Desarrollo de un sistema digital para generación y filtrado de señales

Autor:
Ing. Juan Manuel Guariste

Materia:
Microarquitecturas y Softcores

Carrera de especialización en sistemas embebidos

Índice

Descripción del proyecto realizado **4**

Integración de los módulos del sistema **4**

Implementación del sistema **5**
3.1 Diseño de los bloques y organización de los módulos 6
3.2 Implementación del NCO y comunicación con el bus AXI 7
3.3 Implementación del filtro FIR 8
3.4 Uso de recursos 9

Registros de cambios

Revisión	Detalles de los cambios realizados	Fecha
0	Creación del documento	16 de junio de 2025

Descripción del proyecto realizado

El presente trabajo consiste en el desarrollo de un sistema digital implementado en la lógica programable (PL) de un dispositivo Zynq-7000, el cual integra módulos diseñados en VHDL que interactúan con el procesador embebido (PS) del mismo chip. La comunicación entre ambos dominios se realiza a través del bus AXI Lite, permitiendo modificar parámetros de funcionamiento del sistema desde software, sin necesidad de reconfigurar la FPGA.

El objetivo principal es generar una señal compuesta por la suma de dos senoidales de frecuencia configurable, aplicarle un filtrado digital en tiempo real mediante un filtro FIR, y analizar su comportamiento interno a través del Integrated Logic Analyzer (ILA). El diseño fue estructurado de forma modular, encapsulando cada funcionalidad en un IP Core específico, lo que favorece la escalabilidad, reutilización y depuración del sistema.

Integración de los módulos del sistema

En la figura 1 se presenta el diagrama en bloques que ilustra la arquitectura general del sistema.

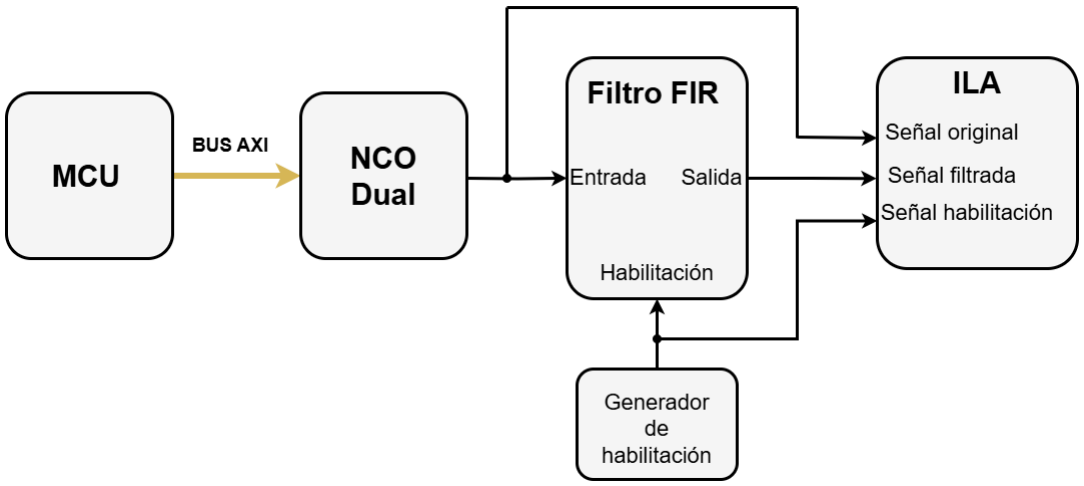

Figura 1. Arquitectura general del sistema.

Para organizar el diseño de forma escalable y reutilizable, cada uno de los módulos funcionales fue implementado como un *IP Core*. La integración se realizó mediante un diseño en bloques dentro de Vivado, donde se vinculan los componentes implementados en la lógica programable (PL) con el procesador embebido (PS) de la plataforma Zynq.

El bloque del Processing System (PS) (también referido como MCU) cumple un rol clave en el control dinámico del sistema. Este componente ejecuta software embebido que configura, a través del bus AXI Lite, los parámetros de funcionamiento de uno de los módulos de la lógica programable: el NCO dual. La comunicación entre PS y PL es posible gracias al mapeo de registros del IP del NCO dentro del espacio de direcciones accesible desde software.

A continuación, se detallan los módulos principales que componen la lógica programable del sistema:

- NCO Dual: este IP Core implementa dos osciladores controlados numéricamente (NCOs). Las señales generadas se suman internamente para obtener una única salida compuesta. Las frecuencias de ambas señales se controlan mediante dos registros: `paso_low` y `paso_high`, escritos desde el software del procesador a través del bus AXI Lite. Esta es la única instancia de comunicación activa entre la PS y la PL.
- Filtro FIR: bloque que implementa un filtro digital de tipo pasa bajos, diseñado para su operación autónoma dentro de la lógica programable. Aplica una convolución discreta sobre la señal de entrada utilizando coeficientes precargados. Dado que no requiere configuración en tiempo de ejecución, este IP no mantiene conexión directa con el PS.
- Generador de habilitación: módulo que genera una señal periódica de habilitación a partir del reloj del sistema. Su propósito es sincronizar la operación del filtro FIR, permitiendo reducir la frecuencia de muestreo y optimizar el uso de recursos.
- ILA: bloque de observación que permite capturar en tiempo real señales internas del sistema, como la entrada y salida del filtro o la señal de habilitación. Es una herramienta fundamental para la validación funcional sin necesidad de instrumentación externa.

Esta arquitectura demuestra una integración efectiva entre el software embebido y los bloques implementados en la PL. El procesador actúa como maestro del bus AXI, configurando parámetros que modifican el comportamiento del hardware en tiempo real. A su vez, la lógica programable ejecuta el procesamiento de señales de manera autónoma y determinista, garantizando alta velocidad y precisión en el dominio digital.

Implementación del sistema

La implementación del sistema se realizó en la herramienta de desarrollo Vivado de Xilinx. El diseño fue desarrollado de manera modular, encapsulando cada bloque funcional como un IP Core. Estos módulos se integraron mediante un esquema de diseño en bloques (Block Design), lo que permitió una organización clara y reutilizable.

Dado que no se disponía de una FPGA física local, el desarrollo y las pruebas se llevaron a cabo sobre una placa proporcionada por la cátedra, accesible de forma remota. Esta modalidad permitió cargar el bitstream, ejecutar validaciones funcionales y observar las señales internas mediante el ILA, todo sin intervención física directa sobre el dispositivo.

3.1. Diseño de los bloques y organización de los módulos

La figura 2 muestra el diseño en bloques generado en Vivado, donde se integran los componentes del sistema.

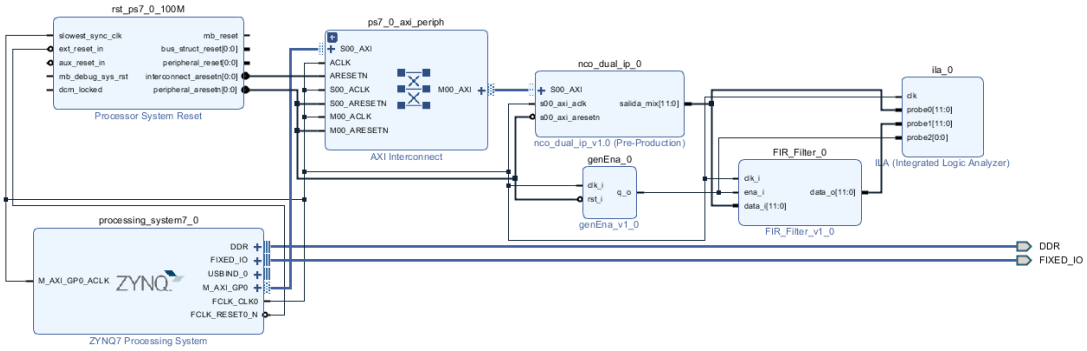

Figura 2. Diseño en bloques del sistema.

A partir de la figura 2 se pueden identificar los siguientes módulos:

- Zynq7 Processing System (PS): ejecuta el código embebido en C. Es el encargado de configurar en tiempo de ejecución los parámetros paso_low y paso_high que controlan la frecuencia de cada NCO. La escritura de estos valores se realiza mediante el bus AXI Lite.
- NCO Dual IP: bloque VHDL encapsulado como IP que genera una señal compuesta por la suma de dos senoidales. Su frecuencia es determinada por los registros internos paso_low y paso_high, accesibles mediante el bus AXI. Este módulo es el único que mantiene una interfaz directa con el PS.
- Filtro FIR IP: módulo autónomo que realiza el filtrado digital sobre la señal compuesta. Fue implementado como un IP sin interfaz AXI, ya que no requiere configuración dinámica. Los coeficientes del filtro están embebidos directamente en el diseño.
- genEna IP: genera pulsos periódicos de habilitación que sincronizan la operación del filtro FIR, desacoplando la tasa de muestreo de la frecuencia del reloj del sistema.
- ILA (Integrated Logic Analyzer): permite observar, en tiempo real, la evolución de señales internas, incluyendo la entrada y salida del filtro y la señal de habilitación.
- Interconexión AXI: el bus AXI Lite conecta el PS con el IP del NCO. Esta conexión permite la escritura de registros internos desde el procesador sin requerir intervención de lógica adicional o protocolos complejos.
- System Reset: módulo encargado de generar señales de reset sincronizadas para los bloques de la lógica programable. Este bloque se conecta directamente al PS y garantiza una inicialización coordinada del sistema.

3.2. Implementación del NCO y comunicación con el bus AXI

El desarrollo comenzó por el IP Core del NCO Dual, el cual incluye una interfaz AXI Lite que le permite al procesador escribir en tiempo de ejecución los parámetros de frecuencia.

En la figura 3 se muestra el código en lenguaje C implementado en el procesador, encargado de configurar los registros paso_low y paso_high. Este código utiliza funciones de la biblioteca xil_io.h para escribir directamente sobre los registros mapeados del IP.

```c
#include "xparameters.h"
#include "xil_io.h"

int main(void) {
    #define NCO_DUAL_SUM_IP_S00_AXI_SLV_REG0_OFFSET 0
    #define NCO_DUAL_SUM_IP_S00_AXI_SLV_REG1_OFFSET 4
    // Variables de paso para las dos senoidales
    int paso_low = 6;
    int paso_high = 62;
    xil_printf("-- Inicio del programa para configurar el NCO dual --\r\n");
    // Se escribe el paso_low en el registro 0 del NCO
    Xil_Out32((XPAR_NCO_DUAL_IP_0_S00_AXI_BASEADDR) + (NCO_DUAL_SUM_IP_S00_AXI_SLV_REG0_OFFSET), (u32)paso_low);
    // Se escribe el paso_high en el registro 1 del NCO
    Xil_Out32((XPAR_NCO_DUAL_IP_0_S00_AXI_BASEADDR) + (NCO_DUAL_SUM_IP_S00_AXI_SLV_REG1_OFFSET), (u32)paso_high);
    xil_printf("Se configuraron los pasos:\r\n");
    xil_printf("paso_low  = %d\r\n", paso_low);
    xil_printf("paso_high = %d\r\n", paso_high);
    xil_printf("-- Fin del programa --\r\n");
    return 0;
}
```

Figura 3. Código implementado en el procesador.

Para validar la correcta comunicación PS–PL a través del bus AXI, se realizaron diversas pruebas variando los valores de paso. La figura 4 muestra el diseño inicial compuesto únicamente por el procesador y el NCO.

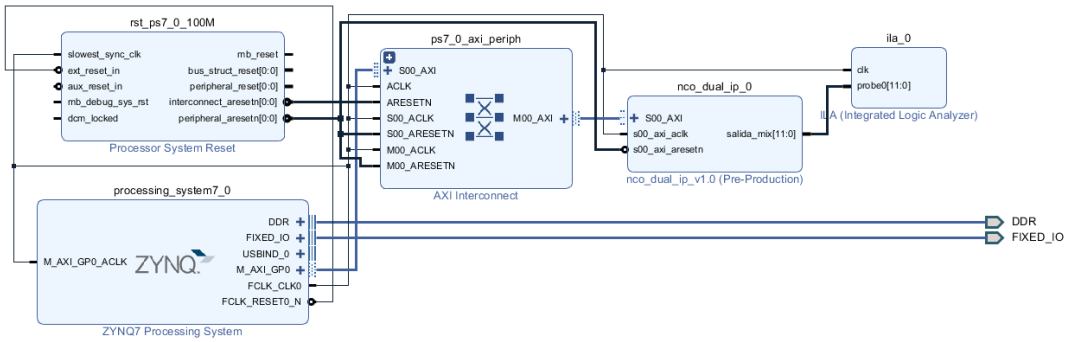

Figura 4. Diseño en bloques del procesador y el NCO.

A continuación se detallan algunas de las pruebas realizadas:

- Paso_low = 6, Paso_high = 0: solo se genera una senoidal de baja frecuencia (figura 5).
- Paso_low = 0, Paso_high = 62: se genera una senoidal de alta frecuencia (figura 6).
- Paso_low = 6, Paso_high = 62: ambas senoidales se combinan, generando una señal compuesta (figura 7).

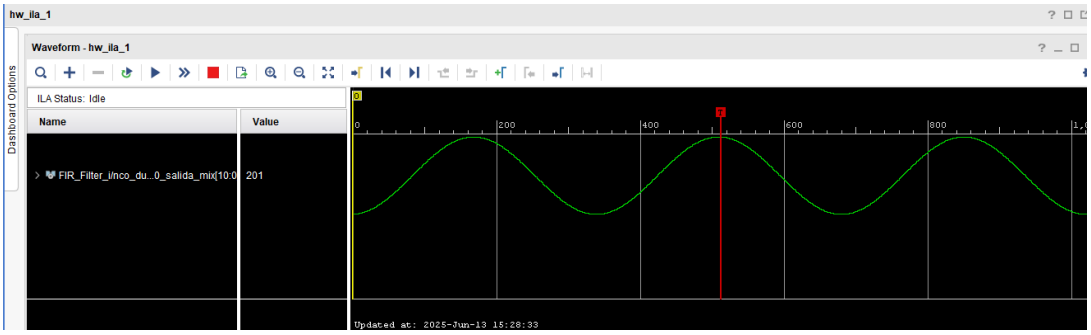

Figura 5. Variable paso_low = 6 y paso_high = 0.

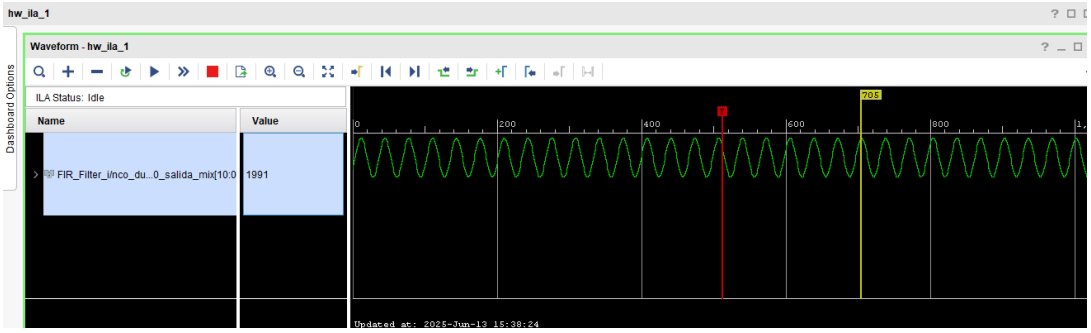

Figura 6. Variable paso_low = 0 y paso_high = 62.

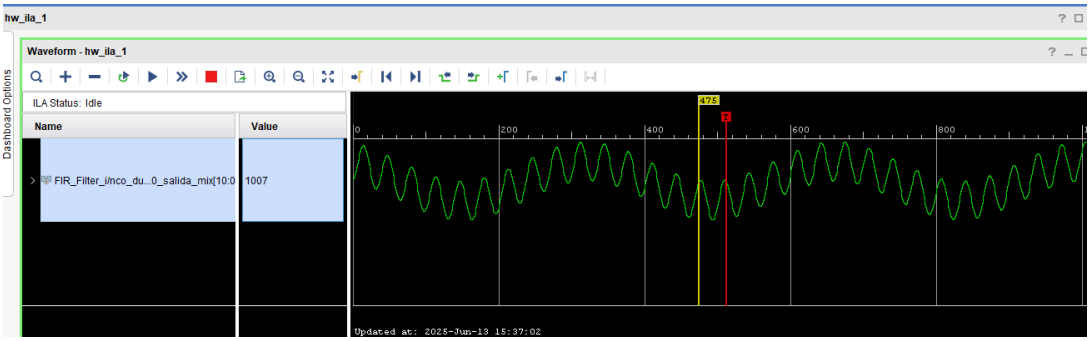

Figura 7. Variable paso_low = 6 y paso_high = 62.

3.3. Implementación del filtro FIR

Una vez verificada la correcta operación del NCO y la comunicación con el procesador, se procedió a integrar el IP del filtro FIR y el generador de habilitación (genEna).

En esta etapa se agregaron nuevas señales de observación al ILA, específicamente:

- Señal de entrada del filtro (salida del NCO dual).
- Señal de salida del filtro.
- Señal de habilitación.

Para validar el funcionamiento del sistema completo, se configuraron los parámetros del NCO con los valores paso_low = 6 y paso_high = 62. La señal compuesta generada por la suma de ambas senoidales fue aplicada como entrada al filtro FIR. El resultado puede observarse en la figura 8.

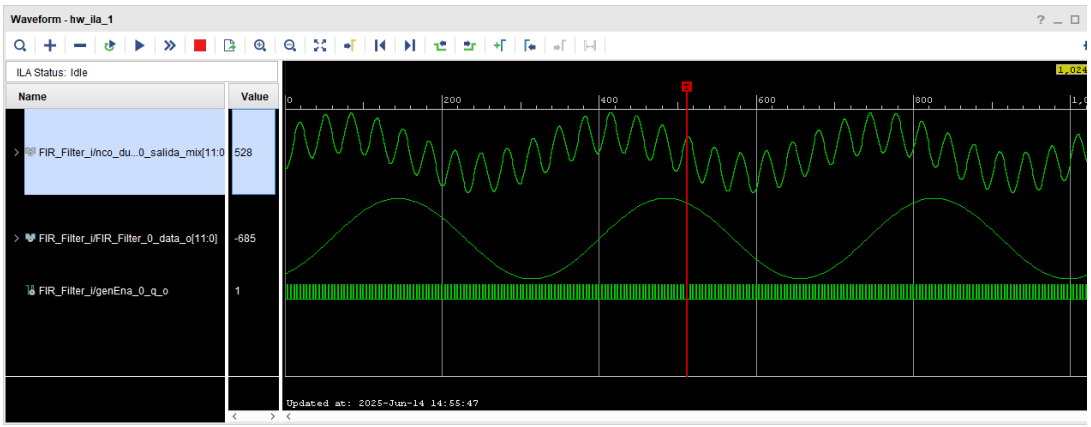

Figura 8. Variable paso_low = 6 y paso_high = 62.

En la salida del filtro se distingue claramente una senoidal de baja frecuencia, correspondiente al valor de paso_low. La componente de alta frecuencia, generada por paso_high, fue atenuada de manera efectiva, confirmando el correcto comportamiento del filtro FIR.

Estos resultados validan tanto la operación del sistema completo como la efectividad del filtrado digital en la lógica programable.

3.4. Uso de recursos

En la figura 9 se presenta la tabla de utilización de recursos de la FPGA una vez implementado el diseño.

Resource	Utilization	Available	Utilization %
LUT	1893	17600	10.76
LUTRAM	181	6000	3.02
FF	2571	35200	7.30
BRAM	1	60	1.67
DSP	21	80	26.25

Figura 9. Tabla de utilización de recursos.

A su vez, la figura 10 muestra una representación gráfica de los valores expuestos en la tabla anterior, permitiendo una visualización más clara de la distribución de recursos dentro del dispositivo.

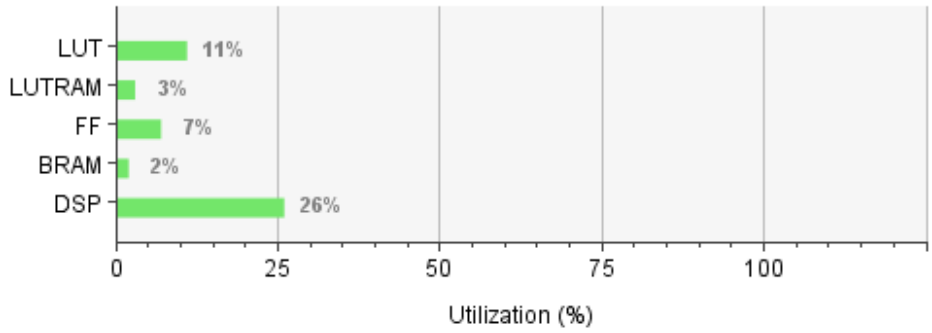

Figura 10. Gráfico de utilización de recursos.