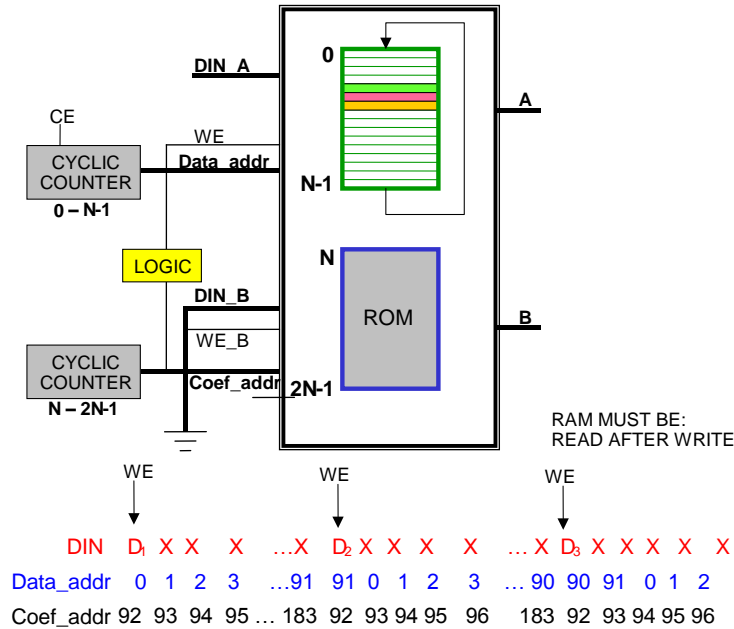


Figura 5-4. Cíclica de búfer de RAM.

A.7.5.1 PROCEDIMIENTO

Este laboratorio consta de siete pasos principales:

1. El análisis de los coeficientes previstos
2. Añadir y parametrizar la lógica de control
3. Añadir la RAM de doble puerto
4. Añadir el relleno y un padding lógica
5. Completar el diseño
6. Probar el diseño con diversas fuentes
7. Realizar hardware-in-the-loop de verificación

Debajo de cada instrucción general para un procedimiento determinado, se encuentra el paso de acompañamiento-por paso y cifras ilustran proporcionar más detalles para la realización de la instrucción general. Si usted se siente confiado acerca de una instrucción específica, no dude en saltarse el paso por paso y pasar a la instrucción general siguiente en el procedimiento.

Nota: Si usted no puede completar el laboratorio en este momento, usted puede descargar los archivos de laboratorio de este módulo desde el sitio del Programa en la Universidad de Xilinx <http://www.xilinx.com/univ>.

ANALIZAR LOS COEFICIENTES PASO 1

Abra el modelo en MATLAB **mac_bandpass.mdl** del directorio **lab5**, analizar los coeficientes de uso de algunos comandos de MATLAB como máximo y mínimo, los coeficientes de ver y entender los parámetros establecidos para las fuentes.

- ❶ Abra la ventana de comandos de MATLAB haciendo doble clic en el icono de MATLAB en su escritorio, o vaya al menú Inicio Programas R2007a MATLAB MATLAB R2007a.
- ❷ Cambia al directorio lab5: Escriba `cd c: / xup/dsp_flow/labs/lab5 /` en la ventana de comandos
- ❸ Abra el modelo `mac_bandpass.mdl`

Nota: Los coeficientes necesarios para el filtro se cargan en el área de trabajo `coef_` variable en la apertura del archivo, junto con los "Yoes" el período de muestreo variable.

- ❹ Tipee en la línea de comandos de MATLAB para ver los coeficientes .

- 5 Típe `max(coef)` para ver el valor del coeficiente máximo

Típe `min(coef)` para ver el valor del coeficiente mínimo.



1. a especificación requiere de 12-bit de datos para los coeficientes.
¿Cuál es el formato óptimo para los coeficientes de 12-bit de este?

Tomar nota de la matriz de fuentes de entrada ya se ha creado para ti.

Haga doble clic en los interruptores para seleccionar entre las fuentes.

A.7.5.2 AÑADIR Y PARAMETRIZAR EL PASO DE CONTROL LOGIC 2

Añadir la lógica de control que creó en el **laboratorio de 4** a su diseño y parametrizar las variables con **coef**. Establezca el período de la muestra a **TS** como estamos interesados en simular sólo la lógica de control. Simular la lógica para comprobar que sigue funcionando como se esperaba.

- 1 Cambia al directorio **c: / xup/dsp_flow/labs/lab4** y **counter_enabled.mdl** abierto desde la ventana de comandos de MATLAB
- 2 Seleccionar todo, excepto el **Sistema Generador** de señal, copiarlos y pegarlos en la hoja de diseño **mac_bandpass.mdl** Change directory to **lab5**.
- 3 Cierre la ventana de **counter_enabled.mdl**
- 4 Cambie el directorio de nuevo a **C: / xup/dsp_flow/labs/lab5** en la ventana de comandos de MATLAB
- 5 Crear un subsistema de la lógica de control que acaba de pegar en el, con exclusión de Gateway Out y ámbito de aplicación.

Nota: Usted puede seleccionar la lógica de control (no seleccione Puerta de enlace de salida y ámbito de aplicación) y pulse Ctrl-G para crear el subsistema.

- 6 Cambiar el nombre de los conectores de página a los nombres apropiados (por ejemplo; `coef_addr`, `data_addr`, `nosotros`)
- 7 Parametrizar la lógica de control integrado de ambos contadores y constante (**período de muestreo** establecidos para **Ts** por ahora como estamos interesados sólo en la simulación de la lógica de control). La lógica de control estará conectado a una memoria de doble puerto. Tenga en cuenta que el ancho de cada puerto de la memoria está determinada por

el ancho de entrada respectivos, y los puertos sólo puede ser diferente si los anchos son de 2, 4, 8, 16 o 32 veces mayor con respecto a la otra.

Sugerencia: Las funciones de MATLAB pueden utilizar los siguientes parámetros en el bloque para que su diseño flexible de

longitud (x) - devuelve la longitud de la matriz X

log2 (Y) - devuelve el registro de Y entero a la base de 2

ceil (N) - devuelve el menor número entero mayor o igual que el número real N



2. Anote las expresiones para el bloque de data_counter que va a entrar para:

Número de bits: _

Conde de valor: _



3. Anote las expresiones para el bloque de coef_counter que va a entrar para:

Número de bits: _

Valor inicial: _

Conde de valor: _

Periodo de muestra: _



4. Escriba las expresiones para el bloque constante de que quieres entrar para:

Valor Constante: _

Número de bits: _

⑧ Añadir el **ámbito** de aplicación a la **salida de puerta de enlace** s, si no están ya presentes

⑧ Set **Stop Time to $T_s * 200$** en la **simulación de parámetros y simular**

Asegúrese de que el solucionador de el resultado de la simulación debe ser similar a la que se muestra en la Figura 5-5

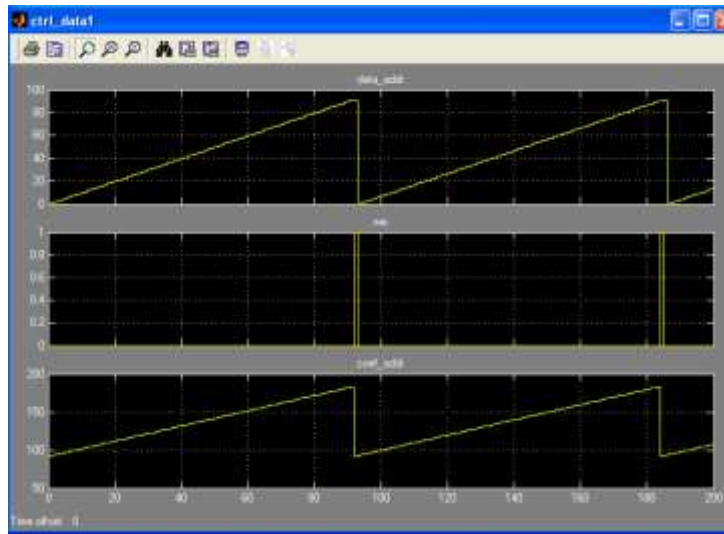


Figura 5-5. Simulación de la Lógica de Control.

A.7.5.3 AGREGUE EL PUERTO DOBLE DE RAM PASO 3

A continuación, agregar la RAM de doble puerto para su sistema y el cable con el bloque de la lógica de control, desarrollado en el laboratorio 3, y la lógica adicional necesaria para que sea funcional en modo mixto como se explica en la sección de descripción de diseño. Añadir constante (valor = 0) al puerto de entrada de datos B. Simular el diseño parcial para ver si obtiene el resultado que se espera de ambos puertos.

- ❶ Añadir elemento de **RAM de doble puerto** de la biblioteca de la **Memoria** del Blockset de **Sistema Generador**
- ❷ Añadir **Boolean** constante con valor **0** para **que el puerto** de entrada de **B**
- ❸ Añadir **firmado** constante de valor **0** (Número de bits = 12 bits y la posición del punto binario = 12) a los **datos** de entrada del **puerto B** y el **puerto de A**
- ❹ Conectar los puertos de entrada apropiado (addra, WEA, addrb) de la RAM para el subsistema de control de salida
- ❺ El programa de instalación del **Vector de valor inicial** para contener el contenido deseado de la RAM en los parámetros de bloque de doble puerto de RAM

Sugerencia: La utilidad las siguientes funciones de MATLAB ejemplos pueden ser útiles para realizar la tarea

Ceros (1, N) - devuelve una matriz de elemento N de ceros

[xy] - devuelve una matriz que es x-y concatenado a

x' - devuelve la matriz x en una forma transpuesta

```
>> x = [1 2 3 4]
```

```
x = 1 2 3 4
```

```
>> x = x'
```

```
x =
```

```
1
```

```
2
```

```
3
```

```
4
```

5. Escriba las expresiones de la memoria RAM de doble puerto que vais a entrar para:

Profundidad: _____

Vector de valor inicial: _____

- ⑥ Establezca el **tipo de bloque de memoria RAM** y escribir **los modos de leer después de escribir** para los dos puertos A y B
- ⑦ Añadir **Puerta de salida** a la A portuarios y B de la memoria RAM y las conectan a un ámbito de aplicación.
- ⑧ Añadir **firmado** constante de valor **0** (Número de bits = 12 bits y la posición del punto binario = 11) a **los datos de entrada del puerto de A y B del puerto** con fines de prueba

Nota: Usted debe tener el diseño de cableado en este punto, como se muestra en la **Figura 5-6**.

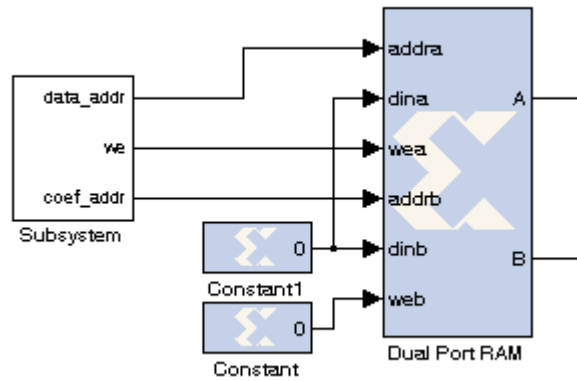


Figura 5-6. Lógica de control conectado a la memoria RAM Dual-Port.

- ③ Establezca el tiempo de parada de $T_s * 100$ en los parámetros de simulación y simular el diseño

El resultado de la simulación debe ser similar a la mostrada en la Figura 5-7

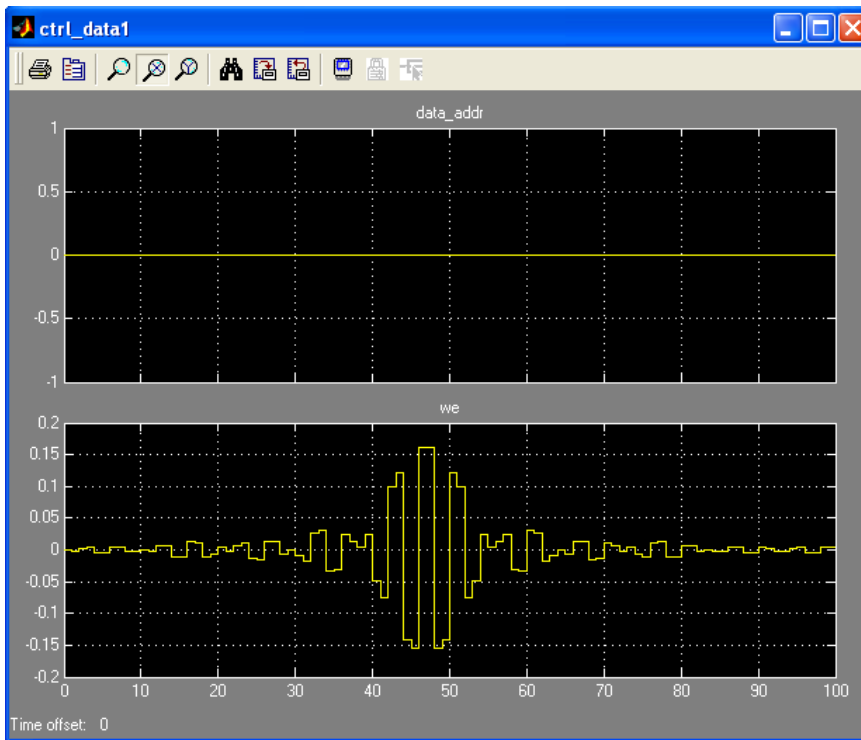


Figura 5-7. Simulación de la memoria RAM Dual-Port y Lógica de Control.

A.7.5.4 AÑADIR RELLENO Y UN PADDING CON EL PASO DE LOS PUERTOS DE DATOS 4

Nota: el ancho de cada puerto de salida de la memoria de doble puerto está determinada por el ancho de entrada respectivos, y los puertos sólo puede ser diferente si los anchos son de 2, 4, 8, 16 o 32 veces mayor con respecto a los demás. Como nuestro ancho deseado no está de acuerdo con esta norma, tendrán que ser los mismos. Por lo tanto los datos deben ser manipuladas antes y después de la RAM. En la entrada, el relleno de la entrada de datos de 8-bits a 12 bits coincidirá con el puerto anchuras. La salida puede ser reajustado a la original de 8 bits usando unpadding. Este concepto se ilustra en la Figura 5-8.

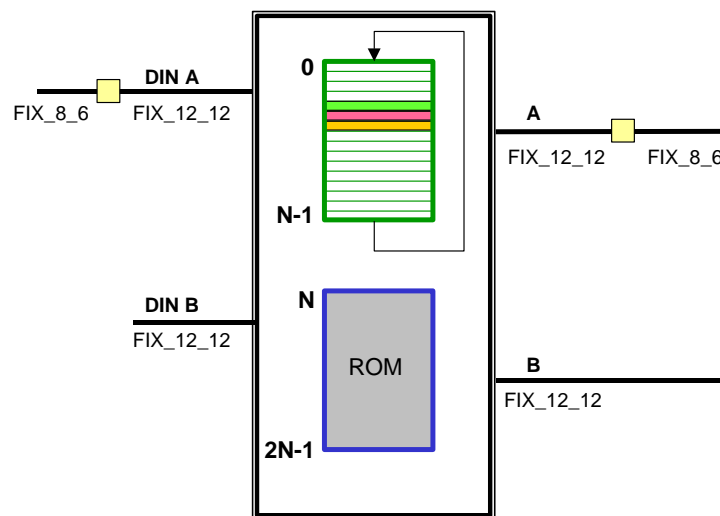


Figura 5-8. Utilización de diferentes anchos de RAM Puerto.



Añadir la lógica de relleno del laboratorio de 3, y conectarlo a la trayectoria de la entrada de un puerto de la RAM de doble puerto.

- 1 Padding.mdl abierta en el laboratorio 3 de directorio
- 2 Copiar y pegar el subsistema de **almohadilla** en la hoja de **mac_bandpass.mdl**
- 3 Conecte la salida del bloque de retardo a la entrada de la sub-sistema **de notas** y la salida del sub-sistema **de notas** a la entrada de un puerto de la memoria RAM de doble puerto



Añadir la lógica unpadding de LAB3, y conectarlo a la ruta de salida de un puerto de la RAM de doble puerto.

- ❶ Unpadding.mdl abierta en el laboratorio 3 de directorio
- ❷ Copiar y pegar el subsistema **unpad** en la hoja de **mac_bandpass.mdl**.
- ❸ Coloque el sub **Unpad** subsistema en la ruta de salida de un puerto de la memoria RAM de doble puerto



Eliminar bloques innecesarios de la hoja de diseño. **Paso** Seleccione la fuente y establezca el tiempo de ejecución de simulación para **Ts * 100**. Ejecutar la simulación y compruebe que la salida es similar a la mostrada en la Figura 5-9

- ❶ Conectar **Gateway Out** y **ámbito** de aplicación a los productos de doble puerto de salida de la memoria.
- ❷ Conecte la salida de datos del elemento de **retardo** a la entrada de la lógica de relleno.
- ❸ Haga doble clic para seleccionar **switch1 Paso** fuente
- ❹ Set **Stop Time to Ts * 100** en los **parámetros de simulación**
- ❺ Ejecutar la simulación y ver la salida

Nota: El resultado de la simulación debe ser similar a como se muestra en la **Figura 5-11**.

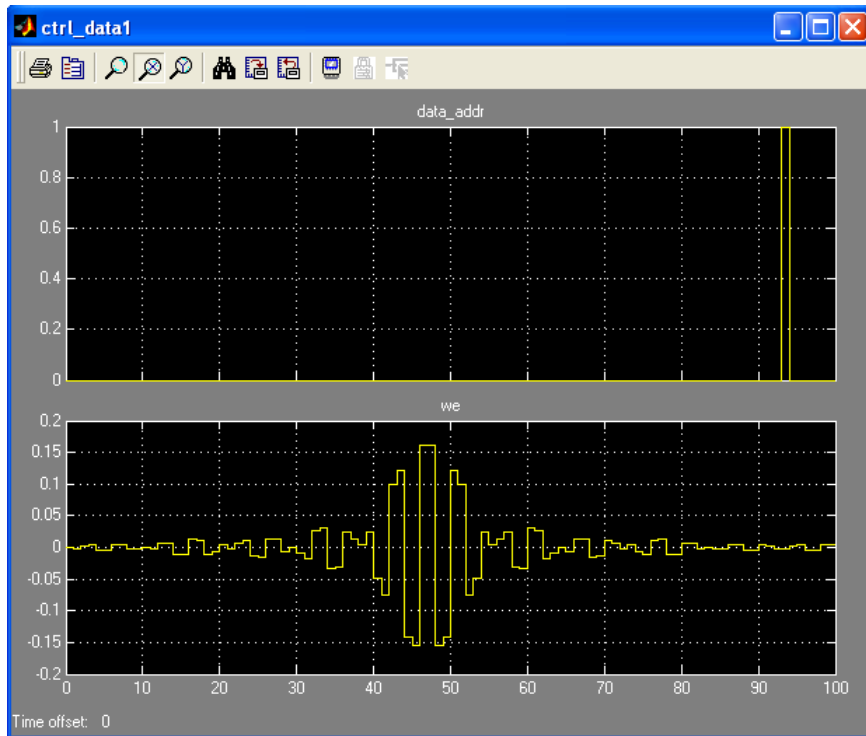


Figura 5-11. Salida de la Dual-Port RAM cuando la fuente es el paso de la Fed.

A.7.5.5 COMPLETAR EL PASO DE DISEÑO 5



Añadir un elemento **de la muestra hasta** entre la entrada y la lógica de relleno. **Set-up** de la **muestra** 's de velocidad a la **longitud (coeficiente)**. Simular el diseño y comprobar que funciona hasta el momento.

- ❶ Agregue el elemento **de muestra entre** la entrada de datos y la lógica de relleno

Nota: Aunque la FIR es un sistema único de tasa (frecuencia de muestreo de salida = tasa de muestreo en), la tasa interna es mucho más rápido debido a la naturaleza de serie de un FIR de MAC, de hecho N (número de coeficientes) veces más rápido. Tenga en cuenta que cada entrada para bloquear un Sistema Generador debe tener el mismo período de muestreo, por lo tanto una imagen "**muestreo**" que se producen en la entrada de datos a la FIR de MAC

- ❷ Haga doble clic en el bloque y establecer **Sampling Rate** a la **longitud (coeficiente)**

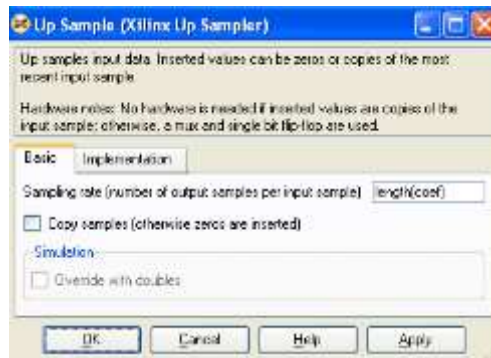


Figura 5-12. Especificar el valor Up Sampler

- ③ Añadir **Gateway Out** en la salida y añadir **ámbito de aplicación** (si es necesario)
- ④ **Switch1** doble clic para seleccionar la fuente de entrada **de impulsos**
- ⑤ Cambiar el período de muestreo de la **coef_counter** del subsistema de control para que coincida con la tasa de muestreo de hasta la entrada de la muestra de datos en el puerto de A



6. ¿Cuál debe ser la expresión para el período de la muestra?

- ⑥ Establezca el tiempo de parada de simulación para simular para **3** muestras a la entrada
- ⑦ Ejecutar la simulación y compruebe que el ciclo de impulso a través del buffer de datos y todos los coeficientes están saliendo como se muestra en la **figura 5-13**

Nota: A partir del período de muestreo ha cambiado, el período de reloj del sistema se debe establecer en **1 / 92 (o TS / longitud (coeficiente))** en lugar de 1. Si el Período de Simulink del sistema está configurado incorrectamente en el sistema de generador de ideas, el error siguiente mensaje puede aparecer.

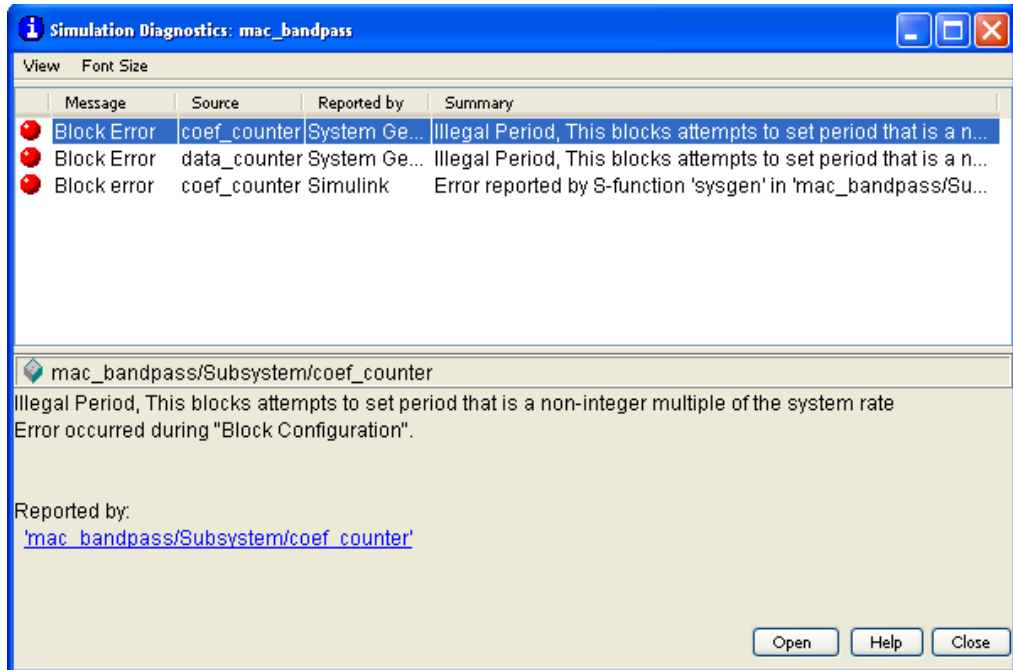


Figura 5-13. El reloj del sistema periodo de actualización.

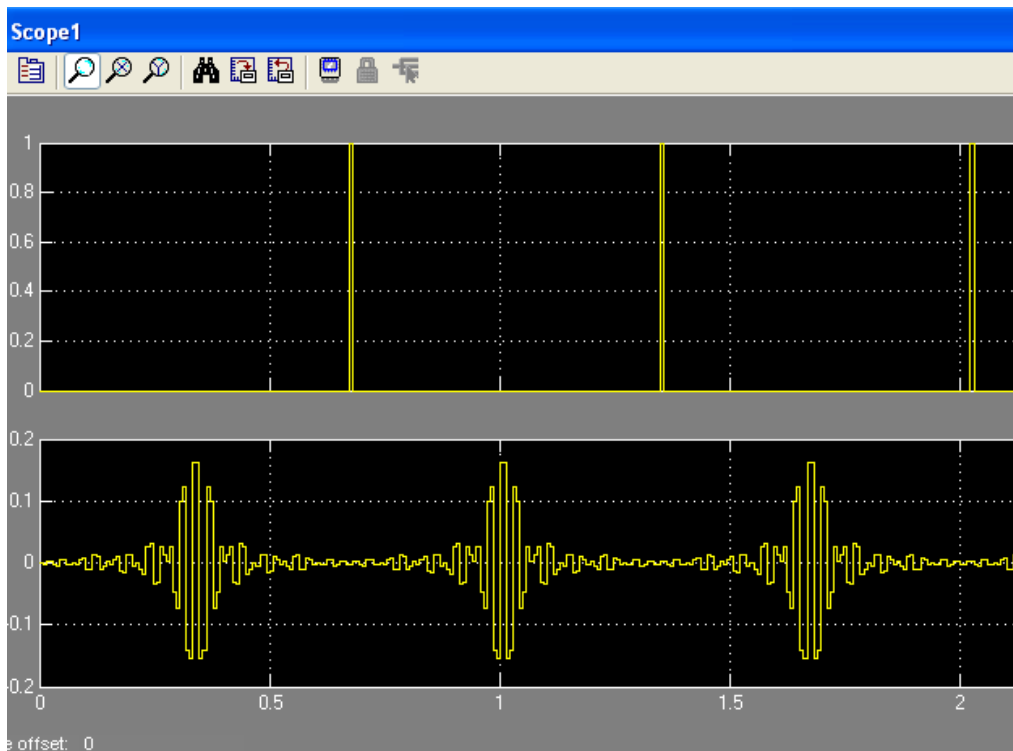


Figura 5-14. Salida de impulsos y coeficientes de salida de la memoria RAM Dual-Port.

- ⑧ Retire la puerta de salida y el alcance



Añadir el **multiplicador** y **acumuladores** (de lab2) a la salida de la lógica unpadding. Establezca los parámetros de bloques del multiplicador y acumuladores para que coincida con la especificación del diseño. Simular el diseño y comprobar que funciona hasta el momento.

- ❶ **Mac.mdl** Abierto de **lab2**
- ❷ Copiar la lógica necesaria y añadirlo a los puertos de salida de la RAM de doble puerto después de la lógica unpadding
- ❸ Asegúrese de que el acumulador tiene su **Proporcionar Perdí Puerto** y **Reinicializar con b de entrada** 'en las **opciones** de restablecimiento **comprobado** en su bloque de parámetros de
- ❹ Haga doble clic en el registro de captura (bloque de retraso) que se conecta a la salida del acumulador, y poner una marca de verificación junto a **Activar Proporcionar puerto**.
- ❺ Asegúrese de que el bloque **multiplicador** ha **incorporado** el **uso de multiplicadores** controlados
- ❻ Set de **latencia** del **multiplicador** a **3**
- ❼ Añadir la lógica necesaria para conectar la entrada de **reset** del **acumulador** y la entrada de registro **CE** de la captura de



7. ¿Qué señal desde el bloque de lógica de control deben estar conectados a la reinicialización del acumulador y la CE del registro de captura?



8. ¿Por qué? ¿Hay alguna lógica extra que se requiere?

- ❸ Crear un sub-sistema del acumulador multiplicador, y la captura de bloques de registro, y el nombre como **MAC**



Agregar el **bloque de abajo muestra** a la salida de la captura de registro seguido por el elemento de retardo para el diseño de alto rendimiento. Simular el diseño y comprobar que funciona para la fuente de impulso.

Guardar la salida al espacio de trabajo y asegurarse de que son capturados todos los coeficientes.

- ❶ Añadir elemento **de la muestra abajo** en la salida del registro de captura de
- ❷ Establezca la **frecuencia de muestreo** por la **longitud (coeficiente)** (longitud de los coeficientes)
- ❸ Añadir un elemento de **retardo** para que la salida de reloj, proporcionando así el diseño de alto rendimiento
- ❹ Juego de simulación de **parada de tiempo a fin de** que de 100 muestras son simuladas
- ❺ Ejecutar la simulación y compruebe que los ciclos de impulso a través del buffer de datos y todos los coeficientes están saliendo como se muestra en la **figura 5-15**

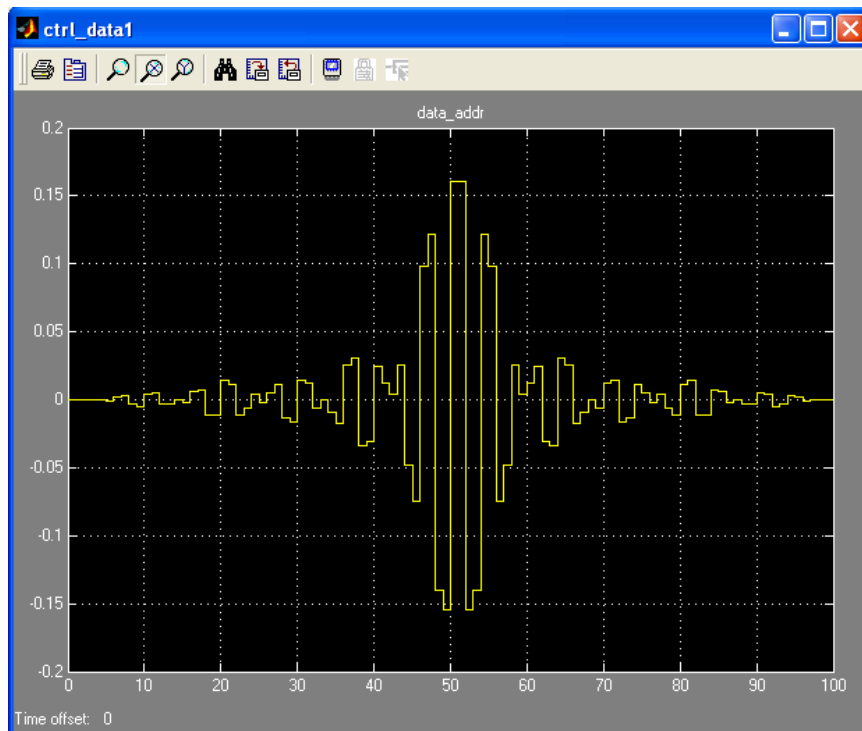


Figura 5-15. Salida de impulsos desde el diseño.

- ❻ Agregar a elemento **de área de trabajo** a la salida del diseño
- ❼ Cambie la propiedad de **formato Guardar** del **área de trabajo** para los elementos de **matriz**

- ⑧ Simular el diseño de nuevo
- ⑧ Ir a la **área de trabajo** en la ventana de comandos de MATLAB, haga doble clic en el **simout1** variable, y desplazarse a través de todas las salidas de asegurarse de que se capturaron todos los coeficientes.

Nota: Puede ver algunos ceros al comienzo de la matriz y al final de la matriz. Sin embargo, entre los ceros todos los coeficientes deben estar presentes.

A.7.5.6 PRUEBAS EN EL DISEÑO CON DISTINTAS FUENTES DE PASO 6



Añadir **ámbito del espectro** y **MUX**, y conectar la fuente de entrada a un canal de MUX y la producción de diseño para la otra entrada del MUX. Seleccione la fuente de **ruido blanco** y simular el diseño después de establecer en el área de trabajo **Ts** MATLAB y tiempo de parada de simulación con el valor apropiado. Compruebe de que el diseño de obras para la fuente de la entrada de ruido blanco. Vea muestras de salida al final de la sección de solución para verificar los resultados esperados.

- ① Añadir **espectro Alcance** de los **fregaderos** de **Signal Processing Blockset** **DSP**
- ② Añadir **MUX** de **Simulink** **señal de enrutamiento** Blockset
- ③ Cable de entrada al canal de la parte superior de **MUX** y la producción de diseño para el canal inferior del MUX

Su diseño debe ser similar a la siguiente

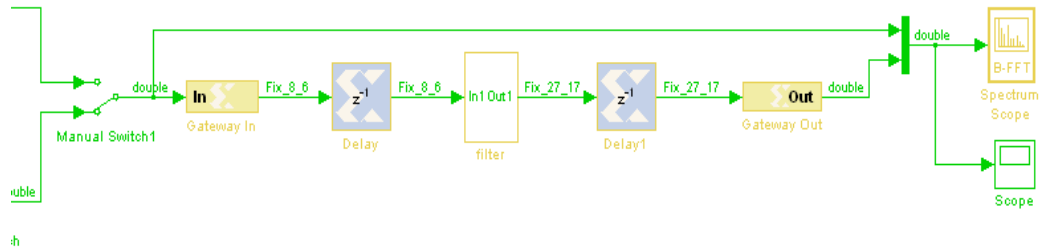


Figura 5-16. Conecte el Ámbito del espectro

- ④ Haga doble clic en el elemento **del espectro** **Ámbito de aplicación** y establecer las propiedades de las siguientes

ÁMBITO DE LAS PROPIEDADES

- **Entrada Buffer: activada**
- **Tamaño de búfer: 256**
- **Buffer se superponen: 64**
- **Tipo de Ventana: Hanning**
- **Especificar la longitud de FFT: 256**
- **Número de promedios espectral: 5**

PROPIEDADES DE PANTALLA

- **Mostrar la rejilla: activada**
- **Número de cuadro: activada**
- **Leyenda Canal: verificado**
- **Abrir el ámbito de aplicación al inicio de la simulación: activada**

PROPIEDADES DEL EJE

- **Unidades de frecuencia: Hertz**
- **Rango de frecuencia: [0 ... F / 2]**
- **Incremento de la muestra Heredar de entrada: activada**
- **Escala de amplitud: dB**
- **Y el límite mínimo: -80**
- **Y el límite máximo: 25**
- **Y-título del eje: Magnitud, dB**

PROPIEDADES DE LÍNEAS

- **Los colores de línea: [1 0 0] | [0 0 1]**

- ⑤ Doble entrada, haga clic en **Cambiar** para seleccionar la fuente de **ruido blanco**
- ⑥ Compruebe **que** el TS se establece en **1 / 150000** en el área de trabajo de Matlab
- ⑦ Configurar el tiempo de parada de **inf** en los parámetros de simulación para la captura de gran número de muestras
- ⑧ Ejecutar la simulación y compruebe el resultado con el que se muestra en la **figura 5-17**

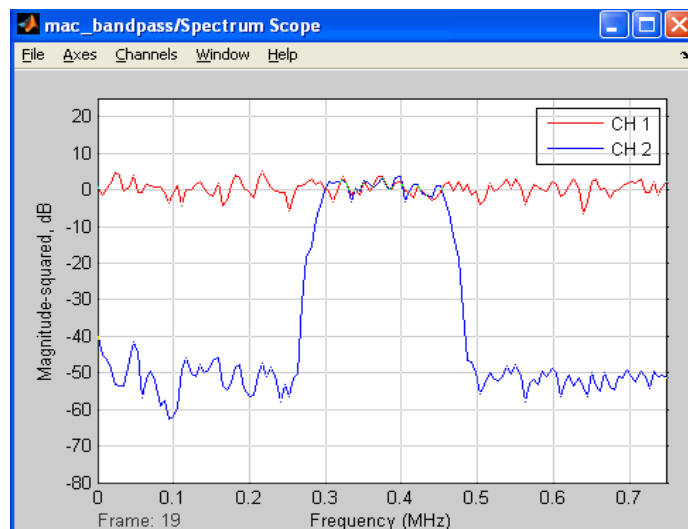


Figura 5-17. Ámbito de aplicación del espectro de salida para la fuente del ruido blanco.



Seleccione la **fuentes de impulso** y simular el diseño. Verificar que el diseño de obras para la fuente de entrada de impulso. Vea muestras de salida al final de la sección de solución para verificar los resultados esperados.

- ① Doble entrada, haga clic en **Cambiar** para seleccionar la entrada **switch1** para seleccionar la entrada **de impulsos**
- ② Añadir un ámbito de aplicación a la salida del diseño
- ③ Ejecutar la simulación para $TS * 100$ muestras y verificar la producción con la que se muestra en la **Figura 5-18**

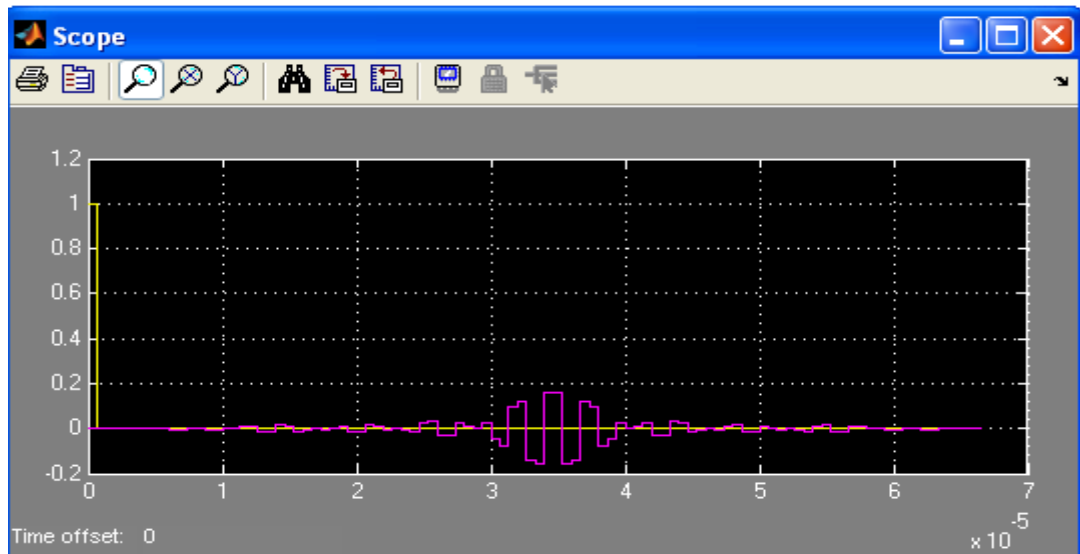


Figura 5-18. Ámbito de aplicación de salida para la fuente de impulso.

A.7.5.7 REALIZACIÓN DE HARDWARE-IN-THE-LOOP DE VERIFICACIÓN PASO 7



Usando el Sistema Generador de modo, generar el hardware y comprobar que el trabajo de diseño a través de la Junta de hardware. Simular el diseño a través de Simulink.

- ❶ Guardar el modelo como **mac_bandpass_hw.mdl**
- ❷ Haga doble clic en el Sistema Generador de señal y ajustar los parámetros de la siguiente
 - Compilación: **Hardware Co-Simulación** xup sp3e_starter_kit
 - Síntesis de la herramienta: **XST**
 - Objetivo del repertorio: **C: / xup/dsp_flow/labs/lab5/hwcosim (o. / Hwcosim)**
 - Crear Testbench: **sin comprobar**

Nota: Asegúrese de que la ventana de bloques del sistema generador muestra Spartan-3E xc3s500e-4fg320 como el dispositivo.

- ❸ Haga clic en el botón Generar y se abre una ventana de comandos que muestran los progresos proceso de compilación, como se muestra en la Figura 5-16

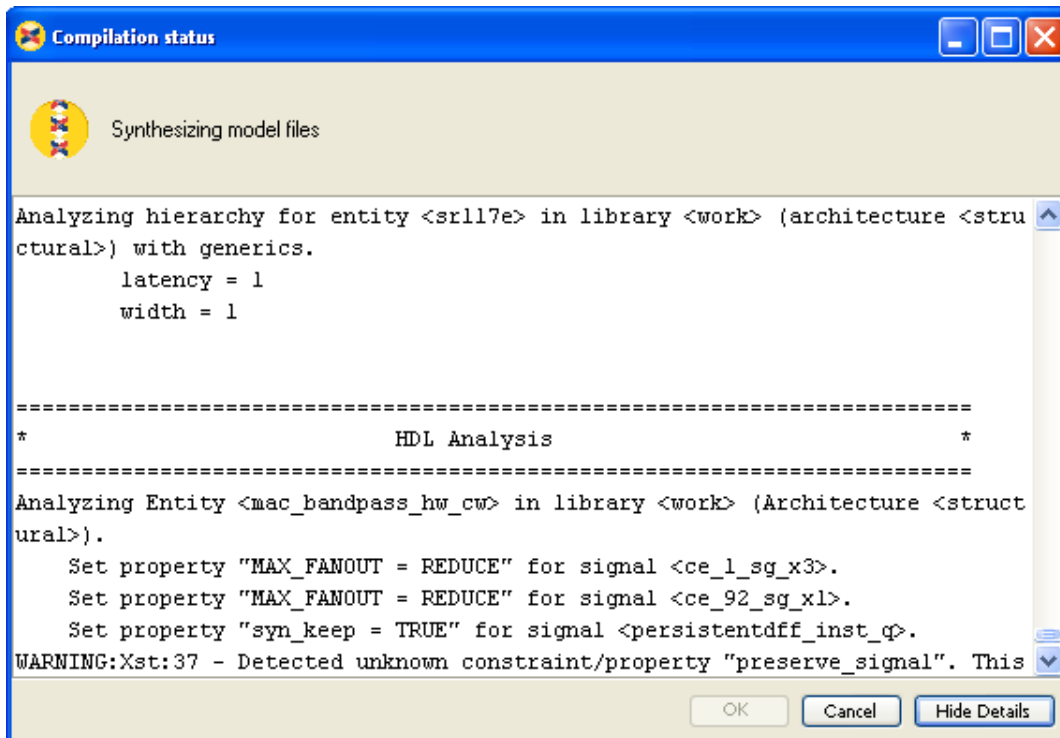


Figura 5-19. Compilación de la ventana de estado

- ④ Cuando la generación se completa con éxito, una nueva ventana de la biblioteca de Simulink se abre y un bloque compilado con el número adecuado de entradas y salidas se mostrará.

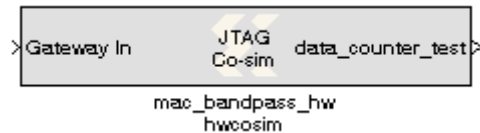


Figura 5-20. Bloque resumen abrirá en una ventana nueva Simulink.

- ⑤ Copia el bloque compilado y conéctelo en el diseño (véase la Figura 5-21).

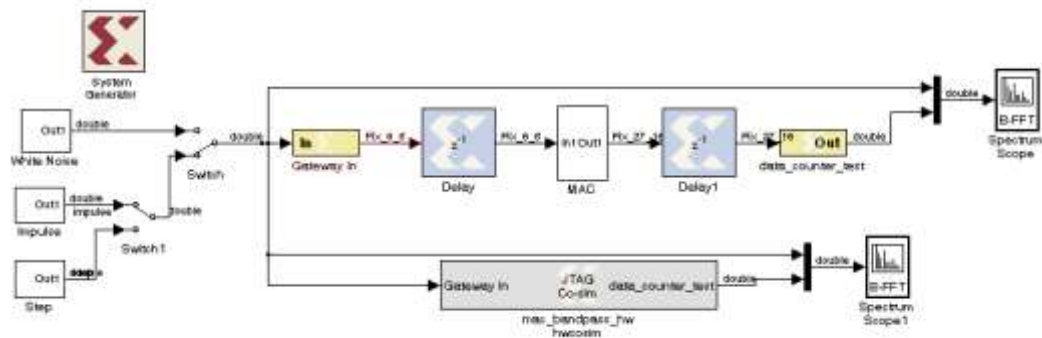


Figura 5-21. El diseño completo de prepararse para el nuevo hardware en la simulación de bucle.



Conecte la tarjeta de hardware y encenderlo. Seleccione la fuente de **ruido blanco** y ejecutar la simulación.

- ❶ Conecte la tarjeta de hardware y encenderlo
- ❷ Haga doble clic en el hardware como co-simulación de bloque y seleccione el **cable USB** de la **Plataforma de** la ficha de cable, que se utiliza para configurar la FPGA en el Spartan-3E kit de iniciación.
- ❸ Seleccione la fuente de **ruido blanco** y ejecutar la simulación para el número de muestras de **inf**
 - Nota: Los resultados de la simulación debe ser similar a la que se muestra en las Figuras 5-22. Observar los resultados similares de la producción de hardware en comparación con los bloques del sistema generador.

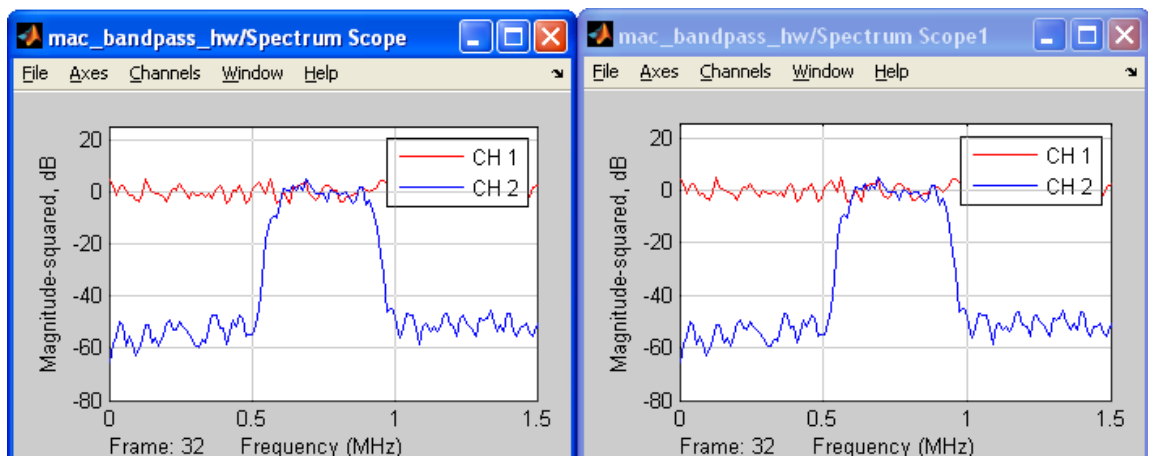


Figura 5-22. SYSGEN y salida de hardware en Spectrum Scope.

- ④ Mientras que la simulación se está ejecutando, puede cambiar la entrada de la fuente y ver el efecto en la salida de
- ⑤ Detener la simulación
- ⑥ Guardar y cierre el modelo de MATLAB

A.7.5.8 CONCLUSIÓN

En esta práctica, aprendió a diseñar, construir y verificar la MAC especificando lo basado en filtro pasa banda con bloques de base del sistema generador. Después de completar este laboratorio, usted sabe cómo construir un búfer de RAM cíclico, utiliza el bloque de reciente adhesión en el Sistema Generador de apoyo a cualquier tamaño de puerto que se desea, y cómo utilizar el Sistema Generador de construir una de las funciones más comunes DSP: la basada en MAC filtro FIR. También ha aprendido a utilizar el alcance del espectro blanco y fuente de ruido para ver la respuesta de frecuencia de un filtro. Además se ve el impacto del crecimiento de bits y los bloques de uso disponibles para seleccionar los bits que desee.

A.7.6 Práctica 6: Diseño de un Filtro Fir(Respuesta Finita Al Impulso

INTRODUCCION

En este laboratorio se mostrara de una manera específica y simulada la implementación de un filtro FIR.

Usando el Sistema de Generador del Fir y el bloque de herramientas fdga. El bloque de herramientas fdga es usado para definir el orden y coeficiente del filtro. Y el bloque de fir es usado para la simulación del simulink y el diseño de la implementación en fdga usando Xilinx Ise. También estarás capacitado para verificar la funcionalidad del diseño a través del hardware actual

OBJETIVOS

Después de completar este laboratorio usted será capaz de:

- Ingresar las características de su filtro y generar los coeficientes a través de la barra de herramientas fdat.
- Simular un diseño usando el bloque fir con los coeficientes generados por la barra de herramientas fdat.
- Trabajar el Hardware en el lazo o curva de verificación

DESCRIPCION DE DISEÑO

Usted es un diseñador dsp en sistemas cyberdyne .

Su compañía está investigando usando filtros digitales en vez de análogos para el sitio común detector de seguridad. Empeñado en optimizar su uso y reducir costos en todo el sistema esto ayudar a la compañía a adelantarse en el campo creciente de seguridad.

Las especificaciones del canal sencillo y el estimado de este en la tabla de abajo.

LA MUESTRA DE LA FRECUENCIA (F_s) = 1.5 MHz

- **Fstop 1 = 270 kHz**
- **Fpass 1 = 300 kHz**
- **Fpass 2 = 450 khz**
- **Fstop 2 = 480 kHz**
- **La atenuación de ambos lados de la banda de paso = 54 dB**
- **PASE DE RIZADO DE LA BANDA = 1**

Cyberdyne ha optado por ir con FPGAs debido a su flexibilidad y tiempo al mercado y ventajas de rendimiento sobre los procesadores DSP. Su experiencia en el diseño HDL es limitado y por lo tanto del Sistema Generador para DSP parece ser una solución excelente para la aplicación del filtro en un FPGA, como ya está familiarizado con los productos de The MathWorks.

Su gerente, Miles Booth, ha solicitado que se crea un prototipo de filtro a ser aplicado en su tablero de Spartan-3E™ prototipo que está casi completa. El prototipo debe terminarse lo antes posible de la convención inminente agresivo de Seguridad, que es la mayor convención de la industria del año, por lo que no se debe perder.

Su administrador ha proporcionado un modelo de partida que incluye las fuentes de entrada y salida de sumidero. Su diseño deberá ser simulado utilizando una fuente de azar y el chirrido de la Blockset DSP. Para analizar la salida del filtro, señales de entrada y salida se muestran en un ámbito de aplicación del espectro. Un ámbito de espectro se utiliza para comparar la respuesta de frecuencia del filtro de punto fijo FIR, que se llevará a cabo en la FPGA.

Dos fuentes diferentes se utilizan para simular el filtro:

- El bloque de chirrido, que barre entre las frecuencias específicas de 0 a 750kHz
- El generador de fuente de azar, que emite una señal aleatoria de distribución uniforme con un rango de 0 a 1. Uniforme es una mejor opción para conducir un filtro de punto fijo, porque está acotada

A.7.6.1 PROCEDIMIENTO

Este laboratorio consta de tres pasos principales

1. Generar los coeficientes para un filtro FIR
2. Modelar y simular el filtro FIR
3. Realizar la verificación en la curva del hardware

Debajo de cada instrucción general para un procedimiento determinado, se encuentra el paso de acompañamiento-por paso y cifras ilustran proporcionar más detalles para la realización de la instrucción general. Si usted se siente confiado acerca de una instrucción específica, no dude en saltarse el paso por paso y pasar a la instrucción general siguiente en el procedimiento.

Nota: Si usted no puede completar el laboratorio en este momento, usted puede descargar los archivos de laboratorio de este módulo desde el sitio del Programa en la Universidad de Xilinx <http://www.xilinx.com/univ>

GENERAR LOS COEFICIENTES DEL FILTRO FIR



Agregar el bloque de herramientas de la FDA Xilinx DSP a un diseño que contiene un filtro FIR de DA. Generar los \rightarrow Blockset coeficientes para el filtro FIR en el bloque con el FDATool para las siguientes especificaciones

- Frecuencia de muestreo (F) = 1,5 MHz
- Fstop 1 = 270 KHz
- Fpass 1 = 300 KHz
- Fpass 2 = 450 kHz
- Fstop 2 = 480 KHz
- La atenuación de ambos lados de la banda de paso = 54 dB
- Rizado en banda pass = 1

- ❶ Abra la ventana de comandos de MATLAB, haciendo doble clic en Programas \rightarrow clic en el icono de MATLAB en su escritorio, o vaya al menú Inicio \rightarrow MATLAB R2007a_MATLAB R2007a.
- ❷ En Matlab, a cambio de directorio `c:\xup\dsp_flow\labs\lab6` /: Escriba `cd C:\xup\dsp_flow\labs\lab6` / en la ventana de comandos.

- ③ Abra el modelo de bandpass_filter.mdl de la ventana de la consola de MATLAB
- ④ Añadir DSP para el diseño de → FDATool por el bloque de Xilinx Blockset
- ⑤ Introduzca los parámetros del filtro siguiente en la ventana de diseño FDATool Filtro (Figura 6-1)
 - Tipo de respuesta: Paso de banda
 - Unidades: KHz
 - Frecuencia de muestreo (F) = 1,5 MHz
 - Fstop 1 = 270 KHz
 - Fpass 1 = 300 KHz
 - Fpass 2 = 450 kHz
 - Fstop 2 = 480 KHz
 - La atenuación de ambos lados de la banda de paso = 54 dB (Astop1 y Astop2 parámetros)
 - Rizado en banda Pass = 1 (CONTRASEÑA)

The screenshot shows the FDATool interface with the following settings:

Response	Filter	Frequency	Magnitude
<input type="radio"/> Lowpass	<input type="radio"/> Specify order: 10	Units: kHz	Units: dB
<input type="radio"/> Highpass	<input checked="" type="radio"/> Minimum order	Fs: 1500	Astop1: 54
<input checked="" type="radio"/> Bandpass	Options	Fstop1: 270	Apass: 1
<input type="radio"/> Bandstop	Density: 16	Fpass1: 300	Astop2: 54
<input type="radio"/> Differentiator		Fpass2: 450	
Design		Fstop2: 480	
<input type="radio"/> IIR: Butterworth			
<input checked="" type="radio"/> FIR: Equiripple			

Figura 6-1. Diseño de un filtro en FDATool.

- ⑥ Haga clic en el botón Diseño de filtro para determinar el orden del filtro

La ventana del espectro, se actualizará y se verá como se muestra en la Figura 6-2

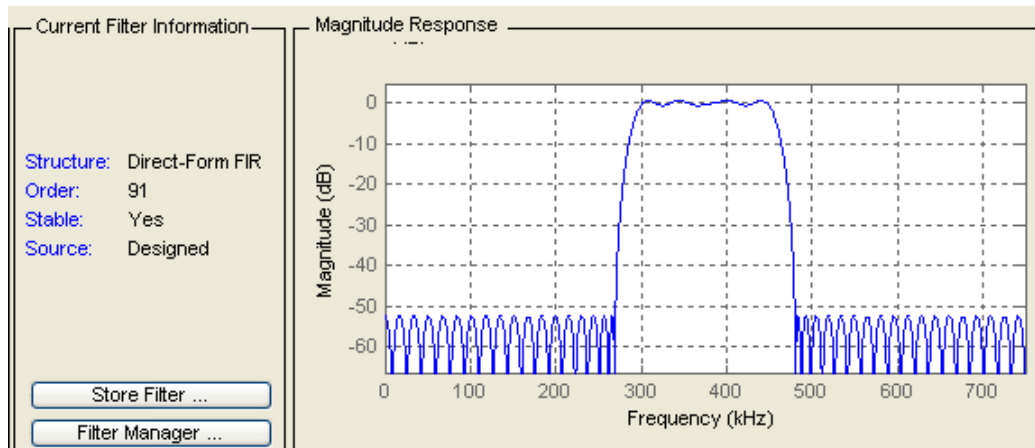


Figura 6-2. Diseñado Magnitud filtro de Respuesta.



En base a las especificaciones definidas, ¿cuál es el orden del filtro mínimo?

- Los coeficientes de exportación en el área de trabajo con el nombre de variable como Numerador Num (Figura 6-3) usando Archivo exportación

Nota: Esto añade la variable Número de votos en su área de trabajo de MATLAB. Para un filtro FIR, Num. Representa los coeficientes que se utilizan en el filtro. Este es también un paso opcional que los coeficientes que todavía están disponibles a través del bloque FDATool



Figura 6-3. Coeficientes de Exportación en el área de trabajo.

- ⑧ Tipo Número de votos en la ventana de la consola de MATLAB para ver la lista de los coeficientes.
- ⑨ Tipo máximo (Num) y mínimo(Num) en la ventana de la consola de MATLAB para determinar el valor del coeficiente máximo que especifique adecuadamente el ancho de coeficiente y binario punto

2. Rellene el siguiente información relacionada con los coeficientes de

Valor máximo: _____

Valor mínimo: _____

A.7.6.2 MODELAR Y SIMULAR EL FILTRO FIR

PASO 2

Agregar el bloque de filtro FIR de la biblioteca de la DSP Xilinx y asociar a los coeficientes generados. Simular el diseño y verificar la funcionalidad. Añadir la conversión en bloque sobre la salida de la FIR de bloque para reducir el rango dinámico. Simular el diseño para comprobar la funcionalidad.

- ❶ Añadir la FIR (DA FIR de v9_0) filtro de bloque de la biblioteca de Xilinx DSP para el diseño de→Blockset
- ❷ Haga doble clic en la FIR de bloque e ingresar los siguientes parámetros en la ventana de parámetros de bloque (Figura 6-4). Haga clic en Aceptar.
 - Coeficientes: xlfda_numerator ('FDATool')
 - Coeficiente de Estructura: inferidos de los coeficientes
 - Número de bits por coeficientes: 12
 - Binario punto para los coeficientes: 12
 - Proporcionar puertos válidos: sin control

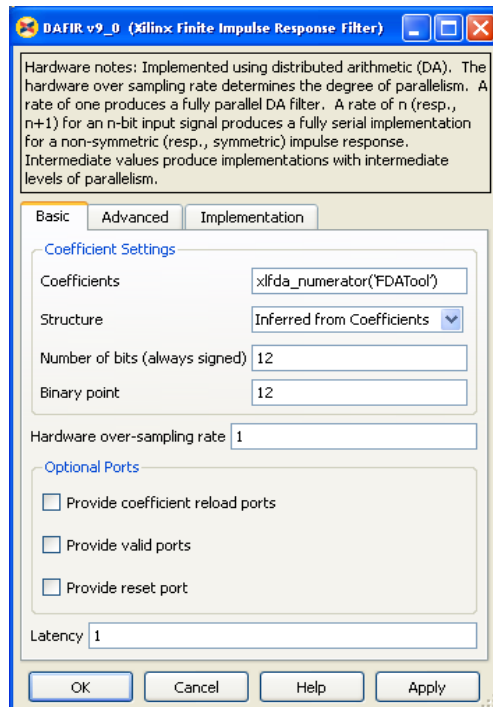


Figura 6-4. FIR de filtro de bloque de parámetros.

- ⑤ Conecte los bloques para que el diseño se asemejen a la Figura 6-5

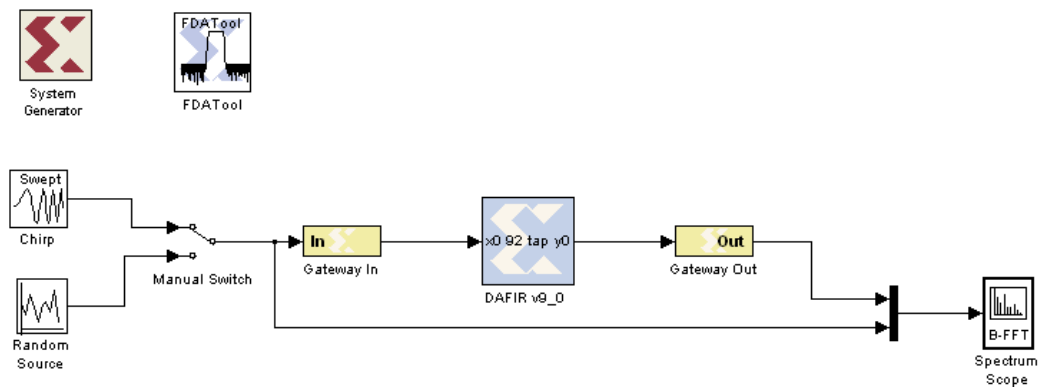


Figura 6-5. FIR de filtro de bloque de diseño basado en simulación listo

- ④ Haga doble clic en Gateway en bloque y establecer el formato de FIX_8_6 y período de muestreo de 1 / 1500000
- ⑤ Seleccione la Fuente del timbre y comenzar la simulación

- 6 Traiga el ámbito de aplicación a la parte frontal y verifique que la señal que sale del filtro FIR se ha atenuado y se ven como en la Figura 6-6 y Figura 6-7, a continuación

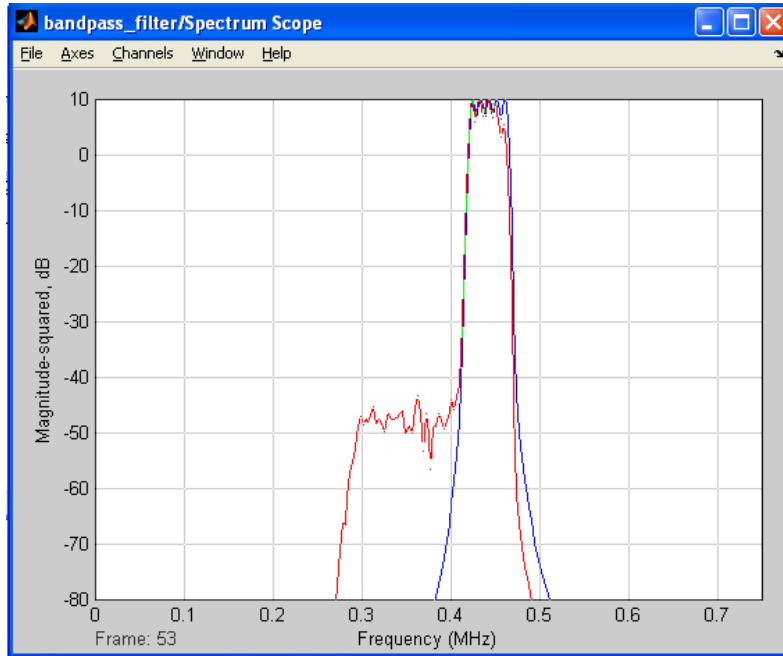


Figura 6-6. Sin atenuación en Paso de Banda (Espectro enÁmbito de aplicación).

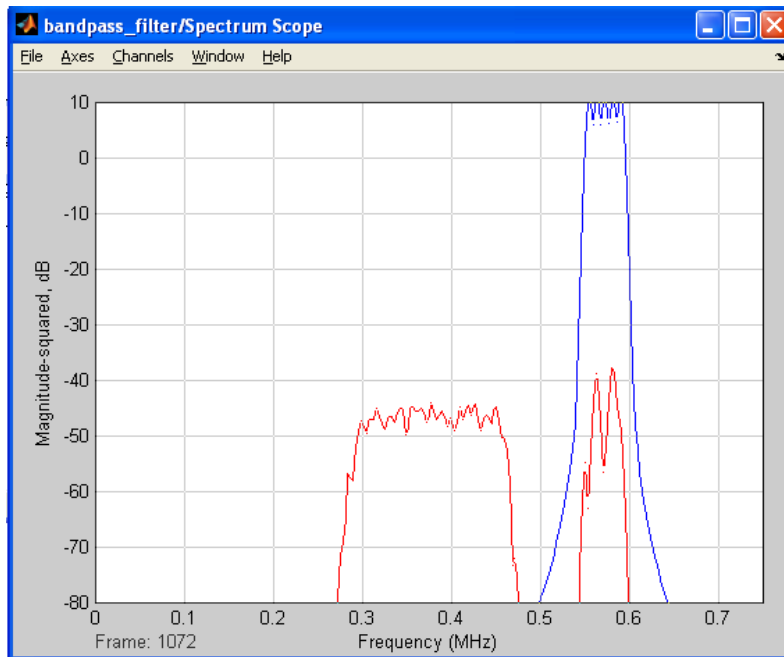


Figura 6-7. Atenuación en banda de detencion (espectro Ámbito de aplicación).

- 7 Seleccione la Fuente del azar y ejecutar la simulación (Figura 6-8)

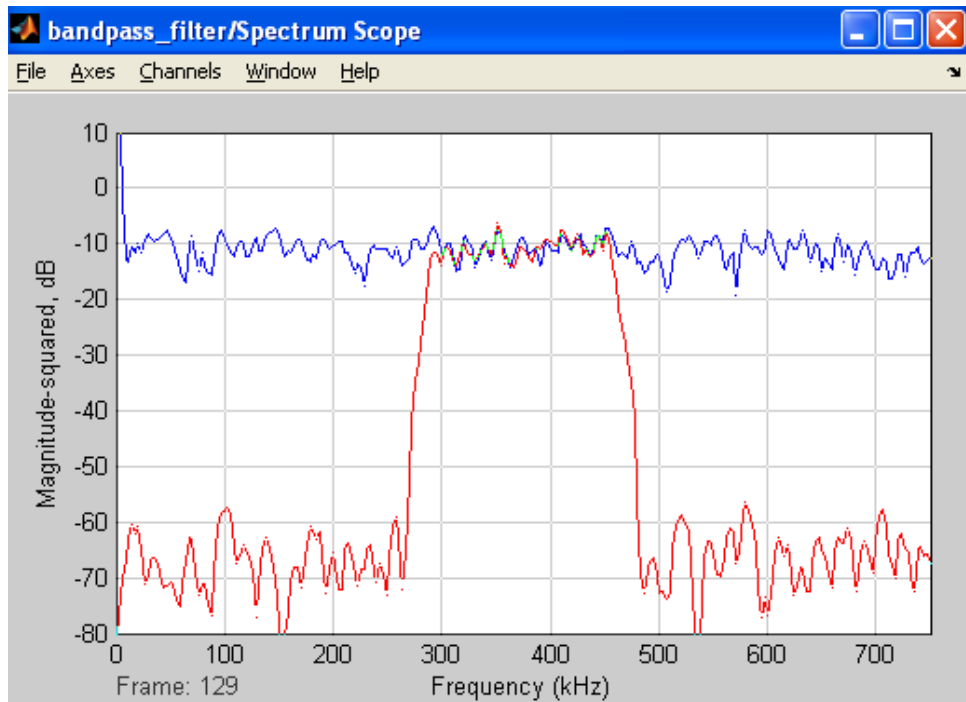


Figura 6-8. Fuente Variable (espectro en Ámbito de aplicación).

- ⑧ Detener la simulación
- ⑧ Añadir un bloque de Convert (Xilinx Elementos básicos) en la salida FIR y configurarlo como cuantización FIX_8_6 como truncar y desbordamiento como Wrap

Nota: El diseño debe ser similar a la que se muestra en la Figura 6-9.

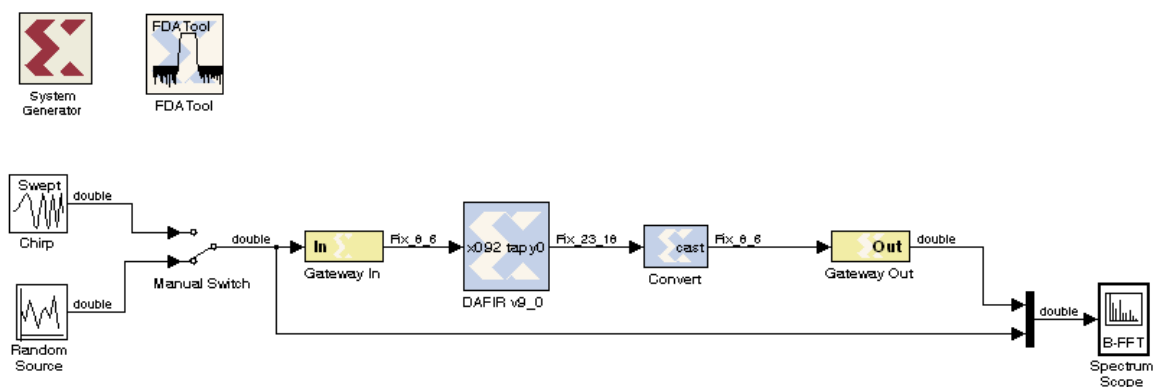


Figure 6-9. Diseño completado de filtro fir

- ⑩ Ejecutar la simulación utilizando la señal de timbre y los insumos de ruido blanco, tomando nota de la reducción en el rango dinámico, debido a la reducción del número de bits de salida. Ver las figuras 6-10 y 6-11.

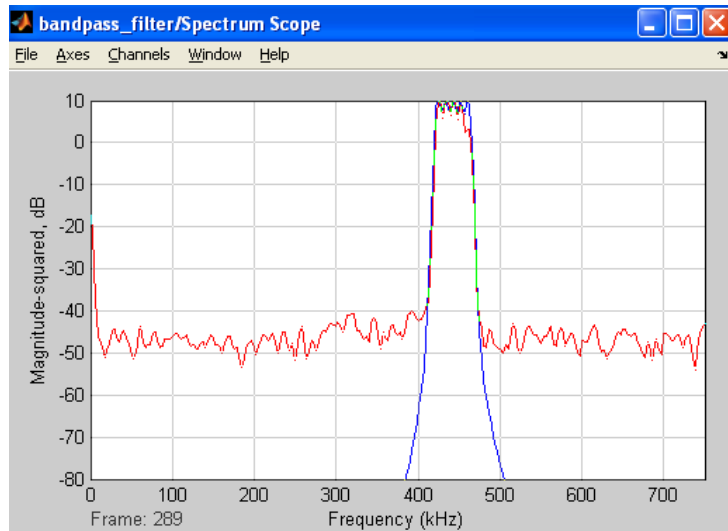


Figura 6-10. De respuesta del filtro debido a cantar de entrada.

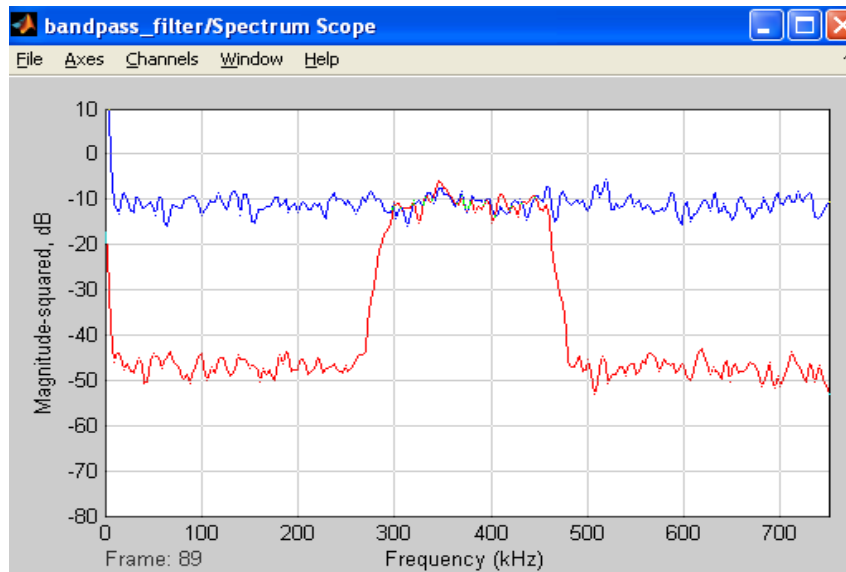


Figura 6-11. De respuesta del filtro debido a la entrada de ruido blanco.

A.7.6.3 REALIZACION DE HARDWARE EN CURVA DE VERIFICACION



Usando el Sistema Generador de modo, generar el hardware y comprobar que el diseño de las obras a través de la Junta de hardware. Simular el diseño a través de Simulink.

- ❶ Guardar el modelo como `bandpass_filter_hwcosim.mdl`
- ❷ Haga doble clic en el Sistema Generador de señal y establecer los siguientes parámetros de
 - `sp3e_starter_kit` Compilación: Hardware Co-simulación
 - Síntesis de la herramienta: XST
 - Objetivo del repertorio: `C: / xup/dsp_flow/labs/lab6/sp3e (o ./sp3e)`
 - Crear Testbench: sin comprobar
 - Sistema de Simulink Período (s): 1 / 1500000
- ❸ Haga clic en el botón Generar y un mensaje de advertencia que indica que la latencia del bloque DA_FIR debe ajustarse a fin de aplicar el diseño de
- ❹ fijar la latencia del bloque DA_FIR a 14, haga clic en el botón Aplicar y, a continuación, haga clic en el botón Generar del Sistema Generador de señal.
- ❺ un cuadro de diálogo que muestran los progresos proceso de compilación, como se muestra en la Figura 6-10

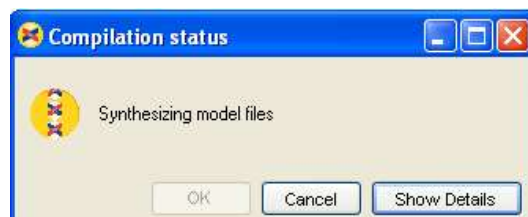


Figura 6-10. Compilación de progreso en ventana de comandos

- ❻ Cuando la generación se completa con éxito, una nueva ventana de la biblioteca de Simulink se abre y un bloque compilado con el número adecuado de entradas y salidas se mostrará

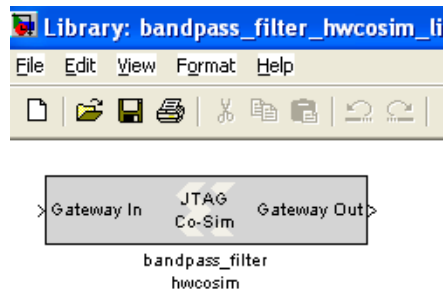


Figura 6-11. El bloque resumido se abrirá en una ventana nueva Simulink.

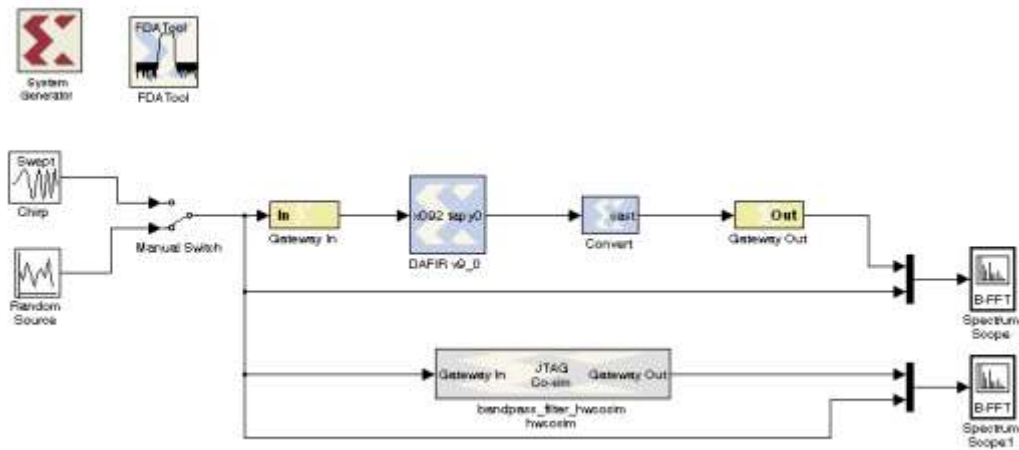


Figura 6-12. Diseño completo preparado para el nuevo hardware en la simulación de bucle.



Conecte la tarjeta de hardware y simular el diseño a través de Simulink.

- ❶ Conecte el cable de alimentación de la placa del hardware
- ❷ Conecte el cable de descarga entre la Junta y el PC
- ❸ Haga doble clic en el hardware como co-simulación de Block y ponga el cable de descarga a la Plataforma de USB en la pestaña de cable

- ④ Seleccione la fuente de azar y haga clic en el botón Ejecutar () en la ventana de Simulink para ejecutar la simulación. El bit de archivo de configuración se descargará y una simulación se llevará a cabo
- ⑤ El resultado de la simulación en el ámbito de la producción se mostrará la salida del simulador de Simulink en la parte superior y la salida de hardware en las parcelas de fondo (Figura 6-13)

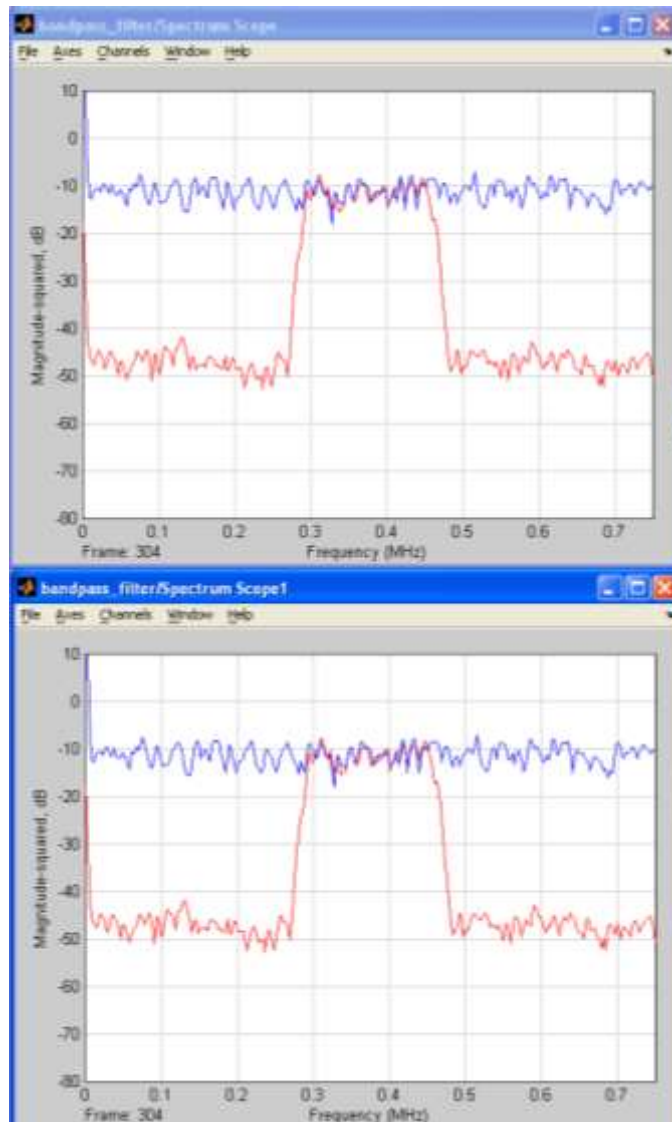


Figura 6-13. Listado de resultados de simulación de la salida del Simulador en la parte superior del producto de hardware en el fondo.

⑥ Haga clic en el botón de parada y apagar la alimentación

⑦ Guardar y cierre el modelo de MATLAB

A.7.6.4 CONCLUSION

En esta práctica, aprendió a utilizar el bloque de herramientas FDAT para crear coeficientes de filtros y utilizarlos desde el bloque de filtro FIR. Usted utiliza el bloque de recursos Estimados para estimar los recursos.

Respuestas

1. En base a las especificaciones definidas, ¿cuál es el orden del filtro mínimo?

92

2. Rellene el siguiente información relacionada con los coeficientes de

Valor máximo: 0,1610

Valor mínimo: -0,1541

RESPUESTAS

1. La especificación requiere 12-bit de datos para los coeficientes. ¿Cuál es el formato óptimo para el 12-bit de los coeficientes de estar?

FIX_12_12

2. Anote las expresiones para el bloque de data_counter que vais a entrar para:

Número de bits: ceil(log2(2 * longitud(coeficiente)))

Conde de valor: (longitud coef) - 1

3. Anote las expresiones para el bloque de coef_counter que vais a entrar para:

Número de bits: $\text{ceil}(\log_2(2 * \text{longitud}(\text{coeficiente})))$

Valor inicial: $\text{longitud}(\text{coeficiente})$

Cuenta hasta Valor: $2 * \text{longitud}(\text{coeficiente}) - 1$

Periodo de muestra: T_s

4. Escriba las expresiones para el bloque constante de que quieres entrar para:

Valor Constante: $2 * \text{longitud}(\text{coeficiente}) - 1$

Número de bits: $\text{ceil}(\log_2(2 * \text{longitud}(\text{coeficiente})))$

5. Escriba las expresiones de la memoria RAM de doble puerto que vais a entrar para:

Profundidad: $2 * \text{longitud}(\text{coeficiente})$

De valor inicial Vector: $[\text{ceros}(1, \text{longitud}(\text{coeficiente})) \text{coef}']$

6. ¿Cuál debe ser la expresión para el período de la muestra?

$T_s / \text{longitud}(\text{coeficiente})$

7. ¿Qué señal desde el bloque de lógica de control deben estar conectados a la reinicialización del acumulador y la CE del registro de captura?

WE

8. ¿Por qué? ¿Hay alguna lógica extra que se requiere?

Debido a que el multiplicador tiene una latencia de 3, y la doble lectura del puerto también es sincrónica, es necesario retrasar la señal de que por cuatro ciclos de reloj. Elemento de retardo de uso y asignar la latencia de 4

