

Implementación de un filtro FIR elimina banda

Microarquitecturas y Softcores

Autor: José Luis Krüger

Docente: Nicolas Alvarez

Índice

Índice.....	1
Introducción.....	2
Filtro FIR.....	2
Filtro suprime banda.....	2
Implementación mediante HDL.....	3
Simulación.....	4
Cálculo de los pasos del NCO.....	5
Síntesis y pruebas en el entorno de Vivado.....	6
Filtro elimina banda Resultados en Vivado.....	6
Desarrollo del IP core.....	8
Conclusiones.....	13

Introducción

Como proyecto final de la asignatura Circuitos Lógicos Programables (CLP) se optó por la implementación de un filtro FIR (elimina banda).

Filtro FIR

Un filtro FIR (Respuesta de Impulso Finita) es un tipo fundamental de filtro digital cuya principal característica es que su respuesta al impulso tiene una duración finita, extinguiéndose completamente después de un período de tiempo determinado. A diferencia de los filtros IIR (Infinite Impulse Response), la salida de un filtro FIR depende únicamente de los valores de entrada presentes y pasados, lo que los hace inherentemente estables. La salida $y[n]$ de un filtro FIR se calcula mediante la convolución de la señal de entrada $x[n]$ con los coeficientes del filtro (b_k):

$$y[n] = \sum_{k=0}^M b_k x[n - k]$$

Filtro suprime banda

Un filtro suprime banda (también conocido como filtro de rechazo de banda o band-stop filter) es un tipo de filtro que está diseñado para atenuar o eliminar un rango específico de frecuencias, permitiendo que las frecuencias que se encuentran por encima y por debajo de esta banda pasen con una atenuación mínima.

Características principales:

- Banda de Rechazo (Stopband): Es el rango de frecuencias que el filtro está diseñado para atenuar significativamente.
- Frecuencias de Corte (Cutoff Frequencies): Son las frecuencias que definen los límites de la banda de rechazo. Se suelen especificar dos frecuencias de corte: una inferior y una superior. Las frecuencias entre estas dos son las que se atenúan.
- Banda de Paso (Passband): Son los rangos de frecuencias que se encuentran por debajo de la frecuencia de corte inferior y por encima de la frecuencia de corte superior. Estas frecuencias idealmente pasan a través del filtro sin atenuación.
- Ancho de Banda de Rechazo (Stopband Bandwidth): Es la diferencia entre la frecuencia de corte superior e inferior.
- Atenuación en la Banda de Rechazo: Especifica cuánto se atenúan las señales dentro de la banda de rechazo (en decibelios, dB).

- Transición entre Banda de Paso y Banda de Rechazo: Idealmente, la transición sería abrupta, pero en filtros reales, hay una zona de transición donde la atenuación aumenta gradualmente.

Diseño del filtro

Para obtener los coeficientes del filtro se utilizó el recurso <http://t-filter.engineerjs.com/> que permite configurar de manera sencilla los parámetros del sistema.

Para este filtro se utilizaron los valores de la figura 1, en el mismo gráfico se observa la respuesta tentativa del diseño.



Figura 1: Cálculo del filtro.

Implementación mediante HDL

Para la implementación se empleó VHDL, adoptando una arquitectura de componentes tal como se ilustra en la figura 2. El diseño comienza con un adaptador de entrada, encargado de la interfaz entre el NCO y los diversos módulos subsiguientes. Tras esta adaptación, se dispone un registro de desplazamiento que genera el corrimiento de muestras requerido para la operación del filtro. A continuación, se encuentran implementados N multiplicadores, donde N corresponde al orden del filtro. Finalmente, la salida converge en un sumador, el cual realiza la agregación de todos los productos para producir la señal filtrada. Los coeficientes del filtro se almacenan en una memoria ROM. Finalmente, la señal de salida se readapta nuevamente al formato unsigned.

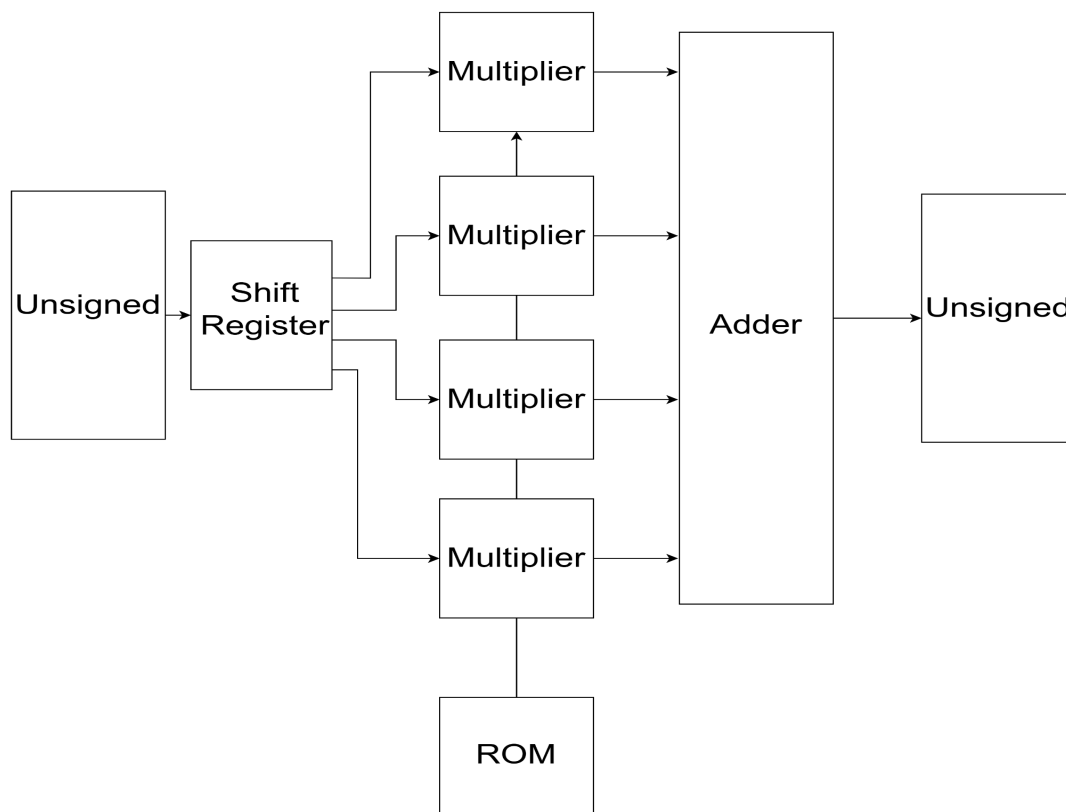


Figura 2: Diagrama en bloques simplificado que ilustra la implementación adoptada

Simulación

La simulación se llevó a cabo utilizando la herramienta ModelSim. Para obtener la respuesta del filtro, se empleó un NCO proporcionado por la cátedra, integrado en un banco de pruebas que, en conjunto con el filtro y un generador de habilitación, permitió reproducir las condiciones de operación deseadas.

Para verificar la atenuación del filtro en la banda adecuada, se seleccionaron tres frecuencias de prueba: una en la banda baja, aproximadamente a 31 kHz; una en la banda media, a 400 kHz; y una en la banda alta, alrededor de 800 kHz. Se observó que el funcionamiento en las tres bandas fue el esperado, a excepción de una distorsión en la señal en la banda alta.

Esta distorsión podría atribuirse a la frecuencia de muestreo utilizada, tanto para el diseño del filtro como en el generador de habilitación, que fue de 1.92 MHz. Esta frecuencia de muestreo establece una frecuencia máxima de entrada de aproximadamente 900 kHz. Los resultados de esta simulación se pueden apreciar en la figura 3.

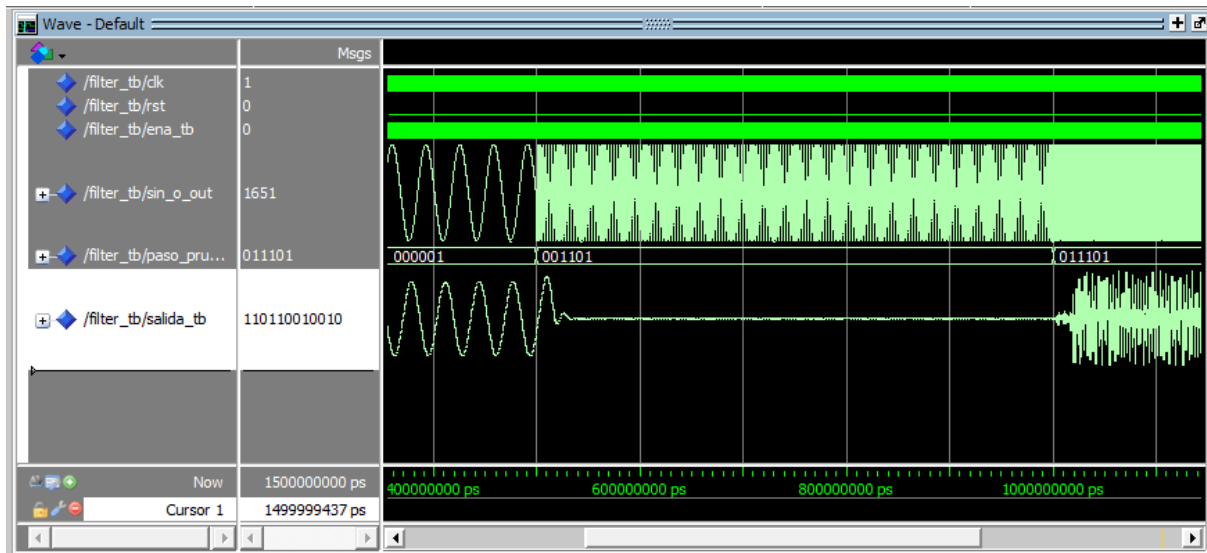


Figura 3: Resultados de la simulación del filtro en ModelSim.

La frecuencia del NCO se estableció mediante su parámetro de paso. Este parámetro se configuró en 6 bits, lo que, con un ancho del acumulador de fase de 12 bits, resulta en un rango de frecuencia de salida de entre 31 kHz y 1.92 MHz.

Cálculo de los pasos del NCO

Como se mencionó anteriormente, para la simulación se utilizaron 3 frecuencias de prueba. La frecuencia de reloj empleada fue de 125 MHz, valor compatible con el clock del kit de desarrollo Arty Z7-10.

La ecuación para calcular el paso (step) del NCO es:

$$Paso = \frac{F_{NCO} \cdot 2^N}{F_{clk}}$$

Donde:

- F_{NCO} es la frecuencia de salida deseada del NCO.
- 2^N representa el tamaño del acumulador de fase (en este caso, $2^{12} = 4096$).
- F_{clk} es la frecuencia del reloj del sistema (125 MHz).

Aplicando la fórmula proporcionada, se obtuvieron los siguientes valores para el paso del NCO correspondientes a las frecuencias de prueba seleccionadas:

- Para 31 kHz: "000001"
- Para 400 kHz: "001101"
- Para 800 kHz: "011010"

Síntesis y pruebas en el entorno de Vivado

Una vez obtenidos resultados positivos en la simulación se creó un proyecto en Vivado para realizar la síntesis e implementación final en el hardware de la fpga. Para ellos se creó un elemento TOP en el cual se implementó un VIO para inyectar los pasos de prueba al NCO y un ILA para observar tanto la señal de entrada como la salida del filtro. En la figura 4 se observa el esquemático final del hardware implementado obtenido desde la herramienta.

Tras obtener resultados satisfactorios en la simulación, se procedió a la creación de un proyecto en Vivado para llevar a cabo la síntesis e implementación final en el hardware de la FPGA. Para ello, se definió un módulo TOP en el cual se integró un Virtual Input/Output (VIO) para inyectar los valores de paso de prueba al NCO y un Integrated Logic Analyzer (ILA) para la observación tanto de la señal de entrada como de la señal de salida del filtro. En la figura 4 se puede apreciar el esquemático final del hardware implementado, tal como fue generado por la herramienta Vivado.

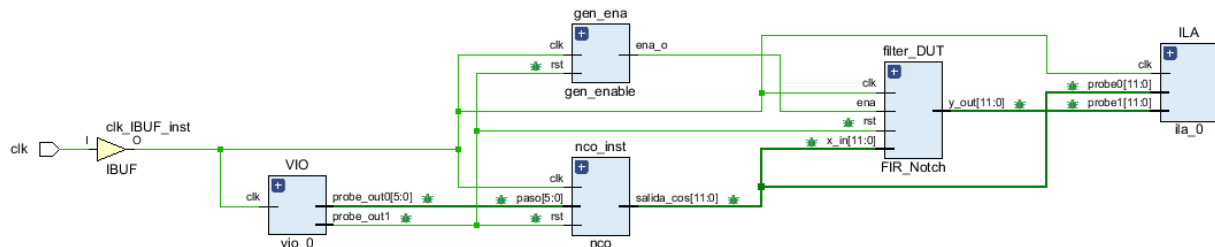


Figura 4: Diagrama de la implementación obtenida desde Vivado.

Filtro elimina banda Resultados en Vivado

A través de la interfaz VIO, se inyectaron al NCO los valores de paso correspondientes a las frecuencias de prueba previamente definidas.

La respuesta obtenida en el ILA al inyectar la señal de baja frecuencia (aproximadamente 31 kHz) se ilustra en la figura 5. En la figura 6, se puede observar la respuesta correspondiente a la señal de frecuencia media (400 kHz). Finalmente, la figura 7 muestra la respuesta obtenida al inyectar la señal de alta frecuencia (alrededor de 800 kHz).



Figura 5: Respuesta del sistema ante la inyección de una señal de baja frecuencia (aproximadamente 31 kHz). Se observa la señal de entrada y la señal de salida del filtro, manteniendo la forma de onda.

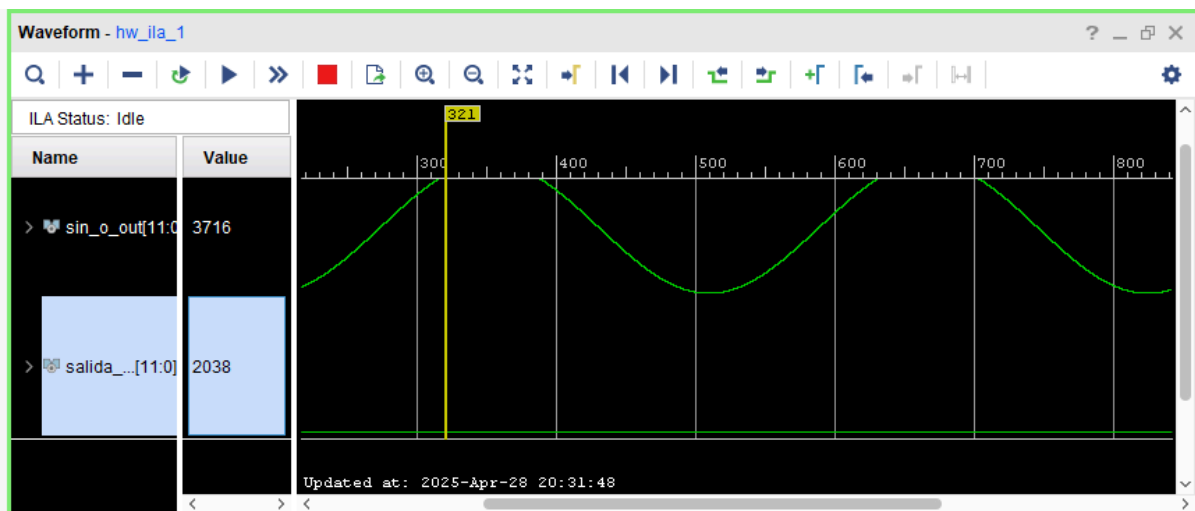


Figura 6: Respuesta del sistema ante la inyección de una señal de frecuencia media (400 kHz). Se observa la señal de entrada y la señal de salida del filtro, donde se aprecia una atenuación total de la señal.

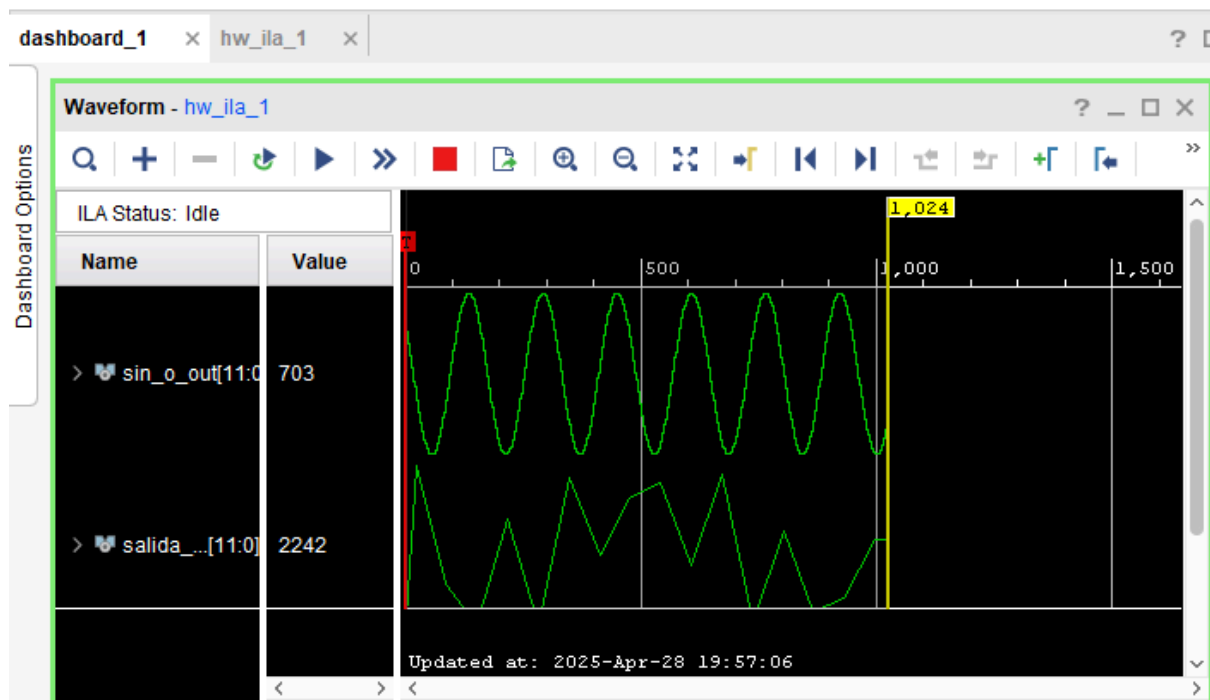


Figura 7: Respuesta del sistema ante la inyección de la señal de alta frecuencia (aproximadamente 800 kHz). Se observa la señal de entrada y la señal de salida del filtro, donde se aprecia una deformación en la forma de onda, similar a lo observado en la simulación.

Desarrollo del IP core

Cómo trabajo final se decidió empaquetar un filtro FIR elimina banda. Para lograrlo, se creó el archivo `filter_pkg.vhd`.

Dicho componente consta con 4 conexiones, dos de ellos corresponden a la entrada y salida del filtro interno, y las dos restantes son las correspondientes a CLK y el paso del NCO. Este último parámetro es el que se configura por medio del bus AXI.

```
entity filter_pkg is
  port(
    clk_top      : in  std_logic;
    fir_in_top   : out unsigned(11 downto 0);
    PASO_W_top   : in  unsigned (5 downto 0);
    y_out_top    : out unsigned(11 downto 0)
  );
end;
```

A continuación se presenta el IP core logrado:

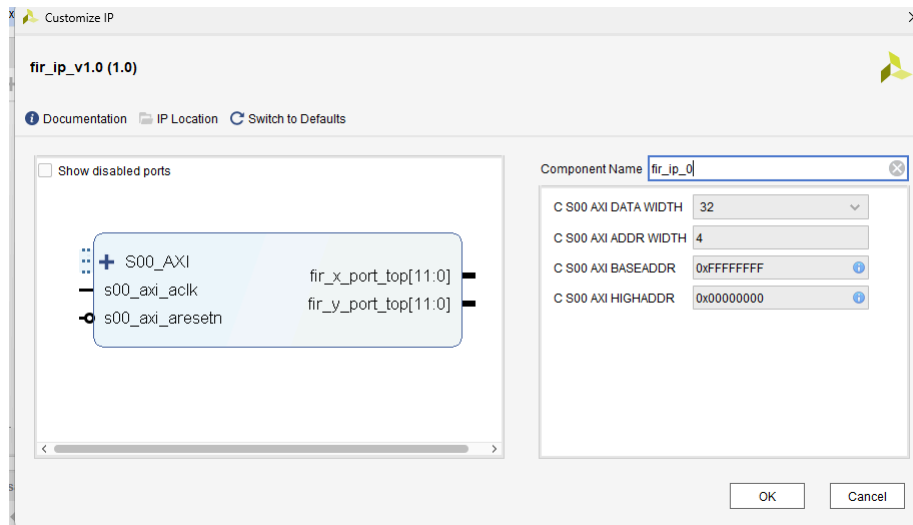


Figura 8: Vista del IP core generado sobre el entorno de Vivado.

El circuito completo cuenta, además, consta con las conexiones de dos switches y cuatro botones (utilizados para establecer el paso del NCO), un VIO (necesario para comandar los botones y los switches) y un ILA (necesario para visualizar las formas de onda):

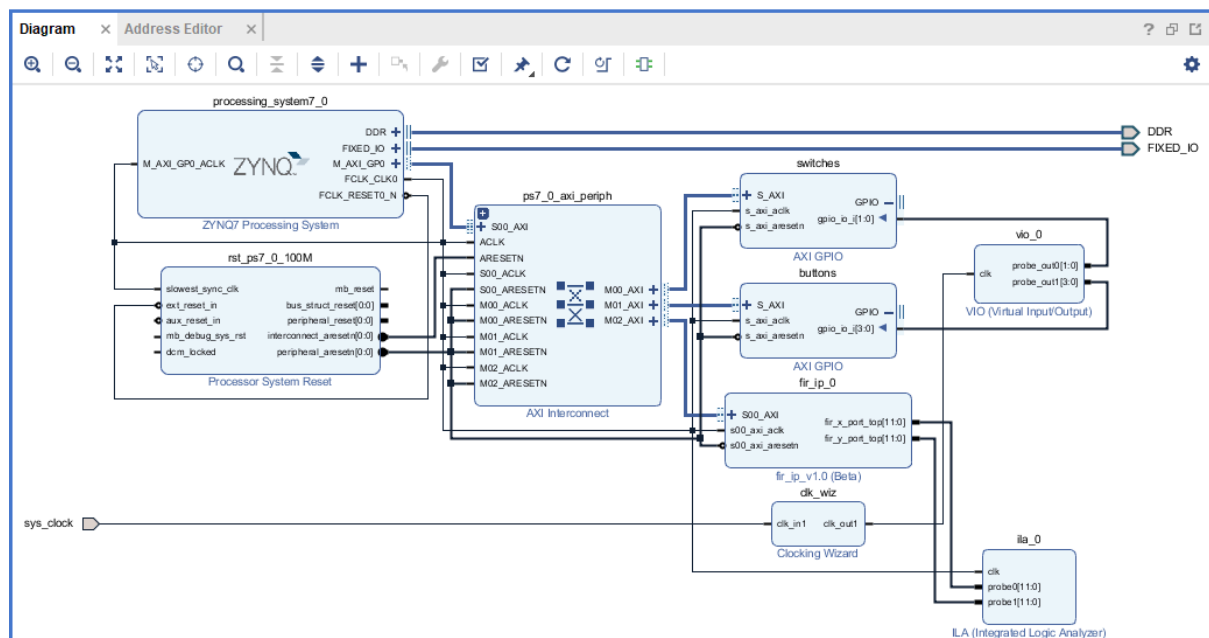


Figura 9: Diagrama del circuito completo implementado en el trabajo.

Ya en el entorno SDK se desarrolló el código necesario para comandar, a través del bus AXI, el paso de control del NCO:

```

psb_check = XGpio_DiscreteRead(&push, 1);
dip_check = XGpio_DiscreteRead(&dip, 1);    // leo switches

xil_printf("Botones: 0x%x | Switches (PASO NCO): 0x%x\r\n",
psb_check, dip_check);

unsigned int paso_w = (dip_check << 4) + psb_check;

// Escribe el valor le'do al registro 0 del IP FIR (PASO_W)
FIR_IP_mWriteReg(XPAR_FIR_IP_0_S00_AXI_BASEADDR,
FIR_IP_S00_AXI_SLV_REG0_OFFSET, paso_w);

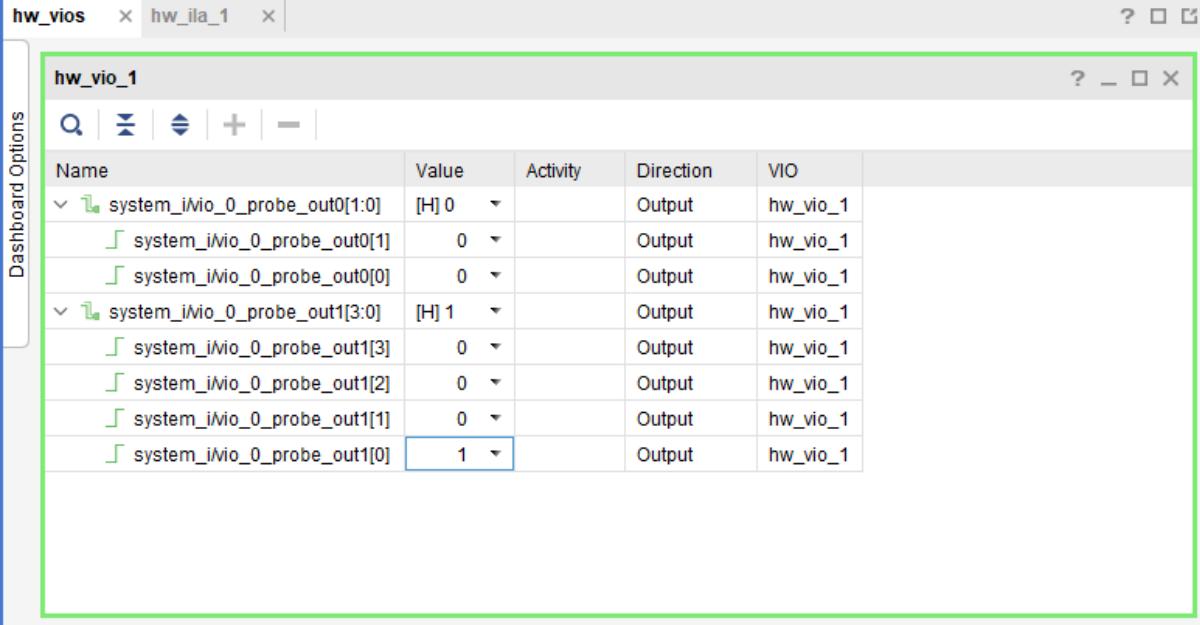
sleep(1);

```

Una vez sintetizado el hardware y compilado el código, se realizaron las pruebas sobre el servidor remoto provisto por la cátedra.

A continuación se presentan las distintas configuraciones realizadas:

Frecuencia baja



Name	Value	Activity	Direction	VIO
system_iVio_0_probe_out0[1:0]	[H] 0		Output	hw_vio_1
system_iVio_0_probe_out0[1]	0		Output	hw_vio_1
system_iVio_0_probe_out0[0]	0		Output	hw_vio_1
system_iVio_0_probe_out1[3:0]	[H] 1		Output	hw_vio_1
system_iVio_0_probe_out1[3]	0		Output	hw_vio_1
system_iVio_0_probe_out1[2]	0		Output	hw_vio_1
system_iVio_0_probe_out1[1]	0		Output	hw_vio_1
system_iVio_0_probe_out1[0]	1		Output	hw_vio_1

Figura 10: Interfaz de configuración de los switches a través de un VIO (en baja frecuencia).

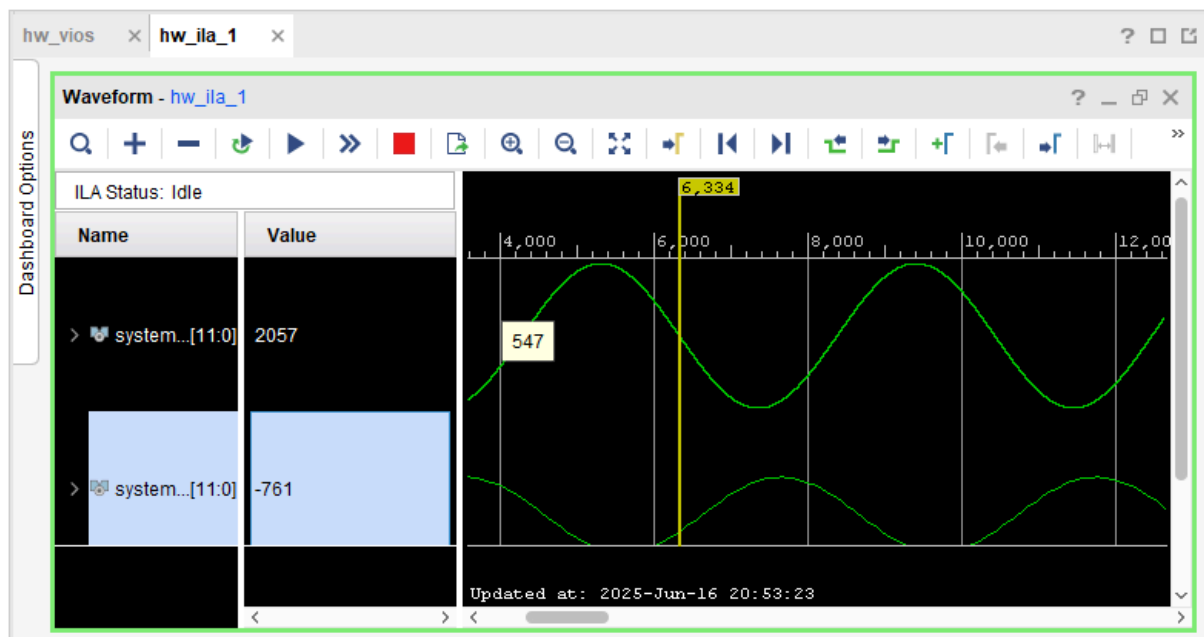


Figura 11: Respuesta del sistema ante la inyección de la señal de baja frecuencia.

Frecuencia media

hw_vio_1

Name	Value	Activity	Direction	VIO
system_iVio_0_probe_out0[1:0]	[H] 0		Output	hw_vio_1
system_iVio_0_probe_out0[1]	0		Output	hw_vio_1
system_iVio_0_probe_out0[0]	0		Output	hw_vio_1
system_iVio_0_probe_out1[3:0]	[H] 9		Output	hw_vio_1
system_iVio_0_probe_out1[3]	1		Output	hw_vio_1
system_iVio_0_probe_out1[2]	0		Output	hw_vio_1
system_iVio_0_probe_out1[1]	0		Output	hw_vio_1
system_iVio_0_probe_out1[0]	1		Output	hw_vio_1

Figura 12: Interfaz de configuración de los switches a través de un VIO (en frecuencia media).



Figura 13: Respuesta del sistema ante la inyección de la señal de frecuencia en mitad de la banda de paso.

Frecuencia alta

The screenshot shows the 'hw_vio_1' configuration window. It contains a table with the following data:

Name	Value	Activity	Direction	VIO
system_iVio_0_probe_out0[1:0]	[H] 1		Output	hw_vio_1
system_iVio_0_probe_out0[1]	0		Output	hw_vio_1
system_iVio_0_probe_out0[0]	1		Output	hw_vio_1
system_iVio_0_probe_out1[3:0]	[H] A		Output	hw_vio_1
system_iVio_0_probe_out1[3]	1		Output	hw_vio_1
system_iVio_0_probe_out1[2]	0		Output	hw_vio_1
system_iVio_0_probe_out1[1]	1		Output	hw_vio_1
system_iVio_0_probe_out1[0]	0		Output	hw_vio_1

Figura 14: Interfaz de configuración de los switches a través de un VIO (en frecuencia alta).

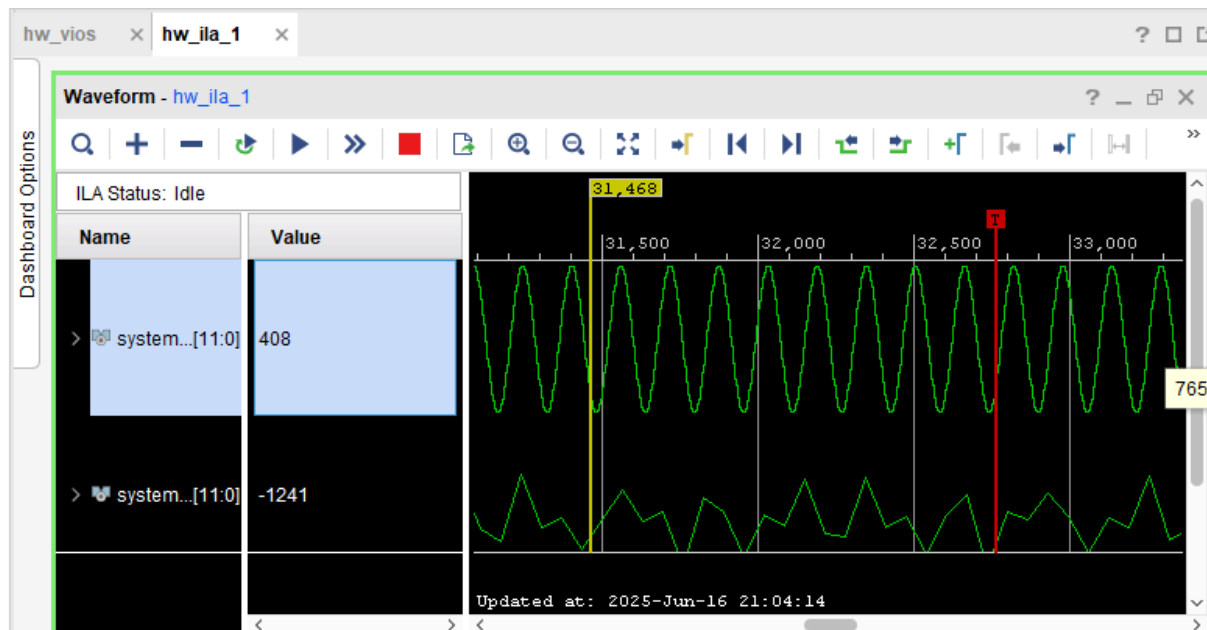


Figura 15: Respuesta del sistema ante la inyección de la señal de frecuencia alta. Se puede observar la deformación producto de que la frecuencia inyectada se encuentra cerca de la frecuencia límite del filtro.

Conclusiones

Las pruebas de simulación implementadas en ModelSim demostraron el funcionamiento esperado del filtro diseñado. Se verificó la correcta transmisión de la señal en la banda de paso (baja frecuencia), la atenuación significativa en la banda de rechazo (frecuencia media) y la presencia de distorsión en la banda superior. Este último resultado se atribuyó a las limitaciones impuestas por la frecuencia de muestreo utilizada en el diseño y el generador de habilitación.

La posterior implementación y prueba en el hardware de la FPGA Arty Z7-10, mediante el entorno Vivado y las herramientas de depuración VIO e ILA, corroboraron en gran medida los resultados obtenidos en la simulación. Se constató el paso de la señal de baja frecuencia sin una atenuación significativa, la efectiva supresión de la señal de frecuencia media y la reproducción de la distorsión observada en la banda de alta frecuencia.

Estos resultados validan la arquitectura de componentes implementada en VHDL para el diseño del filtro y su correcta síntesis e implementación en el hardware. La correspondencia entre los resultados de simulación y las mediciones en el hardware real proporciona una alta confianza en el funcionamiento del diseño. No obstante, la distorsión observada en la banda de alta frecuencia sugiere la necesidad de explorar posibles mejoras en el diseño o en la frecuencia de muestreo para optimizar el rendimiento en todo el espectro de interés.

Finalmente, el resultado obtenido al integrar el hardware propio y el software programado en el microcontrolador embebido evidencian la potencia de este tipo de tecnologías y la inmensa cantidad de aplicaciones posibles.