Neiva, Huila

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

El (Los) suscrito(s):

Alejandro Ule Duque, con C.C. No. 1003806368,

Autor(es) de la tesis y/o trabajo de grado

Titulado FILTRO DIGITAL IIR IMPLEMENTADO EN FPGA DE0 – NANO PARA ELIMINACIÓN DEL RUIDO GAUSSIANO PRESENTE EN SEÑAL DETERMINÍSTICA DE LA DENSIDAD POBLACIONAL DE NEUTRONES presentado y aprobado en el año 2024 como requisito para optar al título de INGENIERO ELECTRÓNICO; Autorizo (amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que, con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con las cuales tenga convenio la Institución.

- Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que, de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, “Los derechos morales sobre el trabajo son propiedad de los autores” , los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

EL AUTOR/ESTUDIANTE:

Firma: 

TÍTULO COMPLETO DEL TRABAJO:

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
ULE DUQUE	ALEJANDRO

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
MOSQUERA CERQUERA	VLADIMIR
SUESCÚN DÍAZ	DANIEL

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre
MOLINA MOSQUERA	JOHAN JULIÁN
RAMÍREZ GUTIÉRREZ	JULIÁN ADOLFO

PARA OPTAR AL TÍTULO DE: INGENIERO ELECTRÓNICO

FACULTAD: INGENIERÍA

PROGRAMA O POSGRADO: INGENIERÍA ELECTRÓNICA

CIUDAD: NEIVA **AÑO DE PRESENTACIÓN:** 2024 **NÚMERO DE PÁGINAS: 53**

TIPO DE ILUSTRACIONES (Marcar con una **X**)**:**

Diagramas X Fotografías X Grabaciones en discos___ Ilustraciones en general___ Grabados___ Láminas___ Litografías___ Mapas___ Música impresa___ Planos___ Retratos___ Sin ilustraciones___ Tablas o Cuadros X

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

	Español	**Inglés**
1.	**REACTIVIDAD NUCLEAR**	**NUCLEAR REACTIVITY**

2. RS-232 RS-232

3. FPGA FPGA

4. MICROPROCESADOR MICROPROCESSOR

5. FILTRO IIR IIR FILTER

6. FILTRO FIR FIR FILTER

7. UART UART

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

Este trabajo desarrolla el diseño y la implementación de un filtro digital con respuesta infinita al impulso (IIR) en la FPGA DE0 – Nano de Altera. El diseño se realiza a partir de un filtro analógico Chebyshev con rizado en la banda pasante, y luego se aplica la transformación bilineal para su digitalización. Se establece la etapa de filtrado con el fin de eliminar el ruido gaussiano presente en la señal determinística de densidad poblacional de neutrones, la cual es necesaria para realizar el cálculo de reactividad nuclear. El cálculo de este parámetro se realiza a partir de la ecuación inversa de la cinética puntual, la cual resulta de resolver las ecuaciones de cinética puntual que modelan la dinámica de un reactor nuclear de potencia. La ecuación inversa de cinética puntual es una ecuación integro diferencial en donde está presente la integral correspondiente al histórico de la densidad poblacional de neutrones; esta integral se resuelve por medio del método de filtro FIR (Finite Impulse Response). También desarrolla el diseño y la implementación de un sistema basado en el procesador Nios II/f, el cual contiene diferentes núcleos como memoria RAM, núcleo de protocolo UART con el estándar RS-232, generador de señal de reloj y controlador en paralelo de entradas y salidas del chip.

ABSTRACT: (Máximo 250 palabras)

This work develops the design and implementation of a digital filter with infinite impulse response (IIR) in the Altera DE0 - Nano FPGA. The design is performed starting from an analog Chebyshev filter with ripple in the passband, and then the bilinear transformation is applied for its digitization. The filtering stage is established in order to eliminate the Gaussian noise present in the deterministic neutron population density signal, which is necessary to perform the nuclear reactivity calculation. The calculation of this parameter is performed from the inverse equation of point kinetics, which results from solving the equations of point kinetics that model the dynamics of a nuclear power reactor. The inverse equation of point kinetics is a differential integral equation where the integral corresponding to the historical neutron population density is present; this integral is solved by means of the FIR (Finite Impulse Response) filter method. It also develops the design and implementation of a system based on the Nios II/f processor, which contains different cores such as RAM memory, UART protocol core with the RS-232 standard, clock signal generator and parallel controller of inputs and outputs of the chip.

APROBACION DE LA TESIS

Nombre Presidente Jurado: Vladimir Mosquera Cerquera

Firma: Vladimir Mosquera C.

Nombre Presidente Jurado: Daniel Suescún Díaz

Firma

Nombre Jurado: Julian Adolfo Ramirez Gallegos

Firma:

Nombre Jurado: Johan Julian Molina Mosquera

Firma:

FILTRO DIGITAL IIR IMPLEMENTADO EN FPGA DE0 – NANO PARA ELIMINACIÓN DEL RUIDO GAUSSIANO PRESENTE EN SEÑAL DETERMINÍSTICA DE LA DENSIDAD POBLACIONAL DE NEUTRONES

ALEJANDRO ULE DUQUE

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA – HUILA
2023

FILTRO DIGITAL IIR IMPLEMENTADO EN FPGA DE0 – NANO PARA ELIMINACIÓN DEL RUIDO GAUSSIANO PRESENTE EN SEÑAL DETERMINÍSTICA DE LA DENSIDAD POBLACIONAL DE NEUTRONES

ALEJANDRO ULE DUQUE

DIRECTOR DE TESIS
Mag. VLADIMIR MOSQUERA CERQUERA

CODIRECTOR DE TESIS
Dr. DANIEL SUESCÚN DÍAZ

Trabajo de grado para optar por el título de
INGENIERO ELECTRÓNICO

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA – HUILA
2023

DEDICATORIA

A mis padres, Alejandro Ule Chávarro y Francia Eyenid Duque Ramos. Doy gracias a Dios por permitirme ser hijo de ustedes, son el regalo más grande que he recibido en toda mi vida. Gracias por ser mi mayor fuente de inspiración y por el amor incondicional que me brindan. Espero poderlos disfrutar por muchos años más. Su constante apoyo, sabiduría y sacrificio han sido un pilar fundamental en mi camino hacia cada logro.

A mis hermanas Geraldyne Ule Duque y Yaneiry Ule Duque. Gracias por motivarme a continuar hacia cada logro personal y académico. Gracias por sus consejos y por ser el ejemplo que debo seguir. Estoy muy agradecido de tenerlas en mi vida.

Este triunfo es de la Familia Ule Duque. Los amo con todo mi corazón.

AGRADECIMIENTOS

Principalmente a Dios, por brindarme salud y sabiduría. Él es mi fortaleza y mi escudo, en él confío mi corazón y de él recibo ayuda. Me sostiene en los momentos difíciles y me brinda palabras de aliento.

A mi director de tesis Vladimir Mosquera Cerquera. Gracias por la guía y orientación a lo largo de este arduo pero gratificante proceso. También por el conocimiento y la dedicación para compartirlo.

A mi codirector de tesis Daniel Suescún Díaz. Gracias por encaminarme en la investigación; por el tiempo y el conocimiento brindado. Gracias por brindarme la oportunidad de trabajar en conjunto y por confiar en mi capacidad para llevar a cabo este proyecto.

TABLA DE CONTENIDO

pág.

LISTA DE FIGURAS

pág.

LISTA DE TABLAS

pág.

ABREVIACIONES

FPGA: Field Programmale Gate Arrays.
PCM: Per Cent Mille.
PC: Personal Computer.
IIR: Infinite Impulse Response.
FIR: Finite Impulse Response.
UART: Universal Asynchronous Receiver/Transmitter
RS-232: Recommended Standard 232.
RAM: Random Access Memory.
ICs: Integrated Circuits.
RISC: Reduced Instruction Set Computer.
DMIPS: Dhrystone Million Instruction Per Second.
ISA: Instruction Set Architecture.
MMU: Memory Management Unit.
MPU: Memory Protection Unit.
JTAG: Join Test Action Group.

RESUMEN

Este trabajo desarrolla el diseño y la implementación de un filtro digital con respuesta infinita al impulso (IIR) en la FPGA DE0 – Nano de Altera. El diseño se realiza a partir de un filtro analógico Chebyshev con rizado en la banda pasante, y luego se aplica la transformación bilineal para su digitalización. Se establece la etapa de filtrado con el fin de eliminar el ruido gaussiano presente en la señal determinística de densidad poblacional de neutrones, la cual es necesaria para realizar el cálculo de reactividad nuclear. El cálculo de este parámetro se realiza a partir de la ecuación inversa de la cinética puntual, la cual resulta de resolver las ecuaciones de cinética puntual que modelan la dinámica de un reactor nuclear de potencia. La ecuación inversa de cinética puntual es una ecuación integro diferencial en donde está presente la integral correspondiente al histórico de la densidad poblacional de neutrones; esta integral se resuelve por medio del método de filtro FIR (Finite Impulse Response).

También desarrolla el diseño y la implementación de un sistema basado en el procesador Nios II/f, el cual contiene diferentes núcleos como memoria RAM, núcleo de protocolo UART con el estándar RS-232, generador de señal de reloj y controlador en paralelo de entradas y salidas del chip.

Palabras clave: Reactividad Nuclear, FPGA, Filtro IIR, Filtro FIR, UART, RS-232, Microprocesador.

ABSTRACT

This work develops the design and implementation of a digital filter with infinite impulse response (IIR) in the Altera DE0 - Nano FPGA. The design is performed starting from an analog Chebyshev filter with ripple in the passband, and then the bilinear transformation is applied for its digitization. The filtering stage is established in order to eliminate the Gaussian noise present in the deterministic neutron population density signal, which is necessary to perform the nuclear reactivity calculation. The calculation of this parameter is performed from the inverse equation of point kinetics, which results from solving the equations of point kinetics that model the dynamics of a nuclear power reactor. The inverse equation of point kinetics is a differential integral equation where the integral corresponding to the historical neutron population density is present; this integral is solved by means of the FIR (Finite Impulse Response) filter method.

It also develops the design and implementation of a system based on the Nios II/f processor, which contains different cores such as RAM memory, UART protocol core with the RS-232 standard, clock signal generator and parallel controller of inputs and outputs of the chip.

Keywords: Nuclear Reactivity, FPGA, IIR Filter, FIR Filter, UART, RS-232, Microprocessor.

1. INTRODUCCIÓN

A causa del acelerado aumento de la población mundial, la búsqueda de soluciones energéticas sostenibles se ha vuelto primordial. La necesidad de suplir la demanda energética sin comprometer la integridad del planeta se ha convertido en un desafío apremiante. El aumento de las emisiones de dióxido de carbono y los problemas asociados, como el calentamiento global y la escasez de recursos hídricos, exigen la búsqueda de alternativas que sean respetuosas con el medio ambiente. Es fundamental adoptar fuentes de energía amigables con el entorno para garantizar un futuro sustentable y preservar un ambiente saludable para generaciones venideras.

Las centrales nucleares destacan como una opción altamente confiable, debido a que, a diferencia de otras fuentes energéticas, ésta no genera emisión de gases de efecto invernadero y puede generar grandes cantidades de electricidad de manera segura y controlada. Esto se logra mediante el uso controlado del proceso de fisión nuclear; proceso durante el cual los átomos se descomponen liberando energía en forma de calor, el cual es utilizado para calentar agua y producir vapor, que posteriormente, es impulsado a presión para girar las turbinas de un generador eléctrico.

No obstante, como sucede con cualquier tecnología que involucre la generación masiva de energía, la seguridad se convierte en un aspecto fundamental. Los efectos de un accidente nuclear son de gran magnitud; incluyen la emisión de radiación perjudicial al entorno, daños a infraestructuras, grandes pérdidas económicas y, en el peor de los casos, posibles pérdidas humanas. Por consiguiente, las centrales nucleares están sometidas a rigurosas normas y protocolos de seguridad para garantizar su funcionamiento en un entorno protegido y seguro.

En lo que respecta a la seguridad de una central nuclear, conocer la reactividad es de gran importancia; controlar esta variable de manera precisa es fundamental para garantizar una operación estable y segura del reactor. Aunque no es posible medir directamente esta variable con un sensor, se puede calcular utilizando la ecuación inversa de la cinética puntual. Para realizar este cálculo, se requiere como entrada, la señal de densidad poblacional de neutrones, la cual se puede medir mediante cámaras de ionización ubicadas cerca del núcleo del reactor. Sin embargo, esta señal de entrada es afectada por un ruido gaussiano, lo que hace necesario implementar una etapa de filtrado antes de realizar el cálculo. Es importante destacar que la señal de densidad poblacional de neutrones está directamente relacionada con la potencia eléctrica generada, lo que permite también, calcular la reactividad a partir de este parámetro.

Para realizar el proceso de filtrado de manera eficiente, es recomendable utilizar un procesador especializado en el tratamiento digital de señales, que esté equipado

con núcleos y bloques diseñados específicamente para llevar a cabo cálculos numéricos intensivos. En este proyecto, se ha optado por implementar un filtro digital con respuesta finita al impulso en un sistema basado en el procesador Nios II/f. Este sistema se encuentra embebido en la placa de desarrollo DE0-Nano, que cuenta con la FPGA Cyclone IV EP4CE22F17C6N. Además, se ha establecido comunicación serial entre el dispositivo y el ordenador para facilitar la transferencia de datos de manera eficiente.

2. OBJETIVOS

2.1. Objetivo General

Implementar un filtro digital Chebyshev tipo 1 en la FPGA DE0-Nano para atenuar el ruido gaussiano presente en una señal de prueba determinística correspondiente a la potencia eléctrica, con el fin de utilizar la señal resultante para calcular la reactividad nuclear mediante un filtro con respuesta finita al impulso.

2.2. Objetivos Específicos

- Realizar una revisión bibliográfica sobre filtros digitales y sus aplicaciones en el procesamiento digital de señales; también sobre el cálculo de la reactividad nuclear y el diseño e implementación de reactímetros.

- Estudiar el funcionamiento y características del filtro digital Chebyshev tipo 1 y su implementación en la FPGA DE0-Nano.

- Realizar el diseño del sistema basado en el procesador Nios II con ayuda de la herramienta Qsys del software Quartus II.

- Diseñar el filtro IIR basado en el prototipo analógico por medio de su análisis en frecuencia y utilizar la transformación bilineal para su digitalización.

- Realizar pruebas y ajustes del filtro digital para optimizar su rendimiento en la atenuación del ruido gaussiano presente en la señal de potencia.

- Establecer la comunicación serial entre el computador y la FPGA mediante el protocolo RS-232 de comunicación UART para facilitar el envío de la señal resultante y su posterior análisis.

- Desarrollar el código en lenguaje C que realice el filtrado de la señal de potencia mediante la ecuación en diferencias del filtro Chebyshev tipo 1 haciendo uso de la herramienta *Nios II Sooftware Build Tools for Eclipse*; también que efectúe la recepción de la señal de potencia con ruido y la transmisión la señal filtrada.

- Evaluar el desempeño del filtro digital implementado en términos de precisión y eficiencia en la atenuación del ruido gaussiano.

3. MARCO TEÓRICO

Para el desarrollo de este proyecto, es necesario comprender conceptos relacionados con la energía nuclear, la física computacional y el procesamiento digital de señales.

Es importante desarrollar el concepto de la Ecuación Inversa de la Cinética Puntual, que es la ecuación matemática que permite realizar el cálculo de la reactividad nuclear a partir de la señal de densidad poblacional de neutrones. También se debe analizar el por qué es necesario añadir una etapa de filtrado en el sistema procesamiento de señales en un reactor de potencia.

Es preciso estudiar los fundamentos de las FPGA, descripción y principios del procesador Nios II, y las bases teóricas de los filtros digitales. Además de esto, también es importante comprender las características de los filtros IIR, como su diagrama de bloques y la ecuación en diferencias que lo define.

4.1. Ecuación Inversa de la Cinética Puntual

Las ecuaciones que describen la dinámica de un reactor son conocidas como las ecuaciones de la cinética puntual. Estas ecuaciones, obtenidas a partir de la ecuación de difusión de neutrones, representan la evolución temporal de la distribución de neutrones y la concentración de precursores de neutrones retardados, donde la reactividad es una variable en función del tiempo y se denota $\rho(t)$ [4]. Las ecuaciones de la cinética puntual son las siguientes:

$$\frac{dP(t)}{dt} = \left[\frac{\rho(t) - \beta}{\Lambda}\right] P(t) + \sum_{i=1}^{6} \lambda_i C_i(t) \tag{4.1.1}$$

$$\frac{dC_i(t)}{dt} = \frac{\beta_i}{\Lambda} P(t) - \lambda_i C_i(t) \tag{4.1.2}$$

Con condiciones iniciales:

$$P(t = 0) = P_0 \tag{4.1.3}$$

$$C_i(t = 0) = \frac{\beta_i}{\Lambda \lambda_i} P_0 \tag{4.1.4}$$

Donde $C_i(t)$ es la concentración del $i-ésimo$ grupo de precursores de neutrones retardados; $P(t)$ es la densidad poblacional de neutrones (directamente proporcional a la potencia nuclear); $\rho(t)$ es la reactividad nuclear; Λ es el tiempo de generación de neutrones; β_i es la fracción efectiva del $i-ésimo$ grupo de neutrones retardados; β es la fracción efectiva total de neutrones retardados ($\beta = \sum \beta_i$); λ_i es la constante de decaimiento del $i-ésimo$ grupo de precursores de neutrones retardados.

Para hallar la ecuación inversa de la cinética puntual, primero se despeja $\rho(t)$ de la ecuación (4.1.1)

$$\rho(t) = \frac{\Lambda}{P(t)}\frac{dP(t)}{dt} - \frac{\Lambda}{P(t)}\sum_{i=1}^{6}[\lambda_i C_i(t)] + \beta \tag{4.1.5}$$

Sin embargo, es necesario conocer $C_i(t)$; para ello, se despeja este parámetro de la ecuación (4.1.2)

$$\frac{dC_i(t)}{dt} + \lambda_i C_i(t) = \frac{\beta_i}{\Lambda}P(t) \tag{4.1.6}$$

Para resolver esta ecuación diferencial, se aplica el método del factor integrante y se obtiene como resultado

$$C_i(t) = e^{-\int \lambda_i dt}\left[\int \frac{\beta_i}{\Lambda}P(t)e^{\lambda_i t}dt + C\right] \tag{4.1.7}$$

Donde C es la condición inicial de la ecuación diferencial. Reemplazando C y definiendo la integral, resulta

$$C_i(t) = e^{-\lambda_i t}\left[\int_0^t \frac{\beta_i}{\Lambda}P(t')e^{\lambda_i t'}dt' + \frac{\beta_i}{\Lambda\lambda_i}P_0\right] \tag{4.1.8}$$

Resolviendo y reacomodando términos

$$C_i(t) = \frac{\beta_i}{\Lambda}\int_0^t P(t')e^{-\lambda_i(t-t')}dt' + \frac{\beta_i}{\Lambda\lambda_i}P_0 e^{-\lambda_i t} \tag{4.1.9}$$

Reemplazando la ecuación (4.1.9) en la ecuación (4.1.5) se llega a la Ecuación Inversa de la Cinética Puntual

$$\rho(t) = \beta + \frac{\Lambda}{P(t)}\frac{dP(t)}{dt} - \frac{P_0}{P(t)}\sum_{i=1}^{6}\beta_i e^{-\lambda_i t} - \frac{1}{P(t)}\sum_{i=1}^{6}\int_0^t \lambda_i \beta_i P(t') e^{-\lambda_i (t-t')} dt' \quad (4.1.10)$$

Sin embargo, resolver esta ecuación para cualquier forma de $P(t)$ supondría un coste computacional altísimo; es por esto, que la literatura propone métodos de aproximación para resolver esta ecuación integro-diferencial, uno de ellos el método de Filtro FIR desarrollado y propuesto en el artículo *Calculation of reactivity using a Finite Impulse Response Filter* que realiza la aproximación con la siguiente ecuación

$$\rho(t) = \beta + \frac{\Lambda}{P(t)}\frac{dP(t)}{dt} - \frac{P_0}{P(t)}\sum_{i=1}^{6}\beta_i e^{-\lambda_i t} - \frac{T}{P(t)}\left[\left(\sum_{i=6}^{6}\lambda_i \beta_i h_i\right) * P(t)\right] \quad (4.1.11)$$

En donde T es el periodo de muestreo de la señal y h_i es la respuesta al impulso de la densidad poblacional de neutrones.

Se simula esta ecuación en Matlab y de esta manera se calcula la reactividad nuclear para cualquier señal de entrada en el sistema, que, para este proyecto, es de forma exponencial.

4.2. Ruido Gaussiano en la señal determinística

Para superar los retos de seguridad que supone la energía nuclear, las centrales nucleares cuentan con sofisticados sistemas de control e instrumentación que mantiene la operación del reactor dentro de los límites seguros. Para medir la densidad poblacional de neutrones, los reactores cuentan con unas cámaras de ionización cerca al núcleo. Estas cámaras están principalmente compuestas por un volumen de gas y dos electrodos ligeramente separados, que son alimentados con un potencial constante mediante una batería. Cuando la cámara es alcanzada por una dosis suficientemente elevada de radiación ionizante, los iones y electrones producidos son emitidos continuamente hacia los electrodos, lo que produce una corriente constante. La intensidad de la corriente iónica constituye una medida directa de la cantidad partículas ionizantes que ingresan a la cámara y, en consecuencia, de la intensidad de la radiación [5]. El problema radica en que la corriente generada es afectada por un ruido gaussiano, producido por la aleatoriedad de las partículas ionizantes que salen del núcleo del reactor.

Para simular este ruido gaussiano, se crea una función que calcule el promedio de la señal de potencia cada diez muestras; el resultado se multiplica por un valor de desviación estándar y un número aleatorio extraído de la distribución normal estándar (para ello se emplea la función *randn* de Matlab).

4.3. Filtro Digital IIR

Los filtros IIR, a diferencia de los filtros FIR, son filtros recursivos, lo que quiere decir que la salida en un instante determinado depende de la salida en instantes anteriores, lo que quiere decir que existe una realimentación entre los datos de salida y entrada [1]. Como consecuencia a lo anterior, su respuesta al impulso se debe calcular recursivamente, y por lo general es infinita. Su ecuación en diferencias está dada por la siguiente ecuación:

$$y[n] = -\sum_{i=1}^{N} a_k y[n-i] + \sum_{j=0}^{M} b_k x[n-j] \qquad (4.3.1)$$

El esquema gráfico del filtro IIR está dado de la siguiente forma:

Fuente: Tratamiento Digital de Señales. John G. Proakis & Dimitris G. Manolakis

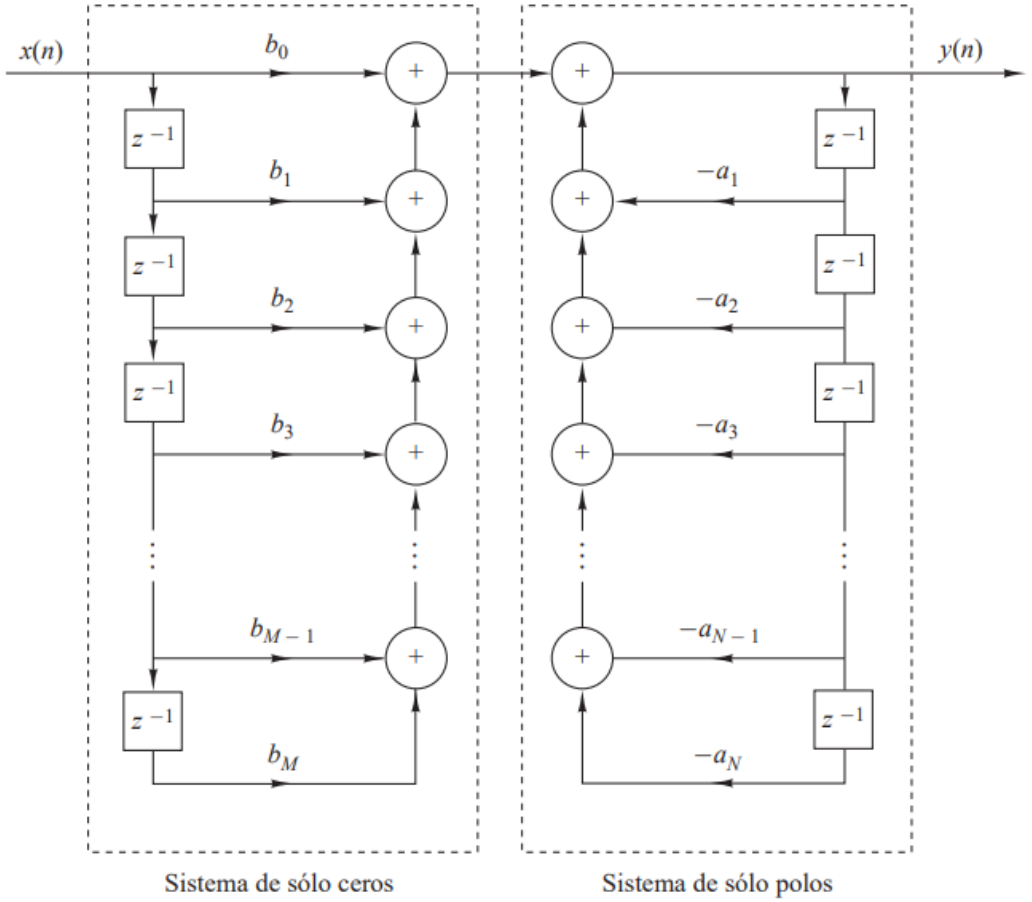

Figura 1. Diagrama de Bloques de Filtro IIR.

Dentro de las principales ventajas que tienen los filtros IIR se encuentra que se consiguen filtros con mayor factor de atenuación con un menor número de coeficientes; esto se debe a la recursividad de su estructura, lo que permite tener ceros y polos en su función de transferencia. Otra de sus ventajas es que tienen una muy buena resolución a bajas frecuencias; a diferencia de los filtros FIR, que para bajas frecuencias requieren un gran número de coeficientes produciendo un retardo significativo. A parte de esto, los filtros IIR pueden ser diseñados a partir de prototipos analógicos, como por ejemplo filtros Butterworth, Chebyshev, Elíptico o Bessel; lo cual representa una gran ventaja debido a que la electrónica analógica ya se ha desarrollado durante bastante tiempo y con muy buenos resultados.

Sin embargo, estos filtros también presentan algunos inconvenientes en su diseño; como, por ejemplo, que la presencia de polos puede producir inestabilidades, lo cual requiere un buen estudio de éste parámetro a través de la transformada Z. Además, los filtros IIR no garantizan que la fase de su función de transferencia sea lineal y, al ser filtros recursivos, este tipo de filtro digital requiere una implementación en hardware más compleja que en el caso de los filtros FIR. Finalmente, en comparación con los filtros FIR, éstos son más afectados por los problemas de redondeo; y se debe a que la posición de los polos y los ceros no es precisamente en donde se calculó, sino que, están ligeramente movidos debido a la aproximación.

4.4. Field Programmable Gate Arrays (FPGA)

Field Programmable Gate Arrays (FPGA) son una serie de circuitos integrados (ICs – Integrated Circuits) que contienen bloques conformados por compuertas lógicas configurables (o programables) junto con interconexiones igualmente configurables entre estos bloques.

Fuente: Architecture of Field-Programmable Gate Arrays. (Jonathan Rose., et al.)

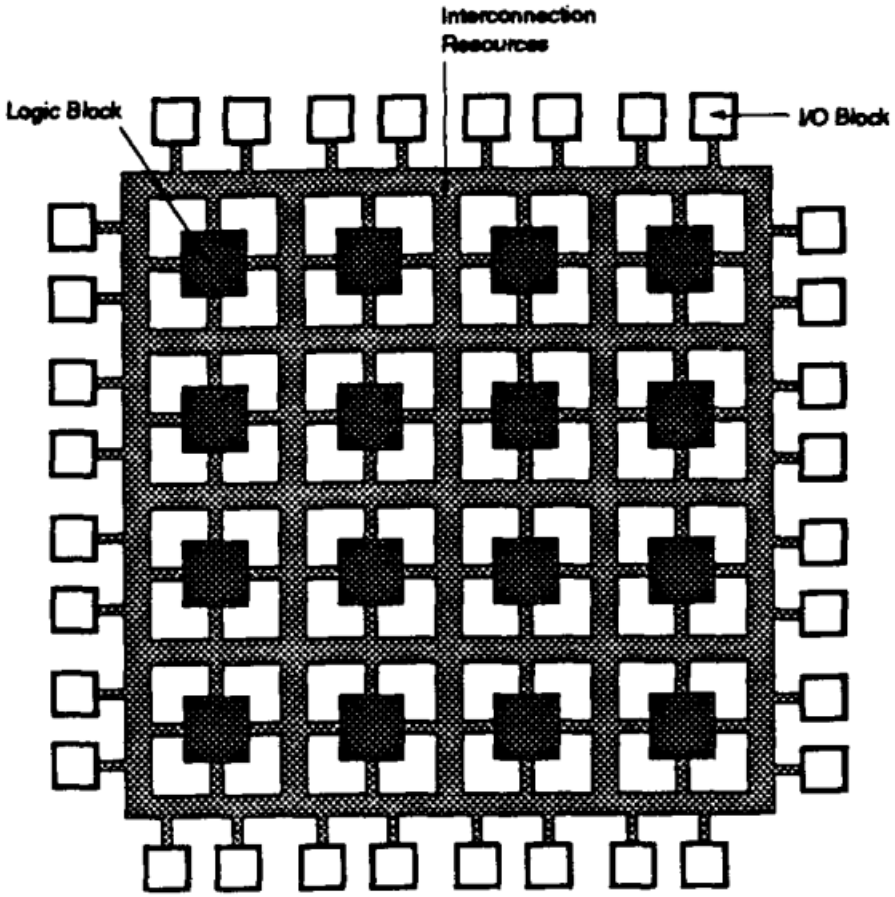

Figura 2. Arquitectura de un chip FPGA

Dependiendo de la forma en la que estén fabricadas, algunas FPGA pueden programarse una sola vez, sin embargo, también existen FPGA que son reprogramables.

La tarjeta de desarrollo que se usó para la implementación de este proyecto fue la FPGA DE0 – Nano desarrollada por Altera/Intel, la cual contiene el chip FPGA Altera Cyclone IV EP4CE22F17C6N.

Fuente: DE0-Nano User Manual. Altera.

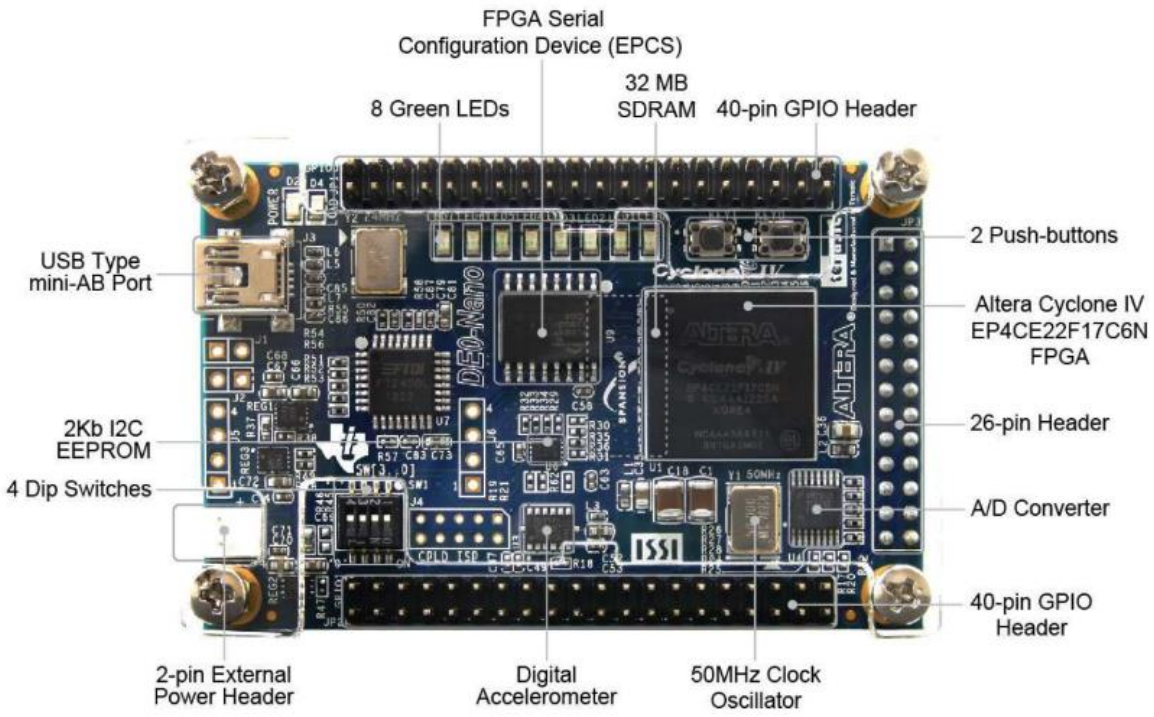

Figura 3. Placa de Desarrollo DE0-Nano

La tarjeta DE0-Nano brinda una plataforma de desarrollo FPGA de tamaño compacto, muy adecuada para una amplia gama de proyectos de diseño portátil. Y gracias a sus periféricos, es una muy buena elección al momento de incursionar en el mundo de estos chips.

4.4.1 Procesador Nios II/f

El procesador Nios II/f es un procesador desarrollado por Altera/Intel con estructura RISC (Reduced Instruction Set Computer). A parte de ser un microprocesador altamente configurable, es una buena herramienta para aplicaciones de procesado digital de señales.

Este procesador cuenta con un conjunto completo de instrucciones de 32 bits, ruta de datos y espacio de direcciones. Además, contiene 32 registros de propósito general y conjuntos opcionales de registro sombra. En total, cuenta con 32 fuentes de interrupción; sin embargo, lo que lo hace un procesador fuerte para el procesamiento digital de datos, es que cuenta con un conjunto de instrucciones únicamente dedicas a calcular productos de multiplicación de 64 y 128 bits; además de contar también con instrucciones opcionales de punto flotante para operaciones de punto flotante de precisión simple.

Este procesador también cuenta con acceso a distintos periféricos del chip, e interfaces a memorias y periféricos fuera del chip, lo que lo hace especialmente rápido en la recepción y envío de datos. También contiene un desplazador de barril de una sola instrucción, lo que permite realizar desplazamiento de bits de manera rápida y eficiente. La función principal del desplazador de barril es desplazar una secuencia de bits hacia la izquierda o hacia la derecha, de forma paralela y en un solo ciclo de reloj. Debido a esto, su rendimiento es de hasta 250 DMIPS (Dhrystone Million Instruction Per Second).

Respecto a sus unidades, el procesador Nios II cuenta con Unidad de Aministración de memoria (del inglés Memory Management Unit, MMU) opcional, esto con el fin de admitir sistemas operativos que lo requieran. También cuenta con Unidad de Protección de Memoria y con Unidad de Protecciones de Memoria (del inglés Memory Protection Unit, MPU).

El procesador cuenta con una arquitectura de Conjunto de Instrucciones (del inglés Instruction Set Architecture, ISA), y está definida por las siguientes unidades:

- Archivo de Registro.
- Unidad Aritmética Lógica (Arithmetic Logic Uniy, ALU).
- Interfaz para la lógica de instrucciones personalizadas.
- Controlador de Excepciones.
- Controlador de interrupciones interno o externo.
- Bus de instrucciones.
- Bus de datos.
- Unidad de Gestión de Memoria. (Memory Management Unit, MMU).
- Unidad de Protección de Memoria (Memory Protection Unit, MPU).
- Memoria caché de instrucciones y datos.
- Interfaces de memoria acoplada estrechamente para instrucciones y datos.
- Módulo de depuración JTAG (Join Test Action Group).

Fuente: Nios II Processor Reference. Intel.

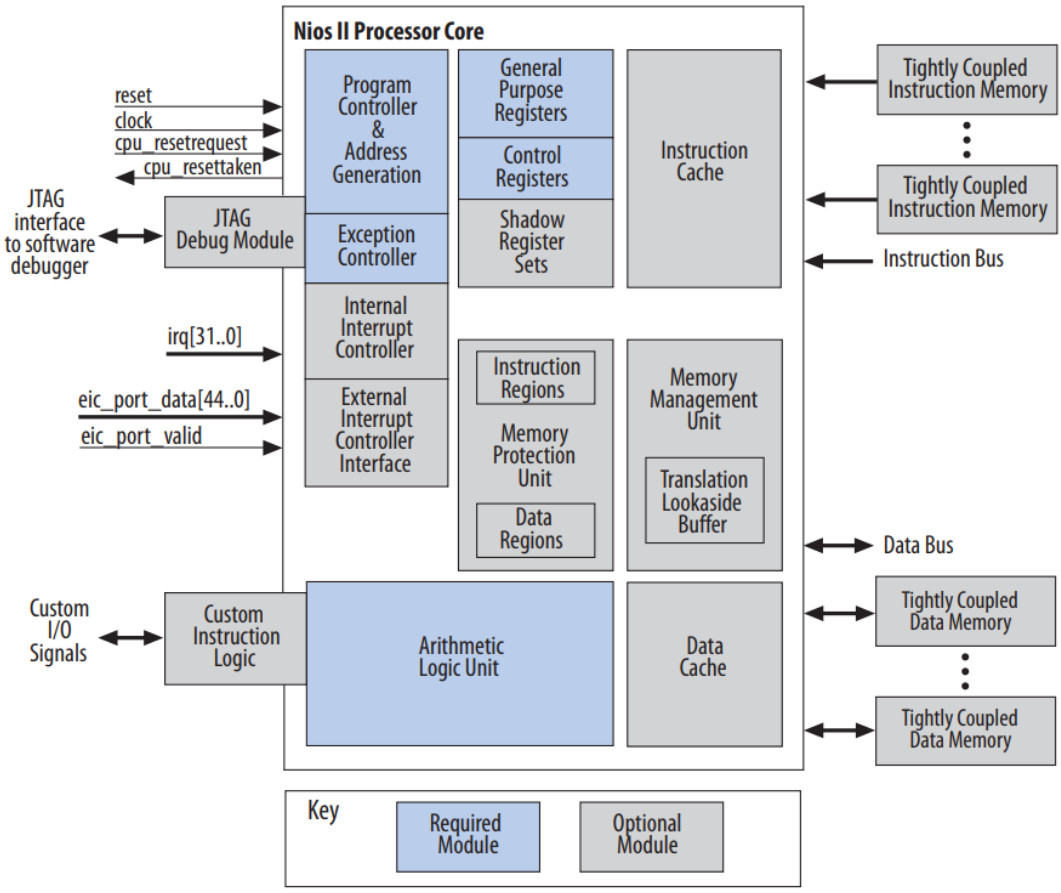

Figura 4. Diagrama de Bloques del Núcleo del Procesador Nios II.

El procesador cuenta con un entorno de desarrollo de software basado en la cadena de herramientas GNU C/C++ y las Herramientas de Compilación de Software por Eclipse (del inglés Software Build Tools, SBT for Eclipse). Esto supone una gran ventaja debido a que facilita su uso y programación.

Sin embargo, este procesador puede ser configurado con diferentes núcleos, es decir, se puede escoger entre los núcleos Nios II/e, Nios II/s o Nios II/f. Para esta aplicación se decide utilizar el núcleo Nios II/f que cuenta con el mayor número de ventajas.

Fuente: Autor.

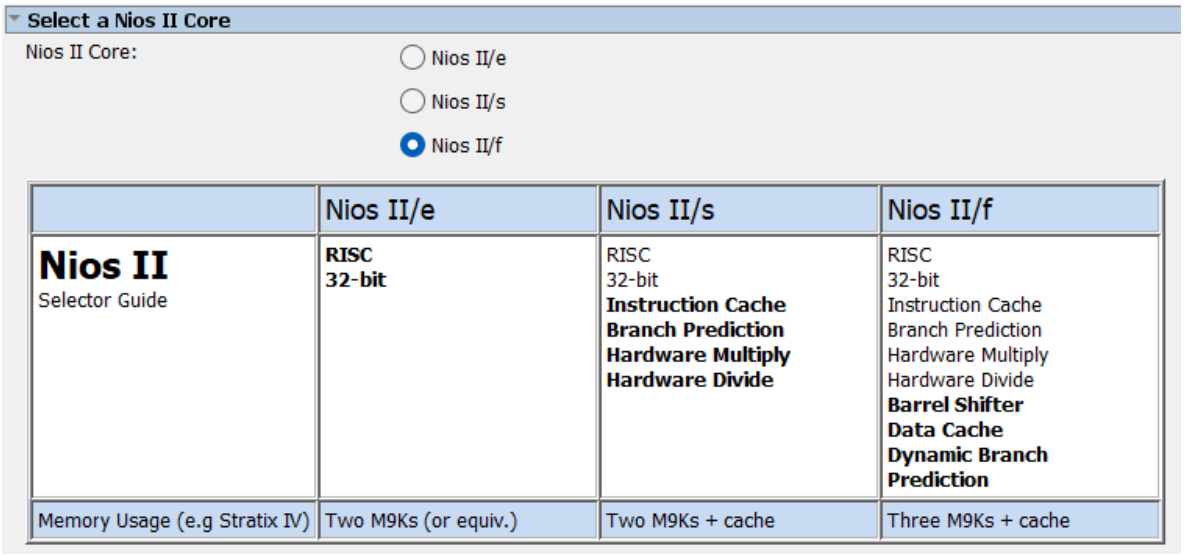

Figura 5. Configuración del núcleo del Procesador Nios II.

En comparación con el núcleo Nios II/s, el núcleo Nios II/f incorpora el desplazador de barril, caché de datos, rama dinámica y predicción. Este núcleo es la mejor opción que se puede elegir y es ideal para la aplicación que se desarrolla.

4.4.2. Compilador de Lenguaje C

Para la configuración del procesador en Lenguaje C, se utiliza la herramienta *Software Build Tools for Eclipse.* Esta herramienta es un entorno de desarrollo integrado (en inglés Integrated Development Environment, IDE) basado en Eclipse que se utiliza en el desarrollo de software para sistemas embebidos. Esta herramienta proporciona una interfaz gráfica y funcionalidades avanzadas para la programación, depuración y compilación de código C/C++ para sistemas embebidos; con ella es posible crear, modificar y depurar proyectos de software, así como gestionar las bibliotecas y dependencias necesarias para el correcto funcionamiento.

5. ANÁLISIS FRECUENCIAL DE SEÑAL DETERMINÍSTICA

Al momento de implementar una etapa de filtrado en un sistema, es importante realizar un buen análisis frecuencial; esto permite identificar las frecuencias presentes en la señal y determinar los componentes de interés que se desean preservar y/o eliminar mediante el filtrado.

Además, el análisis frecuencial ayuda a determinar qué tipo de filtro es el más apropiado para el procesamiento de la señal. Dependiendo de las características del espectro, se pueden elegir filtros pasa-bajas, pasa-banda, rechazo-banda, pasa-

altas o de fase lineal, con el objetivo de lograr la atenuación o fase requeridas para la aplicación.

Al analizar la señal en el dominio de la frecuencia, se pueden establecer los requisitos de atenuación, banda(s) de transición y respuesta en frecuencia necesarios para la aplicación. Esto permite realizar un diseño óptimo del filtro, seleccionando los parámetros adecuados, como la frecuencia de corte, la pendiente de atenuación y orden del filtro.

Para realizar el análisis frecuencia de la señal determinística, se emplea la función *fft* de Matlab que aplica Transformada Rápido de Fourier. Primeramente, se analiza la señal sin ruido para observar sus mayores componentes frecuenciales.

Fuente: Autor

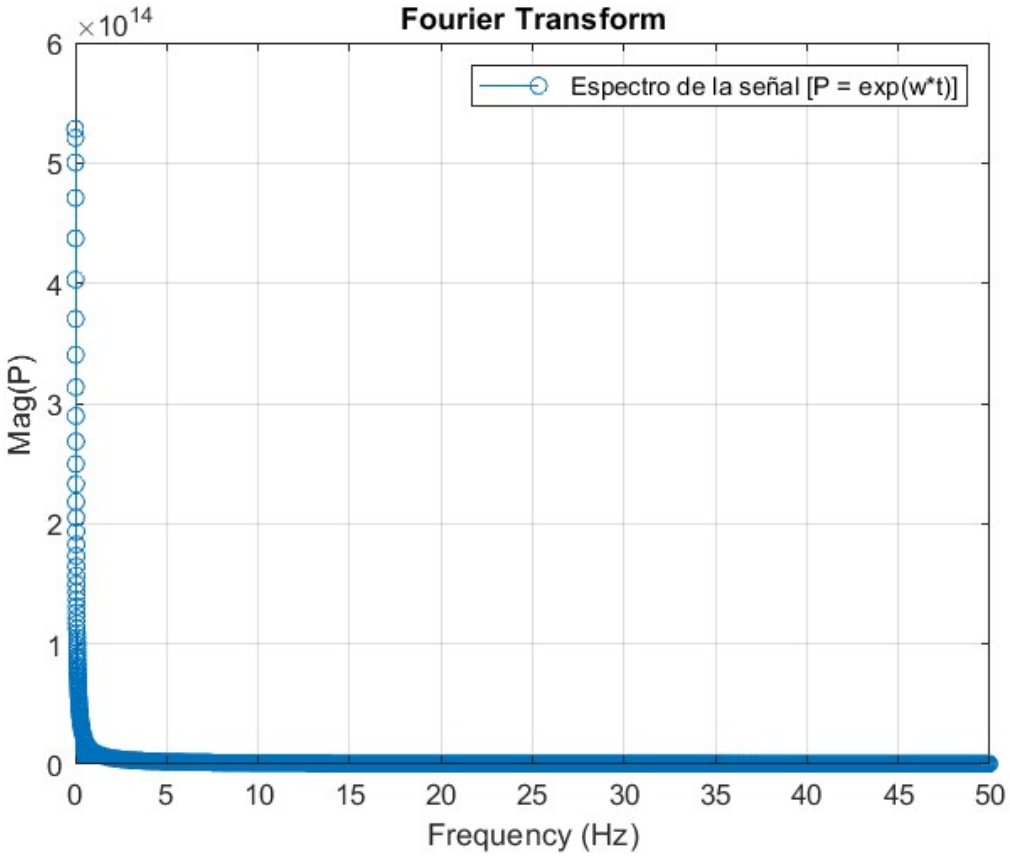

Figura 6. Transformada Rápida de Fourier de la señal sin ruido

Se observa que la señal contiene la mayor parte de su información en los componentes de baja frecuencia. Sin embargo, para localizar los espectros de distorsión, se añade un ruido gaussiano con una desviación estándar de 0.5 y se realiza su análisis frecuencial.

Fuente: Autor

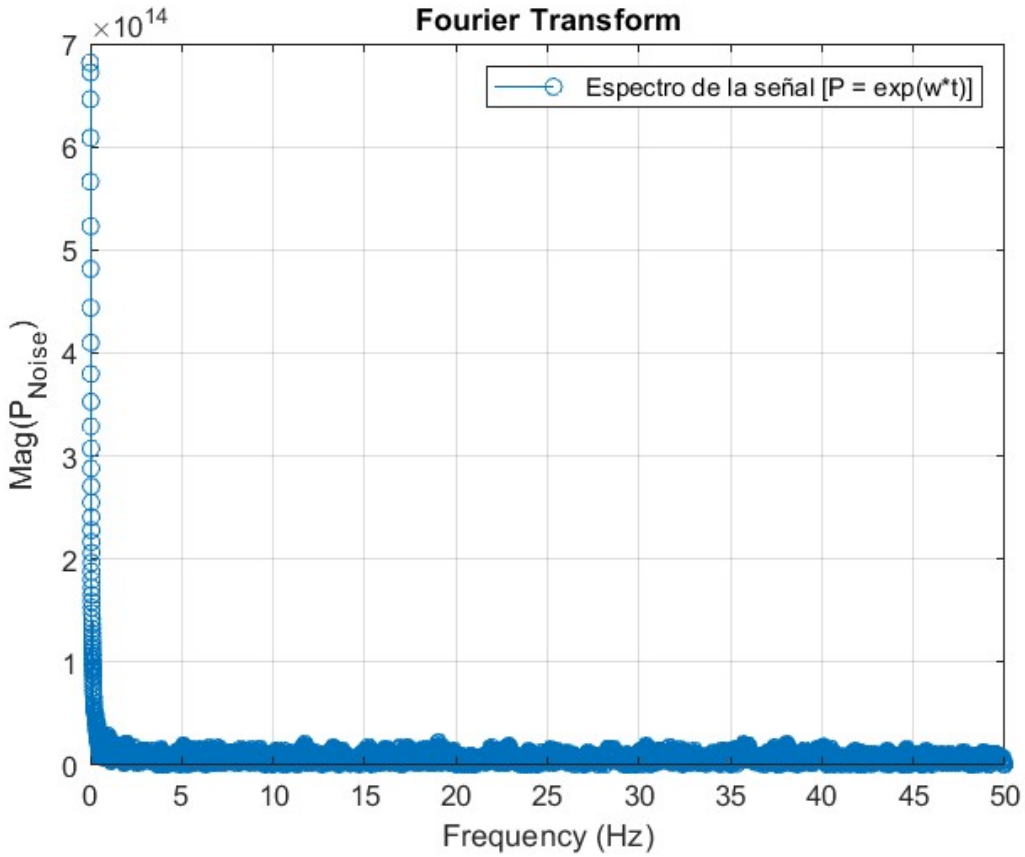

Figura 7. Transformada Rápida de Fourier de la señal con ruido.

Con este análisis, se concluye que lo más conveniente para eliminar el ruido, es implementar un filtro pasa-bajas con frecuencia de corte muy baja; pues se determina que la mayor parte del ruido está presente en los espectros de alta frecuencia.

6. DISEÑO

6.1. Diseño del Filtro Digital IIR

Para diseñar el filtro digital IIR Chebyshev con rizado en la banda pasante, primeramente, se deben especificar cuatro parámetros de diseño, los cuales se escogen a partir del análisis frecuencial anterior. Estos parámetros son:

$$A_p \rightarrow Atenuación\ en\ decibeles\ en\ la\ banda\ de\ paso$$
$$A_s \rightarrow Atenuación\ en\ decibeles\ en\ la\ banda\ de\ rechazo$$
$$f_p \rightarrow Frecuencia\ a\ la\ cual\ se\ presenta\ A_p$$
$$f_s \rightarrow Frecuencia\ a\ la\ cual\ se\ presenta\ A_s$$

Fuente: Diseño Electrónico. (Savant JR., et al.)

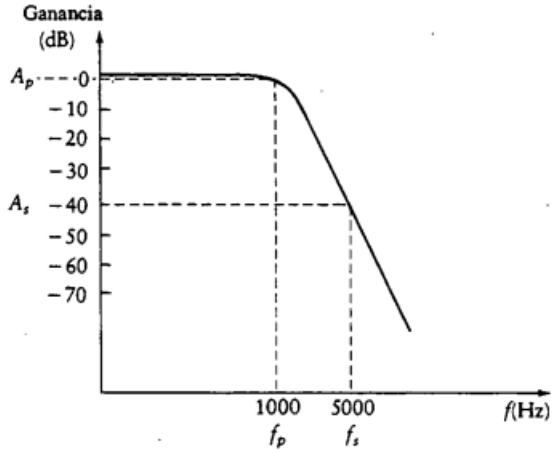

Figura 8. Ejemplo de parámetros de diseño de filtro Chebyshev Analógico

Según el análisis frecuencial previo, se escogieron los siguientes parámetros de diseño para la implementación del filtro:

$$A_p = 1\ dB \tag{6.1.1}$$
$$A_s = 20\ dB \tag{6.1.2}$$
$$f_p = 0.2\ Hz\ \rightarrow w_p \approx 1.256\ rad/s \tag{6.1.3}$$
$$f_s = 0.5\ Hz\ \rightarrow w_s \approx 3.1416\ rad/s \tag{6.1.4}$$

Para realizar el diseño de forma analógica se deben seguir los siguientes pasos:

1. Calcular el Factor de rizado.
2. Determinar el orden del filtro requerido.
3. Hallar los polos de la función de transferencia.
4. Determinar la Función de Transferencia.

6.1.1. Factor de Rizado

El parámetro ϵ determinar el rizado en la banda de paso del filtro analógico, y se calcula con la siguiente ecuación:

$$\epsilon = \sqrt{10^{0.1Ap} - 1} \tag{6.1.1.1}$$

Reemplazando el parámetro A_p se obtiene:

$$\epsilon = \sqrt{10^{0.1(1\ dB)} - 1} \tag{6.1.1.2}$$
$$\epsilon = 0.508847 \tag{6.1.1.3}$$

6.1.2. Orden del Filtro

El orden del filtro es un número entero positivo que indica la cantidad de etapas necesarias para implementar el filtro; a medida que este parámetro aumenta, se puede lograr una mayor selectividad en la eliminación (o atenuación) de las frecuencias no deseadas, pero a su vez, también aumenta la complejidad de la implementación y del coste computacional. Es muy importante seleccionar este parámetro de manera adecuada con el fin de satisfacer los requisitos de la aplicación en términos de precisión, coste computacional y atenuación. Para calcularlo, se emplea la siguiente ecuación:

$$n \geq \frac{cosh^{-1}\left(\sqrt{\frac{10^{0.1As}-1}{10^{0.1Ap}-1}}\right)}{cosh^{-1}\left(\frac{\omega_s}{\omega_p}\right)} \tag{6.1.2.1}$$

Reemplazando todos los parámetros se obtiene el siguiente resultado:

$$n \geq \frac{cosh^{-1}\left(\sqrt{\frac{10^{0.1(20\,dB)}-1}{10^{0.1(1\,dB)}-1}}\right)}{cosh^{-1}\left(\frac{3.1416\,rad/s}{1.256\,rad/s}\right)} \tag{6.1.2.2}$$

$$n \geq 2.33961 \tag{6.1.2.3}$$

$$n = 3 \tag{6.1.2.4}$$

El filtro debe ser mínimo de tercer orden teniendo en cuenta las magnitudes de atenuación y frecuencias de corte que se eligieron para esta aplicación.

6.1.3. Polos de la Función de Transferencia

Los polos de una función de transferencia son los valores en el plano complejo que hacen que una función de transferencia tienda a infinito y son una parte esencial del análisis y diseño de sistemas de control; además, los polos están estrechamente relacionados con la resonancia y estabilidad de los filtros digitales y analógicos. Los polos del filtro Chebyshev están dados por la siguiente ecuación:

$$\begin{aligned} P_k = &-\omega_p \sin\left(\frac{2k-1}{n}\frac{\pi}{2}\right) sinh\left(\frac{1}{n} sinh^{-1}\left(\frac{1}{\epsilon}\right)\right) \\ &+ j\,\omega_p \cos\left(\frac{2k-1}{n}\frac{\pi}{2}\right) cosh\left(\frac{1}{n} sinh^{-1}\left(\frac{1}{\epsilon}\right)\right) \quad k = 1, 2, \dots, n \end{aligned} \tag{6.1.3.1}$$

Se aplica la fórmula para cada uno de los valores de *k*, para hallar la posición en el plano complejo de los tres polos:

$$P_1 = -(1.256\ rad/s)\sin\left(\frac{2(1)-1}{3}\frac{\pi}{2}\right) sinh\left(\frac{1}{3}sinh^{-1}\left(\frac{1}{0.508847}\right)\right) + j\ (1.256\ rad/s)\cos\left(\frac{2(1)-1}{3}\frac{\pi}{2}\right) cosh\left(\frac{1}{3}sinh^{-1}\left(\frac{1}{0.508847}\right)\right) \quad (6.1.3.2)$$

$$P_1 = -0.3105 + j1.2139 \quad (6.1.3.3)$$

$$P_2 = -(1.256\ rad/s)\sin\left(\frac{2(2)-1}{3}\frac{\pi}{2}\right) sinh\left(\frac{1}{3}sinh^{-1}\left(\frac{1}{0.508847}\right)\right) + j\ (1.256\ rad/s)\cos\left(\frac{2(2)-1}{3}\frac{\pi}{2}\right) cosh\left(\frac{1}{3}sinh^{-1}\left(\frac{1}{0.508847}\right)\right) \quad (6.1.3.4)$$

$$P_2 = -0.6210 \quad (6.1.3.5)$$

$$P_3 = -(1.256\ rad/s)\sin\left(\frac{2(3)-1}{3}\frac{\pi}{2}\right) sinh\left(\frac{1}{3}sinh^{-1}\left(\frac{1}{0.508847}\right)\right) + j\ (1.256\ rad/s)\cos\left(\frac{2(3)-1}{3}\frac{\pi}{2}\right) cosh\left(\frac{1}{3}sinh^{-1}\left(\frac{1}{0.508847}\right)\right) \quad (6.1.3.6)$$

$$P_3 = -0.3105 - j1.2139 \quad (6.1.3.7)$$

6.1.4. Función de Transferencia

Una vez se obtiene la posición de todos los polos, la función de transferencia del filtro Chebyshev está descrita por la siguiente ecuación:

$$T(s) = \frac{K\omega_p^n}{\epsilon 2^{n-1}(s-P_1)(s-P_2)\dots(s-P_n)} \quad (6.1.4.1)$$

Donde K es la ganancia que se requiere que tenga el filtro en corriente directa; para esta aplicación, se hace igual a uno, por lo que se tiene:

$$T(s) = \frac{\omega_p^n}{\epsilon 2^{n-1}(s-P_1)(s-P_2)\dots(s-P_n)} \quad (6.1.4.2)$$

Reemplazando todas las variables, la función de transferencia del filtro Chebyshev analógico es:

$$T(s) = \frac{0.9750}{(s^2+0.6210s+1.5700)(s+0.6210)} \quad (6.1.4.3)$$

Se simula esta función de transferencia en Matlab para verificar si el filtro analógico cumple con los parámetros propuestos.

Con esta respuesta en frecuencia de la magnitud y la fase se comprueba que el filtro analógico cumple con los parámetros establecidos.

6.1.5. Transformación Bilineal

La transformación bilineal es un método utilizado para convertir una función de transferencia analógica en una función de transferencia digital. Esta transformación mapea el eje imaginario $j\Omega$ del plano *s* de la frecuencia analógica al círculo unitario en el plano *z* de la frecuencia digital. Este mapeo evita el efecto *aliasing* de frecuencia al limitar la frecuencia más alta a la mitad de la frecuencia de muestreo. Es importante tener en cuenta que la transformación bilineal es una transformación no lineal, por lo que no preserva exactamente la forma de la respuesta en frecuencia del filtro analógico original.

Esta transformación es ampliamente utilizada en el diseño de filtros digitales IIR a partir de prototipos de filtros analógicos, ya que permite obtener una respuesta en frecuencia similar a la del filtro original y garantiza la estabilidad del filtro digital resultante.

$$s = \frac{2}{T}\frac{z-1}{z+1} \tag{6.1.5.1}$$

Aplicando la transformación bilineal a la función de transferencia del filtro analógico se obtiene:

$$T(z) = \frac{0.9750}{\left(\frac{2}{T}\frac{z-1}{z+1}\right)^3 + 1.2420\left(\frac{2}{T}\frac{z-1}{z+1}\right)^2 + 1.9556\left(\frac{2}{T}\frac{z-1}{z+1}\right) + 0.9750} \tag{6.1.5.2}$$

Para esta aplicación, la frecuencia de muestreo es de 100 Hz, por lo que el periodo de muestreo es $T = 0.01s$. Reemplazando este valor en la ecuación (6.1.5.2):

$$T(z) = \frac{0.9750}{\left(\frac{2}{0.01}\frac{z-1}{z+1}\right)^3 + 1.2420\left(\frac{2}{0.01}\frac{z-1}{z+1}\right)^2 + 1.9556\left(\frac{2}{0.01}\frac{z-1}{z+1}\right) + 0.9750} \tag{6.1.5.3}$$

Función de transferencia digital:

$$T(z) = \frac{(1.2111e^{-7})z^3 + (3.633e^{-7})z^2 + (3.633e^{-7})z + 1.2111e^{-7}}{z^3 - 2.9875z^2 + 2.9751z - 0.9877} \tag{6.1.5.4}$$

Una vez obtenido todos los coeficientes del filtro digital, se simula esta función de transferencia en Matlab para verificar si el filtro digital obtenido cumple con los parámetros.

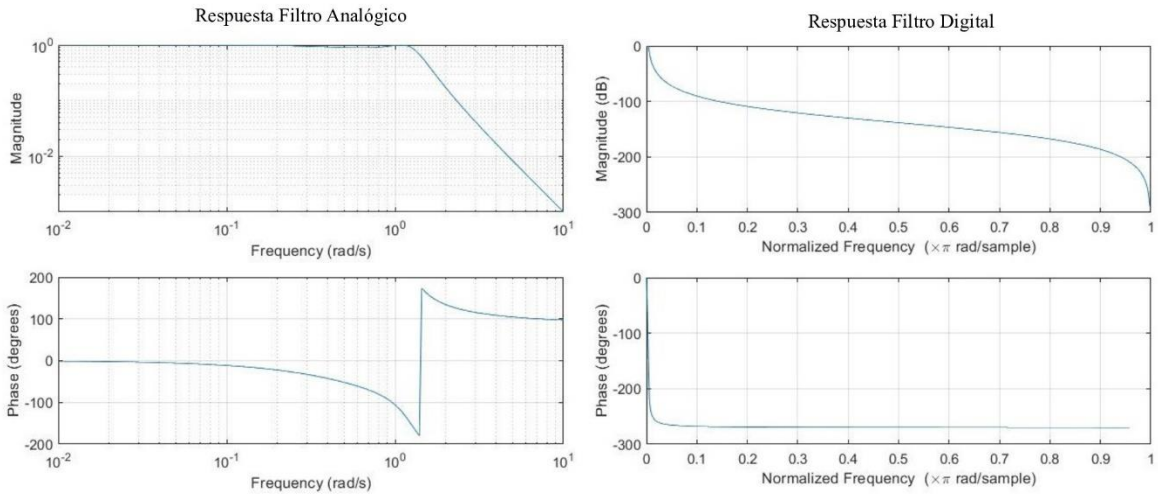

Figura 9. Respuesta Analógica y Digital del Filtro Chebyshev diseñado.

Se observa que el filtro digital cumple con los parámetros establecidos, sin embargo, éste presenta un desfase de aproximadamente -270°, por lo que en la señal de salida del filtro se espera un retraso respecto a la señal de entrada.

6.2. Diseño del Sistema

El sistema embebido requiere principalmente, una Unidad Central de Procesamiento que será la encargada de todo el cálculo y manejo de datos; también se necesita una Unidad de Memoria de Acceso Aleatorio (del inglés Random Access Memory, RAM) que se encargue del almacenamiento a corto plazo de toda la información. Además de eso, requiere un núcleo que establezca la comunicación serial entre la FPGA y el computador; para eso, se añade el núcleo de protocolo UART con el estándar RS-232, para la transmisión y recepción de las señales. Sin embargo, el computador solo cuenta con entrada USB, por lo que es necesario conectar un periférico externo que se encargue de la conversión de datos de UART a USB. Asimismo, se añade un núcleo que controlo las entradas y salidas de propósito general; si bien, esta unidad no es estrictamente necesaria para la aplicación, se añade con el fin de corroborar el funcionamiento adecuado del sistema. Finalmente, se añade la fuente de reloj para que todos los núcleos funcionen armónicamente.

Fuente: Autor

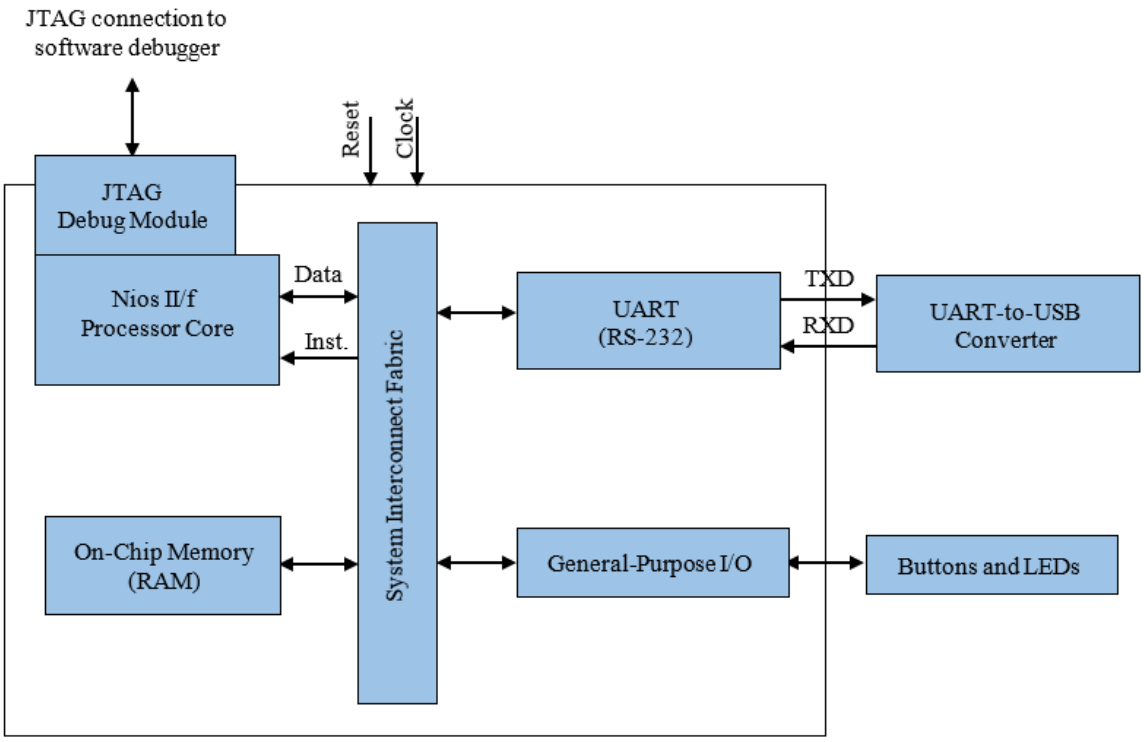

Figura 11. Sistema Diseñado

6.2.1. Fuente de Reloj

La fuente de reloj es un circuito que genera una señal periódica que se utiliza para sincronizar la operación de otros circuitos en un sistema digital. La señal que genera se utiliza para establecer la tasa de procesamiento en el sistema y para controlar el momento en el que se producen los intercambios de datos. En los sistemas digitales complejos, la fuente de reloj es una parte muy importante del diseño del sistema, ya que puede afectar el rendimiento y la precisión de éste.

Para esta aplicación, se usa el reloj que trae la tarjeta FPGA DE0-Nano el cual funciona a una frecuencia de 50 MHz.

6.2.2. Procesador Nios II/f

El núcleo Nios II/f está diseñado para un alto rendimiento de ejecución, y fue diseñado con el objetivo de maximizar la eficiencia de ejecución de instrucciones por ciclo y de optimizar la latencia de las interrupciones; es por esta razón que este núcleo es óptimo para aplicaciones de rendimiento crítico, así como para aplicaciones que requieran grandes cantidades de código y datos.

Por otra parte, este núcleo proporciona diversas opciones de Unidad Aritmética Lógica para mejorar el rendimiento de las multiplicaciones, divisiones y desplazamiento de bits.

En cuanto a su rendimiento de procesamiento de datos, el Nios II/f brinda la opción de un bloque DSP que permite usar los multiplicadores y sumadores embebidos en el dispositivo. También permite crear multiplicadores a partir de los elementos lógicos disponibles en la FPGA; esto supone una gran ventaja para esta aplicación, puesto que el filtrado de una señal es un conjunto de multiplicaciones y sumas de datos de entrada y salida. Al tener un bloque exclusivamente para resolver estas operaciones brinda mayor rendimiento y optimización de recursos en el cálculo.

6.2.3. Memoria RAM interna

La memoria RAM (Random Access Memory) es un tipo de memoria volátil utilizada en sistemas computacionales para almacenar y acceder ´rápidamente a datos de forma temporal. Es un componente esencial, ya que proporciona un espacio de almacenamiento rápido y de acceso aleatorio para el procesador y otros dispositivos. Esta memoria se compone de celdas de almacenamiento que pueden ser leídas y escritas (en inglés readable and writable) de manera directa, lo que le brinda velocidad en el acceso a los datos. La información almacenada en la memoria RAM se pierden cuando el sistema queda sin energía, por lo que es utilizada principalmente para almacenar programas de ejecución, datos temporales y datos que necesitan acceso rápido. La capacidad y velocidad de la memoria RAM tienen una importancia significativa en el rendimiento general del sistema, y es una parte fundamental en la arquitectura de cualquier dispositivo computacional.

Para la implementación de este sistema, se usa el bloque de memoria *On-Chip* configurada como memoria RAM. Las memorias en chip tienen un tiempo de acceso rápido en comparación con las memorias fuera del chip, además de esto, Qsys instala automáticamente la memoria dentro del sistema, por lo que no es necesario realizar ninguna conexión externa manual. Por otra parte, algunos bloques de memoria pueden tener contenidos inicializados cuando se enciende la FPGA. Esta característica es útil, por ejemplo, para almacenar constantes de datos o códigos de arranque del procesador. Otra ventaja relevante, es que las memorias internas soportan accesos a doble puerto, permitiendo a dos hosts acceder a la misma memoria simultáneamente.

La memoria RAM utilizada en este proyecto tiene un tamaño total de 32768 bytes y un ancho de datos de 32 bits.

Fuente: Autor

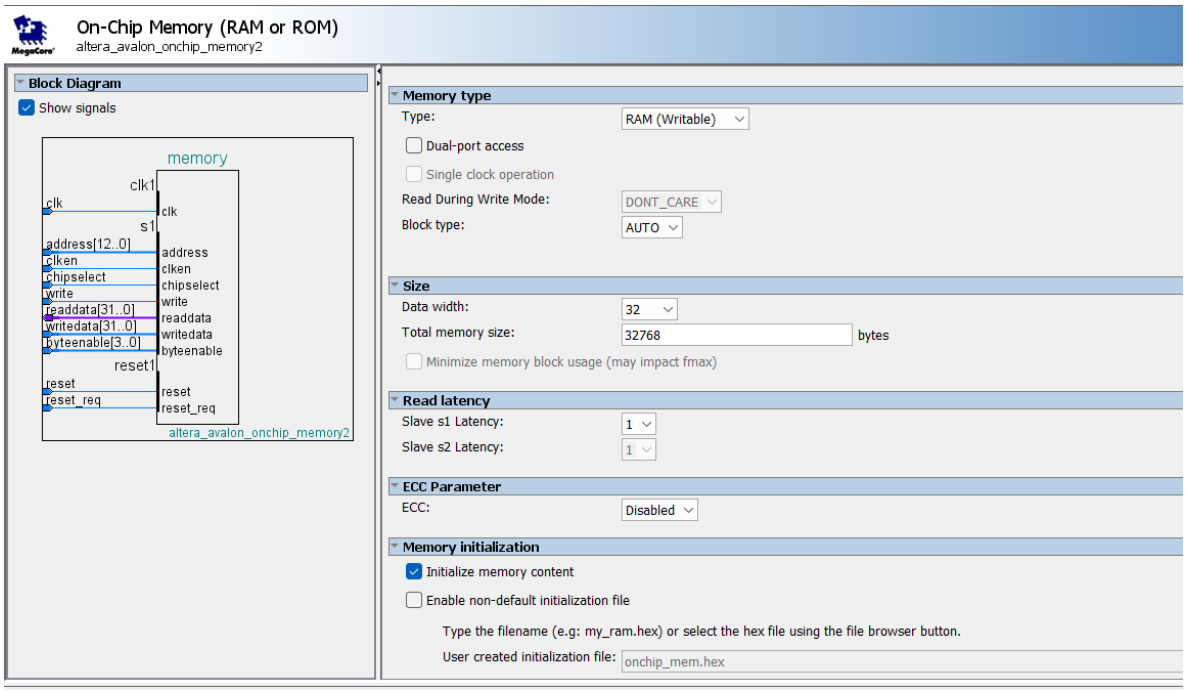

Figura 12. Configuración de Memoria Interna

6.2.4. Protocolo UART

El núcleo UART implementa la lógica de transmisión y recepción asíncrona RS-232. Este núcleo envía y recibe datos serie a través de los puertos TXD y RXD. Los búferes de entrada y salida de la mayoría de las familias de FPGAs Intel no cumplen con los niveles de voltaje RS-232, y pueden dañarse si son accionados directamente por señales de un conector de este tipo. Para cumplir con las especificaciones de señalización de voltaje RS-232, se requiere un búfer externo de cambio de nivel entre los pines E/S de la FPGA y el convertidor externo. En este caso, se selecciona el convertidor UART a USB externo PL2303 de Prolific.

El módulo UART se configura para que no tenga bit de paridad, use 8 bits para la codificación del dato y use 1 bit de parada. Además de esto, se configura para que el Baud Rate sea de 115200 bps (Bits Per Second).

Fuente: Autor

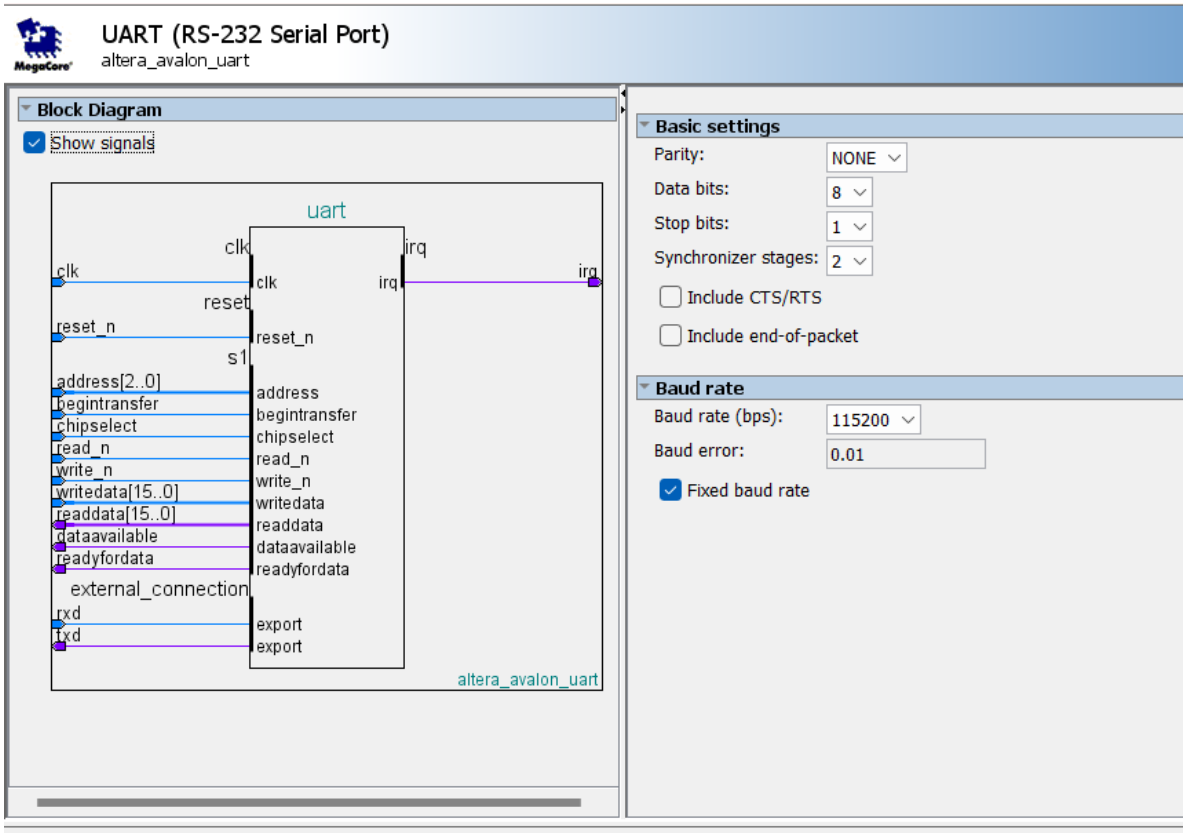

Figura 13. Configuración del Módulo UART

6.2.4.1. Convertidor USB a RS-232

Para permitir el intercambio de datos entre la FPGA y el computador, es necesario establecer un protocolo de comunicación serial entre ambos. Como se mencionó anteriormente, en la FPGA se implementa la lógica de transmisión y recepción asíncrona RS-232, por lo que se necesita un controlador que convierta la información de esta lógica a USB, para que el computador pueda reconocerla.

Se escoge el controlador desarrollado por Prolific PL2303, el cual funciona como puente entre un puerto USB y un puerto serie RS-232 estándar, lo que posibilita la comunicación entre los dos dispositivos con diferente protocolo de comunicación serial.

Fuente: PL-2303 USB to RS-232 Bridge Controller Product Datasheet. Prolific.

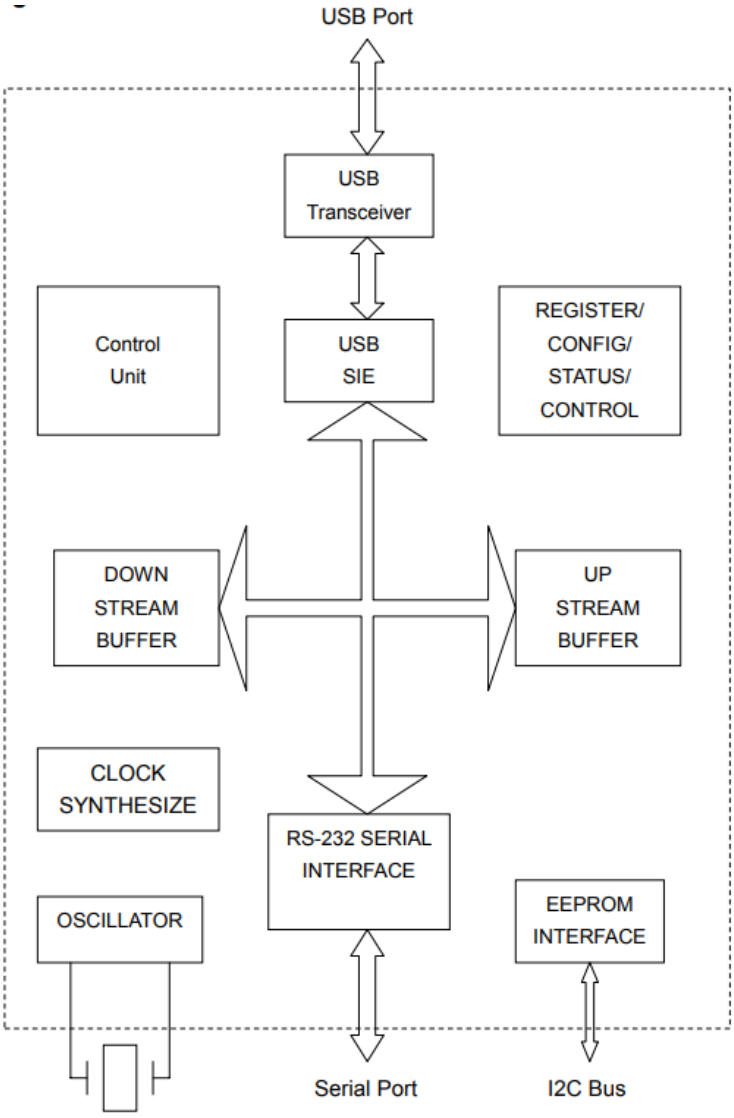

Figura 14. Diagrama de Bloques del Convertidor PL2303

6.2.5. Entrada/Salida de Propósito General

Para finalizar el diseño del sistema, se agrega un núcleo PIO (Parallel Input/Output), el cual permite que los puertos de entrada y salida del núcleo puedan conectarse a la lógica dentro o fuera del chip. Éste permite configurarse con solo entradas, solo salidas, o con entradas y salidas si así se desea. También se puede utilizar para controlar pines de E/S bidireccionales; en ese caso, en núcleo proporciona un modo bidireccional con control de triple estado.

Para este proyecto, se configura el núcleo para solo salida, y se conectan a los ocho LEDs que tiene la tarjeta. Cuando la FPGA esté procesando los datos, los LEDs estarán apagados; y cuando el sistema haya terminado de filtrar la señal, se encenderán todos los LEDs. De esta manera, se corrobora visualmente el correcto funcionamiento del sistema.

7. IMPLEMENTACIÓN

Para realizar la implementación del sistema diseñado, se usa la herramienta Qsys contenida en el Software Quartus II desarrollado por Intel. Qsys es una herramienta de diseño de sistemas en chip (del inglés System on Chip, SoC) que proporciona una interfaz gráfica de usuario que permite diseñar, configurar y conectar componentes de hardware en un sistema integrado. Con Qsys se puede construir sistemas complejos en un entorno de desarrollo visual, utilizando componentes predefinidos o personalizados. Una vez construido el sistema, esta herramienta permite generar un archivo de descripción de hardware en lenguaje VHDL o Verilog, el cual representa la interconexión y configuración de los componentes del sistema. El archivo resultante se utiliza luego en la interfaz de diseño de Quartus II para su síntesis, verificación, implementación y generación de la configuración para la FPGA o cualquier otro dispositivo programable.

7.1. Implementación del Sistema

Ya diseñado el sistema, se procede a realizar sus configuraciones y conexiones en la herramienta Qsys.

Fuente: Autor

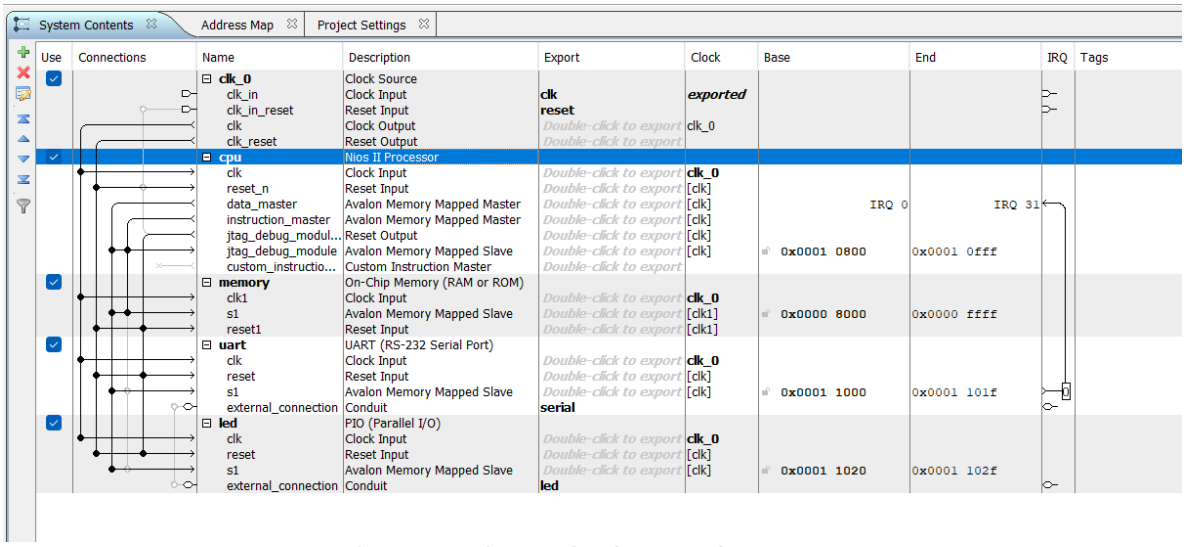

Figura 15. Sistema implementado en Qsys

Ya con el sistema diseñado y configurado, se procede a generar el código en lenguaje de descripción de hardware. Para esta aplicación, se genera lenguaje Verilog.

Fuente: Autor

```
// uartConv.v

// Generated using ACDS version 13.1 162 at 2023.05.27.00:12:54

`timescale 1 ps / 1 ps
module uartConv (
		input  wire       clk_clk,       //    clk.clk
		input  wire       reset_reset_n, //  reset.reset_n
		input  wire       serial_rxd,    // serial.rxd
		output wire       serial_txd,    //       .txd
		output wire [7:0] led_export     //    led.export
	);

	wire  [31:0] mm_interconnect_0_led_s1_writedata;                // mm_interconnect_0:led_s1_writedata -> led:writedata
	wire   [1:0] mm_interconnect_0_led_s1_address;                  // mm_interconnect_0:led_s1_address -> led:address
	wire         mm_interconnect_0_led_s1_chipselect;               // mm_interconnect_0:led_s1_chipselect -> led:chipselect
	wire         mm_interconnect_0_led_s1_write;                    // mm_interconnect_0:led_s1_write -> led:write_n
	wire  [31:0] mm_interconnect_0_led_s1_readdata;                 // led:readdata -> mm_interconnect_0:led_s1_readdata
	wire         cpu_instruction_master_waitrequest;                // mm_interconnect_0:cpu_instruction_master_waitrequest -> cpu:i_waitrequest
	wire  [16:0] cpu_instruction_master_address;                    // cpu:i_address -> mm_interconnect_0:cpu_instruction_master_address
	wire         cpu_instruction_master_read;                       // cpu:i_read -> mm_interconnect_0:cpu_instruction_master_read
	wire  [31:0] cpu_instruction_master_readdata;                   // mm_interconnect_0:cpu_instruction_master_readdata -> cpu:i_readdata
	wire  [15:0] mm_interconnect_0_uart_s1_writedata;               // mm_interconnect_0:uart_s1_writedata -> uart:writedata
	wire   [2:0] mm_interconnect_0_uart_s1_address;                 // mm_interconnect_0:uart_s1_address -> uart:address
	wire         mm_interconnect_0_uart_s1_chipselect;              // mm_interconnect_0:uart_s1_chipselect -> uart:chipselect
	wire         mm_interconnect_0_uart_s1_write;                   // mm_interconnect_0:uart_s1_write -> uart:write_n
	wire         mm_interconnect_0_uart_s1_read;                    // mm_interconnect_0:uart_s1_read -> uart:read_n
	wire  [15:0] mm_interconnect_0_uart_s1_readdata;                // uart:readdata -> mm_interconnect_0:uart_s1_readdata
	wire         mm_interconnect_0_uart_s1_begintransfer;           // mm_interconnect_0:uart_s1_begintransfer -> uart:begintransfer
	wire         mm_interconnect_0_cpu_jtag_debug_module_waitrequest; // cpu:jtag_debug_module_waitrequest -> mm_interconnect_0:cpu_jtag_debug_module_waitrequ
	wire  [31:0] mm_interconnect_0_cpu_jtag_debug_module_writedata;  // mm_interconnect_0:cpu_jtag_debug_module_writedata -> cpu:jtag_debug_module_writedata
	wire   [8:0] mm_interconnect_0_cpu_jtag_debug_module_address;    // mm_interconnect_0:cpu_jtag_debug_module_address -> cpu:jtag_debug_module_address
	wire         mm_interconnect_0_cpu_jtag_debug_module_write;      // mm_interconnect_0:cpu_jtag_debug_module_write -> cpu:jtag_debug_module_write
	wire         mm_interconnect_0_cpu_jtag_debug_module_read;       // mm_interconnect_0:cpu_jtag_debug_module_read -> cpu:jtag_debug_module_read
```

Figura 16. Parte del código generado por Qsys.

Ya generado el código en Verilog, es necesario realizar la configuración de pines en la tarjeta. Para ello, es necesario revisar detalladamente el manual de usuario de la tarjeta de desarrollo.

Fuente: Autor

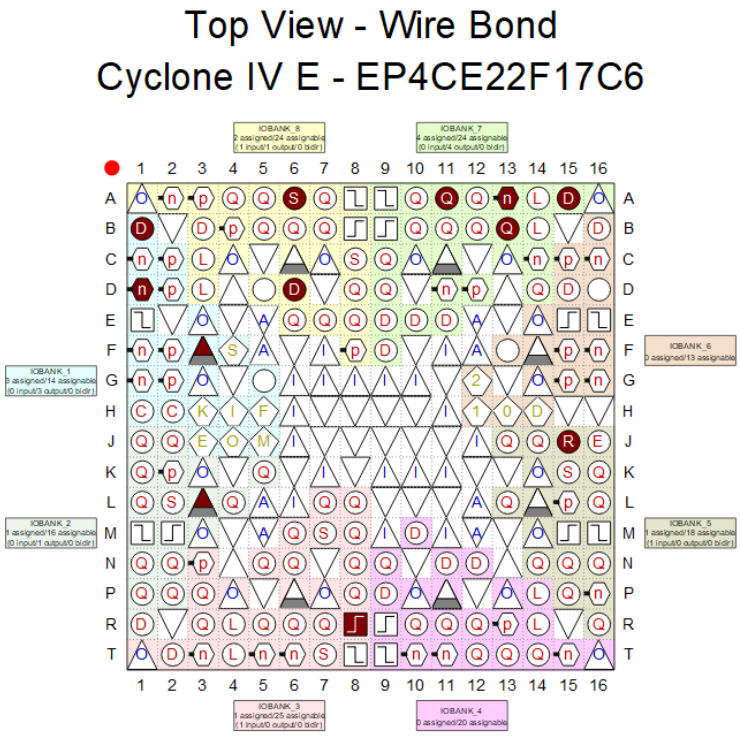

Figura 17. Asignación de Pines de la tarjeta FPGA.

Fuente: Autor.

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate
clk_clk	Input	PIN_R8	3	B3_N0	PIN_R8	2.5 V (default)		8mA (default)	
led_export[7]	Output	PIN_L3	2	B2_N0	PIN_L3	2.5 V (default)		8mA (default)	2 (default)
led_export[6]	Output	PIN_B1	1	B1_N0	PIN_B1	2.5 V (default)		8mA (default)	2 (default)
led_export[5]	Output	PIN_F3	1	B1_N0	PIN_F3	2.5 V (default)		8mA (default)	2 (default)
led_export[4]	Output	PIN_D1	1	B1_N0	PIN_D1	2.5 V (default)		8mA (default)	2 (default)
led_export[3]	Output	PIN_A11	7	B7_N0	PIN_A11	2.5 V (default)		8mA (default)	2 (default)
led_export[2]	Output	PIN_B13	7	B7_N0	PIN_B13	2.5 V (default)		8mA (default)	2 (default)
led_export[1]	Output	PIN_A13	7	B7_N0	PIN_A13	2.5 V (default)		8mA (default)	2 (default)
led_export[0]	Output	PIN_A15	7	B7_N0	PIN_A15	2.5 V (default)		8mA (default)	2 (default)
reset_reset_n	Input	PIN_J15	5	B5_N0	PIN_J15	2.5 V (default)		8mA (default)	
serial_rxd	Input	PIN_D6	8	B8_N0	PIN_D6	2.5 V (default)		8mA (default)	
serial_txd	Output	PIN_A6	8	B8_N0	PIN_A6	2.5 V (default)		8mA (default)	2 (default)

Figura 18. Configuración de Pines en la FPGA.

Ya hecho esto, se ejecuta el archivo para verificar que esté bien diseñado y sin errores.

Fuente: Autor

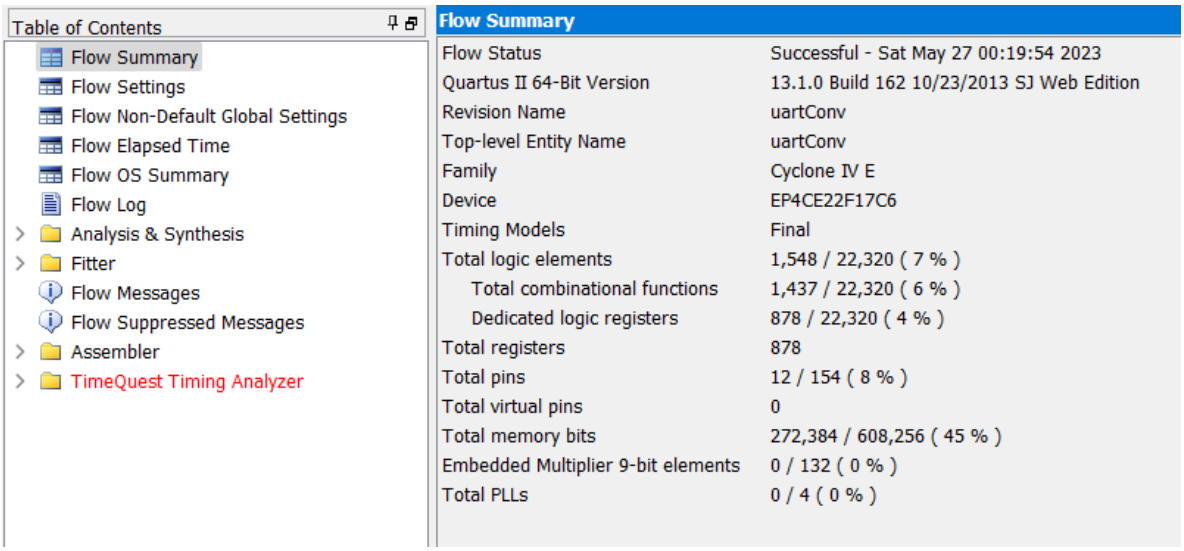

Figura 19. Ejecución del código en Verilog.

La ejecución exitosa quiere decir que el sistema está correctamente diseñado y está listo para ser cargado en la FPGA.

Fuente: Autor

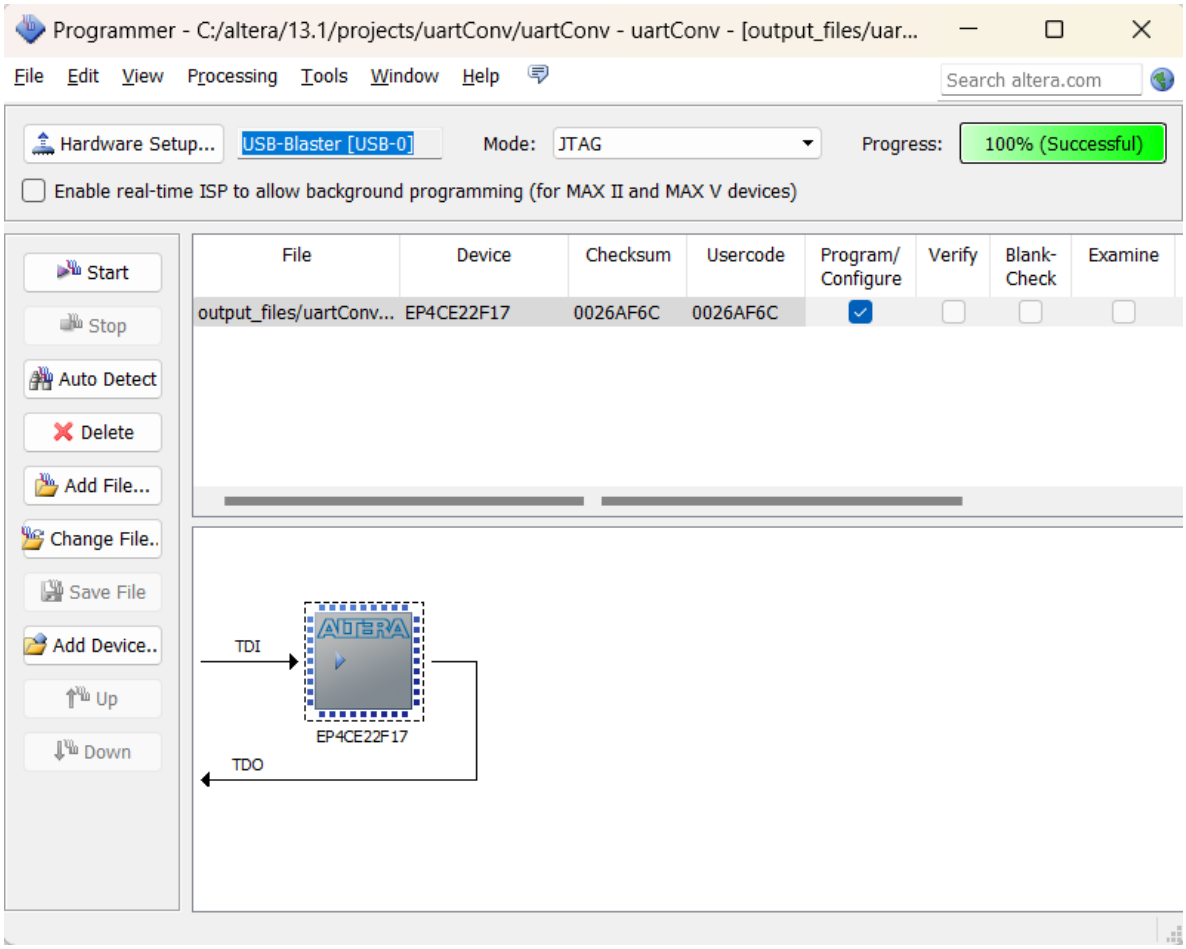

Figura 20. FPGA programada satisfactoriamente.

Una vez programada la FPGA, el sistema diseñado queda listo para ser usado por medio de un compilador de código de lenguaje C. En este proyecto se usó la herramienta incorporada en Quartus II, Software Build Tools for Eclipse.

Fuente: Autor.

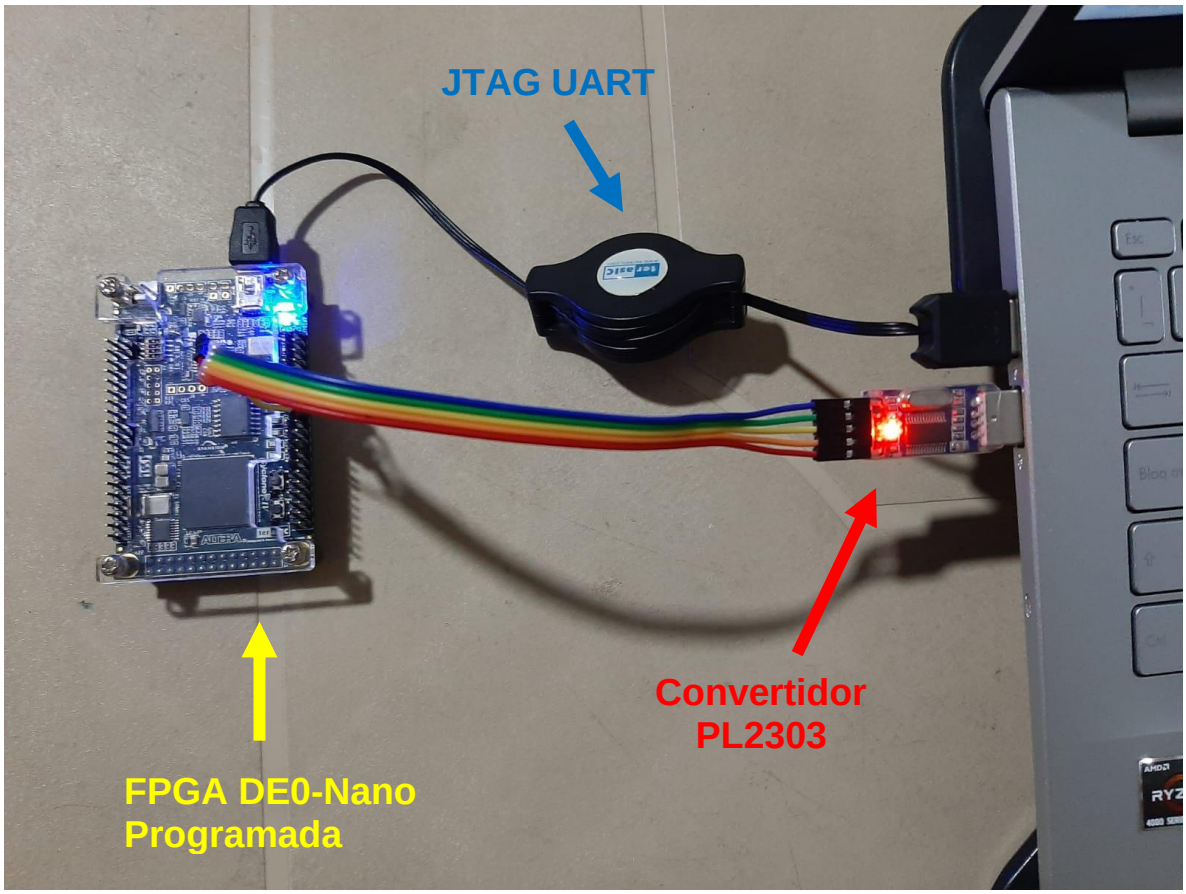

Figura 21. Sistema Físico Implementado

7.2. Ecuación en diferencias del filtro

Para programar el filtro dentro del procesador, es necesario hacerlo mediante su ecuación en diferencias, que está dada por la ecuación (4.3.1). Para hallarla, se realiza el siguiente procedimiento:

$$\frac{Y(z)}{X(z)} = \frac{1.211e^{-7}z^{0} + 3.63e^{-7}z^{-1} + 3.63e^{-7}z^{-2} + 1.211e^{-7}z^{-3}}{1z^{0} - 2.9875z^{-1} + 2.975z^{-2} - 0.987z^{-3}} \qquad (7.2.1)$$

Es importante tener en cuenta que, por limitación de hardware, es necesario realizar la aproximación de los coeficientes del filtro.

$$\begin{aligned} Y(z) = {} & 1.211e^{-7}X(z) + 3.63e^{-7}z^{-1}X(z) + 3.63e^{-7}z^{-2}X(z) \\ & + 1.211e^{-7}z^{-3}X(z) + 2.9875z^{-1}Y(z) - 2.975z^{-2}Y(z) \\ & + 0.987z^{-3}Y(z) \end{aligned} \qquad (7.2.2)$$

Aplicando la Transformada Z inversa se obtiene la salida del filtro en función de las muestras en el tiempo:

$$y(n) = 1.211e^{-7}x(n) + 3.63e^{-7}x(n-1) + 3.63e^{-7}x(n-2) + 1.211e^{-7}x(n-3) + 2.9875y(n-1) - 2.975y(n-2) + 0.987y(n-3) \quad (7.2.3)$$

Finalmente, la ecuación (7.2.3) es la que se implementa en el sistema de la FPGA. Es necesario definir las variables de estado y su correspondiente actualización. Como se puede observar, la señal de salida tiene un retraso de 3 muestras, por lo que es necesario definir condiciones iniciales de filtrado para evitar el sobre impulso que genera el filtro antes de su estabilización. Tanto los datos de entrada, como los datos de salida iniciales, se establecen en 1.

Fuente: Autor

```
// Inicialización de variables de estado
double x1 = 1;
double x2 = 1;
double x3 = 1;
double y1 = 1;
double y2 = 1;
double y3 = 1;
double x0;
int i = 0;
```

Figura 22. Inicialización de las variables de estado.

8. RESULTADOS

La precisión y el rendimiento del sistema se mide mediante la máxima diferencia en *pcm* y el error promedio entre la señal de reactividad calculada a partir de la señal de salida del filtro y la solución matemática exacta de la ecuación de reactividad para la forma de potencia. Con el fin obtener una amplia gama de resultados, se modifica la desviación estándar que corresponde al coeficiente de ruido gaussiano que se añade la señal de potencia. La señal de densidad poblacional de neutrones es directamente proporcional a la señal de potencia eléctrica generada por el reactor nuclear; por esta razón, la reactividad también se puede calcular a partir de este parámetro.

La señal de potencia determinística simulada está dada de la forma:

$$P(t) = e^{0.12353t} \quad (8.1)$$

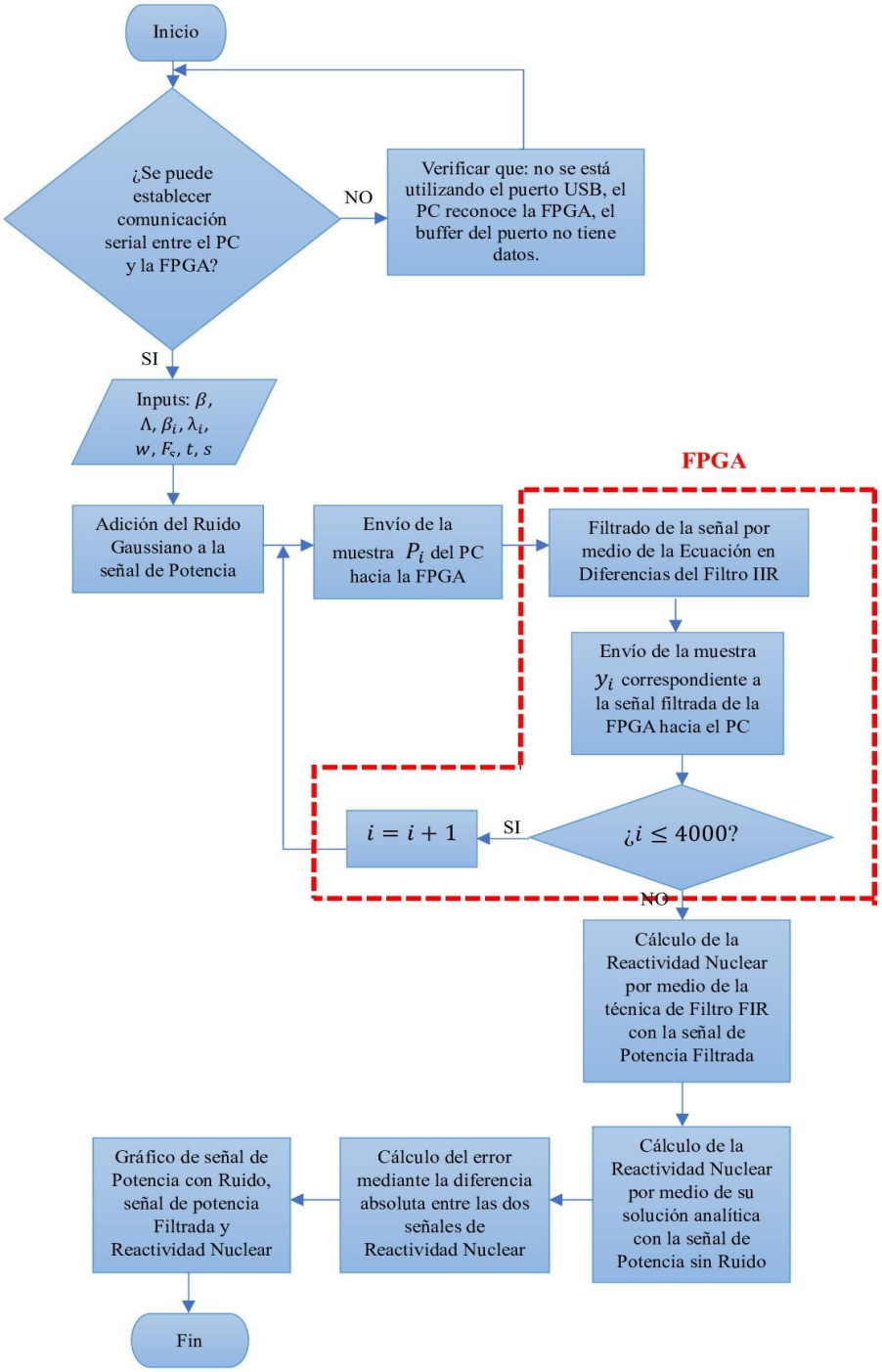

Figura 23. Diagrama de Flujo del Sistema Implementado

8.1. Resultados de Simulación

Para reproducir los resultados simulados con los estados iniciales planteados anteriormente, se evita hacer uso de la función *filter* de Matlab; y en cambio, se crea

una función la cual establece la ecuación en diferencias y añade un retraso de 3.5 segundos.

El tiempo total de simulación fue de 50 segundos, lo que equivale a un total de 5.000 muestras.

Se inicia con una desviación estándar de 0.001; este valor añade muy poco ruido a la señal, por lo que se obtiene que la señal resultante es muy parecida a la señal de reactividad analítica; con una máxima diferencia de 2.8867 pcm presentada a los 3.84 s, y un error medio de 0.4437 pcm.

Luego, se aumenta el valor de desviación estándar a 0.01, lo que correspondería a un nivel de ruido del 1%. Sin embargo, aunque se aumentó 10 veces respecto al ruido anterior, sigue siendo un nivel de ruido bajo. La máxima diferencia con este valor de desviación estándar es de 2.5748 pcm presentada a los 4.02 s, y el error medio es de 0.4597 pcm.

Al incrementar el valor de desviación estándar a 0.1, la señal estaría siendo afectada por un nivel de ruido gaussiano del 10%, lo que ya se puede considerar alto. A pesar de esto, el filtro digital responde bastante bien, y se obtiene una máxima diferencia de 5.4103 pcm a los 6.09 s, con un error medio de 1.7277 pcm

Cuando el valor de desviación estándar aumenta a 0.2, (equivalente al 20% de la señal de potencia), se obtiene una máxima diferencia de 10.3486 pcm en el tiempo de 8.60 s. Este valor de máxima diferencia aumenta casi el doble con respecto al valor de desviación estándar anterior. El error medio se sigue manteniendo bajo, con un resultado de 3.4883 pcm.

Seguidamente, se aumenta el nivel de ruido al 30% de la señal de potencia, lo que significa un valor de desviación estándar de 0.3. Ya este nivel de ruido se considera bastante alto, sin embargo, el filtro digital mantiene una respuesta coherente respecto a la solución analítica. Se obtiene un error medio de 5.3018 pcm con una máxima diferencia de 15.7101 pcm a los 8.59 s.

Finalmente, el valor de desviación estándar es incrementado a 0.4, es decir, el nivel de ruido corresponde a el 40% de la señal de potencia; lo que se traduce, en una distorsión muy grande en la señal de entrada. Sin embargo, a pesar de esto, el filtro logra mantener el resultado en una señal congruente, con una máxima diferencia de 20.9661 pcm en el tiempo de 8.59 s, y un error medio de 7.1164 pcm.

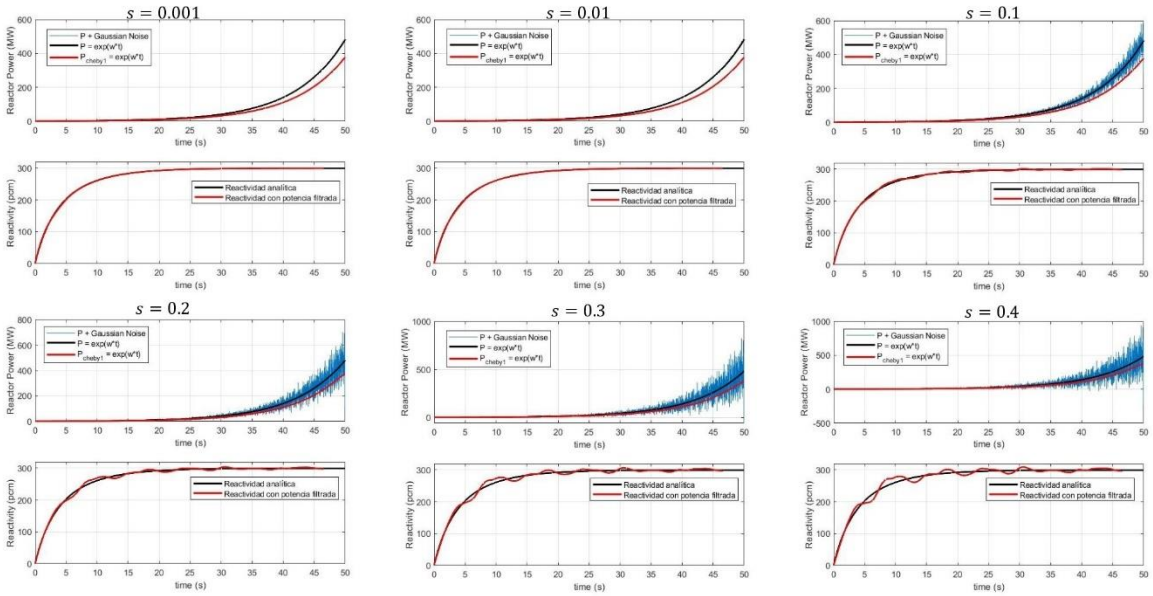

Figura 24. Resultados de la Simulación según su nivel de Desviación

En la tabla a continuación, se resumen los valores de máxima diferencia y error medio para todos los valores de desviación estándar.

Tabla 1. Resultados de la Simulación

Desviación estándar	Máxima Diferencia y tiempo	Error Medio
0.001	$2.8867\ pcm \rightarrow t = 3.84\ s$	0.4437 pcm
0.01	$2.5748\ pcm \rightarrow t = 4.02\ s$	0.4597 pcm
0.1	$5.4103\ pcm \rightarrow t = 6.09\ s$	1.7277 pcm
0.2	$10.3486\ pcm \rightarrow t = 8.60\ s$	3.4883 pcm
0.3	$15.7101\ pcm \rightarrow t = 8.59\ s$	5.3018 pcm
0.4	$20.9661\ pcm \rightarrow t = 8.59\ s$	7.1164 pcm

8.2. Resultados de Implementación

La señal correspondiente a la densidad poblacional de neutrones tendrá un tamaño total de 4000 muestras, es decir, un tiempo de duración de 40 segundos con periodo de muestro de 0.01 segundos. Esta limitación se presenta debido a que la señal de entrada crece de forma exponencial, y la memoria interna del sistema se desborda

para valores de potencia correspondiente a tiempos mayores de 40 segundos aproximadamente.

Al igual que en los resultados simulados, se inicia con un valor de desviación estándar de 0.001. Es un nivel de ruido muy bajo, sin embargo, se obtiene una máxima diferencia de 16.4991 pcm a los 4.33 s, y un error medio de 0.4437 pcm.

Seguidamente, se incrementa el valor de desviación estándar a 0.01, es decir, un nivel de ruido del 1%. Sin embargo, aunque se aumentó este parámetro 10 veces respecto al valor anterior, la máxima diferencia es de 16.0885 pcm en el tiempo de 4.31 s, y un error medio de 0.4597 pcm.

Ahora, cuando se incrementa la desviación estándar 10 veces respeto al valor anterior (es decir, una desviación estándar de 0.1), la máxima diferencia es de 17.3319 pcm en el tiempo de 4.08 s, y un error medio de 1.9854 pcm

Cuando se aumenta el valor de la desviación estándar a 0.2 (equivalente al 20%), se observa una diferencia máxima de 17.4167 pcm en un lapso de 5.89 segundos. Aunque se aumentó el nivel de ruido el doble respecto al valor anterior, el error medio se mantiene bajo, con un valor de 3.4883 pcm.

En el momento que se aumenta la distorsión a un 30% de la señal de potencia, el filtro mantiene su capacidad de producir una respuesta congruente con la esperada; de esta manera, se obtiene como resultado una máxima diferencia de 19.5051 pcm en el tiempo de 5.97 s, y un error medio de 6.8413 pcm.

Por último, se procede aumentar el valor de la desviación estándar a 0.4, lo que implica que el nivel de ruido corresponde al 40% de la señal de entrada. Esta magnitud de distorsión ya provoca fluctuaciones significativas en la potencia. A pesar de esto, el filtro digital logra mantener coherencia en la señal resultante, presentando una máxima diferencia de 22.5839 pcm a los 4.64 s, y un error promedio de 7.5473 pcm.

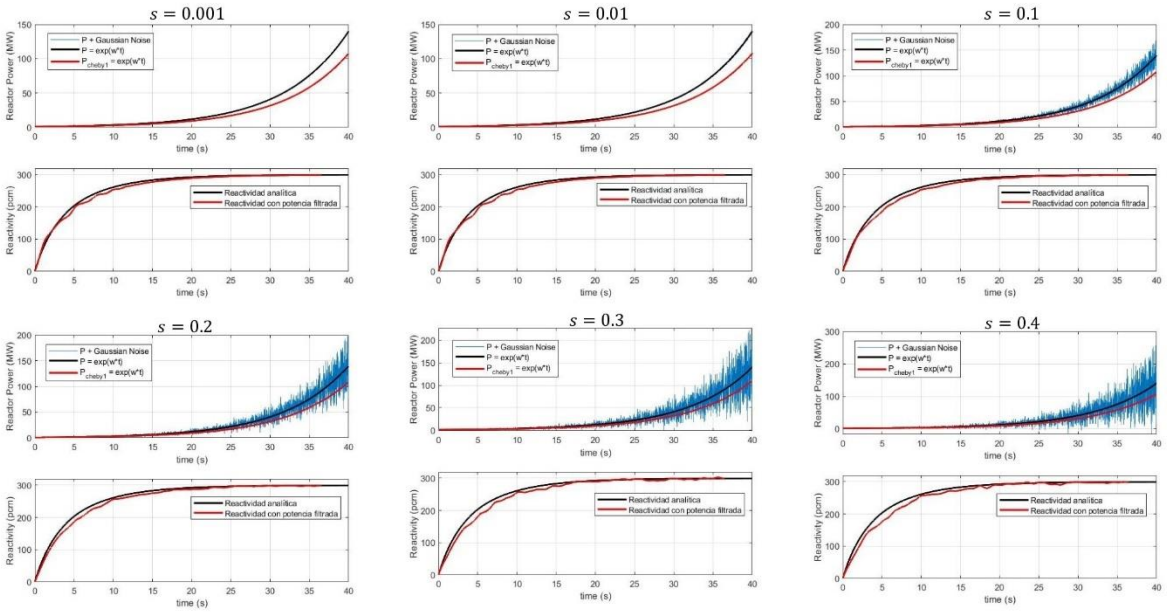

Figura 25. Resultados de la Implementación según su nivel de Desviación

Tabla 2. Resultados de la Implementación

Desviación estándar	Máxima Diferencia y tiempo	Error Medio
0.001	$16.4991\ pcm \rightarrow t = 4.33\ s$	0.4437 pcm
0.01	$16.0885\ pcm \rightarrow t = 4.31\ s$	0.4597 pcm
0.1	$17.3319\ pcm \rightarrow t = 4.08\ s$	1.9854 pcm
0.2	$17.4167\ pcm \rightarrow t = 5.89\ s$	3.4883 pcm
0.3	$19.5051\ pcm \rightarrow t = 5.97\ s$	6.8413 pcm
0.4	$22.5839\ pcm \rightarrow t = 4.64\ s$	7.5473 pcm

8.3. Análisis de Resultados

Los resultados obtenidos en la simulación y en la implementación son muy similares, sin embargo, existen algunas diferencias debido a factores como la capacidad de procesamiento de 64 bits del computador y el sistema embebido de 32 bits en la FPGA.

En comparación, los resultados de la simulación mantienen mejor la relación de desviación estándar y máxima diferencia; en contraste con los de la implementación que la máxima diferencia en niveles bajos de ruido es muy alta, lo que significa que el filtro con la aproximación de los coeficientes no mantiene una buena respuesta a niveles bajos de desviación estándar.

A pesar de que las máximas diferencias en los resultados de la implementación son bastantes altos (en comparación con los obtenidos en la simulación), el error medio

obtenido es aceptablemente bajo, lo que quiere decir que el filtro después de un tiempo logra estabilizarse en el valor de referencia.

Se observa que todas las diferencias máximas se producen en un lapso inferior a los 10 segundos (menos de 1000 muestras), lo cual sugiere que los mayores errores ocurren durante la respuesta transitoria del filtro. Sin embargo, una vez alcanzada la estabilidad, la respuesta obtenida se asemeja de manera significativa al valor de referencia proporcionado por la solución analítica de la señal.

Se percibe consistentemente que el error promedio entre la señal filtrada y la solución analítica (tanto para resultados de simulación, como de implementación) se mantuvo a niveles bajos, independientemente de la desviación estándar del ruido gaussiano aplicado. Esto indica que el filtro diseñado es efectivo para mitigar las fluctuaciones de alta frecuencia, y mantener la fidelidad en la señal original. Estos hallazgos refuerzan la confiabilidad y robustez del filtro digital Chebyshev implementado, demostrando su capacidad para proporcionar resultados precisos y consistentes para diversos niveles de ruido.

9. CONCLUSIONES

La revisión bibliográfica realizada sobre los temas desarrollados proporcionó bases sólidas y comprensión detallada de los fundamentos teóricos abarcados en este proyecto. Fue necesario investigar sobre las características y técnicas de diseño de los filtros digitales; así como también, leer sobre métodos de aproximación de la ecuación inversa de la cinética puntual.

El filtro digital Chebyshev ofrece una ventaja significativa en comparación con el filtro Butterworth al lograr una pendiente más pronunciada en su banda de transición. No obstante, esta mejora viene acompañada de un costo, ya que se produce un rizado en la banda de paso. Por lo tanto, es crucial realizar un estudio exhaustivo de la aplicación y el propósito de la etapa de filtrado, evaluando si se permite una oscilación en la banda pasante. De esta manera, se podrá determinar si los beneficios adicionales del filtro Chebyshev superan las posibles limitaciones introducidas por el rizado en la banda pasante.

El uso de la herramienta Qsys facilitó el diseño y la implementación del sistema, permitiendo integrar, conectar y configurar todos los diferentes módulos utilizados. Además de esto, el procesador Nios II/f ha brindado la flexibilidad y eficiencia en el procesamiento de los datos, lo que ha contribuido a la exitosa elaboración del sistema en la FPGA DE0-Nano.

Dado que la placa de desarrollo utilizada no cuenta con un convertidor UART a USB integrado, fue necesario la adaptación de un periférico externo que cumpliera esta función para poder establecer la comunicación serial entre el computador y la FPGA. Para esto, se empleó el convertidor PL2303 que usa el estándar de comunicación

serial RS-232. Esta elección permitió una conexión estable para el envío y recepción de datos entre ambos dispositivos. La adaptación del convertidor externo proporcionó una solución eficiente y funcional para habilitar la comunicación serial requerida en el proyecto.

La evaluación del filtro digital implementado en términos de precisión y eficiencia en la atenuación del ruido Gaussiano ha arrojado resultados satisfactorios. Durante las pruebas realizadas, se observó que el filtro logra reducir de manera significativa las fluctuaciones presentes en la señal, preservando la integridad de la información relevante y demostrando el cumplimiento del objetivo principal de la etapa de filtrado.

10. RECOMENDACIONES

Para trabajos futuros, se recomienda implementar un convertidor ADC antes de la etapa de filtrado. Esto permite al sistema poder leer señales analógicas en los pines de entrada de la tarjeta, lo cual representa una ventaja puesto que puede obtener señales provenientes de cualquier sensor y aplicar la teoría de filtros digitales en cualquier otra área del conocimiento, como, por ejemplo, en aplicaciones de audio o control de plantas industriales.

Para investigaciones futuras, es recomendable que los tesistas diseñen el sistema en lenguaje de descripción de hardware. Esto brindaría muchas ventajas en el procesamiento de los datos, puesto que el filtrado de la señal resultante es directamente sobre el hardware y no es necesario bajar de niveles computacionalmente, como lo es en el caso de trabajar con microprocesadores.

Se recomienda también implementar el cálculo de la reactividad directamente en la FPGA; para esto es necesario una ardua investigación de todos los métodos de aproximación de la ecuación inversa de la cinética puntual; puesto que es importante escoger un método óptimo y eficiente para el cálculo de este parámetro en tiempo real.

BIBLIOGRAFÍA

[1] Proakis, J. G., & Manolakis, D. G. (2007). *Tratamiento digital de señales*. PRENTICE HALL.

[2] Sedra, A. S., & Smith, K. C. (1987). *Microelectronic Circuits*. Oxford University Press, USA.

[3] Savant, C. J., Roden, M. S., & Carpenter, G. L. (1998). *Diseño electrónico: circuitos y sistemas*.

[4] Stacey, W. M. (2007). *Nuclear Reactor Physics*. John Wiley & Sons.

[5] Glasstone/sesonske. (1998). *Nuclear Reactor Engineering, 4e Vol. I : Reactor Design Basics*.

[6] Yeary, M., & Griswold, N. C. (2002). Adaptive IIR filter design for single sensor applications. *IEEE Transactions on Instrumentation and Measurement*. https://doi.org/10.1109/19.997822

[7] Pan, S. (2011). Evolutionary Computation on Programmable Robust IIR Filter Pole-Placement Design. *IEEE Transactions on Instrumentation and Measurement*, *60*(4), 1469-1479. https://doi.org/10.1109/tim.2010.2086850

[8] Mejía-Mejía, E., & Kyriacou, P. A. (2023). Effects of noise and filtering strategies on the extraction of pulse rate variability from photoplethysmograms. *Biomedical Signal Processing and Control*, *80*, 104291. https://doi.org/10.1016/j.bspc.2022.104291

[9] Rose, J., Gamal, A. E., & Sangiovanni-Vincentelli, A. (1993). Architecture of field-programmable gate arrays. *Proceedings of the IEEE*, *81*(7), 1013-1029. https://doi.org/10.1109/5.231340

[10] Roy, S., & Banerjee, P. (2005). An Algorithm for Trading Off Quantization Error with Hardware Resources for MATLAB-Based FPGA Design. *IEEE Transactions on Computers*, *54*(7), 886-896. https://doi.org/10.1109/tc.2005.106

[11] Wu, J., & Zhang, C. (2011). Half-Band Filter Design Based on MATLAB and FPGA. *Applied Mechanics and Materials*, *130-134*, 2027-2030. https://doi.org/10.4028/www.scientific.net/amm.130-134.2027

[12] Roonizi, E. K. (2023). Kalman filter/smoother-based design and implementation of digital IIR filters. *Signal Processing*, *208*, 108958. https://doi.org/10.1016/j.sigpro.2023.108958

[13] Bulić, P., Guštin, V., Šonc, D., & Štrancar, A. (2013). An FPGA-based integrated environment for computer architecture. *Computer Applications in Engineering Education*, *21*(1), 26-35. https://doi.org/10.1002/cae.20448

[14] De Jesus Rangel-Magdaleno, J., Rivera-Guillen, J. R., De Jesus Romero-Troncoso, R., & Osornio-Rios, R. A. (2013). FPGA-Matlab-based open core for three-time controllers in automatic control applications. *Computer Applications in Engineering Education*, *21*(S1), E132-E140. https://doi.org/10.1002/cae.20526

[15] Guštin, V., & Bulić, P. (2006). Learning computer architecture concepts with the FPGA-based “Move” microprocessor. *Computer Applications in Engineering Education*, *14*(2), 135-141. https://doi.org/10.1002/cae.20072

[16] Maxfield, C. (2004). *The Design Warrior’s Guide to FPGAs: Devices, Tools and Flows*. Elsevier.

[17] Etiemble, D., Bouaziz, S., & Lacassagne, L. (2005). *Customizing 16-bit floating point instructions on a NIOS II processor for FPGA image and media processing*. https://doi.org/10.1109/estmed.2005.1518073

[18] Ansari, S. G. (1991). Development of on-line reactivity meter for nuclear reactors. *IEEE Transactions on Nuclear Science*, *38*(4), 946-952. https://doi.org/10.1109/23.83857

[19] Shimazu, Y., Nakano, Y., & Gakuhari, K. (1994). Real Time Measurement of Large Negative Reactivities by a Modified Digital Reactivity Meter. *Journal of Nuclear Science and Technology*, *31*(5), 479-483. https://doi.org/10.1080/18811248.1994.9735178

[20] Kitano, A., Itagaki, M., & Narita, M. (2000). Memorial-Index-Based Inverse Kinetics Method for Continuous Measurement of Reactivity and Source Strength. *Journal of Nuclear Science and Technology*, *37*(1), 53-59. https://doi.org/10.1080/18811248.2000.9714866

[21] Tamura, S. (2003). Signal Fluctuation and Neutron Source in Inverse Kinetics Method for Reactivity Measurement in the Sub-critical Domain. *Journal of Nuclear Science and Technology*, *40*(3), 153-157. https://doi.org/10.1080/18811248.2003.9715345

[22] Shimazu, Y., Unesaki, H., & Suzuki, N. (2003). Subcriticality Monitoring with a Digital Reactivity Meter. *Journal of Nuclear Science and Technology*, *40*(11), 970-974. https://doi.org/10.1080/18811248.2003.9715440

[23] Dashuk, S. P., & Borisov, V. F. (2007). A simulator of nuclear reactor kinetics. *Instruments and Experimental Techniques*. https://doi.org/10.1134/s0020441207040094

[24] Diaz, D., Martinez, A. S., & De C Da Silva, F. (2007). Formulation for the Calculation of Reactivity Without Nuclear Power History. *Journal of Nuclear Science and Technology*, *44*(9), 1149-1155. https://doi.org/10.1080/18811248.2007.9711358

[25] Díaz, D. C., Martinez, A. S., & De C Da Silva, F. (2008). Calculation of reactivity using a finite impulse response filter. *Annals of Nuclear Energy*, *35*(3), 472-477. https://doi.org/10.1016/j.anucene.2007.07.002

[26] Antolin, M., Martinez, A. S., Fernando, F. S., & Palma, D. A. (2013). Calculation of reactivity in subcritical reactors using the method of partial derivatives. *Annals of Nuclear Energy*, *60*, 34-38. https://doi.org/10.1016/j.anucene.2013.03.028

[27] Díaz, D. C., Londoño, H. F. B., & Jimenez, J. (2014). Filtros FIR y Pasa Bajo Para Calcular la Reactividad Nuclear. *Scientia et technica*, *19*(2), 217-222. https://doi.org/10.22517/23447214.9031

[28] Suescún-Díaz, D., Bonilla-Londono, H. F., & Figueroa-Jimenez, J. H. (2016). Savitzky–Golay filter for reactivity calculation. *Journal of Nuclear Science and Technology*. https://doi.org/10.1080/00223131.2015.1082949

[29] Huo, X., Fan, Z., Xu, L., Chen, X., Hu, Y. H., & Yu, H. (2019). A new and efficient method to measure reactivity in a nuclear reactor. *Annals of Nuclear Energy*, *133*, 455-457. https://doi.org/10.1016/j.anucene.2019.05.047

[30] Messai, A., Abdellani, I., & Mellit, A. (2022). FPGA-based real-time implementation of a digital reactivity-meter. *Progress in Nuclear Energy*, *150*, 104313. https://doi.org/10.1016/j.pnucene.2022.104313

[31] Suescún-Díaz, D., Ule-Duque, G., & Escobar, F. H. (2020). Novel approach to solving the inverse equation of point kinetics by the Bernoulli number generalisation method. *Journal of Nuclear Science and Technology*, *57*(8), 989-999. https://doi.org/10.1080/00223131.2020.1742813

ANEXOS

A. Solución Analítica para señal Exponencial

Ecuación Inversa de Cinética Puntual

$$\rho(t) = \beta + \frac{\Lambda}{P(t)}\frac{dP(t)}{dt} - \frac{P_0}{P(t)}\sum_{i=1}^{6}\beta_i e^{-\lambda_i t} - \frac{1}{P(t)}\sum_{i=1}^{6}\int_0^t \lambda_i \beta_i P(t') e^{-\lambda_i (t-t')} dt' \quad \text{(A.1)}$$

Donde $P(t) = e^{wt}$

$$\int_0^t \lambda_i \beta_i P(t') e^{-\lambda_i (t-t')} dt' \quad \text{(A.2)}$$

$$= \lambda_i \beta_i \int_0^t P(t') e^{-\lambda_i t} e^{\lambda_i t'} dt' \quad \text{(A.3)}$$

Haciendo $t' = \tau$ y reemplazando el valor de $P(t)$

$$\lambda_i \beta_i e^{-\lambda_i t} \int_0^t e^{w\tau} e^{\lambda_i \tau} d\tau \quad \text{(A.4)}$$

$$\lambda_i \beta_i e^{-\lambda_i t} \int_0^t e^{(w+\lambda_i)\tau} d\tau \quad \text{(A.5)}$$

Resolviendo la integral

$$\rho(t) = \lambda_i \beta_i e^{-\lambda_i t} \frac{e^{t(w+\lambda_i)} - 1}{w + \lambda_i} \quad \text{(A.6)}$$