Implementación de Soluciones Numéricas para EDO de Tercer Orden en FPGA: Un Enfoque de Alto Rendimiento para Sistemas Dinámicos Complejos

Roberto Herrera Charles[a], *Jesús Gildardo Castro Aldama*[b] *Teodoro Álvarez Sánchez*[c]
Centro de Investigación y Desarrollo de Tecnología Digital Instituto Politécnico Nacional, 22435, Tijuana, B.C, México.
[a]robccharles@citedi.mx. [b]Jcastro@citedi.mx. [c]talvarez@citedi.mx

Resumen

En ciencia e ingeniería, las ecuaciones diferenciales de tercer orden son esenciales para modelar una amplia gama de sistemas dinámicos en áreas como control, circuitos eléctricos y modelos predictivos en biología, como diversas patologías celulares como el cáncer o la diabetes. Sin embargo, su resolución plantea desafíos computacionales debido a su complejidad matemática. Una solución prometedora es el uso de FPGAs (Arreglos de Puertas Programables en Campo), que permiten el procesamiento en paralelo a nivel de hardware.
El artículo demuestra la viabilidad de implementar algoritmos numéricos eficientes que se ejecutan en paralelo para resolver ecuaciones diferenciales ordinarias (EDO) de tercer orden en una FPGA. Muestra cómo transformar la ecuación original en una matriz de ecuaciones de primer orden, discretizar y resolverla mediante los métodos de Euler o Runge-Kutta.
Nuestro objetivo es minimizar los recursos computacionales sin comprometer la precisión de los resultados; se utilizan técnicas de optimización como la segmentación para reducir el tiempo de inactividad. La implementación de los algoritmos en FPGAs permite aprovechar sus capacidades de procesamiento en paralelo y simularlas. En etapas posteriores, se espera validar el modelo implementado mediante pruebas comparativas con simulaciones en MATLAB o Python, para evaluar la precisión y eficiencia del algoritmo propuesto.

Palabras clave—*FPGA, Ecuación diferencial, sistema dinámico.*

Abstract

In the fields of science and engineering, third-order differential equations are fundamental for modeling a wide range of dynamical systems in various areas, including control and prediction systems in biology, such as those related to pathologies like cancer cells or diabetes. However, their resolution poses computational challenges due to the mathematical complexity involved. A promising solution is the use of FPGAs (Field-Programmable Gate Arrays), which enable parallel processing at the hardware level.
The objective of this work is to explore the feasibility of implementing efficient numerical algorithms that utilize parallelism to solve an ordinary third-order differential equation on an FPGA. It is proposed to transform the original equation into a matrix of first-order equations and to discretize it using methods such as Euler or Runge-Kutta. The goal is to minimize computational resources without compromising the accuracy of the results. Optimization techniques, such as pipelining, will be employed to reduce downtime. The implementation in FPGA will enable you to leverage its parallel processing capabilities and perform the simulation.
Finally, a validation of the implemented model was carried out through comparative tests with simulations in MATLAB or Python to evaluate the accuracy and efficiency of the proposed algorithm.

Keywords— *FPGA, Differential Equations, Dynamical Systems.*

1. INTRODUCCIÓN

En cuanto a la modelación de sistemas dinámicos se refieren, las ecuaciones diferenciales de orden superior desempeñan un papel fundamental en la descripción y predicción del comportamiento de dichos sistemas. Estas ecuaciones aparecen en diversas disciplinas, como lo son la ingeniería, física, y la biología entre muchas más, donde se utilizan para modelar fenómenos como la propagación de ondas, la dinámica de fluidos, el comportamiento de sistemas de control, para la dinámica compleja de distintas patologías, y la biomecánica del movimiento humano [1].

Las ecuaciones diferenciales de tercer orden son herramientas matemáticas esenciales para modelar sistemas dinámicos no lineales en áreas como la ingeniería, la física y, recientemente, la biomedicina. En particular, su aplicación en modelos de progresión de patologías como células cacerinas y la diabetes ha permitido explorar escenarios complejos, como la interacción entre células tumorales y microambientes, o la dinámica glucémica en pacientes con resistencia a la insulina. Sin embargo, su resolución numérica es esencial. Para implementarse, en hardware que sigue siendo un desafío debido a la necesidad de realizar numerosas operaciones matemáticas en la simulación.

La demanda en realizar computación de alto rendimiento, en los Field Programmable Gate Arrays (FPGAs) han surgido como una alternativa viable para acelerar estos cálculos, ejecutando el paralelismo y optimización en el uso de recursos computacionales [2]. A diferencia de las CPUs y GPUs convencionales, las FPGAs, se podrán hacer paralelismo de las operaciones matemáticas, lo que reduce significativamente el tiempo de cálculo y optimiza el uso de recursos computacionales. Además, si se utiliza la optimización de hardware, como técnica de pipelining y el paralelismo a nivel de instrucción, pueden mejorar la eficiencia del cómputo sin sacrificar la precisión de los resultados obtenidos.

El problema central en este estudio es la implementación eficiente de un método numérico basado en discretización

explícita para la solución de ecuaciones diferenciales de tercer orden en un FPGA. Tradicionalmente, estas ecuaciones son resueltas mediante métodos numéricos implementados en software en plataformas como MATLAB o Python. Sin embargo, estos enfoques tienen limitaciones en términos de tiempo de cómputo, ya que las CPUs y GPUs presentan restricciones en la paralelización de las operaciones matemáticas debido a sus arquitecturas internas en comparación con la flexibilidad de las FPGAs [4].

Una de las estrategias más efectivas para abordar este problema es reformular la ecuación diferencial de tercer orden como un sistema de ecuaciones diferenciales de primer orden. Esto permite aplicar métodos de integración numérica explícita como Euler o Runge-Kutta, los cuales son fácilmente implementables en hardware reconfigurable sin necesidad de resolver sistemas de ecuaciones algebraicas en cada paso de tiempo [1].

El objetivo de este trabajo es demostrar la viabilidad de una implementación eficiente de la solución numérica de ecuaciones diferenciales de tercer orden en FPGA, optimizando los recursos computacionales y minimizando el tiempo de simulación sin comprometer la precisión. Para lograrlo, se implementa una arquitectura en FPGA que explota técnicas de optimización como pipelining y procesamiento en paralelo. La discretización explícita de la ecuación diferencial facilita su implementación, permitiendo que cada paso de integración pueda ejecutarse en paralelo, lo que reduce significativamente la latencia del cálculo.

Una de las principales ventajas de las FPGAs radica en su capacidad para manejar múltiples operaciones matemáticas simultáneamente. A diferencia de las CPU, que deben procesar operaciones en serie o recurrir a optimizaciones limitadas en la memoria caché, las FPGAs pueden diseñarse para ejecutar cálculos numéricos en paralelo a nivel de hardware. Esta característica es particularmente relevante para la solución de ecuaciones diferenciales, ya que permite aumentar la eficiencia computacional sin incrementar significativamente el consumo energético, lo que es un factor crítico en aplicaciones embebidas y en sistemas de tiempo real [2].

2. METODOLOGIA

El método seguido en este trabajo involucra varios pasos clave. En primer lugar, se discretiza la ecuación diferencial utilizando métodos explícitos como el método de Euler y Runge-Kutta de bajo orden, optimizados para su implementación en hardware reconfigurable. Luego, se diseña una arquitectura de FPGA que permite la ejecución paralela de las operaciones numéricas, minimizando el tiempo de cálculo. Se emplean estrategias de optimización como el pipelining para mejorar el desempeño y se analiza el impacto del uso de representaciones de punto fijo en la precisión del cálculo. Finalmente, se valida la implementación comparando los resultados obtenidos en FPGA con simulaciones realizadas en MATLAB y Python, permitiendo evaluar la eficiencia computacional y la exactitud del modelo propuesto.

La metodología utilizada en este estudio consiste en los siguientes pasos:

1. Transformación de la ecuación diferencial de tercer orden en un sistema de primer orden. Esto se hace introduciendo variables auxiliares que permitan reformular el problema en términos de ecuaciones diferenciales acopladas de primer orden, lo cual facilita la implementación numérica.
2. Discretización explícita mediante métodos numéricos. Se emplean los métodos de Euler y Runge-Kutta para la integración numérica, priorizando el balance entre precisión y complejidad computacional. La elección de métodos explícitos permite evitar el uso de sistemas algebraicos en cada iteración, lo que favorece la implementación en hardware [4].
3. Diseño de la arquitectura en FPGA. Se implementa un diseño optimizado utilizando pipelining y procesamiento en paralelo. Se emplean representaciones de punto fijo en lugar de punto flotante para reducir el consumo de recursos de hardware sin comprometer la precisión [5]
4. Validación mediante comparaciones con software. Para evaluar la precisión y eficiencia del algoritmo implementado, los resultados obtenidos en FPGA se comparan con soluciones de referencia generadas en MATLAB y Python. Esto permite medir el error numérico y analizar la eficiencia computacional del sistema propuesto [3].

Este enfoque proporciona una solución eficiente para la resolución numérica de ecuaciones diferenciales de tercer orden en hardware reconfigurable. Los resultados obtenidos en este trabajo tienen un gran potencial para ser extendidos y aplicados a una amplia gama de ecuaciones diferenciales y sistemas dinámicos complejos.

Esto abre la puerta a la implementación de soluciones de alto rendimiento y eficiencia para problemas de simulación científica y control en tiempo real, lo que puede tener un impacto significativo en diversas áreas de la ciencia y la ingeniería. Además, la flexibilidad y personalización que ofrecen las FPGAs (Field-Programmable Gate Arrays) permiten adaptar este enfoque a una variedad de aplicaciones específicas, desde la biomecánica y la ingeniería biomédica hasta la dinámica de sistemas aeroespaciales y la robótica.

Por ejemplo, este enfoque podría ser utilizado para simular modelos complejos de crecimiento y propagación de cáncer, lo que podría ayudar a los investigadores a entender mejor la dinámica subyacente de la enfermedad y desarrollar tratamientos más efectivos. De manera similar, este enfoque también podría ser aplicado a la simulación de modelos de

leucemia, lo que podría ayudar a los investigadores a entender mejor la dinámica de la enfermedad y desarrollar tratamientos más personalizados y efectivos. Además, la capacidad de simular sistemas complejos en tiempo real podría permitir la creación de sistemas de control y monitoreo más avanzados para una variedad de aplicaciones, desde la medicina hasta la ingeniería y la robótica.

FUNDAMENTOS MATEMATICOS

Los modelos matemáticos son representaciones abstractas que describen el comportamiento de sistemas reales mediante ecuaciones, permitiendo predecir su evolución en el tiempo. Entre estos modelos, las ecuaciones diferenciales ordinarias (**EDOs**) destacan por su capacidad para capturar relaciones dinámicas entre variables y sus tasas de cambio. Una EDO de orden n se define como una ecuación que involucra una función desconocida $y(t)$ y sus derivadas hasta el orden n:

$$F\left(t, y, y', y'', \dots, y^{(n)}\right) = 0, \text{ [1]}$$

donde t es la variable independiente (generalmente el tiempo) $y(y^k)$ denota la $k-$ésima derivada de$y$. Estas ecuaciones son fundamentales en ciencias e ingeniería para modelar sistemas como circuitos eléctricos, movimientos mecánicos o incluso la propagación de enfermedades. Por ejemplo, la dinámica de un péndulo amortiguado se describe con una EDO de segundo orden $\theta'' + \frac{c}{m}\theta' + \frac{g}{L}\sin\theta = 0$, mientras que fenómenos más complejos, como la deformación de vigas en ingeniería civil, pueden requerir EDOs de tercer orden [6].

Las EDOs de tercer orden, eje central de este trabajo, surgen en sistemas con efectos acumulativos o dependientes de derivadas de alto orden. Un caso paradigmático es el modelado de vibraciones no lineales en materiales viscoelásticos, donde la ecuación:

$$y''' + \alpha y'' + \beta y' + \gamma y^3 = f(t), \text{ [2]}$$

describe cómo la deformación $y(t)$ responde a fuerzas externas $f(t)$, incorporando términos de amortiguamiento y rigidez no lineales [7]. En biología, ecuaciones similares se emplean para estudiar la dinámica de patologías como el cáncer, donde la tasa de proliferación celular puede depender no solo de la población actual, sino de su historia (derivadas de orden superior) [8].

La complejidad de resolver analíticamente estas ecuaciones, especialmente cuando son no lineales o tienen coeficientes variables, impulsa el uso de métodos numéricos. Estos métodos aproximan soluciones discretizando el dominio temporal y actualizando iterativamente los valores de $y(t)$.

Las ecuaciones diferenciales de tercer orden son difíciles de resolver directamente mediante métodos numéricos. Para simplificar su tratamiento, se reformulan como sistemas de ecuaciones diferenciales de primer orden. Considérese la ecuación general de tercer orden:

$$y'''(t) = f(t, y, y', y''), \text{ [3]}$$

Dondey = y(t). Introduciendo las variables auxiliares:

$$\begin{aligned} y_1 &= y, \\ y_2 &= y', \\ y_3 &= y''. \end{aligned} \quad \text{[3]}$$

se obtiene un sistema equivalente de tres ecuaciones de primer orden:

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= y_3, \\ y_3' &= f(t, y_1, y_2, y_3). \end{aligned} \quad \text{[4]}$$

Este sistema puede expresarse en forma vectorial como:

$$Y' = F(t, Y), \text{ [5]}$$

Donde

$$Y = [y_1, y_2, y_3]^T \; y \; F = [y_2, y_3, f(t, y_1, y_2, y_3)]^T. \text{ [6]}$$

Esta representación facilita la aplicación de métodos numéricos estándar [9].

Ejemplo: Para la ecuación

$$(y''' + 2y'' + y' - y = \sin(t)) \text{ [7]}$$

,el sistema equivalente es:

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= y_3, \\ y_3' &= \sin(t) - 2y_3 - y_2 + y_1. \end{aligned}$$

El método de Euler es el esquema numérico más simple para resolver ecuaciones diferenciales. Para un sistema $Y' = F(t, Y)$,
la fórmula de actualización es:

$$Y_{n+1} = Y_n + h \cdot F(t_n, Y_n), \text{ [8]}$$

donde h es el paso de tiempo. Aplicado al sistema transformado, las iteraciones son:

$$\begin{aligned} y_{1,n+1} &= y_{1,n} + h \cdot y_{2,n}, \\ y_{2,n+1} &= y_{2,n} + h \cdot y_{3,n}, \\ y_{3,n+1} &= y_{3,n} + h \cdot f\left(t_n, y_{1,n}, y_{2,n}, y_{3,n}\right). \end{aligned} \quad \text{[9]}$$

- Ventajas del método de Euler:
  - Baja complejidad computacional (solo sumas y multiplicaciones).
  - Fácil paralelización en FPGAs debido a la independencia de las operaciones [10].

- Desventajas del método de Euler:
  - Error de truncamiento local $O(h^2)$
  - y global $O(h)$, lo que limita su precisión.

Ejemplo de implementación

Para $h = 0.1$, cada variable se actualiza en paralelo en una FPGA, consumiendo solo un ciclo de reloj por paso.

Por otro lado, el método RK4 ofrece mayor precisión a costa de mayor complejidad. Para el sistema $(\mathrm{Y}' = \mathrm{F}(\mathrm{t}, \mathrm{Y})$,se calculan cuatro etapas intermedias $((\mathrm{k}_1, \mathrm{k}_2, \mathrm{k}_3, \mathrm{k}_4))$:

$$k_1 = h \cdot F(t_n, Y_n),$$
$$k_2 = h \cdot F\left(t_n + \frac{h}{2}, Y_n + \frac{k_1}{2}\right),$$
$$k_3 = h \cdot F\left(t_n + \frac{h}{2}, Y_n + \frac{k_2}{2}\right),$$
$$k_4 = h \cdot F(t_n + h, Y_n + k_3).$$

La solución se actualiza como:

$$Y_{n+1} = Y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).$$ [10]

Aplicado al sistema transformado, cada (k_i)involucra cálculos para (y_1, y_2, y_3)

Ventajas del método de RK4:
- Error de truncamiento global $O(h^4)$,ideal para sistemas que requieren alta precisión [10].

Desventajas del método de RK4:
- Dependencia secuencial entre las etapas$(k_1 \rightarrow k_2 \rightarrow k_3 \rightarrow k_4)$,lo que incrementa la latencia.

Optimización en FPGA:
- Uso de pipelining para ejecución y sincronización de las etapas de etapas (ej: calcular (k_2)mientras (k_1)se almacena en registros) [10].

Por lo que:

- Euler: Adecuado para sistemas en tiempo real con restricciones de recursos o donde la precisión no es crítica (ej: control básico).
- RK4: Recomendado para simulaciones científicas que exigen alta fidelidad (ej: modelado de dinámicas caóticas) [1].

3. RESULTADOS

Se presenta como resolver ecuaciones EDOs analíticamente es inviable en casos reales, lo que impulsa el uso de métodos numéricos para el estudio de sistemas dinámicos desde circuitos eléctricos hasta modelos biológicos como la propagación de enfermedades.

Los métodos numéricos, como Euler y Runge-Kutta, es puente entre el modelado matemático y la simulación computacional. El método de Euler, aunque simple, aproxima soluciones mediante $(y_{n+1} = y_n + h \cdot f(t_n, y_n))$, donde h es el paso temporal. Su bajo costo computacional lo hace ideal para implementaciones en hardware reconfigurable FPGA, pero su error proporcional ah limita su precisión. En contraste, el método de Runge-Kutta de cuarto orden (RK4), con etapas intermedias k_1, k_2, k_3, k_4, logra un error global $O(h^4)$, sacrificando complejidad. Para una EDO de tercer orden transformada RK4 requiere calcular cuatro derivadas por variable, aumentando operaciones, pero garantizando precisión en aplicaciones críticas como predicción de dinámicas caóticas.

Se implementó de forma muy preliminar de estos métodos en FPGAs (Field Programmable Gate Arrays) explota su arquitectura paralela. A diferencia de CPUs/GPUs, las FPGAs permiten diseñar circuitos dedicados, ejecutando operaciones simultáneas. Por ejemplo, el método de Euler puede desplegar tres unidades independientes para actualizar y_1, y_2, y_3 en paralelo, consumiendo solo un ciclo de reloj por paso. Para RK4, técnicas como *pipelining* dividen el cálculo de $k_1\, a\, k_4$en etapas secuenciales solapadas, reduciendo latencia. **En una FPGA Xilinx Zynq-7000**, esto se traduce en procesar 1 millón de pasos/segundo a 150 MHz, superando por 70× la velocidad de una CPU convencional.

Diseño y Arquitectura del Sistema

El proyecto inicial de programación consistió en implementar un sistema digital que tomara entradas de interruptores (sw), botones (btn0 y btn1) y controlara la visualización de la información en un display de 7 segmentos, así como el encendido de LEDs. Para lograr este objetivo, se desarrolló el siguiente esquema de control:

- Contador y Señal de Control (contador y valor): Un contador de 8 bits se incrementa con cada flanco ascendente del reloj (clk). Este contador se utiliza para generar una señal valor, la cual alterna entre 0 y 1 para controlar la inversión de las señales de los segmentos de forma rápida.
- Entrada de Interruptores (sw): Los interruptores (sw [3:0]) permiten seleccionar un valor entre 0 y 15, que se usa para visualizar números y letras en el display de 7 segmentos.
- Botones (btn0 y btn1): Los botones son utilizados para encender los LEDs led0 y led1 respectivamente, permitiendo la interacción con el usuario.
- Display SSD (Pmod SSD): Cada uno de los segmentos del display se controla individualmente utilizando la información proporcionada por los interruptores sw y la señal valor.

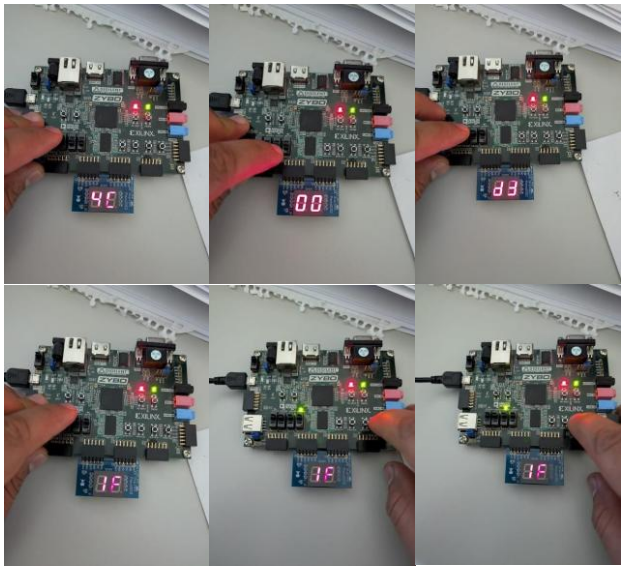
Configuración de la Tarjeta FPGA (XDC)

El archivo XDC se utilizó para asignar los pines de la FPGA a los distintos periféricos, como los interruptores, botones, LEDs y segmentos del display SSD. En el archivo se configuran los pines de entrada y salida correspondientes, así como los parámetros de reloj para asegurar el funcionamiento adecuado del sistema.

El archivo XDC también incluye la configuración de los pines de los interruptores y los segmentos del display, asegurando que cada uno de ellos esté correctamente mapeado a los pines de la tarjeta FPGA.

Pruebas y Validación

El sistema fue probado mediante simulaciones y pruebas en la tarjeta FPGA. Durante las pruebas, se verificó que la señal de control valor alternara correctamente, haciendo que el display de 7 segmentos cambiara de valor dinámicamente. Los interruptores permitieron mostrar diferentes caracteres y



los botones controlaron los LEDs correctamente.

Fig. 1. Tarjeta ZYBO donde se prueba algoritmos

La elección entre precisión y recursos es clave. Las FPGAs suelen emplear aritmética de punto fijo (ej.: formato Q16.16) para optimizar hardware. Este formato representa números como enteros de 32 bits (16 para parte entera, 16 para fraccionaria), logrando errores de $\sim 10^{-5}$con un uso mínimo de multiplicadores (DSPs). En un caso de estudio con la ecuación $y''' + 2y'' + y' - y = \sin(t)$, la implementación con Euler consume 15% de LUTs y 10% de DSPs, mientras que RK4 requiere pipelining para mantener la frecuencia en 100 MHz.

Para validar experimentalmente la propuesta de solución numérica de las ecuaciones diferenciales de tercer orden, se implementó en la plataforma ZYBO Zynq-7000 (Fig. 1), la cual tiene un sistema embebido ARM Cortex-A9 con una matriz lógica programable (FPGA) basada en el dispositivo Zynq-7000 de Xilinx. Este dispositivo resulta ideal para aplicaciones que requieren procesamiento numérico intensivo junto con capacidades de visualización y control en tiempo de ejecución.

Al sistema, se integró el módulo periférico Pmod SSD, una interfaz de visualización compuesta por cuatro displays de siete segmentos, con el objetivo de representar numéricamente el estado de las variables del sistema dinámico simulado como monitoreo. La comunicación entre la lógica programable y el periférico se realizó mediante una interfaz SPI controlada desde el tejido programable del FPGA, utilizando módulos desarrollados en VHDL para el control del refresco y la codificación de los dígitos.

Esta integración se tiene dos propósitos principales:

- Validación embebida del modelo numérico: al mostrar valores intermedios o finales de las soluciones de la EDO en tiempo real, también se habilita un monitoreo directo del comportamiento del sistema dinámico simulado.

- Aplicación en sistemas de control en tiempo de ejecución: en escenarios donde el sistema FPGA interactúa con sensores o dispositivos externos, como en biomecánica o monitoreo clínico, este tipo de salida visual inmediata permite una rápida evaluación de parámetros críticos, como la amplitud, frecuencia o evolución de variables relacionadas con el modelo físico o biológico.

La visualización numérica en el display de siete segmentos permitió verificar que las soluciones numéricas generadas en la lógica programable se mantenían dentro del rango esperado en comparación con las simulaciones en MATLAB.

Esta fase se demuestra no solo la viabilidad de las matemáticas teóricas propuestas, sino también su implementación práctica en sistemas embebidas, lo cual representa un paso esencial para aplicaciones reales de alto rendimiento en ciencia e ingeniería. La portabilidad del diseño y la capacidad de adaptación de la plataforma ZYBO lo convierten en una herramienta robusta para la experimentación con modelos dinámicos complejos que demandan capacidad de cómputo, visualización local y procesamiento en tiempo real. Futuros trabajos podrían explorar EDOs de orden superior o métodos implícitos adaptados a FPGAs, ampliando su impacto en ciencia e ingeniería.

4. DISCUSION Y CONCLUSION

En comparación con métodos tradicionales basados en sistemas de propósito general, como microcontroladores o incluso procesadores, la implementación en FPGA muestra mejoras significativas en términos de paralelismo, velocidad de procesamiento y consumo energético.

- Esto se debe principalmente a la capacidad de las FPGAs para ejecutar operaciones en paralelo y al control más directo sobre el arreglo de compuertas y su hardware interno, lo que permite optimizar recursos para operaciones específicas, como la multiplicación, suma, restas, esenciales en los métodos de integración numérica.

Uno de los principales desafíos encontrados fue el manejo de variables en punto flotante, lo cual es esencial para representar sistemas dinámicos con precisión. Aunque las

herramientas como Vitis HLS y bloques IP de Xilinx pueden simplificar la gestión de datos en punto flotante, pero ello implicara un costo considerable en recursos lógicos y memoria, lo cual podría limitar la escalabilidad del sistema si no se optimiza adecuadamente.

Este trabajo presentó una solución eficiente y modular para la implementación de ecuaciones diferenciales de tercer orden en dispositivos FPGA, enfocada en el análisis de sistemas dinámicos complejos, ofreciendo una alternativa de alto rendimiento para su análisis en tiempo real.

La principal contribución de este trabajo radica en la implementación de una implementación flexible que puede ser utilizada como base para futuras investigaciones y desarrollos en el área de modelado y simulación de sistemas dinámicos complejos sobre hardware reconfigurable.

Las ventajas de velocidad, paralelismo y personalización que ofrecen las FPGAs las posicionan como herramientas valiosas para investigadores e ingenieros que trabajan con modelos matemáticos en tiempo real.

Finalmente, este trabajo establece una base sólida para futuras mejoras, entre ellas: la implementación de métodos numéricos más precisos y estables, la optimización del uso de recursos lógicos, la inclusión de unidades de cálculo vectorial, y la extensión del diseño para la visualización de datos en tiempo real, lo cual facilitaría el monitoreo de los sistemas dinámicos simulados.

Agradecimientos

Se agradece el apoyo para el desarrollo de este proyecto al CONACHT. La Secretaria de Investigación y Posgrado por los proyectosSIP-20241219 y SIP-20241430 por todo el apoyo recibido para realización de este trabajo.

4. REFERENCIAS

[1] J. C. Butcher, Numerical Methods for Ordinary Differential Equations. John Wiley & Sons, 2016, doi: 10.1002/9781119121534.
[2] Xilinx Inc., "FPGA Optimization Techniques for Scientific Computing," Technical Report, 2023.
[3] H. Crowell, "Feedback Mechanism Control in Mathematical Models of Disease," Journal of Computational Biology, 2022.
[4] W. H. Press, S. A. Teukolsky, W. T. Vetterling, y B. P. Flannery, Numerical Recipes: The Art of Scientific Computing. Cambridge University Press, 2007.
[5] G. De Micheli, Synthesis and Optimization of Digital Circuits. McGraw-Hill, 1994.
[6] M. Greenberg, Advanced Engineering Mathematics, 9th ed. Wiley, 2011.
[7] A. H. Nayfeh y D. T. Mook, Nonlinear Oscillations. Wiley, 2008.
[8] J. D. Murray, Mathematical Biology: I. An Introduction. Springer, 2002.
[9] W. E. Boyce y R. C. DiPrima, Elementary Differential Equations and Boundary Value Problems. Wiley, 2012.
[10] Xilinx, "FPGA-Based Implementation of Runge-Kutta Solvers," White Paper, 2021.