

MODELADO EN HARDWARE DE SISTEMAS BIOLÓGICOS E IMPLEMENTACIÓN CON DIFERENTES MÉTODOS NUMÉRICOS

LORENA MEJÍA DUQUE

**Proyecto de grado presentado como
Requisito parcial para optar al título de
Ingeniero Electrónico.**

**UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
PROGRAMA ACADÉMICO DE INGENIERÍA ELECTRÓNICA
SANTIAGO DE CALI**

2012

**MODELADO EN HARDWARE DE SISTEMAS BIOLÓGICOS E
IMPLEMENTACIÓN CON DIFERENTES MÉTODOS NUMÉRICOS**

LORENA MEJÍA DUQUE

DIRECTOR

JAIME VELASCO MEDINA, Ph. D.

ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

UNIVERSIDAD DEL VALLE

UNIVERSIDAD DEL VALLE

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

PROGRAMA ACADÉMICO DE INGENIERÍA ELECTRÓNICA

SANTIAGO DE CALI

2012

Nota de aceptación

Jaime Velasco Medina. Ph.D

Director de Tesis

Jurado

Jurado

Tabla de contenido

CAPÍTULO 1	14
INTRIDUCCIÓN	14
1.1. MOTIVACIÓN	15
1.2. CONTRIBUCIÓN	16
1.3. ORGANIZACIÓN.....	17
CAPÍTULO 2	18
CONCEPTOS BÁSICOS DE BIOLOGÍA DE SISTEMAS	18
2.1. CINÉTICA ENZIMÁTICA.....	19
2.2. SISTEMAS BIOLÓGICOS	21
2.3. EL RITMO CIRCADIANO MOLECULAR.....	22
2.4. FLUJO DE CA^{2+} EN LAS CÉLULAS DEL PÁNCREAS.....	24
2.4.1. CA^{2+} - ON	24
2.4.2. Transducción de señal.....	25
2.4.3. Inositoltrifosfato (InsP3).....	25
2.5. EL OSCILADOR MITÓTICO	26
CAPÍTULO 3	28
SIMULACIÓN FUNCINAL DE: EL RITMO CIRCADIANO, EL FLUJO DE CA^{2+} EN LAS CÉLULAS DEL PÁNCREAS Y EL OSCILADOR MITÓTICO	28
3.1. MODELO DEL RITMO CIRCADIANO	28
3.1.1. Simulación usando el biosimulador CellDesigner	30
3.1.2. Simulación usando Simulink de Matlab	31
3.1. MODELO DEL FLUJO DE CA^{2+} EN LAS CÉLULAS DEL PÁNCREAS.....	32
3.1.1. Simulación usando el biosimulador CellDesigner	34
3.1.2. Simulación usando Simulink de Matlab	34
3.1. MODELO DEL OSCILADOR MITÓTICO.....	36
3.1.1. Simulación usando el biosimulador CellDesigner	38
3.1.2. Simulación usando Simulink de Matlab	39

CAPÍTULO 4	41
IMPLEMENTACIÓN EN HARDWARE DE: EL RITMO CIRCADIANO, EL FLUJO DE CA^{2+} EN LAS CÉLULAS DEL PANCREAS Y EL OSCILADOR MITÓTICO	41
4.1. IMPLEMENTACIÓN EN HARDWARE DEL MODELO DEL RITMO CIRCADIANO	44
4.1.1. Implementación en hardware del ritmo circadiano usando el método de Euler	45
4.1.2. Implementación en hardware del ritmo circadiano usando la regla trapezoidal.....	47
4.1.3. Implementación en hardware del ritmo circadiano usando la regla de Simpson	49
4.2. IMPLEMENTACIÓN EN HARDWARE DEL MODELO DEL FLUJO DE CA^{2+} EN LAS CÉLULAS DEL PÁNCREAS	51
4.1.1. Implementación en hardware del flujo de CA^{2+} usando el método de Euler	52
4.1.2. Implementación en hardware del flujo de CA^{2+} usando la regla trapezoidal	54
4.1.3. Implementación en hardware del flujo de CA^{2+} usando la regla de Simpson.....	55
4.1. IMPLEMENTACIÓN EN HARDWARE DEL MODELO DEL OSCILADOR MITÓTICO.....	57
4.1.1. Implementación en hardware del oscilador mitótico usando el método de Euler	57
4.1.2. Implementación en hardware del oscilador mitótico usando la regla trapezoidal	59
4.1.3. Implementación en hardware del oscilador mitótico usando la regla de Simpson.....	61
CAPÍTULO 5	64
ANÁLISIS Y DISCUSIÓN DE RESULTADOS	64
5.1. ANÁLISIS Y RESULTADOS PARA EL RITMO CIRCADIANO	64
5.2. ANÁLISIS Y RESULTADOS PARA EL OSCILADOR MITÓTICO.....	71
5.3. ANÁLISIS Y RESULTADOS PARA EL FLUJO DE CA^{2+} EN LAS CÉLULAS DEL PÁNCREAS.....	78
CAPÍTULO 6	88
CONCLUSIONES Y TRABAJO FUTURO	88
BIBLIOGRAFÍA	90
ANEXOS.....	93
ANEXO A	
Diagrama de tiempos para el modelo del ritmo circadiano usando en método de Euler	93
ANEXO B	
Código en VHDL de la FSM implementada para el modelo del ritmo circadiano usando el método de Euler	103
ANEXO C	

Test-Bench para la implementación en hardware del modelo del ritmo circadiano usando el método de Euler	109
ANEXO D	
Macro en Excel para conversión de binario a hexadecimal y de hexadecimal a decimal	117
ANEXO E	
Código en Visual Basic para conversión de binario a hexadecimal y de hexadecimal a binario	118
ANEXO F	
Código en VHDL de la FSM implementada para el modelo del ritmo circadiano usando la regla trapezoidal	121
ANEXO G	
Código en VHDL de las FSM implementada para el modelo del ritmo circadiano usando la regla de Simpson	126

INDICE DE FIGURAS

Figura 2.1: <i>Modelo simple para un ritmo circadiano.</i>	23
Figura 2.2: <i>Modelo mínimo en cascada para el oscilador mitótico.</i>	27
Figura 3.1: <i>Pathway en Celldesigner para el ritmo circadiano molecular.</i>	29
Figura 3.2: <i>Gráfica de concentraciones para los metabolitos P y M del ritmo circadiano molecular implementado en Celldesigner.</i>	30
Figura 3.3: <i>Diagrama de bloques para el modelo del ritmo circadiano basado en la dimerización y proteólisis del PER y TIM implementado en Simulink.</i>	31
Figura 3.4: <i>Grafica de resultado para el modelo del ritmo circadiano implementado en Simulink de Matlab.</i>	32
Figura 3.5: <i>Pathway del flujo del CA de DuPont implementado en CellDesigner.</i>	32
Figura 3.6: <i>Gráfica de concentraciones para los metabolitos Z y Y del flujo de CA en CellDesigner.</i>	34
Figura 3.7: <i>Diagrama de bloques para el modelo del oscilador de CA de Dupont implementado en Simulink.</i>	35
Figura 3.8: <i>Grafica de resultado para el modelo del flujo CA sistema simulado en implementado en Simulink.</i>	36
Figura 3.9: <i>Pathway para el oscilador mitótico de Goldebeter implementado en CellDesigner</i>	36
Figura 3.10: <i>Gráfica de resultado para el modelo del oscilador mitótico simulado en CellDesinger</i>	38
Figura 3.11: <i>Diagrama de bloques para el modelo del Oscilador mitótico implementado en Simulink.</i>	39
Figura 3.12: <i>Gráfica de concentraciones para los metabolitos del oscilador mitótico implementado en Simulink.</i>	40
Figura 4.1: <i>Modelo de los diagramas de tiempos implementados para el diseño de los sistemas biológicos en hardware.</i>	41
Figura 4.2: <i>Diagrama de bloques para implementación en hardware de los</i>	

<i>modelos matemáticos de los sistemas biológicos usando el método de Euler.</i>	42
Figura 4.3: <i>Diagrama de bloques para implementación en hardware de los modelos matemáticos de los sistemas biológicos usando la regla del trapecio.</i>	43
Figura 4.4: <i>Diagrama de bloques para implementación en hardware de los modelos matemáticos de los sistemas biológicos usando la regla de Simpson.</i>	44
Figura 4.5: <i>Datapath para las implementaciones en hardware del sistema “modelo del ritmo circadiano basado en la dimerización y proteólisis del PER y TIM”.</i>	44
Figura 4.6: <i>FSM para la implementación del modelo del ritmo circadiano usando el método de Euler.</i>	45
Figura 4.7: <i>Diagrama de estados para la implementación del modelo del ritmo circadiano usando el método de Euler.</i>	46
Figura 4.8: <i>Gráfica de concentraciones de los metabolitos para el modelo del ritmo circadiano implementado en hardware usando el método de Euler.</i>	47
Figura 4.9: <i>FSM para el modelo matemático ritmo circadiano usando la regla trapezoidal y la regla de Simpson.</i>	47
Figura 4.10: <i>Diagrama de estados para el modelo del ritmo circadiano usando la regla trapezoidal.</i>	48
Figura 4.11: <i>Gráfica de resultados para la implementación en hardware del ritmo circadiano usando la regla trapezoidal.</i>	49
Figura 4.12: <i>Diagrama de estados del modelo del ritmo circadiano usando la Regla de Simpson.</i>	50
Figura 4.13: <i>Gráfica de resultados de la implementación para la implementación del ritmo circadiano usando la regla de Simpson.</i>	51
Figura 4.14: <i>Datapath para las implementaciones en hardware del sistema “Modelado de Ca^{2+} flujo en las células del páncreas: papel de la membrana plasmática y las reservas intracelulares”.</i>	51
Figura 4.15: <i>FSM para el modelo de flujo de CA usando el método de Euler.</i>	52
Figura 4.16: <i>Diagrama de estados para el modelo del flujo de CA según el método de Euler.</i>	53
Figura 4.17: <i>Gráfica de concentraciones para la implementación en hardware del</i>	

<i>flujo CA usando el método de Euler.</i>	53
<i>Figura 4.18: FSM para las implementaciones en hardware del modelo del flujo de CA usando la regla trapezoidal y la regla de Simpson.</i>	54
<i>Figura 4.19: Diagrama de estados para el modelo del flujo de CA usando la Regla trapezoidal.</i>	54
<i>Figura 4.20: Gráfica de concentraciones para la implementación en hardware del flujo de CA flujo usando la regla trapezoidal.</i>	55
<i>Figura 4.21: Diagrama de estados para la implementación en hardware del modelo del flujo de CA usando la regla de Simpson.</i>	56
<i>Figura 4.22: Gráfica de concentraciones para la implementación en hardware del modelo del flujo CA usando la regla de Simpson.</i>	56
<i>Figura 4.23: Datapath para las implementaciones en hardware del sistema oscilador mitótico participación de ciclina y cdc2 quinasa.</i>	57
<i>Figura 4.24: FSM para la implementación en hardware del modelo del oscilador mitótico usando el método de Euler.</i>	57
<i>Figura 4.25: Diagrama de estados para el modelo del oscilador mitótico usando el método de Euler.</i>	58
<i>Figura 4.26: Gráfica de concentraciones para la implementación hardware del oscilador mitótico usando el método de Euler.</i>	58
<i>Figura 4.27: FSM para las implementaciones en hardware del modelo del flujo de CA usando la regla trapezoidal y la regla de Simpson.</i>	59
<i>Figura 4.28: Diagrama de estados para modelo del oscilador mitótico usando la regla trapezoidal.</i>	60
<i>Figura 4.29: Gráfica de concentraciones para la implementación en hardware del oscilador mitótico usando la regla trapezoidal.</i>	61
<i>Figura 4.30: Diagrama de estados para el modelo del oscilador mitótico usando la regla de Simpson.</i>	62
<i>Figura 4.31: Gráfica de concentraciones para la implementación hardware del oscilador mitótico usando la regla de Simpson.</i>	63
<i>Figura 5.1: Resultados de las concentraciones para los metabolitos del ritmo circadiano brindada por la base de datos biológica Biocompare database.</i>	65

Figura. 5.2: Gráfica de de concentraciones para los metabolitos P y M implementado en hardware y software usando el Método de Euler.	66
Figura 5.3: Gráfica del error absoluto para las concentraciones de los metabolitos del del ritmo circadiano implementado en hardware usando el Método de Euler.	67
Figura 5.4: Gráfica de concentraciones para P y M del ritmo circadiano implementado en software y hardware usando el método trapezoidal.	67
Figura 5.5: Gráfica del error absoluto para las concentraciones de metabolitos del ritmo circadiano implementado en hardware usando el método trapezoidal	68
Figura 5.6: Gráfica de concentraciones para P y M del ritmo circadiano implementado en software y hardware usando la regla de Simpson.	69
Figura 5.7: Gráfica del error absoluto para las concentraciones de metabolitos del ritmo circadiano implementado en hardware usando la regla de Simpson.	69
Figura 5.8: Diagramas de barras comparativo de la eficiencia de las implementaciones hardware según los tres métodos numéricos versus la implementación software del ritmo circadiano.	70
Figura 5.9: Resultados de las concentraciones para los metabolitos del oscilador mitótico brindada por la base de datos biológica.	71
Figura 5.10: Gráfica de de concentraciones para los metabolitos C, M y X del oscilador mitótico implementado en hardware y software usando el Método de Euler.	72
Figura 5.11: Gráfica del error absoluto para las concentraciones de los metabolitos del oscilador mitótico implementado en hardware usando el Método de Euler.	73
Figura 5.12: Gráfica de concentraciones para C, M y X oscilador mitótico implementado en software y hardware usando el método trapezoidal.	73
Figura 5.13: Gráfica del error absoluto para las concentraciones de los metabolitos del oscilador mitótico implementado en hardware usando el método trapezoidal	74
Figura 5.14: Gráfica de concentraciones para C, M y X oscilador mitótico implementado en software y hardware usando la regla de Simpson con un tiempo de muestreo de 0,005.	75

Figura 5.15: <i>Gráfica del error absoluto para las concentraciones de los metabolitos del oscilador mitótico implementado en hardware usando la regla de Simpson con un tiempo de muestreo de 0,005.</i>	75
Figura 5.16: <i>Diagramas de barras comparativo para las implementaciones hardware usando el método de Euler y la regla trapezoidal.</i>	77
Figura 5.17: <i>Diagramas de barras comparativo de la eficiencia de las implementaciones hardware usando la regla de Simpson versus implementaciones software.</i>	78
Figura 5.18: <i>Gráfica de concentraciones para el metabolito Z del flujo CA de Dupont brindado por la base de datos biológica.</i>	79
Figura 5.19: <i>Gráfica de concentraciones para los metabolitos Y y Z del flujo de CA implementado en software y hardware usando el método de Euler.</i>	80
Figura 5.20: <i>Gráfica del error absoluto para las concentraciones de los metabolitos Z y Y del flujo de CA implementado en hardware usando el método de Euler.</i>	80
Figura 5.21: <i>Gráfica de concentraciones para los metabolitos Y y Z del flujo de CA implementado en software y hardware usando la regla trapezoidal.</i>	81
Figura 5.22: <i>Gráfica de los errores absolutos para las concentraciones de los metabolitos Y y Z del flujo de CA implementado en hardware usando la regla trapezoidal.</i>	82
Figura 5.23: <i>Gráfica de concentraciones para los metabolitos Y y Z del flujo de CA implementado en software y hardware usando la regla Simpson.</i>	82
Figura 5.24: <i>Gráfica de los errores absolutos para las concentraciones de los metabolitos Y y Z del flujo de CA implementado en hardware usando la regla de Simpson.</i>	83
Figura 5.25: <i>Gráfica de concentraciones para los metabolitos Y y Z del flujo de CA implementado en software y hardware usando el método de Euler con un tiempo de muestreo de 0,001.</i>	84
Figura 5.26: <i>Gráfica de los errores absolutos para las concentraciones de los metabolitos Y y Z del flujo de CA implementado en hardware usando el método de Euler con un tiempo de muestreo de 0,001.</i>	84

Figura 5.27: *Diagramas de barras comparativo de la eficiencia de la implementaciones hardware del flujo de CA usando el método de Euler, regla trapezoidal y regla de Simpson versus implementación software.*

86

Figura 5.28: *Diagramas de barras comparativo de la eficiencia de la implementación hardware del flujo de CA usando el método de Euler versus implementación software con un tiempo de muestreo de 0,001.*

87

LISTA DE TABLAS

Tabla 3.1: <i>Tabla de parámetros para el modelo del ritmo circadiano molecular</i>	30
Tabla 3.2: <i>Tabla de parámetros para el modelo matemático del flujo de CA de Dupont.</i>	33
Tabla 3.3: <i>Tabla de parámetros para el modelo matemático del oscilador mitótico.</i>	38
Tabla 5.1: <i>Tabla de valores máximos, mínimos, promedios y amplitudes para las implementaciones hardware del ritmo circadiano usando los tres métodos numéricos y la implementación software.</i>	70
Tabla 5.2: <i>Tabla de valores máximos, mínimos, promedios y amplitudes para las implementaciones hardware del oscilador mitótico usando el método de Euler, regla trapezoidal y Simulink.</i>	73
Tabla 5.3: <i>Tabla de valores máximos, mínimos, promedios y amplitudes para las implementaciones hardware del oscilador mitótico usando la regla de Simpson y la implementación software con un tiempo de muestreo de 0,005</i>	77
Tabla 5.4: <i>Tabla de valores máximos, mínimos, promedios y amplitudes para los resultado obtenidos en las implementaciones del flujo de CA usando el método de Euler, la regla trapezoidal y Simulink.</i>	85
Tabla 5.5: <i>Tabla de valores máximos, mínimos, promedios y amplitudes para los resultado obtenidos en las implementaciones del flujo de CA usando el método de Euler y Simulink con un tiempo de muestreo de 0,001.</i>	86

CAPÍTULO 1

INTRODUCCIÓN

La biología de sistemas se basa en el modelado de los sistemas biológicos, se enfoca en las interacciones de los componentes del sistema y cómo estos dan lugar a una función que describa su comportamiento.

Los sistemas biológicos desde el punto de vista de la biología no se describen cuantitativamente (usando modelos matemáticos), pero si se caracterizan por seguir algunos principios como la continuidad, la evolución, la diversidad, la homeostasis y la universalidad.

Por otro lado, la biosimulación (simulación de sistemas biológicos) emplea las matemáticas para representar cuantitativamente la dinámica de los sistemas biológicos, los cuales son complejos debido a que cambian en el tiempo y a que están conformados por diferentes elementos interconectados entre sí. Debido a su complejidad, la biosimulación consume muchos recursos computacionales y mucho tiempo de procesamiento. En estas simulaciones, el tiempo es proporcional a la complejidad y la precisión de los resultados sistema. Por ello es necesario tener en cuenta alternativas como la implementación en hardware para mitigar los problemas anteriores.

1.1 MOTIVACIÓN

Según afirmó en el 2003 la organización BIO (Biotechnology Industry Organization), la biotecnología es el uso de procesos biológicos para resolver problemas o hacer productos útiles. Esto se logra a partir del conocimiento de los procesos biológicos que permite determinar los sistemas y las características de las moléculas para encontrar vías (formas, maneras, procedimientos) para la modificación de los mismos con el propósito de lograr productos de aplicación industrial o médica. A continuación se proponen tres casos:

1- La apoptosis es el conjunto de procesos que desarrolla la célula para su propia muerte/desaparición en condiciones patológicas o normales. Estos procesos se pueden activar en algunas situaciones para eliminar del organismo poblaciones celulares indeseables, como es el caso de las células tumorales.

2- La caracterización de las vías de señalización intracelular y el movimiento celular de los receptores celulares permitiría desarrollar productos que inhiban la migración celular y por lo tanto la diseminación de las células tumorales.

3- Los productos inhibidores (hormonas, fitoalexinas, etc.) del crecimiento de insectos parásitos producidos por algunas plantas como mecanismo de defensa puede ser utilizado para aumentar la resistencia de los vegetales a los agentes patógenos si se incorpora esa información al genoma del vegetal.

En este contexto, con el propósito de disminuir los altos costos de experimentación en la investigación biotecnológica se ha desarrollado software especializado en simular los sistemas y procesos biológicos, es decir la biosimulación. Sin embargo, la biosimulación de procesos biológicos complejos consume muchos recursos computacionales y mucho tiempo. Entonces con el propósito de mitigar los inconvenientes anteriores esta tesis presenta la simulación de procesos biológicos basada en hardware, es decir la simulación biohardware.

1.2 CONTRIBUCIÓN

En este proyecto de grado se implementaron en hardware tres modelos de sistemas biológicos:

- El ritmo circadiano basado en la dimerización y proteólisis del PER y TIM en la *Drosophila melanogaster*.
- El flujo de Ca^{2+} en las células del páncreas: papel de la membrana plasmática y las reservas intracelulares.
- El oscilador mitótico, participación de ciclina y cdc2 quinasa

En este caso cada modelo fue implementado en hardware usando los métodos numéricos Euler, regla trapezoidal y regla de Simpson. Los resultados obtenidos en las implementaciones hardware son analizados y comparados con los resultados obtenidos en las simulaciones software, del análisis de los resultados se obtiene que los tiempos de ejecución en las implementaciones hardware son menores que los de su contraparte software.

1.3 ORGANIZACIÓN

Este trabajo de grado está organizado de la siguiente manera:

En el capítulo II se describe el modelo de Michaelis-Menten para cinética enzimática y los conceptos básicos sobre biología de sistemas y sistemas biológicos. También se presenta el funcionamiento de tres sistemas biológicos: ritmo circadiano molecular, flujo del Ca^{2+} en las células del páncreas y el oscilador mitótico.

El capítulo III presenta los resultados de las simulaciones de los modelos matemáticos del ritmo circadiano basado en la dimerización y proteólisis del PER y TIM, flujo del Ca^{2+} propuesto por Dupont y el oscilador mitótico propuesto por Goldbeter; usando el biosimulador CellDesigner y Simulink de Matlab.

En el capítulo IV se presenta las implementaciones en hardware del ritmo circadiano molecular, flujo del CA en la célula del páncreas de Dupont y el oscilador mitótico de Goldbeter usando los diferentes métodos numéricos (Euler, regla trapezoidal y regla de Simpson) y la herramienta Modelsim de Altera.

Por último en el capítulo V se presentan la comparación entre los resultados obtenidos en software con respecto a los resultados suministrados por la base de datos biológica BIOMODELS DATABASE, también se presenta la comparación y el análisis de los resultados obtenidos desde los modelos en software frente a los resultados obtenidos con las implementaciones hardware.

CAPÍTULO 2

CONCEPTOS BASICOS SOBRE BIOLOGÍA DE SISTEMAS

La biología tradicional se centra en la identificación y estudio de funciones de genes individuales, proteínas y células. Este camino nos proporciona información muy limitada del funcionamiento de los sistemas vivos. Sin embargo, en los últimos años se han realizado avances científicos en biología molecular, bioinformática, ingeniería y física entre otras disciplinas, las cuales han permitido acceder a información más detallada y compleja sobre los sistemas biológicos.

En este contexto, la biología de sistemas tiene como objetivo integrar la información de las disciplinas mencionadas para conseguir un mejor entendimiento de las interacciones entre los componentes de los seres vivos y, por lo tanto, de sus procesos biológicos. Para alcanzar este objetivo se desarrollan modelos matemáticos, simulaciones y técnicas que complementan la estrategia empírica de las ciencias biológicas.

Las principales características de la biología de sistemas son:

- Estudiar los sistemas biológicos de una forma global, a nivel molecular.
- Integrar el conocimiento de diferentes disciplinas (genómica, transcriptómica, proteómica, metabolómica, fisiología, patología, etc.).
- Manejar un gran repertorio de datos originados en estudios experimentales.
- Proponer modelos matemáticos que puedan explicar los fenómenos biológicos.

- Implementar soluciones matemáticas que permitan obtener predicciones para los procesos biológicos.
- Analizar y comprobar la calidad de los modelos descritos por medio de comparaciones entre las simulaciones numéricas y los datos experimentales.

El análisis teórico en la biología de sistemas utiliza el lenguaje matemático para describir los sistemas biológicos. Las principales herramientas teóricas que se utilizan en la biología de sistemas son las mismas que se utilizan en física para el estudio de los sistemas dinámicos complejos (ecuaciones diferenciales, análisis de bifurcaciones, análisis de balances de flujos, etc.).

Entre las herramientas matemáticas utilizada con mayor frecuencia en la biología de sistemas para estudiar la dinámica de un sistema están las ecuaciones cinéticas planteadas a partir de un modelo cinético determinado. Por ejemplo:

En un sistema ocurren una serie de reacciones químicas en las cuales las velocidades de reacción representan flujos y las afinidades químicas representan las fuerzas. Se conoce explícitamente como están relacionados dichos flujos con las fuerzas, de manera que se puede establecer una serie de ecuaciones cinéticas, en las cuales se considera la variación con el tiempo de las distintas sustancias que intervienen en la reacción en función de sus concentraciones y de las constantes cinéticas de las distintas reacciones, así como de las restantes restricciones que se hayan impuesto en el sistema. Gracias a estos modelos se pueden realizar simulaciones de la respuesta dinámica de los sistemas biológicos frente a diferentes estímulos o perturbaciones del sistema.

2.1. CINÉTICA ENZIMÁTICA

Una de las características de la célula activa es la capacidad para desarrollar reacciones complejas muy rápidamente. Los principales agentes que participan en las transformaciones celulares son las enzimas. Una enzima es una proteína sintetizada por la célula que cataliza (acelera) una reacción termodinámica. La enzima no modifica la constante de equilibrio y como son proteínas hay agentes capaces de desnaturalizar estos compuestos (como el calor, los ácidos, las bases fuertes, los solventes orgánicos, etc.) haciéndoles perder sus propiedades catalíticas.

Las enzimas catalizan solamente una reacción o un grupo limitado de ellas, debido a esto se necesitan miles de ellas en un proceso biológico. Por ello el estudio de la química y la regulación enzimática es un requisito esencial para comprender el mecanismo del crecimiento y reproducción celular.

La velocidad de una reacción catalizada enzimáticamente depende en forma directa de la concentración de la enzima. Un aumento de sustrato (compuesto que se está transformando)

manteniendo fija la concentración de enzima produce al principio un incremento en la velocidad de reacción; a medida que se va aumentando la concentración del sustrato, la velocidad de la reacción empieza a decrecer.

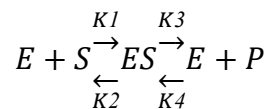
La ecuación de Michaelis-Menten es la expresión matemática que define la relación cuantitativa entre la velocidad de una reacción enzimática y la concentración del sustrato:

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

En esta ecuación, V es la velocidad que se observa en una concentración de sustrato dada [S]; Km es la constante de Michaelis expresada en unidades de concentración (moles/litro), y Vmax es la velocidad máxima al punto de saturación en la concentración del sustrato.

Es posible derivar con facilidad la ecuación mediante la siguiente consideración:

1. Las reacciones enzimáticas incluyen la formación reversible de un complejo enzima-sustrato (ES). Tal complejo se rompe ocasionalmente para regenerar a la enzima (E) y dar lugar al producto (P). Esto se representa mediante la ecuación:



Donde k_1 , k_2 , k_3 y k_4 son las constantes de equilibrio para cada una de las reacciones correspondientes.

2. En unos milisegundos después de que se mezclan las enzimas y el sustrato se establece una concentración de ES que no cambiará mientras exista una alta concentración de S. De esta forma $k_1 \ll k_3$, a esta condición se le llama equilibrio dinámico de la reacción.

En el equilibrio dinámico la velocidad de descomposición de ES se contrarresta con la velocidad de su formación. Teniendo en cuenta que la velocidad de formación de ES es idéntica a su velocidad de descomposición se puede decir que:

$$k_1[E][S] + k_4[E][P] = k_2[ES] + k_3[ES]$$

Y por tanto,

$$\frac{[ES]}{[E]} = \frac{k_1[S] + k_4[P]}{k_2 + k_3}$$

$$\frac{[ES]}{[E]} = \frac{k_1 [S]}{k_2 + k_3} + \frac{k_4 [P]}{k_2 + k_3}$$

3. Si se considera que P es muy pequeño en las primeras etapas de la reacción enzimática, se puede ignorar el término $k_4[P]/(k_2 + k_3)$ y se simplifica a la ecuación:

$$\frac{[ES]}{[S]} = \frac{k_1 [S]}{k_2 + k_3}$$

Al combinar las tres constantes k_1 , k_2 y k_3 en una sola constante K_m se obtiene por medio de la relación:

$$\frac{k_2 + k_3}{k_1} = K_m$$

2.2. SISTEMAS BIOLÓGICOS

Los sistemas biológicos son un tipo particular de los sistemas físicos con la característica de presentar gran complejidad en distintos niveles: organización, evolución, autorreferencia, información, etc.

Independientemente del nivel que se elija (nivel metabólico, celular, de organismo, etc.), los sistemas biológicos son sistemas abiertos, muy alejados del equilibrio, de gran complejidad composicional y en los cuales se dan transformaciones entre sus componentes con propiedades peculiares (por ejemplo, aparición de fenómenos autocatalíticos); para la descripción de estos sistemas es preciso recurrir a relaciones no lineales.

El tipo de orden que aparece en los sistemas biológicos es un orden dinámico, con muchas posibilidades de comportamientos distintos y capacidades de respuesta o variaciones de comportamiento abruptas debido a que son susceptibles a perturbaciones exteriores.

Estos sistemas aparecen como paradigma de la complejidad y la organización. Complejidad, por el alto número de componentes que lo integran; organización, por las características que tienen las ligaduras a las que se ven sometidos estos componentes. Por ejemplo, en una célula ocurren muchas reacciones químicas simultáneamente, pero esto ocurre de una forma organizada, tanto espacial como temporalmente. Si la organización surge espontáneamente, como consecuencia de las propiedades físicas del sistema, entonces el sistema es autoorganizativo.

La importancia de los procesos de autoorganización en la biología es evidente, ya que la vida, si algo es, sobre todo es autoorganización. Autoorganización en el tiempo, coordinando y controlando toda una serie de procesos metabólicos, de división celular, de evolución y selección. Autoorganización en el espacio con la aparición continuada de formas y estructuras sin las cuales difícilmente se podría entender la funcionalidad de dichos sistemas. Y algo más: la aparición de fenómenos emergentes, difícilmente explicables como la suma de los comportamientos individuales de cada uno de los elementos que componen el conjunto, sino más bien como una consecuencia de las interrelaciones existentes entre esos elementos.

Teniendo en cuenta las características acabadas de mencionar, tres sistemas biológicos son simulados en hardware en este trabajo de grado:

1. El ritmo circadiano molecular.
2. El flujo de Ca^{2+} en las células del páncreas
3. El oscilador mitótico

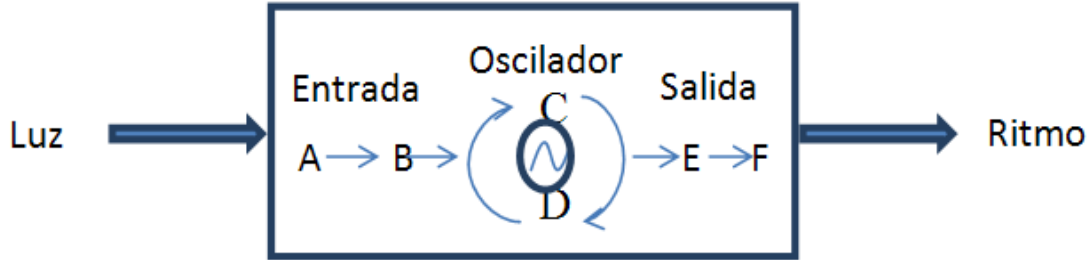
2.3. EL RITMO CIRCADIANO MOLECULAR

En el mundo biológico se presentan constantemente cambios, por lo tanto hay una ventaja para los organismos que se anticipan a los cambios adecuando su fisiología y comportamiento. Esta anticipación a los ciclos diarios es dada por relojes biológicos con un periodo de 1 día.

Las plantas, los hongos y los animales poseen un ritmo metabólico, fisiológico y de comportamiento con un tiempo día- noche. Un ejemplo claro está en nuestro cerebro y nuestro músculo esquelético donde su rendimiento máximo es en el día y en la noche durante el sueño se facilita el crecimiento y reparación de los huesos. En plantas superiores el aparato fotosintético se prepara para sacar el máximo provecho de la luz del sol.

Bajo condiciones constantes los ritmos no se detienen ni desorganizan y se acomodan a los cambios en la temperatura, a diferencia de otras funciones metabólicas. Así que se define un ritmo circadiano como un periodo aproximado de 24 horas que determina cambios fisiológicos y de comportamiento, además puede prescindir de entradas exteriores de información como se observa en la Figura 2.1. Los relojes biológicos deben estar sincronizados con el ciclo solar de luz-oscuridad, la luz brillante en individuos mantenidos en luz tenue puede cambiar sus ritmos; la magnitud dirección del cambio depende del momento del ciclo que se dio la luz.

Fig. 2.1. Modelo simple para el ritmo circadiano molecular.



Los ritmos circadianos deben estar compuestos por:

- Marcapasos o generadores autónomos de ritmos.
- Sensores que detecten cambios ambientales o internos y envíen la información a los marcapasos para producir la sincronización.
- Efectores que son todos los órganos, tejidos y células que respondan a las señales de los marcapasos modificando su actividad.

Hace unos 30 años se descubrió el primer gen relacionado con los ritmos en la mosca de fruta y fue denominado gen PER (period), el gen PER de *Drosophila* se expresa de un modo circadiano y el producto de su síntesis, la proteína PER, inhibe su propia transcripción. En 1994 se descubrió otro gen implicado en el control de la rítmica circadiana, el gen TIM (timeless), de la misma manera que el PER, el RNAm de TIM y PER se expresa durante la noche, generando las proteínas PER y TIM. La proteína PER es fosforilada y es degradada por la proteína DOUBLETIME, también la proteína TIM es degradada por la luz y por un CRIPTOCROMOS (pigmento que se activa por la luz).

Sin embargo la formación de dímeros TIM-PER protege a ambos de su degradación. Estos dímeros entran dentro del núcleo y se unen a los factores de transcripción CLOCK-CYCLE impidiendo la expresión de TIM y PER al separarlos del promotor. Cuando llega la luz, TIM es degradado por CRIPTOCROMOS y por la luz misma, siendo susceptible PER de degradación por DOUBLETIME, con lo que dejan de inhibir su propia transcripción, uniéndose CLOCK-CYCLE a su promotor y todo el ciclo comienza de nuevo. Este ritmo es cíclico como un reloj y el periodo es aproximadamente de 24 horas.

Las fluctuaciones de la proteína PER pueden afectar las transcripciones de genes responsables de los ritmos circadianos de diferentes funciones (producen hormonas, neurotransmisores, etc.). Cualquier pequeña mutación en los genes hace que cambie el periodo del ritmo al cambiar la afinidad de las proteínas.

2.4. FLUJO DE Ca^{2+} EN LAS CÉLULAS DEL PÁNCREAS

El Ion Calcio es usado para la regulación de múltiples actividades en la célula, el Ca^{2+} es una señal de control intracelular para diferentes procesos. Ya que el Ca^{2+} no puede ser degradado, las células regulan las concentraciones de este Ion.

Las fuentes principales de Ca^{2+} son algunos organelos intracelulares (el retículo endoplasmático o el retículo sarcoplasmático) y el medio extracelular. Cuando las concentraciones de Ca^{2+} en el citoplasmático esta en bajos niveles, la célula permanece en estado de reposo, pero cuando las concentraciones suben las células se activan para realizar los mecanismos Ca^{2+} -ON (entrada de Ca^{2+} hacia el citoplasma) o Ca^{2+} -OFF.

Existen sensores intracelulares para la detención de los aumentos de Ca^{2+} , como son la calmodulina y la troponina.

2.4.1 Ca^{2+} - ON

Las concentraciones de Ca^{2+} en el exterior de la célula son ~ 20.000 veces mayores que en el citoplasma y que en el interior de la célula, están cargadas negativamente con respecto al exterior, así que existe un gradiente electroquímico que favorece la entrada de Ca^{2+} . Existen dos grupos de canales encargados de liberar Ca^{2+} hacia el citosol que son los canales de Ca^{2+} de la membrana plasmática (canales sensibles a voltaje, canales activados por liberación de Ca^{2+} y canales regulados por ligando) y los canales de Ca^{2+} intracelulares.

La entrada capacitiva de Ca^{2+} es el fenómeno que describe el vaciamiento de los reservorios intracelulares de Ca^{2+} que induce la entrada de este Ion a través de la membrana plasmática.

El papel fisiológico de la entrada capacitiva de Ca^{2+} es permitir el retorno del Ca^{2+} a los reservorios intracelulares de una manera más rápida. Este tipo de entrada a través de la membrana plasmática en ausencia de despolarización ocurre por medio de los canales activados por liberación de Ca^{2+} y es inactivada por altas concentraciones de Ca^{2+} en el citoplasma.

La función principal del intercambiador Na^{2+}/Ca^{2+} es el transporte desde el citoplasma hacia el medio extracelular, también el intercambio de Na^{2+}/Ca^{2+} en el modo reverso contribuyendo en la entrada de Ca^{2+} .

El incremento de Ca^{2+} citosólico producido por la salida de Ca^{2+} del retículo endoplasmático ocurre principalmente a través de los receptores de $InsP^3$ y rianodina,

2.4.2 Transducción de señal

En la Transducción de señal una proteína del lado externo de la membrana reconoce una señal y activa una regulación a través de una molécula secundaria (segundo mensajero).

Los receptores con actividad enzimática intrínseca se dividen en dos: tirosinquinasa y serina/ treoninaquinasa. El receptor es una proteína transmembrana, con su dominio receptor en la cara extracelular. La molécula señal actúa como un efector que activa el lado enzimático de la proteína. La enzima fosforila los residuos de una proteína iniciando una cascada de regulación para algún proceso bioquímico.

Los receptores ligados a proteínas G constan de una proteína receptora con 7 segmentos transmembrana, esta proteína está asociada a una proteína G y cuando se activa el receptor, se separa para activar otra enzima que produce un segundo mensajero, luego el segundo mensajero puede activar cascadas de regulación.

2.4.3 Inositoltrifosfato (InsP3)

El InsP3 es un segundo mensajero que libera una cantidad de CA^{2+} de las reservas internas de la célula hacia el citosol y se involucra en la toma de CA^{2+} del exterior. Este sistema es secundario a proteínas G o a tirosinquinasa.

Al estar activada la proteína G, se activa una enzima (una fosfolipasa C (PLC)). La PLC degrada una molécula de fosfolípido a inositoltrifosfato y diacilglicerol (DAG) en la membrana plasmática. Cuando el activador de la fosfolipasa C es un tirosinquinasa, la PLC que es activada es una variante de PLC con un dominio de SH2 que se puede unir temporalmente a la membrana.

Al liberar el InsP3, éste se une al receptor IP3R en el retículo endoplasmático, este receptor funciona con canal iónico para el CA^{2+} . Cuando se une el InsP3, se abre el canal durante 10ms. La sensibilidad del receptor depende de varios factores entre ellos de la cantidad de CA^{2+} que este almacenado; el receptor llega a un máximo de sensibilidad de 300nM donde empieza a decrecer de nuevo.

La liberación de CA^{2+} realiza una realimentación positiva al sistema de InsP3 que finaliza en una mayor liberación de CA^{2+} . Existen otros receptores IP3R acoplados a canales iónicos en la membrana celular que permiten la entrada del CA^{2+} del exterior.

2.5. EL OSCILADOR MITÓTICO

Para la división celular es imprescindible:

- Replicar ADN
- La segregación de los cromosomas replicados
- El reparto del material genético equitativamente
- Aumentar el tamaño
- Duplicar orgánulos
- Reorganizar el citoesqueleto

En la sociedad celular algunas células se dividen y otras no, ya que debe haber un equilibrio entre muerte y división. Para lograr este equilibrio son necesarios los factores de control en el ciclo de división celular. Estos factores de control son los factores difusibles de ciclo de división celular, y son los siguientes:

- Factor activador de la fase S
- Factor inhibidor de la re-replicación
- Factor de retraso de la fase M
- Factor promotor de la fase M (MPF)

Los tiempos o intervalos que se miden entre las apariciones de estos factores durante el ciclo de división celular determinan la longitud de dicho ciclo celular; y las relaciones causales entre los factores difusibles hacen que los sucesos del ciclo cromosómico (replicación – mitosis) ocurran en un orden fijo.

Para el proceso celular que se quiere modelar el factor difusible importante es el “Factor Promotor de Maduración” MPF, este factor es el regulador general del paso de G2 a M en la división celular.

Existen unos mutantes que se detienen en diferentes puntos del ciclo denominados CDC (Mutantes del Ciclo de División Celular); el cdc2 está presente en las fases G1 y G2/M. Si se aísla el gen cdc2 codifica para una quinasa, por medio de la fosforilación de proteínas en la regulación del ciclo. Las proteínas quinasa son las reguladoras del ciclo y se les llama: Quinasa del Ciclo de División: CDK (Equivalentes a las cdc).

Las CDK son dependientes de una ciclina. Una ciclina es una proteína que muestra un patrón periódico en el que se sintetiza y se destruye durante el ciclo. Se acumulan durante la interfase y se degradan rápidamente al final de la fase M, su acumulación controla la entrada en Fase M y su degradación controla la salida de fase M.

Con lo anterior tenemos que:

Cdc2: Una quinasa con actividad catalítica dependiente de la ciclina.

Ciclina B: Subunidad reguladora que se requiere para la actividad catalítica de la quinasa cdc2 (cdk)

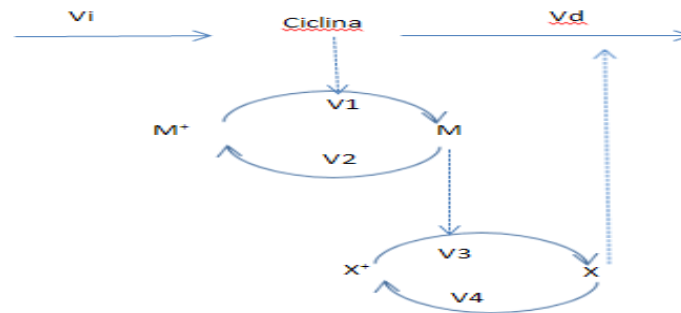
El MPF tiene dos estados, dependiendo de la concentración de Ciclina B:

- A concentraciones bajas de Ciclina B en la célula, es inactivo.
- A concentraciones más altas se vuelve activo.

La ciclina B se degradará cuando el MPF esté activo y lo hará debido a la actividad catalítica que posee la quinasa cdc2.

El análisis se presenta con la estructura mínima de un oscilador mitótico (Figura 2.2), donde los dos factores importantes son la ciclina y la CDC2 quinasa. El mecanismo de esta regulación es que la cdc2 quinasa es inactivada por fosforilación en una tirosina (posiblemente treonina), residuo localizado en el dominio de ATP, la desfosforilación de este residuo termina en la activación de la enzima.

Fig. 2.2. Modelo mínimo en cascada para el oscilador mitótico.



La ciclina (C) es sintetizada a una tasa constante de (vi) y dispara la transformación de inactivo a activo de la cdc2 quinasa (M) para mejorar la rata de un fosfato (E1); una quinasa (E2) revierte esta modificación. En el segundo ciclo de la fosforilación-desfosforilación la cdc2 quinasa provoca la transición de inactivo a activo de una proteasa (X) que degrada la ciclina; la activación de la ciclina proteasa es revertida por un fosfato (E4).

CAPÍTULO 3

SIMULACIÓN FUNCIONAL DE: EL RITMO CIRCADIANO, EL FLUJO DE Ca^{2+} EN LAS CELULAS DEL PÁNCREAS Y EL OSCILADOR MITÓTICO

Los modelos matemáticos del ritmo circadiano, del flujo del Ca^{2+} en las células del páncreas y del oscilador mitótico son simulados en el biosimulador CellDesigner de SBW “*System Biology Workbench*” para realizar la rectificación y comparación respecto a la base de datos biológica “*Biomodels Database*”.

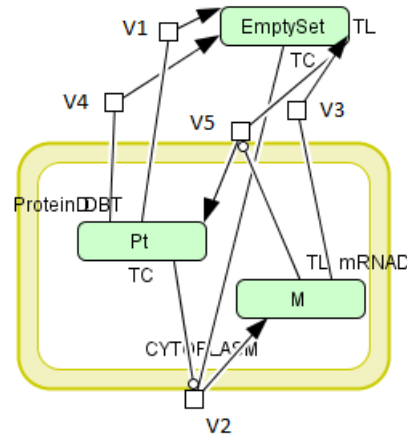
También se realiza la simulación de los tres modelos de los sistemas biológicos con la herramienta Simulink de Matlab, usando la descripción a nivel molecular.

3.1 MODELO DEL RITMO CIRCADIANO

Para la simulación del ritmo circadiano molecular se utilizó el modelo matemático de la BioModels Database “A simple Model of Circadian Rhythms Based on Dimerization and Proteolysis of PER and TIM” (TYSON, HONG, THRON. 1999) en el que a partir de la cinética de Michaelis-Menten se obtienen las ecuaciones de velocidad del modelo que describen las reacciones químicas catalizadas por enzimas.

El diagrama esquemático de las reacciones químicas o pathway del modelo utilizando CellDesigner se observa en la Figura 3.1.

Fig.3.1. Pathway en CellDesigner para el ritmo circadiano molecular.



Las ecuaciones de velocidad que describen las reacciones químicas del sistema son:

$$V1 = \frac{V_m}{1 + \left[\frac{P * \left(1 - \frac{2}{1 + (1 + 8 * P * K_{eq})^{\frac{1}{2}}} \right)}{2 * P_{crit}} \right]^2}$$

$$V2 = V_p * M$$

$$V3 = K_m * M$$

$$V4 = K3 * P$$

$$V5 = \frac{\frac{2 * K1 * P}{1 + (1 + 8 * K_{eq} * P)^{\frac{1}{2}}} + K2 * P}{J + P}$$

Usando la estequiometría de las reacciones se hallan las ecuaciones de diferencia del modelo:

$$\frac{dM}{dt} = V1 - V3$$

$$\frac{dP}{dt} = V2 - V4 - V5$$

Donde M es la concentración para PER y TIM MRNA, y P es la concentración para PER y TIM monómero y dímero. Los valores para los parámetros del modelo se presentan en la Tabla 3.1:

Tabla 3.1. Tabla de parámetros para el modelo matemático del ritmo circadiano molecular.

Parámetro	Valor
K1	10
K2	0.03
Keq	200
J	0.05
Vm	1
Pcrit	0.1
Km	0.1
K3	0.1
Vp	0.5

Las condiciones iniciales para las concentraciones de los metabolitos del ritmo circadiano se encuentran expresadas en mol/L.

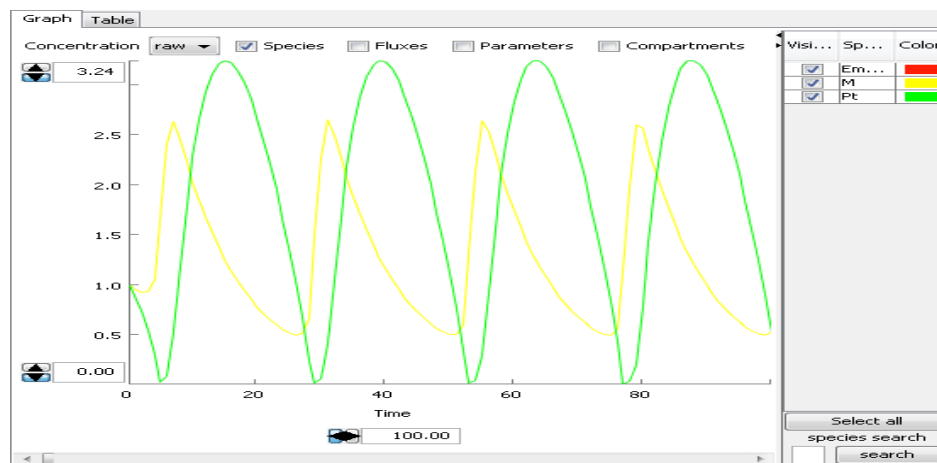
Condición inicial $P = 1$

Condición inicial $M = 1$

3.1.1 Simulación con el biosimulador Celldesigner

El resultado de las concentraciones de P y M para el modelo “A simple Model of Circadian Rhythms Based on Dimerization and Proteolysis of PER and TIM” en CellDesigner se muestra en la Figura 3.2.

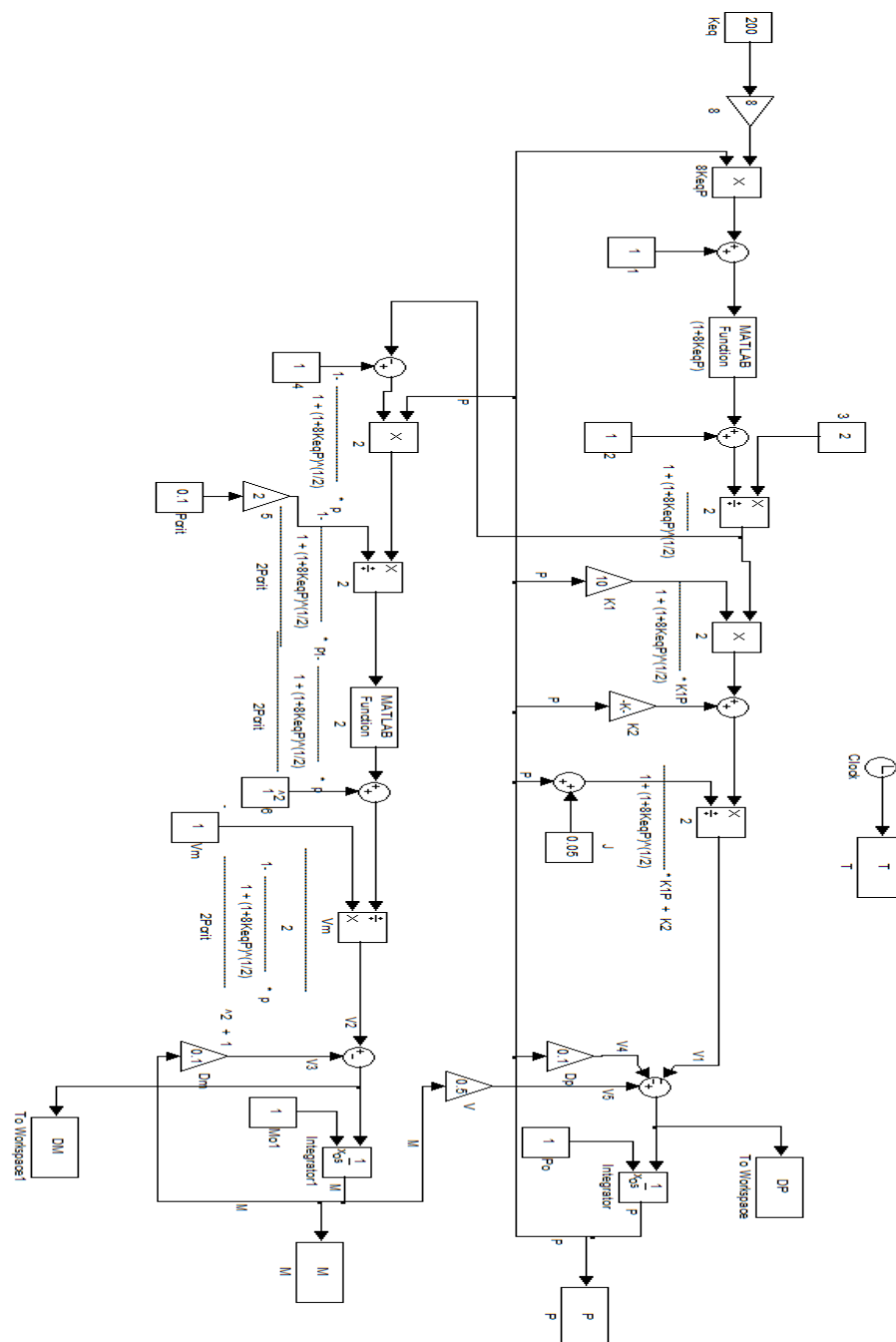
Fig.3.2. Grafica de concentraciones para los metabolitos P y M del ritmo circadiano molecular simulado en CellDesigner.



3.1.2 Simulación con la herramienta Simulink de Matlab

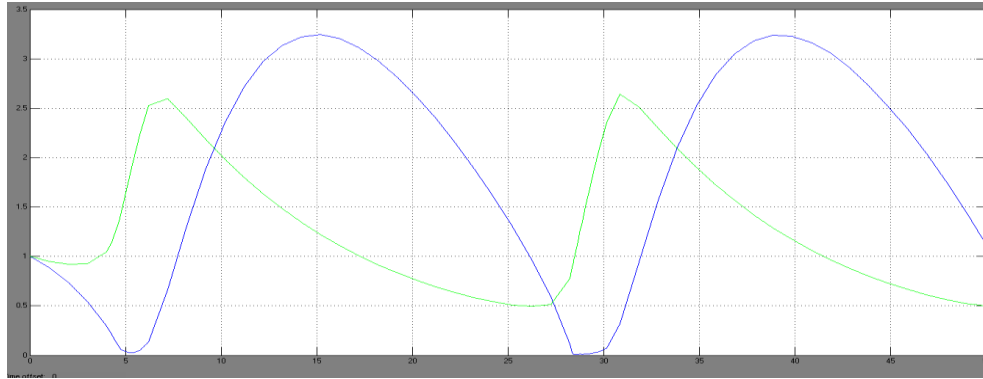
Para la simulación en Simulink del modelo matemático del ritmo circadiano se definen las ecuaciones de diferencia a partir de diagramas de bloques. En la figura 3.3 se muestra el diagrama de bloques para el modelo matemático del ritmo circadiano, usando el método numérico de Runge-Kutta de quinto orden (ode45).

Fig.3.3. Diagrama de bloques para el modelo del ritmo circadiano basado en la dimerización y proteólisis del PER y TIM implementado en Simulink.



Las concentraciones de P y M son almacenadas en bloques “To Workspace” como vectores columnas, para retornar los valores al espacio de trabajo de Matlab. En la Figura 3.4 se muestra el resultado obtenido de la simulación del modelo en Simulink.

Fig.3.4. Gráfica de resultados para el modelo del ritmo circadiano simulado en Simulink de Matlab.

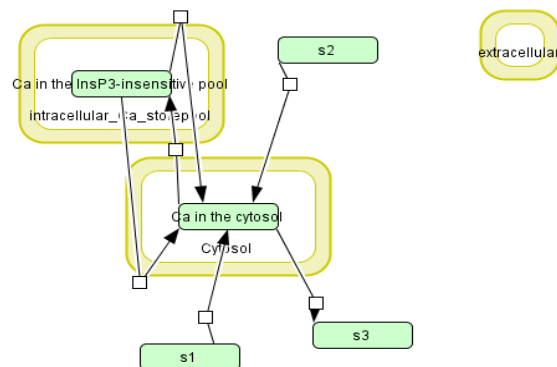


3.2 MODELO DEL FLUJO DE Ca^{2+} EN LAS CÉLULAS DEL PÁNCREAS

El modelo matemático que se seleccionó del BioModels Database para el flujo del Ca^{2+} es el “Signal-induced Ca^{2+} oscillations: properties of a model based on Ca^{2+} induced Ca^{2+} release” de DuPont (1991). En este modelo cinético de las reacciones químicas se estudian las oscilaciones amortiguadas del Ca^{2+} .

El diagrama esquemático de las reacciones químicas del modelo usando CellDesigner se muestra en la Figura 3.5:

Fig.3.5. Pathway del flujo del CA de Dupont implementado en CellDesigner.



Las ecuaciones de velocidad presentadas por el modelo que describen las reacciones son:

$$V1 = V1 * beta * cytosol$$

$$V2 = v0 * cytosol$$

$$V3 = kf * Y * cytosol$$

$$V4 = \frac{Intracell * VM2 * Z^n}{k2^n + Z^n}$$

$$V5 = cytosol * VM3 * \frac{Y^m}{kr^m + Y^m} * \frac{Z^p}{ka^p + Z^p}$$

$$V6 = k * Z * extracell$$

$$beta = Flag * 0.96 * e^{-0.2 * (time - tstime)}$$

$$Flag = 1 \text{ } <= > \text{ } (time > tstime)$$

El sistema de ecuaciones diferenciales del modelo es:

$$\frac{dZ}{dt} = V1 + V2 + V3 + V5 - V4 - V6$$

$$\frac{dY}{dt} = V4 - V3 - V5$$

Donde (Z) es la concentración del calcio citosólico y (Y) la concentración de calcio sensible al InsP3 (InositolTrifosfato). Los valores de los parámetros para el modelo matemático se muestran en la Tabla 3.2.

Tabla 3.2. Tabla de parámetros para el modelo matemático del flujo de CA^{+2} de Dupoont.

Parámetro	Valor
Vo	1
V1	7.3
VM2	65
VM3	500
Kr	2
Ka	0.9
K2	1
Kf	1
K	10
n	2
m	2
P	4
tstime	4

Las condiciones iniciales para los metabolitos del oscilador de CA^{+2} de Dupont en $\mu\text{mol/L}$ son:

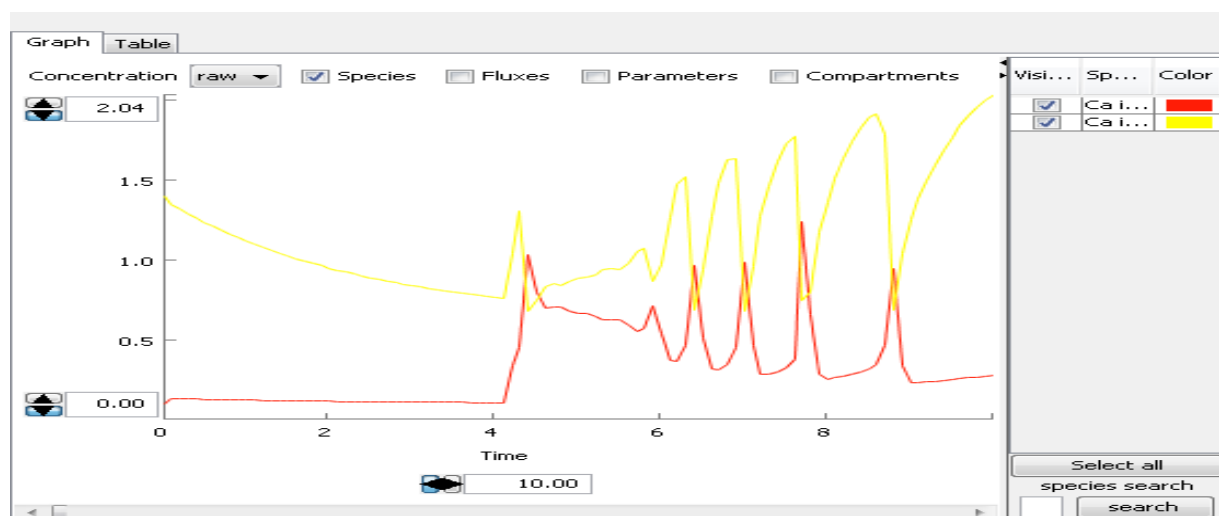
Condición Inicial Z = 0.1

Condición Inicial Y= 1.4

3.2.1 Simulación con el biosimulador Celldesigner

El resultado obtenido de la simulación en CellDesigner del modelo matemático para las oscilaciones de CA^{+2} de Dupont se presenta en la Figura 3.6:

Fig.3.6. Gráfica de concentraciones para los metabolitos Z y Y del flujo de CA^{+2} en CellDesigner.

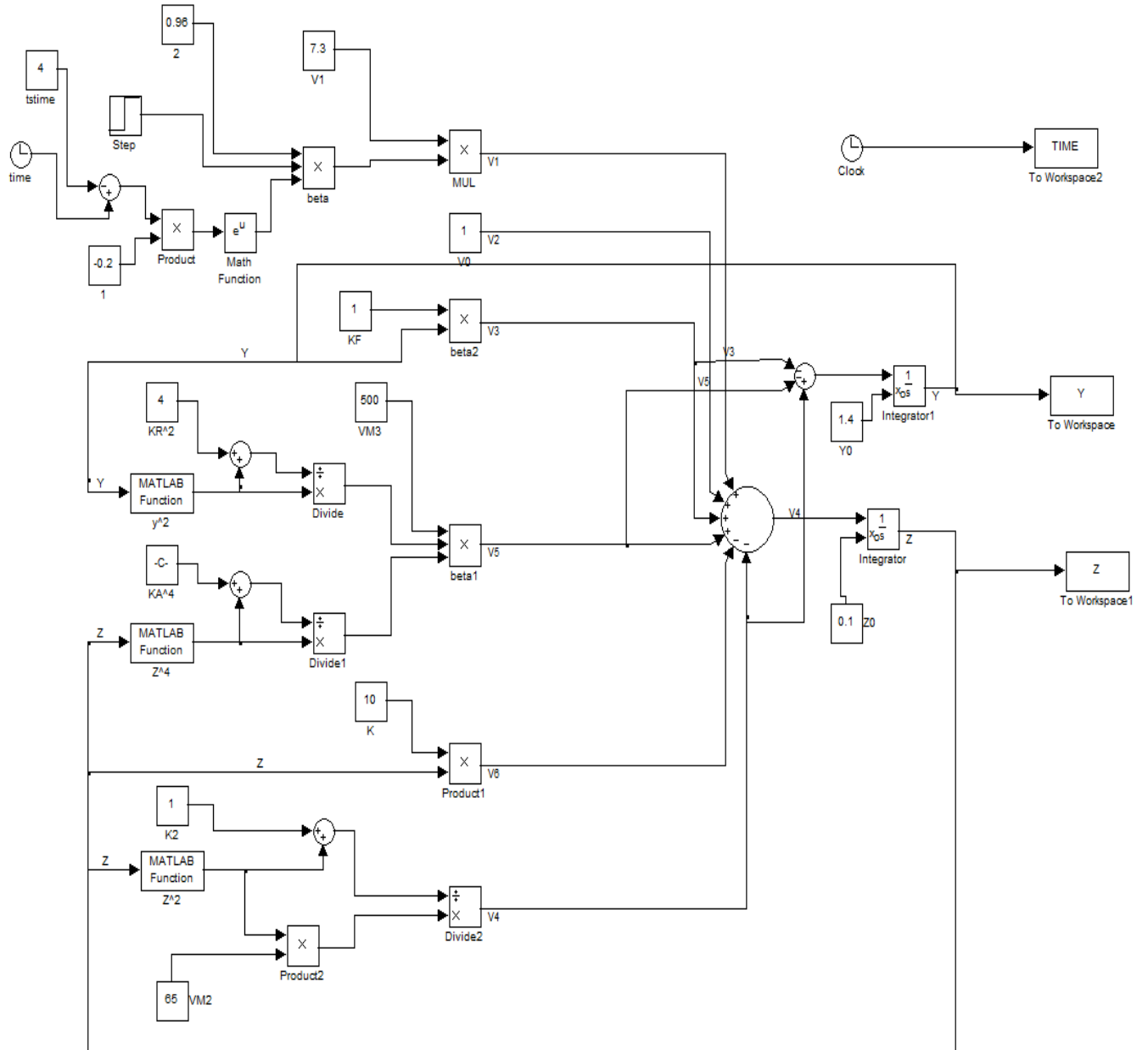


3.2.2 Simulación con la herramienta Simulink de Matlab

Con el propósito de verificar el modelo matemático del oscilador de CA^{+2} de Dupont, se realizó la simulación de las ecuaciones de diferencia en Simulink de Matlab, usando el método numérico de Runge-Kutta de quinto orden (ode45).

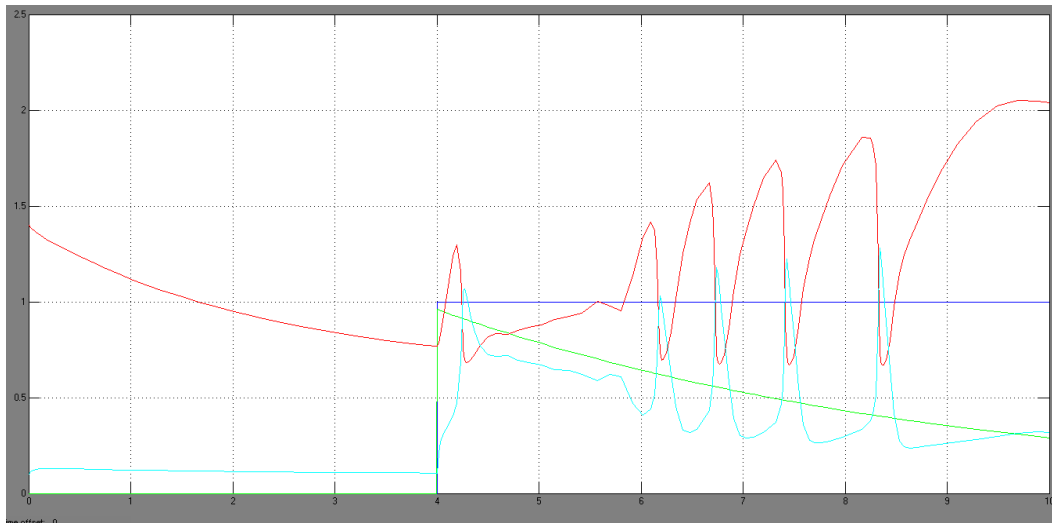
El diagrama de bloques implementado para el modelo se presenta en la Figura 3.7.

Fig.3.7. Diagrama de bloques para el modelo del oscilador de CA^{+2} de Dupont implementado en Simulink.



El resultado para las concentraciones de Y y Z obtenido en Simulink se muestra en la Figura 3.8:

Fig.3.8.Gráfica de Resultadospara el modelo del flujo de CA implementado en Simulink.

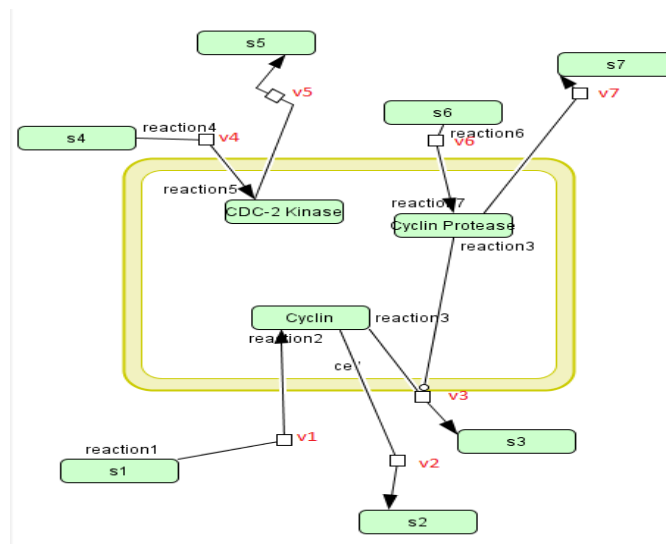


3.3 MODELO DEL OSCILADOR MITÓTICO

El modelo escogido para el oscilador mitótico se encuentra en el Biomoldes DataBase en el artículo “A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 Kinase” de Goldbeter(1991). Este artículo presenta el modelo matemático desde el punto de vista de la cinética enzimática.

El diagrama esquemático del modelo del oscilador mitótico de Goldbeter realizado en CellDesigner se observa en la Figura 3.9:

Fig.3.9. Pathway para el oscilador mitótico de Goldbeter implementado en CellDesigner.



Las tres especies del modelo del oscilador mitótico son la ciclina, la cdc2 quinasa y la ciclina proteasa y las ecuaciones de las velocidades de reacción según el modelo son:

$$V1 = v_i$$

$$V2 = C * k_d$$

$$V3 = V_{dX} \frac{C}{K_d + C}$$

$$V4 = V_{11} \frac{(I - M)}{K_1 + (I - M)}$$

$$V5 = V_{22} \frac{M}{K_2 + M}$$

$$V6 = V_{33} \frac{(I - X)}{K_3 + (I - X)}$$

$$V7 = V_{44} \frac{X}{K_4 + X}$$

$$V_{11} = \frac{C}{K_c + C} V_{m1}, \quad V_{33} = M V_{m3}$$

Las ecuaciones de diferencia de las tres especies están dadas por:

$$\frac{dc}{dt} = V1 - V2 - V3$$

$$\frac{dM}{dt} = V4 - V5$$

$$\frac{dX}{dt} = V6 - V7$$

Donde C es la ciclina, M es la cdc2 quinasa y X la proteasa. Los valores para los parámetros presentados por el modelo se muestran en la Tabla 3.3.

Tabla 3.3. Tabla de parámetros para el modelo matemático del oscilador mitótico.

Parámetros	Valores
V_i	0.025
K_d	0.01
V_d	0.25
K_d	0.02
K_1	0.005
V_{22}	1.5
K_2	0.005
K_3	0.005
K_4	0.005
V_{44}	0.5
VM_1	3
VM_3	1
K_c	0.5

Las condiciones iniciales para los metabolitos del oscilador de Goldbeter en mol/L son:

Condición Inicial $C = 0.01$

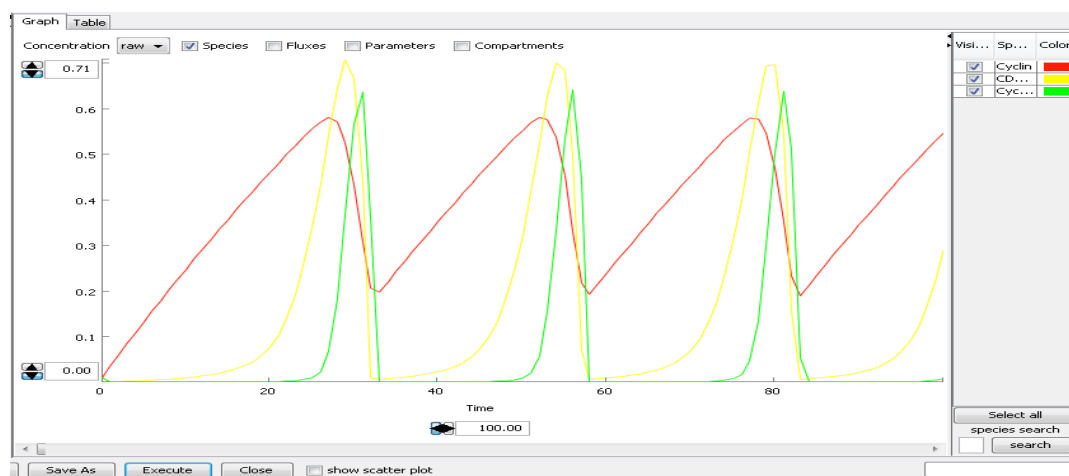
Condición Inicial $M = 0.01$

Condición Inicial $X = 0.01$

3.3.1 Simulación con el biosimulador Celldesigner

El resultado de la simulación para el modelo matemático del sistema “A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 Kinase” en el biosimulador CellDesigner se muestra en la Figura 3.10:

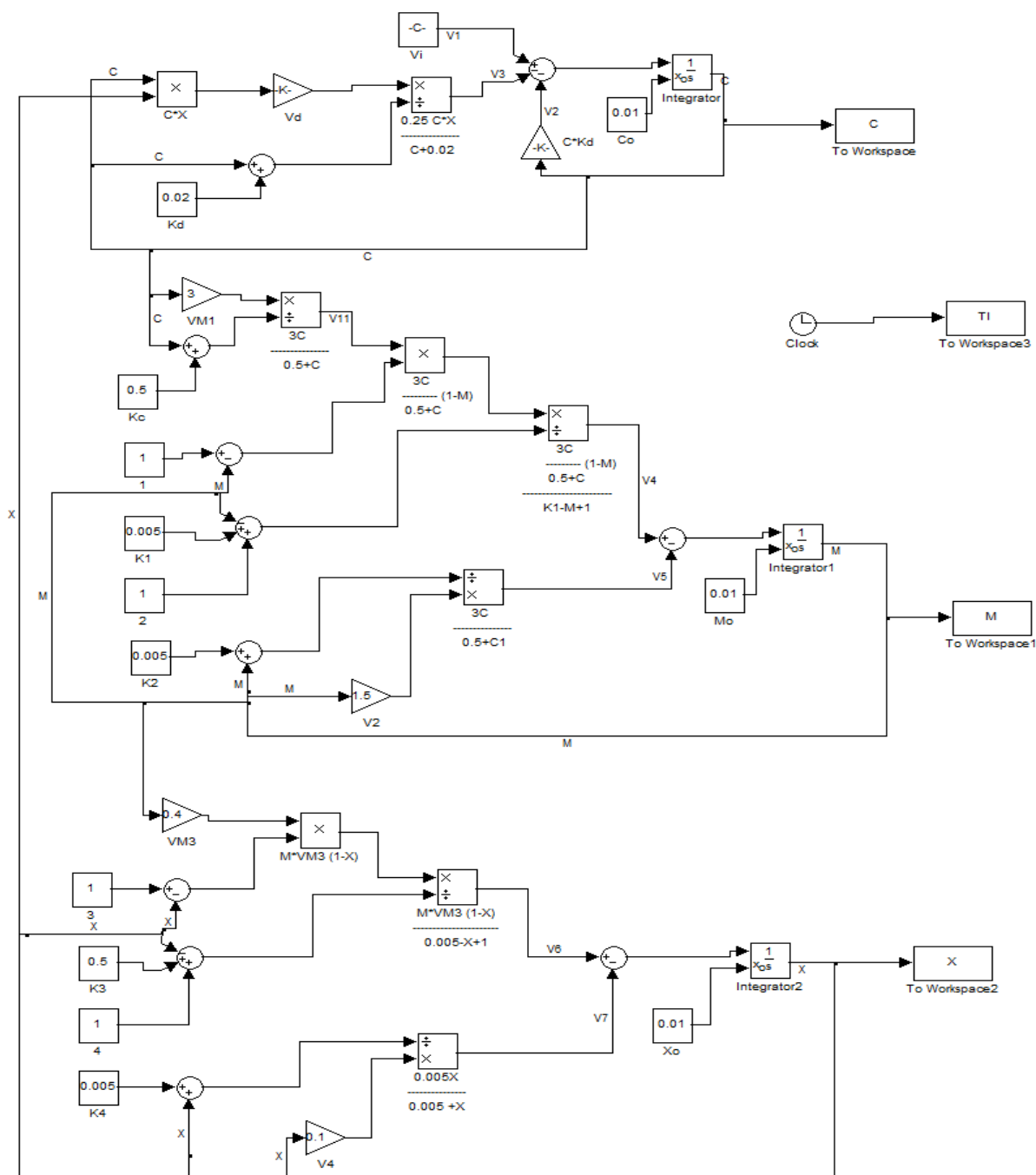
Fig.3.10. Gráfica de Resultados para el modelo del oscilador mitótico de Goldbeter simulado en CellDesigner.



3.3.2 Simulación con la herramienta Simulink de Matlab

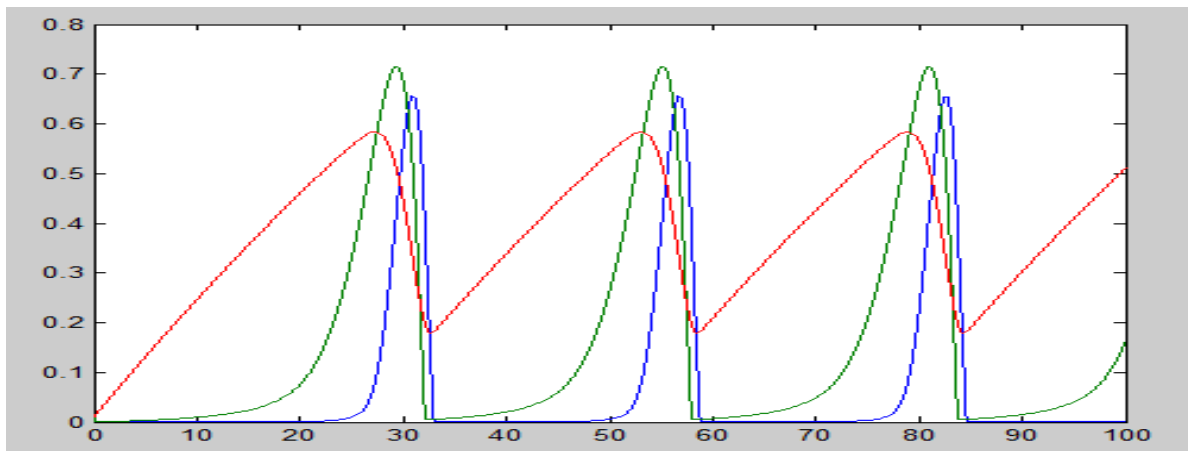
El diagrama de bloques implementado en Simulink para las ecuaciones de diferencia, usando el método numérico de Runge-Kutta de quinto orden (ode45) para el modelo matemático del oscilador mitótico de Goldbeter, se presenta en la Figura 3.11:

Fig.3.11. Diagrama de bloques para el modelo del oscilador mitótico implementado en Simulink.



En la Figura 3.12 se grafican las concentraciones de la ciclina, el cdc quinasa y de la quinasa proteasa obtenidas a partir de la simulación realizada en Simulink para el oscilador mitótico de Goldbeter.

Figura 3.12. Gráfica de concentraciones para los metabolitos del oscilador mitótico implementado en la Simulink.

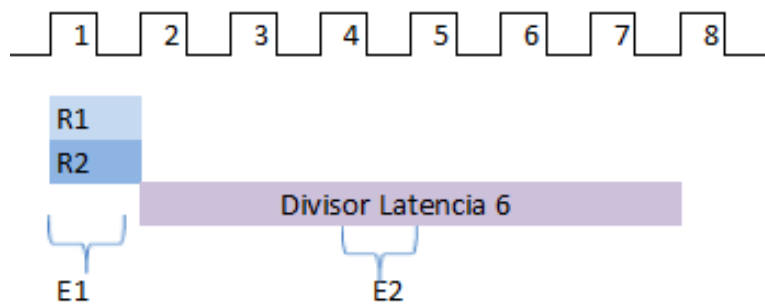


CAPÍTULO 4

IMPLEMENTACIÓN EN HARDWARE DEL RITMO CIRCADIANO, DEL FLUJO DE Ca^{2+} Y EL OSCILADOR MITOTICO

Para la optimización en la implementación del hardware, los sistemas se diseñaron usando máquinas de estado, donde el control para el paso de un estado actual al siguiente estado está dado por un diagrama de tiempos dependiente de la latencia de los diferentes bloques funcionales. Por ejemplo, en la Figura 4.1 se muestra un diagrama de tiempos, donde la carga de datos en los registros R1 y R2 se realiza en el estado E1 y en el estado E2 se realiza la división de estos dos datos.

Fig. 4.1. Modelo de los diagramas de tiempos implementados para el diseño de los sistemas biológicos en hardware.



El datapath utilizado para cada sistema biológico es el mismo para la implementación con los métodos numéricos: Euler, regla trapezoidal y regla de Simpson; pero para cada sistema con cada método numérico se implementó un control diferente (diferentes máquinas de estado). Las implementaciones en hardware fueron realizadas en el lenguaje VHDL con la herramienta Modelsim de Quartus.

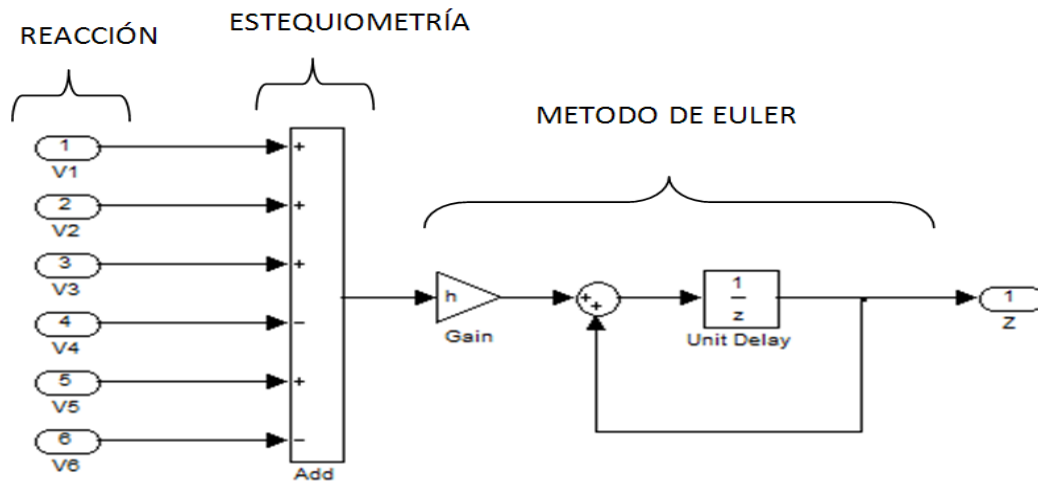
El método numérico de Euler consistente en incrementar paso a paso la variable independiente y hallar la siguiente imagen con la derivada

Este método responde al siguiente algoritmo:

$$y(n + 1) = y_n + hf(tn, y_n)$$

Para implementar en hardware los sistemas de ecuaciones diferenciales usando el método de Euler se utiliza un multiplicador, un sumador y un registro. La Figura 4.2 presenta la implementación en hardware del método de Euler.

Fig. 4.2 Diagrama de bloques para la implementación en hardware de los modelos matemáticos de los sistemas biológicos usando el método de Euler.



Para realizar el método de integración de la regla trapezoidal se tiene en cuenta que el polinomio de interpolación entre dos datoses una línea recta. La integral corresponde al área bajo la línea recta en el intervalo (a,b) que es el área del trapecio que se forma.

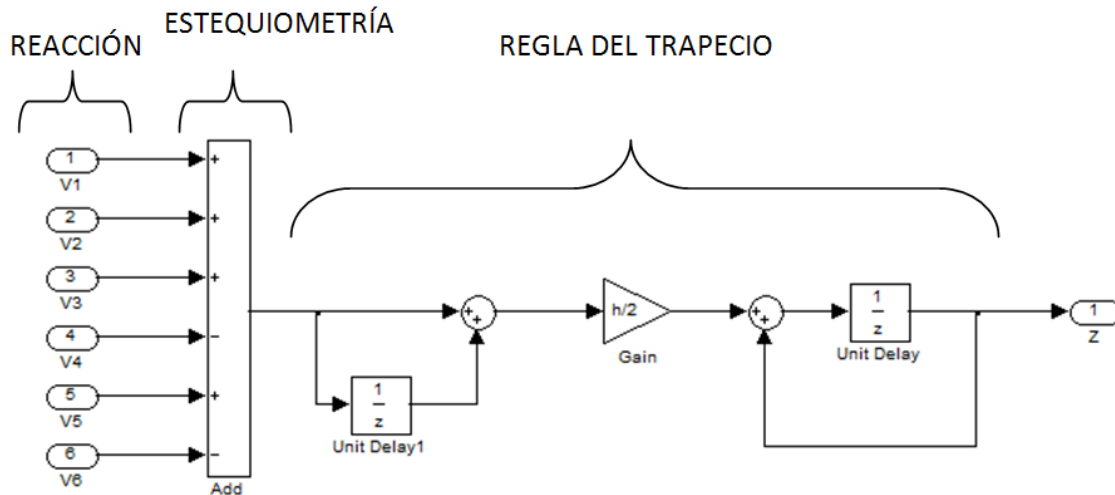
El método trapezoidal responde a algoritmo:

$$y(n + 1) = y_n + \frac{h}{2} (f(tn, y_n) + f(tn + 1, y_n + 1))$$

Para la implementación en hardware de los sistemas de ecuaciones diferenciales utilizando el método de integración de la regla trapezoidal se usan dos registros, un multiplicador y dos sumadores como se muestra en la Figura 4.3:

Fig. 4.3. Diagrama de bloques para la implementación en hardware de los modelos matemáticos de los sistemas biológicos usando la regla del trapecio.

VELOCIDADES DE



Para hallar $[y(n+1)]$ en la primera iteración se hace por medio de la aproximación que nos brinda el método de Euler, después de la primera iteración se desarrolla el método trapezoidal para resolver las ecuaciones diferenciales.

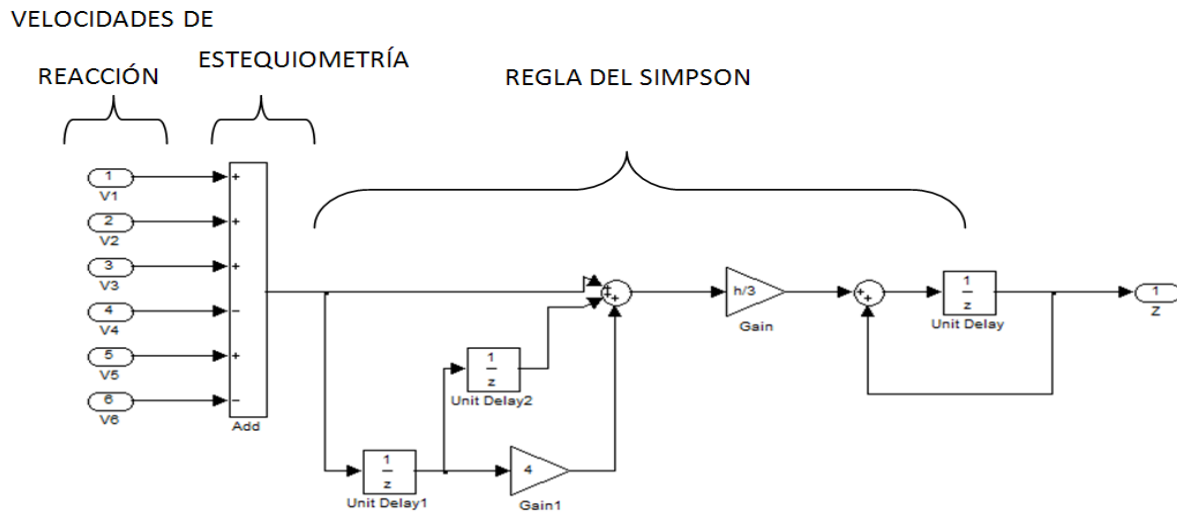
El método de integración de la regla de Simpson usa polinomios de orden superior para conectar los puntos. Si hay un punto medio entre $f(a)$ y $f(b)$, entonces los tres puntos se pueden conectar con un polinomio de tercer orden.

El algoritmo del método de Simpson es el siguiente:

$$y(n+1) = y_n + \frac{h}{3} (f(t_n, y_n) + 4f(t_n + 1, y_n + 1) + f(t_n + 2, y_n + 2))$$

Para la implementación en hardware de los sistemas de ecuaciones lineales desarrollando la integración por medio de la regla de Simpson se usan tres registros, dos multiplicadores y tres sumadores como se presenta en la Figura 4.4.

Fig. 4.4. Diagrama de bloques para la implementación en hardware de los modelos matemáticos de los sistemas biológicos usando la regla de Simpson.

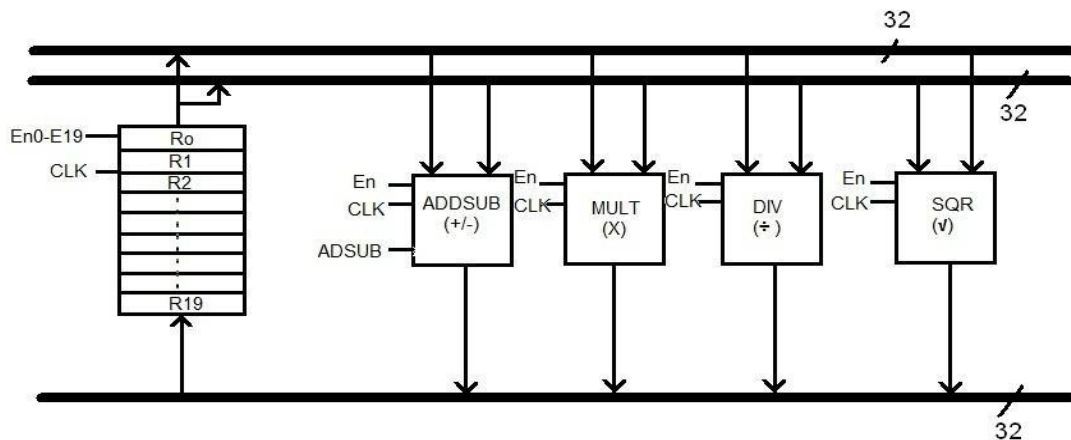


Al resolver las ecuación por medio del método de Simpson para hallar $[y(n+1)]$ y $[y(n+2)]$ las primeras dos iteraciones son halladas por medio de la aproximación que brinda el método de Euler.

4.1 IMPLEMENTACIÓN EN HARDWARE DEL MODELO DEL RITMO CIRCADIANO

Para el diseño en hardware del sistema biológico “A simple Model of CircadianRhythms Basedon Dimerization and Proteolysis of PER and TIM”se implementó el datapath de la Figura 4.5:

Fig. 4.5. Datapath para las implementaciones en hardware del sistema “modelo del ritmo circadiano basado en la dimerización y proteólisis del PER y TIM”.

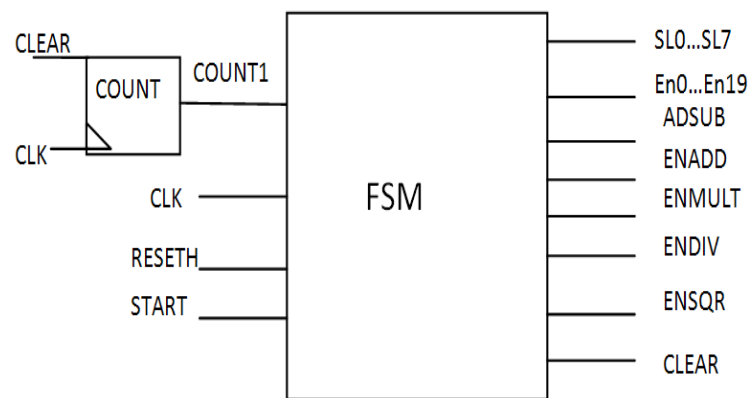


Los bloques funcionales divisor, multiplicador, sumador-restador y raíz fueron implementados usando la herramienta de QuartusII “MegaWizard- Plug in manager” utilizando la librería de punto flotante de Altera “altera_mf”. El formato de representación del número en punto flotante IEEE-754 es de precisión simple (32 bits).

4.1.1 Implementación en hardware del ritmo circadiano usando el Método de Euler.

La máquina de estado para el control del datapath anterior, implementada con el método de Euler (con las respectivas entradas y salidas) se representa en la Figura 4.6:

Fig.4.6. FSM para la implementación del modelo del ritmo circadiano basado en la dimerización y proteólisis del PER y TIM usando el método de Euler.



La señal (Count1) controla el cambio de estado actual al siguiente dependiendo de la latencia de los bloques funcionales. El sumador-restador que se utiliza tiene una latencia de 7 ciclos de reloj, el multiplicador de 5 ciclos de reloj, el divisor de 6 ciclos de reloj y la raíz una latencia de 16 ciclos de reloj.

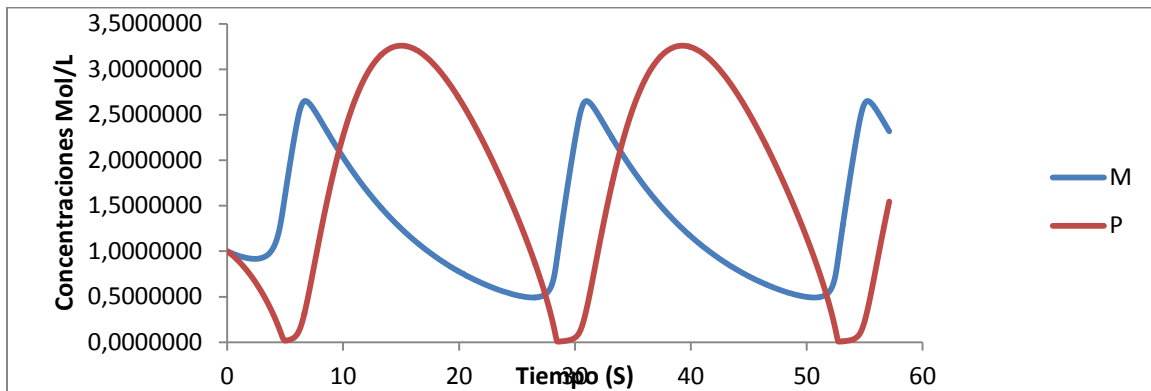
El diagrama de tiempos de los bloques funcionales para la implementación del diagrama de estados se puede observar en el anexo “ANEXO A”, las salidas de la FSM controlan *enables* de dispositivos, *load* de registros y *selects* de multiplexores.

El diagrama de estados Moore para el modelo del ritmo circadiano molecular usando el método de Euler se muestra en la Figura 4.7.

Se implementó un test-bench para probar el correcto funcionamiento del modelo en hardware; el código en VHDL para este test-bench se puede observar en el anexo “ANEXO C”.

46

Fig 4.8. Gráfica de concentraciones de los metabolitos para el modelo del ritmo circadiano implementado en hardware usando el método de Euler.

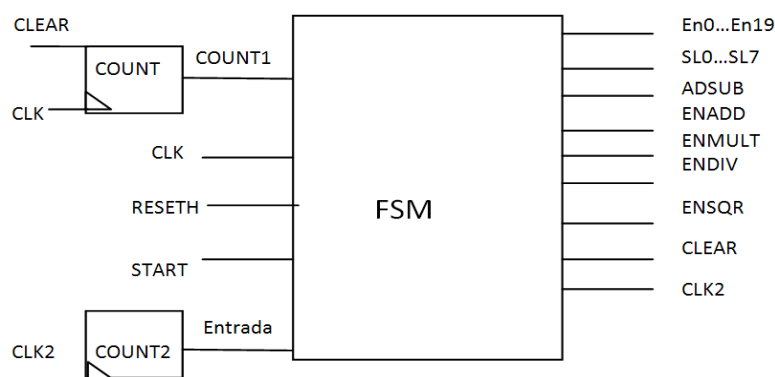


4.1.2 Implementación en hardware del ritmo circadiano usando la regla trapezoidal

Para la implementación de este método de integración, en las señales de control (de entrada) de la máquina de estados, se implementó una señal llamada “Entrada” para controlar el cambio del método numérico después de la primera iteración. Esta señal es la salida de un segundo contador que realiza la cuenta de las iteraciones realizadas en una simulación. La otra señal de control “count1” indica el cambio de estado dependiendo de las latencias de los bloques funcionales.

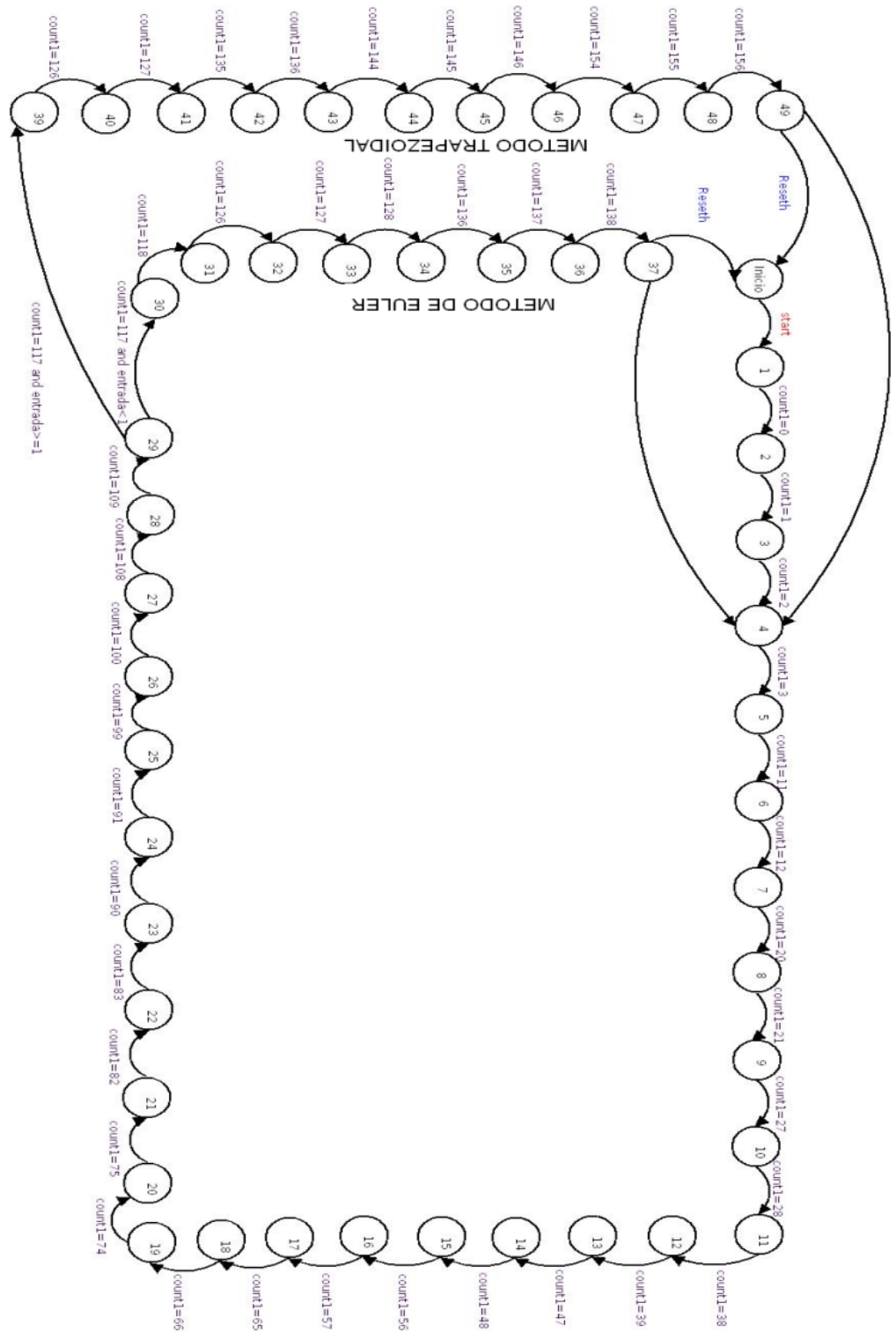
La FSM implementada para el modelo matemático del ritmo circadiano molecular usando el método de integración de la regla trapezoidal se observa en la Figura 4.9:

Fig. 4.9. FSM para el modelo del ritmo circadiano usando la regla trapezoidal y la regla de Simpson.



En la Figura 4.10 se muestra el diagrama de estados para el modelo matemático del sistema, usando el método de integración de la regla trapezoidal.

Fig. 4.10. Diagrama de estados para el modelo del ritmo circadiano usando la regla trapezoidal.

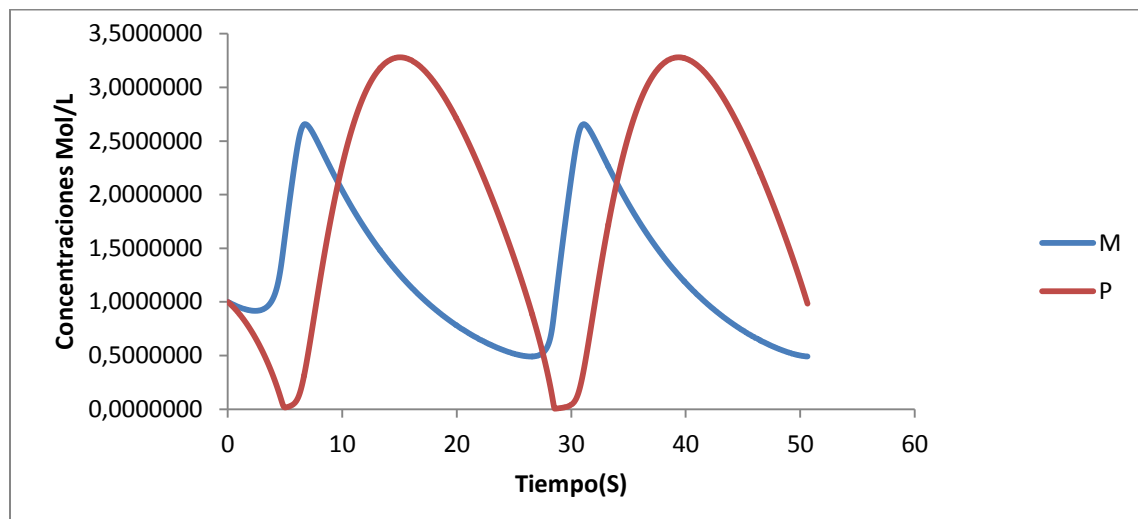


En los estados 1,2 y 3 se realiza el *load* de las condiciones iniciales de los metabolitos. En la primera iteración se usa el método de Euler para resolver las ecuaciones de diferencia;

después, las ecuaciones se resuelven por medio del método trapezoidal pasando del estado 29 al estado 39. El código en VHDL de la FSM implementada para el modelo del ritmo circadiano usando la regla trapezoidal se muestra en el anexo “ANEXO F”-

El resultado obtenido por la implementación en hardware del modelo del ritmo circadiano molecular usando la regla trapezoidal se presenta en la Figura 4.11:

Fig.4.11. Gráfica de resultados para la implementación en hardware del ritmo circadiano usando la regla trapezoidal.



4.1.3 Implementación en hardware del ritmo circadiano usando la regla de Simpson

Para la implementación en hardware del modelo matemático del ritmo circadiano molecular usando el método de integración de la regla de Simpson, se utiliza la misma FSM implementada para el modelo de este sistema usando el método de integración de la Regla trapezoidal.

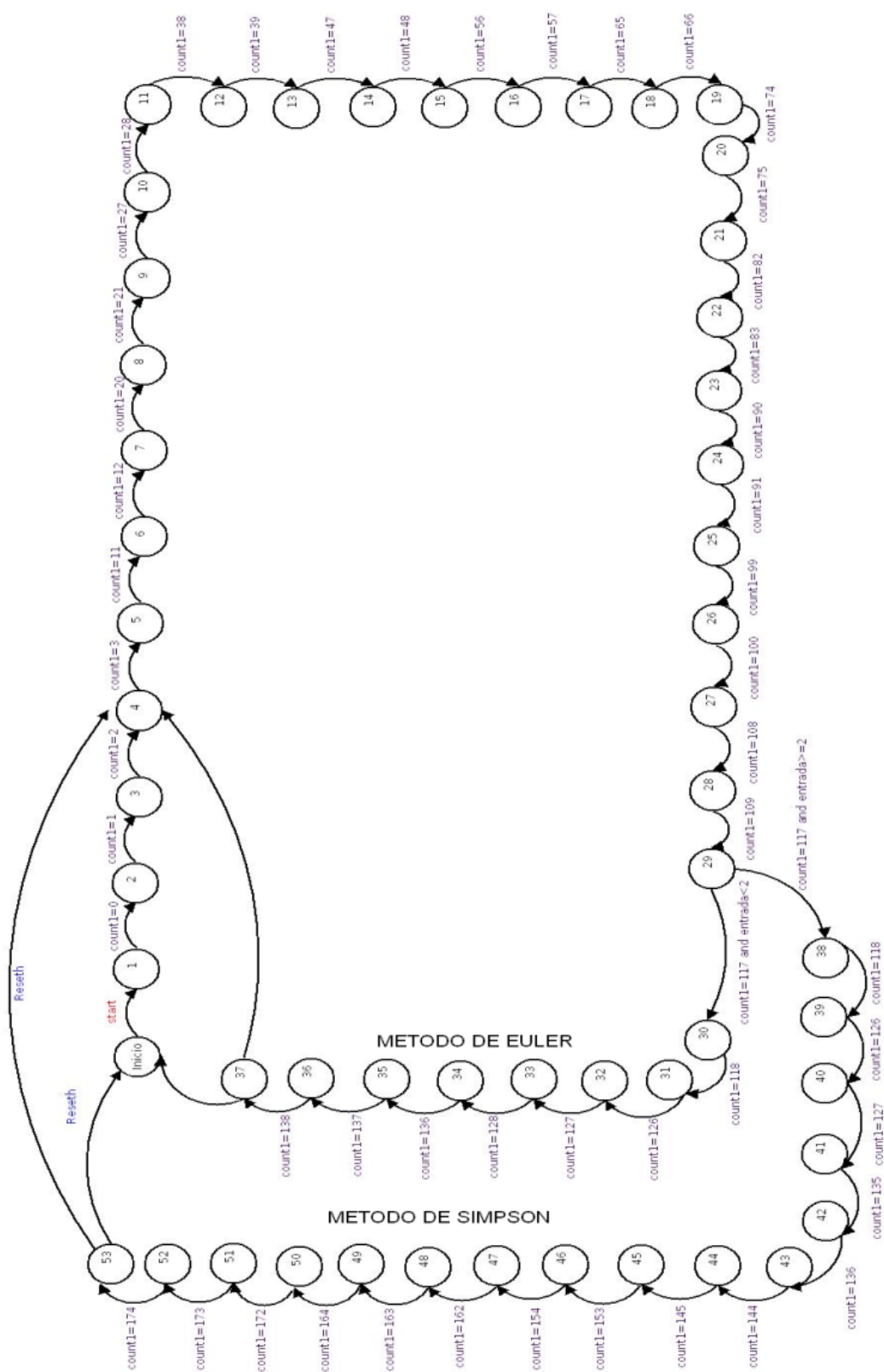
En la máquina de estados las entradas que controlan el paso del estado actual al estado siguiente son:

Count: Señal para el control dependiendo de la latencia de los bloques funcionales.

Entrada: Señal para el control dependiendo de las iteraciones realizadas. En las primeras dos iteraciones las ecuaciones se resuelven por el método de Euler; para iteraciones mayores a dos las ecuaciones se resuelven por el método de Simpson.

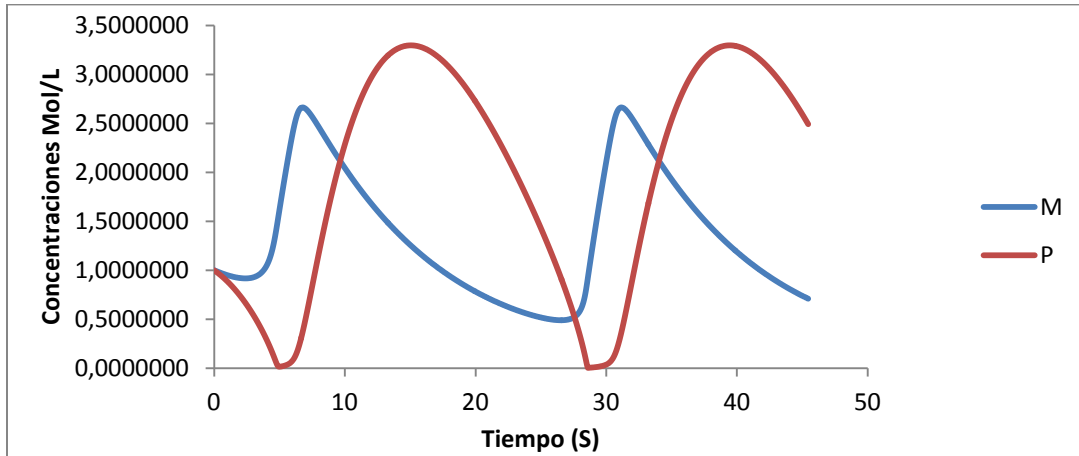
El diagrama de estados del modelo del ritmo circadiano usando la regla del trapecio se muestra en la Figura 4.12

Fig. 4.12. Diagrama de estados del modelo del ritmo circadiano usando la regla de Simpson.



El resultado para las concentraciones de los metabolitos en la implementación hardware del ritmo circadiano usando la regla de Simpson se muestra en la Figura 4.13. El código en VHDL para la implementación de la FSM para el modelo del ritmo circadiano usando la regla de Simpson se muestra en el anexo “ANEXO G”.

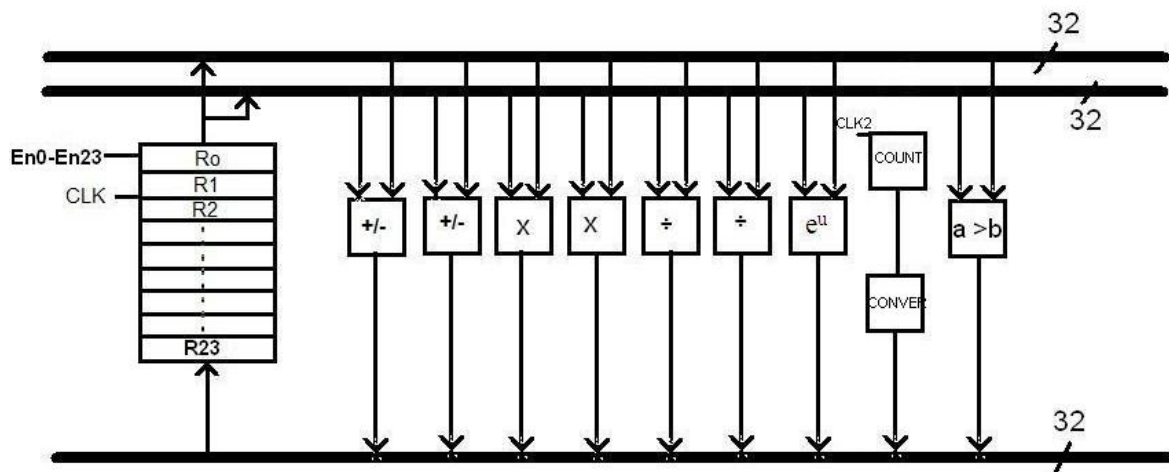
Fig. 4.13. Gráfica de resultados para la implementación del ritmo circadiano usando la regla de Simpson.



4.2 IMPLEMENTACIÓN EN HARDWARE DEL MODELO DEL FLUJO DE Ca^{2+} EN LAS CÉLULAS DEL PÁNCREAS

Para la implementación en hardware del sistema biológico “Modelado de Ca^{2+} flujo en las células del páncreas: papel de la membrana plasmática y las reservas intracelulares”, usando los métodos numéricos Euler, regla trapezoidal y regla de Simpson, se implementó el datapath de la Figura 4.14:

Fig. 4.14. Datapath para las implementaciones en hardware del sistema “Modelado de Ca^{2+} flujo en las células del páncreas: papel de la membrana plasmática y las reservas intracelulares”.

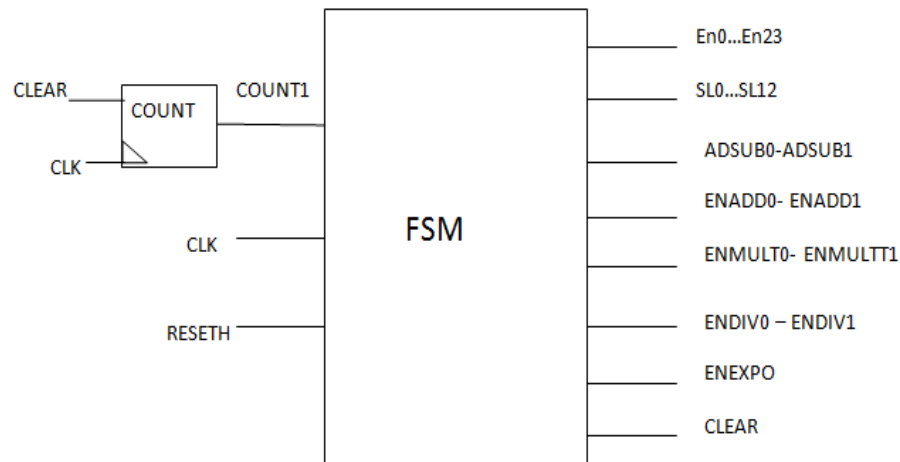


Los bloques funcionales divisor, multiplicador, sumador-restador, exponencial, comparador y el conversor de punto fijo a punto flotante fueron implementados usando la herramienta de QUARTUSII “MegaWizard- Plug in manager” utilizando la librería de punto flotante de altera “altera_mf”.

4.2.1 Implementación en hardware del flujo de CA^{2+} usando el método de Euler

La FSM implementada para el modelo del flujo de CA^{2+} usando el método de Euler, para el control del datapath de la Figura 4.14, se muestra en la Figura 4.15.

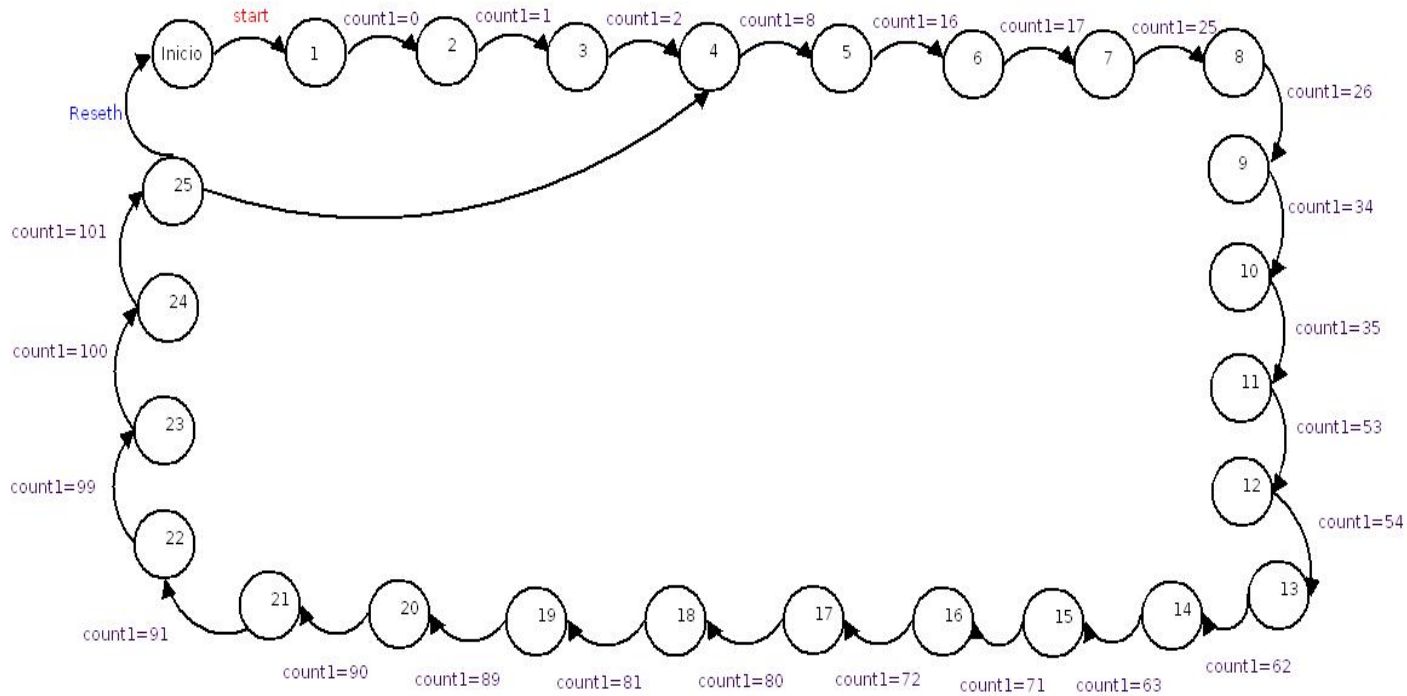
Fig. 4.15. .FSM para el modelo de flujo de CA^{2+} usando método de Euler.



La señal de control (Count1) controla el cambio de estado actual al siguiente estado, dependiendo de la latencia de los bloques funcionales. El sumador-restador utilizado tiene una latencia de 7 ciclos de reloj, el multiplicador de 5 ciclos de reloj, el divisor de 6 ciclos de reloj y la función exponencial una latencia de 17.

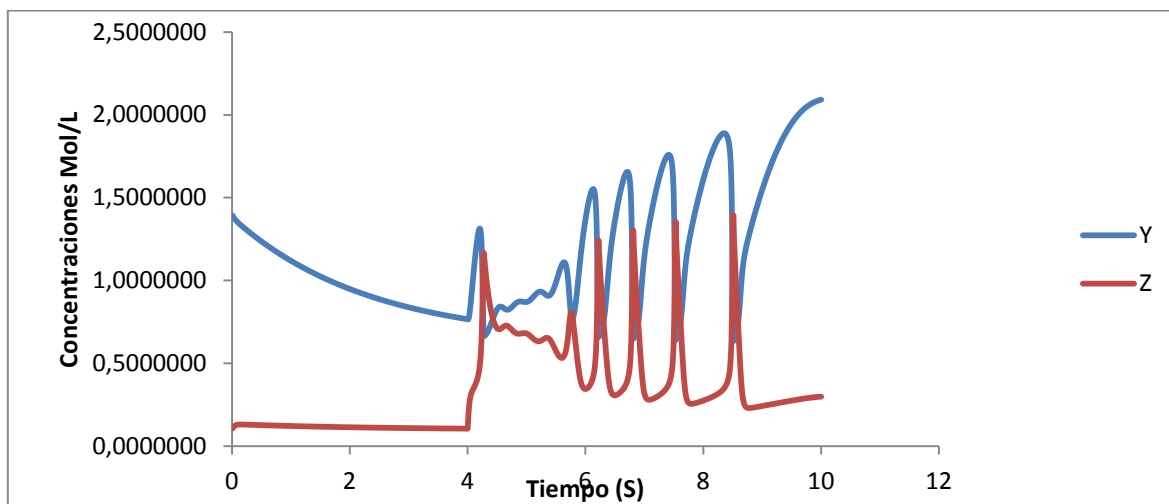
Las salidas de la FSM controlan *enables* de dispositivos, *load* de registros y *selects* de multiplexores. El diagrama de estados Moore del modelo oscilador de CA^{2+} de Dupont se muestra en la Figura 4.16.

Fig. 4.16. Diagrama de estados para el modelo del flujo de CA^{2+} usando el método de Euler.



Los estados 1,2 y 3 cargan las condiciones iniciales de los metabolitos. Después de la primera iteración estos estados se saltan, pasando del estado 25 al estado 4 con un flanco de reloj. Los resultados de la implementación hardware usando el método de Euler se muestran en la Figura 4.17:

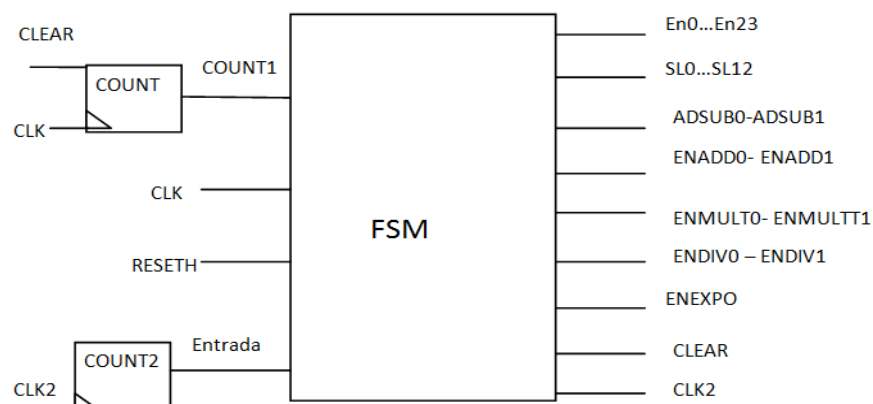
Fig.4.17. Gráfica de concentraciones para la implementación en hardware del flujo CA^{2+} usando el método de Euler.



4.2.2 Implementación en hardware del flujo de CA^{2+} usando la regla Trapezoidal

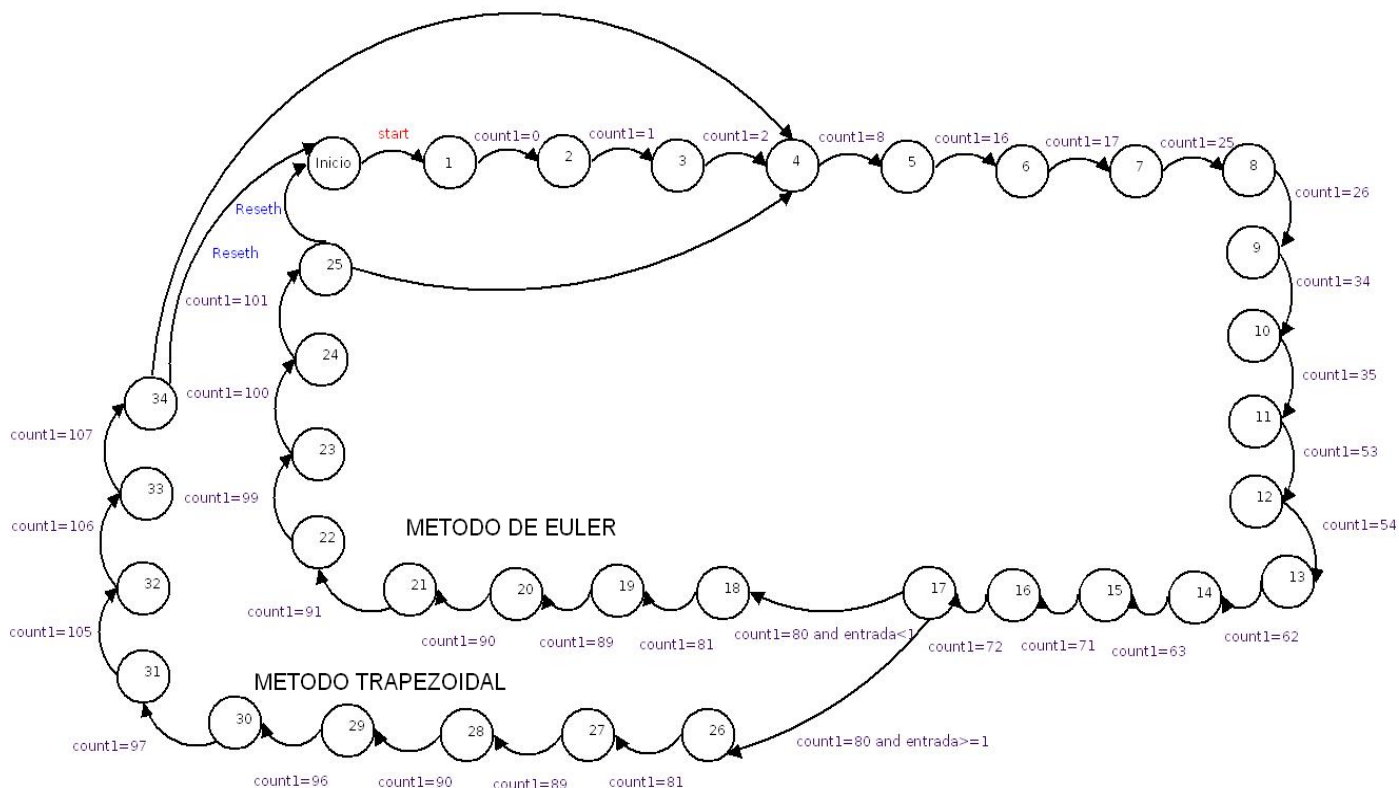
La FSM con las respectivas entradas y salidas para la implementación hardware del modelo del sistema de CA^{2+} de Dupont usando la regla trapezoidal se muestra en la Figura 4.18:

Fig.4.18. FSM para las implementaciones en hardware del modelo del flujo de CA^{2+} usando la regla trapezoidal y la regla de Simpson



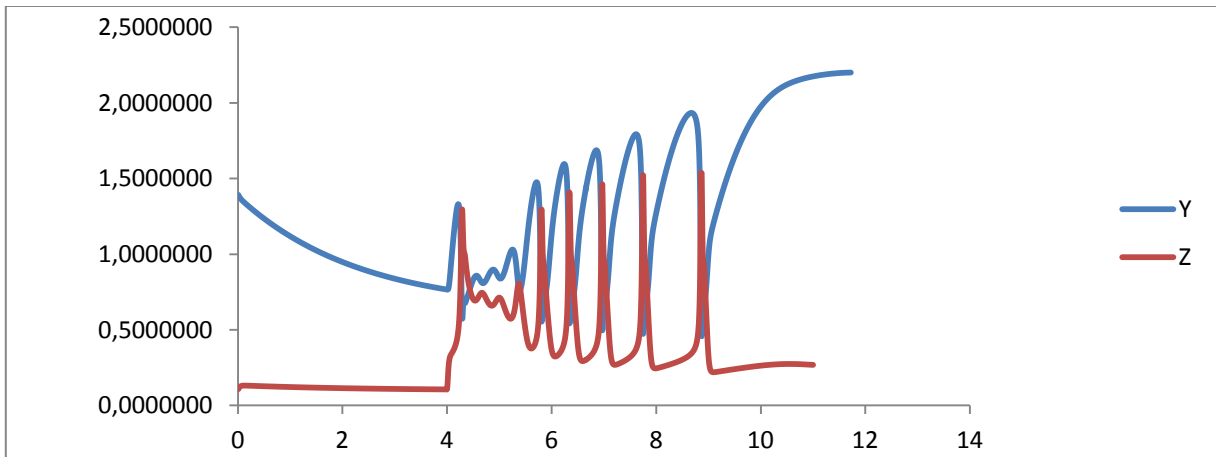
El diagrama de estados para el modelo matemático del sistema usando la regla trapezoidal se observa en la Figura 4.19.

Fig.4.19. Diagrama de estados para el modelo del flujo de CA^{2+} usando la Regla trapezoidal.



En los estados 1,2 y 3 se realiza el load de las condiciones iniciales de las concentraciones de los metabolitos, en la primera iteración las ecuaciones de diferencia son resueltas por medio del método de Euler, en las siguientes iteraciones las ecuaciones se resuelven por el método de integración de la regla trapezoidal. Los resultados obtenidos para la implementación hardware del modelo del Ca^{+2} de Dupont utilizando la regla trapezoidal se presentan en la Figura 4.20:

Fig. 4.20. Gráfica de concentraciones para la implementación en hardware del flujo de Ca^{2+} usando la regla trapezoidal.

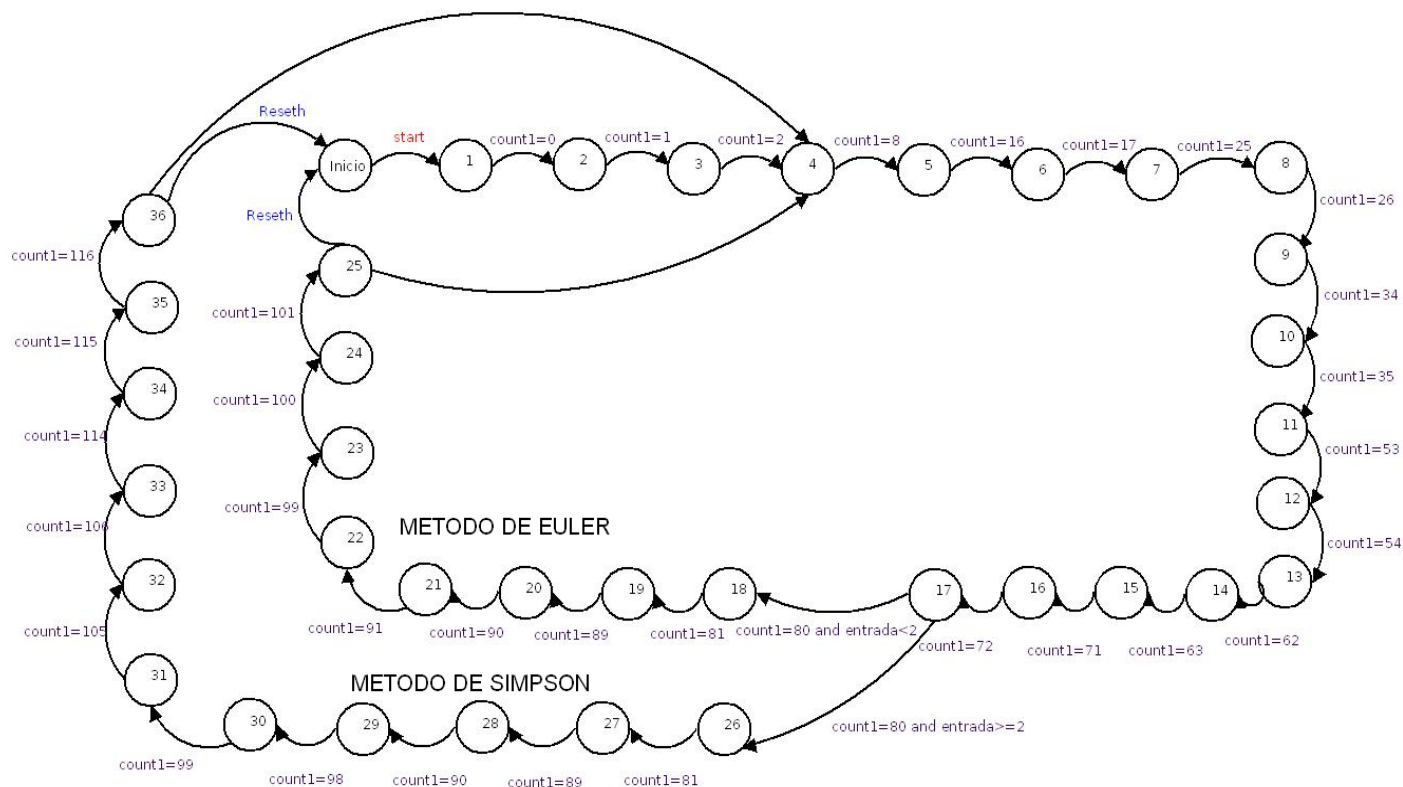


4.2.3 Implementación en hardware del flujo de Ca^{2+} usando la regla de Simpson

La FSM para la implementación en hardware del modelo del oscilador de Ca^{+2} de Dupont usando la regla de Simpson es la misma que la implementada para el modelo de este sistema usando el método trapezoidal.

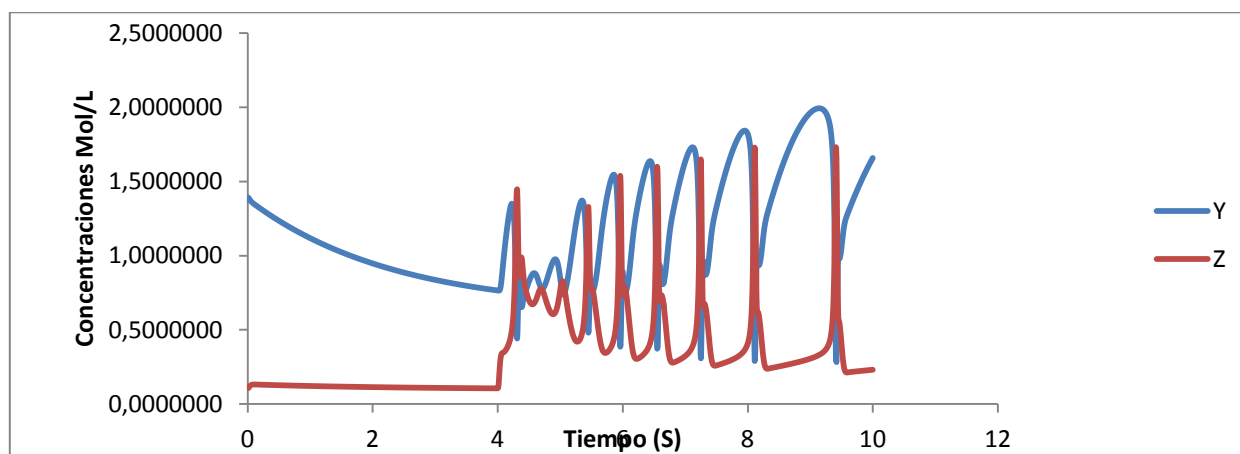
El diagrama de estados para la implementación del modelo matemático del flujo de Calcio de Dupont usando el método de integración de la regla de Simpson se muestra en la Figura 4.21. Para iteraciones mayores a dos, las ecuaciones de diferencia se resuelven por el método de Simpson y para las primeras dos iteraciones las ecuaciones son resueltas por el método de Euler.

Fig.4.21. Diagrama de estados para la implementación en hardware del modelo del flujo de CA^{2+} usando la regla de Simpson.



Los resultados de la implementación para el sistema usando el método de integración de la regla de Simpson se presentan en la Figura 4.22.

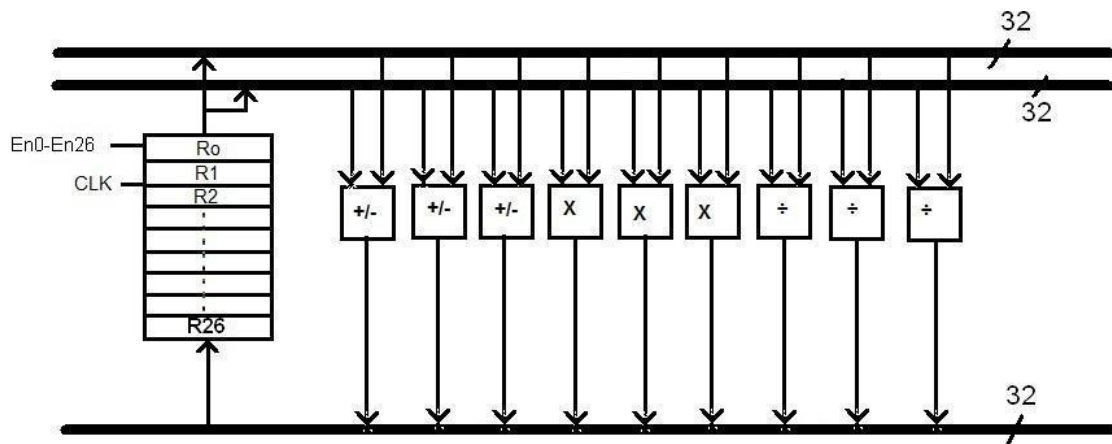
Fig. 4.22. Gráfica de concentraciones para la implementación en hardware del modelode flujo de CA^{2+} usandola regla de Simpson.



4.3 IMPLEMENTACIÓN EN HARDWARE DEL MODELO DEL OSCILADOR MITÓTICO

Para la implementación en hardware del sistema biológico “A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 Kinase” usando los métodos numéricos Euler, regla trapezoidal y regla de Simpson, se diseñó el datapath representado en la Figura 4.23.

Fig. 4.23. Datapath para las implementaciones en hardware del sistema oscilador mitótico participación de ciclina y cdc2 quinasa.

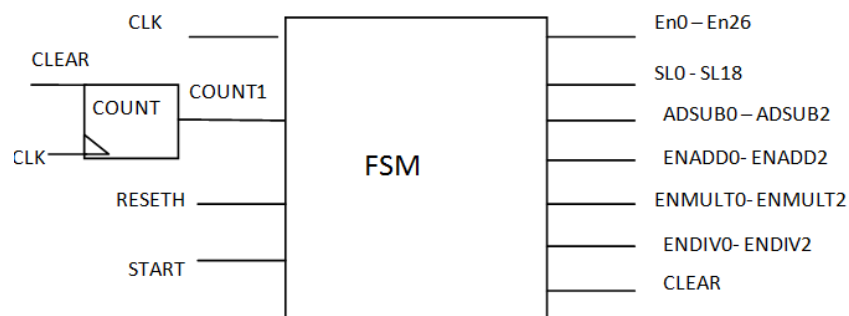


Los bloques funcionales divisor, multiplicador y sumador-restador fueron implementados usando la herramienta de QuartusII “MegaWizard- Plug in manager” usando la librería de punto flotante de altera “altera_mf”.

4.3.1 Implementación en hardware del oscilador mitótico usando el Método de Euler.

Para el modelo del sistema biológico “El oscilador mitótico participación de ciclina y cdc2 quinasa” implementado con el método de Euler, se implementó la FSM que se muestra en la Figura 4.24.

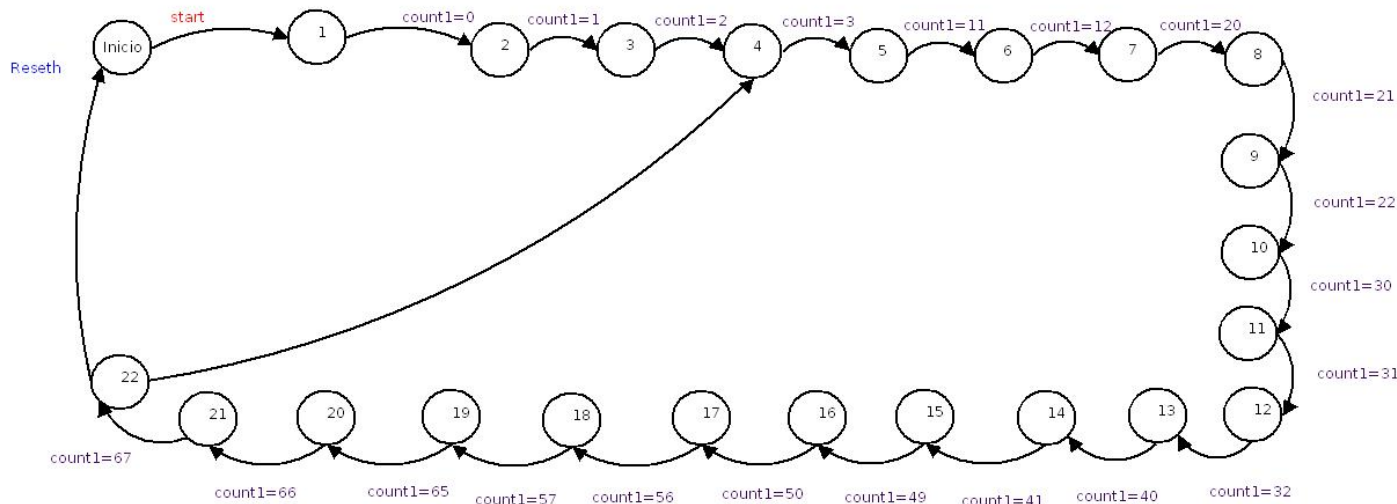
Fig. 4.24. FSM para la implementación en hardware del modelo del oscilador mitótico usando el método de Euler.



La señal de control (Count1) controla el cambio del estado actual a siguiente estado, dependiendo de la latencia de los dispositivos y las salidas de la FSM controlan *enables* de dispositivos, *load* de registros y *selects* de multiplexores. El sumador-restador utilizado tiene una latencia de 7 ciclos de reloj, el multiplicador una latencia de 5 ciclos de reloj y el divisor una latencia de 6 ciclos de reloj.

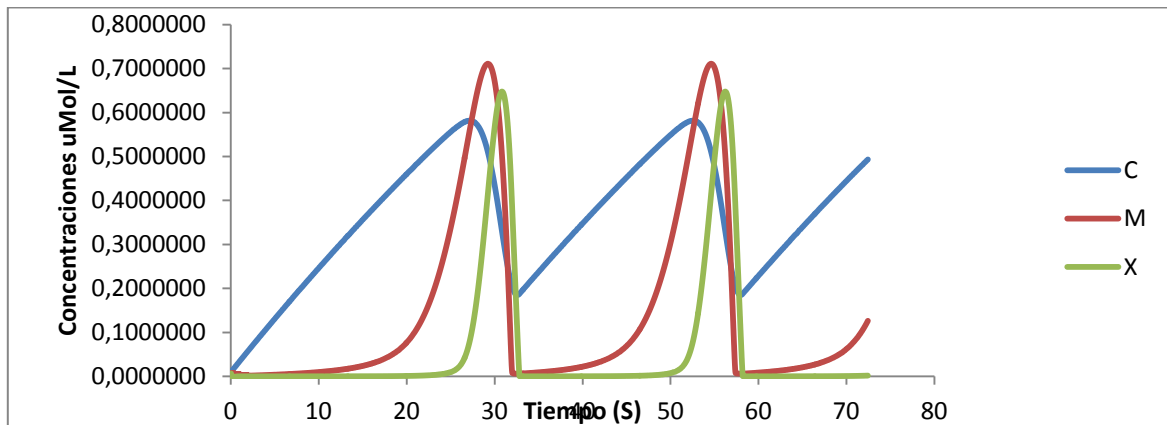
El diagrama de estados Moore del modelo oscilador mitótico usando el método de Euler se muestra en la Figura 4.25.

Fig. 4.25. Diagrama de estados para el modelo del oscilador mitótico usando el método de Euler.



En los estados 1,2 y 3 se realizan la carga de las condiciones iniciales de los metabolitos. En la Figura 4.26 se muestra el resultado de las concentraciones obtenidas en el test-bench realizado para probar la implementación hardware del modelo.

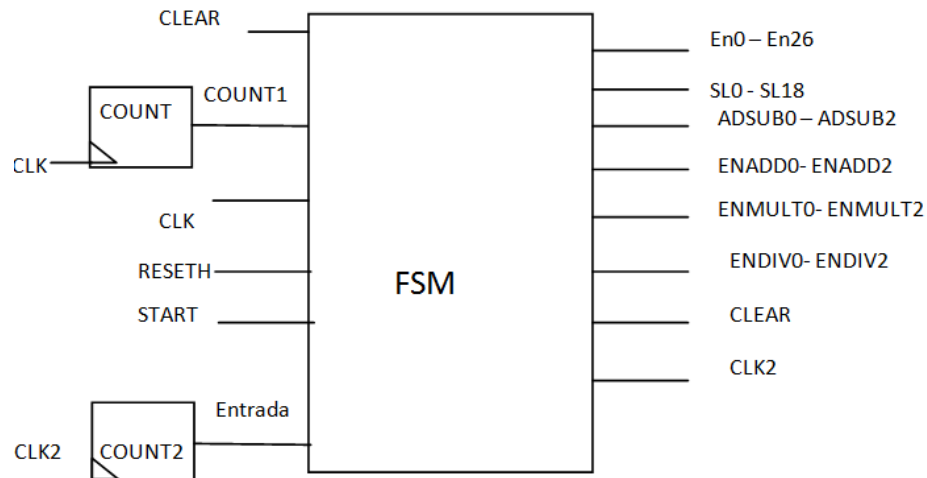
Fig. 4.26. Gráfica de concentraciones para la implementación hardware del oscilador mitótico usando el método de Euler.



4.3.2 Implementación en hardware del oscilador mitótico usando la regla Trapezoidal

La máquina de estados para la implementación hardware del modelo matemático del oscilador mitótico de Goldbeter usando la regla trapezoidal se muestra en la Figura 4.27.

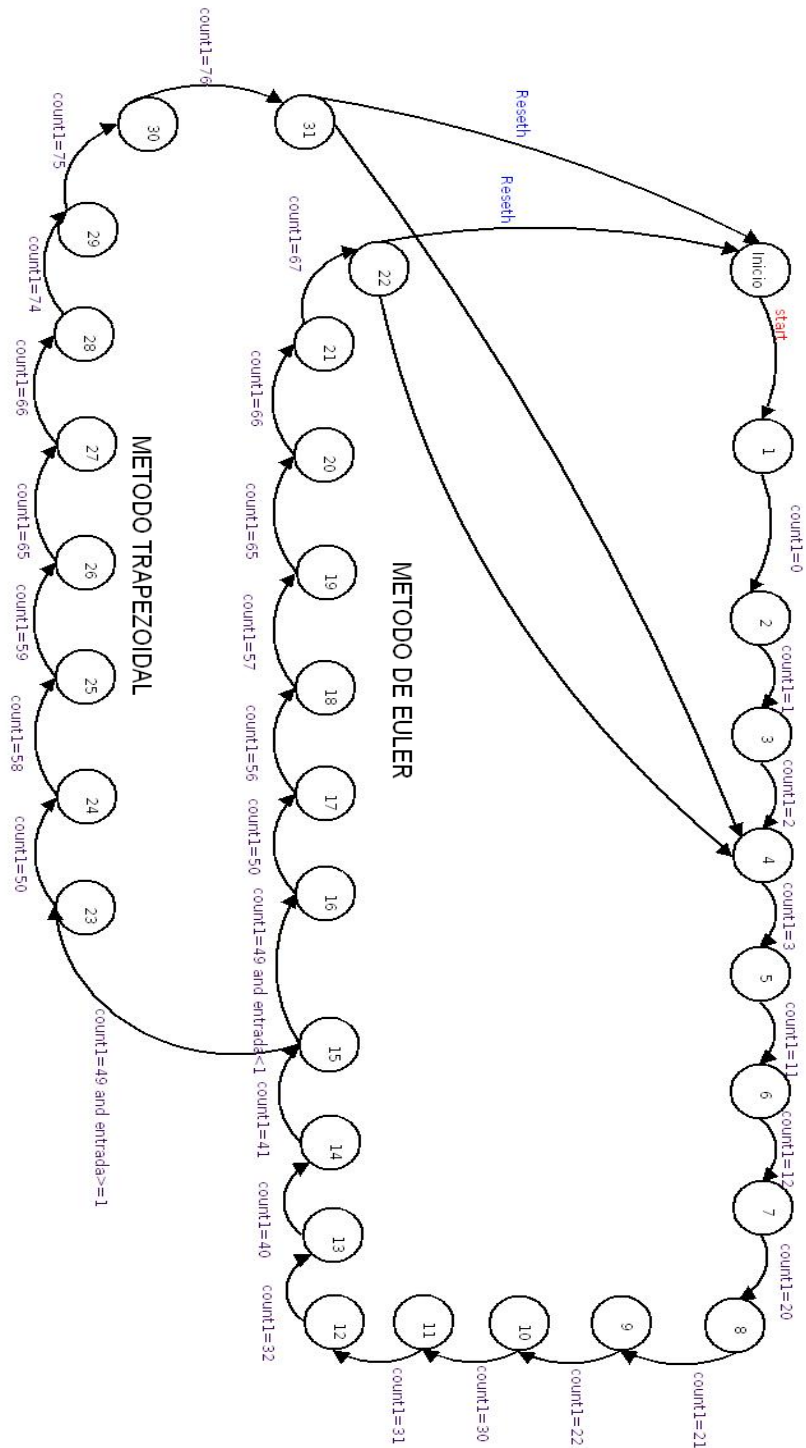
Fig. 4.27. FSM para las implementaciones en hardware del modelo del oscilador mitótico usando la regla trapezoidal y la regla de Simpson.



Las señales de entrada "Count1" y "Entrada" controlan el paso de estado actual al siguiente estado dependiendo de las iteraciones realizadas o de la latencia de los bloques funcionales.

El diagrama de estados para la implementación hardware del sistema usando el método de integración de la regla trapezoidal se presenta en la Figura 4.28.

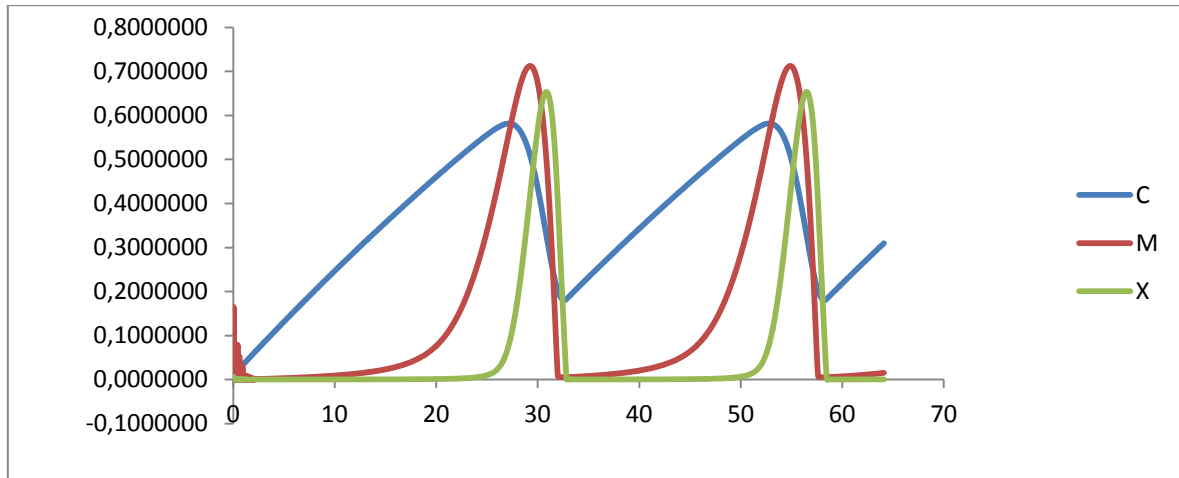
Fig. 4.28. Diagrama de estados para el modelo del oscilador mitótico usando la regla trapezoidal.



Los estados 1,2 y 3 se utilizan para cargar las condiciones iniciales de los metabolitos, en la primera iteración las ecuaciones diferenciales se resuelven por medio del método de Euler y para las siguientes iteraciones las ecuaciones se resuelven por medio del método trapezoidal.

En la Figura 4.29 se presenta el resultado de las concentraciones de los metabolitos del oscilador mitótico de Goldbeter implementado en hardware usando el método de integración de la regla trapezoidal.

Fig. 4.29. Gráfica de concentraciones para la implementación en hardware del oscilador mitótico usando la regla trapezoidal.

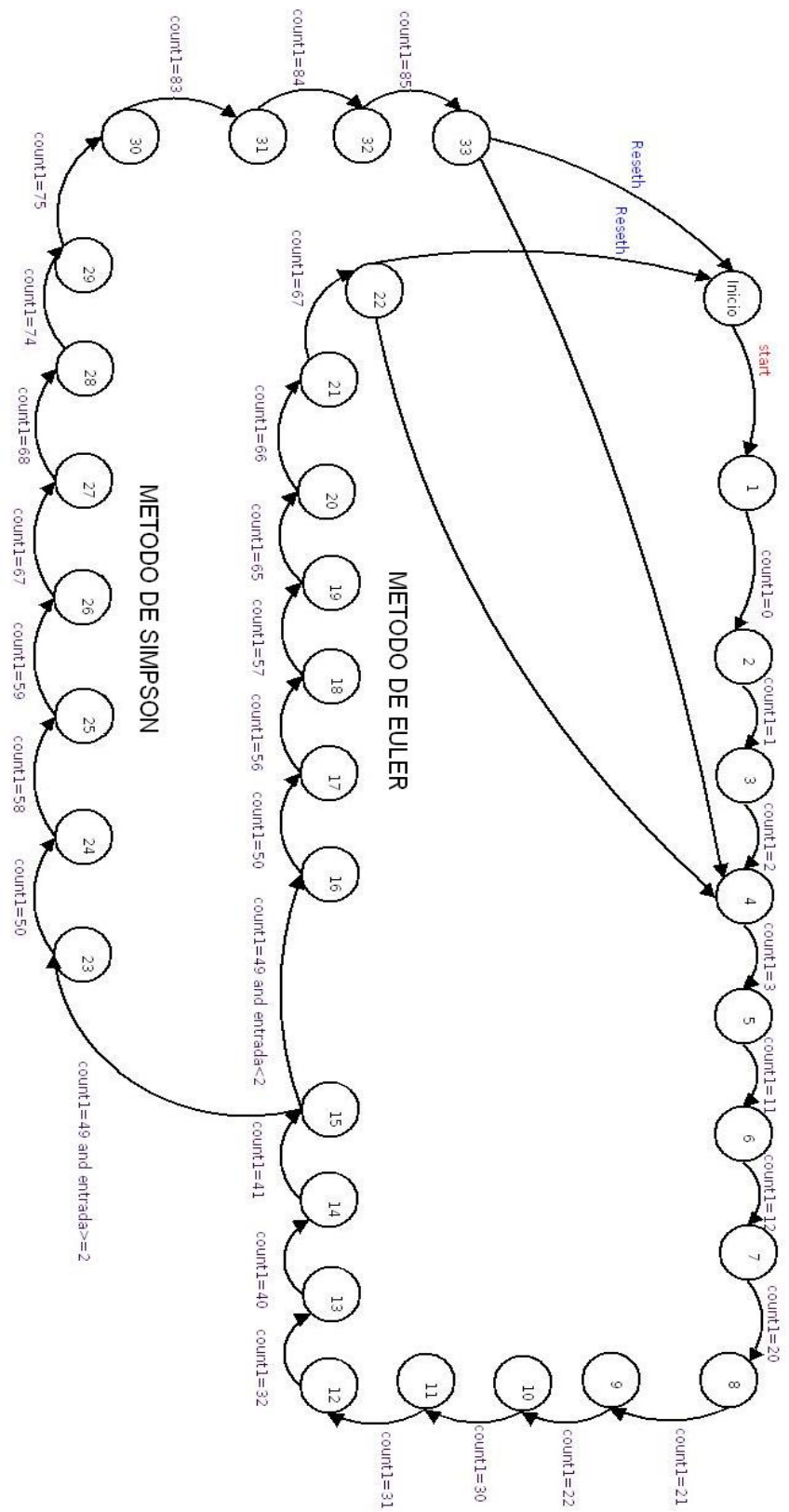


4.3.3 Implementación en hardware del oscilador mitótico usando la regla de Simpson

La máquina de estados que se utiliza para la implementación en hardware del modelo matemático del sistema, usando el método de integración de la regla de Simpson, es la misma implementada para el oscilador mitótico de Goldbeter usando el método de integración de la regla trapezoidal.

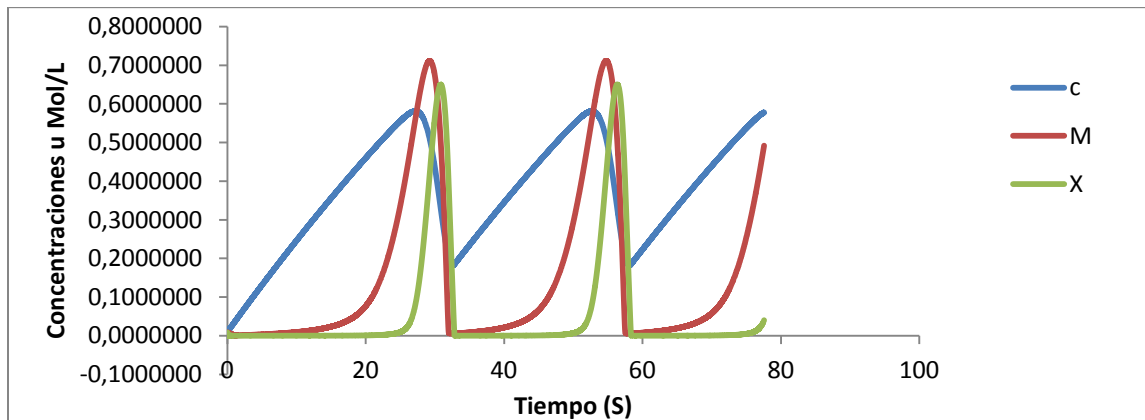
El diagrama de estados correspondiente a la implementación en hardware del sistema utilizando el método de integración de la regla de Simpson se muestra en la Figura 4.30.

62



La gráfica obtenida de la implementación en hardware del modelo del oscilador mitótico de Goldbeter, para las concentraciones de la ciclina, la cdc2 quinasa y la quinasa proteasa se presenta en la Figura 4.31:

Fig. 4.31. Gráfica de concentraciones para la implementación en hardware del oscilador mitótico usando la regla de Simpson.



CAPÍTULO 5

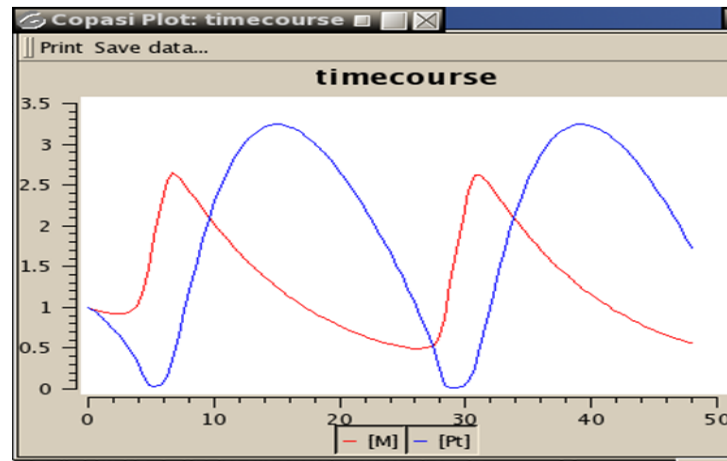
ANÁLISIS Y DISCUSIÓN DE RESULTADOS

Con el fin de evaluar el desempeño de las implementaciones hardware realizadas en la herramienta Modelsim de Quartus, en este capítulo se presenta una comparación de estas implementaciones frente a las implementaciones en software realizadas en la herramienta Simulink de Matlab. Las simulaciones software fueron realizadas en un computador con un procesador Intel core I3-2120 que trabaja a una velocidad del reloj de 3.3 GHz con 2 nucleos y 4 subprocesos.

5.1 ANÁLISIS Y RESULTADOS PARA EL RITMO CIRCADIANO.

El resultado para las concentraciones de los metabolitos M y P ofrecido por la base de datos biológica Biomodels DataBase se muestra en la Figura 5.1. Se observa que los resultados obtenidos en las implementaciones software y hardware tienen similitud en amplitud y en frecuencia

Fig.5.1. Resultado de las concentraciones para los metabolitos del ritmo circadiano brindado por la base de datos biológica Biomodels Database.



La implementación hardware para el ritmo circadiano se simuló con un tiempo de muestreo de 0,01 y un reloj 10 ns= 100MHz, generando 5713 iteraciones equivalentes a 57,13 sg en un tiempo de simulación de 16ms. El tiempo de simulación obtenido para el ritmo circadiano en Simulink de Matlab se muestra a continuación:

```
>> tic
>> sim(modelname, 'STOPTIME', '57.13')
```

```
toc
Simulink.SimulationOutput:
```

```
    M: [5714x1 double]
    P: [5714x1 double]
   DP: [5714x1 double]
   DM: [5714x1 double]
    T: [5714x1 double]
 tout: [1000x1 double]
```

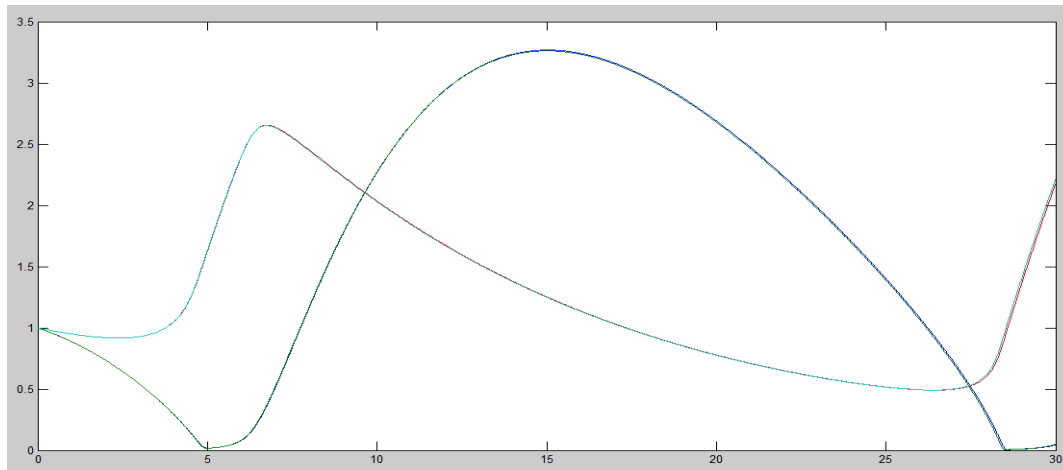
```
Elapsed time is 11.933896 seconds.
```

Teniendo en cuenta los resultados de tiempo de simulación de ambas implementaciones se puede concluir que la implementación hardware para el sistema del ritmo circadiano es aproximadamente 746 veces más rápido que la implementación en software.

La implementación en hardware para el modelo del ritmo circadiano usando el Método de Euler se simuló con un tiempo de muestreo de 0,01 para simular desde 0sg hasta 30sg generando 3000 iteraciones y usando un reloj de 10ns.

En la Figura 5.2 se presentan las concentraciones para P y M obtenidas por Simulink y Modelsim; se observa que las graficas se encuentran superpuestas, lo que quiere decir que los resultados obtenidos son muy similares.

Fig. 5.2. Gráfica de concentraciones para los metabolitos P y M del ritmo circadiano implementado en software y hardware usando el método de Euler.



Para el cálculo de la exactitud de la implementación en hardware (en adelante x'), en comparación con la implementación en software (en adelante x), se define el error de dos maneras:

- i. Error absoluto = $\|x' - x\|$
- ii. Error relativo = $\frac{\|x' - x\|}{\|x\|}$

En donde $\|\cdot\|$ se considera como la norma infinito:

$$\|C\| = \max_{i=1 \dots m} \sum_{j=1}^n |C_{ij}|$$

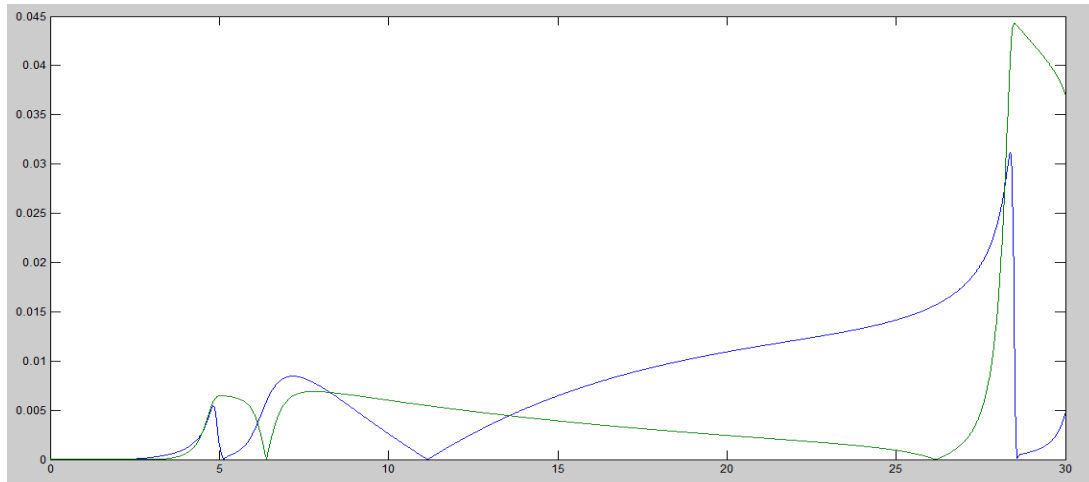
El error absoluto y relativo para la implementación del ritmo circadiano usando el método de Euler para cada metabolito es:

Error absoluto de P = 0,0312 Error relativo de P = 0,00954 ~ 1%

Error absoluto de M = 0,0443 Error relativo de M = 0,0166 ~ 1.6%

La gráfica que representa el error absoluto para las concentraciones de los resultados obtenidos en Modelsim con los obtenidos en Simulink se presenta en la Figura 5.3.

Fig. 5.3. Gráfica del error absoluto para las concentraciones de los metabolitos del ritmo circadiano implementado en hardware usando el método de Euler.

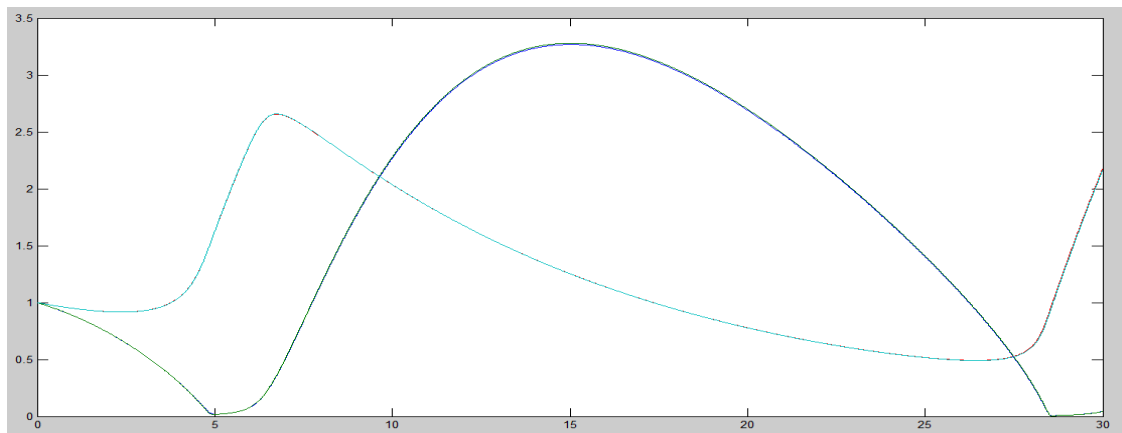


En la figura anterior se observa que la implementación en hardware presenta errores pequeños en exactitud respecto a la implementación en software, también se observa que los errores más altos se presentan cuando la onda cambia de concavidad.

Para la simulación de la implementación en hardware del ritmo circadiano usando la regla trapezoidal se usó un reloj de 10ns, un tiempo de muestreo de 0,01 generando 3000 muestras para una simulación de 0sg a 30 sg.

El resultado obtenido en la simulación hardware versus el resultado obtenido en la simulación software se muestra en la Figura 5.4, donde se observa que los dos resultados se superponen.

Fig. 5.4. Gráfica de concentraciones para P y M del ritmo circadiano implementado en software y hardware usando el método trapezoidal.



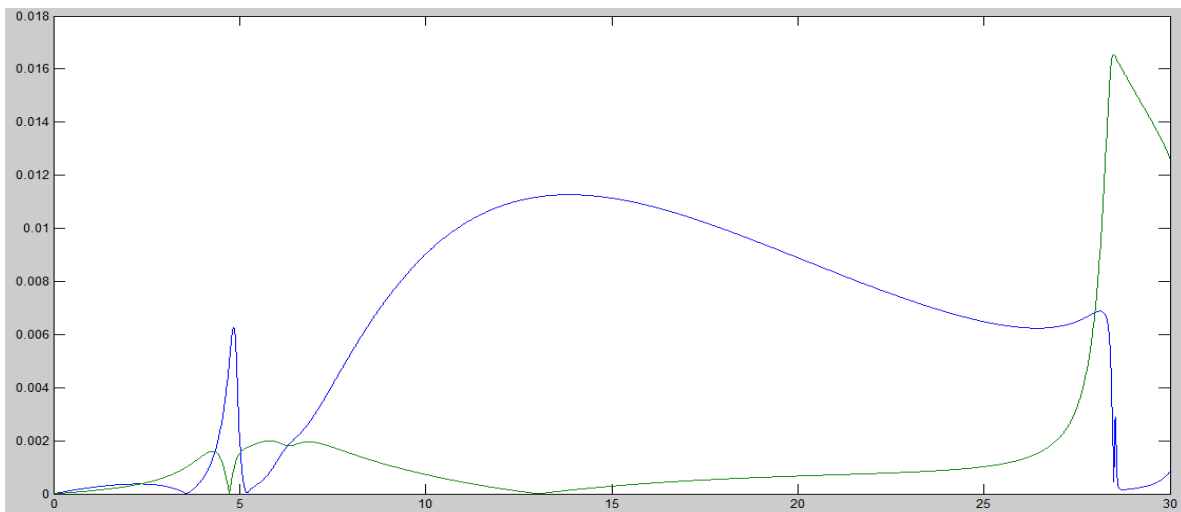
Los errores absolutos y relativos para las concentraciones de los metabolitos obtenidos en la implementación hardware usando la regla trapezoidal son:

Error absoluto de P = 0,0113 Error relativo de P= 0,00345 ~ 0.3%

Error absoluto de M = 0,0166 Error relativo de M= 0,00625 ~ 0.6%

La gráfica de los errores absolutos para las concentraciones de los metabolitos P y M se muestra en la Figura 5.5.

Fig. 5.5. Gráfica del error absoluto para las concentraciones de los metabolitos del ritmo circadiano implementado en hardware usando el método trapezoidal.

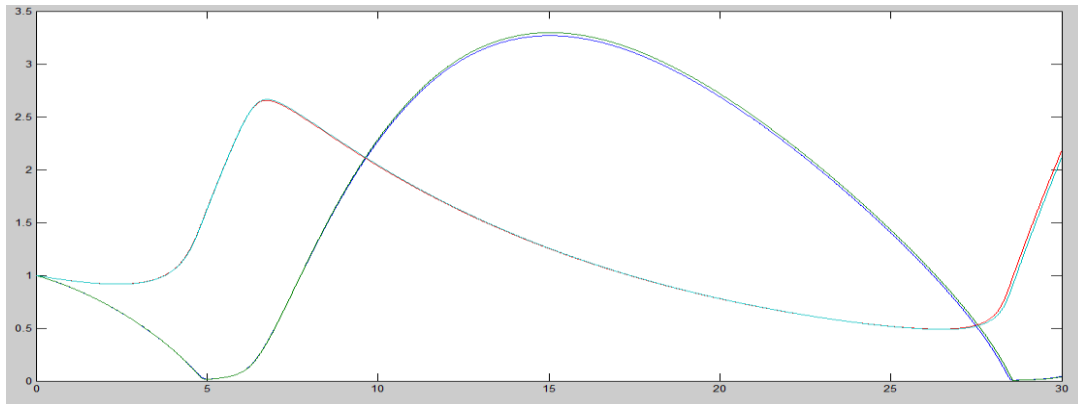


Se observa que el error obtenido por medio del método de integración de la regla trapezoidal es menor que el obtenido por medio del método numérico de Euler.

La implementación en hardware para el sistema del ritmo circadiano usando la regla de Simpson se simuló usando un reloj de 10ns y un tiempo de muestreo de 0,01 generando 3000 muestras para una simulación de 0sg a 30sg.

En la Figura 5.6 se muestran los resultados obtenidos en hardware versus los resultados obtenidos en software y se puede observar la similitud en fase y amplitud de estos dos resultados.

Fig. 5.6. Gráfica de concentraciones para P y M del ritmo circadiano implementado en software y hardware usando la regla de Simpson.



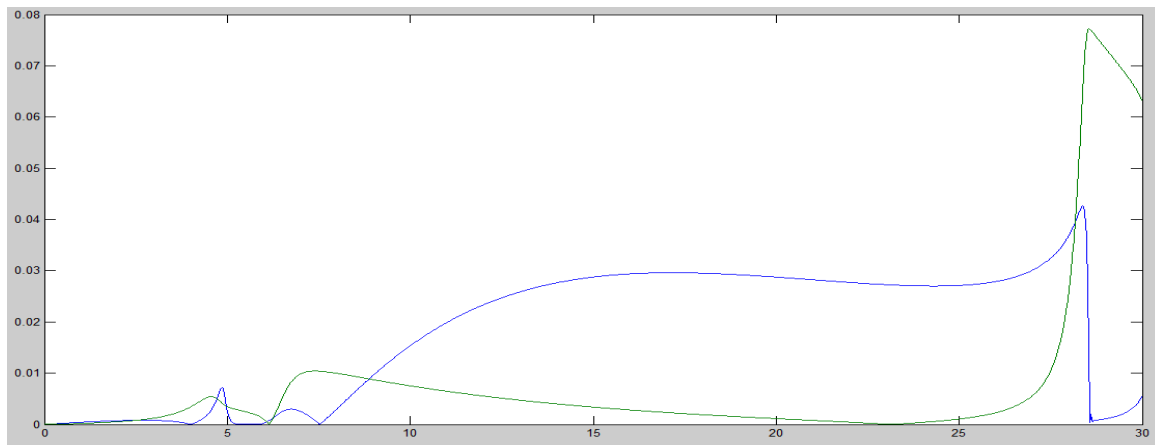
Los errores absolutos y relativos obtenidos para la implementación en hardware del sistema usando la regla de Simpson son:

Error absoluto de P= 0,0427 Error relativo de P = 0,013 ~ 1.3%

Error absoluto de M= 0,0773 Error relativo de M= 0,0291~ 2.9%

Los errores absolutos para las concentraciones de los metabolitos se presentan en la Figura 5.7:

Fig. 5.7. Gráfica del error absoluto para las concentraciones de los metabolitos del ritmo circadiano implementado en hardware usando la regla de Simpson



Para comparar la eficiencia se consideraron los valores máximos, mínimos, promedios y amplitudes para las concentraciones obtenidas en las implementaciones hardware usando los métodos numéricos Euler, regla trapezoidal y regla de Simpson. En la Tabla 5.1 se presentan los valores ya mencionados para cada implementación.

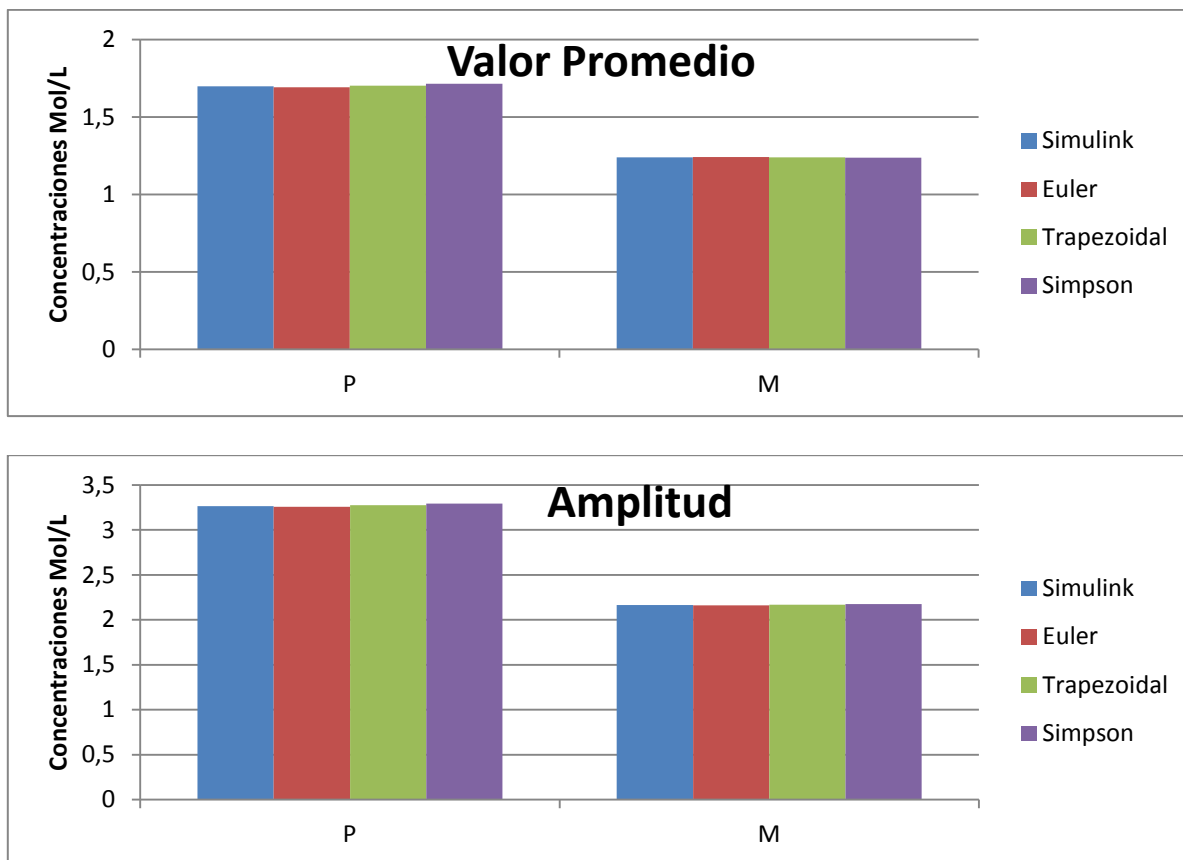
Tabla.5.1. Tabla de valores máximos, mínimos, promedios y amplitudes para las implementaciones hardware del ritmo circadiano usando los tres métodos numéricos y la implementación software.

	SIMULINK T=0,01				METODO DE EULER T=0,01			
Metabolito	V. Máximo	V. Mínimo	V. Promedio	Amplitud	V. Máximo	V. Mínimo	V. Promedio	Amplitud
P	3,2679	0,006	1,6965	3,2619	3,2614	0,0056	1,6912	3,2558
M	2,6553	0,4917	1,2399	2,1636	2,6509	0,492	1,2406	2,1589

REGLA TRAPEZOIDAL T=0,01				REGLA DE SIMPSON T=0,01			
V. Máximo	V. Mínimo	V. Promedio	Amplitud	V. Máximo	V. Mínimo	V. Promedio	Amplitud
3,279	0,0051	1,7026	3,2739	3,2967	0,0039	1,7147	3,2928
2,6573	0,4902	1,2387	2,1671	2,6637	0,4883	1,2369	2,1754

Para mostrar la eficiencia de las implementaciones en hardware con los tres métodos numéricos mencionados, en la Figura 5.8 se presentan los diagramas de barras que se obtienen a partir de los valores consignados en la Tabla 5.1.

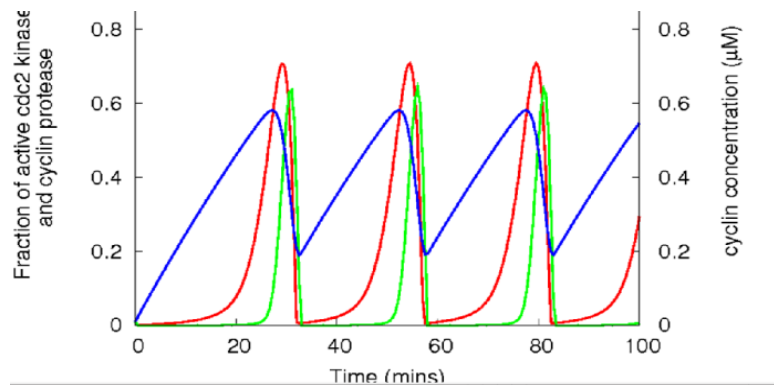
Fig. 5.8. Diagrama de barras comparativo de la eficiencia de las implementaciones hardware según los tres métodos numéricos versus implementación software del ritmo circadiano.



5.2 ANÁLISIS Y RESULTADOS PARA EL OSCILADOR MITOTICO

En la Figura 5.9 se muestra el resultado ofrecido por la base de datos biológica para el modelo del oscilador mitótico de Goldbeter. En la figura se observa que los resultados obtenidos en las implementaciones realizadas en software y en hardware para este sistema tienen similitudes en amplitud y frecuencia.

Fig.5.9.Resultados de las concentraciones de los metabolitos para el oscilador mitótico aportada por la base de datos biológica.



La implementación en hardware para el oscilador mitótico de Goldbeter se simuló con un reloj de 10ns= 100MHz y un tiempo de muestreo de 0.01 para una simulación de 0sg a 72,45sg, generando 7245 iteraciones en un tiempo de 10ms. El tiempo de simulación para la implementación en software del mismo sistema, realizada en Simulink con un tiempo de muestreo de 0,01 para una simulación de 0sg a 72,45sg, es:

```
>> tic
>> sim(modelname, 'STOPTIME', '72.45')

Simulink.SimulationOutput:

    C: [7246x1 double]
    M: [7246x1 double]
    X: [7246x1 double]
    DM: [7250x1 double]
    DX: [7250x1 double]
    DC: [7246x1 double]
    TI: [7246x1 double]
    tout: [1000x1 double]

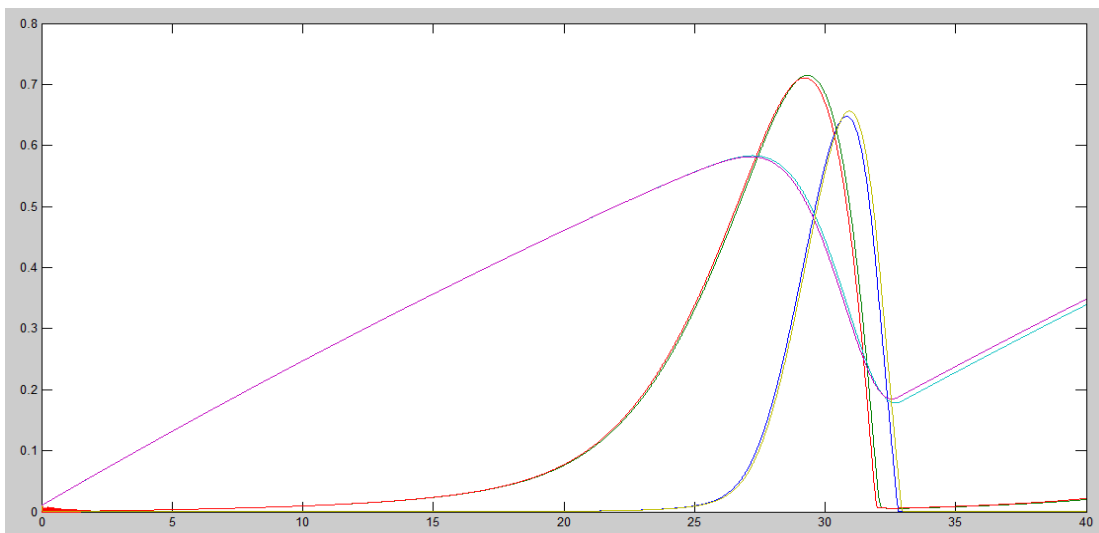
>> toc
Elapsed time is 9.381111 seconds.
```

De acuerdo a los tiempos obtenidos en hardware y software del sistema, se concluye que la implementación hardware es aproximadamente 938 veces más rápida que la implementación software.

En la simulación de la implementación en hardware del oscilador mitótico usando el método de Euler se usó un reloj de 10ns y un tiempo de muestreo de 0,01 generando 4000 muestras para una simulación de 0sg a 40sg.

En la figura 5.10 se muestran los resultados de las concentraciones de los metabolitos C, M y X obtenidos en la implementación hardware usando el método de Euler y en la implementación software realizada en Simulink de Matlab. Se visualizan los resultados para las dos implementaciones superpuestos.

Fig.5.10. Gráfica de concentraciones para los metabolitos C, M y X del oscilador mitótico implementado en software y hardware usando el método de Euler.



Los errores absolutos y relativos para las concentraciones obtenidas en la implementación en hardware del oscilador mitótico usando el método de Euler son:

Error absoluto de C= 0.0118

Error relativo de C= 0,0202 ~ 2%

Error absoluto de M= 0.0592

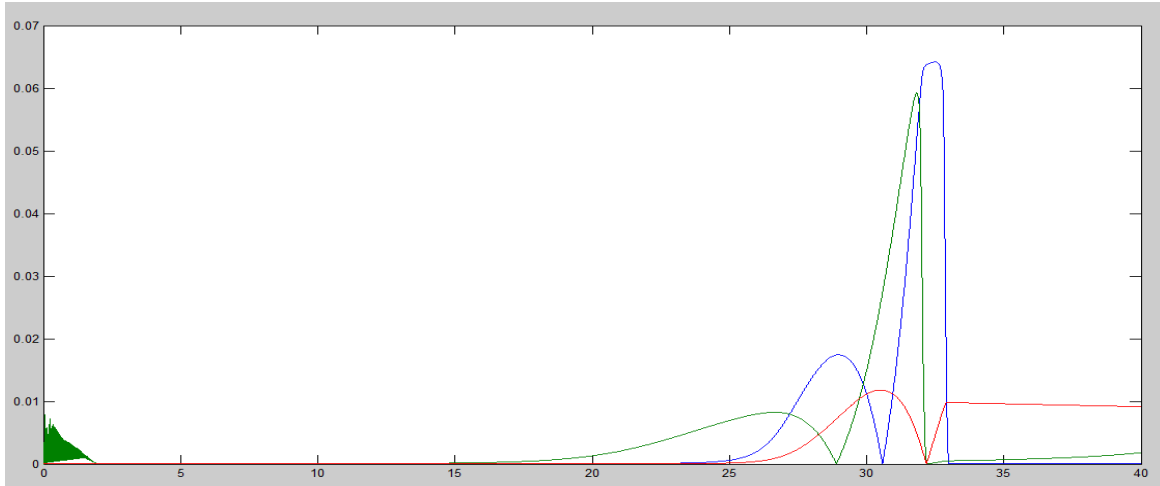
Error relativo de M=0,082797 ~ 8%

Error absoluto de X= 0.0642

Error relativo de X=0,09782~ 9%

La gráfica del error absoluto para el modelo implementado en hardware se muestra en la Figura 5.11.

Fig.5.11. Gráfica del error absoluto para las concentraciones de los metabolitos del oscilador mitótico implementado en hardware usando el método de Euler.

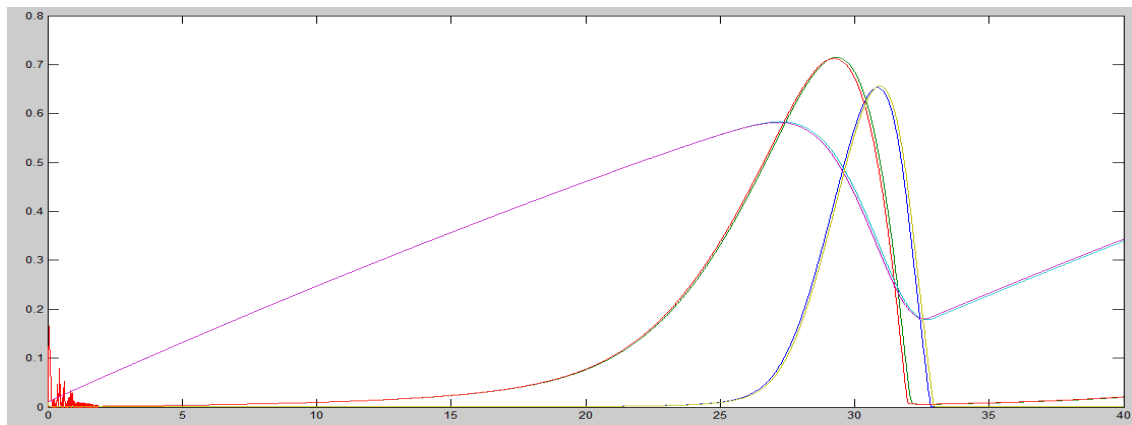


En la figura anterior se observa que la implementación en hardware usando el método de Euler presenta errores pequeños respecto a la implementación en software, también se observa que el error se hace mayor cuando la onda cambia abruptamente de concavidad.

Para la simulación de la implementación en hardware del oscilador mitótico de Goldbeter usando el método de integración de la regla trapezoidal se usó un reloj de 10ns y un tiempo de muestreo de 0,01 generando 4000 muestras para una simulación desde 0sg a 40sg.

La Figura 5.12 muestra los resultados de las concentraciones de los metabolitos obtenidos en la implementación hardware usando la regla trapezoidal y los resultados obtenidos en Simulink de Matlab. Se observa una similitud de amplitud y fase para los dos resultados.

Fig. 5.12. Gráfica de concentraciones para C, M y X del oscilador mitótico implementado en software y hardware usando el método trapezoidal.



Los errores absolutos y relativos para la implementación del sistema usando la regla trapezoidal son:

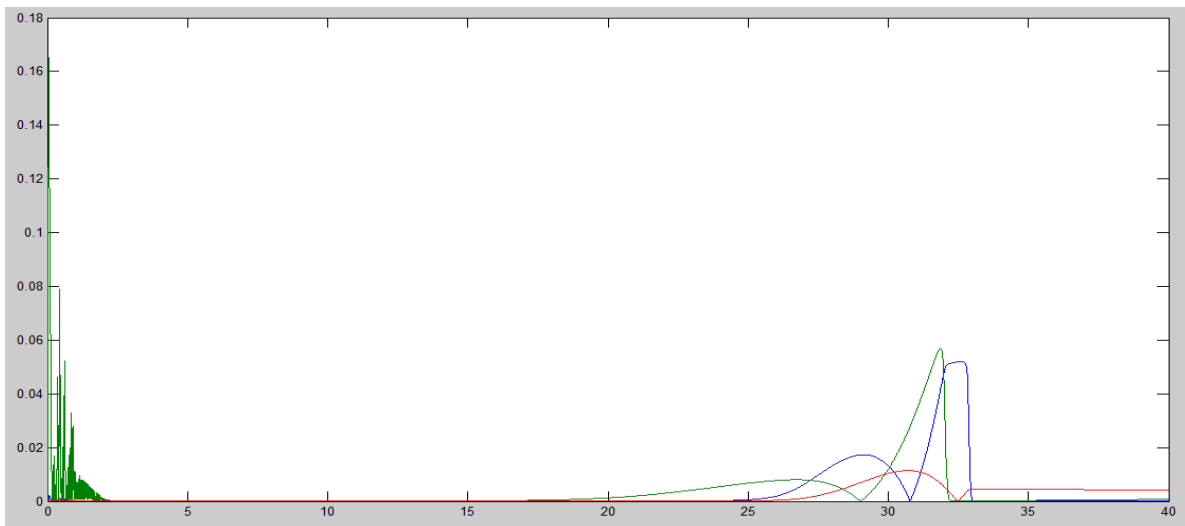
Error absoluto de C= 0.0115 Error relativo de C= 0,01971 ~ 1.9%

Error absoluto de M =0,1651 Error relativo de M=0,2309 ~ 23%

Error absoluto de X=0.0521 Error relativo de X= 0,0793 ~ 7,93%

La gráfica de los errores absolutos para las concentraciones de C, M y X se muestra en la Figura 5.13.

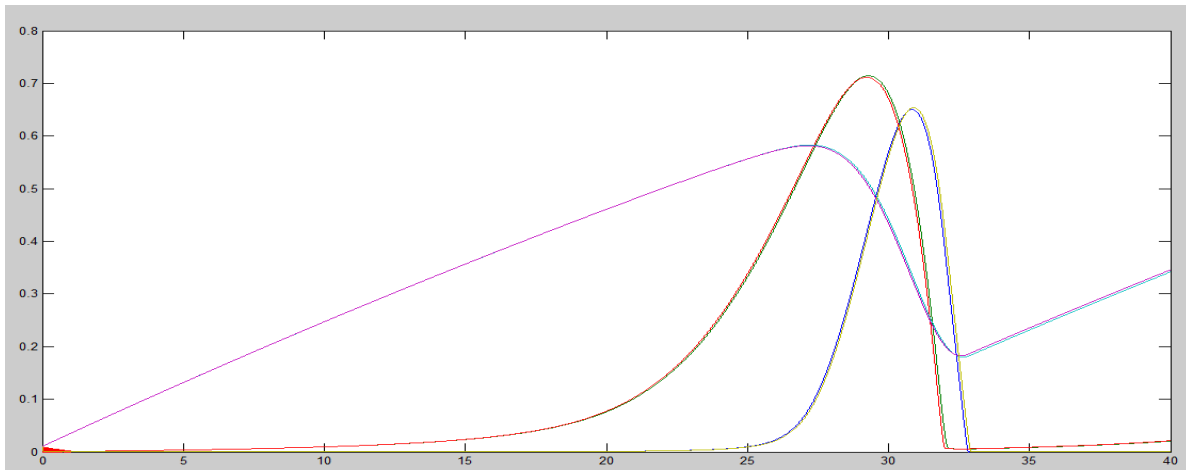
Fig.5.13. Gráfica del error absoluto para las concentraciones de los metabolitos del oscilador mitótico implementado en hardware usando el método trapezoidal.



En la anterior figura se observa que la implementación presenta errores muy bajos para las concentraciones de C y de X, la concentración del metabolito M presenta un mayor error debido a los picos transitorios obtenidos en la primera parte de la simulación.

Para la simulación de la implementación del modelo del oscilador mitótico usando la regla de Simpson se usó un reloj de 10ns y un tiempo de muestreo de 0,005 ya que con un tiempo de muestreo menor el sistema se desestabilizaba. Para una simulación desde 0sg a 40 sg se generaron 8000 muestras, en la Figura 5.14 se muestra el resultado obtenido para las concentraciones de los metabolitos en la implementación realizada en Simulink y la implementación hardware del sistema usando la regla de Simpson.

Fig. 5.14. Gráfica de concentraciones para C, M y X del oscilador mitótico implementado en software y hardware usando la regla de Simpson con un tiempo de muestreo de 0,005.



Los errores absolutos y relativos para las concentraciones obtenidas en la implementación hardware usando la regla de Simpson son:

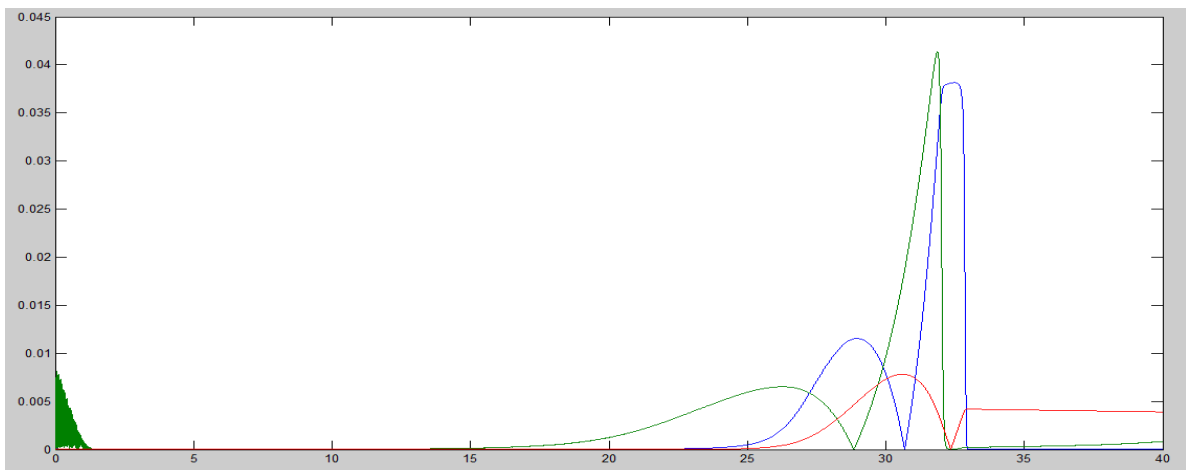
Error absoluto de C= 0,0078 Error relativo de C=0,0134 ~ 1.3%

Error absoluto de M=0,0414 Error relativo de M=0,058 ~ 5.8%

Error absoluto de X= 0.0382 Error relativo de X=0,0595~ 5.9%

La gráfica de los errores absolutos para las concentraciones de los metabolitos se presenta en la Figura 5.15.

Fig.5.15.Gráfica del error absoluto para las concentraciones de los metabolitos del oscilador mitótico implementado en hardware usando la regla de Simpson con un tiempo de muestreo de 0,005.



En la anterior gráfica se observan errores pequeños con respecto a la implementación realizada en Simulink, también se puede observar que los transitorios y los errores disminuyen al reducir el tiempo de muestreo. Al igual que en las anteriores implementaciones los errores son mayores cuando en la onda hay un cambio abrupto de concavidad.

La Tabla 5.2 presenta los valores máximos, mínimos, promedios y amplitudes para las implementaciones en Simulink y en Modelsim con los métodos de Euler y trapezoidal para un tiempo de muestreo de 0,001.

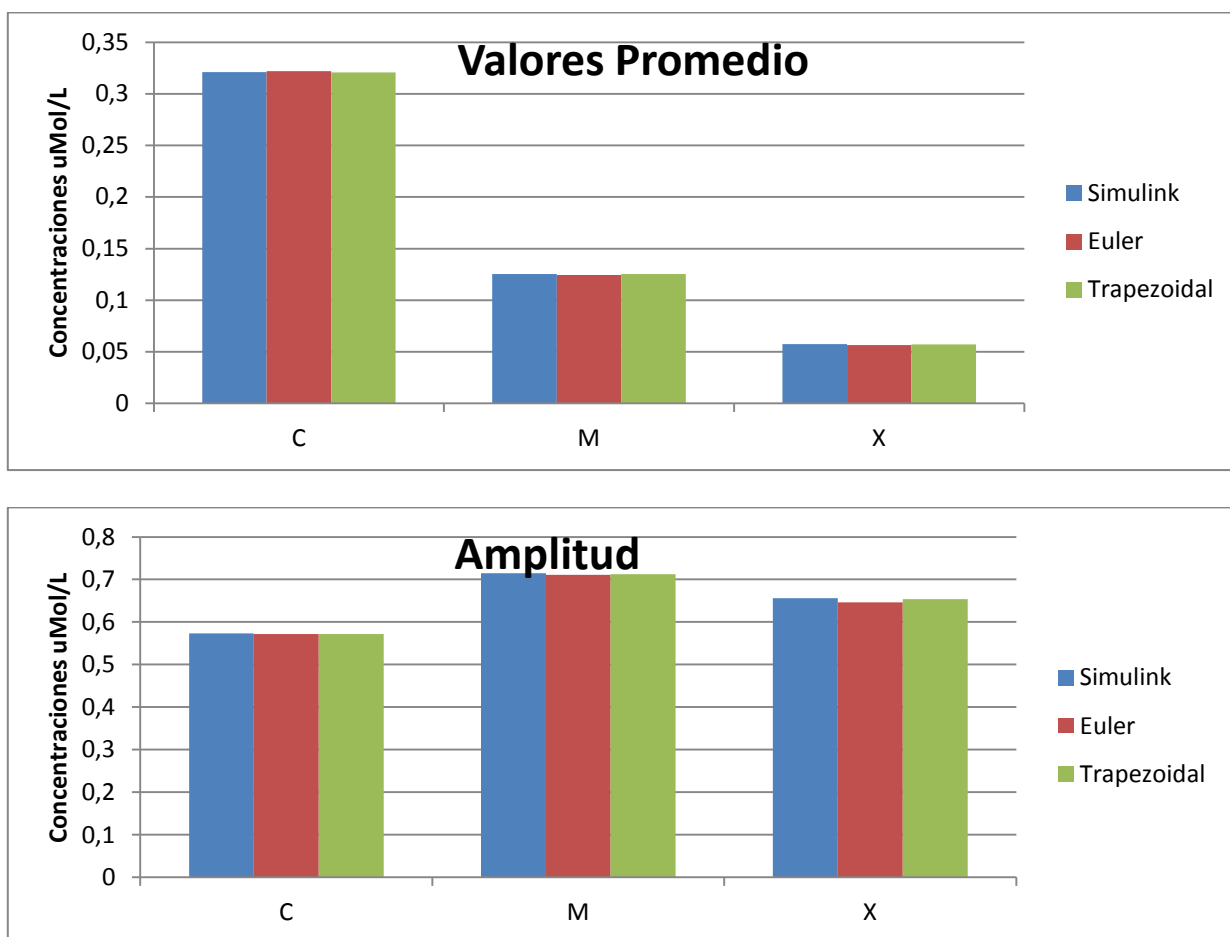
Tabla.5.2. Tabla de valores máximos, mínimos, promedios y amplitudes para los resultados obtenidos en las implementaciones del oscilador mitótico usando el método de Euler, regla trapezoidal y Simulink.

	SIMULINK T=0,01				METODO DE EULER T=0,01			
Metabolito	V. Máximo	V. Mínimo	V. Promedio	Amplitud	V. Máximo	V. Mínimo	V. Promedio	Amplitud
C	0,5833	0,01	0,321	0,5733	0,5816	0,01	0,3218	0,5716
M	0,715	2,25E-04	0,1253	0,714775	0,7109	0	0,1245	0,7109
X	0,6563	2,75E-06	0,0574	0,656297	0,6465	0	0,0564	0,6465

REGLA TRAPEZOIDAL T=0,01			
V. Máximo	V. Mínimo	V. Promedio	Amplitud
0,5818	0,01	0,3208	0,5718
0,7125	0	0,1252	0,7125
0,6535	0	0,057	0,6535

Para observar mejor la eficiencia de las implementaciones en hardware del oscilador mitótico de Goldbeter usando el método de Euler y la regla trapezoidal, se realizó un diagrama de barras que se presenta en la Figura 5.16 a partir de los resultados de la Tabla 5.2.

Fig.5.16. Diagrama de barras comparativo de la eficiencia para las implementaciones hardware usando los métodos numéricos Euler y regla trapezoidal.



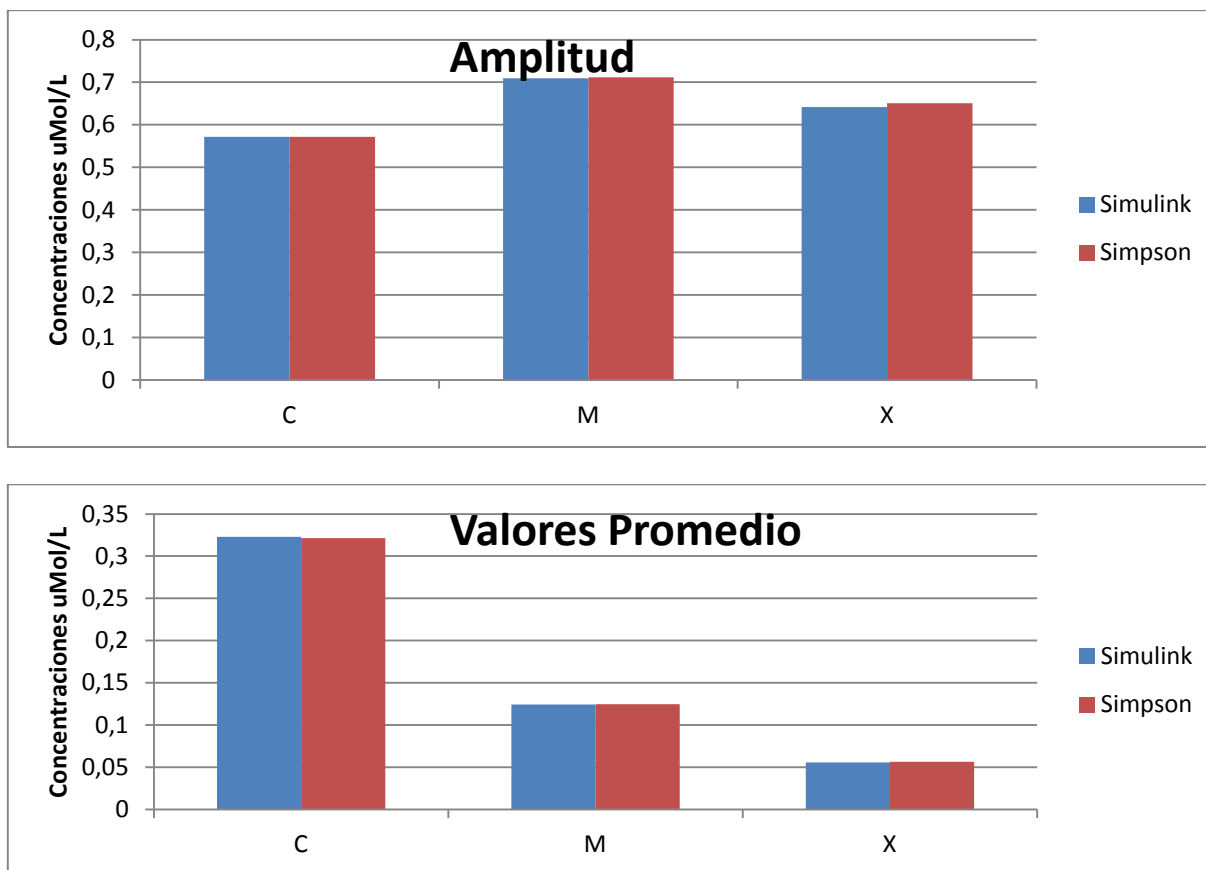
En la Tabla 5.3 se presentan los valores máximos, mínimos, promedios y amplitudes para las implementaciones realizadas para el oscilador mitótico en hardware usando la regla de Simpson y la implementación realizada en Simulink de Matlab con un tiempo de muestreo de 0,005.

Tabla.5.3.Tabla de valores máximos, mínimos, promedios y amplitudes para la implementación hardware del oscilador mitótico usando la regla de Simpson y la implementación software con un tiempo de muestreo de 0,005

Metabolito	SIMULINK T= 0,005				METODO DE SIMPSON T=0,005			
	V. Máximo	V. Mínimo	V. Promedio	Amplitud	V. Máximo	V. Mínimo	V. Promedio	Amplitud
C	0,5815	0,01	0,3228	0,5715	0,5817	0,01	0,3214	0,5717
M	0,7093	2,22E-04	0,1244	0,709078	0,7117	0	0,1246	0,7117
X	0,6416	2,73E-06	0,0558	0,641597	0,6505	0	0,0567	0,6505

El diagrama de barras correspondiente a los datos consignados en la Tabla 5.3 se muestra en la Figura 5.17 para mostrar la eficiencia de la implementación en hardware usando el método de la regla de Simpson.

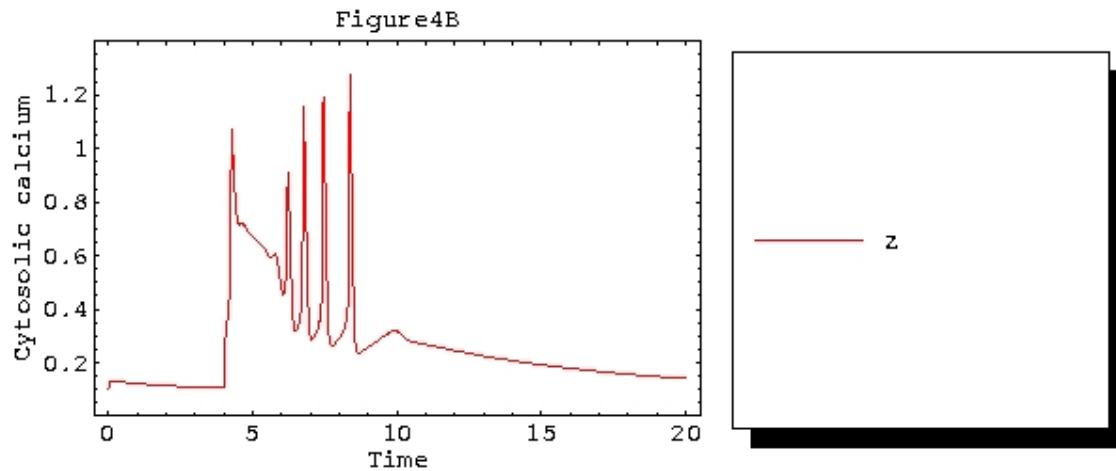
Fig.5.17. Diagrama de barras comparativo de la eficiencia de las implementación Hardware usando la regla de Simpson versus implementación software.



5.3 ANÁLISIS Y RESULTADOS PARA FLUJO DE Ca^{2+} EN LAS CELULAS PANCREATICAS

El resultado ofrecido por la base de datos biológica Biomodel DataBase para el “Signal-induced Ca^{2+} oscillations: properties of a model based on Ca^{2+} -induced Ca^{2+} release” se muestra en la Figura 5.18. Se puede observar que sólo se presenta la concentración para el metabolito Z y que los resultados obtenidos en las implementaciones realizadas en software y hardware para este sistema tienen similitud en amplitud y frecuencia.

Fig.5.18.Gráfica de concentraciones para el metabolito Z del flujo de CA^{2+} de Dupont brindado por la base de datos biológica.



La implementación en hardware para el modelo del flujo de CA^{2+} de Dupont se simuló con un reloj de 10ns=100MHz y un tiempo de muestreo de 0,01, realizando en 3ms un total de 1455 iteraciones para un tiempo de simulación de 0sg a 14.55 sg. El tiempo de simulación obtenido para este sistema en la simulación software realizada en Simulink es:

```
>> tic
>> sim(modelname, 'STOPTIME', '14.55')
toc
```

Simulink.SimulationOutput:

```
Y: [1456x1 double]
Z: [1456x1 double]
TIME: [1456x1 double]
tout: [1000x1 double]
```

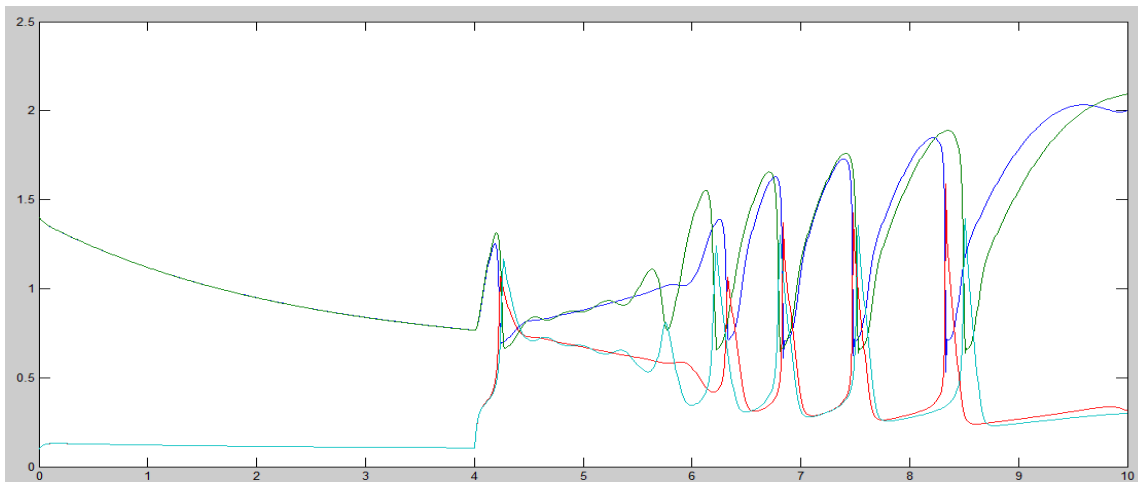
Elapsed time is 3.922856 seconds.

Según los tiempos de simulación obtenidos en la implementación software y en la implementación hardware del modelo del flujo de CA^{2+} de Dupont, se puede concluir que la implementación hardware es aproximadamente 1307 más rápida que la implementación en software.

La implementación del sistema usando el método de Euler fue simulado con un reloj de 10ns y un tiempo de muestreo de 0,01, generando 1000 muestras para una simulación desde 0sg a 10sg. La gráfica de comparación para las concentraciones de los metabolitos Y y Z

obtenidos en Simulink y en la implementación hardware usando el método de Euler se muestra en la Figura 5.19.

Fig. 5.19. Gráfica de concentraciones para los metabolitos Y y Z del flujo de CA implementado en software y hardware usando el método de Euler.



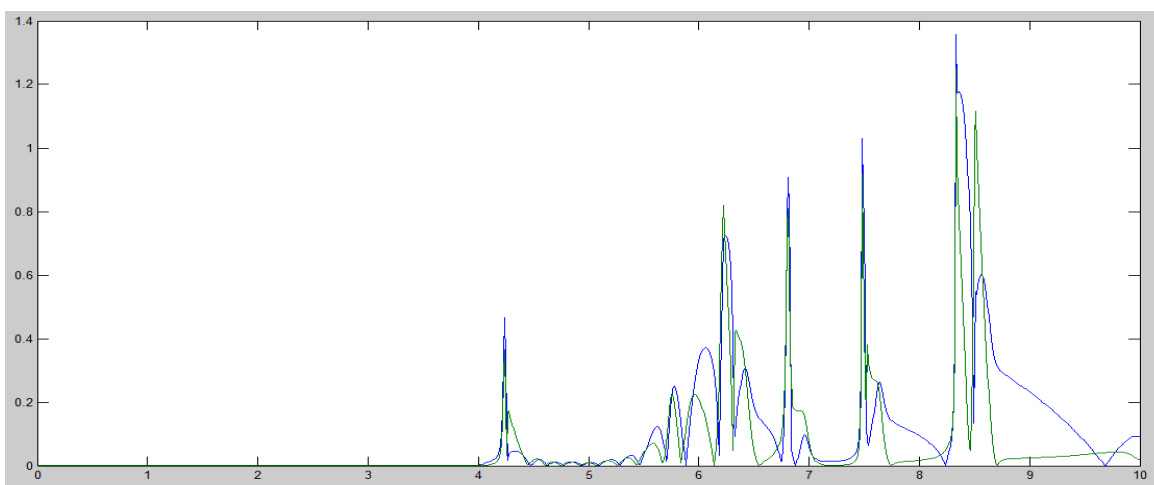
Los errores absolutos y relativos para la implementación hardware del modelo usando el método de Euler son:

Error absoluto de Y = 1,3570 Error relativo de Y= 0,667 ~ 60%

Error absoluto de Z= 1,2470 Error relativo de Z=0,785 ~ 78.5%

En la Figura 5.20 se presenta el error absoluto para las concentraciones de los metabolitos Y y Z obtenidos en la implementación del sistema usando el método de Euler.

Fig. 5.20. Gráfica del error absoluto para las concentraciones Z y Y del flujo de CA implementado en hardware usando el método de Euler.

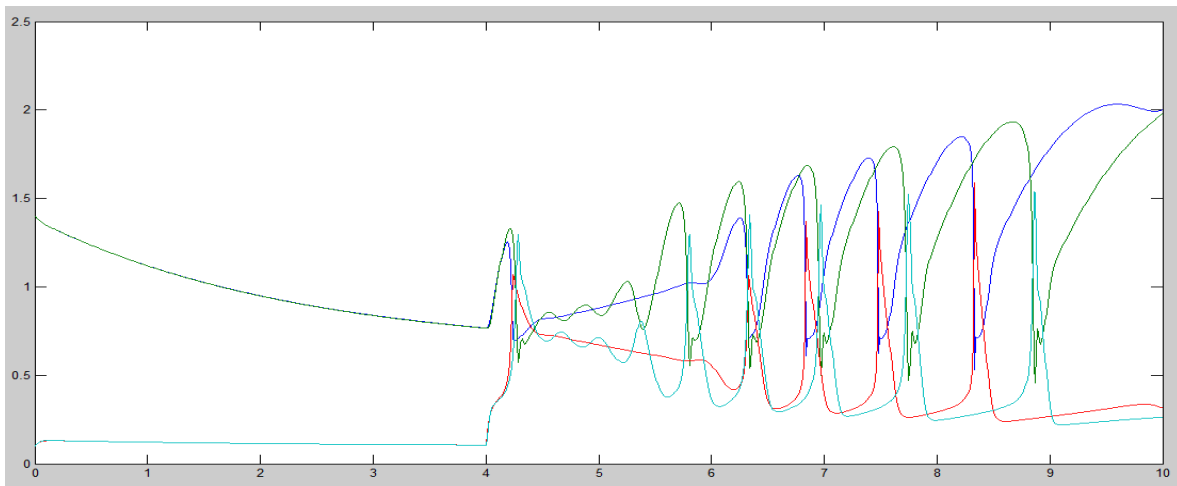


La gráfica de errores absolutos en el tiempo presenta errores muy altos debido al desfase el cual aumenta con el tiempo y a los picos transitorios. Estos errores disminuyen al reducir el tiempo de muestreo.

La simulación de la implementación en hardware del modelo del flujo de CA de Dupont usando la regla trapezoidal, se realizó con un reloj de 10ns y un tiempo de muestreo de 0,01 para una simulación desde 0sg a 10 sg, generando 1000 iteraciones.

La gráfica comparativa de los resultados obtenidos en Simulink de Matlab con los resultados obtenidos en la implementación hardware del sistema usando la regla trapezoidal se muestra en la Figura 5.21.

Fig.5.21. Gráfica de concentraciones para los metabolitos Y y Z del flujo de CA implementado en software y hardware usando la regla trapezoidal.



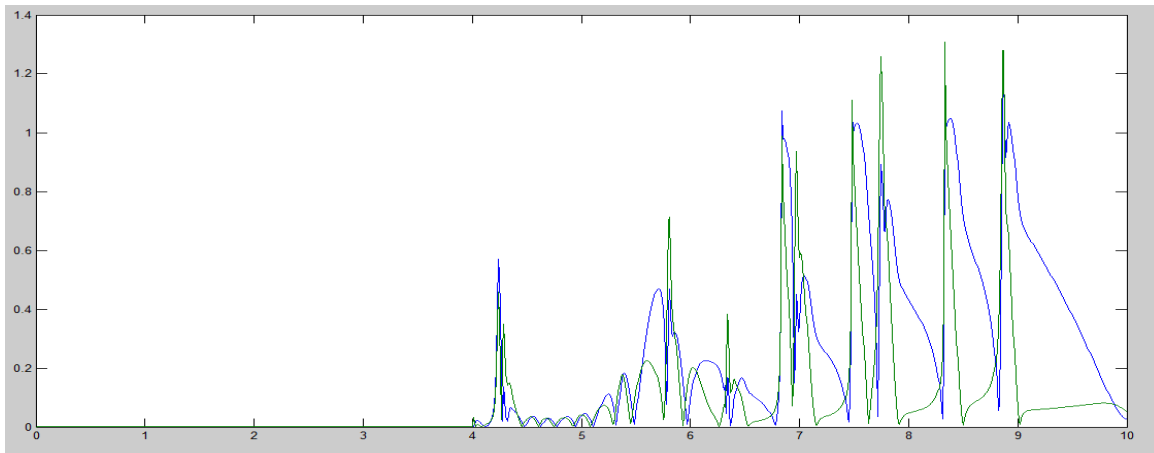
Los errores absolutos y relativos para las concentraciones de los metabolitos Y y Z obtenidos en la implementación en hardware usando la regla trapezoidal son:

Error absoluto de Y= 1,2045 Error relativo de Y= 0,592~ 59.2%

Error absoluto de Z=1,3084 Error relativo de Z=0,8243~ 82.4%

La gráfica de los errores absolutos para las concentraciones de los metabolitos obtenidas en la implementación en hardware del sistema usando la regla trapezoidal se muestra en la Figura 5.22.

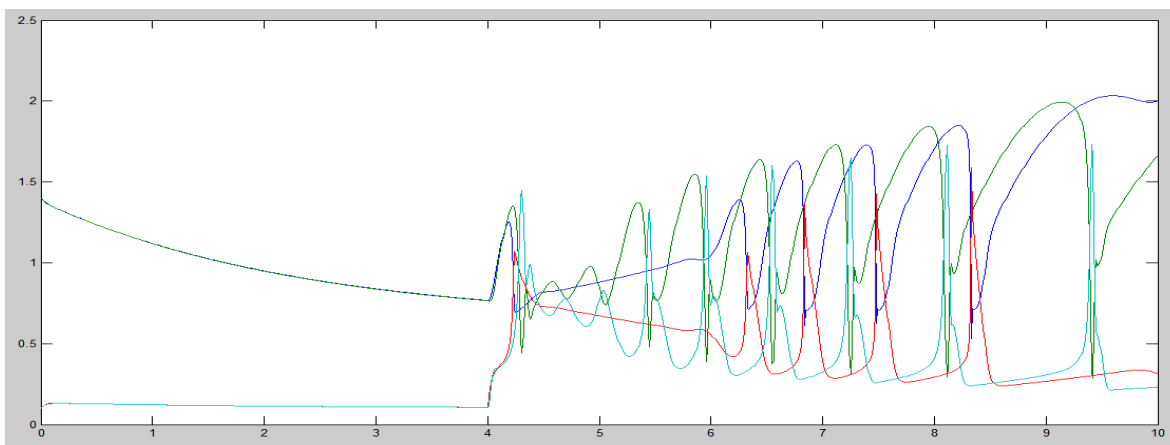
Fig.5.22. Gráfica de errores absolutos para las concentraciones de los metabolitos Y y Z del flujo de CA implementado en hardware usando la regla trapezoidal.



Como se puede observar en las Figuras 5.21 y 5.22, los errores altos de las concentraciones de los metabolitos obtenidos en la implementación en hardware usando la regla del trapecio se generan por el desfase que aumenta en el tiempo y los picos transitorios.

La implementación en hardware del sistema utilizando la regla de Simpson se simuló con un reloj de 10ns y un tiempo de muestreo de 0,01, generando 1000 muestras para una simulación desde 0sg a 10sg. En la Figura 5.23 se presentan los resultados para lo metabolitos Y y Z obtenidos en la implementación realizada en Simulink y en la implementación hardware del sistema usando la regla de Simpson.

Fig.5.23. Gráfica de concentraciones para los metabolitos Y y Z del flujo de CA^{2+} implementado en software y hardware usando la regla de Simpson.



Los errores absolutos y relativos para las concentraciones de los metabolitos obtenidas en la implementación en hardware del flujo de CA usando la regla de Simpson son:

Error absoluto de Y= 1,7242

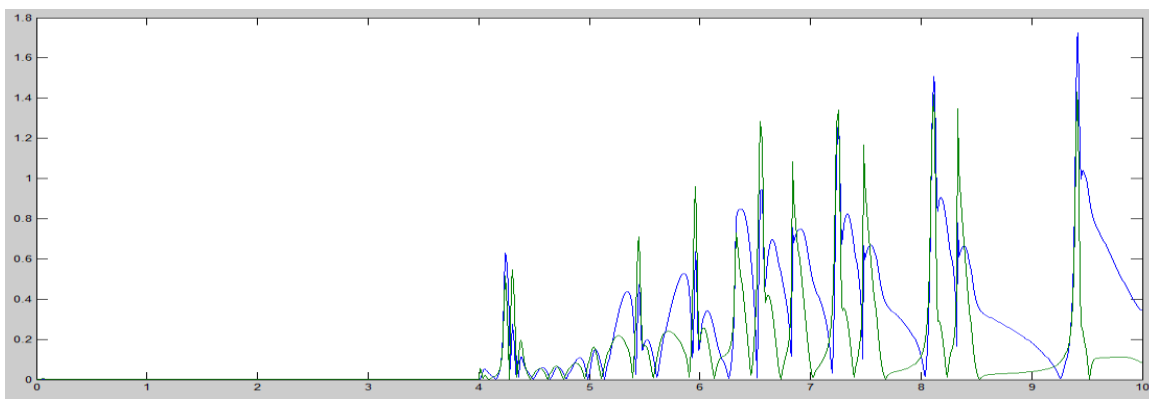
Error relativo de Y= 0,847 ~ 84.7%

Error absoluto de Z = 1,4306

Error relativo de Z= 0,9014 ~ 90.14%

La gráfica de los errores absolutos para las concentraciones de los metabolitos obtenidos en la implementación hardware del sistema usando la regla Simpson se presenta en la Figura 5.24.

Fig.5.24. Gráfica de errores absolutos para las concentraciones de los metabolitos Y y Z del flujo de CA implementado en hardware usando la regla de Simpson.

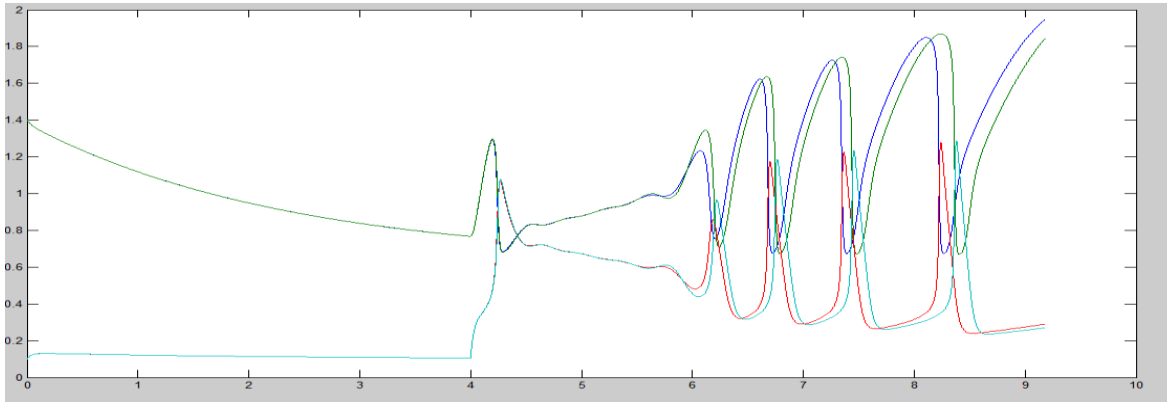


En la gráfica anterior se observa un error alto debido al desfase de las concentraciones obtenidas en la implementación hardware respecto a los resultados de la implementación en software y a los picos transitorios.

Con el fin de reducir los errores obtenidos del sistema del flujo de CA de Dupont en las simulaciones realizadas con un tiempo de muestreo de 0.01, se realizó una simulación con un tiempo de muestreo menor para reducir los picos transitorios y el desfase.

En la Figura 5.25 se muestra los resultados obtenidos en la simulación de la implementación del sistema usando el método de Euler, con un tiempo de muestreo de 0,001 y un reloj de 10ns.

Fig.5.25. Gráfica de concentraciones para los metabolitos Y y Z del flujo de CA implementado en software y hardware usando el método de Euler con un tiempo de muestreo de 0,001.



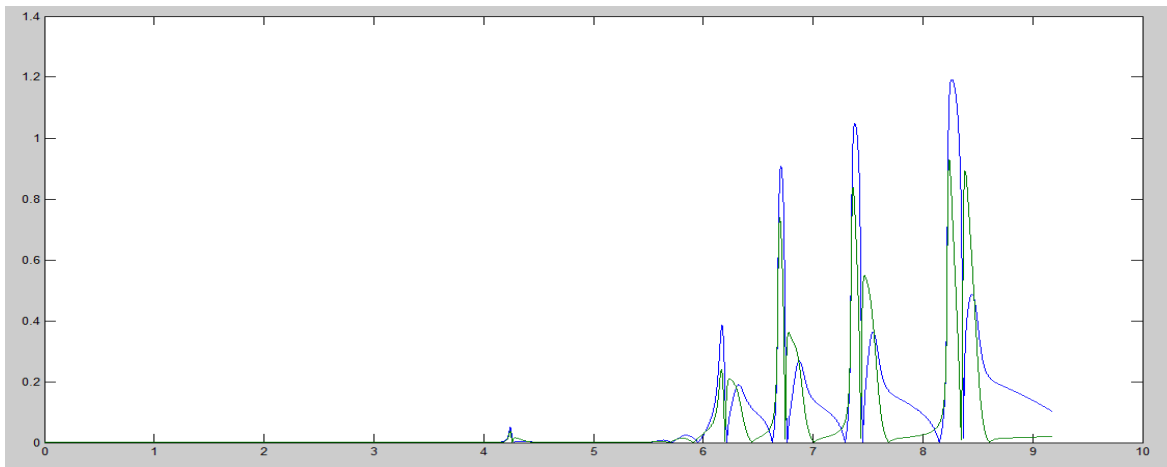
Los errores absolutos para las concentraciones obtenidas en la simulación de la implementación del sistema usando el método de Euler con un tiempo de muestreo de 0,001 son:

Error absoluto de Y= 1.1928 Error relativo de Y= 0,5795 ~ 57.9%

Error absoluto de Z =0.9293 Error relativo de Z= 0,7276 ~ 72.7%

La gráfica de los errores absolutos para las concentraciones de los metabolitos Y y Z obtenidas en la implementación hardware usando el método de Euler con un tiempo de muestreo de 0,001 se presenta en la Figura 5.26.

Fig.5.26. Gráfica de errores absolutos para las concentraciones de los metabolitos Y y Z del flujo de CA implementado en hardware usando el método de Euler con un tiempo de muestreo de 0,001.



En la Figura 5.26 se puede observar un error más bajo que el obtenido en la simulación de la implementación en hardware por el método de Euler con un tiempo de muestreo de 0,01. Al disminuir el paso se obtiene un resultado más exacto.

La tabla para los valores máximos, mínimos, promedios y amplitudes de las concentraciones de los metabolitos obtenidas en las implementaciones realizadas en software y en hardware usando el método de Euler, regla trapezoidal y regla de Simpson con un tiempo de muestreo de 0.01, se muestra en la Tabla 5.4.

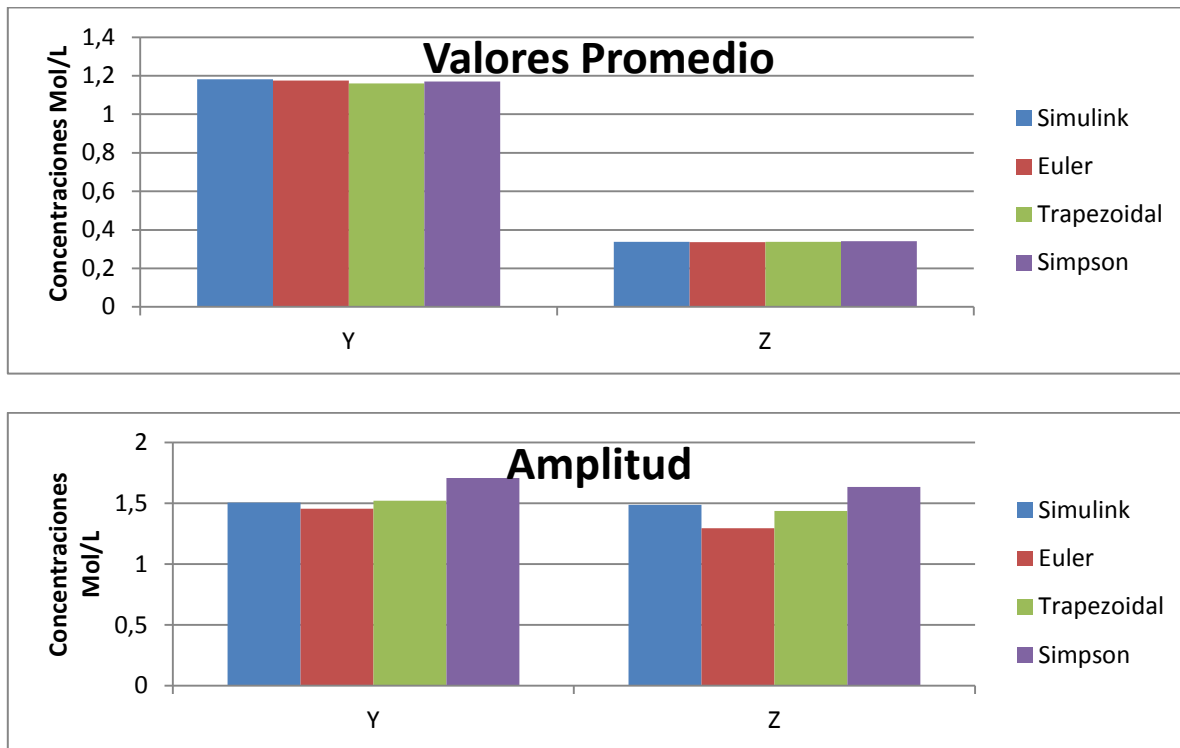
Tabla.5.4. Tabla de valores máximos, mínimos, promedios y amplitudes para los resultados obtenidos en las implementaciones del flujo de CA usando el método de Euler, regla trapezoidal, regla de Simpson y Simulink.

Metabolito	SIMULINK T=0,01				METODO DE EULER T=0,01			
	V. Máximo	V. Mínimo	V. Promedio	Amplitud	V. Máximo	V. Mínimo	V. Promedio	Amplitud
Y	2,033	0,5297	1,1815	1,5033	2,0919	0,6373	1,175	1,4546
Z	1,5872	0,1	0,3371	1,4872	1,3941	0,1	0,3357	1,2941

REGLA TRAPEZOIDAL T=0,01				REGLA DE SIMPSON T=0,01			
V. Máximo	V. Mínimo	V. Promedio	Amplitud	V. Máximo	V. Mínimo	V. Promedio	Amplitud
1,9762	0,4572	1,1595	1,519	1,9942	0,2861	1,1693	1,7081
1,5371	0,1	0,3371	1,4371	1,7324	0,1	0,3405	1,6324

El diagrama de barras correspondiente a la Tabla 5.4 se muestra en la Figura 5.27 para una mejor comparación de la implementación del sistema realizada en software y las implementaciones hardware usando los métodos: Euler, regla trapezoidal y regla de Simpson.

Fig.5.27. Diagrama de barras comparativo de la eficiencia de las implementaciones Hardware del flujo de CA usando el método de Euler, regla trapezoidal y regla de Simpson versus implementación software



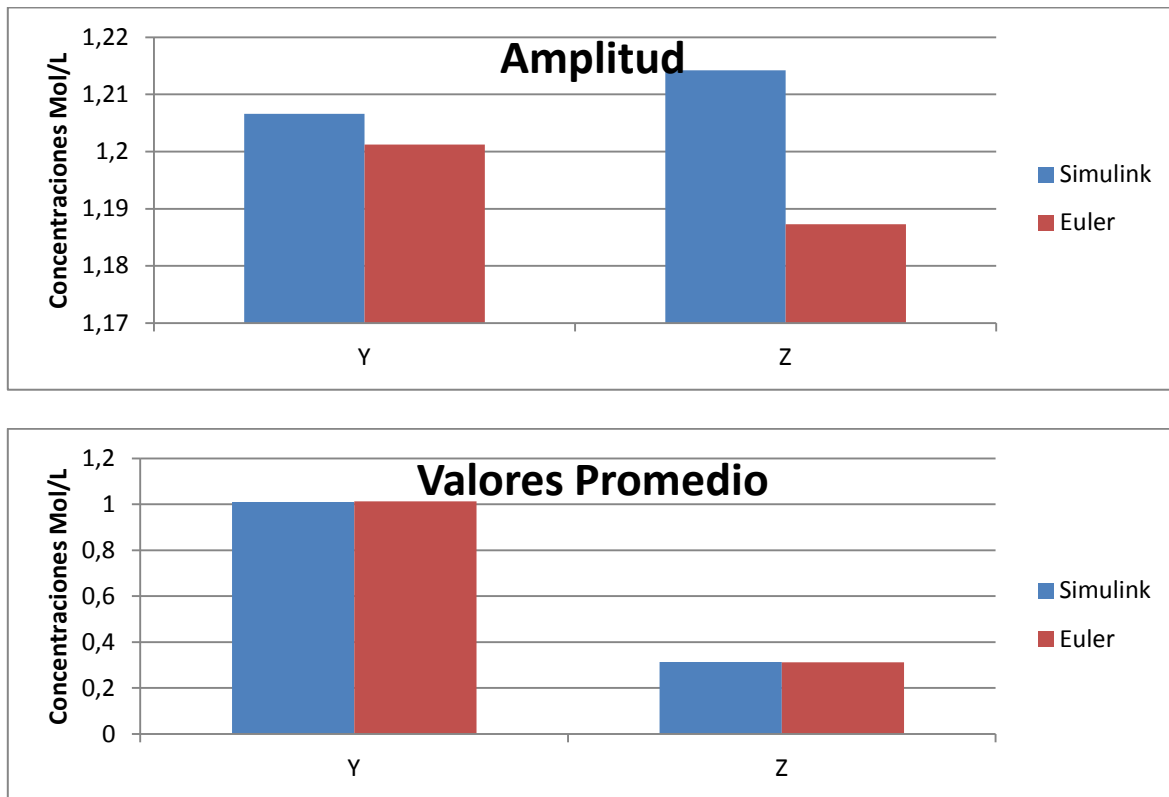
Para la simulación realizada de la implementación hardware del modelo de flujo de CA de Dupont usando el método numérico de Euler con un tiempo de muestreo de 0,001 se presentan los valores máximos, mínimos, promedios y amplitudes en la Tabla 5.5.

Tabla.5.5. Tabla de valores máximos, mínimos, promedios y amplitudes para los resultados obtenidos en las implementaciones del flujo de CA usando el método de Euler y Simulink con un tiempo de muestreo de 0,001.

Metabolito	SIMULINK T=0,001				METODO DE EULER T=0,001			
	V. Máximo	V. Mínimo	V. Promedio	Amplitud	V. Máximo	V. Mínimo	V. Promedio	Amplitud
Y	1,8857	0,6791	1,01	1,2066	1,8694	0,6682	1,013	1,2012
Z	1,3142	0,1	0,3136	1,2142	1,2873	0,1	0,3118	1,1873

Para una mejor comparación del desempeño de la implementación en hardware respecto a la implementación software se realiza un diagrama de barras (Figura 5.28) con los datos consignados en la Tabla 5.5.

Fig.5.28. Diagrama de barras comparativo de la eficiencia de la implementación hardware del flujo de CA usando el método de Euler versus implementación software con un tiempo de muestreo de 0,001.



CAPÍTULO 6

CONCLUSIONES Y TRABAJO FUTURO

En este trabajo de grado se presentan implementaciones software y hardware para tres sistemas biológicos usando los métodos numéricos: Euler, regla del trapecio y regla de Simpson. De tales implementaciones se concluye:

1. Los tiempos de ejecución para las implementaciones en hardware de los sistemas biológicos son mucho menores a los tiempos de ejecución de las implementaciones software. Ello se debe a que en las implementaciones hardware el procesamiento se realiza en paralelo, por lo tanto se pueden simular en hardware sistemas mucho más complejos y con tiempos de simulación mayores, lo cual es difícil de realizar y consume mucho tiempo como ocurre en implementaciones software donde el procesamiento es secuencial.
2. Para la implementación en hardware de un sistema biológico complejo como lo es la célula, se puede dividir el sistema en subsistemas biológicos (módulos) y luego unirlos para obtener el sistema completo. En hardware, la emulación de los módulos se puede realizar en paralelo donde el tiempo de emulación aproximadamente es el tiempo de emulación del módulo que consume más tiempo de procesamiento; en cambio en la implementación en software el tiempo de simulación incrementaría lineal o exponencialmente dependiendo de la cantidad de subsistemas que se vayan a simular.

3. Por otro lado el método numérico de Euler explícito es un método sencillo que consume pocos recursos computacionales y poco tiempo de procesamiento; además da una buena aproximación de los sistemas, pero dependiendo del sistema se puede obtener una aproximación más exacta usando un método implícito como es la regla del trapecio que utiliza un polinomio de primer orden, lo que hace que la implementación consuma más recursos y tiempo por tratarse de un diseño más complejo; además la exactitud del sistema también depende del tiempo de muestreo o paso que se use para resolver las ecuaciones diferenciales.

El trabajo futuro está orientado a obtener la implementación en hardware de los subsistemas biológicos que componen la célula, posteriormente lograr una célula artificial o *in silico* uniendo estos subsistemas biológicos; emular el funcionamiento de una célula o de un sistema biológico mas complejo sería un importante aporte a la biotecnología.

BIBLIOGRAFÍA

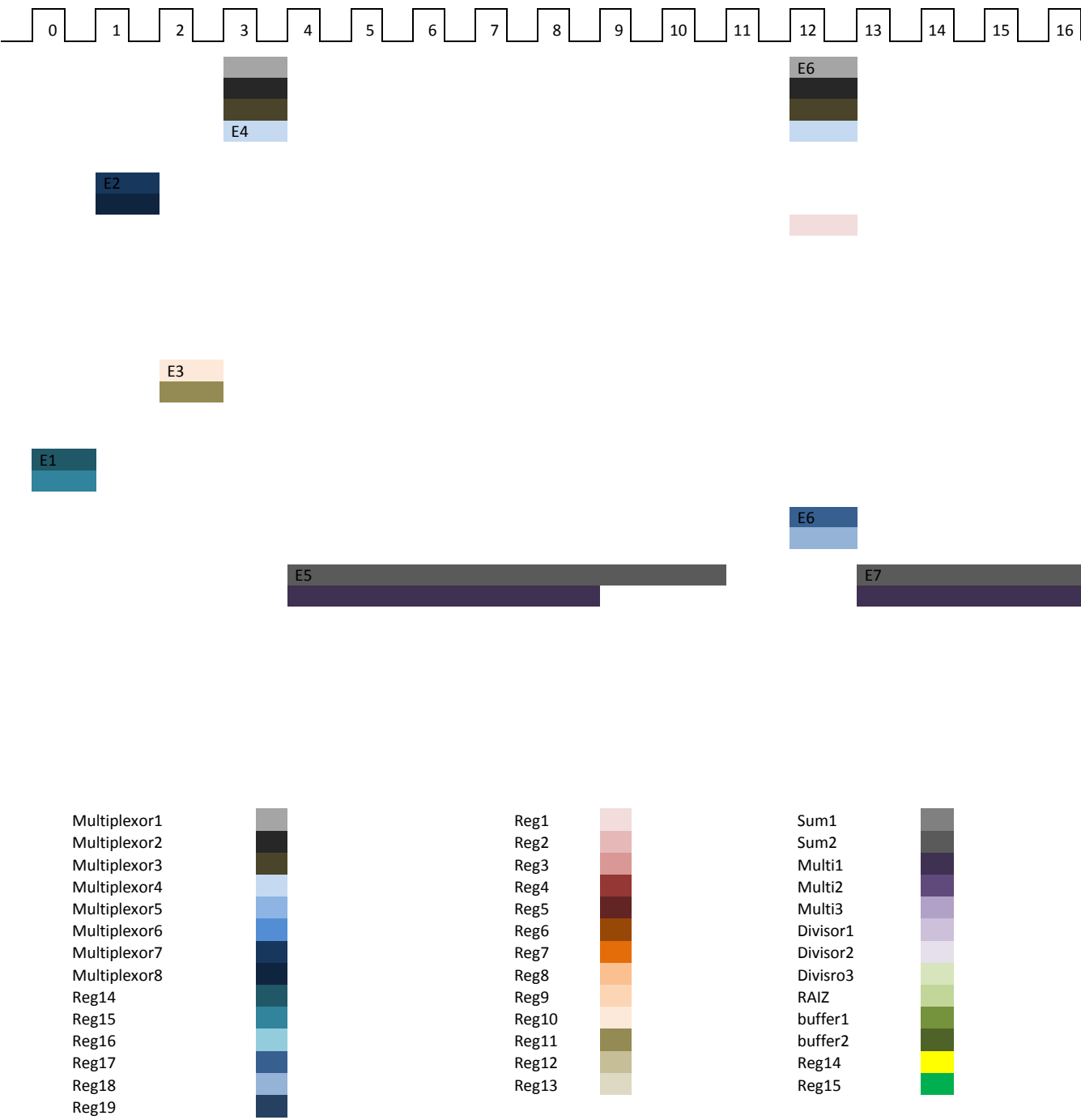
- 1- ABRAMSON, Guillermo. *Las matemáticas de los sistemas biológicos*. Argentina: Universidad Nacional del Cuyo- Instituto Balseiro. 2011.
- 2- BERTALANFFY, Ludwig Von. *Teoría general de sistemas*. México: Fondo de Cultura Económica. 1968.
- 3- BIOMODELS DATABASE. “Dupont1991_CaOscillation” Disponible en: <http://www.ebi.ac.uk/biomodels-main/BIOMD0000000117> Fecha de consulta: septiembre 3 de 2011.
- 4- BIOMODELS DATABASE. Goldbeter1991_MinMitOscil. Disponible en: <http://www.ebi.ac.uk/biomodels-main/BIOMD0000000003> Fecha de consulta: septiembre 2 de 2011.
- 5- BIOMODELS DATABASE. “Tyson1999_CircClock”. Disponible en: <http://www.ebi.ac.uk/biomodels-main/BIOMD0000000036> Fecha de consulta: septiembre 7 de 2011.
- 6- CELL-DESIGNER. Tutoriales para CellDesigner. Disponible en: <http://www.celldesigner.org/tutoriales.htm>. Fecha de consulta: Febrero 4 de 2011.
- 7- CASTILLO, Amanda; MARTÍNEZ-ANTONIO, Agustino. *La evolución de moléculas y desistemas biológicos*. En: Revista Ciencia. Octubre-Diciembre de 2009. Volumen 60. Número 4. pp. 22-35 México.
- 4- CHAVEZ, Karl Heinz. *Métodos numéricos para resolver problemas de ecuaciones diferenciales lineales con valor inicial* (Tesis). Guatemala: Universidad Francisco de Marroquín. 2000.
- 5- CHAPRA, Stevens; CANALE, Raymond. *Numerical methods for engineers*. Singapore: McGraw Hill. 2006.
- 6- CONN, Eric; STUMPF, P. K. *Bioquímica fundamental*. México: Editorial Limusa. 1982.
- 7- DÍAZ, Oscar. *El ion calcio: su regulación y función en la célula β pancreática*. En: Revista Cubana de endocrinología, 2003; volumen 14, número 3. Disponible en: http://bvs.sld.cu/revistas/end/vol14_3_03/end08303.htm Fecha de consulta: noviembre 20 de 2011.
- 8- DUPONT, Geneviève; BERRIDGE, M.J.; GOLDBETER, A. *Signal-induced Ca^{2+} oscillations: Properties of a model based on Ca^{2+} -induced Ca^{2+} release*. En: Revista Cell-Calcium. Feb-Mar 1991; volumen 12. Número 2-3. Pp. 73-85.

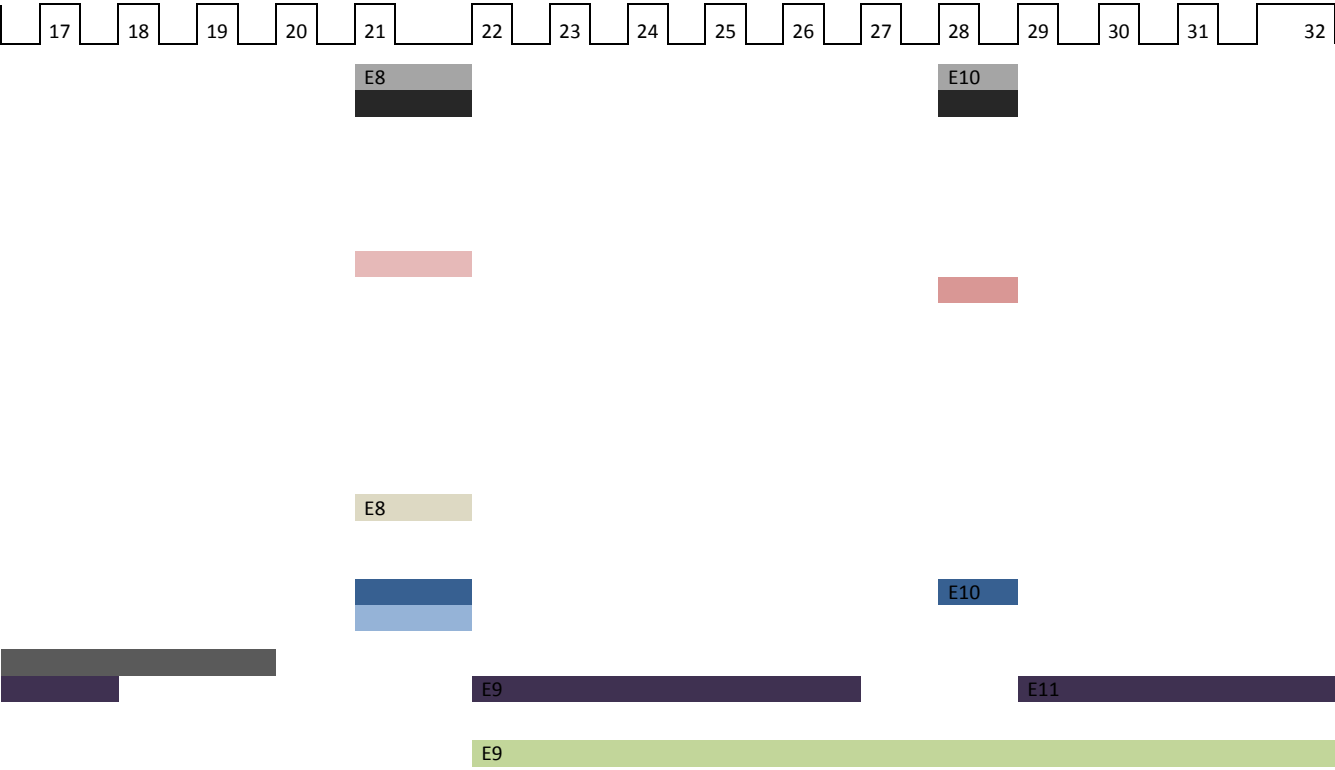
- 9- DUPONT, Geneviève; COMBETTES, Laurent. *Calcium Oscillations*. En: CSH Perspectives in biology. Marzo 2011. <http://cshperspectives.cshlp.org/content/3/3/a004226.full> Fecha de consulta: Noviembre 5 de 2011.
- 10- GENOMA ESPAÑA. *Biología de sistemas. Informe de vigilancia tecnológica*. España: 2007.
- 11- GOLDBETER, Albert. *A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase*. En revista Proc. Natl. Acad. Sci. USA. Vol.88, pp. 9107- 9111, October 1991 CellBiology. Disponible en: <http://www.pnas.org/content/88/20/9107.full.pdf> Fecha de consulta: Abril 9 de 2011.
- 12- GUTIÉRREZ, Julian; PÉREZ, Jorge Andrés; RUEDA, Camilo. *Modelamiento de sistemas biológicos usando cálculos de procesos concurrentes*. En: Revista Epíclidos. Cali (Colombia), Volumen 4, Número 1. Diciembre de 2005.
- 13- LELOUP, Jean-Christophe; GOLDBETER, Albert. *A Model for Circadian Rhythms in Drosophila Incorporating the Formation of a Complex between the PER and TIM Proteins*. JOURNAL OF BIOLOGICAL RHYTHMS, Volumen. 13, Número 1, Febrero 1998, pp. 70-87.
- 14- MUÑOZ, María Antonia. *Biotecnología*. Argentina: Universidad Nacional de Quilmes Editorial. 2006.
- 15- MONTERO, Francisco; MORAN, Federico. *Biofísica. Procesos de autorganización en biología*. Madrid: Editorial Eudema. 1992.
- 16- MOLTO, Germán. *Computación de Altas Prestaciones sobre Entornos Grid en Aplicaciones Biomédicas: Simulación de la Actividad Eléctrica Cardíaca y Diseño de Proteínas*. Tesis Doctoral Disponible en: www.dsic.upv.es/docs/bib-dig/tesis/etd-05242007-143805/gmolto_tesis.pdf Fecha de consulta: Agosto 27 2010.
- 17- MURRAY, Robert; GRANNEL, Daryl; ROWDELL, Victor. *Harper's Illustrated Biochemistry*. United States of America: 2006. McGraw-Hill.
- 18- PEDRONI, Volnei A. *Circuit Design with VHDL*. Cambridge: 2004. MIT Press.
- 19- PERRY, Douglas L. *VHDL Programming by Example*. United States of America: 2002. McGraw-Hill.
- 20- POLITI, Antoni; ANDREW, Thomas; GASPERS, Larry. *Coupled Oscillations of Calcium and IP3: Identification of Feedback Loops*. Nottingham University. Disponible en: http://www.maths.nott.ac.uk/personal/sc/calcium/Notts_Ca_Hofer.pdf Fecha de consulta: Noviembre 18 de 2011.

- 21- SCHIAVI, Emanuele; MUÑOZ, Ana Isabel. *Métodos numéricos para la resolución de problemas de valor inicial*. España: Escuela superior de ciencias experimentales y tecnológicas. 2007. Disponible en: http://www.escet.urjc.es/~matemati/mm_iq/tema5.pdf Fecha de consulta: Noviembre 15 de 2011.
- 22- TYSON, John; HONG, Christian; THRON, C. Dennis; NOVAK, Bela. *A Simple Model of Circadian Rhythms Based on Dimerization and Proteolysis of PER and TIM*. Biophysical Journal. Volumen 77, Noviembre 1999. Pp. 2411–2417.
- 23 -SAURO LAB. *Computational Systems Biology Series*: En revista: Methods in Molecular Biology , 2009, Vol. 541. University of Washington ISBN: 978-1-58829-905-5.
- 24- SYSTEM BIOLOGY WORKBENCH. *Tutoriales Cell Designer*. Disponible en: <http://www.sysbio.org/research/sbwintro.htm>-

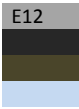
ANEXOS

ANEXO A Diagrama de tiempos para el modelo del ritmo circadiano usando el método de Euler





	33		34		35		36		37		38		39		40		41		42		43		44		45		46		47		48	
--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--



49		50		51		52		53		54		55		56		57		58		59		60		61		62		63		64	
----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--

E16

E16

E15

E17

6	6	6	6	6	7	7	7	7	7	7	7	7	7	7	8
5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0

E18

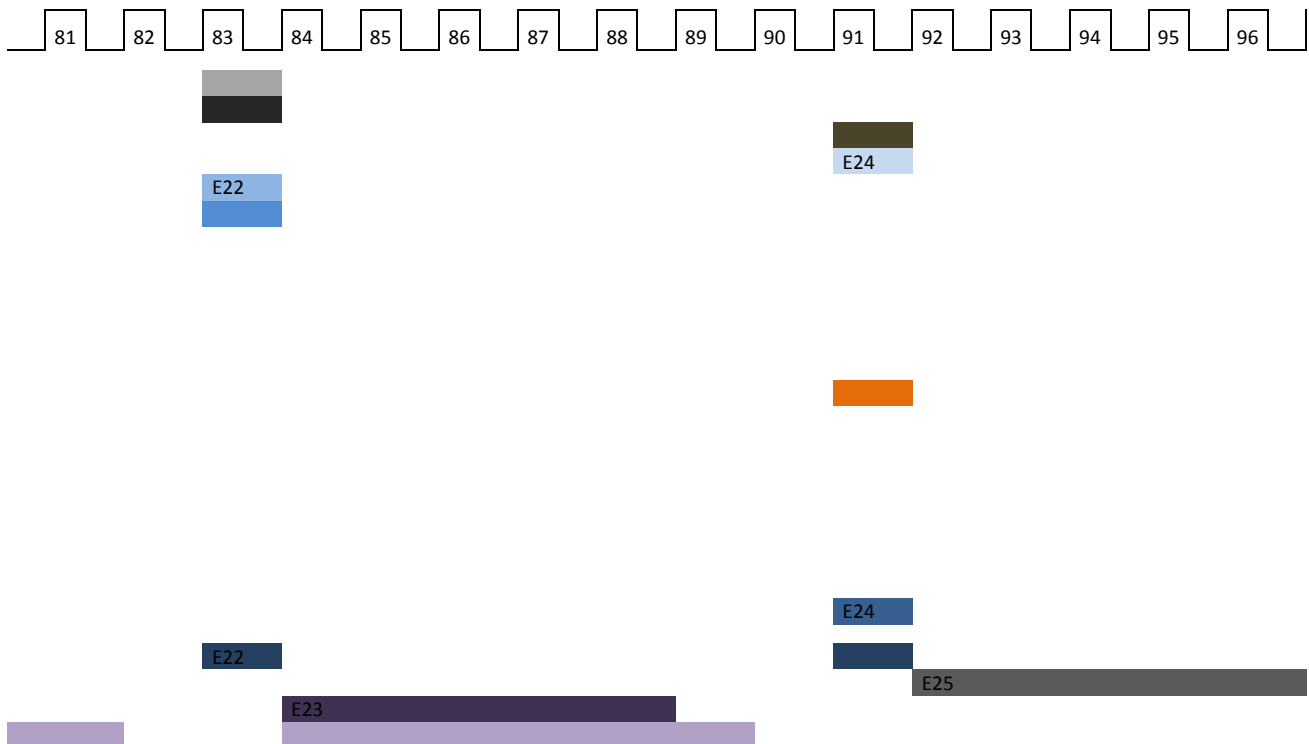
E20

E18

E20

E19

E21



97	98	99	100	101	102	103	104	105	106	107	108	109
----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

E26

E28

E28

E26

E27

110		111		112		113		114		115		116		117		118		119		120		121		122
-----	--	-----	--	-----	--	-----	--	-----	--	-----	--	-----	--	-----	--	-----	--	-----	--	-----	--	-----	--	-----

E30

E30

E29

E31

	123		124		125		126		127		128		129		130		131		132		133		134
--	-----	--	-----	--	-----	--	-----	--	-----	--	-----	--	-----	--	-----	--	-----	--	-----	--	-----	--	-----

E33

E33

E32

E34

	135		136		137		138	
--	-----	--	-----	--	-----	--	-----	--



E36

E35



ANEXO B Código en VHDL de la FSM implementada para el modelo del ritmo circadiano usando el método de Euler.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
USE ieee.numeric_std.all;
ENTITY PyT IS
PORT (
sEntrada: IN std_logic_vector (8 downto 0);
    CLK: IN std_logic;
    sReseth: IN std_logic;
    Start: IN std_logic;

    SEN1, SEN2, SEN3, SEN4, SEN5, SEN6, SEN7, SEN8, SEN10,
    SEN11, SEN12, SEN13, SEN14, SEN15, SEN16, SEN17, SEN18,
    SEN19, SEN20, SEN21, SEN22, SEN23, ADSU1 : OUT std_logic;
    SL6, SL5 : OUT std_logic_vector (1 downto 0);
    SL2, SL3 : OUT STD_LOGIC_VECTOR (2 DOWNTO 0);
    SL1, SL4: OUT std_logic_vector (3 DOWNTO 0);
    SL7, SL8 : OUT STD_LOGIC;
    rese1 :OUT std_logic;
    rese :OUT std_logic);

END PyT;

ARCHITECTURE PyT1 OF PyT IS

TYPE state_type IS (E0, E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11, E12, E13, E14, E15, E16,
E17, E18, E19, E20, E21, E22, E23, E24, E25, E26, E27, E28, E29, E30, E31, E32, E33, E34, E35,
E36, E37 );

SIGNAL tEstadoActual, tEstadoSiguiente: state_type;

Begin

clocked: PROCESS (CLK, sReseth)
    BEGIN
        IF (sReseth='1') THEN
tEstadoActual<= E0;
        ELSIF (CLK'EVENT AND CLK= '1') THEN
tEstadoActual<= tEstadoSiguiente;
        END IF;
    END PROCESS clocked;

LOGICA_ESTADO: PROCESS (tEstadoActual, sEntrada, start)
BEGIN
CASE (tEstadoActual) IS
```

```

WHEN E0 =>
IF (start = '1') THEN
tEstadoSiguiente<= E1;
END IF;

WHEN E1=>
IF(sEntrada = "000000000") THEN tEstadoSiguiente <= E2;
END IF;
WHEN E2=>
IF(sEntrada = "000000001") THEN tEstadoSiguiente <= E3;
END IF;
WHEN E3=>
IF(sEntrada = "000000010") THEN tEstadoSiguiente <= E4;
END IF;
WHEN E4=>
IF(sEntrada = "000000011") THEN tEstadoSiguiente <= E5;
END IF;
WHEN E5=>
IF(sEntrada = "000001011") THEN tEstadoSiguiente <= E6;
END IF;
WHEN E6=>
IF(sEntrada = "000001100") THEN tEstadoSiguiente <= E7;
END IF;
WHEN E7=>
IF(sEntrada = "000010100") THEN tEstadoSiguiente <= E8;
END IF;
WHEN E8=>
IF(sEntrada = "000010101") THEN tEstadoSiguiente <= E9;
END IF;
WHEN E9=>
IF(sEntrada = "000011011") THEN tEstadoSiguiente <= E10;
END IF;
WHEN E10=>
IF(sEntrada = "000011100") THEN tEstadoSiguiente <= E11;
END IF;
WHEN E11=>
IF(sEntrada = "000100110") THEN tEstadoSiguiente <= E12;
END IF;
WHEN E12=>
IF(sEntrada = "000100111") THEN tEstadoSiguiente <= E13;
END IF;
WHEN E13=>
IF(sEntrada = "000101111") THEN tEstadoSiguiente <= E14;
END IF;
WHEN E14=>
IF(sEntrada = "000110000") THEN tEstadoSiguiente <= E15;
END IF;
WHEN E15=>
IF(sEntrada = "000111000") THEN tEstadoSiguiente <= E16;
END IF;

```

```

WHEN E16=>
IF(sEntrada = "000111001") THEN tEstadoSiguiente <= E17;
END IF;
WHEN E17=>
IF(sEntrada = "001000001") THEN tEstadoSiguiente <= E18;
END IF;
WHEN E18=>
IF(sEntrada = "001000010") THEN tEstadoSiguiente <= E19;
END IF;
WHEN E19=>
IF(sEntrada = "001001010") THEN tEstadoSiguiente <= E20;
END IF;
WHEN E20=>
IF(sEntrada = "001001011") THEN tEstadoSiguiente <= E21;
END IF;
WHEN E21=>
IF(sEntrada = "001010010") THEN tEstadoSiguiente <= E22;
END IF;
WHEN E22=>
IF(sEntrada = "001010011") THEN tEstadoSiguiente <= E23;
END IF;
WHEN E23=>
IF(sEntrada = "001011010") THEN tEstadoSiguiente <= E24;
END IF;
WHEN E24=>
IF(sEntrada = "001011011") THEN tEstadoSiguiente <= E25;
END IF;
WHEN E25=>
IF(sEntrada = "001100011") THEN tEstadoSiguiente <= E26;
END IF;
WHEN E26=>
IF(sEntrada = "001100100") THEN tEstadoSiguiente <= E27;
END IF;
WHEN E27=>
IF(sEntrada = "001101100") THEN tEstadoSiguiente <= E28;
END IF;
WHEN E28=>
IF(sEntrada = "001101101") THEN tEstadoSiguiente <= E29;
END IF;
WHEN E29=>
IF(sEntrada = "001110101") THEN tEstadoSiguiente <= E30;
END IF;
WHEN E30=>
IF(sEntrada = "001110110") THEN tEstadoSiguiente <= E31;
END IF;
WHEN E31=>
IF(sEntrada = "001111110") THEN tEstadoSiguiente <= E32;
END IF;
WHEN E32=>
IF(sEntrada = "001111111") THEN tEstadoSiguiente <= E33;

```

```

END IF;
WHEN E33=>
IF(sEntrada = "010000000") THEN tEstadoSiguiente <= E34;
END IF;
WHEN E34=>
IF(sEntrada = "010001000") THEN tEstadoSiguiente <= E35;
END IF;
WHEN E35=>
IF(sEntrada = "010001001") THEN tEstadoSiguiente <= E36;
END IF;
WHEN E36=>
IF(sEntrada = "010001010") THEN tEstadoSiguiente <= E37;
END IF;
WHEN E37=> tEstadoSiguiente <= E4;

```

```

END CASE;
END PROCESS LOGICA_ESTADO;

```

```

SEN1 <= '1' WHEN (tEstadoActual = E6)
ELSE '0';
SEN2 <= '1' WHEN (tEstadoActual = E8)
ELSE '0';
SEN3 <= '1' WHEN (tEstadoActual = E10)
ELSE '0';
SEN4 <= '1' WHEN (tEstadoActual = E12)
ELSE '0';
SEN5 <= '1' WHEN (tEstadoActual = E14)
ELSE '0';
SEN6 <= '1' WHEN (tEstadoActual = E16)
ELSE '0';
SEN7 <= '1' WHEN (tEstadoActual = E24)
ELSE '0';
SEN8 <= '1' WHEN (tEstadoActual = E28)
ELSE '0';
SEN10 <= '1' WHEN (tEstadoActual = E3 or tEstadoActual = E36)
ELSE '0';
SEN11 <= '1' WHEN (tEstadoActual = E3 or tEstadoActual = E33)
ELSE '0';
SEN12 <= '1' WHEN (tEstadoActual = E30)
ELSE '0';
SEN13 <= '1' WHEN (tEstadoActual = E16 or tEstadoActual = E28)
ELSE '0';
SEN14 <= '1' WHEN (tEstadoActual = E8)
ELSE '0';
SEN15 <= '1' WHEN (tEstadoActual = E1)
ELSE '0';
SEN16 <= '1' WHEN (tEstadoActual = E1 )
ELSE '0';
SEN17 <= '1' WHEN (tEstadoActual = E12)

```

```

ELSE '0';
SEN18 <= '1' WHEN (tEstadoActual = E6 or tEstadoActual = E8 or tEstadoActual = E10 or
tEstadoActual = E12 or tEstadoActual = E14 or tEstadoActual = E16 or tEstadoActual = E18 or
tEstadoActual = E20 or tEstadoActual = E24 or tEstadoActual = E30 or tEstadoActual = E32)
ELSE '0';
SEN19 <= '1' WHEN (tEstadoActual = E6 or tEstadoActual = E8 or tEstadoActual = E14 or
tEstadoActual = E16 or tEstadoActual = E18 or tEstadoActual = E20 or tEstadoActual = E26 or
tEstadoActual = E28 or tEstadoActual = E30 or tEstadoActual = E32 or tEstadoActual = E35)
ELSE '0';
SEN20 <= '1' WHEN (tEstadoActual = E16 or tEstadoActual = E22 or tEstadoActual = E24 or
tEstadoActual = E28)
ELSE '0';
SEN21 <= '1' WHEN (tEstadoActual = E5 or tEstadoActual = E7 or tEstadoActual = E9 or
tEstadoActual = E11 or tEstadoActual = E13 or tEstadoActual = E15 or tEstadoActual = E17 or
tEstadoActual = E19 or tEstadoActual = E23 or tEstadoActual = E29 or tEstadoActual = E31)
ELSE '0';
SEN22 <= '1' WHEN (tEstadoActual = E5 or tEstadoActual = E7 or tEstadoActual = E13 or
tEstadoActual = E15 or tEstadoActual = E17 or tEstadoActual = E19 or tEstadoActual = E25 or
tEstadoActual = E27 or tEstadoActual = E29 or tEstadoActual = E31 or tEstadoActual = E34 )
ELSE '0';
SEN23 <= '1' WHEN (tEstadoActual = E15 or tEstadoActual = E21 or tEstadoActual = E23 or
tEstadoActual = E27)
ELSE '0';
SL1 <= "0000" WHEN (tEstadoActual= E4) ELSE
    "0001" WHEN (tEstadoActual= E6) ELSE
    "0010" WHEN (tEstadoActual= E8) ELSE
    "0011" WHEN (tEstadoActual= E10) ELSE
    "0100" WHEN (tEstadoActual= E12 or tEstadoActual= E14 ) ELSE
    "0101" WHEN (tEstadoActual= E16) ELSE
    "0110" WHEN (tEstadoActual= E18) ELSE
    "0111" WHEN (tEstadoActual= E22) ELSE
    "1000" WHEN (tEstadoActual = E28 or tEstadoActual = E30);

SL2 <= "000" WHEN (tEstadoActual= E4 or tEstadoActual = E6 or tEstadoActual = E8 or
tEstadoActual = E12 or tEstadoActual = E18) ELSE
    "001" WHEN (tEstadoActual= E10) ELSE
    "010" WHEN (tEstadoActual= E14) ELSE
    "011" WHEN (tEstadoActual= E16 or tEstadoActual = E22 ) ELSE
    "100" WHEN (tEstadoActual= E28) ELSE
    "101" WHEN (tEstadoActual= E30);

SL3 <= "000" WHEN (tEstadoActual= E4) ELSE
    "001" WHEN (tEstadoActual= E6 or tEstadoActual= E18 or tEstadoActual = E24 or
tEstadoActual = E30 or tEstadoActual = E33 ) ELSE
    "010" WHEN (tEstadoActual= E12) ELSE
    "011" WHEN (tEstadoActual= E14) ELSE
    "100" WHEN (tEstadoActual= E16) ELSE
    "101" WHEN (tEstadoActual= E26) ELSE
    "110" WHEN (tEstadoActual= E28);
SL4 <= "0000" WHEN (tEstadoActual= E4) ELSE

```

```

"0001" WHEN (tEstadoActual= E6 or tEstadoActual = E12 or tEstadoActual= E24 ) ELSE
"0010" WHEN (tEstadoActual= E14) ELSE
"0011" WHEN (tEstadoActual= E16) ELSE
"0100" WHEN (tEstadoActual= E18) ELSE
"0101" WHEN (tEstadoActual= E26) ELSE
"0110" WHEN (tEstadoActual= E28) ELSE
"0111" WHEN (tEstadoActual= E30) ELSE
"1000" WHEN (tEstadoActual = E33);

SL5 <= "00" WHEN (tEstadoActual= E14) ELSE
"01" WHEN (tEstadoActual= E20) ELSE
"10" WHEN (tEstadoActual= E22) ELSE
"11" WHEN (tEstadoActual= E26);

SL6 <= "00" WHEN (tEstadoActual= E4 or tEstadoActual = E26) ELSE
"01" WHEN (tEstadoActual= E20) ELSE
"10" WHEN (tEstadoActual= E22);

SL7 <= '0' WHEN (tEstadoActual= E2) ELSE
'1' WHEN (tEstadoActual= E35);

SL8 <= '0' WHEN (tEstadoActual= E2) ELSE
'1' WHEN (tEstadoActual= E32);

ADSU1 <= '1' WHEN (tEstadoActual = E5 or tEstadoActual = E7 or tEstadoActual = E13 or
tEstadoActual = E19 or tEstadoActual = E25 or tEstadoActual = E31 or tEstadoActual = E34 )
ELSE
'0' WHEN (tEstadoActual = E15 or tEstadoActual = E17 or tEstadoActual = E27 or
tEstadoActual = E29 );

rese1 <= '1' WHEN (tEstadoActual= E37 or tEstadoActual= E0) ELSE
'0';

rese<= '1' WHEN (tEstadoActual= E37 or tEstadoActual= E0) ELSE
'0';

END PyT1;

```

ANEXO C Test-bench para la implementación en hardware del modelo del ritmo circadiano usando el método de Euler

```
library ieee; USE ieee.std_logic_1164.all; use ieee.std_logic_arith.all; use ieee.std_logic_signed.all;
use std.textio.all; use IEEE.std_logic_textio.all;
LIBRARY altera_mf;
USE altera_mf.altera_mf_components.all;
entity PruebaPer is
```

```
end PruebaPer;
ARCHITECTURE tb OF PruebaPer IS
```

```
component Divi is
  PORT
  (
    aclr          : IN STD_LOGIC ;
    clk_en        : IN STD_LOGIC ;
    clock         : IN STD_LOGIC ;
    dataa         : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
    datab        : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
    result        : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
  );
end component Divi;
```

```
      component regtri20 is
port(
  clk:in std_logic;
  en:in std_logic;
  datoin2 :in std_logic_vector(31 downto 0);
  datoout2:out std_logic_vector(31 downto 0)
);
end component regtri20;
```

```
component Multi is
  PORT
  (
    aclr          : IN STD_LOGIC ;
    clk_en        : IN STD_LOGIC ;
    clock         : IN STD_LOGIC ;
    dataa         : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
    datab        : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
    result        : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
  );
end component Multi;
```

```
-----mux2-----
component multiplexor2 is
PORT ( a, b : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
sel: IN STD_LOGIC;
y: OUT STD_LOGIC_VECTOR (31 DOWNTO 0));
```

end component multiplexor2;

-----mux7-----

component multiplexor7 is
PORT (a, b, c, d, e, f, g: IN STD_LOGIC_VECTOR (31 DOWNT0 0);
sel: IN STD_LOGIC_VECTOR (2 DOWNT0 0);
y: OUT STD_LOGIC_VECTOR (31 DOWNT0 0));
end component multiplexor7;

-----mux4-----

component multiplexor4 is
PORT (a, b, c, d: IN STD_LOGIC_VECTOR (31 DOWNT0 0);
sel: IN STD_LOGIC_VECTOR (1 DOWNT0 0);
y: OUT STD_LOGIC_VECTOR (31 DOWNT0 0));
end component multiplexor4;

-----mux3-----

component multiplexor3 is
PORT (a, b, c: IN STD_LOGIC_VECTOR (31 DOWNT0 0);
sel: IN STD_LOGIC_VECTOR (1 DOWNT0 0);
y: OUT STD_LOGIC_VECTOR (31 DOWNT0 0));
end component multiplexor3;

-----mux9-----

component multiplexor9 is
PORT (a, b, c, d, e, f, g, h, i: IN STD_LOGIC_VECTOR (31 DOWNT0 0);
sel: IN STD_LOGIC_VECTOR (3 DOWNT0 0);
y: OUT STD_LOGIC_VECTOR (31 DOWNT0 0));
end component multiplexor9;

-----mux6-----

component multiplexor6 is
PORT (a, b, c, d, e, f: IN STD_LOGIC_VECTOR (31 DOWNT0 0);
sel: IN STD_LOGIC_VECTOR (2 DOWNT0 0);
y: OUT STD_LOGIC_VECTOR (31 DOWNT0 0));
end component multiplexor6;

-----mux8-----

component multiplexor8 is
PORT (a, b, c, d, e, f, g, h: IN STD_LOGIC_VECTOR (31 DOWNT0 0);
sel: IN STD_LOGIC_VECTOR (2 DOWNT0 0);
y: OUT STD_LOGIC_VECTOR (31 DOWNT0 0));
end component multiplexor8;

component sum is

PORT

(

aclr : IN STD_LOGIC ;
add_sub : IN STD_LOGIC ;

```

        clk_en      : IN STD_LOGIC ;
        clock       : IN STD_LOGIC ;
        dataa       : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
        datab      : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
        result      : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
    );
end component sum;

component PyT is
PORT (
sEntrada: IN std_logic_vector (8 downto 0);
    CLK: IN std_logic;
    sReseth: IN std_logic;
    Start: IN std_logic;
SEN1 : OUT std_logic;
SEN8 : OUT std_logic;

    SEN2, SEN3, SEN4, SEN5, SEN6, SEN7, SEN10, SEN11,
    SEN12, SEN13, SEN14, SEN15, SEN16, SEN17, SEN18, SEN19, SEN20, SEN21, SEN22,
    SEN23, ADSU1 : OUT std_logic;
    SL6, SL5 : OUT std_logic_vector (1 downto 0);
    SL2, SL3: OUT STD_LOGIC_VECTOR (2 DOWNTO 0);
    SL1, SL4: OUT std_logic_vector (3 DOWNTO 0);
    SL7, SL8 : OUT STD_LOGIC;
rese1 :OUT std_logic;
rese :OUT std_logic);

end component PyT;

component contador is
port(
clk:in std_logic;
reset:in std_logic;
time:out std_logic_vector(8 downto 0)
);
end component contador;

component raiz IS
PORT
(
    aclr      : IN STD_LOGIC ;
    clk_en    : IN STD_LOGIC ;
    clock     : IN STD_LOGIC ;
    data      : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
    result    : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
);
end component raiz;

component regtri0 is
port(

```

```

clk:in std_logic;
ld:in std_logic;
    datoout2:out std_logic_vector(31 downto 0)
);
end component regtri0;

```

```

    signal s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12, s13, s14, s15, s16, s3b, s5b, s10b, s4b,
M, P, Mo, Po, DP, DM, time, d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, v1, v2, v3, v4, v5 :
std_logic_vector (31 downto 0);
signal clk3, reseth, star, E1, E2, E3, E4, E5, E6, E7, E8, E10, E11, E12, E13, E14, E15, E16, E17,
E18, E19, E20, E21, E22, E23, E24, ADSUB1 : std_logic;
signal entrada1 : std_logic_vector(8 downto 0);
signal sm6, sm5 : std_logic_vector (1 downto 0);
signal sm2, sm3 : STD_LOGIC_VECTOR (2 DOWNTO 0);
signal sm4, sm1: std_logic_vector (3 DOWNTO 0);
signal sm7, sm8 : std_logic;
signal reset1 : std_logic;

```

```

BEGIN

```

```

    count1: contador

port map( clk3, reset1, entrada1 );

registro1 : regtri20

port map (clk3, E1, s5, s13);

registro2 : regtri20

port map (clk3, E2, s3, s12);

registro3 : regtri20

port map (clk3, E3, s3, s11);

registro4 : regtri20

port map (clk3, E4, s3, v2);

registro5 : regtri20

port map (clk3, E5, s3, v4);

registro6 : regtri20

port map (clk3, E6, s3, v3);

```

```

registro7 : regtri20

port map (clk3, E7, s10, v5);

registro8 : regtri20

port map (clk3, E8, s10, v1);

registro10 : regtri20

port map (clk3, E10, s15, M);

registro11 : regtri20

port map (clk3, E11, s16, P);

registro12 : regtri20

port map (clk3, E12, s5, DM);

registro13 : regtri20

port map (clk3, E13, s5, DP);

registro14 : regtri20

port map (clk3, E14, s5, s14);

registro15 : regtri0

port map (clk3, E15, Mo);

registro16 : regtri0

port map (clk3, E16, Po);

registro17 : regtri20

port map (clk3, E17, s4, s4b);

registro18 : regtri20

port map (clk3, E18, s3, s3b);

registro19 : regtri20

port map (clk3, E19, s5, s5b);

registro20 : regtri20

```

```

    port map (clk3, E20, s10, s10b);

multi1 : Multi

    port map (reseth, E21, clk3, s1, s2, s3);

    divisor1 : Divi

    port map (reseth, E23, clk3, s8, s9, s10);

    suma1 : sum

    port map (reseth, ADSUB1, E22, clk3, s6, s7, s5);

Mux1 : Multiplexor9

    Port map (d1, d2, d3, M, d4, s11, s5b, s10b, d12, sm1,
s1);

Mux2 : Multiplexor6

    Port map (P, d5, M, s10b, DP, DM, sm2, s2);

Mux3: Multiplexor7

    Port map (P, s3b, s4b, v2, d6, DP, v1, sm3, s6);

Mux4 : Multiplexor9

    Port map (d7, d8, v4, s10b, s12, v5, v3, P, M, sm4, s7);

Mux5 : Multiplexor4

    Port map (d9, s3b, s5b, d10, sm5, s8);

Mux6 : Multiplexor3

    Port map (s5b, d11, s13, sm6, s9);

Mux7 : Multiplexor2

    Port map (Mo, s5b, sm7, s15);

Mux8 : Multiplexor2

    Port map (Po, s5b, sm8, s16);

raiz1 : raiz

    Port map (reseth, E24, CLK3, s14, s4);

```

machine : PyT

Port map (entrada1, CLK3, reseth, star, E1, E8, E2, E3, E4, E5, E6, E7, E10, E11, E12, E13, E14, E15, E16, E17, E18, E19, E20, E21, E22, E23, ADSUB1, sm6, sm5, sm2, sm3, sm1, sm4, sm7, sm8, reset1, reseth);

reloj : process

begin

clk3 <= '0';

wait for 10 ns;

clk3 <= '1';

wait for 10 ns;

end process ;

inici: process

begin

d1 <= "01000100110010000000000000000000";

d2 <= "00111100111101011100001010001111";

d3 <= "01000001001000000000000000000000";

d4 <= "00111101110011001100110011001101";

d5 <= "00111111000000000000000000000000";

d6 <= "00111111100000000000000000000000";

d7 <= "00111101010011001100110011001101";

d8 <= "00111111100000000000000000000000";

d9 <= "01000000000000000000000000000000";

d10 <= "00111111100000000000000000000000";

d11 <= "00111110010011001100110011001101";

d12 <= "00111100001000111101011100001010";

E24 <= '1';

reseth <= '0';

wait;

end process inici;

pruebap: process

begin

for i in 0 to 3 loop

wait until clk3 = '1'; -- <=====

star <= '1';

end loop;

end process pruebap;

savec: process

file file_out : TEXT open write_mode is "M.txt";

variable line_out : LINE;

begin

```

        WRITE(line_out, M);
        wait until reset1='1';
        WRITELINE(file_out, line_out);
    end process;

savex: process
    file file_out : TEXT open write_mode is "P.txt";
    variable line_out : LINE;
begin

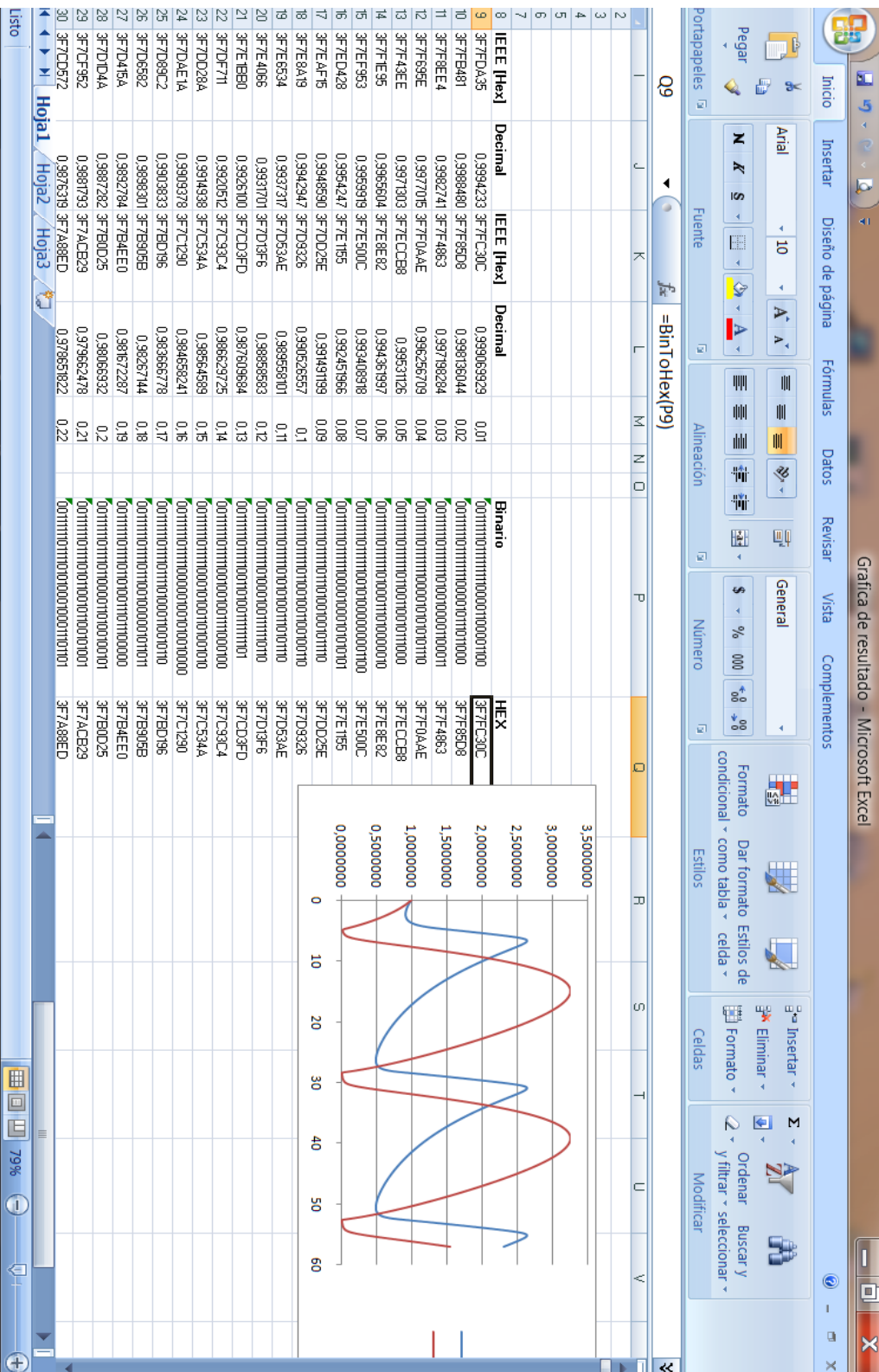
    WRITE(line_out, P);

    wait until reset1='1';
    WRITELINE(file_out, line_out);
end process;

END tb;

```

ANEXO D(Macro en Excel para convertir de binario a hexadecimal y de hexadecimal a decimal)



ANEXO E Código en Visual Basic para conversión de binario a hexadecimal y de hexadecimal a binario

```
Option Explicit
Type uab4: ab(0 To 3) As Byte: End Type
Type uab8: ab(0 To 7) As Byte: End Type
Type uflt: f As Single: End Type
Type udbl: d As Double: End Type

Function Sng2Hex(f As Single) As String
    Const sPad As String = "0"
    Dim uf As uflt
    Dim ub As uab4
    Dim i As Long
    uf.f = f
    LSet ub = uf
    For i = 0 To 3
        Sng2Hex = Right(sPad & Hex(ub.ab(i)), 2) & Sng2Hex
    Next i
    Sng2Hex = Left(Sng2Hex, Len(Sng2Hex))
End Function
```

```
Function Hex2Sng(ByVal s As String) As Single
    Const sPad As String = "00000000"
    Dim i As Long
    Dim ub As uab4
    Dim ab(0 To 3) As Byte
    s = Replace(s, " ", "")
    If Len(s) > 8 Then Exit Function
    If Len(s) < 8 Then s = Right(sPad & s, 8)
    For i = 0 To 3
        ab(i) = CByte("&H" & Mid(s, 2 * i + 1, 2))
    Next i
    Hex2Sng = Byte2Sng(ab)
End Function
```

```
Function BinToHex(Binary As String)
    Dim Value&, i&, Base#: Base = 1
    For i = Len(Binary) To 1 Step -1
        Value = Value + IIf(Mid(Binary, i, 1) = "1", Base, 0)
        Base = Base * 2
    Next i
    BinToHex = Hex(Value)
End Function
```

ANEXO F Código en VHDL de la FSM implementada para el modelo del ritmo circadiano usando la regla trapezoidal

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
USE ieee.numeric_std.all;
ENTITY PyT IS
PORT (sEntrada2 : IN std_logic_vector (31 downto 0);
sEntrada: IN std_logic_vector (8 downto 0);
      CLK: IN std_logic;
      sReseth: IN std_logic;
      Start: IN std_logic;

      SEN1, SEN2, SEN3, SEN4, SEN5, SEN6, SEN7, SEN8 SEN10,
      SEN11,SEN12, SEN13, SEN14, SEN15, SEN16, SEN17, SEN18,
      SEN19, SEN20, SEN21, SEN22, SEN23, SEN25, SEN26, ADSU1 : OUT std_logic;
      SL6, SL5 : OUT std_logic_vector (1 downto 0);
      SL2, SL3 : OUT STD_LOGIC_VECTOR (2 DOWNTO 0);
      SL1, SL4: OUT std_logic_vector (3 DOWNTO 0);
      SL7, SL8 : OUT STD_LOGIC;
      rese1 :OUT std_logic;
      rese :OUT std_logic;
      salida :OUT std_logic);

END PyT;

ARCHITECTURE PyT1 OF PyT IS

TYPE state_type IS (E0, E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11, E12, E13, E14, E15, E16,
E17, E18, E19, E20, E21, E22, E23, E24, E25, E26, E27, E28, E29, E30, E31, E32, E33, E34, E35,
E36, E37, E38, E39, E40, E41, E42, E43, E44, E45, E46, E47, E48, E49 );

SIGNAL tEstadoActual, tEstadoSiguiente: state_type;

Begin

clocked: PROCESS (CLK, sReseth)
BEGIN
  IF (sReseth='1') THEN
tEstadoActual<= E0;
  ELSIF (CLK'EVENT AND CLK= '1') THEN
tEstadoActual<= tEstadoSiguiente;
  END IF;
END PROCESS clocked;
```

```

LOGICA_ESTADO: PROCESS (tEstadoActual, sEntrada, sEntrada2, start)
BEGIN
CASE (tEstadoActual) IS

WHEN E0 =>
IF (start = '1') THEN
tEstadoSiguiente<= E1;
END IF;

WHEN E1=>
IF(sEntrada = "000000000" ) THEN tEstadoSiguiente <= E2;
END IF;
WHEN E2=>
IF(sEntrada = "000000001") THEN tEstadoSiguiente <= E3;
END IF;
WHEN E3=>
IF(sEntrada = "000000010") THEN tEstadoSiguiente <= E4;
END IF;
WHEN E4=>
IF(sEntrada = "000000011") THEN tEstadoSiguiente <= E5;
END IF;
WHEN E5=>
IF(sEntrada = "000001011") THEN tEstadoSiguiente <= E6;
END IF;
WHEN E6=>
IF(sEntrada = "000001100") THEN tEstadoSiguiente <= E7;
END IF;
WHEN E7=>
IF(sEntrada = "000010100") THEN tEstadoSiguiente <= E8;
END IF;
WHEN E8=>
IF(sEntrada = "000010101") THEN tEstadoSiguiente <= E9;
END IF;
WHEN E9=>
IF(sEntrada = "000011011") THEN tEstadoSiguiente <= E10;
END IF;
WHEN E10=>
IF(sEntrada = "000011100") THEN tEstadoSiguiente <= E11;
END IF;
WHEN E11=>
IF(sEntrada = "000100110") THEN tEstadoSiguiente <= E12;
END IF;
WHEN E12=>
IF(sEntrada = "000100111") THEN tEstadoSiguiente <= E13;
END IF;
WHEN E13=>
IF(sEntrada = "000101111") THEN tEstadoSiguiente <= E14;
END IF;
WHEN E14=>
IF(sEntrada = "000110000") THEN tEstadoSiguiente <= E15;

```

```

END IF;
WHEN E15=>
IF(sEntrada = "000111000") THEN tEstadoSiguiente <= E16;
END IF;
WHEN E16=>
IF(sEntrada = "000111001") THEN tEstadoSiguiente <= E17;
END IF;
WHEN E17=>
IF(sEntrada = "001000001") THEN tEstadoSiguiente <= E18;
END IF;
WHEN E18=>
IF(sEntrada = "001000010") THEN tEstadoSiguiente <= E19;
END IF;
WHEN E19=>
IF(sEntrada = "001001010") THEN tEstadoSiguiente <= E20;
END IF;
WHEN E20=>
IF(sEntrada = "001001011") THEN tEstadoSiguiente <= E21;
END IF;
WHEN E21=>
IF(sEntrada = "001010010") THEN tEstadoSiguiente <= E22;
END IF;
WHEN E22=>
IF(sEntrada = "001010011") THEN tEstadoSiguiente <= E23;
END IF;
WHEN E23=>
IF(sEntrada = "001011010") THEN tEstadoSiguiente <= E24;
END IF;
WHEN E24=>
IF(sEntrada = "001011011") THEN tEstadoSiguiente <= E25;
END IF;
WHEN E25=>
IF(sEntrada = "001100011") THEN tEstadoSiguiente <= E26;
END IF;
WHEN E26=>
IF(sEntrada = "001100100") THEN tEstadoSiguiente <= E27;
END IF;
WHEN E27=>
IF(sEntrada = "001101100") THEN tEstadoSiguiente <= E28;
END IF;
WHEN E28=>
IF(sEntrada = "001101101") THEN tEstadoSiguiente <= E29;
END IF;
WHEN E29=>
IF(sEntrada = "001110101" and sEntrada2= "00000000000000000000000000000000") THEN
tEstadoSiguiente <= E30;
ELSIF (sEntrada = "001110101" and sEntrada2 /= "00000000000000000000000000000000" )
THEN tEstadoSiguiente <= E38;
END IF;
WHEN E30=>

```

```

IF(sEntrada = "001110110" ) THEN tEstadoSiguiente <= E31 ;
END IF;
WHEN E31=>
IF(sEntrada = "001111110") THEN tEstadoSiguiente <= E32;

END IF;
WHEN E32=>
IF(sEntrada = "001111111") THEN tEstadoSiguiente <= E33;
END IF;
WHEN E33=>
IF(sEntrada = "010000000") THEN tEstadoSiguiente <= E34;
END IF;
WHEN E34=>
IF(sEntrada = "010001000") THEN tEstadoSiguiente <= E35;
END IF;
WHEN E35=>
IF(sEntrada = "010001001") THEN tEstadoSiguiente <= E36;
END IF;
WHEN E36=>
IF(sEntrada = "010001010") THEN tEstadoSiguiente <= E37;
END IF;
WHEN E37=> tEstadoSiguiente <= E4;
WHEN E38=>
IF(sEntrada = "001110110") THEN tEstadoSiguiente <= E39;
END IF;
WHEN E39=>
IF(sEntrada = "001111110") THEN tEstadoSiguiente <= E40;
END IF;
WHEN E40=>
IF(sEntrada = "001111111") THEN tEstadoSiguiente <= E41;
END IF;
WHEN E41=>
IF(sEntrada = "010000111") THEN tEstadoSiguiente <= E42;
END IF;
WHEN E42=>
IF(sEntrada = "010001000") THEN tEstadoSiguiente <= E43;
END IF;
WHEN E43=>
IF(sEntrada = "010010000") THEN tEstadoSiguiente <= E44;
END IF;
WHEN E44=>
IF(sEntrada = "010010001") THEN tEstadoSiguiente <= E45;
END IF;
WHEN E45=>
IF(sEntrada = "010010010") THEN tEstadoSiguiente <= E46;
END IF;
WHEN E46=>
IF(sEntrada = "010011010") THEN tEstadoSiguiente <= E47;
END IF;
WHEN E47=>

```

```

IF(sEntrada = "010011011") THEN tEstadoSiguiente <= E48;
END IF;
WHEN E48=>
IF(sEntrada = "010011100") THEN tEstadoSiguiente <= E49;
END IF;

```

```

WHEN E49=> tEstadoSiguiente <= E4;

```

```

END CASE;
END PROCESS LOGICA_ESTADO;

```

```

SEN1 <= '1' WHEN (tEstadoActual = E6)
ELSE '0';
SEN2 <= '1' WHEN (tEstadoActual = E8)
ELSE '0';
SEN3 <= '1' WHEN (tEstadoActual = E10)
ELSE '0';
SEN4 <= '1' WHEN (tEstadoActual = E12)
ELSE '0';
SEN5 <= '1' WHEN (tEstadoActual = E14)
ELSE '0';
SEN6 <= '1' WHEN (tEstadoActual = E16)
ELSE '0';
SEN7 <= '1' WHEN (tEstadoActual = E24)
ELSE '0';
SEN8 <= '1' WHEN (tEstadoActual = E28)
ELSE '0';
SEN10 <= '1' WHEN (tEstadoActual = E3 or tEstadoActual = E36 or tEstadoActual = E48)
ELSE '0';
SEN11 <= '1' WHEN (tEstadoActual = E3 or tEstadoActual = E33 or tEstadoActual = E45)
ELSE '0';
SEN12 <= '1' WHEN (tEstadoActual = E30 or tEstadoActual = E38)
ELSE '0';
SEN13 <= '1' WHEN (tEstadoActual = E16 or tEstadoActual = E28)
ELSE '0';
SEN14 <= '1' WHEN (tEstadoActual = E8)
ELSE '0';
SEN15 <= '1' WHEN (tEstadoActual = E1)
ELSE '0';
SEN16 <= '1' WHEN (tEstadoActual = E1 )
ELSE '0';
SEN17 <= '1' WHEN (tEstadoActual = E12)
ELSE '0';
SEN18 <= '1' WHEN (tEstadoActual = E6 or tEstadoActual = E8 or tEstadoActual = E10 or
tEstadoActual = E12 or tEstadoActual = E14 or tEstadoActual = E16 or tEstadoActual = E18 or
tEstadoActual = E20 or tEstadoActual = E24 or tEstadoActual = E30 or tEstadoActual = E32 or
tEstadoActual = E38 or tEstadoActual = E42 or tEstadoActual = E44)
ELSE '0';

```

```

SEN19 <= '1' WHEN (tEstadoActual = E6 or tEstadoActual = E8 or tEstadoActual = E14 or
tEstadoActual = E16 or tEstadoActual = E18 or tEstadoActual = E20 or tEstadoActual = E26 or
tEstadoActual = E28 or tEstadoActual = E30 or tEstadoActual = E32 or tEstadoActual = E35 or
tEstadoActual = E38 or tEstadoActual = E40 or tEstadoActual = E42 or tEstadoActual = E44 or
tEstadoActual = E47 )
ELSE '0';
SEN20 <= '1' WHEN (tEstadoActual = E16 or tEstadoActual = E22 or tEstadoActual = E24 or
tEstadoActual = E28)
ELSE '0';
SEN21 <= '1' WHEN (tEstadoActual = E5 or tEstadoActual = E7 or tEstadoActual = E9 or
tEstadoActual = E11 or tEstadoActual = E13 or tEstadoActual = E15 or tEstadoActual = E17 or
tEstadoActual = E19 or tEstadoActual = E23 or tEstadoActual = E29 or tEstadoActual = E31 or
tEstadoActual = E41 or tEstadoActual = E43)
ELSE '0';
SEN22 <= '1' WHEN (tEstadoActual = E5 or tEstadoActual = E7 or tEstadoActual = E13 or
tEstadoActual = E15 or tEstadoActual = E17 or tEstadoActual = E19 or tEstadoActual = E25 or
tEstadoActual = E27 or tEstadoActual = E29 or tEstadoActual = E31 or tEstadoActual = E34 or
tEstadoActual = E39 or tEstadoActual = E41 or tEstadoActual = E43 or tEstadoActual = E46 )
ELSE '0';
SEN23 <= '1' WHEN (tEstadoActual = E15 or tEstadoActual = E21 or tEstadoActual = E23 or
tEstadoActual = E27)
ELSE '0';
SEN25 <= '1' WHEN (tEstadoActual = E32 or tEstadoActual = E44)
ELSE '0';
SEN26 <= '1' WHEN (tEstadoActual = E32 or tEstadoActual = E44)
ELSE '0';
SL1 <= "0000" WHEN (tEstadoActual= E4) ELSE
    "0001" WHEN (tEstadoActual= E6) ELSE
    "0010" WHEN (tEstadoActual= E8) ELSE
    "0011" WHEN (tEstadoActual= E10) ELSE
    "0100" WHEN (tEstadoActual= E12 or tEstadoActual= E14 ) ELSE
    "0101" WHEN (tEstadoActual= E16) ELSE
    "0110" WHEN (tEstadoActual= E18 or tEstadoActual= E40 or tEstadoActual= E42) ELSE
    "0111" WHEN (tEstadoActual= E22) ELSE
    "1000" WHEN (tEstadoActual = E28 or tEstadoActual = E30);

SL2 <= "000" WHEN (tEstadoActual= E4 or tEstadoActual = E6 or tEstadoActual = E8 or
tEstadoActual = E12 or tEstadoActual = E18) ELSE
    "001" WHEN (tEstadoActual= E10) ELSE
    "010" WHEN (tEstadoActual= E14) ELSE
    "011" WHEN (tEstadoActual= E16 or tEstadoActual = E22 ) ELSE
    "100" WHEN (tEstadoActual= E28) ELSE
    "101" WHEN (tEstadoActual= E30) ELSE
    "110" WHEN (tEstadoActual= E40 or tEstadoActual= E42 );

SL3 <= "000" WHEN (tEstadoActual= E4) ELSE
    "001" WHEN (tEstadoActual= E6 or tEstadoActual= E18 or tEstadoActual = E24 or
tEstadoActual = E30 or tEstadoActual = E33 or tEstadoActual = E42 or tEstadoActual = E45 )
ELSE
    "010" WHEN (tEstadoActual= E12) ELSE

```

```

"011" WHEN (tEstadoActual= E14) ELSE
"100" WHEN (tEstadoActual= E16) ELSE
"101" WHEN (tEstadoActual= E26 or tEstadoActual = E38) ELSE
"110" WHEN (tEstadoActual= E28) ELSE
"111" WHEN (tEstadoActual= E40);

SL4 <= "0000" WHEN (tEstadoActual= E4) ELSE
"0001" WHEN (tEstadoActual= E6 or tEstadoActual = E12 or tEstadoActual= E24 ) ELSE
"0010" WHEN (tEstadoActual= E14) ELSE
"0011" WHEN (tEstadoActual= E16) ELSE
"0100" WHEN (tEstadoActual= E18) ELSE
"0101" WHEN (tEstadoActual= E26) ELSE
"0110" WHEN (tEstadoActual= E28) ELSE
"0111" WHEN (tEstadoActual= E30 or tEstadoActual = E42) ELSE
"1000" WHEN (tEstadoActual = E33 or tEstadoActual = E45)ELSE
"1001" WHEN (tEstadoActual= E38) ELSE
"1010" WHEN (tEstadoActual = E40);

SL5 <= "00" WHEN (tEstadoActual= E14) ELSE
"01" WHEN (tEstadoActual= E20) ELSE
"10" WHEN (tEstadoActual= E22) ELSE
"11" WHEN (tEstadoActual= E26);

SL6 <= "00" WHEN (tEstadoActual= E4 or tEstadoActual = E26) ELSE
"01" WHEN (tEstadoActual= E20) ELSE
"10" WHEN (tEstadoActual= E22);

SL7 <= '0' WHEN (tEstadoActual= E2) ELSE
'1' WHEN (tEstadoActual= E35 or tEstadoActual = E47);

SL8 <= '0' WHEN (tEstadoActual= E2) ELSE
'1' WHEN (tEstadoActual= E32 or tEstadoActual = E44);

ADSU1 <= '1' WHEN (tEstadoActual = E5 or tEstadoActual = E7 or tEstadoActual = E13 or
tEstadoActual = E19 or tEstadoActual = E25 or tEstadoActual = E31 or tEstadoActual = E34 or
tEstadoActual = E39 or tEstadoActual = E41 or tEstadoActual = E43 or tEstadoActual = E46 )
ELSE
'0' WHEN (tEstadoActual = E15 or tEstadoActual = E17 or tEstadoActual = E27 or
tEstadoActual = E29 );

rese1 <= '1' WHEN (tEstadoActual= E37 or tEstadoActual= E0 or tEstadoActual= E49) ELSE
'0';

rese<= '1' WHEN (tEstadoActual= E37 or tEstadoActual= E0 or tEstadoActual= E49) ELSE
'0';

salida<= '1' WHEN (tEstadoActual= E37 or tEstadoActual= E49) ELSE
'0';

END PyT1;

```

ANEXO G Código en VHDL de la FSM implementada para el modelo del ritmo circadiano usando la regla de Simpson

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
USE ieee.numeric_std.all;
ENTITY PyT IS
PORT ( sEntrada2 : IN std_logic_vector (31 downto 0);
sEntrada: IN std_logic_vector (8 downto 0);
      CLK: IN std_logic;
      sReseth: IN std_logic;
      Start: IN std_logic;
SEN1 : OUT std_logic;
SEN8 : OUT std_logic;

      SEN1, SEN2, SEN3, SEN4, SEN5, SEN6, SEN7, SEN8, SEN10,
      SEN11, SEN12, SEN13, SEN14, SEN15, SEN16, SEN17, SEN18,
      SEN19,SEN20, SEN21, SEN22, SEN23, SEN25, SEN26, SEN27,
      SEN28, SEN29, ADSU1 : OUT std_logic;
      SL6, SL5 : OUT std_logic_vector (1 downto 0);
      SL1, SL4, SL2, SL3: OUT std_logic_vector (3 DOWNT0 0);
      SL7, SL8 : OUT STD_LOGIC;
rese1 :OUT std_logic;
rese :OUT std_logic;
salida :OUT std_logic );

END PyT;

ARCHITECTURE PyT1 OF PyT IS

TYPE state_type IS (E0, E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11, E12, E13, E14, E15, E16,
E17, E18, E19, E20, E21, E22, E23, E24, E25, E26, E27, E28, E29, E30, E31, E32, E33, E34, E35,
E36, E37, E38, E39, E40, E41, E42, E43, E44, E45, E46, E47, E48, E49, E50, E51, E52, E53 );

SIGNAL tEstadoActual, tEstadoSiguiente: state_type;

Begin

clocked: PROCESS (CLK, sReseth)
BEGIN
  IF (sReseth='1') THEN
tEstadoActual<= E0;
  ELSIF (CLK'EVENT AND CLK= '1') THEN
tEstadoActual<= tEstadoSiguiente;
  END IF;
END PROCESS clocked;
```

```

LOGICA_ESTADO: PROCESS (tEstadoActual, sEntrada, sEntrada2, start)
BEGIN
CASE (tEstadoActual) IS

WHEN E0 =>
IF (start = '1') THEN
tEstadoSiguiente<= E1;
END IF;

WHEN E1=>
IF(sEntrada = "000000000") THEN tEstadoSiguiente <= E2;
END IF;
WHEN E2=>
IF(sEntrada = "000000001") THEN tEstadoSiguiente <= E3;
END IF;
WHEN E3=>
IF(sEntrada = "000000010") THEN tEstadoSiguiente <= E4;
END IF;
WHEN E4=>
IF(sEntrada = "000000011") THEN tEstadoSiguiente <= E5;
END IF;
WHEN E5=>
IF(sEntrada = "000001011") THEN tEstadoSiguiente <= E6;
END IF;
WHEN E6=>
IF(sEntrada = "000001100") THEN tEstadoSiguiente <= E7;
END IF;
WHEN E7=>
IF(sEntrada = "000010100") THEN tEstadoSiguiente <= E8;
END IF;
WHEN E8=>
IF(sEntrada = "000010101") THEN tEstadoSiguiente <= E9;
END IF;
WHEN E9=>
IF(sEntrada = "000011011") THEN tEstadoSiguiente <= E10;
END IF;
WHEN E10=>
IF(sEntrada = "000011100") THEN tEstadoSiguiente <= E11;
END IF;
WHEN E11=>
IF(sEntrada = "000100110") THEN tEstadoSiguiente <= E12;
END IF;
WHEN E12=>
IF(sEntrada = "000100111") THEN tEstadoSiguiente <= E13;
END IF;
WHEN E13=>
IF(sEntrada = "000101111") THEN tEstadoSiguiente <= E14;
END IF;
WHEN E14=>
IF(sEntrada = "000110000") THEN tEstadoSiguiente <= E15;

```

```

END IF;
WHEN E15=>
IF(sEntrada = "000111000") THEN tEstadoSiguiente <= E16;
END IF;
WHEN E16=>
IF(sEntrada = "000111001") THEN tEstadoSiguiente <= E17;
END IF;
WHEN E17=>
IF(sEntrada = "001000001") THEN tEstadoSiguiente <= E18;
END IF;
WHEN E18=>
IF(sEntrada = "001000010") THEN tEstadoSiguiente <= E19;
END IF;
WHEN E19=>
IF(sEntrada = "001001010") THEN tEstadoSiguiente <= E20;
END IF;
WHEN E20=>
IF(sEntrada = "001001011") THEN tEstadoSiguiente <= E21;
END IF;
WHEN E21=>
IF(sEntrada = "001010010") THEN tEstadoSiguiente <= E22;
END IF;
WHEN E22=>
IF(sEntrada = "001010011") THEN tEstadoSiguiente <= E23;
END IF;
WHEN E23=>
IF(sEntrada = "001011010") THEN tEstadoSiguiente <= E24;
END IF;
WHEN E24=>
IF(sEntrada = "001011011") THEN tEstadoSiguiente <= E25;
END IF;
WHEN E25=>
IF(sEntrada = "001100011") THEN tEstadoSiguiente <= E26;
END IF;
WHEN E26=>
IF(sEntrada = "001100100") THEN tEstadoSiguiente <= E27;
END IF;
WHEN E27=>
IF(sEntrada = "001101100") THEN tEstadoSiguiente <= E28;
END IF;
WHEN E28=>
IF(sEntrada = "001101101") THEN tEstadoSiguiente <= E29;
END IF;
WHEN E29=>
IF(sEntrada = "001110101" and sEntrada2 <= "00000000000000000000000000000001") THEN
tEstadoSiguiente <= E30;
ELSIF (sEntrada = "001110101" and sEntrada2 > "00000000000000000000000000000001" )
THEN tEstadoSiguiente <= E38;
END IF;
WHEN E30=>

```

```

IF(sEntrada = "001110110") THEN tEstadoSiguiente <= E31;
END IF;
WHEN E31=>
IF(sEntrada = "001111110") THEN tEstadoSiguiente <= E32;
END IF;
WHEN E32=>
IF(sEntrada = "001111111") THEN tEstadoSiguiente <= E33;
END IF;
WHEN E33=>
IF(sEntrada = "010000000") THEN tEstadoSiguiente <= E34;
END IF;
WHEN E34=>
IF(sEntrada = "010001000") THEN tEstadoSiguiente <= E35;
END IF;
WHEN E35=>
IF(sEntrada = "010001001") THEN tEstadoSiguiente <= E36;
END IF;
WHEN E36=>
IF(sEntrada = "010001010") THEN tEstadoSiguiente <= E37;
END IF;
WHEN E37=> tEstadoSiguiente <= E4;
WHEN E38=>
IF(sEntrada = "001110110") THEN tEstadoSiguiente <= E39;
END IF;
WHEN E39=>
IF(sEntrada = "001111110") THEN tEstadoSiguiente <= E40;
END IF;
WHEN E40=>
IF(sEntrada = "001111111") THEN tEstadoSiguiente <= E41;
END IF;
WHEN E41=>
IF(sEntrada = "010000111") THEN tEstadoSiguiente <= E42;
END IF;
WHEN E42=>
IF(sEntrada = "010001000") THEN tEstadoSiguiente <= E43;
END IF;
WHEN E43=>
IF(sEntrada = "010010000") THEN tEstadoSiguiente <= E44;
END IF;
WHEN E44=>
IF(sEntrada = "010010001") THEN tEstadoSiguiente <= E45;
END IF;
WHEN E45=>
IF(sEntrada = "010011001") THEN tEstadoSiguiente <= E46;
END IF;
WHEN E46=>
IF(sEntrada = "010011010") THEN tEstadoSiguiente <= E47;
END IF;
WHEN E47=>
IF(sEntrada = "010100010") THEN tEstadoSiguiente <= E48;

```

```

END IF;
WHEN E48=>
IF(sEntrada = "010100011") THEN tEstadoSiguiente <= E49;
END IF;
WHEN E49=>
IF(sEntrada = "010100100") THEN tEstadoSiguiente <= E50;
END IF;
WHEN E50=>
IF(sEntrada = "010101100") THEN tEstadoSiguiente <= E51;
END IF;
WHEN E51=>
IF(sEntrada = "010101101") THEN tEstadoSiguiente <= E52;
END IF;
WHEN E52=>
IF(sEntrada = "010101110") THEN tEstadoSiguiente <= E53;
END IF;
WHEN E53=> tEstadoSiguiente <= E4;

```

```

END CASE;
END PROCESS LOGICA_ESTADO;

```

```

SEN1 <= '1' WHEN (tEstadoActual = E6)
ELSE '0';
SEN2 <= '1' WHEN (tEstadoActual = E8)
ELSE '0';
SEN3 <= '1' WHEN (tEstadoActual = E10)
ELSE '0';
SEN4 <= '1' WHEN (tEstadoActual = E12)
ELSE '0';
SEN5 <= '1' WHEN (tEstadoActual = E14)
ELSE '0';
SEN6 <= '1' WHEN (tEstadoActual = E16)
ELSE '0';
SEN7 <= '1' WHEN (tEstadoActual = E24)
ELSE '0';
SEN8 <= '1' WHEN (tEstadoActual = E28)
ELSE '0';
SEN10 <= '1' WHEN (tEstadoActual = E3 or tEstadoActual = E36 or tEstadoActual = E52)
ELSE '0';
SEN11 <= '1' WHEN (tEstadoActual = E3 or tEstadoActual = E33 or tEstadoActual = E49)
ELSE '0';
SEN12 <= '1' WHEN (tEstadoActual = E30 or tEstadoActual= E38)
ELSE '0';
SEN13 <= '1' WHEN (tEstadoActual = E16 or tEstadoActual = E28)
ELSE '0';
SEN14 <= '1' WHEN (tEstadoActual = E8)
ELSE '0';
SEN15 <= '1' WHEN (tEstadoActual = E1)
ELSE '0';

```

```

SEN16 <= '1' WHEN (tEstadoActual = E1 )
ELSE '0';
SEN17 <= '1' WHEN (tEstadoActual = E12)
ELSE '0';
SEN18 <= '1' WHEN (tEstadoActual = E6 or tEstadoActual = E8 or tEstadoActual = E10 or
tEstadoActual = E12 or tEstadoActual = E14 or tEstadoActual = E16 or tEstadoActual = E18 or
tEstadoActual = E20 or tEstadoActual = E24 or tEstadoActual = E30 or tEstadoActual = E32 or
tEstadoActual= E38 or tEstadoActual= E40 or tEstadoActual= E42 or tEstadoActual= E44 or
tEstadoActual= E48)
ELSE '0';
SEN19 <= '1' WHEN (tEstadoActual = E6 or tEstadoActual = E8 or tEstadoActual = E14 or
tEstadoActual = E16 or tEstadoActual = E18 or tEstadoActual = E20 or tEstadoActual = E26 or
tEstadoActual = E28 or tEstadoActual = E30 or tEstadoActual = E32 or tEstadoActual = E35 or
tEstadoActual= E38 or tEstadoActual= E40 or tEstadoActual= E42 or tEstadoActual= E44 or
tEstadoActual= E46 or tEstadoActual= E48 or tEstadoActual= E51)
ELSE '0';
SEN20 <= '1' WHEN (tEstadoActual = E16 or tEstadoActual = E22 or tEstadoActual = E24 or
tEstadoActual = E28)
ELSE '0';
SEN21 <= '1' WHEN (tEstadoActual = E5 or tEstadoActual = E7 or tEstadoActual = E9 or
tEstadoActual = E11 or tEstadoActual = E13 or tEstadoActual = E15 or tEstadoActual = E17 or
tEstadoActual = E19 or tEstadoActual = E23 or tEstadoActual = E29 or tEstadoActual = E31 or
tEstadoActual = E39 or tEstadoActual = E41 or tEstadoActual = E43 or tEstadoActual = E47)
ELSE '0';
SEN22 <= '1' WHEN (tEstadoActual = E5 or tEstadoActual = E7 or tEstadoActual = E13 or
tEstadoActual = E15 or tEstadoActual = E17 or tEstadoActual = E19 or tEstadoActual = E25 or
tEstadoActual = E27 or tEstadoActual = E29 or tEstadoActual = E31 or tEstadoActual = E34 or
tEstadoActual = E39 or tEstadoActual = E41 or tEstadoActual = E43 or tEstadoActual = E39 or
tEstadoActual = E41 or tEstadoActual = E43 or tEstadoActual = E45 or tEstadoActual = E47 or
tEstadoActual = E50)
ELSE '0';
SEN23 <= '1' WHEN (tEstadoActual = E15 or tEstadoActual = E21 or tEstadoActual = E23 or
tEstadoActual = E27)
ELSE '0';
SEN25 <= '1' WHEN (tEstadoActual = E32 or tEstadoActual = E48 )
ELSE '0';
SEN26 <= '1' WHEN (tEstadoActual = E32 or tEstadoActual = E48 )
ELSE '0';
SEN27 <= '1' WHEN (tEstadoActual = E32 or tEstadoActual = E48 )
ELSE '0';
SEN28 <= '1' WHEN (tEstadoActual = E32 or tEstadoActual = E48 )
ELSE '0';
SEN29 <= '1' WHEN (tEstadoActual = E44)
ELSE '0';

SL1 <= "0000" WHEN (tEstadoActual= E4) ELSE
"0001" WHEN (tEstadoActual= E6) ELSE
    "0010" WHEN (tEstadoActual= E8) ELSE
    "0011" WHEN (tEstadoActual= E10) ELSE
    "0100" WHEN (tEstadoActual= E12 or tEstadoActual= E14 ) ELSE

```

```

"0101" WHEN (tEstadoActual= E16) ELSE
"0110" WHEN (tEstadoActual= E18 or tEstadoActual= E42 or tEstadoActual= E46) ELSE
    "0111" WHEN (tEstadoActual= E22) ELSE
"1000" WHEN (tEstadoActual = E28 or tEstadoActual = E30)ELSE
"1001" WHEN (tEstadoActual = E38 or tEstadoActual = E40);

SL2 <= "0000" WHEN (tEstadoActual= E4 or tEstadoActual = E6 or tEstadoActual = E8 or
tEstadoActual = E12 or tEstadoActual = E18) ELSE
    "0001" WHEN (tEstadoActual= E10) ELSE
    "0010" WHEN (tEstadoActual= E14) ELSE
    "0011" WHEN (tEstadoActual= E16 or tEstadoActual = E22 ) ELSE
    "0100" WHEN (tEstadoActual= E28) ELSE
    "0101" WHEN (tEstadoActual= E30) ELSE
    "0110" WHEN (tEstadoActual= E38) ELSE
    "0111" WHEN (tEstadoActual= E40) ELSE
    "1000" WHEN (tEstadoActual= E42 or tEstadoActual= E46);

SL3 <= "0000" WHEN (tEstadoActual= E4) ELSE
    "0001" WHEN (tEstadoActual= E6 or tEstadoActual= E18 or tEstadoActual = E24 or
tEstadoActual = E30 or tEstadoActual = E33 or tEstadoActual= E40 or tEstadoActual= E42 or
tEstadoActual= E49 ) ELSE
    "0010" WHEN (tEstadoActual= E12) ELSE
    "0011" WHEN (tEstadoActual= E14) ELSE
    "0100" WHEN (tEstadoActual= E16) ELSE
    "0101" WHEN (tEstadoActual= E26 or tEstadoActual= E38) ELSE
    "0110" WHEN (tEstadoActual= E28) ELSE
    "0111" WHEN (tEstadoActual= E44) ELSE
    "1000" WHEN (tEstadoActual= E46);

SL4 <= "0000" WHEN (tEstadoActual= E4) ELSE
    "0001" WHEN (tEstadoActual= E6 or tEstadoActual = E12 or tEstadoActual= E24 ) ELSE
    "0010" WHEN (tEstadoActual= E14) ELSE
    "0011" WHEN (tEstadoActual= E16) ELSE
    "0100" WHEN (tEstadoActual= E18) ELSE
    "0101" WHEN (tEstadoActual= E26) ELSE
    "0110" WHEN (tEstadoActual= E28) ELSE
    "0111" WHEN (tEstadoActual= E30 or tEstadoActual= E46) ELSE
    "1000" WHEN (tEstadoActual = E33 or tEstadoActual= E49) ELSE
    "1001" WHEN (tEstadoActual= E38) ELSE
    "1010" WHEN (tEstadoActual= E40 or tEstadoActual= E44) ELSE
    "1011" WHEN (tEstadoActual = E42);

SL5 <= "00" WHEN (tEstadoActual= E14) ELSE
    "01" WHEN (tEstadoActual= E20) ELSE
    "10" WHEN (tEstadoActual= E22) ELSE
    "11" WHEN (tEstadoActual= E26);

SL6 <= "00" WHEN (tEstadoActual= E4 or tEstadoActual = E26) ELSE
    "01" WHEN (tEstadoActual= E20) ELSE
    "10" WHEN (tEstadoActual= E22);

```

```
SL7 <= '0' WHEN (tEstadoActual= E2) ELSE  
    '1' WHEN (tEstadoActual= E35 or tEstadoActual = E51);
```

```
SL8 <= '0' WHEN (tEstadoActual= E2) ELSE  
    '1' WHEN (tEstadoActual= E32 or tEstadoActual = E48);
```

```
ADSU1 <= '1' WHEN (tEstadoActual = E5 or tEstadoActual = E7 or tEstadoActual = E13 or  
tEstadoActual = E19 or tEstadoActual = E25 or tEstadoActual = E31 or tEstadoActual = E34 or  
tEstadoActual = E39 or tEstadoActual = E41 or tEstadoActual = E43 or tEstadoActual = E45 or  
tEstadoActual = E47 or tEstadoActual = E50 ) ELSE  
    '0' WHEN (tEstadoActual = E15 or tEstadoActual = E17 or tEstadoActual = E27 or  
tEstadoActual = E29 );
```

```
rese1 <= '1' WHEN (tEstadoActual= E37 or tEstadoActual= E0 or tEstadoActual= E53) ELSE  
    '0';
```

```
rese<= '1' WHEN (tEstadoActual= E37 or tEstadoActual= E0 or tEstadoActual= E53) ELSE  
    '0';
```

```
salida<= '1' WHEN (tEstadoActual= E37 or tEstadoActual= E53 ) ELSE  
    '0';
```

```
END PyT1;
```