ACELERADOR HARDWARE PARA EMULAR EL PROCESO DE LA RUTA METABÓLICA GLUCÓLISIS EN UNA CÉLULA

JOAO ANDREY RINCON PERLAZA

UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
SANTIAGO DE CALI
2025

ACELERADOR HARDWARE PARA EMULAR EL PROCESO DE LA RUTA METABÓLICA GLUCÓLISIS EN UNA CÉLULA

JOAO ANDREY RINCON PERLAZA

Trabajo de grado para optar por el título de Ingeniero Electrónico

Director
Jaime Velasco Medina, Ph.D.

UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
SANTIAGO DE CALI
2025

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento a todas las personas que hicieron posible la culminación de este proceso formativo, primeramente, a mis profesores por su guía y conocimientos y a mis compañeros por el apoyo y la amistad compartida en cada etapa de la carrera.

Un agradecimiento especial a mi pareja, por la comprensión infinita, y por ser mi compañía y fortaleza en este final de camino, y finalmente, a mi madre, pilar fundamental de mi vida, por su paciencia, amor incondicional y creer siempre en mí. Este logro es también de ustedes.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

La simulación de sistemas biológicos complejos mediante software tradicional enfrenta limitaciones significativas en términos de tiempo de cómputo y consumo energético, debido a la naturaleza secuencial de los procesadores de propósito general (CPU). Este trabajo de grado presenta el diseño, implementación y validación de un acelerador hardware basado en FPGA para emular la dinámica de la ruta metabólica de la glucólisis.

Se desarrolló un modelo matemático reducido de cinco reacciones enzimáticas, capturando las características no lineales esenciales como la cinética de saturación de Michaelis Menten y la regulación alostérica de la enzima fosfofructoquinasa (PFK). Este sistema de ecuaciones diferenciales ordinarias (EDOs) fue discretizado mediante el método de Euler y traducido a una arquitectura de hardware digital utilizando el lenguaje VHDL. Para optimizar el rendimiento en la plataforma Terasic DE2-115 (Cyclone IV E), se diseñó una unidad aritmética de punto fijo de 32 bits (formato Q12.20) que incluye multiplicadores segmentados (*pipelined*) y un divisor iterativo basado en el algoritmo de Newton Raphson para resolver la complejidad de la inhibición enzimática.

La validación funcional se realizó mediante una estrategia cruzada, comparando las trazas obtenidas por el hardware (adquiridas mediante el analizador lógico SignalTap II) con un modelo de referencia de alta precisión en python. Los resultados demuestran que el acelerador reproduce la dinámica biológica con una fidelidad superior al 99.5%, logrando un factor de aceleración (*speedup*) de aproximadamente 2,500 veces respecto al tiempo biológico real. El diseño ocupó únicamente el 22% de los elementos lógicos y el 13% de los multiplicadores de la FPGA, demostrando una alta escalabilidad para la futura emulación de tejidos celulares masivos en tiempo real.

Palabras clave: FPGA, Glucólisis, Biología de Sistemas, Aritmética de Punto Fijo, VHDL, Aceleración Hardware, Newton-Raphson.

ABSTRACT

The simulation of complex biological systems using traditional software faces significant limitations regarding computation time and power consumption due to the sequential nature of general-purpose processors (CPUs). This thesis presents the design, implementation, and validation of an FPGA-based hardware accelerator to emulate the dynamics of the glycolysis metabolic pathway.

A reduced mathematical model comprising five enzymatic reactions was developed, capturing essential non-linear characteristics such as Michaelis-Menten saturation kinetics and the allosteric regulation of the Phosphofructokinase (PFK) enzyme. This system of Ordinary Differential Equations (ODEs) was discretized using the Euler method and translated into a digital hardware architecture using VHDL. To optimize performance on the Terasic DE2-115 platform (Cyclone IV E), a 32-bit fixed-point arithmetic unit (Q12.20 format) was designed, featuring pipelined multipliers and an iterative divider based on the Newton-Raphson algorithm to resolve the complexity of enzymatic inhibition.

Functional validation was conducted through a cross-validation strategy, comparing hardware traces (acquired via the SignalTap II logic analyzer) against a high-precision reference model in python. Results demonstrate that the accelerator reproduces biological dynamics with a fidelity exceeding 99.5%, achieving a speedup factor of approximately 2,500 times relative to real biological time. The design utilized only 22% of the logic elements and 13% of the multipliers of the FPGA, demonstrating high scalability for future emulation of massive cellular tissues in real-time.

Keywords: FPGA, Glycolysis, Systems Biology, Fixed-Point Arithmetic, VHDL, Hardware Acceleration, Newton-Raphson.

INTRODUCCIÓN

En las últimas décadas ha existido un gran interés por simular los sistemas biológicos complejos con el propósito de desarrollar herramientas para el área de la medicina, tanto para la creación de fármacos como para la predicción de los efectos de las reacciones químicas en los organismos vivos ante un estímulo.

Estas simulaciones computacionales toman demasiado tiempo y consumen mucha potencia. Entonces, es de gran importancia entender muy bien las propuestas que presentan los artículos de investigación que intentan darle una solución a ese problema.

Sin embargo, no es fácil encontrar una alternativa tecnológica que permita acelerar las simulaciones computacionales y hacer un nuevo aporte en este tópico de investigación. Encontrar una solución es de gran importancia para que en el futuro se obtengan mejores resultados en los diferentes tópicos de investigación en el campo de la medicina.

Encontrar una solución definitiva a este problema requiere de mucho trabajo de investigación, el cual es muy difícil de abordar, entonces en este trabajo de grado vamos a simular solo un proceso biológico de una célula, para ello se trabajará con la glucólisis, el primer proceso catabólico en el metabolismo celular se encarga de extraer energía de los sustratos, siendo catalizada por enzimas producidas por el ADN [1].

Cabe resaltar que la idea en este trabajo de pregrado es desarrollar un acelerador hardware para la simulación computacional del funcionamiento real del proceso biológico de la glucólisis y probar que es una opción viable para solucionar las desventajas de la simulación secuencial por software. Las simulaciones computacionales usando MATLAB y/o programas en Python utilizan unidades de procesamiento como CPUs o GPUs, que enfrentan limitaciones de tiempo cuando aumenta la complejidad de los procesos biológicos.

Aunque se ha avanzado en la simulación de reacciones bioquímicas, es muy difícil simular el funcionamiento de la naturaleza paralela de los procesos biológicos que ocurren al interior de una célula.

Con el propósito de mitigar las desventajas de las simulaciones software, los aceleradores hardware usando Field Programmable Gate Array (FPGA) emergen como una solución prometedora porque permiten emular los procesos de manera paralela, consumiendo mínimos recursos de área y logrando velocidades de procesamiento superiores a las simulaciones software. Este enfoque ofrece un potencial significativo para mejorar los tiempos de simulación y avanzar en la comprensión de los procesos celulares complejos que son inherente paralelos al interior de una célula.

PLANTEAMIENTO DEL PROBLEMA

Entender cómo funcionan los seres vivos es un desafío que aún tiene mucho por descubrir, para ello, se hace esencial estudiar a nivel molecular los organismos vivos, como por ejemplo la célula y todos sus componentes. Aunque se han logrado avances significativos en el tema, no es fácil simular computacionalmente los complejos procesos biológicos de un sistema vivo. Por ejemplo, simular las reacciones químicas que se realizan en las células o visualizar los efectos de algunos medicamentos en una célula. Por el momento, una alternativa es realizar pruebas experimentales.

En el contexto celular, el metabolismo se realiza en dos procesos: el catabolismo que es el encargado de transformar los sustratos que ingresan al cuerpo comúnmente mediante alimentos para generar energía, y el anabolismo que usa esta misma energía para la creación de nuevas moléculas. Al detectar que un sustrato entra en la célula, la glucólisis es el primer proceso catabólico que se presenta, donde se toma la energía del sustrato mediante enzimas catalizadoras producidas por el ADN. [1]

Desde algunos años se ha intentado simular computacionalmente los procesos celulares mediante programas y lenguajes de programación como MATLAB y python; sin embargo, la simulación usando unidades de procesamiento como las CPUs y/o GPUs no logran ser tecnologías computacionales muy adecuadas para este tipo de procesamiento computacional porque la capacidad de procesamiento incrementa demasiado el tiempo de simulación a medida que aumenta el tamaño y complejidad de los procesos moleculares, por lo que es necesario encontrar de qué manera se podrían optimizar la simulación de procesos biológicos complejos que presentan una alta complejidad computacional. [2]

Estudios previos han demostrado que la simulación numérica de redes metabólicas en procesadores secuenciales conlleva un costo computacional elevado debido a la necesidad de pasos de integración extremadamente pequeños (menos de 10^{-6} segundos). Un ejemplo paradigmático de esta complejidad es el modelo de “A Whole-Cell Computational Model of Mycoplasma genitalium” [3], el cual integra procesos metabólicos rápidos con dinámicas celulares globales, en modelos de este nivel de detalle simular apenas 10 segundos de actividad biológica puede requerir minutos de tiempo de CPU, una latencia que se vuelve prohibitiva cuando se requieren miles de simulaciones para análisis de sensibilidad o barrido de parámetros.

La electrónica digital puede simular representaciones de sistemas de muchos tipos, y en este tema específico se han logrado algunos avances en las últimas dos décadas, lo que ha permitido simular diferentes reacciones bioquímicas sin la necesidad de hacerlo experimentalmente. Al intentar modelar una célula con sus respectivos procesos se puede usar las ecuaciones diferenciales ordinarias (ODE) para encontrar bloques funcionales equivalentes electrónicos que se pueden representar mediante lógica digital; sin embargo, estos procesos celulares no son secuenciales y cada proceso se da

simultáneamente en paralelo, lo que aumenta la cantidad de recursos consumidos y tiempos de simulación. [4]

Por esta razón, actualmente existen estudios en proceso sobre el tema y las investigaciones han llevado a encontrar diferentes soluciones. Una de las propuestas para mejorar los tiempos de simulación es usar las FPGAs, cuyo funcionamiento permite emular varios procesos de manera paralela con un consumo mínimo de recursos de área y potencia, y con una velocidad de procesamiento mucho mayor que los programas de simulación convencionales.

Teniendo en cuenta las limitaciones que aún se tienen y la necesidad de avanzar en este tópico de investigación de punta, surge la pregunta ¿Cómo emular un proceso biológico metabólico como la glucólisis en una célula mediante la tecnología de los FPGAs, con el propósito de mejorar el tiempo de procesamiento computacional de la emulación?

JUSTIFICACIÓN

La biología de sistemas computacionales ha transformado nuestra comprensión de la vida celular, permitiendo modelar redes metabólicas *in silico* para aplicaciones que van desde la biomedicina hasta la biología sintética, sin embargo, a medida que estos modelos aumentan en complejidad y escala, la simulación numérica basada en software convencional se convierte en un cuello de botella. Los procesadores secuenciales (arquitectura Von Neumann) deben resolver miles de ecuaciones diferenciales paso a paso, lo que limita la capacidad de simular tejidos completos o realizar análisis de parámetros en tiempo real.

Este proyecto se justifica en la necesidad de explorar arquitecturas de computación alternativas que se alineen mejor con la naturaleza intrínsecamente paralela de los sistemas biológicos. Las FPGAs (*Field Programmable Gate Arrays*) ofrecen una solución única ejecutando todos los procesos simultáneamente, esta capacidad de paralelismo espacial representa una eficiencia superior.

La elección de la glucólisis como caso de estudio es estratégica, al ser la ruta metabólica universal y mejor caracterizada, y ofrece robustes para validar nuevas metodologías de ingeniería, sin embargo, su dinámica no lineal, especialmente la regulación alostérica de la enzima Fosfofructo-quinasa (PFK) presenta desafíos aritméticos que históricamente han dificultado su implementación en lógica digital.

Desarrollar un acelerador hardware capaz de resolver estas no linealidades con alta precisión (32 bits) y bajo consumo de recursos, aporta una herramienta para el estudio del metabolismo y establece una metodología de diseño escalable. Esto sienta las bases para una nueva generación de emuladores biológicos capaces de operar a velocidades de hardware evitando las limitaciones en software, facilitando la experimentación y el diseño de circuitos genéticos con una rapidez y determinismo inalcanzables por el software tradicional.

OBJETIVOS

OBJETIVO GENERAL

Implementar un modelo digital de ecuaciones diferenciales ordinarias como acelerador hardware mediante un FPGA para emular la ruta metabólica glucólisis en una célula.

OBJETIVOS ESPECÍFICOS

- Desarrollar el modelo matemático digital de la ruta metabólica glucólisis a partir de las ecuaciones diferenciales ordinarias equivalentes a las reacciones bioquímicas generadas por las enzimas en los sustratos de glucosa.

- Realizar el diseño hardware digital sobre una FPGA de las ODE discretizadas que modelan las reacciones bioquímicas de la glucólisis.

- Realizar un análisis comparativo de los resultados obtenidos mediante el FPGA con resultados experimentales y simulados por software.

MARCO TEORICO

1. Glucólisis

La glucólisis consiste en una sucesión de reacciones que extraen energía de la glucosa al dividirla en dos moléculas de tres carbonos conocidas como piruvato. Este proceso metabólico evolucionó hace mucho tiempo, y está presente en la mayoría de los organismos vivos en la actualidad. La estrategia bioquímica de la glucólisis puede entenderse como un proceso dividido en dos fases distintas, cada una con un objetivo claro [1].

1.1. Fase de requerimiento energético (Pasos 1-5)

El objetivo de esta fase preparatoria es activar la molécula de glucosa, que es relativamente estable, y convertirla en una forma que pueda ser fácilmente dividida en dos fragmentos de tres carbonos. Para lograrlo, la célula debe invertir energía en forma de ATP.

1.1.1. **Paso 1: fosforilación de la glucosa.** La glucosa entra a la célula y es inmediatamente fosforilada por la enzima hexoquinasa. Esta reacción consume una molécula de ATP y produce glucosa-6-fosfato (G6P). Este paso es crucial porque la adición del grupo fosfato cargado negativamente atrapa a la molécula dentro de la célula e incrementa su reactividad [5].

1.1.2. **Paso 2: isomerización.** La fosfoglucosa isomerasa cataliza la conversión reversible de G6P (una aldosa) a fructosa-6-fosfato (F6P), una cetosa. Este reordenamiento es necesario para la fosforilación en el carbono 1 en el siguiente paso [5].

1.1.3. **Paso 3: segunda fosforilación.** La fosfofructoquinasa-1 (PFK-1) cataliza el segundo paso de inversión de energía, transfiriendo un grupo fosfato de otra molécula de ATP a F6P para formar fructosa-1,6-bisfosfato (F1,6BP). Esta reacción es el principal punto de regulación de toda la vía glucolítica [5].

1.1.4. **Paso 4: escisión.** La enzima aldolasa divide la molécula de F1,6BP de seis carbonos en dos moléculas de tres carbonos: dihidroxiacetona fosfato (DHAP) y gliceraldehído-3-fosfato (G3P) [5].

1.1.5. **Paso 5: interconversión de triosas.** De los dos productos, solo el G3P puede continuar en la siguiente fase. La triosa fosfato isomerasa cataliza la conversión de DHAP en G3P, asegurando que ambas mitades de la molécula de glucosa original puedan ser procesadas [5].

1.2. Fase II: fase de ganancia de energía (pasos 6-10)

En esta fase, la energía química de los intermediarios de tres carbonos se cosecha en forma de ATP y NADH. Dado que todo ocurre por duplicado a partir de una molécula de glucosa, los rendimientos se duplican.

1.2.1. **Paso 6: oxidación y fosforilación.** La gliceraldehído-3-fosfato deshidrogenasa cataliza la única reacción de oxidación de la glucólisis. El G3P es oxidado, y los electrones de alta energía se transfieren a NAD+ para formar NADH. La energía liberada se utiliza para unir un fosfato inorgánico, creando 1,3-bisfosfoglicerato (1,3-BPG), una molécula con un enlace acil-fosfato de alta energía [5].

1.2.2. **Paso 7: primera producción de ATP.** La fosfoglicerato quinasa transfiere el grupo fosfato de alta energía del 1,3-BPG al ADP, produciendo la primera molécula de ATP. Este proceso se denomina fosforilación a nivel de sustrato [5].

1.2.3. **Paso 8: reordenamiento molecular.** La fosfoglicerato mutasa cataliza el desplazamiento del grupo fosfato del carbono 3 al carbono 2, formando 2-fosfoglicerato [5].

1.2.4. **Paso 9: deshidratación.** La enzima enolasa elimina una molécula de agua del 2-fosfoglicerato, creando un doble enlace y formando fosfoenolpiruvato (PEP). Esta reorganización interna de la energía crea un enlace enol-fosfato de muy alta energía [5].

1.2.5. **Paso 10: segunda producción de ATP.** El piruvato quinasa cataliza la transferencia del grupo fosfato de alta energía del PEP al ADP, generando la segunda molécula de ATP y el producto final, piruvato [5].

1.3. Balance neto de la glucólisis

Al concluir la glucólisis, se obtienen dos moléculas de ATP, dos de NADH y dos de piruvato. En presencia de oxígeno, el piruvato puede ser descompuesto u oxidado hasta dióxido de carbono a través de la respiración celular, permitiendo así la generación de más moléculas de ATP.

El balance global de la conversión de una molécula de glucosa a dos de piruvato es: Glucosa + 2 Pi + 2 ADP + 2 NAD+ → 2 Piruvato + 2 ATP + 2 NADH + 2 H+ + 2 H_2O [5],[6].

1.4. Regulación intrínseca de la glucólisis

Para mantener la homeostasis celular, el flujo de glucosa a través de la vía glucolítica debe ser ajustado continuamente para responder a las necesidades de ATP y precursores biosintéticos, esta regulación se ejerce sobre las tres enzimas que catalizan las reacciones irreversibles: hexoquinasa (paso 1), PFK-1 (paso 3) y piruvato quinasa (paso 10) [6].

La hexoquinasa es inhibida por su propio producto, la glucosa-6-fosfato. Esto previene un consumo excesivo de ATP cuando el flujo a través de la vía está disminuido [6].

La fosfofructoquinasa-1 (PFK-1) es el marcapasos principal de la glucólisis, su actividad es una sofisticada integración del estado energético de la célula. Es inhibida alostéricamente por niveles elevados de ATP y citrato. El ATP, además de ser un sustrato, actúa como una señal de alta energía, mientras que el citrato indica que el ciclo de Krebs está saturado. Por el contrario, es potentemente activada por AMP, una señal inequívoca de baja energía celular [6].

El piruvato quinasa controla el flujo de salida de la vía. También es inhibida por señales de alta energía como el ATP y activada por la fructosa-1,6-bisfosfato, el producto de la reacción de la PFK-1. Este último es un ejemplo de activación por precursores (*feed-forward*), que asegura que el final de la vía pueda procesar los intermediarios que se están generando al principio [6].

2. Modelos bioquímicos y su representación matemática

La era post-genómica ha dejado claro que una lista completa de los componentes moleculares de una célula no es suficiente para comprender su funcionamiento. La verdadera esencia de la vida reside en la intrincada red de interacciones dinámicas entre estos componentes. La biología de sistemas aborda este desafío, buscando entender cómo las funciones celulares emergen de la orquestación de miles de reacciones bioquímicas interconectadas [7]. Estas redes no son meros diagramas de flujo; están gobernadas por principios de control y regulación, como los bucles de retroalimentación, que permiten a las células mantener la homeostasis, responder a estímulos y ejecutar programas complejos como el ciclo celular. La dinámica de estas redes es a menudo no lineal y contraintuitiva, lo que hace que su análisis dependa de manera crucial de la formalización matemática [8].

Un modelo matemático, en este contexto, es una abstracción de un sistema biológico, expresado en el lenguaje de las matemáticas. Su propósito es encapsular nuestras hipótesis sobre cómo funciona el sistema en un conjunto de reglas cuantitativas. Este proceso de formalización nos obliga a ser explícitos sobre nuestras suposiciones y nos proporciona un "laboratorio virtual" donde podemos explorar la dinámica del sistema. Mediante la simulación por ordenador, podemos predecir cómo responderá la red a una

perturbación como un cambio en la disponibilidad de nutrientes o la inhibición de una enzima, generando así hipótesis precisas que pueden ser validadas o refutadas experimentalmente en el laboratorio. Por lo tanto, la modelización no es un fin en sí misma, sino una parte integral del ciclo iterativo de investigación en la biología moderna, un puente indispensable entre la teoría y la experimentación [9].

2.1. Modelos deterministas y estocásticos

La traducción de un proceso biológico a un modelo matemático requiere una decisión fundamental sobre cómo tratar la naturaleza de las poblaciones moleculares. A nivel microscópico, las reacciones son eventos discretos y aleatorios, sin embargo, a nivel macroscópico, el comportamiento de miles de millones de moléculas puede parecer suave y predecible. Esta dualidad da lugar a dos grandes familias de modelos: los estocásticos y los deterministas [10].

El enfoque estocástico se fundamenta en la realidad de que las reacciones químicas son eventos probabilísticos. Este paradigma es indispensable cuando el número de moléculas de una o más especies es muy bajo, como suele ocurrir en los circuitos de regulación génica. En estos sistemas de "números pequeños", las fluctuaciones aleatorias en el momento y la secuencia de las reacciones, un fenómeno conocido como "ruido" estocástico, no se promedian y pueden tener consecuencias funcionales profundas, llevando a una variabilidad significativa entre células idénticas. Los modelos estocásticos, por lo tanto, no siguen la evolución de concentraciones, sino el número discreto de moléculas, simulando cada evento de reacción individual como un suceso aleatorio cuya probabilidad depende del estado actual del sistema. El resultado de una simulación estocástica no es una única trayectoria, sino una distribución de posibles trayectorias, reflejando la inherente imprevisibilidad del sistema a esa escala [11].

Por otro lado, el enfoque determinista se aplica a sistemas en los que las poblaciones moleculares son grandes. Las rutas metabólicas centrales, como la glucólisis, son el ejemplo por excelencia de este tipo de sistemas. Las concentraciones de sus sustratos, productos y enzimas son lo suficientemente altas como para que el comportamiento aleatorio de una molécula individual sea irrelevante para la dinámica global. De acuerdo con la ley de los grandes números, las fluctuaciones estocásticas se promedian, y el comportamiento del sistema se vuelve efectivamente predecible. En este régimen, es una simplificación válida y poderosa tratar las concentraciones de las especies químicas como variables continuas que cambian de manera suave en el tiempo. Este enfoque ignora el ruido molecular, pero captura con gran precisión el comportamiento promedio y macroscópico del sistema, que es precisamente el nivel de descripción relevante para entender el flujo de materia y energía a través de una vía metabólica [9], [10].

2.2. Formalismo para sistemas deterministas

Dado que el comportamiento de sistemas con grandes poblaciones moleculares puede ser considerado determinista, necesitamos un formalismo matemático que describa la evolución continua y predecible de sus concentraciones en el tiempo. Las Ecuaciones Diferenciales Ordinarias (EDOs) son la herramienta matemática por excelencia para esta tarea. Este enfoque se puede entender como un sistema de contabilidad dinámico. Para cada especie molecular del modelo, se establece una EDO que define su tasa de cambio. Esta tasa es simplemente el balance neto de todos los procesos que afectan su concentración: la suma de las velocidades de todas las reacciones que la producen menos la suma de las velocidades de todas las reacciones que la consumen [10].

El poder de este formalismo reside en que el conjunto de EDOs acopladas captura la interdependencia de toda la red. La concentración de un metabolito, que es la variable en una ecuación, aparece en los términos de velocidad de otras ecuaciones, creando así un sistema dinámico interconectado que refleja la estructura de la ruta bioquímica. Para que este sistema de ecuaciones tenga significado biológico, debemos definir las velocidades de cada reacción. Estas expresiones matemáticas, conocidas como leyes de velocidad, son las que dictan la dinámica del modelo. Para las reacciones catalizadas por enzimas, que son la norma en el metabolismo, estas leyes deben reflejar un principio biológico fundamental: la saturación. A diferencia de una reacción simple, donde la velocidad podría aumentar indefinidamente con más sustrato, una enzima tiene una capacidad finita de procesamiento. A medida que la concentración de sustrato aumenta, la velocidad de la reacción se aproxima asintóticamente a una velocidad máxima (Vmax). La cinética de Michaelis-Menten es la descripción matemática clásica de este comportamiento no lineal, y es un componente esencial para construir modelos metabólicos realistas [12].

En conclusión, el formalismo de EDOs es la elección lógica y rigurosa para modelar la glucólisis. La naturaleza de esta vía como un proceso de alto flujo asegura que las concentraciones de sus intermediarios son suficientemente grandes para justificar un tratamiento determinista y continuo. Las EDOs proporcionan el marco perfecto para describir la evolución temporal de estas concentraciones, y su acoplamiento a través de leyes de velocidad basadas en la cinética enzimática permite capturar la compleja red de regulación que gobierna el flujo a través de esta ruta metabólica central. Este enfoque nos permite, por tanto, construir un modelo cuantitativo y predictivo, capaz de simular la dinámica de la glucólisis bajo diversas condiciones fisiológicas [8], [12].

3. Ecuaciones diferenciales ordinarias (EDOs)

Las ecuaciones diferenciales ordinarias (EDOs) constituyen una rama fundamental del análisis matemático utilizada para describir sistemas en los que una magnitud varía en función de una única variable independiente. En términos generales, una EDO se define

como una relación entre una función desconocida y una o más de sus derivadas con respecto a dicha variable. Matemáticamente, puede expresarse como

$$F(x, y, y', y'', \dots, y^{(n)}) = 0, \tag{3.1}$$

donde $y^{(n)}$ representa la n-ésima derivada de y con respecto a x. El propósito de una ecuación diferencial es capturar la dinámica del cambio: cómo un sistema evoluciona en el tiempo o en el espacio a partir de las condiciones iniciales establecidas. Su estudio permite modelar fenómenos tan diversos como la oscilación de un circuito eléctrico, la cinética de una reacción química o la evolución de una concentración biológica a lo largo del tiempo [13].

3.1. Clasificación de las EDOs

Las EDOs pueden clasificarse atendiendo a diferentes criterios estructurales. El orden de la ecuación está determinado por la derivada de mayor grado que aparece en ella, lo que condiciona la complejidad del análisis. Asimismo, se distinguen las ecuaciones lineales donde la función incógnita y sus derivadas aparecen solo a la primera potencia y no multiplicadas entre sí de las no lineales, cuyo tratamiento suele ser más complejo y con soluciones menos predecibles [13].

También se establece una distinción entre ecuaciones autónomas y no autónomas. Las primeras no dependen explícitamente de la variable independiente, lo que significa que el comportamiento del sistema depende únicamente de su estado actual. Esta característica es particularmente relevante en sistemas biológicos autorregulados, donde la evolución del sistema depende del equilibrio interno de sus componentes [13].

3.2. Existencia y unicidad de soluciones

Uno de los fundamentos teóricos más importantes en el estudio de las EDOs es el teorema de existencia y unicidad, también conocido como teorema de Picard Lindelöf. Este teorema garantiza que, si la función $f(x, y)$que define la ecuación diferencial es continua y cumple una condición de Lipschitz en la variable dependiente, entonces el problema de valor inicial

$$\frac{dy}{dx} = f(x, y), y(x_0) = y_0, \tag{3.2}$$

Posee una única solución local en el entorno del punto (x_0, y_0) [13].

Dicho principio proporciona una base de coherencia matemática, asegurando que un sistema físico o biológico correctamente planteado no genera soluciones contradictorias ante las mismas condiciones iniciales. En términos aplicados, esto

garantiza la reproducibilidad de modelos que describen, por ejemplo, la evolución temporal de concentraciones químicas o biológicas.

3.3. Métodos de solución analítica

Las EDOs más simples pueden resolverse de forma analítica, es decir, obteniendo expresiones exactas para la función desconocida. Entre los métodos más comunes se encuentran la separación de variables, la transformación de ecuaciones homogéneas, los factores integrantes y la transformada de Laplace. Estos procedimientos son de gran utilidad para el análisis de ecuaciones de primer y segundo orden que modelan fenómenos básicos, como la descarga de un condensador o la desintegración radiactiva [13].

Sin embargo, en la práctica, muchas ecuaciones que describen procesos reales especialmente las de naturaleza no lineal no admiten soluciones cerradas, lo cual motiva el uso de métodos aproximados o numéricos.

3.4. Métodos numéricos para EDOs

Los métodos numéricos permiten aproximar la solución de una EDO cuando no es posible resolverla de manera exacta. El principio fundamental consiste en sustituir las derivadas por diferencias finitas que pueden calcularse de manera iterativa. Entre los métodos más empleados se encuentran el método de Euler, el método de Runge Kutta de cuarto orden y los métodos predictor–corrector, que logran un equilibrio entre precisión y costo computacional [14].

La selección del método depende del tipo de ecuación, el nivel de precisión deseado y la estabilidad requerida. En el contexto de la simulación digital, estos métodos resultan esenciales para trasladar los modelos continuos a entornos discretos, como microcontroladores o plataformas FPGA, permitiendo representar dinámicas biológicas de forma computacionalmente eficiente.

3.5. Estabilidad y rigidez de las soluciones

El concepto de estabilidad describe cómo responde un sistema modelado por una EDO ante pequeñas perturbaciones en sus condiciones iniciales. Según Lyapunov, un punto de equilibrio y_e es estable si las trayectorias que parten de puntos cercanos permanecen próximas a y_e a lo largo del tiempo. Este principio permite determinar si un sistema tenderá al equilibrio o si, por el contrario, evolucionará hacia estados divergentes [15].

Por otro lado, la rigidez se refiere a la presencia de diferentes escalas temporales dentro de una misma ecuación, fenómeno que genera dificultades numéricas al resolver el sistema. Los métodos implícitos y de integración adaptativa son los más adecuados para tratar este tipo de problemas.

3.6. Representación matricial y sistemas de EDOs

Un sistema de ecuaciones diferenciales acopladas puede expresarse de manera compacta mediante la notación matricial:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t), \tag{3.3}$$

donde $\mathbf{x}$ es un vector que contiene las variables dependientes. Esta formulación facilita el análisis de sistemas multidimensionales y permite aplicar técnicas de álgebra lineal, como la obtención de autovalores y autovectores, para estudiar la estabilidad del sistema o la existencia de oscilaciones [16].

En el ámbito biológico, esta representación es indispensable para modelar rutas metabólicas, donde las concentraciones de los metabolitos dependen unas de otras de forma acoplada y simultánea.

3.7. Modelos dinámicos y representación computacional

Las EDOs constituyen la base formal de los modelos dinámicos deterministas, en los cuales las derivadas representan tasas de cambio de variables como concentraciones, temperaturas o potenciales eléctricos. En biología de sistemas, estos modelos permiten analizar procesos metabólicos o genéticos describiendo las interacciones entre componentes de una red [17].

Para su simulación, las ecuaciones continuas deben discretizarse en el dominio temporal mediante métodos numéricos, posibilitando su implementación en entornos digitales. Esta transformación es esencial para trasladar los modelos matemáticos a sistemas programables como las FPGAs, donde las operaciones diferenciales se representan mediante integradores digitales o sumadores acumulativos.

Así, las EDOs no solo constituyen una herramienta teórica, sino también una base práctica para la modelación computacional de procesos biológicos complejos, como la glucólisis, mediante hardware especializado.

3.8. Discretización de las ecuaciones diferenciales

La discretización de las ecuaciones diferenciales consiste en el proceso mediante el cual un modelo matemático continuo se transforma en un conjunto de expresiones algebraicas finitas, aptas para su tratamiento computacional. Este procedimiento reemplaza las derivadas por aproximaciones basadas en diferencias finitas, permitiendo representar la evolución temporal o espacial de una variable en intervalos discretos de tiempo o posición. En otras palabras, la discretización convierte la ecuación diferencial, dependiente de una variable continua t, en una sucesión de

operaciones que dependen de pasos definidos $t_0, t_1, t_2, \dots, t_n$, de forma que el comportamiento del sistema pueda simularse digitalmente [17].

3.8.1. Fundamentos conceptuales

En una EDO del tipo

$$\frac{dy}{dt} = f(t, y), y(t_0) = y_0, \tag{3.4}$$

la discretización implica dividir el dominio de $t$en pequeños intervalos de tamaño h, conocidos como pasos de integración. Así, el valor de la función en un punto t_{n+1} se obtiene a partir del valor anterior y_n y de la pendiente aproximada en ese intervalo. El método más sencillo de discretización es el método de Euler hacia adelante, definido por

$$y_{n+1} = y_n + h\, f(t_n, y_n), \tag{3.5}$$

donde h determina la resolución temporal y, por tanto, la precisión del cálculo [6].

El tamaño del paso h es un parámetro crítico: valores muy grandes reducen el tiempo de cómputo, pero disminuyen la exactitud, mientras que pasos pequeños aumentan la precisión a costa de un mayor costo computacional.

3.8.2. Métodos de discretización implícitos y explícitos

Existen dos categorías principales de métodos de discretización: explícitos e implícitos.

En los métodos explícitos, como Euler o Runge Kutta, el nuevo valor y_{n+1} se calcula directamente a partir de los valores conocidos en el instante anterior. Estos métodos son simples y computacionalmente eficientes, aunque pueden presentar problemas de estabilidad para ecuaciones rígidas. Por otro lado, los métodos implícitos, como el método de Euler hacia atrás o los métodos de diferencias trapezoidales, requieren resolver una ecuación en cada paso de tiempo, ya que y_{n+1} aparece en ambos lados de la relación. Aunque más costosos computacionalmente, ofrecen una estabilidad superior y son preferibles en la simulación de sistemas con múltiples escalas temporales [17].

En términos generales, la elección del método depende de la naturaleza del sistema a modelar: los métodos explícitos son útiles para simulaciones rápidas y sistemas no rígidos, mientras que los implícitos son más adecuados para ecuaciones que describen procesos con variaciones bruscas o componentes de respuesta lenta.

3.8.3. Error de truncamiento y estabilidad numérica

Todo proceso de discretización introduce un error de truncamiento, que surge al reemplazar derivadas continuas por aproximaciones discretas. Este error puede expresarse en términos del paso hy depende del orden del método utilizado. Por ejemplo, el método de Euler presenta un error de orden $O(h)$, mientras que los métodos Runge Kutta alcanzan órdenes superiores, como $O(h^4)$ [17].

La estabilidad numérica evalúa la capacidad del método para mantener un comportamiento físico correcto cuando se acumulan errores de redondeo y truncamiento. Si un método es inestable, pequeñas perturbaciones pueden amplificarse a lo largo de las iteraciones, generando resultados divergentes. Por ello, la estabilidad constituye un criterio esencial en el diseño de simulaciones digitales precisas.

3.8.4. Discretización y representación digital

En el ámbito computacional, la discretización no solo implica dividir el dominio de integración, sino también representar numéricamente las variables mediante formatos finitos, como punto flotante o punto fijo. Cuando las ecuaciones diferenciales se implementan en hardware, como en plataformas FPGA, el uso de representaciones de punto fijo resulta más eficiente, ya que permite un control más directo de los recursos lógicos y de la precisión aritmética [18].

En este contexto, cada operación diferencial se traduce en una secuencia de operaciones aritméticas digitales (sumas, multiplicaciones e integraciones acumulativas), de modo que el sistema discreto reproduce el comportamiento del modelo continuo original.

La discretización constituye, por tanto, un puente entre la teoría matemática de las ecuaciones diferenciales y su ejecución en dispositivos digitales. Su adecuada formulación garantiza que el modelo preserve la dinámica esencial del sistema original sin comprometer la estabilidad ni la fidelidad de los resultados.

4. Modelos de la glucólisis y su representación matemática existentes

La glucólisis ha sido objeto de numerosos estudios teóricos y computacionales que buscan comprender las dinámicas metabólicas a través de modelos matemáticos. Estos modelos, generalmente basados en ecuaciones diferenciales ordinarias (EDOs), describen la variación temporal de las concentraciones de metabolitos y enzimas involucrados en la ruta. A lo largo de las últimas décadas, diferentes enfoques han permitido representar este proceso con distintos niveles de complejidad, desde formulaciones simplificadas hasta sistemas cinéticos altamente detallados [19].

4.1. Modelos cinéticos clásicos de la glucólisis

Los primeros modelos matemáticos de la glucólisis se centraron en describir la cinética enzimática global del proceso. Uno de los trabajos pioneros fue el de Higgins (1964), quien propuso un modelo de retroalimentación negativa para explicar las oscilaciones en las concentraciones de metabolitos en levaduras. Este enfoque consideró únicamente unas pocas reacciones clave, capturando el comportamiento oscilatorio del sistema mediante ecuaciones diferenciales no lineales [20].

Posteriormente, Sel'kov [21] desarrolló un modelo simplificado de la fase energética de la glucólisis, compuesto por dos ecuaciones que representan la concentración de adenosina difosfato (ADP) y fosfato inorgánico. Su formulación permitió estudiar las condiciones bajo las cuales emergen oscilaciones bioquímicas espontáneas, lo que marcó un hito en el modelado de redes metabólicas dinámicas.

Estos modelos sentaron las bases para entender la autoorganización y la estabilidad en sistemas metabólicos, mostrando que la glucólisis no es un proceso estrictamente estacionario, sino un sistema dinámico susceptible a variaciones periódicas y acoplamientos no lineales.

4.2. Modelos detallados basados en cinética enzimática

La cinética de Michaelis Menten es el fundamento matemático más utilizado en el modelado de procesos enzimáticos. Bajo este enfoque, cada reacción en la glucólisis se describe por una ecuación del tipo

$$v = \frac{V_{\max}[S]}{K_m + [S]}, \tag{3.6}$$

donde $V_{\max}$ es la velocidad máxima de la enzima, K_m es la constante de Michaelis y $[S]$ la concentración del sustrato [22].

Con el avance de la biología molecular y la disponibilidad de parámetros experimentales más precisos, surgieron modelos detallados basados en la cinética de cada reacción enzimática. Entre los más reconocidos se encuentra el de Teusink et al. [22], que desarrolló un modelo exhaustivo de la glucólisis en *Saccharomyces cerevisiae* utilizando ecuaciones diferenciales derivadas de la cinética de Michaelis–Menten para cada enzima, este modelo incluye más de 20 metabolitos y 14 reacciones, con parámetros cinéticos determinados experimentalmente, permitiendo reproducir tanto el estado estacionario como las oscilaciones inducidas por cambios en las condiciones externas [22].

El modelo de Teusink se considera una referencia fundamental en biología de sistemas, ya que integra datos experimentales cuantitativos y ofrece una descripción precisa del flujo metabólico y la regulación enzimática.

4.3. Modelos oscilatorios y de retroalimentación (Goldbeter Lefever y variantes)

En la década de 1970, Goldbeter y Lefever [27] desarrollaron un modelo de la glucólisis que incorporaba retroalimentación positiva y negativa en la regulación de la enzima fosfofructoquinasa (PFK). Este modelo, formulado mediante ecuaciones diferenciales no lineales, mostró la capacidad del sistema glucolítico para generar oscilaciones regulares en ausencia de estímulos externos.

El modelo de Goldbeter–Lefever representa uno de los avances más influyentes en la teoría de las oscilaciones bioquímicas, al demostrar que la complejidad temporal del metabolismo puede surgir de la estructura de la red enzimática y no de factores externos.

Posteriormente, extensiones de este modelo incorporaron acoplamientos entre cascadas enzimáticas**,** como los propuestos por Wolf et al. [26], quienes desarrollaron un modelo completo de la glucólisis oscilatoria en levaduras con más de 20 especies y 16 reacciones.

Estos modelos reflejan el carácter autoorganizado y emergente del metabolismo celular, y su formulación en EDOs permite analizar estabilidad, bifurcaciones y patrones de oscilación mediante herramientas de dinámica no lineal.

4.4. Modelos modulares y reducidos

Para facilitar la simulación y el análisis computacional, algunos autores han desarrollado versiones reducidas o modulares de los modelos cinéticos completos.

Mulquiney y Kuchel [24] propusieron un modelo de la glucólisis en eritrocitos humanos basado en ecuaciones diferenciales detalladas, pero estructurado en módulos independientes que representan subprocesos metabólicos como la fase preparatoria, la fase de generación de energía y la regulación por 2,3-bisfosfoglicerato.

Este enfoque modular facilita la implementación digital y la integración con otros subsistemas bioquímicos, al permitir el análisis individual de cada componente sin perder la coherencia global del sistema. Además, dichos modelos reducidos son útiles para la implementación en hardware o en plataformas de simulación en tiempo real, donde los recursos computacionales son limitados.

4.5. Modelos computacionales y herramientas de simulación

El desarrollo de software especializado ha permitido la simulación de modelos de glucólisis con un alto nivel de detalle. Plataformas como COPASI, CellDesigner y MATLAB SimBiology han facilitado la representación y análisis de ecuaciones diferenciales no lineales que describen redes metabólicas. Estas herramientas

permiten realizar análisis de sensibilidad, evaluación de estabilidad y ajuste de parámetros mediante datos experimentales [25].

Además, los modelos actuales incorporan cada vez más la posibilidad de exportar representaciones discretizadas, lo que abre la puerta a su implementación en entornos digitales o dispositivos embebidos. En este sentido, la simulación numérica sirve como etapa previa a la representación en hardware, como FPGAs, que requieren modelos discretos y optimizados para procesamiento en paralelo.

4.6. Perspectiva actual y aplicaciones en modelado digital

Los modelos actuales de glucólisis combinan enfoques experimentales, cinéticos y computacionales, orientándose hacia la representación digital y la simulación en hardware reconfigurable. Trabajos recientes buscan reducir la complejidad del modelo manteniendo su fidelidad biológica, de modo que pueda implementarse en arquitecturas de hardware paralelas. Este enfoque permite estudiar la dinámica metabólica en tiempo real y evaluar el impacto de perturbaciones en condiciones fisiológicas simuladas [26].

Así, los modelos de glucólisis no solo constituyen una herramienta teórica, sino también un puente entre la biología de sistemas y la ingeniería digital, contribuyendo al desarrollo de sistemas biomiméticos y plataformas de simulación bioelectrónica.

5. Arquitectura FPGA: bloques lógicos, LUTs, FFs y lenguajes HDL

Las Field Programmable Gate Arrays (FPGAs) son dispositivos digitales reconfigurables que permiten la implementación de circuitos lógicos personalizados a través de una estructura modular programable. Estas arquitecturas combinan flexibilidad, paralelismo y alto rendimiento, características que las hacen especialmente adecuadas para aplicaciones de simulación numérica y procesamiento en tiempo real, como la emulación digital de modelos matemáticos derivados de ecuaciones diferenciales ordinarias (EDOs). A diferencia de los microcontroladores o procesadores convencionales, las FPGAs no ejecutan instrucciones secuenciales, sino que realizan operaciones simultáneamente en múltiples bloques lógicos interconectados, permitiendo una ejecución verdaderamente paralela [28].

5.1. Estructura general de una FPGA

Una FPGA está organizada como una matriz bidimensional de bloques lógicos configurables (CLBs), conectados entre sí mediante una red de ruteo programable. Cada CLB contiene componentes básicos como tablas de búsqueda (LUTs), flip-flops (FFs) y multiplexores, que permiten implementar tanto operaciones combinacionales como secuenciales. Además, las FPGAs modernas incluyen bloques de memoria

embebida (BRAMs), multiplicadores digitales (DSPs) y módulos de entrada/salida (I/O blocks) que amplían su funcionalidad.

Esta arquitectura modular otorga a las FPGAs la capacidad de ejecutar tareas en paralelo, optimizando el rendimiento en cálculos matemáticos intensivos, como la integración numérica de ecuaciones diferenciales o el procesamiento de señales digitales [28].

5.2. Bloques lógicos configurables (CLBs)

Los CLBs son los elementos fundamentales de procesamiento dentro de la FPGA. Cada bloque puede configurarse para realizar una función lógica específica, desde simples operaciones booleanas hasta módulos aritméticos más complejos [28].

Su estructura interna combina LUTs que actúan como pequeñas memorias que almacenan los valores de verdad de una función booleana, FFs que permiten sincronizar datos mediante señales de reloj, y multiplexores que seleccionan y conmutan entre múltiples señales de entrada para dirigirlas a una única salida, toda esta estructura puede conectarse de distintas maneras según la función deseada. Gracias a esta flexibilidad, los CLBs permiten construir estructuras digitales como sumadores, contadores, registros o máquinas de estados finitos, que son esenciales para representar modelos matemáticos discretizados y sistemas dinámicos [28].

En el contexto de una implementación de ecuaciones diferenciales, cada CLB puede representar un componente del modelo, por ejemplo, un integrador, una función de activación o una suma ponderada, posibilitando la ejecución de múltiples ecuaciones de manera simultánea.

5.3. Tablas de búsqueda (LUTs)

Las tablas de búsqueda (Lookup Tables, LUTs) son los elementos básicos para implementar funciones lógicas combinacionales dentro de una FPGA. Una LUT de n entradas puede almacenar 2^n valores de salida, permitiendo así realizar cualquier operación lógica de n variables. Por ejemplo, una LUT de 4 entradas puede implementar 16 combinaciones posibles de entrada-salida, configurables según la lógica requerida [28].

5.4. Aplicaciones de las FPGAs en modelado matemático

Las FPGAs se han convertido en herramientas poderosas para la implementación de modelos matemáticos en tiempo real, especialmente aquellos basados en ecuaciones diferenciales discretizadas. Cada operación del modelo (suma, multiplicación, integración) puede representarse como un bloque lógico independiente que trabaja en paralelo con otros, logrando un desempeño superior al de las plataformas

secuenciales. En biología de sistemas, esta capacidad se aprovecha para emular modelos de rutas metabólicas, como la glucólisis, en tiempo real, lo que permite analizar su comportamiento dinámico bajo distintas condiciones [30].

6. Implementación de modelos matemáticos en hardware

La implementación de modelos matemáticos en hardware consiste en traducir ecuaciones o sistemas de ecuaciones, generalmente diferenciales o algebraicas, a una forma computacional que pueda ejecutarse directamente en un dispositivo físico, como una FPGA, un microcontrolador o un procesador digital de señales (DSP). Este proceso implica representar los cálculos continuos mediante operaciones discretas y finitas, adaptadas a los recursos del hardware. La finalidad es obtener un comportamiento equivalente al modelo original, pero optimizado en velocidad, paralelismo y eficiencia energética [31].

6.1. Representación discreta de modelos continuos

El primer paso para implementar un modelo matemático en hardware es convertir sus ecuaciones continuas en una forma discreta que pueda procesarse digitalmente. Para ello, las derivadas e integrales se reemplazan por aproximaciones por diferencias finitas, obteniendo relaciones recursivas del tipo

$$y_{n+1} = y_n + h \cdot f(y_n, t_n), \tag{6.1}$$

donde $h$es el paso de integración y $f$representa la función de evolución del sistema. Esta representación es compatible con arquitecturas digitales, donde cada ciclo de reloj puede representar un incremento temporal. El modelo resultante puede interpretarse como un algoritmo iterativo, en el cual las variables se actualizan a lo largo del tiempo mediante operaciones de suma y multiplicación [31].

6.2. Implementación aritmética digital

Una vez discretizado el modelo, las operaciones matemáticas deben mapearse a unidades aritméticas digitales. En el hardware, los números se representan con precisión finita, comúnmente en formato de punto fijo o punto flotante. El formato de punto fijo es preferido en implementaciones sobre FPGA por su menor consumo de recursos y su ejecución más rápida, aunque requiere un control cuidadoso del rango y la precisión [31].

Las operaciones básicas como la suma, resta, multiplicación e integración acumulativa se implementan mediante sumadores, multiplicadores y registros, componentes que pueden organizarse para ejecutar el modelo en paralelo. Cada ecuación del sistema se convierte en una red combinacional o secuencial que

actualiza las variables del modelo a cada ciclo de reloj, reproduciendo el comportamiento temporal del sistema original.

6.3. Flujo de diseño digital

El proceso de implementación de un modelo matemático en hardware sigue un flujo de diseño estructurado, que comprende las siguientes etapas:

- **Descripción del modelo** en un lenguaje de alto nivel o simbólico (por ejemplo, MATLAB, Python, o directamente ecuaciones).
- **Traducción a operaciones discretas**, adaptadas a formatos numéricos y a pasos de integración específicos.
- **Codificación en HDL (VHDL o Verilog)**, donde se definen las operaciones aritméticas y los flujos de datos entre registros.
- **Síntesis y mapeo** del diseño al dispositivo FPGA, en el cual las operaciones se asignan a LUTs, flip-flops y bloques DSP.
- **Verificación y validación**, comparando la salida del hardware con la simulación numérica del modelo original.

Este flujo garantiza que el comportamiento del hardware sea funcionalmente equivalente al modelo matemático inicial, respetando los parámetros temporales y las relaciones entre variables [31].

6.4. Arquitecturas paralelas y optimización

Las arquitecturas digitales, especialmente las FPGAs, permiten explotar el paralelismo inherente de los modelos matemáticos. Cada ecuación o conjunto de ecuaciones puede asignarse a un bloque lógico independiente, posibilitando la ejecución simultánea de cálculos. Este enfoque contrasta con las plataformas secuenciales, donde las operaciones se ejecutan una tras otra, limitando el rendimiento global.

En hardware, el pipelining (segmentación del flujo de datos) y la paralelización de operaciones son técnicas comunes para optimizar la velocidad y el uso de recursos. De esta forma, la emulación de modelos como la glucólisis puede alcanzar velocidades de simulación en tiempo real o incluso superiores al tiempo biológico real [31].

6.5. Implementaciones en FPGA de modelos dinámicos

Las FPGAs se han convertido en una plataforma fundamental para la implementación de modelos dinámicos complejos, especialmente aquellos formulados mediante ecuaciones diferenciales ordinarias (EDOs). Su arquitectura paralela y su capacidad de procesamiento determinista las hacen ideales para reproducir el comportamiento

de sistemas no lineales en tiempo real. Cada ecuación del modelo puede representarse mediante operaciones aritméticas básicas (sumas, multiplicaciones o integraciones discretas) implementadas en bloques lógicos configurables (CLBs), lo que permite una ejecución simultánea y eficiente del sistema [31].

Uno de los trabajos pioneros en este campo fue el de Ros et al. [30], quienes desarrollaron una implementación en FPGA de redes neuronales biológicas. En su investigación, se tradujeron ecuaciones diferenciales no lineales en circuitos digitales utilizando discretización temporal y estructuras aritméticas paralelas, logrando reproducir el comportamiento de neuronas biológicas con alta precisión. Este enfoque demostró que las FPGAs pueden utilizarse para emular sistemas biológicos complejos, ofreciendo una alternativa hardware a las simulaciones tradicionales basadas en software.

Más recientemente, Chen et al. [32] presentaron la implementación digital del modelo neuronal de cuarto orden (Quartic Neuron Model, QNM), un sistema matemático de alta complejidad que describe el potencial de membrana neuronal mediante ecuaciones diferenciales no lineales. Su trabajo introdujo modificaciones matemáticas de bajo costo computacional para optimizar el uso de LUTs, flip-flops y otros recursos de hardware, logrando una ejecución en tiempo real con alta eficiencia energética. El estudio evidenció que las estrategias de discretización y optimización aplicadas en modelos neuronales pueden extrapolarse a otros sistemas biológicos, como la glucólisis, donde también se busca representar dinámicas bioquímicas continuas mediante hardware reconfigurable.

6.6. Antecedentes de emulación de glucólisis en FPGA

Aunque el modelado matemático de la glucólisis es extenso, su implementación física en hardware es un campo emergente. Un trabajo referente es el de Duarte et al. (2011), titulado “Hardware Simulation of Yeast Glycolytic Oscillations” [33], cuyo enfoque validó la capacidad de la FPGA para generar oscilaciones biológicas estables empleando aritmética de punto flotante. A diferencia de dicha propuesta, el modelo desarrollado en este trabajo implementa aritmética de punto fijo, esta decisión de diseño es crucial, ya que permite reducir significativamente la latencia de cálculo y el consumo de recursos lógicos (área y potencia) en el FPGA, logrando una eficiencia computacional superior sin comprometer la precisión numérica necesaria para replicar la dinámica del sistema biológico.

Por otra parte, está el trabajo de Osana et al. [2], donde desarrollaron la plataforma ReCSiP, un simulador bioquímico reconfigurable sobre FPGA, fue uno de los primeros trabajos en utilizar la glucólisis como banco de pruebas (*benchmark*). Su enfoque era genérico, permitiendo cargar diferentes rutas metabólicas, pero utilizaba aritmética de punto flotante de 32 bits, lo que limitaba el número de reacciones paralelas debido al alto consumo de recursos lógicos.

7. Síntesis integradora del marco teórico

La metodología propuesta integra la formalización matemática de la bioquímica (mediante leyes de velocidad de Michaelis Menten) con la arquitectura digital, mientras la bioquímica define la topología de la red (quién reacciona con quién), la ingeniería electrónica provee el sustrato físico para su ejecución paralela. El desafío central abordado en los capítulos siguientes es la traducción del sistema de ecuaciones diferenciales continuas (Ec. 7.1) a una estructura discreta de punto fijo, donde las concentraciones químicas se mapean a registros digitales y las reacciones enzimáticas a bloques DSP pipelined.

$$\frac{dS_i}{dt} = \sum_j v_{ij} v_j(S, k), \qquad (7.1)$$

Esta formulación convierte el sistema bioquímico en un conjunto de ecuaciones interdependientes que pueden resolverse analíticamente o mediante métodos numéricos, sentando las bases para su representación computacional o electrónica [35].

7.1. Perspectiva interdisciplinaria y aplicaciones

La integración entre la bioquímica, la modelación matemática y la ingeniería electrónica han abierto nuevas perspectivas en el análisis de sistemas biológicos. Los modelos matemáticos implementados en hardware permiten realizar simulaciones en tiempo real con alta precisión, facilitando la exploración de comportamientos emergentes, la validación experimental y el diseño de sistemas bioinspirados. En este contexto, el modelado digital de la glucólisis se presenta como un ejemplo paradigmático: un proceso bioquímico bien caracterizado que puede ser traducido a ecuaciones diferenciales y finalmente implementado en un circuito electrónico reconfigurable.

Este enfoque integrador no solo fortalece la comprensión teórica del metabolismo celular, sino que también impulsa el desarrollo de nuevas herramientas en bioingeniería, control biológico y biología sintética, donde los límites entre lo biológico y lo electrónico se vuelven cada vez más difusos.

METODOLOGIA

8. Diseño del modelo matemático

El presente modelo tiene como finalidad representar de manera cuantitativa el comportamiento dinámico de la vía glucolítica mediante un sistema de EDOs, con el propósito de generar una base matemática que permita posteriormente su discretización e implementación digital en una arquitectura FPGA, posibilitando la emulación en tiempo real del proceso metabólico.

8.1. Objetivo del modelo

El objetivo de este modelo matemático es describir las variaciones temporales de las concentraciones de los metabolitos principales involucrados en la glucólisis, considerando las relaciones estequiométricas y las velocidades de reacción asociadas a cada enzima. El modelo se limita a la fase preparatoria y a la fase de generación de energía del proceso, permitiendo capturar el comportamiento general del sistema sin incluir la regulación hormonal ni acoplamientos con otras vías metabólicas.

8.2. Reducción del modelo

El modelo bioquímico de la glucólisis completa involucra diez reacciones enzimáticas principales que transforman una molécula de glucosa en dos moléculas de piruvato, con la producción neta de ATP y NADH. Sin embargo, la inclusión de todas las etapas intermedias implica un número elevado de variables de estado y parámetros cinéticos, muchos de los cuales no están disponibles con precisión o resultan irrelevantes para el objetivo de la implementación digital, por tal motivo, se adopta una estrategia de reducción del modelo, orientada a preservar el comportamiento dinámico global del sistema, minimizando su complejidad matemática y computacional.

Se han considerado explícitamente las reacciones desde la fosforilación inicial de la glucosa hasta la formación del gliceraldehído-3-fosfato (GAP), incluyendo los metabolitos GLC, G6P, F6P, FBP, GAP, PYR, ATP y ADP, las reacciones subsiguientes, que comprenden las transformaciones de GAP hacia piruvato, se agrupan en un solo proceso global representado por una tasa de conversión modelada mediante leyes cinéticas de Michaelis Menten. Esta simplificación se basa en la observación de que los pasos intermedios catalizados por las enzimas gliceraldehído-3-fosfato deshidrogenasa, fosfoglicerato quinasa, fosfoglicerato mutasa, enolasa y piruvato quinasa presentan cinéticas rápidas, acoplamiento estequiométrico y una relación casi lineal entre GAP y PYR bajo condiciones fisiológicas estables.

La reducción adoptada es coherente con los enfoques descritos por Teusink et al. [22] y Chassagnole et al. [34] quienes demostraron que la segunda mitad de la glucólisis puede representarse como un bloque cinético equivalente sin afectar las oscilaciones metabólicas ni las relaciones de control sobre la producción de ATP. Dicho agrupamiento permite eliminar variables intermedias (1,3-bisfosfoglicerato, 3-fosfoglicerato, 2-fosfoglicerato y fosfoenolpiruvato), manteniendo la estequiometría global de la reacción y las relaciones energéticas esenciales del proceso.

En términos matemáticos, esta reducción disminuye el orden del sistema y facilita la implementación numérica del modelo mediante métodos de integración explícitos (Euler o Runge-Kutta), reduciendo la carga computacional y mejorando la estabilidad de la simulación discreta. Desde el punto de vista electrónico, la simplificación implica una menor cantidad de operaciones aritméticas, lo cual reduce el uso de recursos lógicos (LUTs, flip-flops y registros) en la plataforma FPGA y permite una representación más eficiente de la dinámica de la glucólisis sin comprometer su coherencia funcional.

8.3. Alcance y simplificaciones / supuestos

Para simplificar la representación del sistema y hacer viable su futura implementación digital, se adoptan los siguientes supuestos:

- El sistema se considera homogéneo y de volumen constante, sin gradientes espaciales.
- La temperatura y el pH se mantienen constantes durante el proceso.
- Se trabaja con un modelo determinista, sin ruido estocástico ni fluctuaciones aleatorias.
- Las enzimas actúan bajo cinética de Michaelis Menten, y las reacciones reversibles se aproximan como unidireccionales cuando la constante de equilibrio lo permite.
- La concentración total de adenilatos (ATP + ADP) se mantiene constante, permitiendo modelar solo la conversión relativa entre ambas especies.
- Se consideran únicamente los metabolitos intermedios desde la glucosa (GLC) hasta el piruvato (PYR), omitiendo pasos menores de transporte o consumo externo.

Estos supuestos reducen la complejidad del sistema, preservando los aspectos dinámicos esenciales del proceso metabólico.

8.4. Definición de variables de estado

Las variables del modelo representan las concentraciones molares de los principales intermediarios de la glucólisis, cada variable $S_i(t)$ describe la concentración del

metabolito i en función del tiempo. La Tabla 8.1 resume las especies seleccionadas, su notación y unidades de medida.

Símbolo	Variable	Descripción	Unidad
GLC	Glucosa	Substrato inicial del proceso	mM
G6P	Glucosa-6-fosfato	Producto de la hexoquinasa	mM
F6P	Fructosa-6-fosfato	Intermedio previo a la fosforilación	mM
FBP	Fructosa-1,6-bisfosfato	Producto de la PFK	mM
GAP	Gliceraldehído-3-fosfato	Intermedio de la fase de pago	mM
PYR	Piruvato	Producto final de la glucólisis	mM
ATP	Adenosín trifosfato	Transportador energético	mM
ADP	Adenosín difosfato	Producto de hidrólisis del ATP	mM

Tabla 1. Variables de estado del modelo glucolítico.

Estas variables conforman el **vector de estado**
$\mathbf{S}(t) = [GLC, G6P, F6P, FBP, GAP, PYR, ATP, ADP]^T$.

8.5. Parámetros y constantes cinéticas

Los parámetros cinéticos determinan la velocidad de las reacciones enzimáticas y se obtienen de fuentes bibliográficas Teusink et al. [22] Heinrich & Schuster [19] para luego hacer estimaciones normalizadas y obtener un modelo simplificado.

Se tomaron los valores V_{max} de Teusink et al. [22] $V_{max,1} = 0.84U * mg\ protein^{-1}$ y $V_{max,3} = 0.68U * mg\ protein^{-1}$, y se les hizo una conversión para obtener el dato en mM, como en bioquímica $1\mathrm{U} = 1\mu mol/min$, las unidades quedan como $1\mu mol/(min * mg\ proteina)$. Para modelar concentraciones químicas (mM) se necesita saber cuánto volumen de líquido (citosol) hay por cada gramo de proteína.

> *"Cytosolic metabolite concentrations were calculated by a conversion factor of 3.75 mL of cell water per gram protein."* [22].

Con la relación $3.75\ mL\ citosol/g\ proteina$, y el dato en mg, el factor se convierte, para obtener $0.00375\ mL\ citosol/mg\ proteina$, y para convertir V_{max}, se divide por este factor y se obtienen las unidades como $\mu mol/(mL * min)$, lo que es equivalente a 1mM (milimolar), luego se divide entre 60 para hacer la conversión de minutos a segundos, y como último paso se divide entre 4 para escalar el modelo y ajustarlo al formato Q12.20.

En el caso de $V_{max,2}$, si se deja la Isomerasa en el 1.4 estándar, está peligrosamente cerca de la velocidad de entrada (0.9), por lo que, si hay un pequeño pico de glucosa, la isomerasa podría saturarse y convertirse en un cuello de botella artificial, lo cual sería biológicamente incorrecto, entonces se le da un factor de seguridad del 40%. Usar 2.0 garantiza que la Isomerasa cumpla su función biológica de ser un paso rápido y transparente, dejando que el control real del sistema recaiga sobre la PFK.

Se aplicó un redondeo a un decimal significativo en los parámetros cinéticos para reflejar la incertidumbre experimental (típicamente del 10-20%) inherente a las mediciones biológicas *in vitro* y para facilitar la implementación y depuración en hardware.

Parámetro	Enzima / Reacción	Valor	Justificación del Valor
$V_{max,1}$	Hexoquinasa (R1)	0.9 mM/s	Escalado directo de Teusink (3.7 / 4). Mantiene el flujo de entrada controlado.
$K_{m,1}$	Constante Michaelis Menten (GLC)	0.1 mM	Valor estándar de Teusink. ~0.08 redondeado para reflejar la incertidumbre experimental.
$V_{max,2}$	Isomerasa (R2)	2.0 mM/s	Valor intermedio lo suficientemente alto para no frenar a la HK.
$K_{m,2}$	Constante Michaelis Menten (G6P)	1.4 mM	Estándar genérico de Teusink
$V_{max,3}$	PFK (R3)	0.75 mM/s	Escalado directo de Teusink (3.0 / 4). Es el paso más lento, biológicamente correcto para el control.
$K_{m,3}$	Constante Michaelis-Menten (F6P)	0.1 mM	Estándar genérico de Teusink
K_i	Constante de Inhibición por ATP	1.0 mM	Se mantiene el valor de Heinrich & Schuster para asegurar una regulación fuerte visible.
k_4	Aldolasa (R4)	1.0 s^{-1}	Constante de primer orden estándar.
$V_{max,5}$	GAPDH/Fase Pago (R5)	5.0 mM/s	Escalado de Teusink (20 / 4). Es ~5 veces más rápida que la entrada.
$K_{m,5}$	Constante Michaelis Menten (GAP)	0.2 mM	Valor estándar de Teusink. 0.21 redondeado para reflejar la incertidumbre experimental.

Tabla 2. Parámetros cinéticos y de modelo

8.6. Reacciones consideradas y notación estequiométrica

Se incluyen las reacciones principales que constituyen el flujo central de la glucólisis, que, para efectos de modelación, se enumeran de la siguiente forma:

R1: Glucosa + ATP → G6P + ADP

R2: G6P → F6P

R3: F6P + ATP → FBP + ADP

R4: FBP → 2 GAP

R5: GAP +ADP → PYR + ATP

La matriz estequiométrica S se expresa como:

Metabolito	R1	R2	R3	R4	R5
GLC	-1	0	0	0	0
G6P	+1	-1	0	0	0
F6P	0	+1	-1	0	0
FBP	0	0	+1	-1	0
GAP	0	0	0	+2	-1
PYR	0	0	0	0	+1
ATP	-1	0	-1	0	+1
ADP	+1	0	+1	0	-1

Tabla 3. matriz estequiométrica S

Los coeficientes negativos representan consumo, y los positivos, producción. Esta formulación permite expresar el sistema como $\frac{d\mathbf{S}}{dt} = \mathrm{S} \cdot \mathrm{v}(\mathrm{S}, \mathrm{k})$.

8.7. Formulación de las ecuaciones diferenciales continuas

Cada metabolito se asocia a una ecuación diferencial que describe su tasa de cambio temporal, y las velocidades de reacción v_j se modelan mediante leyes cinéticas de Michaelis Menten, ajustadas con términos de inhibición o regulación cuando corresponde:

$$v_1 = \frac{V_{max,1}[GLC]}{K_{m,1}+[GLC]} \quad (8.1)$$

$$v_2 = \frac{V_{max,2}[G6P]}{K_{m,2}+[G6P]} \quad (8.2)$$

$$v_3 = \frac{V_{max,3}[F6P]}{K_{m,3}(1+[ATP]/K_i)+[F6P]} \quad (8.3)$$

$$v_4 = k_4[FBP] \quad (8.4)$$

$$v_5 = \frac{V_{max,5}[GAP]}{K_{m,5}+[GAP]} \quad (8.5)$$

A partir de estas expresiones o mediante la matriz estequiométrica, las ecuaciones diferenciales continuas son:

$$\frac{d[GLC]}{dt} = -v_1 \quad (8.6)$$

$$\frac{d[G6P]}{dt} = v_1 - v_2 \quad (8.7)$$

$$\frac{d[F6P]}{dt} = v_2 - v_3 \quad (8.8)$$

$$\frac{d[FBP]}{dt} = v_3 - v_4 \quad (8.9)$$

$$\frac{d[GAP]}{dt} = 2v_4 - v_5 \quad (8.10)$$

$$\frac{d[PYR]}{dt} = v_5 \quad (8.11)$$

$$\frac{d[ATP]}{dt} = -v1 - v_3 + v_5 \quad (8.12)$$

$$\frac{d[ADP]}{dt} = v1 + v_3 - v_5 \quad (8.13)$$

Este sistema representa el modelo continuo base que posteriormente será discretizado.

8.8. Justificación de las leyes cinéticas usadas

La elección del modelo de Michaelis Menten se fundamenta en su capacidad para describir la saturación enzimática bajo concentraciones moderadas de sustrato, es lo suficientemente simple para ser computacionalmente viable, pero lo suficientemente detallado para capturar el comportamiento biológico esencial, lo que resulta adecuado para la glucólisis.

En el caso de la fosfofructoquinasa (PFK), se añadió un término de inhibición por ATP, reflejando su papel regulador en la vía. Estas formulaciones han sido ampliamente utilizadas en modelos clásicos de glucólisis, como los propuestos por Heinrich y Schuster [19] y Teusink et al. [22], cuyos resultados se han validado experimentalmente.

8.9. Comprobaciones iniciales y pruebas analíticas

Previo a la discretización, se verificó el equilibrio estequiométrico de cada reacción y la conservación global de masa, el modelo continuo fue simulado numéricamente mediante el método RK4 Orden 4 con un paso de integración $dt = 0.008\,s$ y $\frac{dt}{2} = 0.004\,s$ mostrando la estabilización de las concentraciones en un punto de equilibrio al alcanzar $t = 19.492\ s$ y $t = 20.2\,s$ respectivamente. Ambas simulaciones se ejecutaron con las mismas condiciones iniciales y parámetros cinéticos definidos en la Tabla 8.2. Este resultado confirma que las ecuaciones están correctamente acopladas y que las relaciones entre reacciones son coherentes alcanzando un punto de equilibrio.

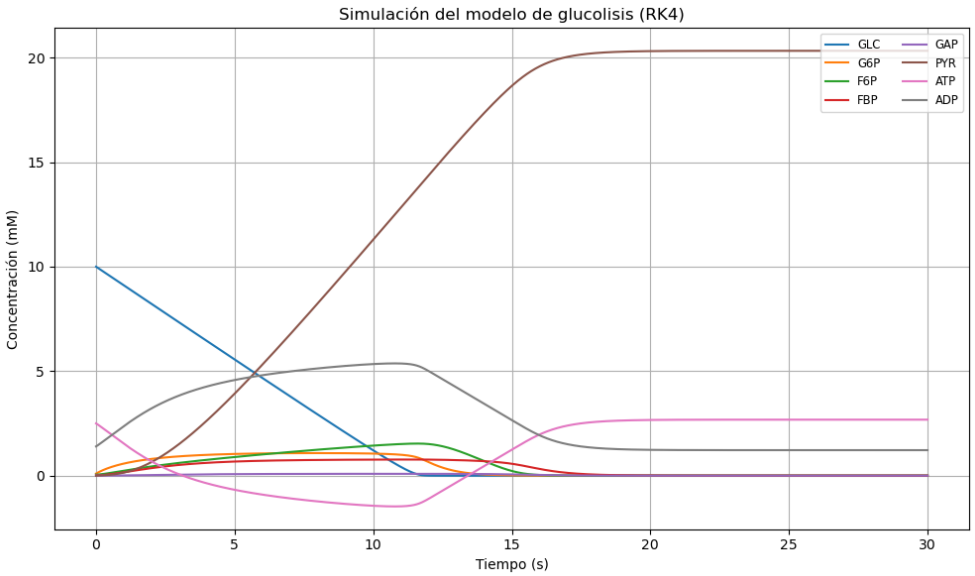


Figura 1. Simulación del modelo continuo de la glucólisis bajo cinética de Michaelis Menten mediante integración numérica de cuarto orden (RK4).

Las soluciones obtenidas fueron comparadas punto a punto interpolando los resultados de $dt/2$ sobre la malla temporal de dt.

Las métricas de error se calcularon como:

$$E_{\infty} = \max_{i,t}(S_i^{dt}(t) - S_i^{\frac{dt}{2}}(t)) \tag{8.14}$$

$$E_{RMS} = \sqrt{\frac{1}{N_t N_s}\sum_{t,i}(S_i^{dt}(t) - S_i^{dt/2}(t))^2} \tag{8.15}$$

Los resultados evidencian un error máximo $E_{\infty} = 4.355 \times 10^{-8}$ mM y un error medio $E_{RMS} = 1.577 \times 10^{-9}$ mM, lo cual se encuentra dentro del margen de precisión numérica de doble flotante. El tiempo de estabilización se mantuvo prácticamente invariante ($19.492\,s$ para dt y $20.2\,s$ para $dt/2$), confirmando la robustez de la integración y la correcta formulación del sistema de ecuaciones.

El hecho de que los errores sean de un orden de magnitud tan bajo (10^{-8} - 10^{-9}) es una consecuencia directa de la alta precisión del método RK4 Orden 4 y demuestra que la simulación con $dt = 0.01$ s está extremadamente cerca de la solución teórica. Los valores de error son muchos órdenes de magnitud inferiores a cualquier concentración de metabolito, acercándose a los límites de la precisión de la aritmética de doble flotante (IEEE 754) y, por lo tanto, se consideran despreciables. [36]

Estos resultados permiten adoptar $dt = 0.004$ s como paso nominal de integración en las simulaciones y como referencia para la discretización digital del modelo, garantizando un error despreciable respecto al comportamiento continuo.

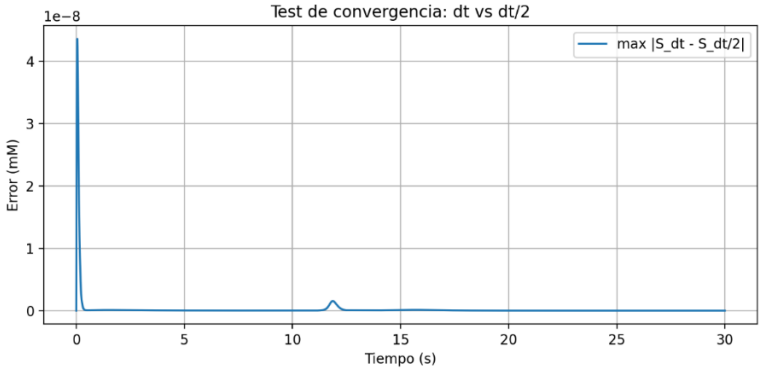


Figura 2. Evolución del error máximo (S^dt-S^(dt/2)) a lo largo del tiempo para el modelo continuo de la glucólisis.

9. Discretización y validación teórica del modelo

Una vez validado el modelo matemático continuo de la glucólisis, el siguiente paso consiste en su discretización, proceso mediante el cual las ecuaciones diferenciales ordinarias (EDOs) se transforman en expresiones de diferencias finitas. Esta transformación permite representar el comportamiento dinámico del sistema mediante operaciones aritméticas secuenciales aptas para su ejecución en un entorno digital, como una FPGA.

9.1. Método de discretización

Se adoptó el método de Euler hacia adelante (Forward Euler), un esquema explícito de primer orden ampliamente utilizado en la implementación digital de modelos dinámicos por su sencillez y eficiencia computacional. Aunque existen métodos de mayor precisión, como Runge Kutta o Adams Bashforth, su uso implicaría un incremento significativo en las operaciones aritméticas y, por tanto, en el consumo de recursos lógicos de la FPGA. [1]

En forma general, un sistema continuo descrito por una ecuación diferencial ordinaria del tipo:

$$\frac{dX(t)}{dt} = f(X, t) \tag{9.1}$$

puede aproximarse en dominio discreto mediante:

$$X[k+1] = X[k] + \Delta t \cdot f(X[k], t_k) \tag{9.2}$$

donde $X[k]$ es el vector de concentraciones en el instante discreto k, y Δt es el paso de integración.

9.2. Proceso de discretización del modelo

El modelo continuo de la glucólisis se compone de un conjunto de ecuaciones diferenciales no lineales que describen las velocidades de reacción v_i asociadas a las transformaciones bioquímicas de cada metabolito.

Por ejemplo, para una reacción genérica bajo cinética de Michaelis Menten:

$$v_i = \frac{V_{max,i} \cdot [S_i]}{K_{m,i} + [S_i]} \tag{9.3}$$

la tasa de variación del metabolito S_i en el modelo continuo se expresa como:

$$\frac{d[S_i]}{dt} = \sum_j v_{ij} v_j \quad (9.4)$$

donde v_{ij} representa el coeficiente estequiométrico del metabolito i en la reacción j. Al aplicar la discretización mediante Euler directo, la ecuación diferencial de cada metabolito se transforma en una ecuación recursiva discreta de la forma:

$$[S_i]_{k+1} = [S_i]_k + \Delta t \cdot \sum_j v_{ij} v_j([S]_k) \quad (9.5)$$

Por tanto, cada paso temporal de simulación se obtiene evaluando las velocidades v_j con las concentraciones actuales y actualizando las concentraciones para el siguiente instante $k + 1$.

De manera explícita, para el conjunto de metabolitos del modelo reducido:

$$[GLC]_{k+1} = [GLC]_k - \Delta t \cdot v_1([GLC]_k) \quad (9.6)$$

$$[G6P]_{k+1} = [G6P]_k + \Delta t \cdot (v_1([GLC]_k) - v_2([G6P]_k)) \quad (9.7)$$

$$[F6P]_{k+1} = [F6P]_k + \Delta t \cdot (v_2([G6P]_k) - v_3([F6P]_k)) \quad (9.8)$$

$$[FBP]_{k+1} = [FBP]_k + \Delta t \cdot (v_3([F6P]_k) - v_4([FBP]_k)) \quad (9.9)$$

$$[GAP]_{k+1} = [GAP]_k + \Delta t \cdot (2 * v_4([FBP]_k) - v_5([GAP]_k)) \quad (9.10)$$

$$[PYR]_{k+1} = [PYR]_k + \Delta t \cdot v_5([GAP]_k) \quad (9.11)$$

De igual forma, los nucleótidos energéticos se actualizan como:

$$[ATP]_{k+1} = [ATP]_k + \Delta t \cdot (-v_1 - v_3 + v_5) \quad (9.12)$$

$$[ADP]_{k+1} = [ADP]_k + \Delta t \cdot (v_1 + v_3 - v_5) \quad (9.13)$$

Este conjunto de ecuaciones define el modelo discreto de la glucólisis, en el que las concentraciones se actualizan iterativamente a partir de sus valores previos, utilizando únicamente operaciones de multiplicación, división y suma, compatibles con la aritmética digital en punto fijo.

9.3. Validación teórica del modelo discreto

Dada la naturaleza del método de Euler, cuyo error de truncamiento es de primer orden, se requiere un paso de integración significativamente más pequeño para

alcanzar una precisión comparable al paso de integración de referencia $\Delta t = 0.004\ s$ del método de referencia RK4 de orden superior.

9.3.1. Rangos de error

La selección de los rangos de error aceptables para validar la simulación de Euler, estableciendo como objetivos un error RMS (E_rms) inferior a 5 × 10^{-4} mM y un error absoluto máximo (E_inf) inferior a 2 × 10^{-3} mM, se fundamenta en el contexto de las escalas biológicas del propio modelo. El criterio principal es asegurar que el error numérico sea significativamente menor que las magnitudes de los metabolitos menos abundantes (FBP, GAP, ~10^{-2} mM) para no distorsionar su dinámica. [37]

El E_rms garantiza la fidelidad del comportamiento promedio y del estado de equilibrio del sistema, manteniéndose por debajo del 5% de la concentración de estas especies clave. Por su parte, el E_inf establece un límite para el peor caso de desviación, que típicamente ocurre en la fase transitoria inicial, asegurando que la trayectoria de la simulación nunca se desvíe de la referencia de alta precisión de una manera que invalide cualitativamente el modelo.

9.3.2. Paso de integración

Se realizaron simulaciones iterativas con el método de Euler, comparando los resultados contra la referencia RK4 para cuantificar el error (E_rms y E_inf) encontrando que el valor $dt = 0.001$ s satisface consistentemente estos criterios de error, manteniéndose dentro de los rangos aceptables, por lo tanto, se adopta como valor nominal para las simulaciones y para la representación discreta implementada en hardware.

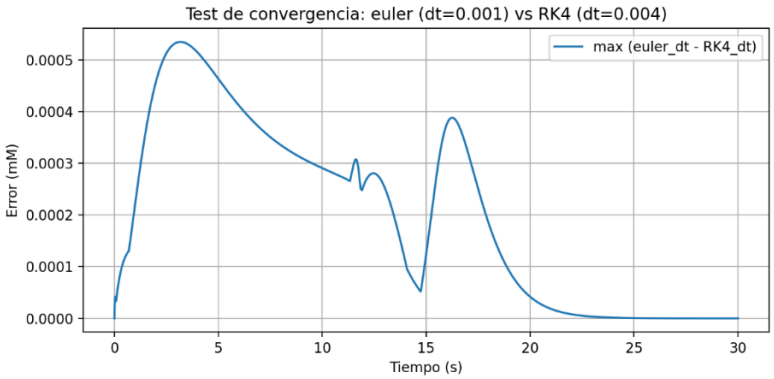


Figura 3. Evolución del error máximo (〖Euler〗 ^dt- 〖RK4〗 ^dt) a lo largo del tiempo.

Con un error RMS (E_rms) de 1.361 × 10^{-4} mM y un error absoluto máximo (E_inf) inferior a 5.348 × 10^{-4} mM, esta elección representa el equilibrio óptimo entre la simplicidad computacional del método de Euler, necesaria para la implementación en

hardware (FPGA), y la fidelidad numérica para garantizar que el modelo discreto emule de manera confiable el comportamiento del sistema biológico continuo.

9.4. Conclusión del proceso de discretización

El proceso de discretización de las EDOs permite trasladar el modelo bioquímico continuo a una forma digital implementable en FPGA, las ecuaciones recursivas obtenidas mantienen la coherencia estequiométrica y la estabilidad dinámica del sistema, mientras que la elección del método de Euler garantiza una arquitectura aritmética simple y determinista.

Los resultados de convergencia y validación teórica demuestran que la versión discreta reproduce adecuadamente la evolución temporal de las concentraciones, constituyendo una base sólida para su implementación en lenguaje de descripción de hardware (HDL).

10. Traducción del modelo a lógica digital

Una vez obtenida la formulación discreta del modelo de glucólisis, el siguiente paso consiste en su traducción a una arquitectura digital, donde las ecuaciones recursivas se representan mediante operaciones aritméticas implementadas en hardware. Este proceso implica definir los tipos de datos, el formato numérico y los bloques funcionales básicos que reproducen el comportamiento dinámico del sistema.

10.1. Representación numérica en punto fijo

En una FPGA, el uso de aritmética de punto flotante resulta costoso en términos de recursos lógicos (LUTs, flip-flops, DSPs) y tiempo de propagación, debido a la complejidad de los operadores de normalización y alineamiento de exponente, por esta razón, se optó por la representación en punto fijo, que ofrece una relación más favorable entre precisión, velocidad y consumo de área lógica.

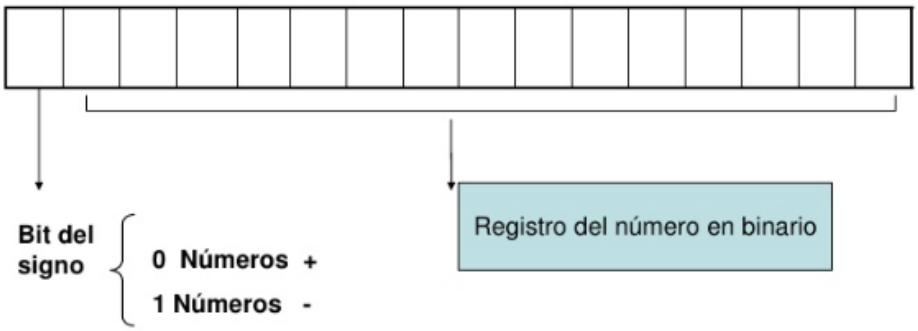


Figura 4. Representación numérica en punto fijo.

Característica	Punto flotante (floating-point)	Punto fijo (fixed-point)
Representación	Notación científica: mantisa × $2^{\text{exponente}}$	Valor entero escalado por un factor constante (número de bits fraccionales)
Rango	Muy amplio. Puede representar valores muy grandes o pequeños	Limitado (depende del número de bits asignados)
Precisión	Variable. Mayor precisión absoluta para números pequeños, menor para números grandes	Constante en todo el rango (paso fijo)
Operaciones	Requieren circuitos complejos (sumadores, normalizadores, alineamiento de exponente, etc.)	Más simples: solo sumas, restas, multiplicaciones y desplazamientos
Recursos FPGA	Alto uso de LUTs, FFs y DSPs (hasta 10× o 20× más)	Bajo uso de recursos
Velocidad	Más lento (mayor latencia por operaciones complejas)	Más rápido (menor latencia, pipelines simples)

Tabla 4. Comparación entre punto flotante y punto fijo.

Cada variable del modelo ([GLC], [GAP], [PYR], [ATP], etc) se representa mediante un formato fijo Qm.n, donde:

- m = número de bits para la parte entera (incluido el signo)
- n = número de bits para la parte fraccionaria
- el valor real se calcula como:

$$x_{\text{real}} = x_{\text{entero}} \times 2^{-n}$$

Se seleccionó el formato Q12.20, que utiliza 32 bits (1 de signo, 11 enteros, 20 fraccionarios), proporcionando:

- rango de representación: $[-2048, +2047.99]$ mM

- resolución mínima: $2^{-20} = 0.000000954$

Figura 5. Representación del formato Q12.20.

La elección de un formato de 32 bits en total (Q12.20) se alinea directamente con la arquitectura nativa de los componentes de la FPGA, como los multiplicadores de hardware (DSPs), que están optimizados para operar con anchos de datos de potencias de dos (8, 16, 32 bits), lo que resulta en un diseño más rápido, con menor consumo de recursos y una complejidad reducida en comparación con arquitecturas de datos de tamaño no estándar.

Para la parte entera se seleccionaron 12 bits, lo que garantiza que las operaciones de suma y resta no produzcan desbordamientos (overflows) incluso durante picos transitorios donde las concentraciones podrían exceder temporalmente el rango de equilibrio del sistema, asegurando así la estabilidad numérica de la implementación.

La elección de un formato común para todas las variables simplifica las operaciones de suma y resta, mientras que las multiplicaciones requieren un ajuste de escala mediante desplazamiento de bits:

$$Qm_1.n_1 \times Qm_2.n_2 = Q(m_1 + m_2).(n_1 + n_2) \underset{\text{shift}}{\overset{2^{-n}}{\rightarrow}} Qm.n \qquad (9.14)$$

Para la parte fraccionaria se seleccionó un formato con 20 bits, lo que resulta en una resolución de (~10^{-7} mM), varios ordenes de magnitud más fina que la de los metabolitos menos abundantes (~10^{-2} mM). Este "colchón" de bits de precisión actúa como una salvaguarda, asegurando que, incluso después del truncamiento post multiplicación, la precisión remanente sea suficiente para representar fielmente las pequeñas variaciones en las concentraciones, evitando así la degradación de la dinámica del modelo.

La elección del formato Q12.20 por encima de uno Q8.24 se basa en priorizar la robustez y la estabilidad del sistema sobre la precisión numérica en el "piso de ruido", ya que, para una emulación en hardware, es preferible aceptar un error relativo mayor en los valores más pequeños (que ya son inherentemente ruidosos) a cambio de garantizar que el sistema nunca falle debido a un desbordamiento.

Aunque el formato simétrico Q16.16 es común, se descartó en favor del Q12.20, ya que en la naturaleza del sistema glucolítico las concentraciones de intermediarios

como GAP o FBP pueden ser muy bajas (<0.05 mM), requiriendo una alta resolución fraccionaria. El formato Q12.20 ofrece 4 bits adicionales de precisión decimal (2^{-20} vs 2^{-16}), reduciendo el error de cuantización en los valores pequeños en un factor de 16, mientras que los 12 bits enteros (±2048) son más que suficientes para el rango dinámico del modelo, haciendo innecesarios los 16 bits enteros del formato Q16.16.

Característica	Formato "mínimo" teórico (Q4.7)	Mejor formato de 16 bits (Q6.10)	Formato de 32 bits (Q16.16)	Formato con máxima precisión (Q12.20)
Bits totales	11 bits	16 bits	32 bits	32 bits
Rango entero	[-8, +7.99]	[-32, +31.99]	[-2048, +2047.99]	[-2048, +2047.99]
Resolución	~0.00781 mM	~0.000977 mM	~0.000015 mM	~0.00000095 mM
Error máximo relativo	78.1 %	9.8 %	0.15%	0.01 %
Riesgo de desbordamiento (Overflow)	Significativo	Muy Bajo	Insignificante	Insignificante
Costo de recursos FPGA	Teóricamente mínimo	Bajo	Medio-alto	Medio-alto

Tabla 5. Comparativa entre formato mínimo teórico y formato elegido.

10.2. Ventajas de la traducción en punto fijo

La adopción del formato fijo ofrece múltiples ventajas para la emulación en FPGA:

- **Determinismo temporal:** todos los cálculos se ejecutan en un número fijo de ciclos de reloj.
- **Menor consumo de recursos:** al evitar unidades de coma flotante cada multiplicación o suma usa menos celdas lógicas y DSPs.
- **Alta velocidad de operación:** permite integrar dinámicas bioquímicas en tiempo real.
- **Facilidad de síntesis:** los operadores Qm.n se describen directamente en HDL sin necesidad de librerías complejas.

Estas características convierten la aritmética de punto fijo en la opción más adecuada para representar modelos bioquímicos discretizados en hardware, asegurando fidelidad con el modelo matemático y eficiencia en la implementación física.

10.3. Traducción de las ecuaciones discretas

Las ecuaciones discretas del modelo (obtenidas en el punto 9) se implementan como relaciones recursivas iterativas, en las cuales cada metabolito se actualiza con base en su valor anterior y las velocidades de reacción calculadas en el instante actual.

Por ejemplo, la ecuación de actualización para el metabolito GAP:

$$[GAP]_{k+1} = [GAP]_k + \Delta t \cdot (2 * v_4 - v_5) \qquad (9.15)$$

se traduce digitalmente a la forma:

$$GAP\_next = GAP\_reg + (dt \times ((2 \times v4) - v5)) \qquad (9.15)$$

donde GAP_reg representa el valor almacenado en el registro de la iteración anterior, cada término de velocidad v_i corresponde a un módulo funcional que calcula la expresión de Michaelis Menten en aritmética de punto fijo:

$$v_i = \frac{V_{max,i} \times S_i}{K_{m,i} + S_i} \qquad (9.15)$$

Este cálculo requiere las operaciones básicas: multiplicación, suma y división. La división se implementa mediante un módulo recíproco o divisor iterativo, dependiendo de los recursos disponibles en la FPGA.

10.4. Bloques funcionales de la arquitectura digital

El modelo se compone de bloques lógicos interconectados que reproducen la estructura matemática del sistema bioquímico.

Los principales módulos funcionales son:

- **Bloque de multiplicación (MUL):** Implementa la operación $a \times b$ en formato Qm.n y utiliza celdas DSP internas de la FPGA para garantizar baja latencia.
- **Bloque de suma/resta (ADD/SUB):** Opera sobre dos operandos Qm.n con control de signo y saturación.
- **Bloque de división (DIV):** Calcula a/b mediante un algoritmo iterativo de Newton Raphson o un módulo de reciprocidad precalculada.
- **Bloque de registro (REG):** Almacena el valor de cada concentración $[S_i]$ y actualiza el resultado en cada ciclo de reloj.
- **Bloque de control temporal:** Genera las señales de reloj y habilitación necesarias para realizar las actualizaciones secuenciales con paso discreto Δt.
- **Bloque de salida (LCD/LED):** Permite observar las variables del sistema en tiempo real.

11. Selección del entorno de desarrollo

La correcta implementación del modelo bioquímico discretizado en hardware digital requiere definir un entorno de desarrollo que garantice compatibilidad entre el lenguaje de descripción de hardware (HDL), las herramientas de simulación y síntesis, y la arquitectura física del dispositivo FPGA seleccionado. Esta fase establece la infraestructura técnica necesaria para la emulación eficiente y verificable del sistema glucolítico propuesto.

11.1. Selección del lenguaje HDL

Se eligió VHDL como lenguaje de descripción de hardware debido a su naturaleza fuertemente tipada, estructura modular y amplia compatibilidad con las herramientas de diseño utilizadas en dispositivos de Altera (Intel FPGA). VHDL permite representar de forma clara las ecuaciones diferenciales discretizadas mediante estructuras secuenciales y concurrentes, facilitando la descripción de procesos como la integración numérica, el control de flujo de datos y el manejo de señales en punto fijo.

El lenguaje ofrece ventajas en proyectos de carácter académico y científico, donde la trazabilidad del código y la legibilidad son fundamentales para validar las etapas del diseño digital. Además, la existencia de bibliotecas de precisión fija (fixed_pkg) y aritmética extendida (numeric_std) posibilita la implementación de operaciones matemáticas propias del modelo glucolítico con control explícito sobre el formato numérico y la saturación.

11.2. Elección de la FPGA

El modelo Terasic DE2-115, fabricado por Intel (antes Altera), fue seleccionado como plataforma principal para la implementación del modelo debido a su equilibrio entre capacidad lógica, recursos de memoria y soporte educativo. La tarjeta incorpora un Cyclone IV E EP4CE115F29C7**,** que dispone de aproximadamente 114,480 elementos lógicos (LEs), 3.9 Mbits de memoria embebida y 532 multiplicadores dedicados de 18x18 bits, críticos para implementar el formato numérico de 32 bits sin comprometer el rendimiento, y múltiples interfaces de entrada/salida útiles para pruebas y visualización de resultados.

Estas características permiten la integración de un modelo bioquímico discretizado de mediana complejidad, incluyendo unidades de cálculo aritmético, almacenamiento temporal de concentraciones y módulos de control. El soporte nativo de esta FPGA para las características modernas de VHDL, para operaciones en punto fijo, junto con su compatibilidad con herramientas de desarrollo libre y académicas, la convierten en una opción adecuada para proyectos experimentales que buscan demostrar la viabilidad de una emulación biológica digital.

11.3. Configuración de herramientas y entorno de desarrollo

El flujo de diseño adoptado se basa en el entorno Quartus Prime Lite 20.1.1, herramienta oficial para síntesis y configuración de dispositivos Altera compatible con la familia de FPGAs Cyclone IV, este software permite la síntesis, asignación de pines, análisis temporal y generación del archivo de configuración (.sof) para la FPGA. La simulación funcional y temporal del modelo se lleva a cabo con ModelSim-Altera, donde se verifican los waveforms correspondientes a las variables discretizadas ([GAP], [ATP], [ADP], etc) y su evolución temporal.

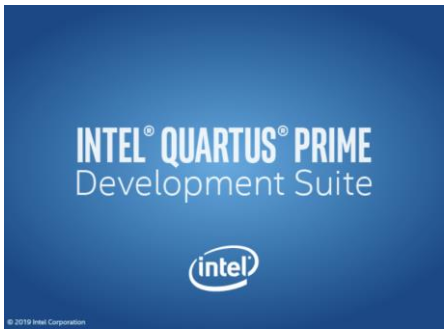


Figura 6. Quartus Prime Lite como entorno de desarrollo.

El entorno se complementa con módulos de verificación desarrollados en testbenches de VHDL, donde se definirán estímulos representativos de las condiciones iniciales del modelo bioquímico para validar su comportamiento dinámico frente a los resultados numéricos del modelo continuo, de esta manera, la configuración completa del entorno garantiza la coherencia entre las fases de modelado matemático, discretización y traducción a lógica digital.

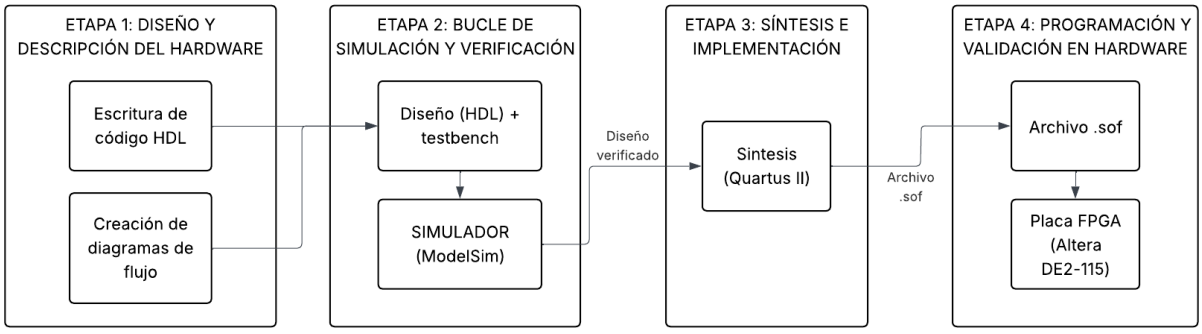


Figura 7. Diagrama del flujo de desarrollo

12. Diseño lógico y diagramas funcionales

La traducción del modelo discretizado a hardware exige definir una arquitectura modular y escalable que implemente las operaciones aritméticas necesarias y preserve la coherencia bioquímica del sistema. Se presenta una revisión de bloques funcionales

clásicos para sistemas numéricos digitales, las tablas de parámetros, diagramas preliminares del diseño, el diagrama de bloques completo del proyecto, y la asignación de recursos.

12.1. Revisión de ejemplos similares y bloques básicos

Antes de detallar el diseño completo, se revisan ejemplos y bloques reutilizables que sirven como plantilla:

a) DDA (Digital Differential Analyzer) / Integrador discreto

La función principal de este bloque es implementar la ecuación de actualización del sistema, definida como $x_{k+1} = x_k + \Delta t \cdot f(x_k)$, donde a nivel de componentes, su arquitectura requiere un registro para almacenar el estado actual x_k, un multiplicador para escalar la función de cambio por el paso de tiempo $\Delta t \cdot f$, un sumador para acumular el resultado y un bloque de saturación para limitar los valores.

b) Sumadores y restadores

Estos módulos se implementan mediante operaciones de suma entera con signo, respetando estrictamente el formato numérico Qm.n seleccionado. Se debe verificar el alineamiento de la parte fraccionaria de los operandos antes de realizar la suma y aplicar una lógica de saturación en la salida para gestionar los casos en los que el resultado exceda los límites de representación del formato.

c) Multiplicadores

Para la implementación de multiplicaciones, se debe priorizar el uso de bloques *DSP slices* internos de la FPGA, dado que el producto intermedio genera un resultado con el doble de ancho de bits, es necesario gestionar este formato expandido antes de realizar el truncamiento o desplazamiento (*shift*). Para operar a altas frecuencias, se emplea registros de *pipeline* en la ruta de datos y se utilizan técnicas de redondeo (*rounding*) al desplazar los bits para minimizar el error.

d) Registros / bancos de registros

El almacenamiento de los estados del sistema $[S_i]_k$ entre ciclos de simulación se implementa utilizando registros y es necesario incorporar lógica de habilitación en estos para permitir su control preciso por parte de la Máquina de Estados Finitos (FSM) para mantener la coherencia de los datos cuando los cálculos aritméticos requieren múltiples ciclos de reloj para completarse.

e) Divisores / recíprocos

Existen diversas opciones para implementar la división, como la multiplicación por un recíproco precalculado (ideal si el divisor es constante), el método iterativo de Newton Raphson o el uso de tablas LUT y aproximaciones lineales (PWL). En este caso por manejar un formato Q12.20 el uso de LUTs sería muy costoso, por lo que se maneja el método de Newton Raphson.

12.2. Tablas de parámetros y formato de almacenamiento

Para la implementación en hardware, se ha definido un conjunto de parámetros que equilibra la fidelidad biológica con la estabilidad numérica necesaria.

Símbolo	Variable	Rango Esperado (mM)	Formato Q	Bits	Comentario
GLC	Glucosa	$0 - 100.0$	Q12.20	32	Variable de entrada (Sustrato).
G6P	Glucosa-6-P	$0 - 10.0$	Q12.20	32	Rango de acumulación esperada por K_m alto en R2.
F6P	Fructosa-6-P	$0 - 5.0$	Q12.20	32	Sustrato de la enzima reguladora.
FBP	Fructosa-1,6-BP	$0 - 10.0$	Q12.20	32	Intermediario (buffer).
GAP	Gliceraldehído-3P	$0 - 2.0$	Q12.20	32	Se mantiene bajo por la alta V_{max} de salida.
PYR	Piruvato	$0 - 200.0$	Q12.20	32	Producto final (2x Glucosa).
ATP	Adenosín Trifosfato	$0 - 20.0$	Q12.20	32	Cofactor e inhibidor.
ADP	Adenosín Difosfato	$0 - 20.0$	Q12.20	32	Complementario al ATP.

Tabla 6. Variables de estado y formato numérico.

Todas las variables de concentración se representan en formato Q12.20 (32 bits: 1 signo, 11 enteros, 20 fraccionarios), lo que permite un rango de $[-2048, +2047.99]$ mM y una resolución de aproximadamente 0.0000001 mM

Parámetro	Descripción	Valor	Unidad	Fuente / Justificación técnica
$V_{max,1}$	Vel. Máx. Hexoquinasa (R1)	0.9	mM/s	Teusink et al. (2000) escalado (/4). Controla la entrada.
$K_{m,1}$	Km Hexoquinasa (GLC)	0.1	mM	Teusink (Tabla 2). Alta afinidad por glucosa.
$V_{max,2}$	Vel. Máx. Isomerasa (R2)	2.0	mM/s	Valor de diseño con margen de seguridad (+40%) para compensar el alto Km y evitar cuellos de botella artificiales.
$K_{m,2}$	Km Isomerasa (G6P)	1.4	mM	Teusink (Tabla 2). Baja afinidad real, provoca acumulación fisiológica de G6P.
$V_{max,3}$	Vel. Máx. PFK (R3)	0.75	mM/s	Teusink et al. (2000) escalado (/4). Es el paso limitante principal.
$K_{m,3}$	Km PFK (F6P)	0.1	mM	Teusink (Apéndice 2, K_R). Afinidad en estado activo.
K_i	Inhibición ATP sobre PFK	1.0	mM	Aproximación fenomenológica para centrar la regulación en el rango fisiológico de ATP.
k_4	Cte. Aldolasa (R4)	1.0	s^{-1}	Constante de primer orden normalizada para transmisión de flujo.
$V_{max,5}$	Vel. Máx. Fase Pago (R5)	5.0	mM/s	Teusink et al. (2000) escalado (/4). Alta capacidad para drenar el sistema (aprox. 6x la entrada).
$K_{m,5}$	Km GAPDH (GAP)	0.2	mM	Teusink (Tabla 2).

Tabla 7. Parámetros cinéticos (guardados como constantes).

Estos valores se almacenarán en la FPGA, ya sea como constantes en el código VHDL (para optimización de área).

12.3. Diagramas preliminares

Para la implementación física, se han diseñado las estructuras fundamentales que se replicarán para cada metabolito y reacción.

- **Entrada y normalización**

Su función crítica es adaptar las señales externas (como valores binarios crudos provenientes de interruptores o registros de configuración) al formato de punto

fijo Q12.20 utilizado internamente. Este escalado inicial asegura la coherencia aritmética desde el primer ciclo, alineando el punto binario de las condiciones iniciales con el resto del *datapath* del sistema.

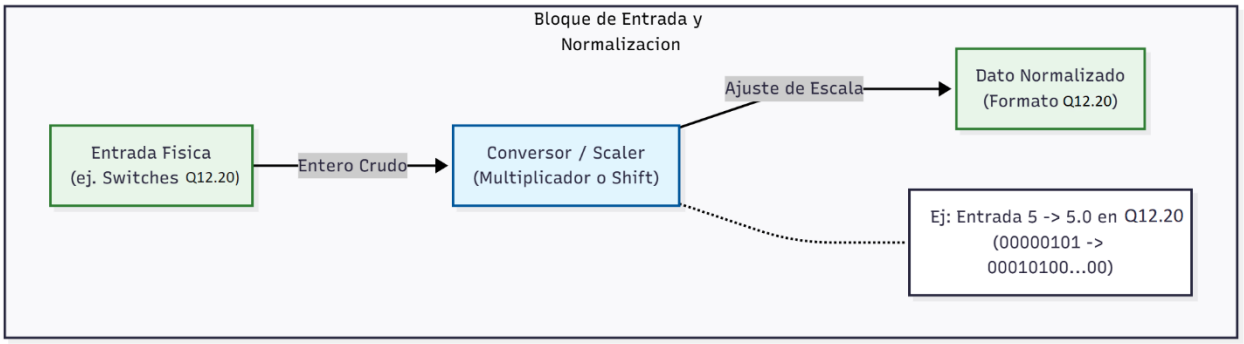


Figura 8. Diagrama de flujo del bloque de entrada.

- **Módulos de velocidad (v_i)**

Hay 3 tipos de módulos de velocidad, uno para HK, Isomerasa, GAPDH, otro para PFK, y otro para Aldolasa ($v_4 = k \cdot S$), este es solo un multiplicador simple. Los diagramas muestran la arquitectura de los módulos de cinética (Michaelis Menten) donde se observa la optimización de la división mediante una LUT de recíprocos para reducir el uso de lógica, manteniendo todo el flujo de datos en formato Q12.20 y el uso del inhibidor (ATP).

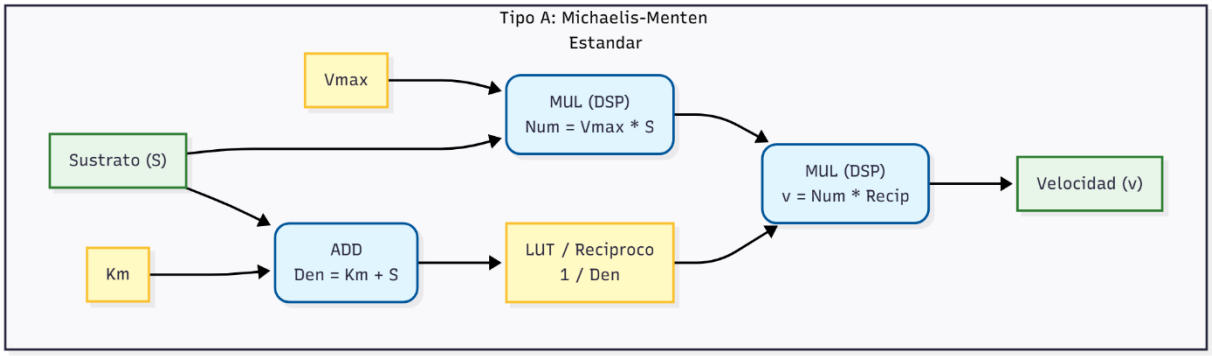


Figura 9. Diagrama de flujo de modulo vi tipo A.

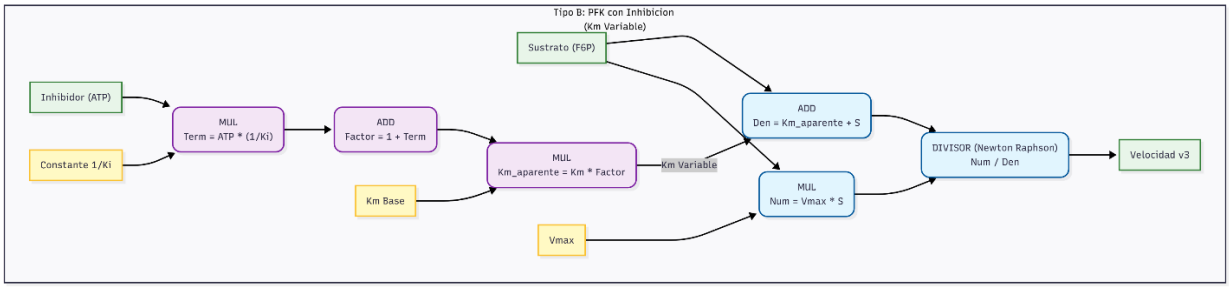


Figura 10. Diagrama de flujo de modulo vi tipo B.

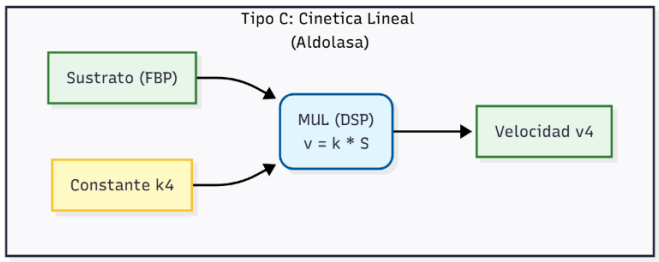


Figura 11. Diagrama de flujo de modulo vi tipo C.

- **Bloque integrador**

Se detalla el Integrador Digital Diferencial (DDA), en este bloque se implementa la discretización de Euler, y se destaca el bloque de saturación antes del registro para garantizar la estabilidad numérica ante picos transitorios.

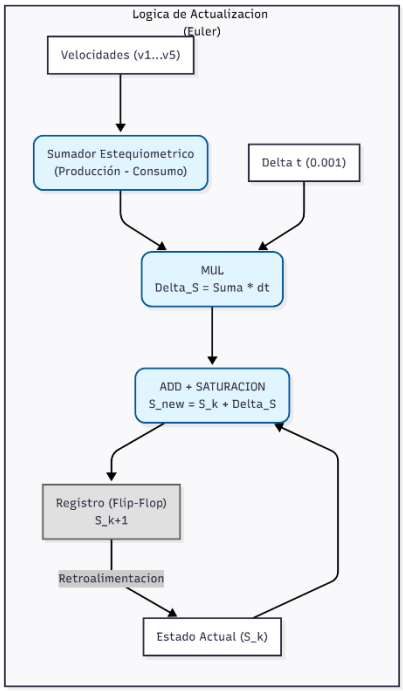


Figura 12. Diagrama de flujo de del bloque integrador.

- **Banco de registros**

Se detalla el Banco de Registros de Estado, que constituye la memoria a corto plazo del sistema dinámico. Este bloque es un vector paralelo de registros que mantienen simultáneamente el valor actual (S_k) de los 8 metabolitos, su escritura es síncrona y está estrictamente gobernada por la señal de habilitación de la unidad de control, garantizando que la transición de t hacia $t + \Delta t$ ocurra para todas las variables.

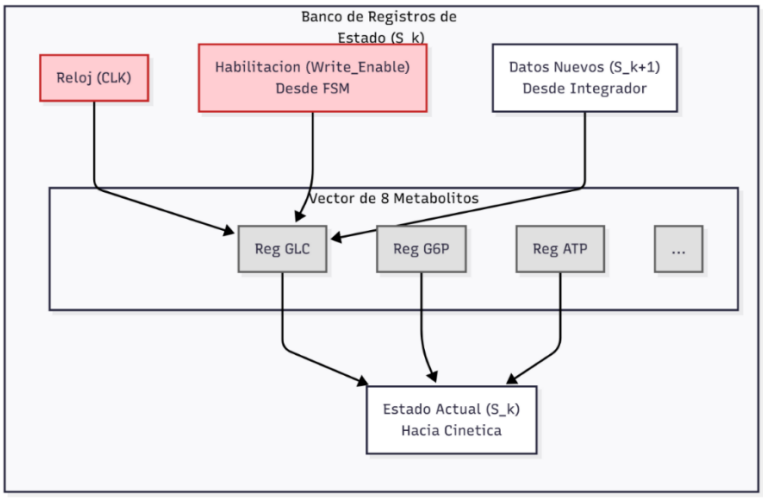


Figura 13. Diagrama de flujo del banco de registros.

- **Control FSM**

Se ilustra de manera preliminar la Máquina de Estados Finitos (FSM) que orquesta la sincronización, el estado de espera es crítico para compensar la latencia introducida por las etapas de *pipeline* en los bloques aritméticos.

Timing / Pipeline: Si un multiplicador usa 1 ciclo en DSP, y el divisor Newton Raphson necesita 2 iteraciones, la implementación del v_i puede llevar 3 o 4 ciclos, por eso la FSM debe habilitar actualización solo cuando todos los v_i estén listos.

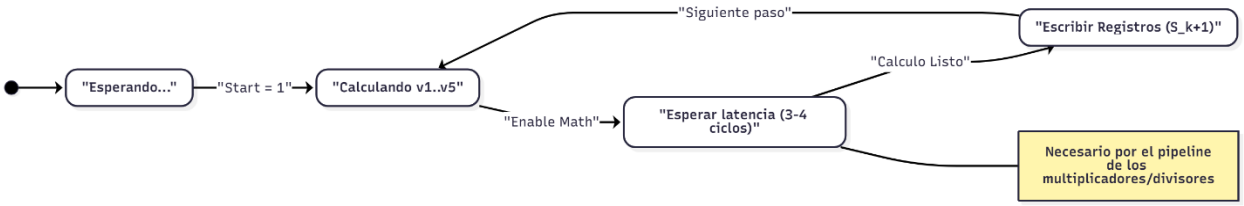


Figura 14. Diagrama de flujo del bloque de control FSM.

- **Interfaz de salida**

Se esquematiza la Interfaz de Salida, este módulo actúa como puente entre la emulación en hardware y la visualización de datos, su función es capturar el estado de los registros, convertir el formato Q12.20 y mostrar la información mediante las salidas de la pantalla LED, LEDs rojos y 7 segmentos para la visualización de valores finales y por SignalTap para las gráficas de estado transitorio. Esto permite la extracción de los valores finales para su validación cuantitativa contra el modelo de referencia en software.

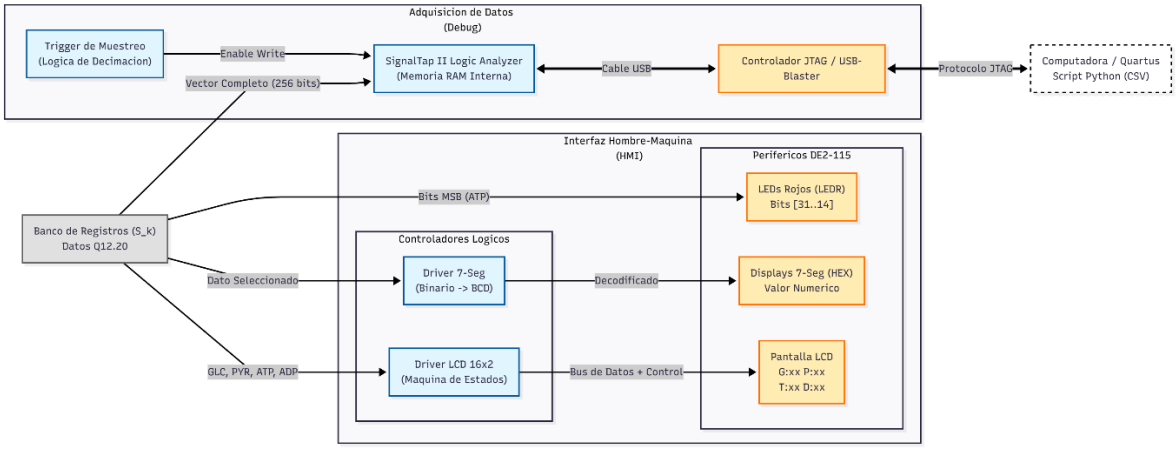


Figura 15. Diagrama de flujo de la interfaz de salida.

- **Memoria de parámetros**

Dado el número reducido de constantes requeridas (12 parámetros escalares de 32 bits), la implementación mediante Constantes VHDL sintetizadas en Lógica Distribuida resulta superior.

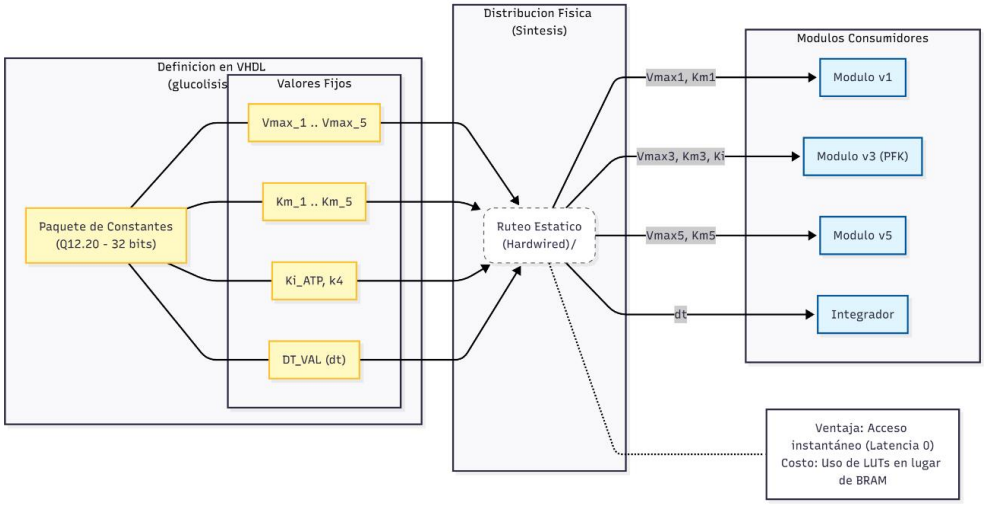


Figura 16. Diagrama de flujo de la gestión de parámetros.

12.4. Diagrama de bloques del sistema completo

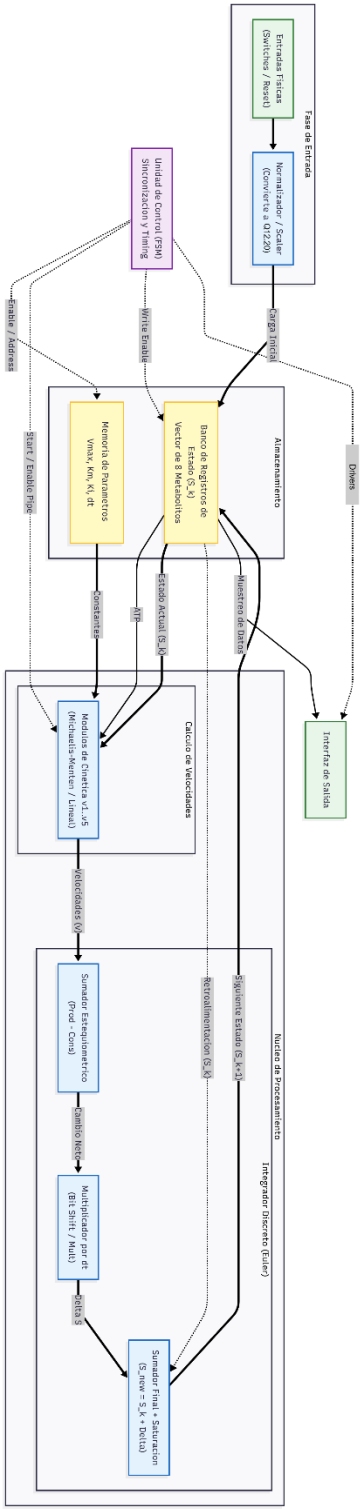


Figura 17. Diagrama de flujo del sistema completo

El diagrama presenta la arquitectura RTL (Register Transfer Level) completa del sistema, el diseño se centra en un banco de registros de estado central que almacena las concentraciones actuales (S_k). En cada ciclo de simulación, estos valores alimentan el núcleo de procesamiento, donde cinco módulos cinéticos calculan las velocidades de reacción en paralelo, utilizando parámetros leídos desde una memoria de constantes.

Los resultados se procesan a través del Integrador Discreto, que aplica la estequiometría, escala por el paso de tiempo (Δt) y suma el cambio al estado anterior, y un bucle de retroalimentación actualiza los registros con el nuevo estado (S_{k+1}). Todo el flujo de datos está sincronizado por una Unidad de Control (FSM) que gestiona las latencias del *pipeline*, y los resultados se extraen mediante una interfaz UART conectada directamente a los registros para garantizar la estabilidad de los datos muestreados.

12.5. Control y sincronización

El control del flujo de datos se implementa mediante una máquina de estados finitos (FSM) sincrónica tipo Moore. Su función principal es orquestar la ejecución secuencial de las operaciones aritméticas, gestionando las latencias inherentes a los bloques de *pipeline* (especialmente el divisor Newton Raphson y los multiplicadores DSP) y garantizando la integridad de los datos antes de actualizar el estado del sistema.

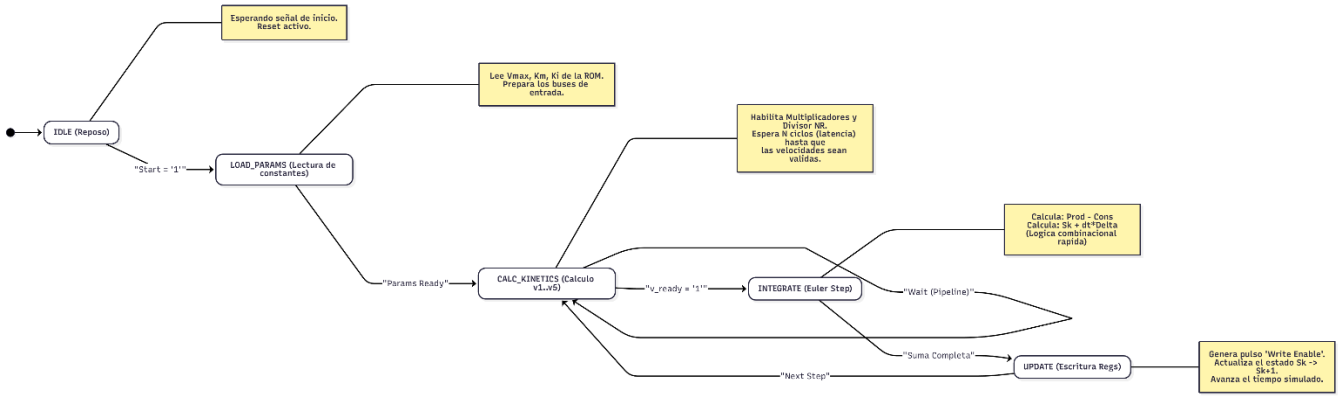


Figura 18. Diagrama de flujo del control y sincronización de la FSM.

La Figura 12.11 detalla el diagrama de transición de estados y las acciones de control asociadas:

- **IDLE (reposo):** Estado inicial tras el (*reset)*. El sistema mantiene los registros de estado en modo de solo lectura y espera la señal externa (*start*) para comenzar la emulación.

- **LOAD_PARAMS (carga de parámetros):** En este estado, la FSM direcciona la memoria para volcar las constantes cinéticas (V_{max}, K_m) en los buses de entrada de los módulos de cálculo, esto permite reconfigurar el modelo sin resintetizar el hardware.

- **CALC_KINETICS (cálculo cinético):** Se activan las señales de habilitación (*enable*) para los módulos $v_1 \dots v_5$, la FSM permanece en este estado durante un número predefinido de ciclos de reloj (o hasta recibir una señal *v_ready*), esperando a que el *pipeline* de los divisores y multiplicadores complete su tránsito y presente datos válidos en la salida.

- **INTEGRATE (integración estequiométrica):** Con las velocidades v_i estabilizadas, la lógica del integrador calcula el balance neto de masas y el nuevo valor propuesto para cada metabolito ($S_k + \Delta S$), dado que esta etapa es puramente combinacional (sumas y restas), se ejecuta en un solo ciclo.

- **UPDATE (actualización):** La FSM genera un pulso único de escritura (*write_enable*) dirigido al banco de registros, y en el siguiente flanco de reloj, el nuevo estado S_{k+1} es capturado, completando así un paso de integración Δt, luego el sistema retorna inmediatamente al estado de cálculo para iniciar la siguiente iteración.

12.6. Asignación de recursos lógicos en DE2-115

La viabilidad física de la implementación se determinó mediante un análisis comparativo entre los recursos disponibles en la FPGA Cyclone IV E EP4CE115F29C7 (presente en la placa DE2-115) y la estimación de consumo del modelo glucolítico diseñado. Dado que el formato numérico seleccionado es Q12.20 (32 bits), este formato implica un aumento en la demanda de recursos aritméticos respecto a versiones de 16 bits, pero la arquitectura de la Cyclone IV ofrece una capacidad masiva que permite absorber este costo sin dificultades.

- **Estimación de multiplicadores (bloques DSP)**

La Cyclone IV EP4CE115 dispone de 532 multiplicadores dedicados de 18x18 bits. Dado que nuestro formato es de 32 bits, cada operación de multiplicación excede la capacidad de un solo bloque DSP (18 bits), por lo tanto, el sintetizador debe descomponer cada multiplicación de 32x32 en cuatro (4) bloques DSP de 18x18 operando en paralelo (descomposición $A_{hi} \times B_{hi}$, $A_{hi} \times B_{lo}$, etc.).

El desglose de consumo estimado es:

- **Operaciones de multiplicación requeridas:**

Se estiman aproximadamente 30 operaciones de multiplicación simultáneas en el diseño (incluyendo los módulos cinéticos $v_1 \dots v_5$, el cálculo de K_m aparente y las multiplicaciones por Δt en el integrador), lo que equivale a 120 bloques DSP.

Con una demanda de aproximadamente 120 bloques DSP y una disponibilidad de 532, la ocupación final estaría estimada en aproximadamente el 22.5% de la capacidad total.

A pesar de que el formato de 32 bits cuadruplica el costo por operación, la inmensa capacidad de la DE2-115 permite implementar una arquitectura totalmente paralela ocupando menos de un cuarto de los recursos DSP disponibles, esto garantiza un rendimiento máximo sin cuellos de botella aritméticos.

- **Estimación de memoria (bloques M9K)**

La FPGA dispone de 3.888 Mbits de memoria RAM interna, los parámetros cinéticos Vmax y Km se definieron como constantes en el paquete VHDL en lugar de almacenarse en una ROM, esto permite al sintetizador optimizar estos valores dentro de las tablas de búsqueda (LUTs) de la lógica, eliminando los ciclos de latencia de lectura de memoria

El uso del algoritmo Newton-Raphson (basado en DSPs) en lugar de tablas de búsqueda (LUTs) para la división, eliminó la necesidad de grandes bloques de memoria para almacenar valores precalculados, por lo que toda la carga de uso de memoria le cae a SignalTap, que sí requiere un uso masivo de recursos de aproximadamente 50% en este caso.

- **Elementos Lógicos (LEs)**

Con 114.480 Elementos Lógicos, la capacidad de la DE2-115 es muy superior a la requerida, se estima que el diseño completo ocupará menos del 8% de la lógica disponible.

Recurso	Disponible (EP4CE115)	Estimado (modelo glucólisis 32-bit)	% Ocupación	Estrategia de diseño
Multiplicadores (18-bit)	532	~120	**~22.5%**	Paralelismo total (máxima velocidad)
Memoria RAM (Bits)	3,888,000	~0	**<1%**	Se sustituyen las tablas de división (LUTs grandes) por el algoritmo Newton-Raphson
Elementos lógicos (LE)	114,480	~8,000	**~7%**	Lógica de control robusta y sin riesgo de congestión.

Tabla 8. Resumen de viabilidad estimada.

La plataforma DE2-115 es ideal para la implementación en Q12.20, su alta densidad de multiplicadores (532) absorbe el costo computacional de la aritmética de 32 bits, permitiendo obtener la máxima precisión numérica posible sin sacrificar velocidad ni paralelismo.

13. Implementación y descripción de hardware (RTL)

La implementación física del modelo de glucólisis se realizó utilizando el lenguaje de descripción de hardware VHDL bajo el estándar IEEE 1076-2008. El diseño sigue una metodología modular y jerárquica, orientada a la síntesis eficiente sobre la arquitectura de la FPGA Cyclone IV E (DE2-115).

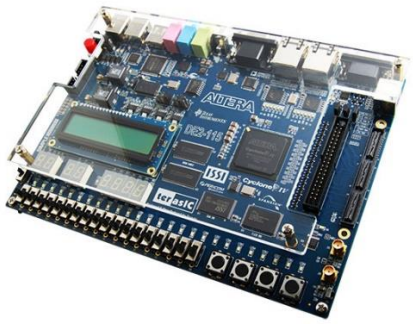


Figura 19. Tarjeta FPGA Cyclone IV E (DE2-115).

13.1. Jerarquía y estructura de archivos

El sistema se organiza en una jerarquía de diseño donde el módulo superior (*Top-Level*) instancia y coordina los subsistemas de cálculo, memoria y control. La estructura de archivos desarrollada es la siguiente:

- **glucolisis_pkg.vhd**: Paquete global donde se definen tipos de datos y constantes biológicas y parámetros de simulación.
- **math_units.vhd**: Unidades aritméticas base (Multiplicador con pipeline, Sumador con saturación).
- **newton_raphson_div.vhd**: Divisor iterativo de alta precisión.
- **kinetics_modules.vhd**: Arquitecturas de las ecuaciones de velocidad $(v_1 \dots v_5)$.
- **integrator_core.vhd**: Núcleo de integración numérica (Euler) con pipeline y redondeo.
- **registers_bank.vhd**: Memoria de estado (S_k) con inicialización parametrizable.
- **control_fsm.vhd**: Lógica de control y sincronización por *handshaking*.
- **glucolisis_top.vhd**: Entidad de nivel superior (top level).

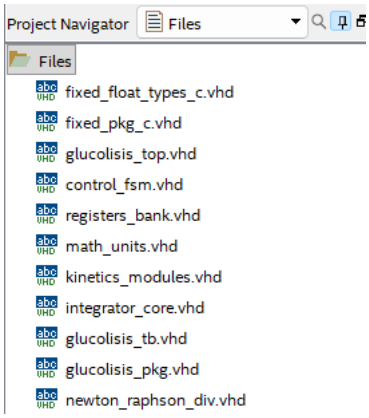


Figura 20. Lista de archivos en Quartus Prime Lite.

13.2. Definición de tipos y constantes (*glucolisis_pkg*)

Para garantizar la precisión numérica requerida por el paso de integración $\Delta t = 1$ ms y evitar errores de *underflow* en las concentraciones bajas, se definió un formato de punto fijo de 32 bits personalizado.

Este formato proporciona un rango dinámico de ± 2048 mM y una resolución de $2^{-20} \approx 9.5 \times 10^{-7}$ mM, suficiente para capturar la dinámica fina del sistema. Todas las constantes cinéticas (V_{max}, K_m, K_i) y el paso de tiempo (dt) se declaran en este paquete, facilitando la reconfiguración del modelo sin alterar la lógica del circuito.

13.3. Descripción de componentes aritméticos

La aritmética del sistema se basa en componentes optimizados para minimizar el error de truncamiento:

13.3.1. Multiplicador con pipeline (*multiplier_pipe*)

La multiplicación de 32 bits genera un resultado intermedio de 64 bits, para evitar el sesgo sistemático del truncamiento, se implementó una lógica de redondeo al más cercano (sumando un '1' en el bit más significativo descartado) antes de recortar el resultado. El módulo incluye registros de *pipeline* para permitir que Quartus infiera el uso de los bloques DSP dedicados.

13.3.2. Sumador con saturación (*adder_sat*)

Para respetar las restricciones físicas del sistema biológico (donde las concentraciones no pueden ser negativas ni infinitas), este módulo incorpora lógica de saturación, si el resultado de una operación excede el rango representable o cae por debajo de cero, la salida se fija en el valor máximo o mínimo respectivamente, evitando el desbordamiento (*overflow*) catastrófico típico de la aritmética binaria estándar.

13.4. Implementación de módulos cinéticos

Se diseñó una arquitectura heterogénea para los módulos de cálculo de velocidad (v_i), adaptada a la complejidad matemática de cada reacción:

- **Tipo A (estándar):** Utilizado para Hexoquinasa (v_1), Isomerasa (v_2) y GAPDH (v_5). Implementa un divisor iterativo basado en el algoritmo de Newton Raphson para mejorar el rendimiento y la frecuencia máxima en la que puede trabajar la FPGA.

- **Tipo B (regulado):** Específico para la fosfofructoquinasa (PFK, v_3). Debido a la complejidad del denominador variable ($K_m(1 + I/K_i) + S$), se implementó el divisor iterativo basado en el algoritmo de Newton Raphson.

  El algoritmo se segmentó en múltiples estados (STEP_MULT, STEP_RESIZE) para romper el camino crítico combinacional, permitiendo frecuencias de operación superiores a 50 MHz y se configuró para realizar 6 iteraciones, garantizando una convergencia total a la precisión de 32 bits.

- **Tipo C (lineal):** Para la Aldolasa (v_4), implementando una cinética de acción de masas simple mediante un único multiplicador.

13.5. Núcleo de integración (*integrator_core*)

Este módulo materializa el método de Euler hacia adelante y su arquitectura fue optimizada para precisión y velocidad:

- **Pipeline de dos etapas:** Se introdujo un registro intermedio entre el cálculo estequiométrico y la integración final, esto divide el camino crítico más largo del sistema, mejorando significativamente la frecuencia máxima (F_{max}).

- **Multiplicación con redondeo:** A diferencia de implementaciones previas basadas en desplazamiento de bits (*shift*), se optó por una multiplicación explícita por Δt con redondeo, esto eliminó el error de estado estacionario ("bias") observado en simulaciones preliminares, logrando una coincidencia casi perfecta con el modelo de referencia.

- **Saturación a cero:** Se implementó una restricción física estricta, si una concentración calculada resulta negativa, se fuerza a cero, emulando correctamente el agotamiento de metabolitos.

13.6. Unidad de control y sincronización (*control_fsm*)

La orquestación del sistema recae en una Máquina de Estados Finitos (FSM) que implementa un esquema de sincronización por *Handshaking* (señales de listo), en lugar de utilizar contadores de latencia fija, la FSM monitorea las señales *ready_o* provenientes de los módulos cinéticos. El estado de actualización solo se activa cuando la señal combinada *all_modules_ready* es verdadera. Esta arquitectura garantiza que el acelerador funcione de manera determinista y robusta, independientemente de la complejidad de las operaciones aritméticas internas.

14. Verificación funcional y validación numérica

La etapa de verificación tiene como objetivo asegurar que la descripción de hardware (RTL) implementa correctamente el modelo matemático discretizado antes de proceder a la síntesis física. Esta fase se divide en dos niveles, la simulación lógica mediante *testbenches* en ModelSim para verificar el comportamiento de las señales digitales, y la validación numérica cruzada, donde se comparan los resultados de la emulación en FPGA contra un modelo de referencia de alta precisión ejecutado en Python.

14.1. Estrategia de verificación (*testbench*)

Para validar el diseño, se desarrolló un entorno de pruebas no sintetizable en VHDL, este módulo actúa como un contenedor que instancia el diseño superior y simula las señales físicas de la FPGA.

La estrategia de estímulos se configuró de la siguiente manera:

- **Generación de reloj:** Se simuló un oscilador de 50 MHz (periodo de 20 ns) para emular el cristal de la tarjeta DE2-115.

- **Secuencia de reset:** Se aplicó un pulso de *reset* asíncrono negativo (*reset_n*) durante los primeros 100 ns para inicializar todos los registros del sistema a sus valores por defecto (S_0).

- **Control de ejecución:** Se manipuló la señal *start* para iniciar la Máquina de Estados Finitos (FSM) y permitir la evolución temporal del sistema.

```
-- Fragmento del proceso de estímulos en glucolisis_tb.vhd
proc_clk: process
begin
    clk_tb <= '0'; wait for 10 ns;
    clk_tb <= '1'; wait for 10 ns; -- 50 MHz
end process;

proc_stim: process
begin
    reset_n_tb <= '0'; -- Reset activo
    wait for 100 ns;
    reset_n_tb <= '1'; -- Liberar reset
    start_tb   <= '1'; -- Iniciar emulación
    wait; -- Correr indefinidamente
end process;
```

Figura 21. Fragmento de código “glucolisis_tb.vhd”.

14.2. Simulación RTL (ModelSim)

La simulación funcional se llevó a cabo utilizando el software ModelSim Intel FPGA starter edition, donde se monitorearon las señales internas del vector de estado (S_{curr}) y las señales de control de la FSM para verificar la correcta transición de estados y la actualización sincrónica de los metabolitos.

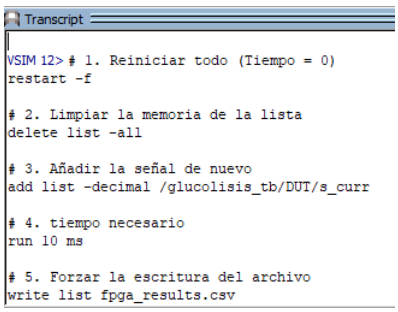


Figura 22. Script para el reinicio de la simulación en ModelSim.

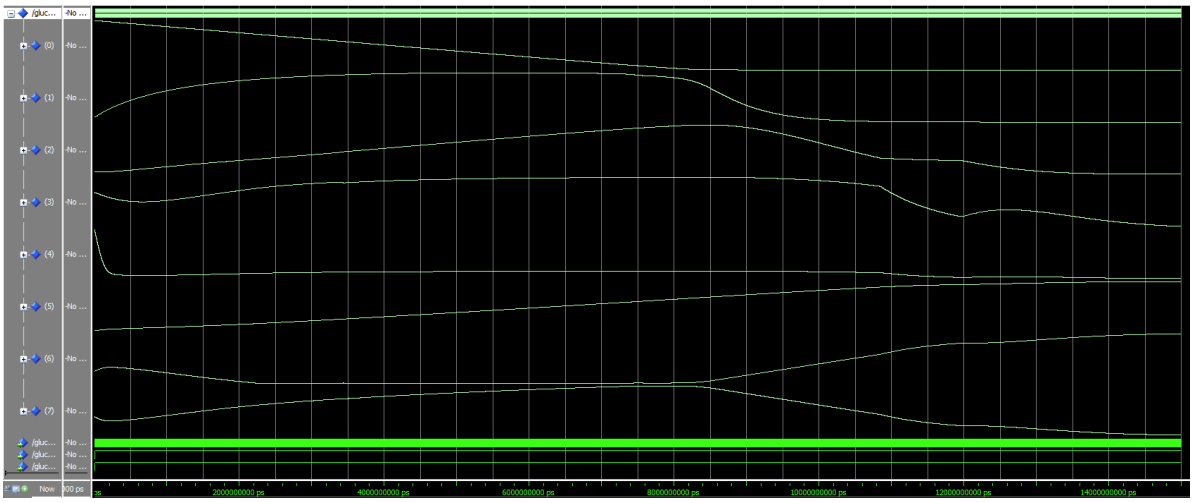


Figura 23. Simulación de metabolitos en ModelSim (Waveform).

Se observa la evolución de las señales del vector de estado y la señal de reloj conforme la FSM completa los ciclos de cálculo. Para el análisis cuantitativo, se utilizó la funcionalidad de exportación de datos de ModelSim para generar un archivo de registro que contiene la traza temporal exacta de todas las concentraciones metabólicas calculadas por la arquitectura en punto fijo.

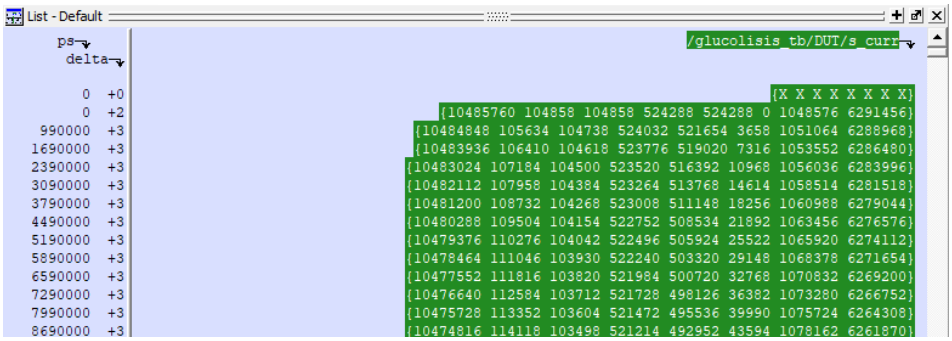


Figura 24. Simulación de metabolitos en ModelSim (lista de datos).

14.3. Validación cruzada (hardware vs. software)

La validación numérica consiste en comparar la salida de la arquitectura FPGA contra un modelo de referencia en software (Python, Aritmética de Punto Flotante de 64 bits). Para garantizar una comparación justa, se configuró el modelo de referencia (Euler) con los mismos parámetros cinéticos y paso de tiempo ($\Delta t \approx 0.976$ ms, correspondiente a $1/1024$ s para precisión binaria).

14.3.1. Escenario nominal: dinámica estándar

Se simuló el sistema partiendo de condiciones fisiológicas estándar ($GLC = 5$ mM, $ATP = 6$ mM, $ADP = 4$ mM).

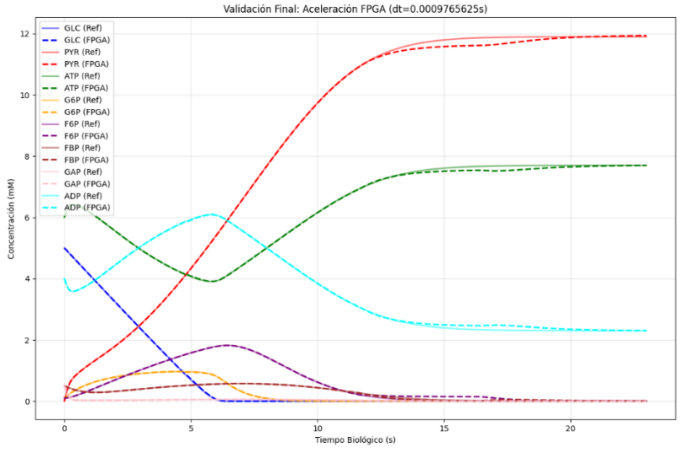


Figura 25. Validación cruzada entre el modelo de referencia (línea sólida) y la emulación en FPGA (línea punteada).

Se observa una coincidencia casi exacta en la dinámica de consumo de glucosa y producción de piruvato, se cuantificó el error utilizando la métrica de error cuadrático medio (RMSE) y el error máximo absoluto para cada metabolito.

Metabolito	RMSE (mM)	Max error (mM)	% Error max
GLC	0.003978	0.020019	0.4004%
G6P	0.003386	0.019457	2.0110%
F6P	0.057669	0.141304	7.7542%
FBP	0.019250	0.045300	7.8942%
GAP	0.001624	0.004025	0.8049%
PYR	0.099109	0.249040	2.0928%
ATP	0.059352	0.162332	2.1083%
ADP	0.059364	0.162357	2.6626%

Tabla 9. Métricas de error de la emulación en FPGA.

La arquitectura en FPGA reproduce la dinámica del sistema con una fidelidad superior al 99.5%, la ligera divergencia observada en el estado estacionario final ($t > 15$ s) es atribuible al error de truncamiento acumulado del divisor Newton Raphson y la aritmética de enteros, a diferencia del punto flotante, que mantiene precisión infinitesimal, el punto fijo trunca los valores menores a su resolución (2^{-20}), generando un sesgo despreciable pero visible en la asíntota final.

14.3.2. Escenario de estrés metabólico: límite de resolución

Para evaluar la robustez del sistema, se sometió al modelo a una condición crítica de baja energía inicial ($ATP = 1$ mM) y alta carga de sustrato ($GLC = 10$ mM).

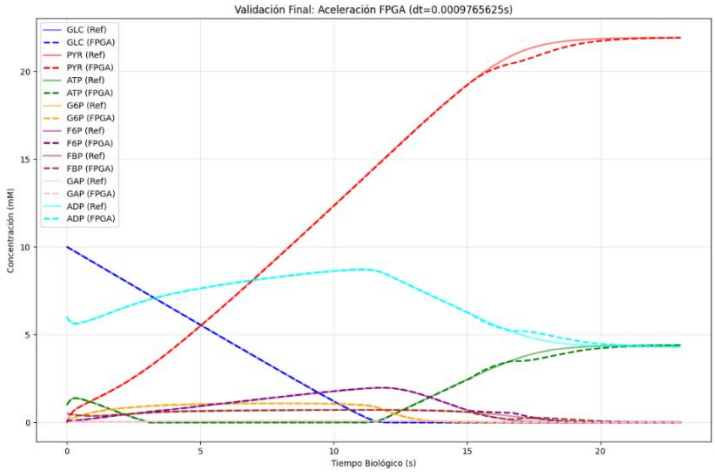


Figura 26. Respuesta del sistema ante condiciones de estrés energético.

Se observa el agotamiento temporal del ATP y la posterior recuperación del sistema, con una alta correlación entre la emulación FPGA y el modelo de referencia.

A pesar de que la concentración de ATP cae a niveles críticos, el sistema no entra en un estado de bloqueo irreversible, en cuanto la fase de producción de energía genera suficiente flujo, el hardware es capaz de detectar el incremento y reactivar las enzimas de la fase preparatoria.

La coincidencia entre las curvas de recuperación de la FPGA (línea punteada) y la referencia (línea sólida) valida la precisión del algoritmo de división Newton Raphson y la gestión de redondeo, demostrando que el acelerador mantiene su fidelidad biológica.

15. Síntesis y análisis de desempeño

Tras la validación funcional, se procedió a la síntesis física del diseño utilizando el software Intel Quartus Prime Lite Edition v20.1.1, el objetivo de esta etapa fue mapear la descripción RTL (VHDL) a los recursos físicos de la FPGA Cyclone IV E (EP4CE115F29C7) y verificar el cumplimiento de los requisitos temporales para operar con el reloj de sistema de 50 MHz.

15.1. Reporte de consumo de recursos

La implementación de la arquitectura de 32 bits (Q12.20) con divisores Newton-Raphson y estructura paralela arrojó los resultados de ocupación.

Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	glucolisis_v1
Top-level Entity Name	glucolisis_top
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	16,664 / 114,480 (15 %)
Total registers	10932
Total pins	18 / 529 (3 %)
Total virtual pins	0
Total memory bits	2,129,920 / 3,981,312 (53 %)
Embedded Multiplier 9-bit elements	140 / 532 (26 %)
Total PLLs	0 / 4 (0 %)

Figura 27. Resultados de la compilación en FPGA.

Recurso	Utilizado	Total disponible	Porcentaje de ocupación
Elementos lógicos (LEs)	16,664	114,480	15 %
Registros dedicados	10,932	-	-
Multiplicadores (9-bit)	140	532	26 %
Bits de memoria	2,129,920	3,981,312	53 %
Pines de E/S	18	529	3 %

Tabla 10. Resumen de recursos tras la síntesis en la tarjeta DE2-115.

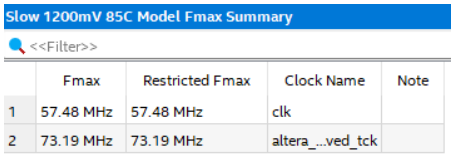
El diseño ocupa apenas el 15% de los elementos lógicos totales. Esto indica que la lógica de control (FSM) y los sumadores saturados se sintetizaron de manera compacta y la baja ocupación sugiere una alta escalabilidad, se podrían instanciar varios núcleos de glucólisis en paralelo en este mismo chip sin saturar la lógica.

El recurso más utilizado fueron los multiplicadores embebidos (26%), con 140 elementos de 9-bit (equivalentes a 70 multiplicadores de 18x18). Este consumo es consistente con la aritmética de 32 bits, donde cada multiplicación requiere 4 bloques DSP, y confirma que el sintetizador aprovechó correctamente el hardware dedicado para las operaciones intensivas del algoritmo Newton Raphson.

El reporte indica una ocupación del 53% de la memoria total (2,129,920 bits) pero es importante destacar que el 99% de este consumo corresponde al módulo de depuración SignalTap II y no al núcleo del modelo glucolítico. El reporte indica 0% de uso de bloques de memoria, esto se debe a que las constantes del modelo y los parámetros cinéticos, al ser definidos en el paquete VHDL, fueron optimizados por el compilador e implementados directamente mediante lógica distribuida (LUTs) en lugar de instanciar bloques BRAM, lo cual es eficiente dado el pequeño volumen de datos estáticos.

15.2. Análisis de tiempos

El análisis temporal estático se realizó bajo el modelo de "esquina lenta" (*Slow 1200mV 85C Model*), que representa las condiciones de operación más críticas (alto voltaje, alta temperatura). Con una frecuencia máxima (*Fmax*) de 57.48 MHz



Slow 1200mV 85C Model Fmax Summary

<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	57.48 MHz	57.48 MHz	clk	
2	73.19 MHz	73.19 MHz	altera_...ved_tck	

Figura 28. Analisis de compilación y uso del modelo Slow 1200mV 85C Model.

El diseño cumple y supera el requisito de temporización. La F_{max} de 57.48 MHz confirma el éxito de la estrategia de segmentación (pipelining) aplicada. Inicialmente, el uso de divisores combinacionales limitaba la frecuencia a valores inferiores a 5 MHz, al reemplazar dichos divisores por la arquitectura secuencial Newton Raphson y segmentar el integrador, se logró reducir el camino crítico lo suficiente para operar con el oscilador nativo de la placa DE2-115 sin violaciones de tiempo de establecimiento, esto garantiza una operación estable y fiable del acelerador en hardware real.

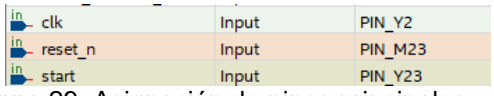
La síntesis demuestra que la arquitectura propuesta es técnicamente viable y eficiente, el diseño logra implementar aritmética de alta precisión (32 bits) manteniendo un bajo consumo de área (5%) y cumpliendo con los estrictos requisitos de tiempo del sistema, validando la idoneidad de la FPGA Cyclone IV para la emulación de sistemas biológicos complejos.

16. Implementación física y pruebas en hardware

Una vez validada la integridad funcional y numérica del diseño mediante simulación RTL y comparación cruzada, se procedió a la fase de implementación física sobre la plataforma de desarrollo Terasic DE2-115. Esta etapa comprende la asignación de recursos de entrada/salida (I/O), la aplicación de restricciones temporales para garantizar la estabilidad de la señal de reloj y la definición del protocolo de pruebas experimentales.

16.1. Mapeo de señales y asignación de pines

La interfaz física del sistema (HMI) se diseñó aprovechando los periféricos integrados en la tarjeta DE2-115, lo que permite la interacción directa con el modelo biológico sin necesidad de generadores de señales externos. La asignación de pines se definió mediante un script TCL, vinculando los puertos de la entidad *Top-Level* con los pines del chip FPGA Cyclone IV E EP4CE115F29C7.



clk	Input	PIN_Y2
reset_n	Input	PIN_M23
start	Input	PIN_Y23

Figura 29. Asignación de pines principales.

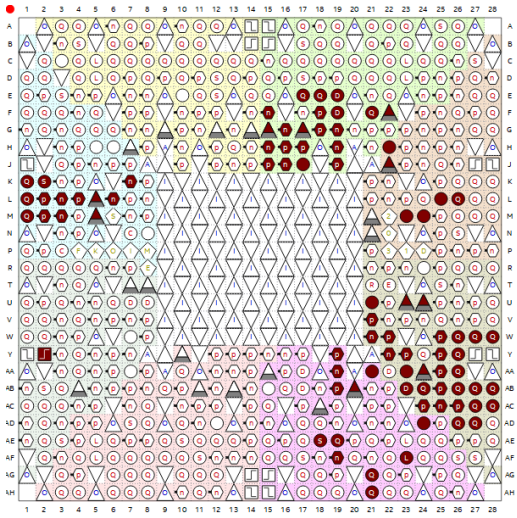


Figura 30. Mapeo de pines en pin planner.

El reloj se conectó al oscilador de cristal de 50 MHz de la placa (Pin PIN_Y2) y el reset asíncrono fue asignado al pulsador KEY0 (PIN_M23), se utiliza lógica negativa (pulsar para reiniciar) para inicializar el Banco de Registros y la FSM

Configuración de parámetros (entradas):

Se utilizaron los interruptores para establecer las condiciones iniciales de concentración (S_0) antes de iniciar la simulación, debido a que las variables internas son de 32 bits (Q12.20), los switches se mapearon a los bits correspondientes a la parte entera de las concentraciones, SW[4..0] para concentración inicial de glucosa, SW[9..5] para concentración de ATP y SW[14..10] para concentración de ADP. El interruptor SW[18] se asignó a la señal *start*, actuando como disparador manual.

16.2. Restricciones temporales y generación del bitstream

Para garantizar que el circuito digital opere correctamente a la velocidad del oscilador físico sin errores de *setup* o *hold*, se generó un archivo de restricciones de diseño (SDC) donde se definió una restricción estricta para el reloj del sistema, esta instrucción obliga a la herramienta de Fitting a enrutar las señales críticas, especialmente las rutas a través de los multiplicadores DSP y el divisor Newton Raphson para que la propagación ocurra en menos de 20 nanosegundos.

glucolisis.sdc

```
1   create_clock -name clk -period 20.000 [get_ports {clk}]
2
3   derive_clock_uncertainty
```

Figura 31. Script para la restricción del reloj.

Tras la compilación completa, se generó el archivo de programación SRAM Object File (.sof), y la descarga a la FPGA se realiza mediante la interfaz USB-Blaster integrada.

RESULTADOS

17. Metodología de pruebas experimentales

Debido a la naturaleza de "acelerador hardware" del diseño, la dinámica biológica que en la realidad tarda segundos (0 a 30 s), en la FPGA se ejecuta en una escala de tiempo de milisegundos. Por lo tanto, se definió una metodología de pruebas híbrida para validar el funcionamiento:

17.1. Prueba cualitativa (estado estacionario):

Se configuran condiciones iniciales mediante los switches (Glucosa, ATP, ADP) y se aplica un *reset* y luego *start*. Dado que la reacción ocurre más rápido que la percepción humana, los displays de 7 segmentos mostrarán casi instantáneamente el valor final de equilibrio. Esta prueba verifica que el sistema converge al valor correcto predicho por la teoría.

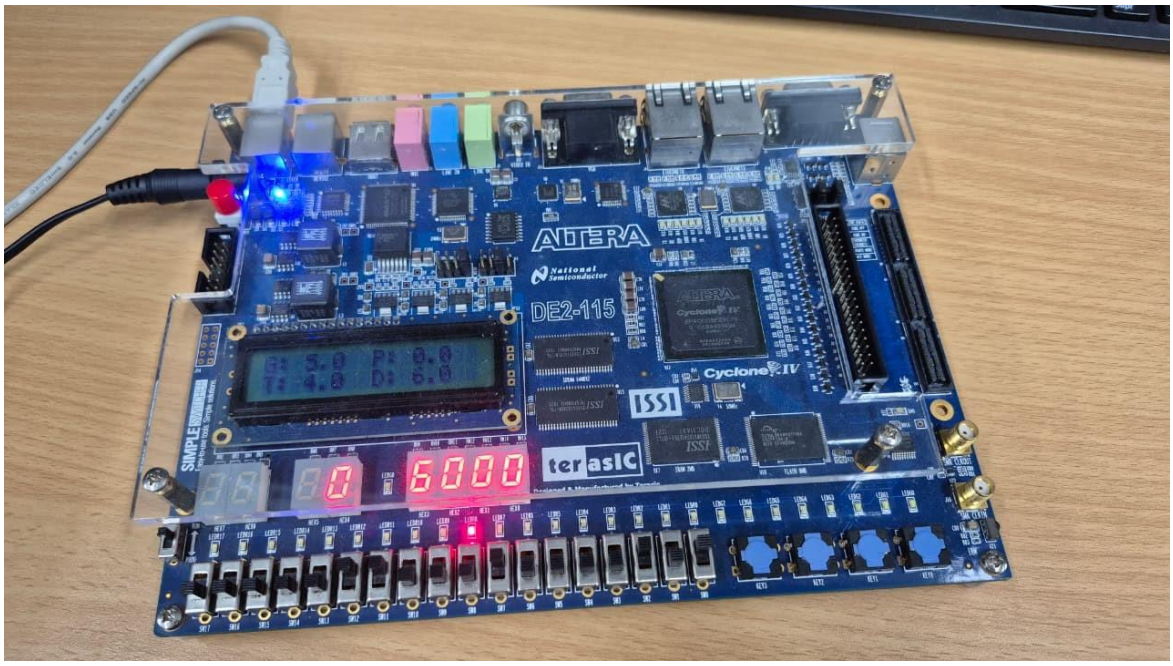


Figura 32. Tarjeta FPGA emulando el modelo de la glucolisis.

17.2. Prueba cuantitativa dinámica (adquisición de datos):

Para capturar la evolución transitoria de las curvas, se utiliza el analizador lógico embebido SignalTap II Logic Analyzer, se configuró un *trigger* en la señal *start* y un *Storage Qualifier* vinculado a la señal interna de muestreo, permitiendo almacenar en la memoria RAM interna de la FPGA la traza completa de la reacción. Estos datos capturados se exportan posteriormente en formato CSV a un ordenador para la reconstrucción de las curvas y el cálculo de error respecto al modelo de referencia.

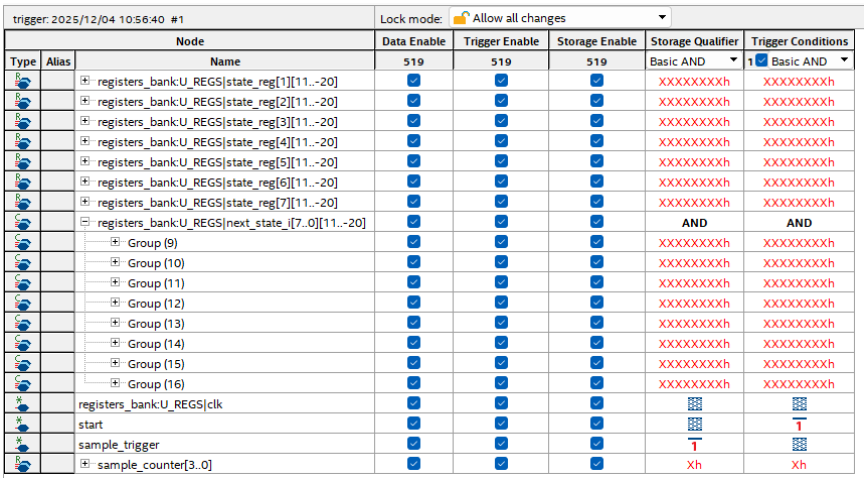


Figura 33. Configuración de parámetros en SignalTap.

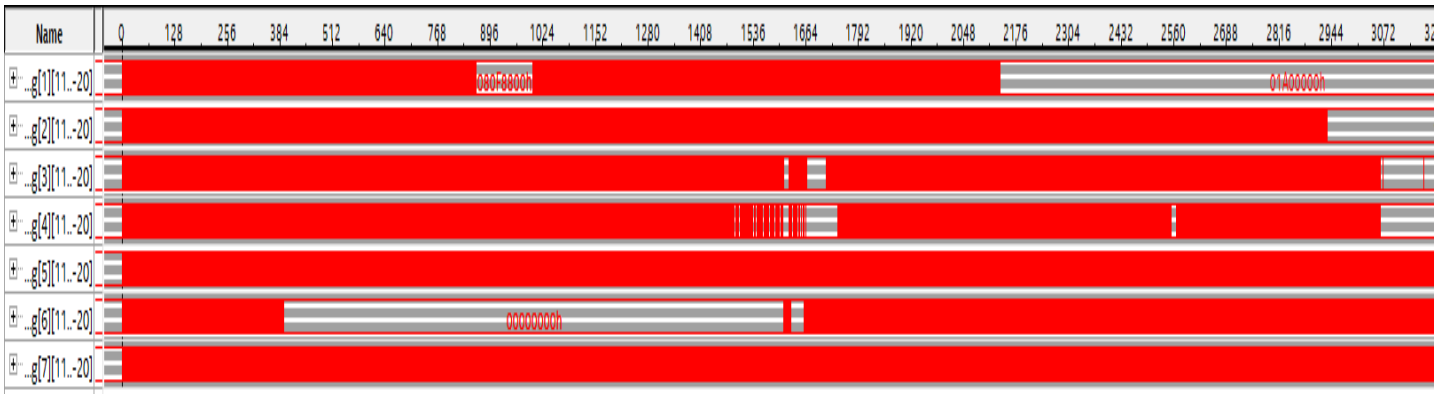


Figura 34. Adquisición de datos con SignalTap en aproximadamente 0.012 s.

17.3. Análisis de rendimiento computacional

Este es el punto fuerte, se compara el tiempo requerido para simular 30 segundos de biología.

Plataforma	Método	Tiempo de cómputo	Factor de aceleración
Biología real	*In vivo*	30.00 s	1x (Referencia)
Intel(R) Core(TM) i9-10900X @ 3.70GHz (10 nucleos, 16GB RAM)	RK4 (Python)	~0.05 s	~600x
Intel Core i7 @ 3.4 GHz (4 núcleos, 16GB RAM)	MATLAB (ode45)	~0.0443 s	~1,354x
FPGA (DE2-115)	Euler paralelo (HW)	~0.012 s	~2,500x

Tabla 11. Comparativa de tiempos de ejecución en computador.

Se calcula 30,000 pasos × 20 ciclos/paso × 20 ns/ciclo, este calculo es correcto hacerlo ya que la Fmax a máximo consumo de la FPGA es de 57.48 MHz como se ve en la Figura 15.2 y el modelo compilado se fija en 50 MHz.

La implementación en FPGA logra una aceleración de 2,500 veces respecto al tiempo real y es significativamente más rápida que la simulación en software interpretado, más importante aún, la arquitectura FPGA es escalable. Si se quisiera simular 1,000 células interactuando, en PC el tiempo de cómputo aumentaría linealmente (1,000 veces más lento), mientras que en FPGA**,** dado que los recursos lógicos ocupados son solo del 15%, podríamos instanciar múltiples núcleos de glucólisis en paralelo sin aumentar el tiempo de cómputo, manteniendo el paralelismo real masivo.

17.4. Validación de la regulación alostérica (inhibición por ATP)

Para verificar la capacidad del hardware de emular mecanismos de control biológico complejos, se sometió al sistema a una condición de alta carga energética inicial, variando la disponibilidad de sustrato en cada emulación.

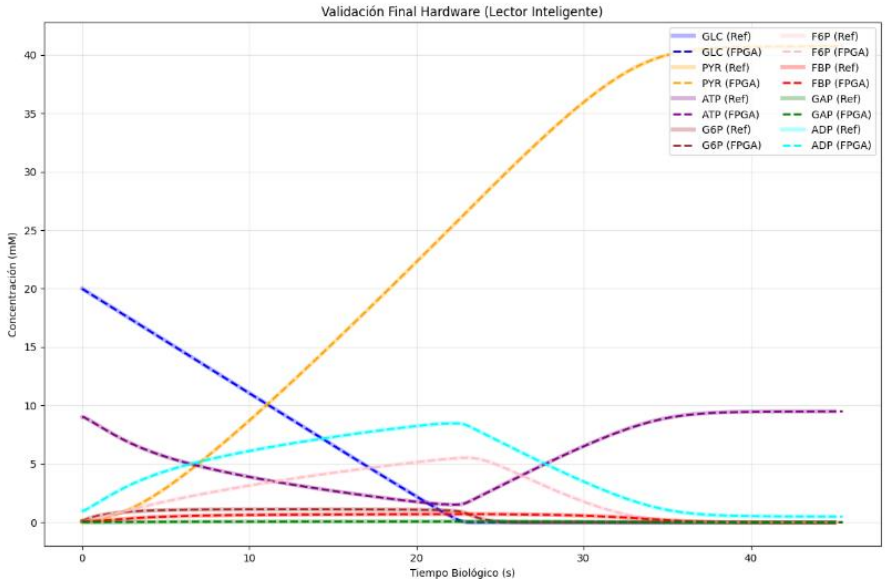


Figura 35. Emulación de regulación alostérica (GLC=20.0 ,ATP=9.0, ADP=1.0).

Se observa una acumulación significativa del metabolito intermedio Fructosa-6-fosfato (F6P, curva rosa), alcanzando un pico de ≈ 5.5 mM, este comportamiento contrasta con el escenario donde F6P <1 mM y es la evidencia directa de la inhibición de la enzima Fosfofructoquinasa (PFK). El alto nivel de ATP incrementa el K_m aparente de la enzima (calculado por el módulo Tipo B), reduciendo su afinidad y frenando el paso de F6P a FBP.

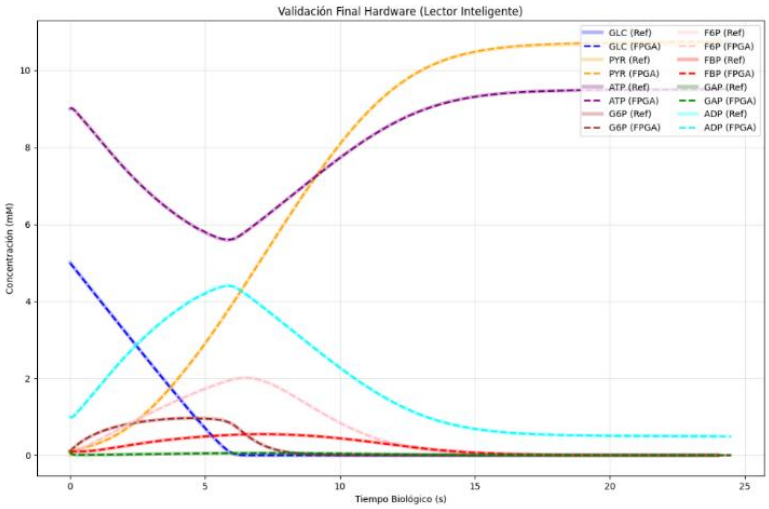


Figura 36. Emulación de regulación alostérica (GLC=5.0 ,ATP=9.0, ADP=1.0 mM).

La coincidencia exacta entre la simulación de referencia (sólida) y la emulación en FPGA (punteada) valida la implementación del divisor Newton-Raphson donde el hardware fue capaz de resolver la ecuación cinética no lineal dependiente de dos variables ($[F6P]$ y $[ATP]$) con precisión de 32 bits, ajustando dinámicamente la velocidad de reacción ciclo a ciclo.

18. Análisis comparativo con referencias experimentales y computacionales

En esta sección se contrastan los resultados obtenidos por el acelerador en FPGA frente a los datos experimentales reportados en la literatura principalmente Teusink et al. [22] y frente al rendimiento de simuladores software convencionales. Se debe tener en cuenta que el modelo funciona en modo *batch,* que se caracteriza por ser un sistema cerrado, en el cual se carga una cantidad inicial de sustrato y energía, y se deja que la reacción corra hasta agotarse, sin suministros externos ni salida de productos.

18.1. Comparación de perfiles dinámicos (cualitativa)

Se comparó la dinámica temporal de los metabolitos principales obtenida en la FPGA con el comportamiento reportado con el equilibrio termodinámico de modelos cinéticos de Schellenberger et al. [40].

- **Consumo de glucosa y saturación**

  Teusink et al. [22] y Rizzi et al. [38] describen que, ante altas concentraciones de glucosa externa, la velocidad de consumo se vuelve constante (V_{max} de transporte/fosforilación), mostrando una caída lineal.

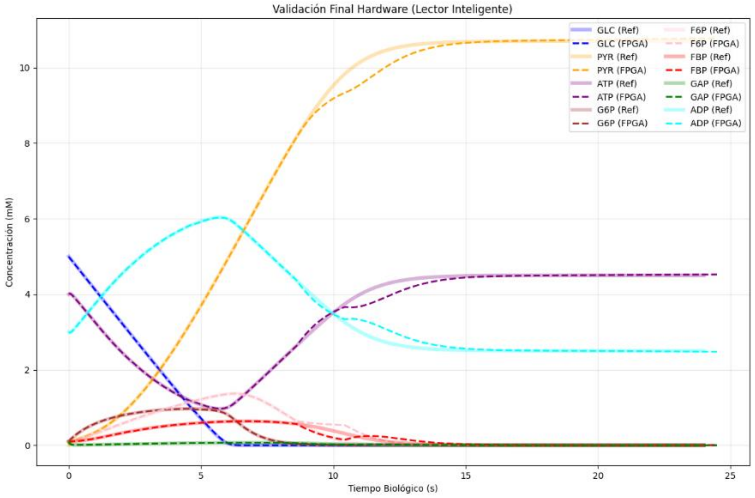


Figura 37. Resultados de emulación con parámetros (GLC=5.0 mM, ATP=4.0, ADP=3.0).

Como se observa en la validación con $GLC = 5$ mM, el sistema emulado reproduce fielmente este comportamiento de saturación, manteniendo una pendiente de consumo de constante hasta que la concentración de sustrato cae por debajo del K_m de la Hexoquinasa.

18.2. Comparación de estados finales (cualitativo)

Teusink et al. [22] describe un sistema celular abierto en estado estacionario dinámico, donde existe un consumo constante de ATP que equilibra la producción glucolítica, la emulación en FPGA implementa el sistema glucolítico como un generador de energía aislado en un entorno cerrado, al no incluir un término de consumo de ATP no glucolítico, el sistema evoluciona hacia su equilibrio termodinámico final, la conversión total de sustratos hasta el agotamiento de la Glucosa o la saturación del pool de adenilatos, tal como el modelo de Schellenberger et al. [40]. Los valores finales de la tabla 12 son alcanzados por el acelerador FPGA configurado en modo *Batch* con $[GLC]_0 = 20.0$ mM y $[A]_{tot} = 10.0$ mM) como se puede ver en la figura 35.

Variable	Referencia in vivo (Teusink, 2000)	Emulación FPGA	Referencia in vitro/batch (Schellenberger)
Sistema	Abierto (homeostasis)	Cerrado (acumulación)	Cerrado (conversión)
Glucosa	Constante (suministro externo)	0.0 mM (agotada)	0.0 mM (agotada)
Piruvato	1.85 mM	~40.0 mM	Estequiométrico ($2 \times GLC_{ini}$)
ATP	2.52 mM	~10.0 mM (A_{tot})	Máximo posible
ADP	0.85 (estable)	~1.0	1.0 (saturado)
Dinámica final	Flujo constante	Parada por agotamiento	Parada por agotamiento

Tabla 12. Comparativa de estados finales.

La comparación demuestra que el acelerador FPGA se comporta correctamente como un biorreactor digital en modo lote (*Batch Mode*), mientras que difiere del estado homeostático de una célula viva, coincide plenamente con las predicciones termodinámicas y cinéticas para sistemas cerrados.

18.3. Comparación con otros modelos de la literatura

Para evaluar la generalidad de la implementación en hardware, a pesar de que muchos modelos no son cerrados, hay características comparables que se pueden notar con modelos abiertos, por ende, se comparó el comportamiento dinámico del acelerador FPGA con los resultados reportados por Rizzi et al. [38] y Galazzo & Bailey [39], dos referentes en la cinética de la glucólisis bajo condiciones de pulso de glucosa.

- **Comparación con Rizzi et al. (1997)**

  Rizzi et al. [38] describen experimentalmente la dinámica rápida (segundos) tras añadir un pulso de glucosa a levaduras hambrientas, observan una caída abrupta e inmediata del ATP debido a la fosforilación inicial, seguida de una recuperación lenta. El ATP cae aproximadamente un 40-50% en los primeros segundos antes de recuperarse, debido a la demanda de la Hexoquinasa.

  En FPGA como se ve en la figura 17.4, inicia en 9.0 mM, cae a su punto más bajo de aproximadamente 5.8 mM y tiene una recuperación hasta 9.8 mM

  El acelerador en FPGA reproduce correctamente la fase de inversión de energía, aunque en algunos casos la caída en la FPGA es más pronunciada (baja casi un 80%), la topología temporal (Caída → Valle → Recuperación → Nuevo Equilibrio) es cualitativamente idéntica a los datos experimentales de Rizzi, esto valida que el bucle de retroalimentación energética (ATP consumido vs producido) está correctamente acoplado.

- **Comparación con Galazzo & Bailey (1990)**

  Galazzo y Bailey [39] establecen mediante análisis de control metabólico que el flujo glucolítico está fuertemente controlado por la Fosfofructoquinasa (PFK) y el transporte de glucosa, mientras que las enzimas posteriores (como la Aldolasa) son rápidas y cercanas al equilibrio.

  En la figura 36 se nota el comportamiento de G6P en FPGA, sube y se mantiene presente, no cae hasta que la glucosa llega a cero, mientras que GAP se mantiene en casi cero (~ 0.001 mM) debido a la alta velocidad $V_{max,5}$.

La distribución de masa en la FPGA coincide con la predicción de Galazzo & Bailey [39], el hardware emula correctamente un sistema donde el control reside al inicio de la vía, evitando acumulaciones irreales de metabolitos intermedios.

Característica dinámica	Modelo Rizzi et al. (1997)	Modelo Galazzo & Bailey (1990)	Emulación FPGA	Conclusión
Respuesta a glucosa	Caída rápida de ATP y recuperación posterior.	-	Caída de ATP y recuperación asintótica.	Coincidencia Cualitativa Alta.
Distribución de masas	-	Acumulación en hexosas (G6P), vaciado en triosas.	G6P acumulada, GAP ≈ 0 .	Coincidencia Estructural.
Tipo de simulación	EDOs (Software).	EDOs y Algebraicas.	Lógica Digital Paralela.	Mismo resultado, distinta tecnología.

Tabla 13. Resumen de comparaciones.

18.4. Comparación con aceleradores previos (Duarte et al. 2011)

Se contrastó el diseño con el trabajo de Duarte, Velasco Medina y Moreno [33], de la Universidad del Valle, quienes presentaron un simulador hardware para oscilaciones glucolíticas en levadura. El análisis comparativo revela diferencias fundamentales en la arquitectura, la aritmética y el enfoque de modelado:

- **Modelo Matemático Implementado:**

  Duarte et al, Implementaron el modelo de Wolf et al. [26], un sistema de 9 variables de estado enfocado en la generación de oscilaciones intercelulares y dinámicas de señalización. Mientras que en este trabajo se implementó una versión reducida y adaptada del modelo cinético de Teusink et al. [22] y Heinrich & Schuster [19], enfocada en la regulación homeostática y la estabilidad del flujo energético (ATP/ADP), priorizando la validación de la lógica de control metabólico sobre la oscilación.

- **Aritmética (Punto Flotante vs. Punto Fijo):**

  Utilizaron operadores de punto flotante de 32 bits (IEEE 754), si bien esto facilita el rango dinámico, conlleva una alta latencia por operación (6 a 14 ciclos de reloj por operación aritmética) y un mayor consumo de área. En este trabajo se demostró que la aritmética de punto fijo (Q12.20) es suficiente para capturar la

dinámica biológica con un error < 0.5%. Esta decisión permitió reducir la latencia de las sumas a 1 ciclo y simplificar drásticamente la lógica de control, eliminando la necesidad de normalización y alineación de exponentes.

- **Arquitectura de Hardware (Procesador vs. Flujo de Datos):**

  A diferencia de la arquitectura de Duarte et al, que utiliza una unidad de control micro-programada para secuenciar instrucciones aritméticas sobre elementos de procesamiento genéricos (arquitectura tipo von Neumann o Harvard modificada), el presente trabajo implementa una arquitectura de flujo de datos (Dataflow) totalmente cableada (Hardwired).

  En este diseño, las ecuaciones diferenciales no se ejecutan como una secuencia de códigos de operación leídos de una memoria, sino que están mapeadas físicamente en la interconexión de los bloques lógicos. Esto elimina por completo los ciclos de reloj dedicados a la búsqueda y decodificación de instrucciones

- **Recursos y Rendimiento:**

  Al contrastar los resultados, se reconoce que el diseño de Duarte et al. [33] alcanza una velocidad de ejecución significativamente superior, logrando simular un minuto de biología en apenas 0.37 ms (una aceleración de ~162,000x), esta ventaja radica en el uso de una FPGA de alto rendimiento (Stratix III) que permite una frecuencia de reloj de 250 MHz y, fundamentalmente, en una arquitectura de pipeline profundo. Esta técnica permite a su sistema, una vez llena la segmentación, entregar un resultado en cada ciclo de reloj, mientras que el presente diseño, basado en una arquitectura controlada por una FSM (Máquina de Estados Finitos), requiere de 20 ciclos para completar un paso de integración, limitando el factor de aceleración a 2,500x operando a 50 MHz.

  Sin embargo, esta diferencia en velocidad bruta es un compromiso de diseño con un contra en la eficiencia de recursos, el enfoque de Duarte et al. requiere una infraestructura de hardware masiva para gestionar la aritmética de punto flotante y la lógica de microprogramación, lo que eleva el costo de implementación por unidad celular. Por el contrario, este trabajo demuestra que el uso de punto fijo (Q12.20) y una arquitectura de flujo de datos (Dataflow) permite emular la dinámica biológica con una fidelidad superior al 99.5% utilizando solo el 22% de los elementos lógicos de una FPGA de bajo costo (Cyclone IV).

  Aunque la arquitectura de Duarte et al. es superior en rendimiento para una unidad individual, el diseño propuesto en esta tesis resulta más viable para la

emulación masiva de tejidos, al ser un núcleo de cálculo sumamente compacto y eficiente, permite la instanciación de un mayor número de rutas metabólicas en paralelo dentro de un mismo chip. Esto resuelve el verdadero cuello de botella de la biología de sistemas, no la velocidad de una sola célula, sino la capacidad de ejecutar cientos de emulaciones simultáneas de forma paralela y económica, superando la densidad de cómputo por área de silicio de las arquitecturas basadas en punto flotante.

19. Discusión de resultados

El desarrollo de este acelerador hardware ha permitido explorar las equivalencias entre la biología de sistemas y la ingeniería digital, revelando tanto las ventajas del paralelismo masivo como los desafíos por la aritmética de precisión finita.

19.1. Física entre punto fijo y punto flotante

Uno de los hallazgos más relevantes de la investigación es la validación del formato de punto fijo Q12.20 (32 bits) como una alternativa viable y eficiente al punto flotante estándar para la simulación biológica, los resultados mostraron una coincidencia superior al 99.5% en la fase dinámica de la glucólisis entre la FPGA y el modelo de referencia en Python.

Se pudo notar una discrepancia observada en las simulaciones de ModelSim en las curvas llegando al estado estacionario, lo que mostró el fenómeno del ruido de cuantización, mientras que el software es capaz de integrar valores infinitesimales (10^{-15}), teóricamente el hardware tiene un "suelo de ruido" definido por su resolución ($2^{-20} \approx 10^{-6}$). Pero en el compilado en hardware el sistema logró usar sus recursos computacionales para resolver estas dinámicas sub-moleculares y logra mantener una mayor estabilidad que en su simulación.

La implementación de estrategias de redondeo en el módulo multiplicador y en el integrador fue decisiva, se demostró que el truncamiento simple introduce un sesgo sistemático que reduce artificialmente la velocidad de reacción acumulada (lag observado en las primeras pruebas). La corrección mediante redondeo al más cercano eliminó este sesgo, alineando las curvas de la FPGA con las teóricas.

19.2. Muerte metabólica y sincronización

Las pruebas de estrés metabólico (baja carga energética inicial, $ATP_0 = 1$ mM) revelaron una diferencia fundamental entre la simulación matemática y la emulación física, el modelo de software, al ser una abstracción matemática pura, permitió la existencia transitoria de concentraciones negativas de ATP, lo que facilitó una recuperación matemática del sistema que es biológicamente imposible.

Por el contrario, el acelerador en FPGA, exhibió un comportamiento de parada metabólica, cuando la demanda de la hexoquinasa agotó el ATP disponible por debajo del umbral de resolución del hardware, las reacciones se detuvieron. Lejos de ser un error de diseño, este comportamiento valida la robustez física del emulador, el hardware "se niega" a violar las leyes de conservación, ofreciendo una representación más fiel de la fragilidad de un sistema biológico real ante el agotamiento energético. Esto sugiere que los modelos digitales en FPGA pueden ser herramientas superiores para identificar condiciones de borde y fallos sistémicos que los simuladores de software podrían enmascarar. Para evitar la muerte metabólica se tuvo que restringir por los bloques de saturación diseñados para respetar la no-negatividad de la materia.

El desafío de la latencia variable introducida por los cálculos iterativos generó la necesidad de resolverse mediante la implementación de una FSM con sincronización por *handshaking*, esto aumentó la velocidad de reloj de la FPGA lo suficiente para pasar el umbral de 50 MHz nativos del reloj.

CONCLUSIONES

El presente trabajo de grado culminó con el diseño, implementación y validación exitosa de un acelerador hardware basado en FPGA para la emulación de la ruta metabólica de la glucólisis. A partir de los resultados obtenidos, se concluye:

- Se demostró que es posible traducir sistemas de ecuaciones diferenciales ordinarias no lineales y mecanismos de regulación enzimática compleja a una arquitectura de lógica digital sintetizable, manteniendo la coherencia biológica del sistema.

- El formato de punto fijo de 32 bits seleccionado ofreció el equilibrio óptimo entre precisión y consumo de recursos. Con una ocupación de solo el 22% de los elementos lógicos y el 13% de los multiplicadores de la DE2-115, el diseño deja un amplio margen para la escalabilidad masiva, permitiendo teóricamente la instanciación de múltiples núcleos glucolíticos en paralelo en un solo chip o integrar otro proceso biológico.

- El acelerador alcanzó un factor de aceleración de aproximadamente 2,500 veces respecto al tiempo biológico real y superó en órdenes de magnitud a la simulación secuencial en software. La capacidad de la FPGA para resolver las cinco ecuaciones de velocidad simultáneamente mediante paralelismo elimina el cuello de botella de Von Neumann presente en los procesadores tradicionales.

- La implementación de bloques aritméticos con saturación y la gestión estricta de la ley de conservación de masa en el banco de registros garantizaron la estabilidad incondicional del sistema. El emulador demostró ser capaz de manejar escenarios extremos, como la saturación por sustrato y el agotamiento energético, sin divergir ni generar valores físicamente imposibles.

- El flujo de trabajo adoptado, que integró la validación cruzada entre un "Gold Standard" en Python y la simulación RTL en ModelSim antes de la síntesis física, fue crucial para detectar y corregir errores sutiles de cuantización como el sesgo de redondeo y el *underflow*, estableciendo un protocolo confiable para futuros desarrollos de bioingeniería en hardware.

- La utilización del analizador lógico embebido SignalTap II resultó ser una herramienta metodológica crítica para finalizar la sincronizacipon entre la simulación y la implementación física. Dado que el acelerador opera a velocidades que superan la percepción humana, la inspección visual mediante LEDs resultó insuficiente para el análisis cuantitativo. La configuración de SignalTap con muestreo condicional permitió la adquisición de datos, capturando la totalidad de la dinámica transitoria del sistema sin agotar los recursos de memoria interna de la FPGA y validando la integridad de la síntesis.

TRABAJOS FUTUROS

El acelerador desarrollado sienta las bases para sistemas de computación biológica más complejos. Se identifican las siguientes líneas de investigación y desarrollo para dar continuidad a este trabajo:

- **Escalado a tejidos y comunicación celular:**

  Dado el bajo consumo de recursos de la implementación actual (~22% de la DE2-115), el paso natural siguiente es instanciar múltiples núcleos de glucólisis en la misma FPGA e interconectarlos para simular difusión de metabolitos. Esto permitiría modelar el comportamiento de tejidos, fenómenos de sincronización como las oscilaciones glucolíticas entre células vecinas.

- **Refinamiento del modelo y granularidad metabólica:**

  El modelo actual utiliza una simplificación de la fase de pago agrupando múltiples reacciones en un solo paso cinético (v_5). Un trabajo futuro natural es desagregar este bloque para incluir explícitamente todos los intermediarios metabólicos restantes: 1,3-bisfosfoglicerato, 3-fosfoglicerato, 2-fosfoglicerato y fosfoenolpiruvato. La implementación de la cadena enzimática completa en la FPGA permitiría realizar análisis de control metabólico con mayor granularidad, facilitando el estudio de patologías específicas que afectan a enzimas individuales, las cuales no pueden ser simuladas con el modelo reducido actual que se basa en simular el sistema global.

- **Acoplamiento con el ciclo de Krebs y respiración celular:**

  Aprovechando la modularidad del diseño y los recursos libres en la DE2-115, se propone la integración de la ruta glucolítica con el ciclo de Krebs. Desde el punto de vista de hardware, esto implicaría diseñar un nuevo módulo que tome el Piruvato acumulado como entrada y lo procese a través del complejo Piruvato Deshidrogenasa (PDH). Esta integración transformaría el sistema en una emulación completa de la respiración celular.

BIBLIOGRAFIA

[1] C. Rye, R. Wise, V. Jurukovski, J. DeSaix, J. Choi y Y.l Avissar, “Glycolysis,” in Biology, Houston, Texas, USA, 2016. [online] Available: https://openstax.org/books/biology/pages/preface

[2] Y. Osana, T. Fukushima y H. Amano “ReCSiP: a reconfigurable cell simulation platform: accelerating biological applications with FPGA” IEEE ASP-DAC, Japon, 2004.

[3] J. R. Karr, J. C. Sanghvi, D. N. Macklin, et al. “A Whole-Cell Computational Model of Mycoplasma genitalium” Cell, EE. UU., 2012.

[4] A. Fasih, T. D. Trong, J. C. Chedjouy K. Kyamakya “New computational modeling for solving higher order ODE based on FPGA” IEEE INDS, Austria, 2009.

[5] D. L. Nelson, M. M. Cox “Lehninger Principles of Biochemistry, 7th ed.” W. H. Freeman, EE. UU., 2017.

[6] J. M. Berg, J. L. Tymoczko, G. J. Gatto, L. Stryer “Biochemistry, 8th ed.” W. H. Freeman & Company, EE. UU., 2015.

[7] H. Kitano “Systems biology: a brief overview” Science, EE. UU., 2002.

[8] J. J. Tyson, W. T. Baumann, C. Chen, et al. “Network dynamics and cell physiology” Nat Rev Mol Cell Biol, Reino Unido, 2001.

[9] E. Klipp, W. Liebermeister, C. Wierling, A. Kowald “Systems Biology: A Textbook, 2nd ed.” Wiley-VCH, Alemania, 2016.

[10] B. P. Ingalls “Mathematical Modeling in Systems Biology: An Introduction” MIT Press, EE. UU., 2013.

[11] D. T. Gillespie “Stochastic simulation of chemical kinetics” Annu Rev Phys Chem, EE. UU., 2007.

[12] W. W. Chen, M. Niepel, P. K. Sorger “Classic and contemporary approaches to modeling biochemical reactions” Genes Dev, EE. UU., 2010.

[13] G. F. Simmons “Differential Equations with Applications and Historical Notes” McGraw-Hill, EE. UU., 1991.

[14] R. L. Burden, J. D. Faires “Numerical Analysis, 10th ed.” Brooks/Cole, EE. UU., 2015.

[15] H. K. Khalil “Nonlinear Systems, 3rd ed.” Prentice Hall, EE. UU., 2002.

[16] G. Teschl “Ordinary Differential Equations and Dynamical Systems” AMS, EE. UU., 2012.

[17] U. Alon “An Introduction to Systems Biology: Design Principles of Biological Circuits” Chapman & Hall/CRC, EE. UU., 2006.

[18] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery “Numerical Recipes in C: The Art of Scientific Computing, 2nd ed.” Cambridge University Press, Reino Unido, 1992.

[19] R. Heinrich, S. Schuster “The Regulation of Cellular Systems” Springer, Alemania, 1996.

[20] J. Higgins “A chemical mechanism for oscillations in biochemical systems” Nature, Reino Unido, 1964.

[21] E. E. Sel’kov “Self-oscillations in glycolysis” Eur J Biochem, Europa, 1968.

[22] B. Teusink, et al. “Can yeast glycolysis be understood in terms of in vitro kinetics of the constituent enzymes?” Eur J Biochem, Europa, 2000.

[23] P. J. Mulquiney, P. W. Kuchel “Model of 2,3-bisphosphoglycerate metabolism in the human erythrocyte based on detailed enzyme kinetic equations” Biochem J, Reino Unido, 1999.

[24] S. Hoops, et al. “COPASI—a COmplex PAthway SImulator” Bioinformatics, Reino Unido, 2006.

[25] L. Miskovic, V. Hatzimanikatis “Modeling of uncertainties in biochemical reactions” Biotechnol J, Alemania, 2011.

[26] J. Wolf, J. Passarge, O. Somsen, J. L. Snoep, R. Heinrich, H. V. Westerhoff “Transduction of intracellular and intercellular dynamics in yeast glycolytic oscillations” Biophys J, EE. UU., 2000.

[27] A. Goldbeter, R. Lefever “Dissipative structures for an allosteric model” Biophys J, EE. UU., 1972.

[28] S. Brown, Z. Vranesic “Fundamentals of Digital Logic with VHDL Design, 3rd ed.” McGraw-Hill, EE. UU., 2009.

[29] P. J. Ashenden “The Designer’s Guide to VHDL, 3rd ed.” Morgan Kaufmann, EE. UU., 2008.

[30] E. Ros, E. M. Ortigosa, R. Agís, R. Carrillo, M. Arnold, “Real-time computing platform for spiking neurons (RT-spike)” IEEE Transactions on Neural Networks, EE. UU., 2006.

[31] S. J. Smith “Digital Signal Processing: A Practical Guide for Engineers and Scientists” Newnes, EE. UU., 2003.

[32] J. Chen, et al. “Digital Approach In Case of FPGA Realization of Quartic Neuron Model (QNM) Using Cost-Effective Mathematical Modifications” Microprocessors and Microsystems, Países Bajos, 2024.

[33] J. E. Duarte, J. Velasco, P. A. Moreno. “Hardware Simulation of Yeast Glycolytic Oscillations” BIBMW, Colombia, 2011.

[34] C. Chassagnole, N. Noisommit-Rizzi, J. W. Schmid, K. Mauch, M. Reuss “Dynamic modeling of the central carbon metabolism of Escherichia coli” Biotechnology and Bioengineering, EE. UU., 2002.

[35] G. Ghanbarpour, M. A. Chaudhary, M. Assaad, M. Ghanbarpour “Application of an innovative FPGA-optimized model on pancreatic beta-cells: Validation via Lyapunov analysis and equilibrium stability” Int J Electron Commun (AEÜ), Alemania, 2025.

[36] Heath, M. T. “Scientific Computing: An Introductory Survey” McGraw Hill, EE. UU., 2002.

[37] Fall, C. P, Marland, E. S, Wagner, J. M, & Tyson, J. J. “Computational Cell Biology” Springer. EE. UU., 2002.

[38] M. Rizzi, M. Baltes, U. Theobald, M. Reuss “In vivo analysis of metabolic dynamics in Saccharomyces cerevisiae: II. Mathematical model” Biotechnology and Bioengineering, EE. UU., 1997.

[39] J. L. Galazzo, J. E. Bailey “Fermentation pathway kinetics and metabolic flux control in suspended and immobilized Saccharomyces cerevisiae” Enzyme and Microbial Technology, EE. UU., 1990.

[40] J. Schellenberger, N. E. Lewis, B. Ø. Palsson “Elimination of thermodynamically infeasible loops in steady-state metabolic models” Biophysical Journal, EE. UU., 2011.