

Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica



**Diseño de una unidad de deslinealización y desnormalización
para un sistema de optimización de energía de uso en paneles
fotovoltaicos**

Informe de Proyecto de Graduación para optar por el título de
Ingeniero en Electrónica con el grado académico de Licenciatura

Juan José Rojas Salazar

Cartago, 21 de noviembre de 2016

INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA ELECTRÓNICA

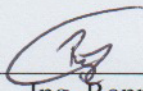
PROYECTO DE GRADUACIÓN

ACTA DE APROBACIÓN

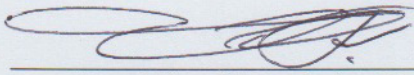
**Defensa de Proyecto de Graduación
Requisito para optar por el título de Ingeniero en Electrónica
Grado Académico de Licenciatura
Instituto Tecnológico de Costa Rica**

El Tribunal Evaluador aprueba la defensa del proyecto de graduación denominado Diseño de una unidad de deslinealización y desnormalización para un sistema de optimización de energía de uso en paneles fotovoltaicos, realizado por el señor Juan José Rojas Salazar y, hace constar que cumple con las normas establecidas por la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica.

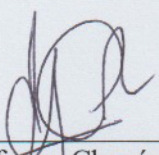
Miembros del Tribunal Evaluador


Ing. Ronny García Ramírez

Profesor lector


Ing. Miguel Hernández Rivera

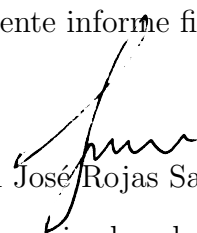
Profesor lector


Ing. Alfonso Chacón Rodríguez
Profesor asesor

Cartago, 21 de Noviembre 2016

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.



Juan José Rojas Salazar

Cartago, 21 de noviembre de 2016

Céd: 3-0458-0051

Resumen

En este trabajo, se realizó una unidad general de desnormalización-deslinealización para un sistema de optimización de energía de paneles fotovoltaicos, la cuál permite obtener los parámetros necesarios para la búsqueda del punto de máxima de potencia de los mismos. Esta se encuentra basada tanto en el estándar IEEE754 como también el algoritmo de CORDIC, se brinda así mismo una adecuada solución en software y una implementación mediante un HDL. Se muestran además, los resultados de las simulaciones Post Place & Route (recursos, potencia, reporte de tiempos) y una verificación funcional de la misma en una FPGA con un error menor al 0,5% para ambos parámetros de tensión y corriente buscados.

Palabras clave: HDL, FPGA, IEEE754, CORDIC, Verilog, Aritmética Binaria.

Abstract

In this work, a general delinearization-denormalization unit for an energy optimization system of photovoltaic panels was implemented. The unit is based on the IEEE754 standard, and uses the CORDIC algorithm. A suitable implementation is provided in HDL. Post Place & Route simulations (resources, power, timing report) and functional verification in an FPGA are also given with a maximum error of 0.5 % for both current (I_s) and voltage (α) parameters.

Keywords: HDL, FPGA, IEEE754, CORDIC, Verilog, Binary Arithmetic.

a mi querida familia

Agradecimientos

Dedico primordialmente este trabajo a Dios, quién ha sido el encargado de darme la fortaleza y entendimiento para llegar a este momento en el que daré por concluido un ciclo tan importante en mi vida.

A muchos profesores que fueron parte de mi formación como profesional, en especial al Dr. Ing. Alfonso Chacón Rodríguez, por el conocimiento brindado, así también por permitirme la oportunidad de desarrollar este proyecto y confiar en mi capacidad para el buen desenvolvimiento del mismo.

A mi querida familia, en especial mis padres, por inculcar valores y actitudes que me han permitido desenvolverme en diversos aspectos y que me han formado como la persona que soy hoy en día, por todo el apoyo a lo largo de mi vida y los sacrificios realizados a través de los años para brindarme el acceso a una educación de calidad.

Por último, agradecer infinitamente a todas las personas que han formado parte de este camino y han colaborado de manera significativa en la consecución de este logro.

Juan José Rojas Salazar

Cartago, 21 de noviembre de 2016.

Índice general

Índice de figuras	ii
Índice de tablas	iii
1 Introducción	1
1.1 Entorno del proyecto	1
1.2 Descripción del problema y justificación	2
1.3 Síntesis del problema	2
1.4 Meta	3
1.5 Objetivos	3
1.5.1 Objetivo general	3
1.5.2 Objetivos específicos	3
1.6 Enfoque de la solución	4
1.7 Estructura	5
2 Marco teórico	6
2.1 Panel Fotovoltaico	6
2.1.1 Curvas Corriente-Tensión (I-V) para un PV	6
2.1.2 Modelo general del panel fotovoltaico	7
2.2 Modelo de estimación	9
2.3 Estándar IEEE 754	11
2.4 Formato de representación binaria coma fija	11
2.4.1 Representación de números negativos	12
2.5 Algoritmo de CORDIC	12
2.5.1 Sistema de coordenadas hiperbólico	16
2.5.2 Exponencial utilizando el algoritmo hiperbólico de CORDIC	17
2.5.3 Expansión del rango de convergencia del algoritmo hiperbólico	18
3 Modelo teórico del algoritmo estimador de parámetros	20
3.1 Estimación de parámetros	20
3.2 Dinámica en la fase plana	23
3.3 Resultados del modelo teórico	27
4 Sistema de conversión coma fija a coma flotante y desnormalización	29
4.1 Diseño del sistema de conversión, desnormalización y control de datos	29

4.2	Convertidor coma fija - coma flotante y desnormalizador	31
4.3	Control para el convertidor coma fija - coma flotante y desnormalizador . .	35
4.4	Verificación y resultados del módulo conversión-desnormalización mediante Verilog HDL	36
5	Sistema de deslinealización	40
5.1	Función exponencial utilizando el algoritmo de CORDIC hiperbólico en software	40
5.2	Sistema de deslinealización implementado en hardware (CORDIC)	44
5.3	Diseño e implementación del coprocesador de deslinealización por medio del algoritmo de CORDIC	46
5.4	Ajuste de rango para unidad de deslinealización	49
5.5	Sistema de control para la unidad de coprocesamiento CORDIC por medio de un máquina de estados finita (FSM)	51
5.6	Reducción de hardware en la arquitectura CORDIC	51
5.7	Verificación y resultados del sistema de deslinealización mediante Verilog HDL	54
5.7.1	Resultados de la simulación del rango de convergencia del circuito deslinealizador CORDIC	55
6	Prueba completa de la unidad sobre una placa de desarrollo Nexys 4	57
6.1	Descripción de la unidad completa bajo prueba	57
6.2	Simulación generada para la unidad completa	59
6.3	Sistema para realización de pruebas en una placa de desarrollo Nexys 4 . .	60
6.4	Resultados de las pruebas de la unidad general en una placa de desarrollo Nexys 4	61
6.5	Recursos utilizados	65
6.6	Reporte de tiempos	65
6.7	Consumo de potencia	66
7	Conclusiones y recomendaciones	68
7.1	Conclusiones	68
7.2	Recomendaciones	69
	Bibliografía	71

Índice de figuras

1.1	Diagrama de solución para el sistema completo de aumento de eficiencia energética en un panel fotovoltaico	4
1.2	Diagrama de solución para el sistema de desnormalización y deslinealización, con entradas $\hat{\theta}_1 - \hat{\theta}_2$ en coma fija-normalizadas y salidas $\alpha - I_s$ en coma flotante-desnormalizadas.	5
2.1	Curva característica de corriente(A)-tensión(V) y potencia(P)-tensión(V) para un un panel fotovoltaico [1]	7
2.2	Modelo general para un panel fotovoltaico, compuesto por un diodo, una fuente de corriente y pérdidas resistivas por cada celda	8
2.3	Vector bidimensional antes y después de ser rotado. [2]	13
2.4	Pseudorotación de un vector bidimensional. [3]	13
3.1	Gráficas $i-v$ y $P-v$ obtenidas con el modelo para el panel Kyocera Solar KC65T con parámetros $\theta_1 = 0.625$ y $\theta_2 = \ln(I_s) = -12.178$	22
3.2	Evolución de parámetros $\hat{\theta}_1$ y $\hat{\theta}_2$ en el plano bidimensional antes de alcanzar la trayectoria de la isoclina de crecimiento cero	24
3.3	Dinámica de los parámetros $\hat{\theta}_1$ y $\hat{\theta}_2$ mostrando su evolución en el tiempo. Tiempo de simulación= 0.030 segundos	25
3.4	Evolución de parámetros $\hat{\theta}_1$ y $\hat{\theta}_2$ en el plano bidimensional una vez alcanzado el punto de equilibrio planteado	26
3.5	Dinámica de los parámetros $\hat{\theta}_1$ y $\hat{\theta}_2$ mostrando su evolución en el tiempo. Tiempo de simulación= 15 segundos	26
3.6	Porcentaje de error obtenido para la convergencia de $\theta_1 = \alpha$	27
3.7	Porcentaje de error obtenido para la convergencia de $\theta_2 = \ln(I_s)$	28
4.1	Diagrama general de entradas y salidas para el convertidor coma fija a coma flotante y desnormalizador	29
4.2	Sistema de conversión, desnormalización y control utilizando una máquina de estados.	30
4.3	Diagrama general del sistema de conversión de coma fija a coma flotante y desnormalización de parámetros $\alpha - \ln(I_s)$ (parámetros dependientes de la irradiancia y temperatura)	31

4.4	Circuito de conversión coma fija a coma flotante y desnormalización de parámetros α y $\ln(I_s)$, con un dato de entrada en coma fija, posteriormente desnormalizado y salida en coma flotante	32
4.5	Control básico ejecutado por una máquina de estados finita para la correspondiente conversión-desnormalización	35
4.6	Esquema general para la comprobación de datos del módulo conversión-desnormalización	36
4.7	Simulación del circuito de conversión y desnormalización con una entrada de 1000 valores provenientes del modelo teórico del estimador	37
4.8	Porcentaje de error obtenido para la conversión-desnormalización de α teórica y la simulación post-implementación del circuito	37
4.9	Porcentaje de error obtenido para la conversión-desnormalización de $\ln I_s$ teórico y la simulación post-implementación del circuito	38
5.1	Algoritmo de CORDIC en Python para el cálculo de la función exponencial	41
5.2	Algoritmo de CORDIC en Python con expansión del rango de convergencia con dos iteraciones negativas	42
5.3	Representación binaria punto flotante de cada una de las tablas 5.1 y 5.2 .	43
5.4	Bloque de deslinealización: Entradas y salidas para el cálculo de la función exponencial basada en el algoritmo de CORDIC e implementación en hardware	44
5.5	Sistema de deslinealización: coprocesador punto flotante de precisión simple (32 bits) basado en el algoritmo de CORDIC y una máquina de estados finita	44
5.6	Coprocesador segmentado para el cálculo de una función exponencial con el algoritmo de CORDIC en hardware	46
5.7	Circuito de comparación de signo actual δ , para las variables X_{i+1} , Y_{i+1} y Z_{i+1} de la iteración siguiente, utilizando la tabla 5.3.	48
5.8	Esquema ideado incluyendo la multiplicación para determinar el cálculo de la función exponencial	49
5.9	Coprocesador segmentado modificado para el cálculo de una función exponencial con el algoritmo de CORDIC para el parámetro $\theta_2 = \ln(I_s)$	50
5.10	Máquina de estados finitos para la unidad de coprocesamiento CORDIC. .	51
5.11	Algoritmo de CORDIC en Python con expansión del rango de convergencia con siete iteraciones negativas	52
5.12	Coprocesador segmentado con reducción de hardware para el cálculo de una función exponencial con el algoritmo de CORDIC	53
5.13	Máquina de estados finitos para la unidad de coprocesamiento CORDIC. .	53
5.14	Esquema general para la comprobación de datos del sistema de deslinealización	54
5.15	Simulación de la arquitectura modificada implementada en Verilog, ingresando un archivo de texto, con 1000 valores del intervalo de convergencia para θ_2	54

5.16	Simulación de la arquitectura reducida implementada en Verilog, ingresando un archivo de texto, con 1000 valores del intervalo de convergencia para θ_2	54
5.17	Porcentaje de error para los datos de la simulación post-implementación del deslinealizador, tomando mil valores de entrada dentro del rango de convergencia y 15 iteraciones para el algoritmo de CORDIC	55
5.18	Porcentaje de error para los datos de la simulación post-implementación de la arquitectura reducida para el algoritmo de CORDIC	56
6.1	Unidad general del sistema conversión - desnormalización y deslinealización de parámetros	57
6.2	Unidad general completa que contiene los sistemas de conversión coma fija a coma flotante y deslinealización	58
6.3	Simulación de la unidad completa para un modelo de estimación de parámetros	59
6.4	Diagrama de flujo general para el ambiente de verificación utilizado en la FPGA, utilizando un tester con una memoria ROM para los vectores de entrada y una UART para la transmisión de los resultados hacia el computador	60
6.5	Detalle del ambiente de verificación ingresando los datos de estímulo por medio de una memoria ROM y enviando los datos de salida por medio de transmisión serial (UART) desde la Nexys 4 hacia un computador	60
6.6	Porcentaje de error para la conversión-desnormalización de α obtenido mediante el sistema de verificación	62
6.7	Porcentaje de error para la conversión-desnormalización/deslinealización de I_s obtenido mediante el sistema de verificación	63
6.8	Resultado de la conversión-desnormalización/deslinealización para I_s obtenido mediante el sistema de verificación	64
7.1	Floorplanning realizado para ambas unidades.	69

Índice de tablas

2.1	Modelos para un PV: ideal, con pérdidas en serie R_s y con pérdidas en paralelo R_p	9
2.2	Sistema de coordenadas unificado (CORDIC): circular, lineal e hiperbólico. [4]	16
2.3	Valores de χ_{max} con el algoritmo hiperbólico extendido.	19
3.1	Especificaciones del comportamiento eléctrico del panel fotovoltaico KC65T, bajo condiciones estándar de prueba (STC) [5].	21
3.2	Parámetros considerados para la simulación del estimador para un tiempo reducido	25
3.3	Parámetros considerados para la simulación del estimador para un tiempo extenso	27
4.1	Resultados experimentales obtenidos de la simulación post-implementación, para los valores de entrada α y $\ln I_s$, presentes en el circuito de conversión-desnormalización.	38
5.1	Cálculo de las primeras dos iteraciones negativas para e^{-3}	43
5.2	Cálculo de las primeras dos iteraciones negativas para e^3	43
5.3	Signo δ de la iteración siguiente para las variables X_{i+1} , Y_{i+1} y Z_{i+1} , comparando el signo de las variables X_i , Y_i y Z_i contra el signo de Z_i	47
5.4	Resultados experimentales obtenidos por medio de una simulación post-implementación para I_s	55
6.1	Comparación de resultados de simulación vs experimentales de α obtenidos por el sistema de verificación implementado en una placa de desarrollo Nexys 4. El funcionamiento en la FPGA es por tanto correcto al 100% contra el modelo.	63
6.2	Comparación de resultados de simulación vs experimentales de I_s obtenidos por el sistema de verificación implementado en una placa de desarrollo Nexys 4. El funcionamiento en la FPGA es por tanto correcto al 100% contra el modelo.	64
6.3	Resumen del reporte post implementación del uso de dispositivos generado por la herramienta Vivado.	65
6.4	Resumen del reporte post implementación de tiempos del circuito completo, a partir de la herramienta Vivado. CLK=100 MHz	66

6.5	Resumen del reporte post implementación de la potencia estática, dinámica y total, de la herramienta Vivado para la unidad general.	66
6.6	Resumen del reporte generado por la herramienta Vivado que indica el consumo de potencia de diversos elementos.	67

Capítulo 1

Introducción

1.1 Entorno del proyecto

Actualmente a nivel mundial el impacto de las energías renovables se encuentra en un constante crecimiento y dentro de ellas, la mayoría viene generada por medios indirectos o directos del sol. La energía solar es usualmente utilizada para ser convertida en electricidad por medio de paneles fotovoltaicos, cuyo principio de funcionamiento proviene de sus materiales semiconductores de construcción [6]. Nuestro país no escapa a este fenómeno y es por esto que actualmente ha aumentado el uso de los mismos de manera drástica, basado en el hecho de que estos brindan múltiples beneficios en una gran cantidad de sectores debido a su fiabilidad y versatilidad [7].

Los sistemas fotovoltaicos requieren de diversos componentes, entre los que se pueden citar: paneles solares, acumuladores de energía, inversores de onda sinusoidal, protectores de sobretensión y reguladores [8]; no obstante, estos no cuentan con un mecanismo que permita adecuar la tensión para operar constantemente en el punto de máxima potencia, por lo que este varía por efectos de las magnitudes eléctricas de la corriente y la tensión con respecto a la temperatura e irradiancia del medio en el que se encuentra, de manera que si la tensión varía, la potencia asociada a esa tensión también lo hace. El rendimiento de un panel fotovoltaico está estrictamente relacionado a la eficiencia de convertir luz solar en electricidad; sin embargo, no toda la luz es ciertamente aprovechada por el material del que se encuentra constituido; típicamente la eficiencia se sitúa en un 15%, cercano a un sexto de la luz incidente en las celdas [9] [10].

El objetivo principal se centra en buscar el punto de tensión donde se obtenga la máxima potencia por lo cual es importante aprovechar las mejores condiciones ambientales presentes con un sistema capaz de controlar más eficientemente la energía.

El diseño realizado tanto del modelo como de las unidades presentes en esta tesis se basará en las características eléctricas de operación de un panel previamente escogido: KC65T Kyocera Solar [5]. El Laboratorio de Diseño de Circuitos Integrados [DCILab] (Instituto Tecnológico de Costa Rica) ya cuenta con una pequeña incursión en este problema [11], donde se dio la aproximación general del sistema total requerido para cumplir con tal objetivo. Este mismo no fue concluido de manera total, por lo que se requiere el diseño de las unidades restantes.

1.2 Descripción del problema y justificación

El principal motivo del desarrollo de esta tesis surge debido a la falta de un modelo práctico para estimar parámetros que sea capaz de aproximar los valores de corriente de saturación y voltaje térmico necesarios para aproximar el punto de potencia máximo de un panel. Debido a que este estimador será implementado en un lenguaje de descripción de hardware, idealmente en formato coma fija con una linealización previa, es necesario crear un par de unidades que sean capaces tanto de convertir datos a formato coma flotante, como también deslinealizarlos.

La aproximación de parámetros se da con base en los datos de las curvas características de $I_{pv} - V_{pv}$ y $P_{pv} - V_{pv}$ de una celda solar. Sin embargo, la corriente I_{pv} no tiene un comportamiento lineal, por lo que en [11] se dió el trabajo necesario para linealizar y normalizar los datos para que en un desarrollo futuro del estimador este sea capaz de procesarlos.

La deslinealización depende de las operaciones aritméticas en coma flotante suma y multiplicación. En este aspecto el DCILab ya tiene un mínimo de experiencia en el desarrollo de las mismas ([12] [13]).

Los siguientes requerimientos serán contemplados como parte de la solución:

- El deslinealizador se deberá basar en el estándar IEEE 754 (coma flotante).
- Utilización de arquitecturas de 32 bits.
- Utilizar Verilog como lenguaje de descripción de hardware y el software suite Vivado (Xilinx).
- Optimización de las unidades para requerir la menor cantidad de recursos y menor tiempo de ejecución.
- Utilización de lenguajes de alto nivel para determinar la fiabilidad de las unidades.

1.3 Síntesis del problema

¿Cómo implementar las unidades de desnormalización y deslinealización en un lenguaje de descripción de hardware de manera que permita optimizar la eficiencia de paneles fotovoltaicos?

1.4 Meta

Desarrollar un sistema en un lenguaje de descripción de hardware que logre incrementar sustancialmente la eficiencia de un panel fotovoltaico, el cual permita seguir impulsando de manera paulatina la utilización de energías limpias en una amplia cantidad de sectores.

1.5 Objetivos

1.5.1 Objetivo general

Desarrollar las unidades de desnormalización y deslinealización para un estimador de parámetros corriente (I)-tensión (V) de un panel fotovoltaico.

1.5.2 Objetivos específicos

- Desarrollar un modelo en un lenguaje de alto nivel del comportamiento del algoritmo estimador de parámetros, con predicción de la dinámica de $\hat{\theta}_1 = \alpha$ y $\hat{\theta}_2 = \ln(I_s)$.
Indicador: obtener la dinámica de la trayectoria de los parámetros en la fase plana y alcance del equilibrio con al menos una precisión del 95%.
- Crear un circuito de desnormalización para los parámetros $\hat{\theta}_1$ y $\hat{\theta}_2$ en coma fija, a la salida del estimador de parámetros.
Indicador: Verificar mediante un programa de alto nivel la precisión del desnormalizador con un error menor al 5%.
- Crear un circuito capaz de convertir de coma fija a coma flotante, las salidas θ_1 y θ_2 del desnormalizador.
Indicador: Verificar mediante un programa de alto nivel la precisión del convertidor con un error menor al 5%.
- Crear un circuito que deslinealice el parámetro θ_2 , por medio de una operación exponencial, para obtener como parámetro de salida I_s .
Indicador: Verificar mediante un programa de alto nivel la precisión del algoritmo implementado con un error menor al 5%.
- Integrar las unidades de desnormalización y deslinealización creadas con el estimador de parámetros, para determinar el funcionamiento del sistema.
Indicador: Comprobar que las unidades presentan un error de 0 % al ser interconectadas y que la precisión de los algoritmos se mantenga con un error menor al 5%.

1.6 Enfoque de la solución

Se determinarán primeramente las curvas características y modelos de un panel fotovoltaico, debido a que son el fundamento de la estimación de parámetros. Para ello se desarrollará un modelo teórico de todo el sistema en un programa de alto nivel, específicamente en Python. Se investigarán temas tales como: estándar IEEE 754, representación binaria en coma fija y aspectos relacionados al algoritmo de CORDIC. Se seguirá una metodología de diseño Top-Down, para diseño y simulación de sistemas digitales:

- Definición de la interfaz de entrada-salida de cada una de las unidades, con una descripción funcional de la operación en conjunto.
- Creación de una estrategia factible de conexión entre bloques.
- Descripción mediante una herramienta EDA, en este caso Vivado (Xilinx), con el lenguaje de descripción de hardware Verilog. Esto para obtener una correspondiente simulación y verificación funcional antes de su posterior implementación.
- Comparación de resultados de la implementación del diseño en un dispositivo programable FPGA Nexys 4 contra resultados obtenidos mediante herramientas de software como Python y GNU Octave.

Todo esto formará parte del sistema general el cuál se muestra en la figura 1.1, en el que se desarrollarán los bloques enmarcados.

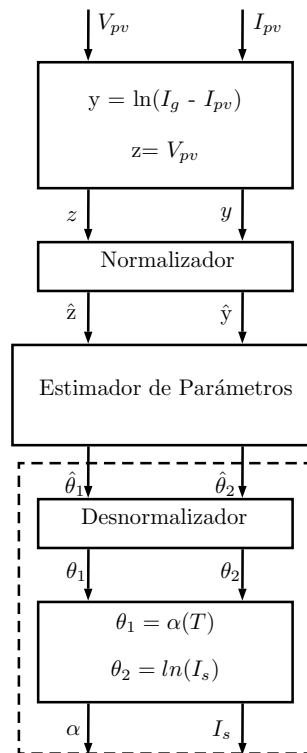


Figura 1.1: Diagrama de solución para el sistema completo de aumento de eficiencia energética en un panel fotovoltaico.

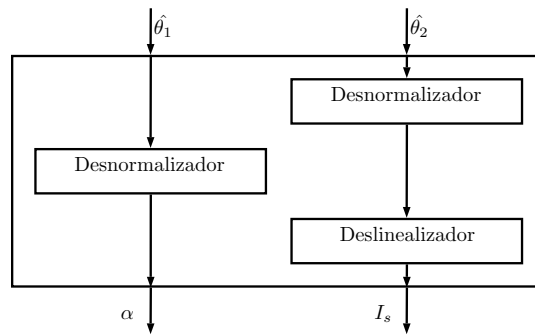


Figura 1.2: Diagrama de solución para el sistema de desnormalización y deslinealización, con entradas $\hat{\theta}_1 - \hat{\theta}_2$ en coma fija-normalizadas y salidas $\alpha - I_s$ en coma flotante-desnormalizadas.

1.7 Estructura

El capítulo 2 muestra la información que será utilizada a lo largo de la tesis y que fundamentará los conceptos básicos de los que se hacen uso. En el capítulo 3 se desarrollará el modelo teórico del estimador de parámetros, a través de simulaciones en un programa de alto nivel. En el capítulo 4 y 5 se da el desarrollo, implementación y comprobación del error de las unidades de conversión-desnormalización y deslinealización respectivamente. En el capítulo 6 se muestra una comprobación general de las unidades en la FPGA Nexys 4 y por último en el capítulo 7 se presentan las conclusiones y recomendaciones.

Capítulo 2

Marco teórico

2.1 Panel Fotovoltaico

Un panel fotovoltaico está compuesto por una gran cantidad de celdas solares, estas consisten en la unión de un par de capas delgadas conocida como junta $p-n$ compuesta por materiales semiconductores no similares. Las capas n semiconductoras se encuentran hechas de silicón cristalino que es dopado con una cantidad pequeña de impurezas (fósforo, por ejemplo) para generar una gran cantidad de electrones libres. Igualmente las capas p son de silicón cristalino pero dopadas generalmente con boro, haciendo que esta tenga un déficit de electrones libres (huecos) [14].

Si la energía proveniente de la incidencia solar producida por un fotón es igual o mayor que la banda prohibida del semiconductor, esta hará que en la junta se dé una recombinación de electrones y huecos, de la capa n a la p y viceversa. Es por esto que se produce un efecto en el que se da una mayor cantidad de cargas positivas alrededor de la capa n semiconductor. Los semiconductores del panel, mayormente poseen un comportamiento exponencial y no lineal en sus regiones intrínsecas, contrario a las regiones extrínsecas [14].

2.1.1 Curvas Corriente-Tensión (I-V) para un PV

Un panel fotovoltaico produce su máxima corriente cuando no tiene ninguna resistencia en el circuito, es decir cuando existe un corto circuito en las terminales positiva y negativa. Esta corriente es generalmente conocida como corriente de corto circuito o I_{sc} y en este el voltaje en el circuito es de cero [15].

Contrario a I_{sc} , el voltaje máximo ocurre cuando no se tiene ninguna carga presente en el circuito. Este se conoce como voltaje de circuito abierto V_{oc} y bajo esta condición la resistencia es aproximada a infinito y no existe ninguna corriente circulante debido a que no se tiene ninguna carga presente [16].

Cuando se realiza la caracterización de un PV, se toman en consideración ambas pruebas debido a que la unión de ambas generará una curva de corriente-tensión que describirá el modelo eléctrico del panel. Esta caracterización es utilizada para determinar el rendimiento de un PV. La curva I-V se genera bajo condiciones estándar de prueba conocidas como STC, en la que se establece una determinada cantidad de irradiancia, generalmente 1000 W/m^2 y un valor de temperatura del dispositivo a 25 grados centígrados. [17]

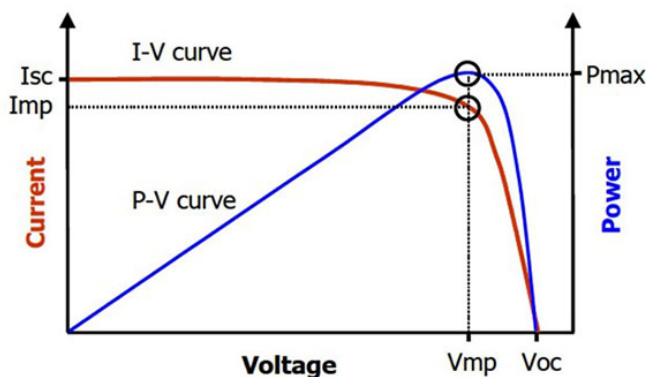


Figura 2.1: Curva característica de corriente(A)-tensión(V) y potencia(P)-tensión(V) para un panel fotovoltaico [1].

El punto de máxima potencia de un PV, puede ocurrir para una gran variedad de valores de corriente (I) y voltaje (V), al incrementar la carga resistiva desde el corto circuito I_{sc} hasta un valor muy alto cercano a circuito abierto V_{oc} , donde es posible determinar P_{max} , el punto en el que se genera la máxima salida eléctrica de potencia que la celda es capaz de proveer a un cierto nivel de irradiancia. Está se expresa como $P_{max} = V_{max} I_{max}$ [18]

2.1.2 Modelo general del panel fotovoltaico

Un panel fotovoltaico se puede modelar utilizando una fuente de corriente en paralelo, un diodo, una resistencia paralela y una resistencia en serie, como se ejemplifica en la figura 2.2 [11].

En esta se incluyen las variables características del panel:

- Dependencia de la temperatura, corriente de saturación del diodo (I_s) y fotocorriente (I_g).
- Pérdidas debidas al flujo de corriente (R_s) y pérdidas con referencia a tierra (R_p).
- Número de celdas en análisis n .

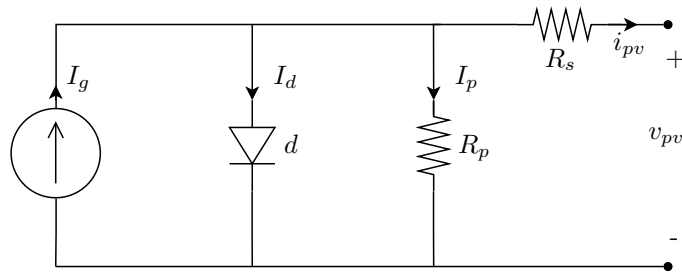


Figura 2.2: Modelo general para un panel fotovoltaico, compuesto por un diodo, una fuente de corriente y pérdidas resistivas por cada celda.

A partir de este modelo es posible deducir la ecuación que describe las corrientes i_{pv} e I_g , como sigue a continuación [11]:

$$i_{pv} = I_g - i_d + i_p \quad (2.1)$$

$$I_g = 2i_{pv} + \frac{v_{pv} + i_{pv}R_s}{R_p} - I_s + I_s e^{\frac{v_{pv} + i_{pv}v_{pv}}{nv_t}} \quad (2.2)$$

Despejando i_{pv}

$$i_{pv} = \frac{1}{2} \left[I_s + I_g - \frac{v_{pv} + i_{pv}R_s}{R_p} - I_s e^{\frac{v_{pv} + i_{pv}v_{pv}}{nv_t}} \right] \quad (2.3)$$

La corriente que fluye por las terminales de un generador fotovoltaico está determinada por tres funciones de corriente:

- I_g : Corriente generada debido al efecto fotoeléctrico.
- i_d : Corriente de pérdida debido a la juntura p-n.
- i_p : Corriente de pérdida de naturaleza resistiva.

Las siguientes suposiciones se consideran para el comportamiento estático del generador fotovoltaico:

- I_g : depende de la irradiancia (S), pero no depende de la tensión en las terminales del generador fotovoltaico v_{pv} .
- i_p e i_d : dependen de la tensión v_{pv} .
- i_p : Depende de la temperatura (T).

De esta forma, la expresión que define i_{pv} es:

$$i_{pv}(v_{pv}, T, S) = i_g(v_{pv}) - i_d(v_{pv}, T) \quad (2.4)$$

Según se definan las funciones i_{pv} e i_d , se obtendrán modelos con complejidad y precisiones distintas, a partir de los casos enumerados en la tabla 2.1

Tabla 2.1: Modelos para un PV: ideal, con pérdidas en serie R_s y con pérdidas en paralelo R_p

Modelos	i_g	i_p	i_d
1	KS	-	$I_s(T) \left[e^{\frac{V_{pv}}{v_t}} - 1 \right]$
2	KS	$G_p V_{pv}$	$I_s(T) \left[e^{\frac{V_{pv}}{v_t}} - 1 \right]$
3	KS	-	$I_s(T) \left[e^{\frac{V_{pv} + i_{pv} R_s}{v_t}} - 1 \right]$
4	KS	$G_p V_{pv} + G_p i_{pv} R_s$	$I_s(T) \left[e^{\frac{V_{pv} + i_{pv} R_s}{v_t}} - 1 \right]$

De manera general se tiene para el modelo general del comportamiento estático de un generador PV:

$$i_{pv}(v_{pv}) = KS - G_p v_{pv} - G_p i_{pv} R_s - i_d \quad (2.5)$$

$$i_{pv} = KS - G_p v_{pv} + I_s(T) - G_p i_{pv} R_s - I_s(T) e^{\frac{v_{pv}}{v_t}} e^{\frac{i_{pv} R_s}{v_t}} \quad (2.6)$$

2.2 Modelo de estimación

Para propósitos del modelo del panel fotovoltaico se considera al mismo como un sistema de dos puertos, de donde se conocen sus magnitudes eléctricas: corriente circulante a través del dispositivo i_{pv} y tensión en sus terminales v_{pv} , respectivamente. Estas se encuentran asociadas por una función no lineal dependiente de los parámetros de temperatura T e irradiación solar S . Del panel se conocen también un set de parámetros $\rho = (\rho_1, \rho_2, \dots, \rho_n)$ asociados a los materiales de fabricación empleados. La cantidad de n dependerá del número de componentes utilizados en la creación del panel [19], [20], [21], [22].

En la mayoría de caracterizaciones de paneles, la relación entre la corriente y tensión, i_{pv} y v_{pv} , está representada por

$$i_{pv} = \xi(S, T, \rho) - \varphi(S, T, v_{pv}, \rho) \quad (2.7)$$

con $\xi(\cdot) \geq 0$ y $\varphi(\cdot) \geq 0$ funciones dependientes de la tecnología del panel.

La ecuación (2.7) para un gran cantidad de paneles fotovoltaicos (mono y policristalinos) [19] puede ser aproximada a:

$$i_{pv} = \xi(S, \rho) - \varphi_1(S, v_{pv}, i_{pv}, \rho) - \varphi_2(T, v_{pv}, i_{pv}, \rho) \quad (2.8)$$

donde efectivamente se separa la ecuación para establecer la relación directa a los parámetros de temperatura e irradiación solar (T y S). Sin embargo es posible simplificar la ecuación 2.8 considerando el caso en el que

$$\varphi_2(T, v_{pv}, i_{pv}, \rho) = \Psi(T, \rho)e^{\alpha(T, \rho)z(\rho, v_{pv}, i_{pv})} \quad (2.9)$$

con Ψ y α dependientes de la temperatura y ρ . Z es una nueva variable introducida y está estrictamente relacionada a ρ , v_{pv} e i_{pv} . Se puede reescribir la ecuación (2.8) en una manera más sencilla

$$y(S, \rho, i_{pv}, v_{pv}) = \theta_1(T, \rho)z(i_{pv}, v_{pv}, \rho) + \theta_2(T, \rho) \quad (2.10)$$

donde

$$y = \ln(\xi(S, \rho) - i_{pv} - \varphi_1(S, v_{pv}, i_{pv}, \rho)) \quad (2.11)$$

$$\theta_1 = \alpha(T, \rho) \quad \theta_2 = \ln(\Psi(T, \rho)) \quad (2.12)$$

Un modelo simple de estimación puede plantearse para obtener los valores de θ_1 y θ_2 a partir de las relaciones dadas por las ecuaciones anteriores y la tabla 2.1. Se deduce del modelo eléctrico de un panel fotovoltaico que cuenta con un único diodo y una resistencia serie, que su corriente i_{pv} está dada en términos de

$$i_{pv} = I_g(S) + I_s(T) - I_s(T)e^{\alpha(T)V_{pv}} \quad (2.13)$$

A partir de esta relación, se puede obtener mediante una linealización con $I_g \gg I_s$

$$\ln(I_s(T)e^{\alpha(T)v_{pv}}) = \ln(I_g(S) - i_{pv}) \quad (2.14)$$

$$\alpha V_{pv} + \ln(I_s) = \ln(I_g - i_{pv}) \quad (2.15)$$

con

$$y = \alpha V_{pv} + \ln(I_s) \quad (2.16)$$

Una vez que el estimador es capaz de determinar los parámetros necesarios, la ecuación 2.17 se mantiene constante en un plano de fases [19] [23].

$$y = \theta_1 z + \theta_2 = \hat{\theta}_1 z + \hat{\theta}_2 \quad (2.17)$$

2.3 Estándar IEEE 754

El estándar IEEE754 coma flotante es una de las representaciones más comunes para números reales en diversa variedad de sistemas computacionales hoy en día. Este estándar permite representar un número por medio del uso de la notación científica, con un número base y un exponente [24], [25]. La representación de un dato en coma flotante para un número real está representado por tres partes principales: signo, exponente y mantisa (número normalizado en notación científica), de la forma:

$$(-1)^{\text{signo}} \cdot b^{\text{exponente}} \cdot \text{mantisa}(\text{significando})$$

Esta representación posee un 1 bit de signo, 8 bits de exponente y 23 bits de mantisa para precisión simple (32 bits), donde valores de 0 y 1 en el signo representan valores positivos y negativos respectivamente [26]. El exponente necesita representar tanto exponentes positivos como negativos, un valor de *bias* es añadido con tal de obtener un valor adecuado en flotante, generalmente para el estándar en precisión simple se encuentra definido en 127, por lo tanto el rango original de -126 a +127 con la suma del bias puede variar el rango desde 1 hasta 254 (los valores de 0 y 255 poseen valores especiales en este formato) [27].

La mantisa o significando representa la precisión del número y está compuesto por un bit implícito (a la izquierda de la coma flotante) y los bits de precisión a la derecha de este, típicamente los valores son guardados en su forma normalizada.

2.4 Formato de representación binaria coma fija

La mayoría de computadoras poseen soporte para números en formato coma flotante, sin embargo, esta no es la única representación que existe para números fraccionales. La representación binaria coma fija tiene un conocido uso en el procesamiento digital de señales (DSP) y diversidad de aplicaciones donde se considera más importante el rendimiento que la precisión. La aritmética de coma fija se considera mucho más veloz que la de coma flotante ya que permite realizar operaciones con un comportamiento similar al de un número entero [28].

La ventaja que posee es que permite conversiones sumamente veloces e inmediatas, con la desventaja en el rango o dominio en comparación con la notación científica, además de que las operaciones tienden a perder precisión por redondeo. La representación fraccional de un número posee el concepto de una coma binaria el cual es similar a cuando se habla de coma decimal bajo un sistema decimal, éste actúa como un divisor entre la parte entera y la parte fraccional. Para representar un número binario en coma fija se necesita definir un valor o posición donde se encontrará, para esto se necesita definir al menos dos parámetros:

- Tamaño de la representación numeral.
- Posición de la coma binaria dentro del número.

El esquema general para representar un número en binario punto fijo se define como

$$\textit{Signo} - \textit{Magnitud} - \textit{Fraccional}$$

A excepción del signo (*tamaño=1*), la magnitud y la parte fraccional pueden ser modificadas a discreción para representar un rango adecuado según la aplicación que se ejecute.

2.4.1 Representación de números negativos

Debido a que se pueden encontrar números negativos en este formato, es necesario expresarlos en una forma adecuada. La representación del complemento a dos del número es la manera más común de hacerlo. Una de las propiedades de este complemento es que las operaciones aritméticas de suma y resta de números positivos o negativos son idénticas, esto incluye también al desplazamiento. Incluso, es posible dividir números negativos representados en complemento a dos con un simple desplazamiento a la derecha de 1 bit con extensión de signo, al igual que como se hace con los números positivos [28].

2.5 Algoritmo de CORDIC

Coordinate Rotational Digital Computer (CORDIC) es un algoritmo simple y eficiente de iteraciones básicas, que se clasifica generalmente como uno de los algoritmos de la clase desplazamiento y suma (*shift and add*), donde se genera un método de rotación de vectores con el fin de calcular un valor numérico deseado. La variedad de usos es sumamente amplia, entre los que se puede encontrar: funciones trigonométricas, logarítmicas, exponenciales, procesamiento digital de señales, transformada discreta de Fourier y transformadas discretas de Hartley [29].

Este algoritmo no es precisamente una de las técnicas más rápidas para implementar todas las funciones anteriormente descritas; sin embargo, debido a su simplicidad de hardware, es ampliamente utilizado, por ser muy eficiente en términos de potencia y complejidad [30]. CORDIC ofrece una solución a los problemas de recursos en hardware, generalmente en recursos como multiplicación, donde puede ser reemplazada por sumadores y *shifters*.

El algoritmo en su forma original [31] describe la rotación de un vector bidimensional en el plano cartesiano, como el mostrado en la Figura 2.3. El funcionamiento del algoritmo se deduce de la fórmula general de rotación de vectores [32]

$$\begin{aligned} x' &= x \cos \theta - y \sin \theta \\ y' &= y \cos \theta + x \sin \theta \end{aligned} \tag{2.18}$$

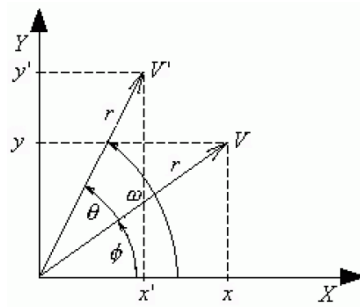


Figura 2.3: Vector bidimensional antes y después de ser rotado. [2]

Se puede asumir que $\cos\theta \neq 0$ y simplificando la ecuación 2.18, se obtiene [32]

$$\begin{aligned} x' &= \cos\theta(x - y\tan\theta) \\ y' &= \cos\theta(y + x\tan\theta) \end{aligned} \quad (2.19)$$

Se puede reexpresar $\cos\theta$ como:

$$\cos\theta = \frac{1}{\sqrt{1 + \tan^2\theta}} \quad (2.20)$$

lo que deriva en

$$\begin{aligned} x' &= \frac{x - y\tan\theta}{\sqrt{1 + \tan^2\theta}} \\ y' &= \frac{y + x\tan\theta}{\sqrt{1 + \tan^2\theta}} \end{aligned} \quad (2.21)$$

Las rotaciones pueden ser substituidas por pseudorotaciones vectoriales como se muestra en la figura 2.4.

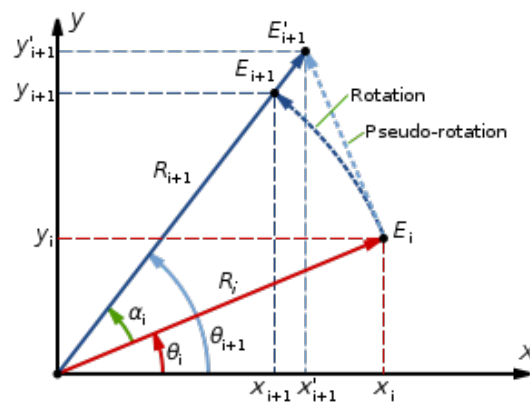


Figura 2.4: Pseudorotación de un vector bidimensional. [3]

En una rotación real no se cambia la magnitud del vector V una vez realizada la rotación, pero en una pseudorotación esto no aplica, ya que incrementa su magnitud en

$$V'' = V\sqrt{1 + \tan^2\theta} \quad (2.22)$$

La rotación puede ser descompuesta en una sumatoria de rotaciones más pequeñas. Suponiendo inicialmente que $x = x_0$, $y = y_0$ y $z = z_0$, después de realizadas n iteraciones para una rotación real, se obtiene que

$$x'_n = x\cos\theta\left(\sum_{i=0}^{n-1}\theta_i\right) - y\sin\theta\left(\sum_{i=0}^{n-1}\theta_i\right) \quad (2.23)$$

$$y'_n = y\cos\theta\left(\sum_{i=0}^{n-1}\theta_i\right) + x\sin\theta\left(\sum_{i=0}^{n-1}\theta_i\right) \quad (2.24)$$

$$z'_n = z - \left(\sum_{i=0}^{n-1}\theta_i\right) \quad (2.25)$$

Para una pseudorotación se tiene que

$$x''_n = \left(x\cos\theta\left(\sum_{i=0}^{n-1}\theta_i\right) - y\sin\theta\left(\sum_{i=0}^{n-1}\theta_i\right)\right) \prod_{i=0}^{n-1}\sqrt{1 + \tan^2\theta_i} \quad (2.26)$$

$$y''_n = \left(y\cos\theta\left(\sum_{i=0}^{n-1}\theta_i\right) + x\sin\theta\left(\sum_{i=0}^{n-1}\theta_i\right)\right) \prod_{i=0}^{n-1}\sqrt{1 + \tan^2\theta_i} \quad (2.27)$$

$$z''_n = z - \left(\sum_{i=0}^{n-1}\theta_i\right) \quad \text{con} \quad \left(\sum_{i=0}^{n-1}\theta_i\right) = \theta \quad (2.28)$$

Las ecuaciones (2.25) y (2.28) son referidas como el parámetro *acumulador angular* y contienen la rotación parcial realizada.

Para poder hacer un cambio de la operación multiplicación por una de desplazamiento se restringe la rotación de manera que $\tan\theta = \pm 2^{-i}$, $i \in N$, haciendo que se pueda reducir factores como área y tiempo a nivel computacional con esta consideración. Los cálculos se reducen a realizar rotaciones sucesivas de valores muy pequeños en cada iteración, pero únicamente se debe decidir en qué sentido se debe de rotar. Al sustituir la aproximación en la fórmula general de rotación de vectores (2.19) y sabiendo que una de las identidades del coseno es $\cos(\theta) = \cos(-\theta)$, la anterior ecuación se puede reescribir de manera tal que

$$\begin{aligned} x_{i+1} &= K_i(x_i - y_i d_i 2^{-i}) \\ y_{i+1} &= K_i(y_i + x_i d_i 2^{-i}) \end{aligned} \quad (2.29)$$

donde $K_i = \frac{1}{\sqrt{1+2^{-2i}}}$ y $d_i = \pm 1$ depende del sentido de rotación [32].

K_i es definido por las pseudorotaciones, ya que estas aumentan el tamaño del vector, por lo que este valor se encarga de eliminar un posible escalado. Una vez que K_i se remueve se tiene un algoritmo capaz de realizar cálculos a partir de sumas y desplazamientos. El factor K_i se aplica como constante K_n generalmente al inicio del proceso y el valor de la misma es determinada por la cantidad de iteraciones por realizar, definida como

$$K_n = \lim_{n \rightarrow \infty} \prod_{i=0}^n K_i \cong 0.607253 \quad (2.30)$$

El algoritmo define únicamente el sentido de rotación y cada ángulo tiene una representación adecuada mediante un vector de signo correspondiente a un ángulo de secuencia elemental. Los valores de los ángulos se almacenan en una Look-Up Table(LUT) y mediante esto es posible modificar el acumulador angular, de forma que

$$z_{i+1} = z_i - d_i \arctan(2^{-i}) \quad (2.31)$$

Los métodos de operación del algoritmo pueden variar entre *rotación* o *vectorización*:

- Rotación: el acumulador angular comienza con un valor inicial de un ángulo y se realiza la decisión de sentido de rotación en cada iteración con el fin de minimizar la magnitud del ángulo a cero.
- Vectorización: el ángulo ingresado rota hasta alinearse con el eje X. Para esto se utiliza el signo de la variable y para definir el sentido de rotación.

Las ecuaciones iterativas para la *rotación*, son

$$\begin{aligned} x_{i+1} &= x_i - y_i d_i 2^{-i} \\ y_{i+1} &= y_i + x_i d_i 2^{-i} \\ z_{i+1} &= z_i - d_i e(i) \end{aligned} \quad (2.32)$$

donde, $d_i = \begin{cases} -1, z_i < 0 \\ +1, z_i \geq 0 \end{cases}$

Las ecuaciones iterativas para la *vectorización*, son

Para el modo de *vectorización*, las ecuaciones iterativas son

$$\begin{aligned} x_{i+1} &= x_i - y_i d_i 2^{-i} \\ y_{i+1} &= y_i + x_i d_i 2^{-i} \\ z_{i+1} &= z_i - d_i \arctan(2^{-i}) \end{aligned} \quad (2.33)$$

donde, $d_i = \begin{cases} -1, y_i \geq 0 \\ +1, y_i < 0 \end{cases}$

Una manera general para expresar las ecuaciones 2.32 y 2.33 se resume como:

$$x_{i+1} = x_i - md_i 2^{-i} y_i \quad (2.34)$$

$$y_{i+1} = y_i - d_i 2^{-i} x_i \quad (2.35)$$

$$z_{i+1} = z_i - d_i \chi(i) \quad (2.36)$$

Donde $\chi(i)$ se muestra en la tabla 2.3 según corresponde cada caso.

Tabla 2.2: Sistema de coordenadas unificado (CORDIC): circular, lineal e hiperbólico.
[4]

m	Sistema de coordenadas	Valor de $\chi(i)$
1	Circular	$\tan^{-1}(2^{-i})$
0	Lineal	2^{-i}
-1	Hiperbólico	$\tanh^{-1}(2^{-i})$

2.5.1 Sistema de coordenadas hiperbólico

Para el cálculo de algunas funciones con el algoritmo aumenta la complejidad, de manera que se deben utilizar las siguientes identidades [4]:

$$\tan z = \frac{\sin z}{\cos z} \quad (2.37)$$

$$\tanh z = \frac{\sinh z}{\cosh z} \quad (2.38)$$

$$\exp z = \sinh z + \cosh z \quad (2.39)$$

$$\ln \omega = 2 \cdot \tanh^{-1} \left(\frac{y}{x} \right) \quad (2.40)$$

Donde

$$x = \omega + 1 \quad y = \omega - 1 \quad (2.41)$$

2.5.2 Exponencial utilizando el algoritmo hiperbólico de CORDIC

Para una función e^φ , con el algoritmo de CORDIC hiperbólico, se deben de utilizar las ecuaciones [29]

$$x_{i+1} = x_i + d_i 2^{-i} y_i \quad (2.42)$$

$$y_{i+1} = y_i + d_i 2^{-i} x_i \quad (2.43)$$

$$z_{i+1} = z_i - d_i \tanh^{-1}(2^{-i}) \quad (2.44)$$

Los valores finales de x y y , representan el resultado de $\cosh(\varphi)$ y $\sinh(\varphi)$ respectivamente y que al ser sumados generan el valor del exponencial (vease la ecuación 2.39). Aquí, i es el índice de cada iteración ($i=1,2,3,..N$) y se debe tomar en cuenta la repetición de las iteraciones $4, 13, 40, \dots k, 3k+1$ para garantizar la convergencia dando una mejor precisión en el cálculo. El rango básico de convergencia del algoritmo se define mediante las siguientes ecuaciones [33] [34]

Modo Rotación:

$$|z_0| = \chi_N + \left(\sum_{i=1}^N \chi_i \right) \quad (2.45)$$

$$|z_0| = \tanh^{-1}(2^{-N}) + \sum_{i=1}^N \tanh^{-1}(2^{-N}) \quad (2.46)$$

$$|z_0|_{max} \approx 1.182 \quad \text{para } N \rightarrow \infty \quad (2.47)$$

Este es el dominio impuesto por el argumento de la función hiperbólica en el modo de rotación.

Modo Vectorización:

$$\left| \tanh^{-1}\left(\frac{y_0}{x_0}\right) \right| \leq \chi_N + \left(\sum_{i=1}^N \chi_i \right) \quad (2.48)$$

$$\left| \tanh^{-1}\left(\frac{y_0}{x_0}\right) \right| \leq 1.182 \quad \text{para } N \rightarrow \infty \quad (2.49)$$

$$\left| \frac{y_0}{x_0} \right| \approx 0.80694 \quad \text{para } N \rightarrow \infty \quad (2.50)$$

Este es el dominio impuesto por el argumento de la función hiperbólica en el modo de vectorización. El dominio de las funciones \tanh^{-1} es $]-1, +1[$ [35]. Los valores iniciales para calcular la función exponencial se definen como constantes: $x_0 = 1,20753406$, $y_0 = 0$ y $z_0 = \varphi$, con φ como el valor del argumento que se desea calcular; d_i es el signo de z_i . El rango de convergencia para la función exponencial se puede definir como:

$$-1.12642 < \varphi < 1.12642 \quad (2.51)$$

2.5.3 Expansión del rango de convergencia del algoritmo hiperbólico

El rango expresado en las ecuaciones 2.47 y 2.50 es insuficiente para realizar la mayoría de funciones que dependen de este algoritmo. Se propone un esquema para agregar iteraciones al cálculo y este consiste en introducir índices negativos en las mismas, modificando $\chi(i)$, de la forma [36]

$$\chi_i = \tanh^{-1}(1 - 2^{i-2}), \text{ para } i \leq 0 \quad (2.52)$$

Teniendo en cuenta esta consideración, las ecuaciones 2.42, 2.43 y 2.44 se pueden reescribir de manera que se tomen los índices negativos como [35]

Para $i \leq 0$

$$x_{i+1} = x_i + d_i(1 - 2^{i-2})y_i \quad (2.53)$$

$$y_{i+1} = y_i + d_i(1 - 2^{i-2})x_i \quad (2.54)$$

$$z_{i+1} = z_i - d_i \tanh^{-1}(1 - 2^{i-2}) \quad (2.55)$$

Para $i > 0$

$$x_{i+1} = x_i + d_i 2^{-i} y_i \quad (2.56)$$

$$y_{i+1} = y_i + d_i 2^{-i} x_i \quad (2.57)$$

$$z_{i+1} = z_i - d_i \tanh^{-1}(2^{-i}) \quad (2.58)$$

Esto produce la expansión de los valores máximos de z_0 y el arcotangente de la relación $\frac{y_0}{x_0}$ de modo en que se convierte cada uno en (-M: índices negativos):

$$|z_0| \leq \chi_{max} \quad (2.59)$$

$$|\tanh^{-1}\left(\frac{y_0}{x_0}\right)| \leq \chi_{max} \quad (2.60)$$

$$\chi_{max} = \sum_{i=-M}^0 \tanh^{-1}(1 - 2^{i-2}) + [\tanh^{-1}(2^{-N}) + \sum_{i=1}^N \tanh^{-1}(2^{-i})] \quad (2.61)$$

Tabla 2.3: Valores de χ_{max} con el algoritmo hiperbólico extendido.

M	χ_{max}
0	2.09113
1	3.44515
2	5.16215
3	7.23371
4	9.65581
5	12.42644
6	15.54462
7	19.00987
8	22.82194
9	26.98070
10	31.48609

Por ejemplo, si M=1 (dos iteraciones negativas), el rango de χ_{max} para las funciones cosh y sinh es expandido a [-3.44515, 3.44515] para la función exponencial, es decir $[e^{-3.44515}, e^{3.44515}]$, obteniendo valores de [0.03189, 31.347989]. La función tanh con un M mayor hace que el dominio se expanda hasta prácticamente]-1,+1[, el cuál es el dominio completo de esta función.

Capítulo 3

Modelo teórico del algoritmo estimador de parámetros

En este capítulo se describe el desarrollo de un algoritmo estimador que tiene como finalidad determinar los parámetros necesarios para calcular el punto de máxima potencia en un panel fotovoltaico. También se presenta la dinámica de los parámetros en la fase plana o plano de fase como parte de la solución matemática en un plano bidimensional (θ_1, θ_2) . El estimador está fundamentado en el uso del mismo para una gran variedad de tecnologías de paneles presentes en el mercado, pero ciertamente no es válido para todas ellas.

Este modelo parte de las investigaciones realizadas en [19] y [23], que demuestran el funcionamiento del estimador para un diagrama de un cargador de baterías (PVG Battery Charger) y un arreglo de paneles (PVG Full Bridge Inverter), respectivamente. Para la correspondiente verificación del sistema se genera un modelo en alto nivel, específicamente en Python, donde se describe el comportamiento esperado para un panel específico que a su vez simula la operación real bajo condiciones estándar de prueba.

3.1 Estimación de parámetros

La estimación, capaz de aproximar los parámetros dependientes de la temperatura y los elementos característicos de un panel fotovoltaico, es necesaria para predecir el punto de máxima potencia sin hacer uso de algoritmos heurísticos MPPT (maximum power point tracker) [37] [38], los cuales requieren de condiciones de irradiancia y temperatura específicas lo que los hace más costosos y complejos. Mediante las siguientes ecuaciones es posible crear un método alternativo necesario para acelerar la búsqueda del punto máximo de potencia de un PV, se parte de las ecuaciones (2.10) a (2.16), las cuales pueden ser reducidas a

$$y = \ln(I_g - i_{pv}) \quad z = v_{pv} \quad (3.1)$$

$$\theta_1 = \alpha(T) \quad \theta_2 = \ln(I_s) \quad (3.2)$$

Estas ecuaciones pueden sintetizarse en una única ecuación dada por:

$$y = \theta_1 z + \theta_2 \quad (3.3)$$

Se parte de los siguientes supuestos para la obtención de los parámetros θ_1 y θ_2 :

- v_{pv} e i_{pv} son medidas y provienen del panel fotovoltaico.
- I_g (corriente de corto circuito) y S (irradiancia solar) son medidas.
- T (temperatura del medio) no se mide.
- I_s y α son desconocidos

Debido a que se necesita trabajar con datos lo más cercanos posibles a un modelo de panel fotovoltaico previamente escogido (Kyocera Solar KC65T [5]), es necesario generar un modelo en un lenguaje de alto nivel que represente las curvas I-V (corriente - tensión) y P-V (potencia - tensión) de manera adecuada, realizando una aproximación con los siguientes parámetros provistos por el fabricante según la tabla 3.2.

Tabla 3.1: Especificaciones del comportamiento eléctrico del panel fotovoltaico KC65T, bajo condiciones estándar de prueba (STC) [5].

Parámetro	Valor	Unidad
Potencia Máxima (P_{max})	65.25	W
Voltaje a la Pot.Máxima (V_{mpp})	17.4	V
Corriente a la Pot.Máxima (I_{mpp})	3.75	A
Voltaje de circuito abierto (V_{oc})	21.7	V
Corriente de corto circuito (I_{sc} o I_g)	3.99	A
Máximo voltaje del sistema (V_{max})	600	V

Se sabe que para un PV, el voltaje de circuito abierto V_{oc} representa el máximo voltaje que puede entregar una celda solar, cuando la corriente es cero [16]. El V_{oc} corresponde a la cantidad de polarización directa que permite polarizar la junta p-n. La ecuación típica del voltaje de circuito abierto está dada por:

$$V_{oc} = \frac{nkT}{q} \ln\left(\frac{I_g}{I_s} + 1\right) \quad (3.4)$$

La relación de V_{oc} en la ecuación (3.4), puede aproximarse a la ecuación (3.5) con $I_g \gg I_s$. Sin embargo, se necesita de al menos una aproximación inicial de uno de los valores a estimar, debido a que se tiene una única ecuación lineal con dos incógnitas. A razón de

ello, se crea una reducción a la mejor zona para obtener las curvas en el punto máximo de potencia, en un valor aproximado de $\alpha=0.625$.

$$V_{oc} = \frac{1}{\alpha} \ln\left(\frac{I_g}{I_s}\right) \quad (3.5)$$

Mediante el valor aproximado para α se puede determinar el otro parámetro desconocido, I_s , con un simple despeje

$$21.7 = \frac{1}{0.625} \ln\left(\frac{3.99}{I_s}\right) \quad (3.6)$$

$$I_s = 5,1387085 \cdot 10^{-6} \quad (3.7)$$

Con el valor obtenido en la ecuación (3.7), el valor de α y las siguientes relaciones, es posible graficar las curvas de la figura 3.1 las cuales representan que los valores aproximados de θ_1 y θ_2 son adecuados para obtener el máximo punto de potencia del panel.

- $i_{pv} = I_g - I_s * e^{V_{oc}-V_{test}}$ con restricción: si $V_{oc} - V_{test} > \left(\frac{\ln(I_g)-\ln(I_s)}{\alpha}\right)$, $i_{pv} = 0$, ya que en ningún momento la tensión $V_{oc} - V_{test}$ puede ser mayor que la tensión de circuito abierto v_{oc} .
- $V_{test} =$ muestras continuas que forman una función exponencial desde $e^{\ln(0.001)}$ hasta $e^{\ln(V_{pv})}$.
- $P_{pv} = i_{pv} * (V_{oc} - V_{test})$ con $(V_{oc} - V_{test}) = v_{pv}$.

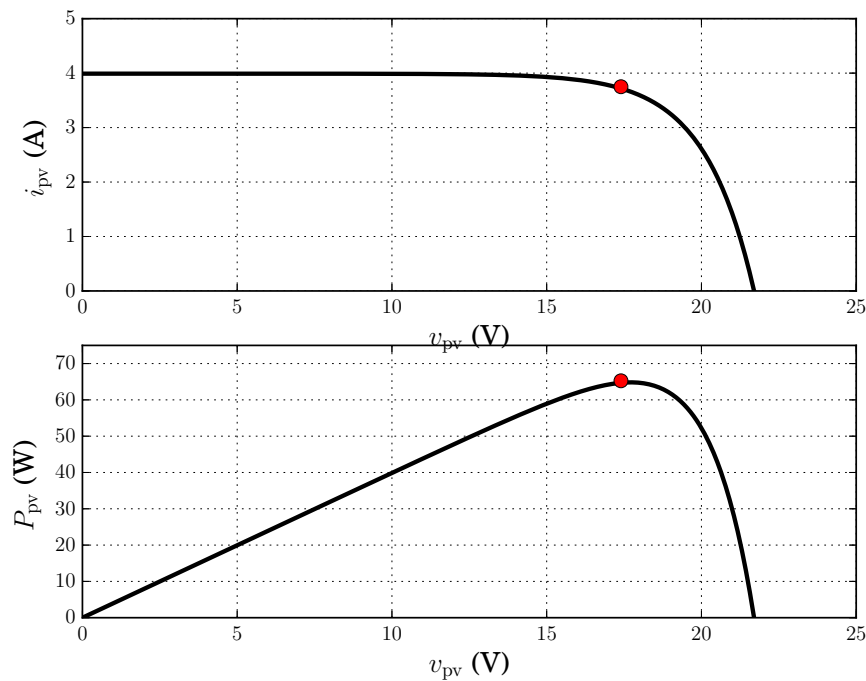


Figura 3.1: Gráficas $i-v$ y $P-v$ obtenidas con el modelo para el panel Kyocera Solar KC65T con parámetros $\theta_1 = 0.625$ y $\theta_2 = \ln(I_s) = -12.178$.

Se puede entonces considerar la reescritura de la ecuación (3.3) en

$$y = \theta^T \Phi \quad (3.8)$$

Donde $\Phi = [V_{pv} \ 1]^T$, $y = \ln(I_g - I_{pv})$ y $\theta = [\theta_1 \ \theta_2]^T$ representa los parámetros por estimar $\theta_1 = \alpha$ y $\theta_2 = \ln(I_s)$.

En [19] [23], se definen las siguientes consideraciones y del mismo modo se toman como base para el desarrollo del modelo del estimador:

- $\hat{\theta} = [\hat{\theta}_1 \ \hat{\theta}_2]^T$ definición de los valores una vez estimados.
- Estimador por implementar (Γ definida como una matriz de tamaño 2x2)

$$\dot{\hat{\theta}} = \Gamma \Phi (y - \Phi^T \hat{\theta}) \quad (3.9)$$

- Si se define el parámetro de error como $\tilde{\theta} = \hat{\theta} - \theta$, se puede reescribir (3.9) de la forma

$$\dot{\tilde{\theta}} = -\Gamma \Phi \Phi^T \tilde{\theta} \quad (3.10)$$

3.2 Dinámica en la fase plana

Una vez definido el esquema sugerido en la ecuación (3.9), se define la matriz Γ como:

$$\Gamma = \begin{bmatrix} \gamma_{11} & \gamma_{12} \\ \gamma_{21} & \gamma_{22} \end{bmatrix}$$

Sus componentes γ_{ii} serán escogidos de manera que sean positivos. Debido a que el ajuste de los mismos es crítico en términos de comportamiento y rapidez de convergencia del estimador, se opta por crear una reducción a únicamente dos valores, los cuáles serán los elementos de la diagonal principal γ_{11} y γ_{22} , el par de valores restantes se establecen en un valor de cero, es decir $\gamma_{12} = \gamma_{21} = 0$.

Para representar de una mejor manera las ecuaciones que se introducen al modelo de alto nivel, se sintetiza la ecuación (3.9) para $\hat{\theta}_1$ y $\hat{\theta}_2$, con $\hat{y} = \hat{\theta}_1 z + \hat{\theta}_2$ de la siguiente forma:

$$\dot{\hat{\theta}}_1 = (\gamma_{11} z + \gamma_{12})(y - \hat{y}) \quad (3.11)$$

$$\dot{\hat{\theta}}_2 = (\gamma_{21} z + \gamma_{22})(y - \hat{y}) \quad (3.12)$$

Una vez aplicada la reducción:

$$\dot{\hat{\theta}}_1 = (\gamma_{11}z)(y - \hat{y}) \quad (3.13)$$

$$\dot{\hat{\theta}}_2 = (\gamma_{22})(y - \hat{y}) \quad (3.14)$$

Adicionalmente para poder graficar la respuesta del estimador se debe de considerar:

- $v_{pv} = \bar{v}_{pv} + 0.3\bar{v}_{pv} \sin(200 * \pi * t)$ (tensión de excitación permanente en estado estable).
- $y = \alpha v_{pv} + \ln I_s$ de la ecuación 2.16.
- Reescribir (3.13) y (3.14) como

$$\dot{\hat{\theta}}_1 = (\gamma_{11}v_{pv})(\alpha v_{pv} + \ln I_s - \hat{\theta}_1 v_{pv} - \hat{\theta}_2) \quad (3.15)$$

$$\dot{\hat{\theta}}_2 = (\gamma_{22})(\alpha v_{pv} + \ln I_s - \hat{\theta}_1 v_{pv} - \hat{\theta}_2) \quad (3.16)$$

Conociendo todas las ecuaciones necesarias para el modelo a simular, es posible generar la dinámica de $\hat{\theta}_1$ y $\hat{\theta}_2$ cuando no se ha alcanzado la isocline de crecimiento cero. Este método se emplea debido a que se trata de aproximar un sistema no lineal como uno lineal utilizando una aproximación gráfica ([39], [40]) alrededor de un punto de equilibrio dado. El comportamiento del sistema se muestra en la figura 3.2.

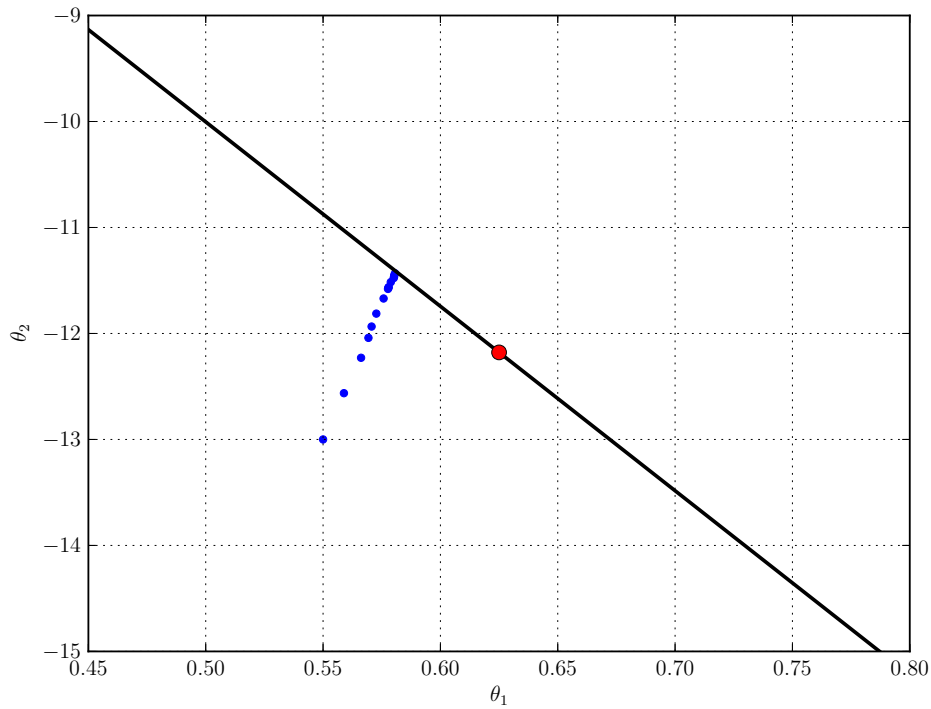


Figura 3.2: Evolución de parámetros $\hat{\theta}_1$ y $\hat{\theta}_2$ en el plano bidimensional antes de alcanzar la trayectoria de la isocline de crecimiento cero.

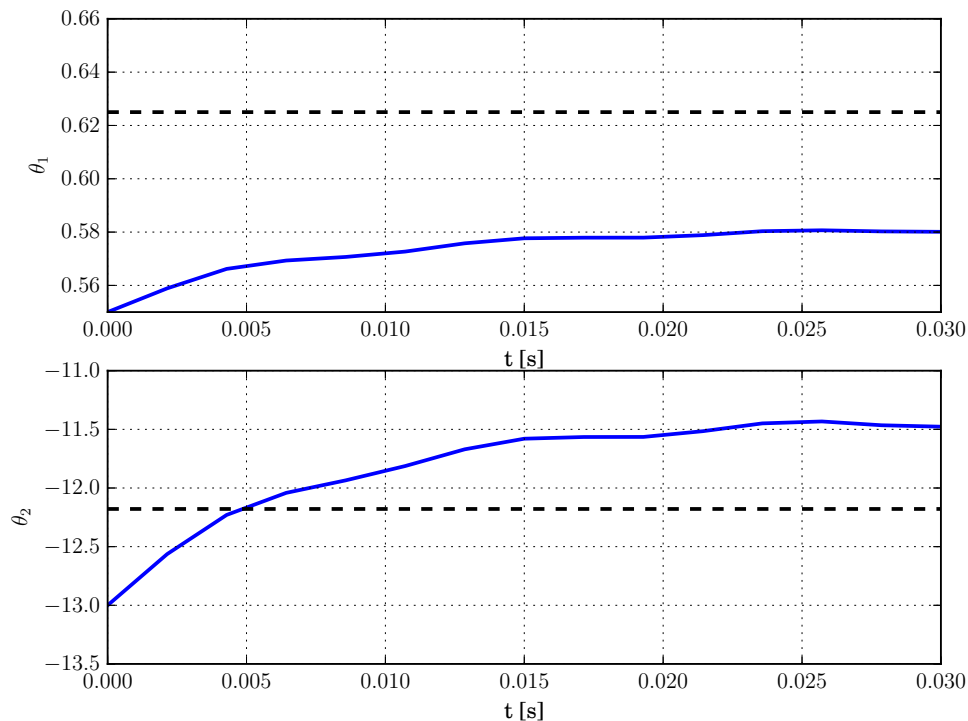


Figura 3.3: Dinámica de los parámetros $\hat{\theta}_1$ y $\hat{\theta}_2$ mostrando su evolución en el tiempo. Tiempo de simulación= 0.030 segundos.

La figura 3.2 muestra que para un tiempo muy pequeño de simulación el estimador no es capaz de alcanzar el equilibrio establecido con el punto rojo en la gráfica, ni el comienzo del *nullcline*. En la figura 3.3 se aprecia con mayor detalle la diferencia que existe con respecto al punto de equilibrio deseado para θ_1 y θ_2 . A medida de que se incremente el tiempo de simulación en el modelo, este mostrará cómo se genera el comportamiento de los parámetros en el plano de fases. La tabla 3.2 resume los valores utilizados.

Tabla 3.2: Parámetros considerados para la simulación del estimador para un tiempo reducido

Parámetro	Valor
γ_{11}	0.1
γ_{22}	100
$\hat{\theta}_1(t=0)$	0.55
$\hat{\theta}_2(t=0)$	-13
Número de puntos	15
Tiempo de simulaciones [s]	0.03
Tolerancia absoluta y relativa	$5 \cdot 10^{-8}$

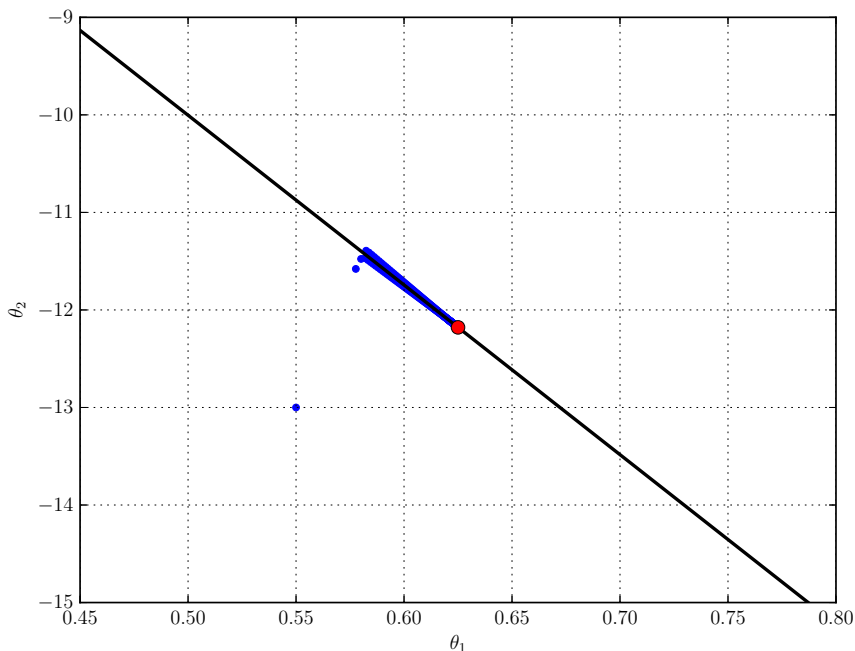


Figura 3.4: Evolución de parámetros $\hat{\theta}_1$ y $\hat{\theta}_2$ en el plano bidimensional una vez alcanzado el punto de equilibrio planteado.

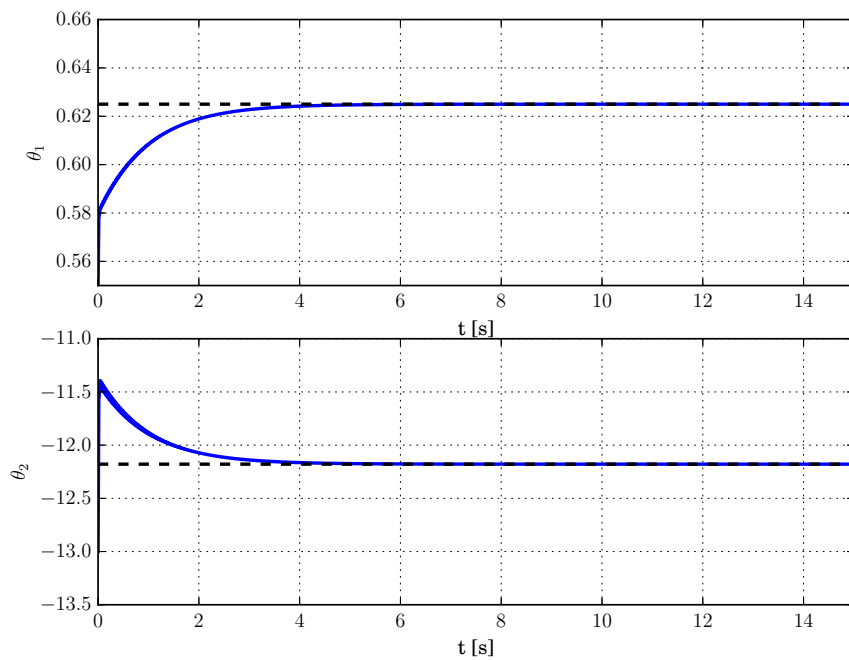


Figura 3.5: Dinámica de los parámetros $\hat{\theta}_1$ y $\hat{\theta}_2$ mostrando su evolución en el tiempo. Tiempo de simulación= 15 segundos.

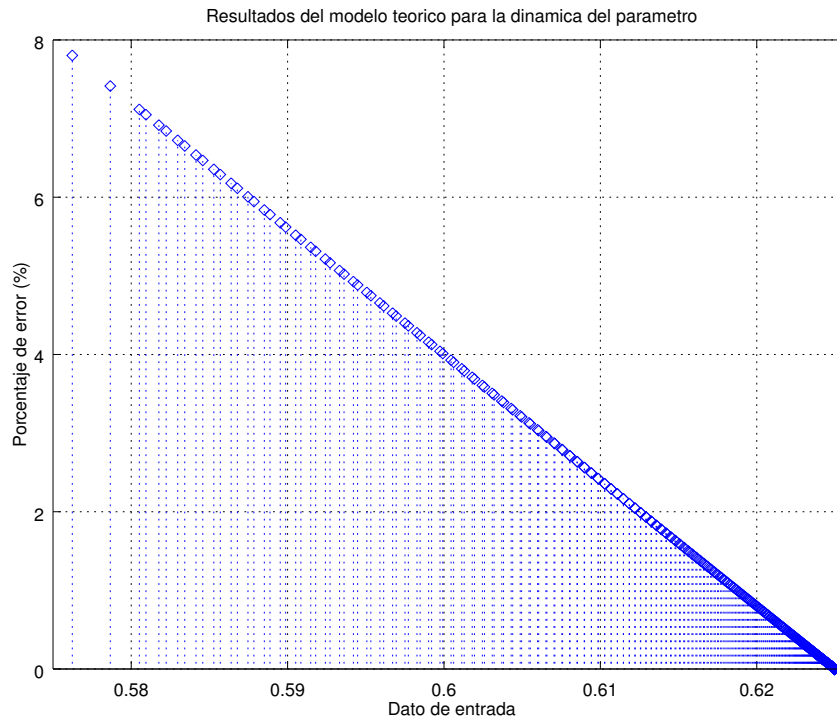
Tabla 3.3: Parámetros considerados para la simulación del estimador para un tiempo extenso

Parámetro	Valor
γ_{11}	0.1
γ_{22}	100
$\hat{\theta}_1(t=0)$	0.55
$\hat{\theta}_2(t=0)$	-13
Número de puntos	1000
Tiempo de simulación(s)	15
Tolerancia absoluta y relativa	$5 \cdot 10^{-8}$

La tabla 3.3 muestra que incrementando el tiempo de simulación y el número de puntos utilizados se puede representar la dinámica de manera clara (ver figuras 3.4 y 3.5), tanto en la fase plana como en la evolución del tiempo una vez que la ecuación (2.17) se mantiene constante.

3.3 Resultados del modelo teórico

En las figuras 3.6 y 3.7 se muestra el porcentaje de error para $\hat{\theta}_1$ y $\hat{\theta}_2$, en el que se observa que este supera el 5% para ambos casos cuando los valores iniciales se encuentran distantes de $\alpha = 0.625$ y $\ln I_s = -12.178708$ respectivamente.

**Figura 3.6:** Porcentaje de error obtenido para la convergencia de $\theta_1 = \alpha$.

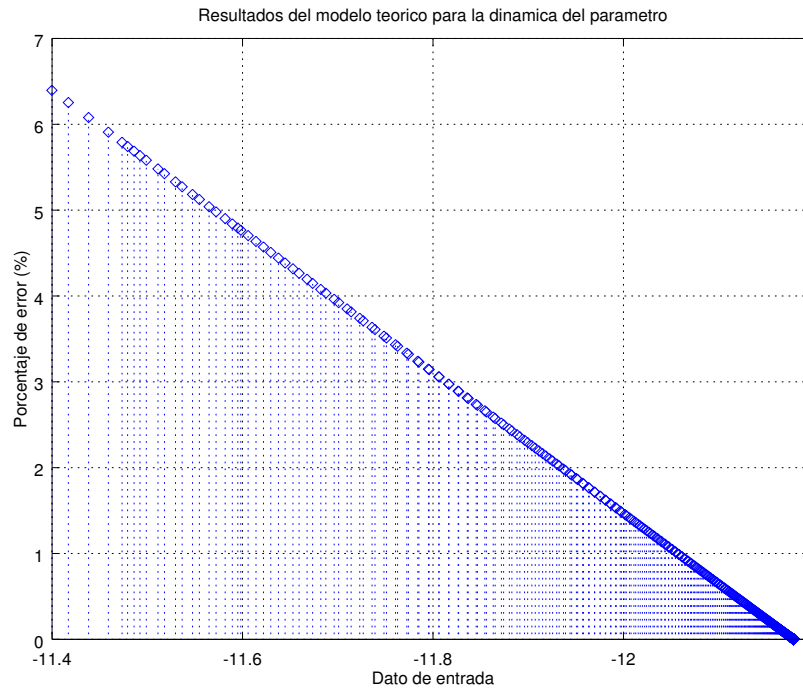


Figura 3.7: Porcentaje de error obtenido para la convergencia de $\theta_2 = \ln(I_s)$.

El error decrece linealmente a medida que el valor converge hasta alcanzar un valor de 0%, este puede ser aún menor para valores iniciales que se encuentren más cercanos a los deseados. Para efectos de demostrar el comportamiento del modelo se tomaron estos valores arbitrariamente. Es importante aclarar que se debe de realizar un análisis del modelo teórico y una modificación de γ_{11} y γ_{22} con miras a una implementación en un dispositivo programable. Lo anterior se debe, a que estos valores pueden permitir acelerar la convergencia notablemente haciendo que el tiempo de ejecución sea muy bajo.

El tiempo de estimación mostrado en la figura 3.5 es de aproximadamente 5 segundos, lo que se considera aceptable debido a que los parámetros desconocidos sólo cambiarán estrictamente cuando la temperatura varíe en el PV.

Capítulo 4

Sistema de conversión coma fija a coma flotante y desnormalización

En este capítulo se presenta la creación de un sistema capaz de convertir datos representados en formato coma fija al estándar IEEE 754. El estimador de parámetros se realizará en coma fija ya que permitirá obtener los cálculos necesarios de una forma rápida y utilizando menos recursos comparado con una implementación en coma flotante.

4.1 Diseño del sistema de conversión, desnormalización y control de datos

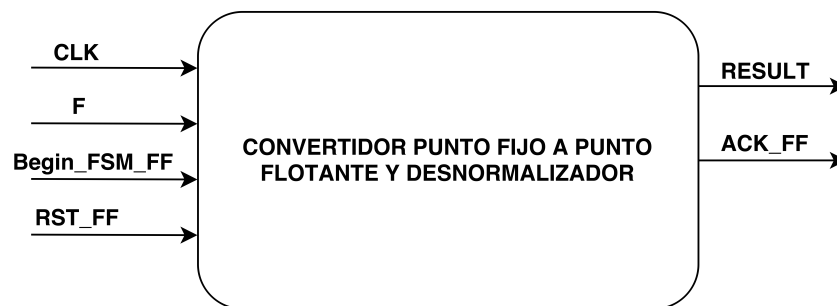


Figura 4.1: Diagrama general de entradas y salidas para el convertidor coma fija a coma flotante y desnormalizador.

Se puede apreciar en la figura 4.1 el diagrama general de la unidad a desarrollar en la que se denotan debidamente las entradas y salidas para la correspondiente conversión y desnormalización. Las señales de entrada en su respectivo orden están compuestas por: CLK, F , Begin_FSM_FF , RST_FF y las dos salidas presentes: ACK_FF y RESULT.

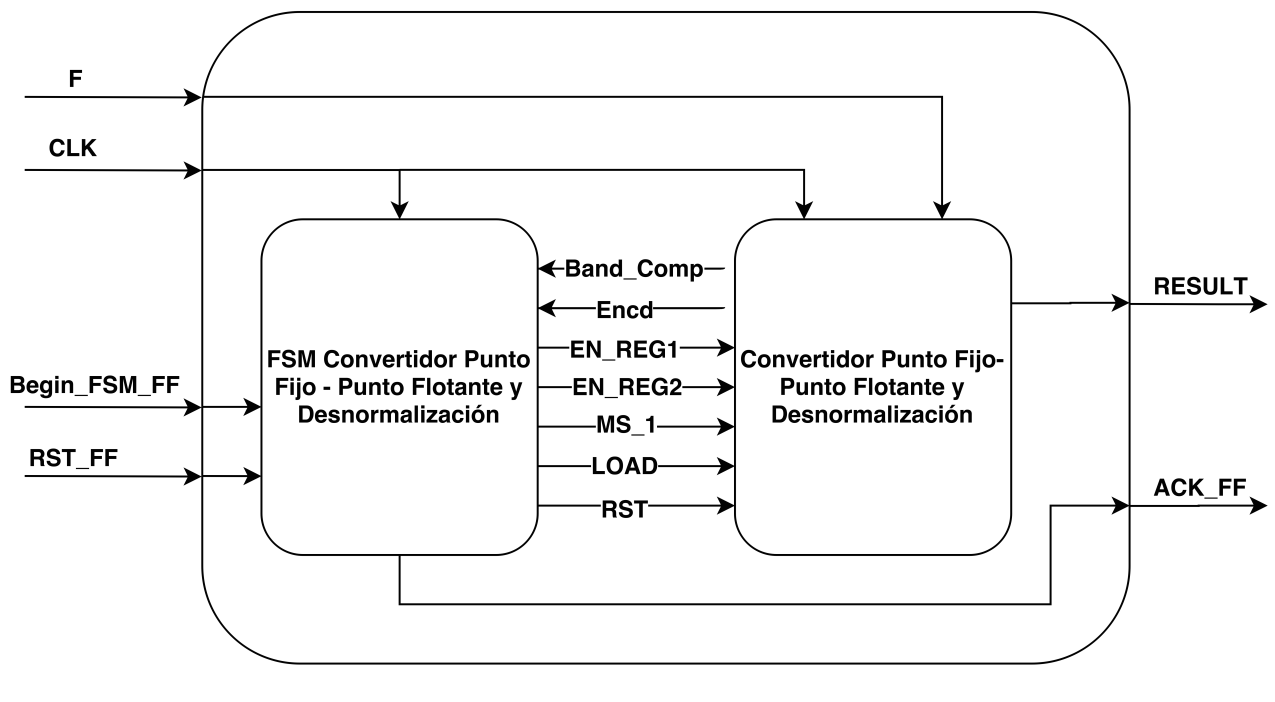


Figura 4.2: Sistema de conversión, desnormalización y control utilizando una máquina de estados.

En la figura 4.2 se puede observar claramente como la unidad posee dos bloques principales.

- *Convertidor-Desnormalizador*: se encarga de la conversión del formato coma fija a coma flotante y de la desnormalización de los datos que entran en la unidad.
- *Máquina de estados (FSM)*: genera las señales necesarias para que el Convertidor-Desnormalizador pueda realizar las funciones anteriormente citadas.

Señales de datos:

- *F*: dato de entrada en formato coma fija.
- *RESULT*: dato de salida en representación *IEEE 754* (precisión simple, 32 bits).

Señales de control:

- *CLK*: reloj del sistema. Coordina el funcionamiento de múltiples circuitos.
- *Begin_FSM_FF*: señal dedicada a iniciar la acción de la máquina de estados e iniciar la función del bloque Convertidor-Desnormalizador.
- *RST_FF*: encargada de reestablecer los valores iniciales de la unidad, en este caso detener la operación de la unidad.
- *ACK_FF*: bandera generada cuando la unidad ha procesado un dato y ha aplicado la correcta función de la misma.

4.2 Convertidor coma fija - coma flotante y desnormalizador

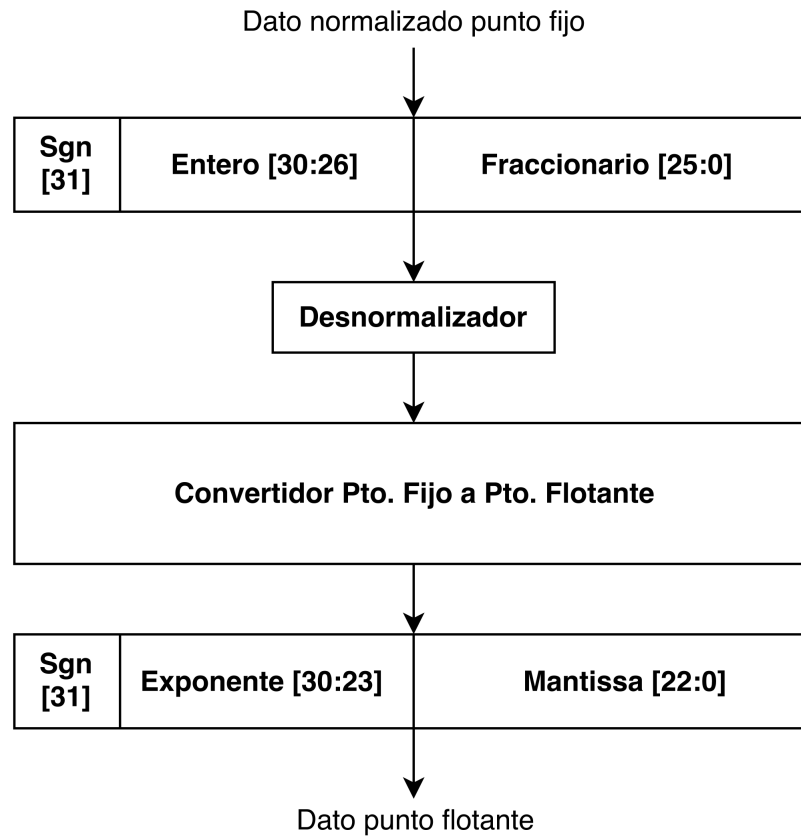


Figura 4.3: Diagrama general del sistema de conversión de coma fija a coma flotante y desnormalización de parámetros $\alpha - \ln(I_s)$ (parámetros dependientes de la irradiancia y temperatura).

La figura 4.3 muestra el procedimiento a realizar con la unidad para obtener un resultado en coma flotante, a partir de un dato coma fija que posee una previa normalización [11].

Al número en formato coma fija, 32 bits, con representación 1 bit signo, 5 magnitud y 26 fraccional, se le aplica una desnormalización y su conversión respectiva. Dicho dato provendrá del algoritmo estimador de parámetros y representará respectivamente a α y $\ln I_s$ que son dos parámetros desconocidos, constantemente variantes y dependientes de la temperatura e irradiancia del panel fotovoltaico.

Se ajusta el dato con 26 bits en la parte fraccional para obtener una mejor precisión en los cálculos y establecer una relación uno a uno, para que el número convertido no sea diferente al de entrada debido a que si se varía la posición de la coma fija, se obtienen resultados dispares.

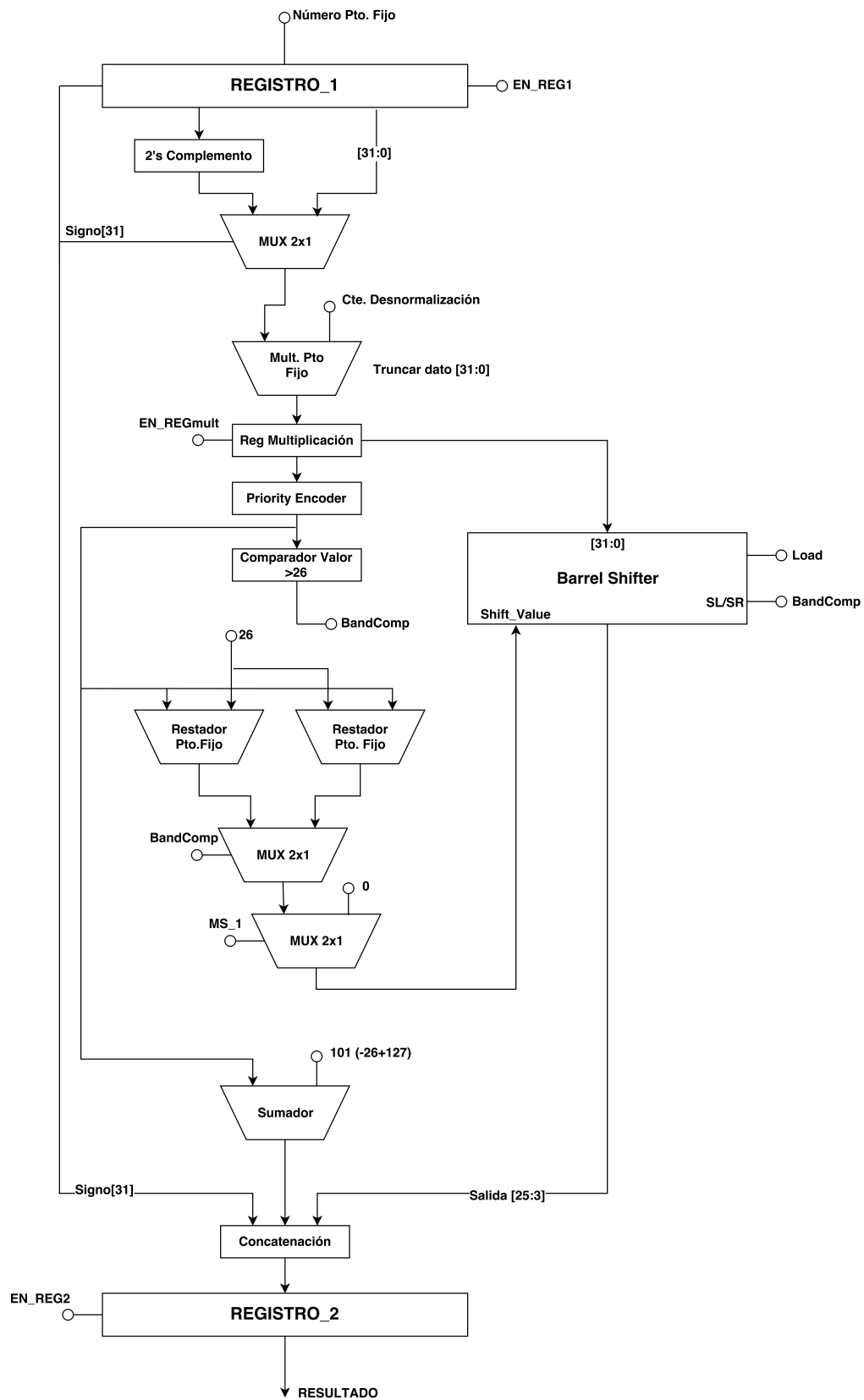


Figura 4.4: Circuito de conversión coma fija a coma flotante y desnormalización de parámetros α y $\ln(I_s)$, con un dato de entrada en coma fija, posteriormente desnormalizado y salida en coma flotante.

En la figura 4.4, se muestran los diferentes bloques que componen el sistema de conversión y desnormalización, inicialmente para el diseño de la misma se investigó cómo realizar dicho cálculo.

En [41] se da un acercamiento inicial sobre cómo realizar dicha operación pero este no contempla una posible desnormalización, únicamente determina la conversión de fijo a flotante.

La siguiente serie de pasos permite realizar el cómputo:

1. El valor del signo influye en el valor final, esto debido a que si el mismo es 1 (negativo) se debe de calcular el complemento a dos del dato punto fijo; por el contrario, si el signo es cero la secuencia se toma en su forma original, ya que es de valor positivo.
2. Una vez que se determina si es positivo o negativo, se desnormaliza. Esta desnormalización consiste en desplazar la ubicación de la coma fija hasta alcanzar el uno más significativo en la secuencia binaria.
3. La nueva secuencia binaria a la derecha de la nueva posición del punto fijo, se denomina mantisa o significando, la cual representa la fracción o entero y forma parte de un cierto número dado en punto flotante.
4. La cantidad de desplazamientos a la izquierda representa el nuevo número que tendrá el exponente de la desnormalización; por ejemplo, si en un número dado en binario se requieren diez desplazamientos a la izquierda el exponente será 2 a la 10 (2^{10}). De este desplazamiento se encarga el desplazador de barril (*barrel shifter*).
5. El exponente del número en punto flotante será la suma de los desplazamientos más un valor comúnmente conocido como bias y que cambia según la precisión del punto flotante que se esté calculando. En este caso si se trabaja en precisión simple (32 bits) el valor del bias será 127 o en precisión doble (64 bits) el correspondiente valor será 1023.

El cálculo procede con el paso (1). Un mux 2x1 se encarga de seleccionar el dato y continuar su ejecución hacia el paso (2). Una vez desnormalizado, se necesita conocer el valor de desplazamiento, debido a que el valor truncado en la multiplicación del número original con la constante de desnormalización desplaza la coma (coma original + coma constante = 30 + 26). Esta se encontrará en el bit 26 (se trunca el dato de 64 a 32 bits, de la posición 62 a la 30), por lo que si el primer uno se encuentra en 26 no existirá desplazamiento (2^0).

Si el valor es mayor a 26 se realiza la operación $valor - 26$, en caso contrario $26 - valor$, lo que determina el correspondiente movimiento. Se procede con los pasos (3) y (4), por último el paso (5) donde se calcula el exponente y se termina de concatenar el resultado en punto flotante con el respectivo signo y mantissa o significando (signo, exponente, mantissa).

Para la ejecución de la desnormalización el dato se multiplica por el máximo valor. En este caso será el valor inverso dado a la normalización [11]. Sin embargo, este proceso únicamente requiere de un multiplicador punto fijo encargado de reestablecer el rango original de los valores previamente estimados.

Las constantes de normalización y desnormalización tienen una correspondencia directa a las ecuaciones (4.1) y (4.2) respectivamente

$$C_{norm} = \frac{1}{Valor_{MAX}} \quad (4.1)$$

$$C_{desnorm} = Valor_{MAX} \quad (4.2)$$

Generalmente las constantes varían con respecto al panel que se utiliza. En el presente desarrollo se utiliza un panel modelo KC65T (*Kyocera Solar*) [5], por lo que vendrán dadas como siguen:

$$Cv_{norm} = \frac{1}{21.7} = 0,04608294 \quad Cv_{desnorm} = 21,7 \quad (4.3)$$

$$Ci_{norm} = \frac{1}{Ln(3.99)} = 0,72265236 \quad Ci_{desnorm} = Ln(3.99) = 1,38379123 \quad (4.4)$$

Donde:

- Cv_{norm} : constante de normalización de la tensión V_{pv} .
- $Cv_{desnorm}$: constante de desnormalización del parámetro α .
- Ci_{norm} : constante de normalización de la corriente I_{pv} .
- $Ci_{desnorm}$: constante de desnormalización del parámetro $\ln I_s$.

Sin embargo, para efectos de demostrar la validez de los valores obtenidos mediante el modelo del estimador de parámetros se establecen las constantes $Cv_{desnorm}$ y $Ci_{desnorm}$ en un valor de uno (1), de manera que la desnormalización no influya en los valores originales estimados correspondientes a α y $\ln I_s$. Un ejemplo de este comportamiento se reflejará si entra un cierto número X en coma fija, se obtendrá el mismo X pero en coma flotante.

4.3 Control para el convertidor coma fija - coma flotante y desnormalizador

El esquema planteado en la figura 4.4 para la correspondiente conversión y desnormalización planteadas, muestra el uso de una cantidad considerable de bloques combinacionales. Sin embargo, la presencia de operaciones como desplazamientos, señales de control de multiplexores y registros hace que sea necesario implementar un control básico mediante una máquina de estados finitos para realizar cada uno de ellos de manera adecuada.

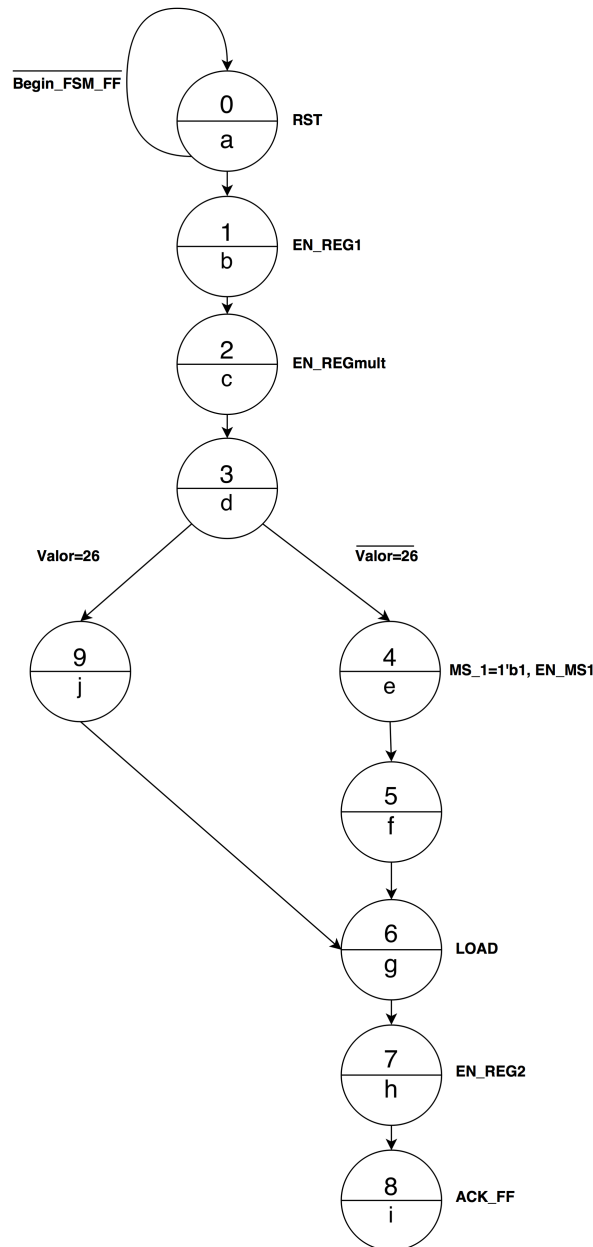


Figura 4.5: Control básico ejecutado por una máquina de estados finitos para la correspondiente conversión-desnormalización.

Se detalla a continuación cada uno de los estados de la máquina:

- Estado (a): espera señal *Begin_FSM_FF* para iniciar la conversión-desnormalización, se hace un reset en registros.
- Estado (b): habilita señal de carga en el *Registro 1*.
- Estado (c): habilita señal de carga en el *Registro Multiplicación*.
- Estado (d): verifica la condición $Valor = 26$, la señal de esta determina posteriores desplazamientos.
- Estado (e): señal de habilitación de mux 2x1.
- Estado (f): estado de espera.
- Estado (g): habilitación del desplazador de barril para carga de valor de desnormalización.
- Estado (h): habilita señal de carga para valor final en punto flotante en el *Registro 2*.
- Estado (i): bandera *ACK_FF* indica finalización del cálculo llevado a cabo por la unidad.
- Estado (j): estado de salto si $Valor = 26$.

4.4 Verificación y resultados del módulo conversión-desnormalización mediante Verilog HDL

Debido a la importancia de comprobar el error de la unidad bajo prueba, se implementa en Verilog una prueba de banco (*testbench*) con al menos 1000 valores provenientes del modelo teórico del estimador tanto para α como también $\ln I_s$. Los datos son extraídos de la figura 3.5 y a partir de los mismos se generan las simulaciones post-implementación.

El esquema a seguir para la comprobación de los datos se presenta en la figura 4.6

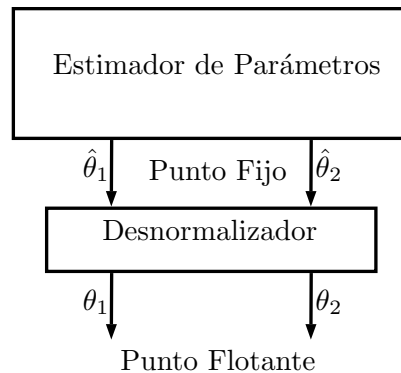


Figura 4.6: Esquema general para la comprobación de datos del módulo conversión-desnormalización.

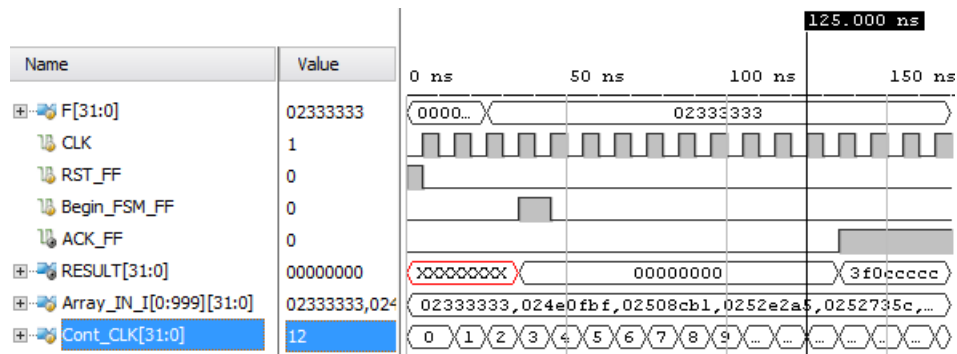


Figura 4.7: Simulación del circuito de conversión y desnormalización con una entrada de 1000 valores provenientes del modelo teórico del estimador.

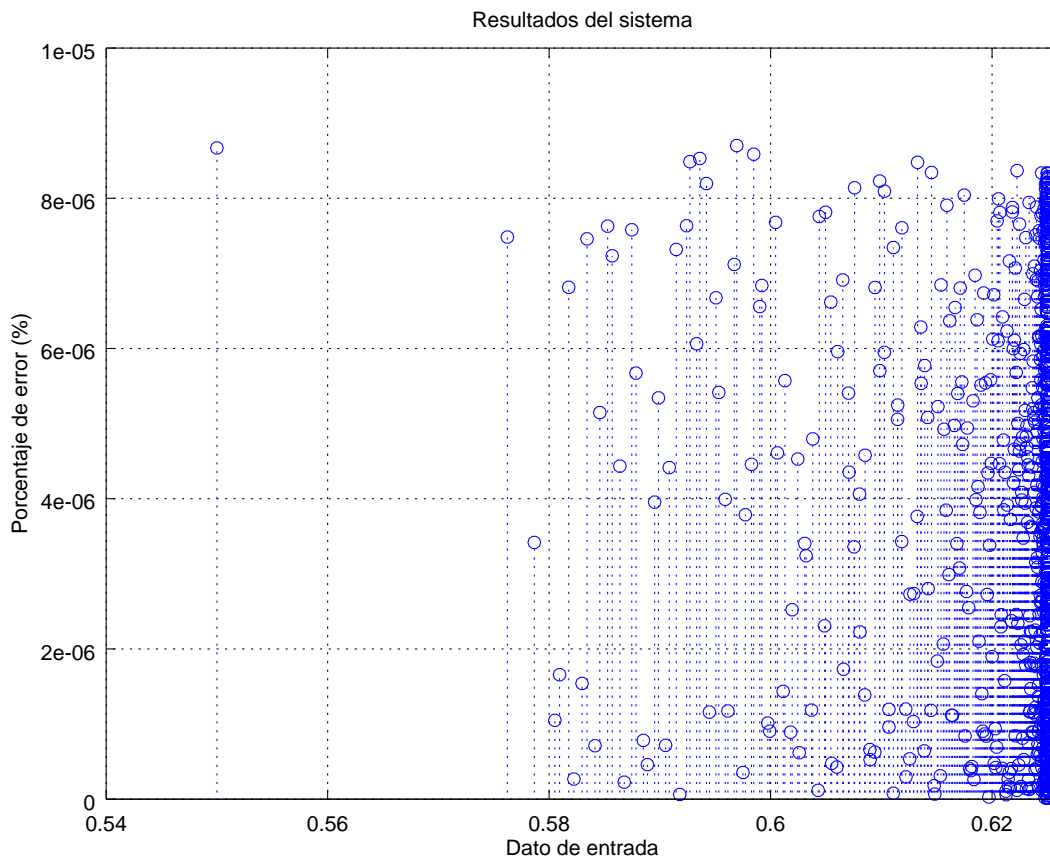


Figura 4.8: Porcentaje de error obtenido para la conversión-desnormalización de α teórica y la simulación post-implementación del circuito.

Se muestra en las figuras 4.8 y 4.9 el porcentaje de error obtenido de la comparación entre los resultados teóricos y experimentales, esta comparación se realiza mediante las simulaciones post-implementación, donde se toman valores de los parámetros normalizados y en coma fija, para posteriormente obtener valores desnormalizados y en coma flotante.

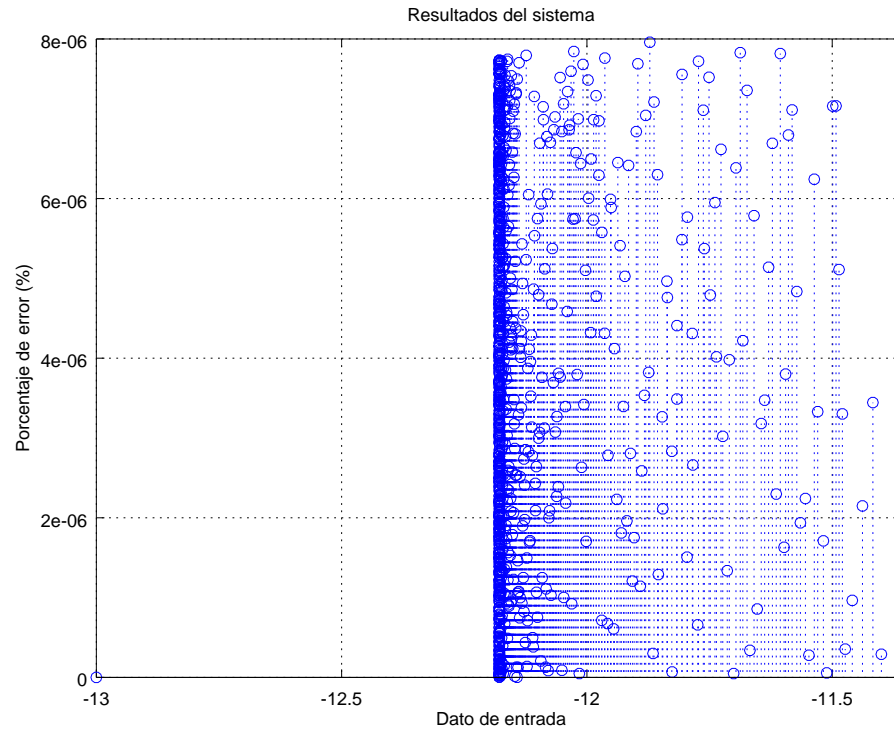


Figura 4.9: Porcentaje de error obtenido para la conversión-desnormalización de $\ln I_s$ teórico y la simulación post-implementación del circuito.

En la tabla 5.4 se presenta un resumen de los resultados obtenidos mediante la prueba de banco.

Tabla 4.1: Resultados experimentales obtenidos de la simulación post-implementación, para los valores de entrada α y $\ln I_s$, presentes en el circuito de conversión-desnormalización.

	α	$\ln I_s$
Error máximo ($\times 10^{-6}$)	8,7021	7,9589
Error promedio ($\times 10^{-6}$)	3,899	3,913
Desviación estándar ($\times 10^{-6}$)	2,7223	2,392
Número de ciclos	9	9
Tiempo de ejecución (ns)	90	90

Se comprueba mediante la figura 4.7 que el cálculo de un valor toma un tiempo exacto de 90 ns, es decir 9 ciclos de reloj para ejecutarse, en base a la prueba con un CLK de 100 MHz, congruente con los valores provistos en la tabla 4.1. La unidad toma un 0.111% sobre la frecuencia de operación establecida como prueba.

Se puede apreciar en la figura 4.8 el error establecido para los parámetros α y $\ln I_s$, en el que se observa que ninguno de los dos sobrepasa un porcentaje de error máximo mayor a $9 \cdot 10^{-6}$, el error promedio no sobrepasa el $4 \cdot 10^{-6}$ y la desviación estándar es $3 \cdot 10^{-6}$.

Ambos resultados demuestran que la unidad planteada puede realizar la conversión-desnormalización con un error ínfimo, lo cual presenta una alta confiabilidad en el diseño implementado en el software suite Vivado de Xilinx. Cómo se mencionó también en la sección 4.2 la prueba es realizada haciendo un set de las constantes de desnormalización a 1, por lo que es importante establecer un valor acorde a este en la conversión-normalización [11], con miras a una prueba completa del sistema cuando este integre todas las unidades.

Capítulo 5

Sistema de deslinealización

En este capítulo se presenta el proceso de desarrollo de una arquitectura ideada para el cálculo de una función exponencial con el fin de deslinealizar el parámetro estimado $\theta_2 = \ln(I_s)$, para el desarrollo de la misma se utiliza la base de una arquitectura segmentada [11].

Para la validación del sistema se realiza de manera inicial un script en un programa de alto nivel. Luego se realiza un análisis para expansión de iteraciones, se procede al diseño de los bloques generales, se genera una descripción del sistema en hardware en el lenguaje Verilog y por último se expone el error obtenido mediante las simulaciones post-implementación realizadas.

5.1 Función exponencial utilizando el algoritmo de CORDIC hiperbólico en software

El parámetro θ_2 posee un comportamiento lineal una vez que alcanza el equilibrio, como se observó en la figura 3.5; no obstante, para una posible manipulación por medio de algoritmos de control, es necesario eliminar la dependencia del logaritmo natural presente esto mediante la función exponencial con la que se podrá obtener el valor de I_s necesario para calcular el punto de potencia máximo del PV.

Esta función exponencial se implementa mediante un algoritmo de cálculo de funciones lineales, hiperbólicas y trigonométricas llamado algoritmo de CORDIC, que permite generar una aproximación cercana a los valores de una función de una manera iterativa y recursiva con un carácter finito.

Originalmente este algoritmo posee una limitación en el rango de convergencia en los valores $-1,12642 < T < 1,12642$ (ecuac. 2.51), con T como el argumento del exponencial (e^T). Se implementa en este caso un método de comprobación en el programa Python para demostrar la convergencia del algoritmo únicamente para valores dentro del rango.

```

import math
def CordicEx(T): #Algoritmo de cordic para resolver un Ex

    #Condiciones iniciales X_ant=0.60376703
    X_ant=0.60376703
    Y_ant=0.60376703
    Z_ant=T

    #LUT Look-up Table
    Bm=[0.54930614, 0.25541281, 0.12565721, 0.06258157, 0.06258157, 0.03126018, 0.03126018, 0.01563009, 0.01563009, 0.00781504, 0.00781504, 0.00390752, 0.00390752, 0.00195376, 0.00195376, 0.00097688, 0.00097688]

    #Cantidad de desplazamientos por iteracion
    Em=[ 2**-1, 2**-2, 2**-3, 2**-4, 2**-4, 2**-5, 2**-6, 2**-7, 2**-7, 2**-8, 2**-9, 2**-10, 2**-10, 2**-11, 2**-11, 2**-12, 2**-12, 2**-13, 2**-13, 2**-14, 2**-14, 2**-15, 2**-15, 2**-16, 2**-16, 2**-17, 2**-17, 2**-18, 2**-18, 2**-19, 2**-19, 2**-20, 2**-20, 2**-21, 2**-21, 2**-22, 2**-22, 2**-23, 2**-23, 2**-24, 2**-24, 2**-25, 2**-25, 2**-26, 2**-26, 2**-27, 2**-27, 2**-28, 2**-28, 2**-29, 2**-29, 2**-30, 2**-30, 2**-31, 2**-31, 2**-32, 2**-32, 2**-33, 2**-33, 2**-34, 2**-34, 2**-35, 2**-35, 2**-36, 2**-36, 2**-37, 2**-37, 2**-38, 2**-38, 2**-39, 2**-39, 2**-40, 2**-40, 2**-41, 2**-41, 2**-42, 2**-42, 2**-43, 2**-43, 2**-44, 2**-44, 2**-45, 2**-45, 2**-46, 2**-46, 2**-47, 2**-47, 2**-48, 2**-48, 2**-49, 2**-49, 2**-50, 2**-50, 2**-51, 2**-51, 2**-52, 2**-52, 2**-53, 2**-53, 2**-54, 2**-54, 2**-55, 2**-55, 2**-56, 2**-56, 2**-57, 2**-57, 2**-58, 2**-58, 2**-59, 2**-59, 2**-60, 2**-60, 2**-61, 2**-61, 2**-62, 2**-62, 2**-63, 2**-63, 2**-64, 2**-64, 2**-65, 2**-65, 2**-66, 2**-66, 2**-67, 2**-67, 2**-68, 2**-68, 2**-69, 2**-69, 2**-70, 2**-70, 2**-71, 2**-71, 2**-72, 2**-72, 2**-73, 2**-73, 2**-74, 2**-74, 2**-75, 2**-75, 2**-76, 2**-76, 2**-77, 2**-77, 2**-78, 2**-78, 2**-79, 2**-79, 2**-80, 2**-80, 2**-81, 2**-81, 2**-82, 2**-82, 2**-83, 2**-83, 2**-84, 2**-84, 2**-85, 2**-85, 2**-86, 2**-86, 2**-87, 2**-87, 2**-88, 2**-88, 2**-89, 2**-89, 2**-90, 2**-90, 2**-91, 2**-91, 2**-92, 2**-92, 2**-93, 2**-93, 2**-94, 2**-94, 2**-95, 2**-95, 2**-96, 2**-96, 2**-97, 2**-97, 2**-98, 2**-98, 2**-99, 2**-99, 2**-100, 2**-100]

    while i<15:

        Dm= Z_ant/(abs(Z_ant)) # Signo de Z
        Z_act= Z_ant + Dm*-1*Bm[i] # Calcula el valor de Z actual
        X_act= X_ant + Em[i]*Dm*Y_ant # Calcula el valor de X actual
        Y_act= Y_ant + Em[i]*Dm*X_ant # Calcula el valor de Y actual
        Z_ant = Z_act # Agrega el valor actual al valor anterior de Z
        Y_ant = Y_act # Agrega el valor actual al valor anterior de Y
        X_ant = X_act # Agrega el valor actual al valor anterior de X
        i = i + 1 # Contador para cada iteracion
        ex=X_act+Y_act # Resultado para cada iteracion
        # X=C=cosh X representa la funcion cosh (coseno hiperbolico)
        # Y=S=sinh Y representa la funcion sinh (seno hiperbolico)
        # Por definicion el resultado de la funcion exponencial se puede calcular
        # como exp = cosh(T)+ sinh(T)

    return ex # Resultado final del valor exponencial

#Probando valor dentro del rango 0.5
>>> CordicEx(0.5)
1.64876172622357

```

Figura 5.1: Algoritmo de CORDIC en Python para el cálculo de la función exponencial.

Se observa en la fig 5.11 la codificación del algoritmo en lenguaje Python, en el que se implementan las ecuaciones (2.42, 2.43, 2.44), para el algoritmo de CORDIC en sistema de coordenadas hiperbólico. Se tienen en cuenta los siguientes aspectos para la verificación del algoritmo en el programa:

- Sustitución de la resta por una suma en el valor actual de Z_{i+1} .
- Modificación del valor previo de $X_{ant} = 1,20753406$ para dividir el término en dos partes iguales e incluir la otra parte en el valor previo de Y_{ant} ; es decir, los nuevos coeficientes serán respectivamente $X_{ant} = Y_{ant} = 0,60376703$.
- Inclusión del signo en la tabla LUT de Z (tabla de búsqueda de valores en binario punto flotante).
- Escalado de los valores de la LUT Z. Éste se multiplica por un valor de 1, para evitar agregar hardware en la arquitectura de cálculo.

Debido a que el rango de convergencia es muy limitado, se considera aplicar una expansión del mismo por medio de la inclusión de iteraciones con índices negativos en el cómputo [35]. Para esto se parte de las investigaciones planteadas en [29] y [33], que sugieren la creación de arquitecturas de bajo costo para lograrlo.

Sin embargo, debido a que el área es un factor importante a considerar en el diseño del sistema, se decide finalmente analizar el comportamiento de los valores binarios fuera del rango, con el fin de encontrar una relación en los exponentes y significandos que permita reducir una cantidad de hardware considerable en comparación con dichas arquitecturas realizando la misma función. Primeramente se debe validar la expansión del rango en Python (ver figura 5.11).

```
import math
def CordicEx(T):

    #Condiciones iniciales
    Z_ant=T
    C_ant=1.8855
    S_ant=1.8855

    #LUT (Valores Modificados, iteraciones negativas)
    Bm=[1.354025101, 0.9729550745, 0.54930614, 0.25541281, 0.1
    i=0

    #Cantidad de desplazamientos por iteracion
    Em=[(1-2**-3), (1-2**-2),2**-1, 2**-2, 2**-3,2**-4,2**-4,2

    while i<15:

        Dm= Z_ant/(abs(Z_ant))
        Z_act= Z_ant + Dm*-1*Bm[i]
        C_act= C_ant + Em[i]*Dm*S_ant
        S_act= S_ant + Em[i]*Dm*C_ant
        Z_ant = Z_act
        S_ant = S_act
        C_ant = C_act
        i = i + 1

    ex=C_act+S_act #C_act = X_ant // S_act = Y_ant

    #Resultado
    return ex

>>> CordicEx(-3)
0.04980293761682773
>>> CordicEx(3)
20.079311180816877
```

Figura 5.2: Algoritmo de CORDIC en Python con expansión del rango de convergencia con dos iteraciones negativas.

Deben de tomarse en cuenta las siguientes consideraciones:

- Incremento del valor de las condiciones iniciales a 1,8855.
- Modificación de los dos valores iniciales de la LUT con la ecuación (2.52).
- Incremento en la cantidad de desplazamientos por medio de $1 - 2^i$, con i como valor negativo incluyendo al 0.

Tabla 5.1: Cálculo de las primeras dos iteraciones negativas para e^{-3}

Iterac	$Z_{i+1} = Z_i + d_i * -Bm$	$X_{i+1} = X_i + d_i * 2^{-i} * Y_i$	$Y_{i+1} = Y_i + d_i * 2^{-i} * Z_i$	d_i	Bm
-	-3	1.8855	1.8855	-1	1.354025101
-1	-1.645974899	0.2356875	0.2356875	-1	0.972955075
0	-0.673019825	0.058921875	0.058921875	-1	0.54930614

Tabla 5.2: Cálculo de las primeras dos iteraciones negativas para e^3

Iterac	$Z_{i+1} = Z_i + d_i * -Bm$	$X_{i+1} = X_i + d_i * 2^{-i} * Y_i$	$Y_{i+1} = Y_i + d_i * 2^{-i} * Z_i$	d_i	Bm
-	3	1.8855	1.8855	+1	1.354025101
-1	1.645974899	3.5353125	3.5353125	+1	0.972955075
0	0.673019825	6.186796875	6.186796875	+1	0.54930614

EX EXTENDIDO 3 2 iteraciones

	Z	X	Y
-2	0100000001000000000000000000000000	00111111111100010101100000010000	00111111111100010101100000010000
-1	0100000001000000000000000000000000 + 1011111110101101010101000010110010 ----- 0011111110100101010111101001110	00111111111100010101100000010000 + 00111111110100110010110100001110 ----- 01000000011000100100001010001111	00111111111100010101100000010000 + 00111111110100110010110100001110 ----- 01000000011000100100001010001111
0	0011111110100101010111101001110 + 1011111101110010001001110010101 ----- 00111111001011000100101100000111	01000000011000100100001010001111 + 01000000010100110000111101100 ----- 0100000011000101111110100011101	01000000011000100100001010001111 + 11000000010100011110100011101100 ----- 0100000011000101111110100011101

EX EXTENDIDO -3 2 iteraciones

	Z	X	Y
-2	1100000001000000000000000000000000	00111111111100010101100000010000	00111111111100010101100000010000
-1	1100000001000000000000000000000000 + 00111111101011010101000010110010 ----- 1011111110100101010111101001110	00111111111100010101100000010000 + 10111111110100110010110100001110 ----- 00111110011100010101100000010000	00111111111100010101100000010000 + 10111111110100110010110100001110 ----- 00111110011100010101100000010000
0	1011111110100101010111101001110 + 0011111101110010001001110010101 ----- 10111111001011000100101100000111	00111110011100010101100000010000 + 10111110011101010000001000001100 ----- 00111101011100010101100000010000	00111110011100010101100000010000 + 10111110011101010000001000001100 ----- 00111101011100010101100000010000

Figura 5.3: Representación binaria punto flotante de cada una de las tablas 5.1 y 5.2.

La figura 5.3 muestra que el valor del exponente es invariante para X_{i+1} y Y_{i+1} (la LUT debe ser modificada). La mantisa para la suma de la iteración negativa -1 se mantiene constante tanto para números negativos como positivos. Sin embargo, la mantisa para la suma de la iteración 0 cambia según sea el valor por calcular, positivo o negativo. Esta estructura repetitiva puede usarse para reducir hardware significativamente y expandir el rango original de convergencia a $[-3, 44515, 3, 44515]$.

5.2 Sistema de deslinealización implementado en hardware (CORDIC)

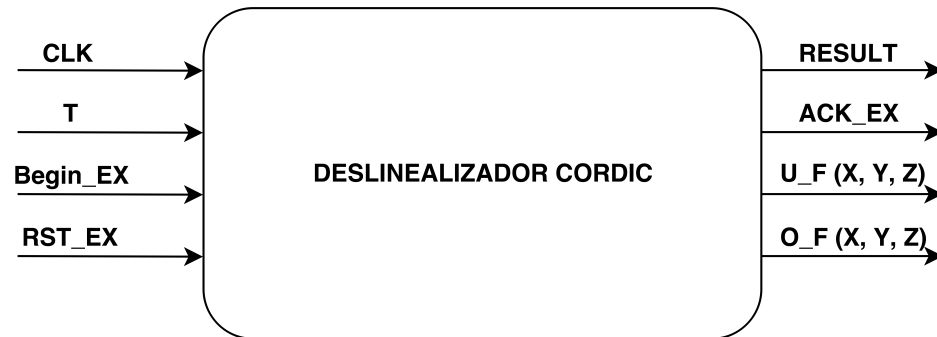


Figura 5.4: Bloque de deslinealización: Entradas y salidas para el cálculo de la función exponencial basada en el algoritmo de CORDIC e implementación en hardware.

Se muestra la figura 5.4 la cuál se considera el bloque principal del algoritmo de CORDIC, que tiene cuatro entradas: CLK, T, Begin_EX, RST_EX y ocho salidas: ACK_EX, RESULT, U_F (X,Y,Z), O_F (X,Y,Z).

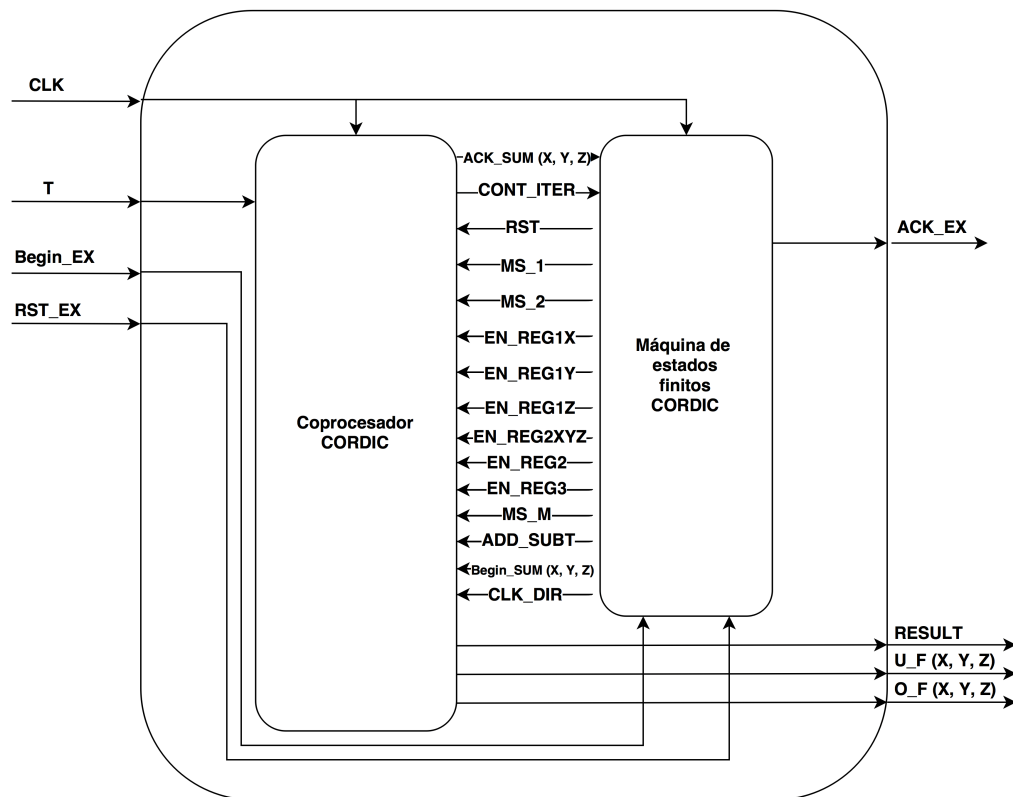


Figura 5.5: Sistema de deslinealización: coprocesador punto flotante de precisión simple (32 bits) basado en el algoritmo de CORDIC y una máquina de estados finita.

El sistema de deslinealización de la figura 5.5 cuenta con dos módulos principales:

- *Coprocesador Cordic*: realiza las operaciones necesarias requeridas por la unidad, en la misma ingresan y salen los datos en formato punto flotante.
- *Máquina de estados finitos*: determina las acciones de control necesarias para el funcionamiento adecuado del coprocesador, se encarga de generar la sincronía correcta del sistema.

Señales de datos:

- *T*: dato de entrada, argumento de la función exponencial.
- *RESULT*: resultado del cálculo de la función exponencial.

Señales de control:

- *CLK*: reloj del sistema. Coordina el funcionamiento de múltiples circuitos.
- *Begin_EX*: encargada de dar inicio al cálculo de la función exponencial.
- *RST_EX*: realiza un reset al bloque general, compuesto por el coprocesador y la máquina de estados finitos.
- *ACK_EX*: bandera generada para indicar que la unidad ha terminado de procesar un dato.
- *Begin_SUM (X,Y,Z)*: encargada de iniciar cada una de las tres unidades de suma-resta coma flotante.
- *ACK_SUM (X,Y,Z)*: bandera generada por cada una de las unidades de suma-resta coma flotante para indicar que el cálculo ha sido realizado.
- *O_F (X,Y,Z)*: bandera de overflow generada por las unidades de suma-resta coma flotante.
- *U_F (X,Y,Z)*: bandera de underflow generada por las unidades de suma-resta coma flotante.
- *CLK_DIR*: habilitación del enable para el contador de iteraciones.
- *CONT_ITER*: encargada de llevar el registro del número de iteraciones llevadas a cabo por la unidad, lo que permite a la máquina de estados detenerse cuando se ha alcanzado un determinado número finito.
- *RST*: realiza un reset de todos los registros del coprocesador.
- *MS_1* , *MS_M* , *MS_2*: determinan la selección de cada multiplexor, Mux 1ra etapa, Mux etapa intermedia, Mux 2da etapa.
- *EN_REG1X* , *EN_REG1Y* , *EN_REG1Z*: habilitación de enable en registros de la primera etapa REG1X, REG1Y, REG1Z, establecidos para almacenar datos.
- *EN_REG2*: activa enable para registro que contiene los valores desplazados de la segunda etapa.
- *EN_REG2XYZ*: activa enable del registro REG2XYZ de la segunda etapa, valores anteriores de X, Y, Z.
- *EN_REG3*: activa el enable del registro REG3 que contiene el valor final calculado.
- *ADD/SUBT*: elección del modo de operación de la unidad suma/resta flotante.

5.3 Diseño e implementación del coprocesador de deslinealización por medio del algoritmo de CORDIC

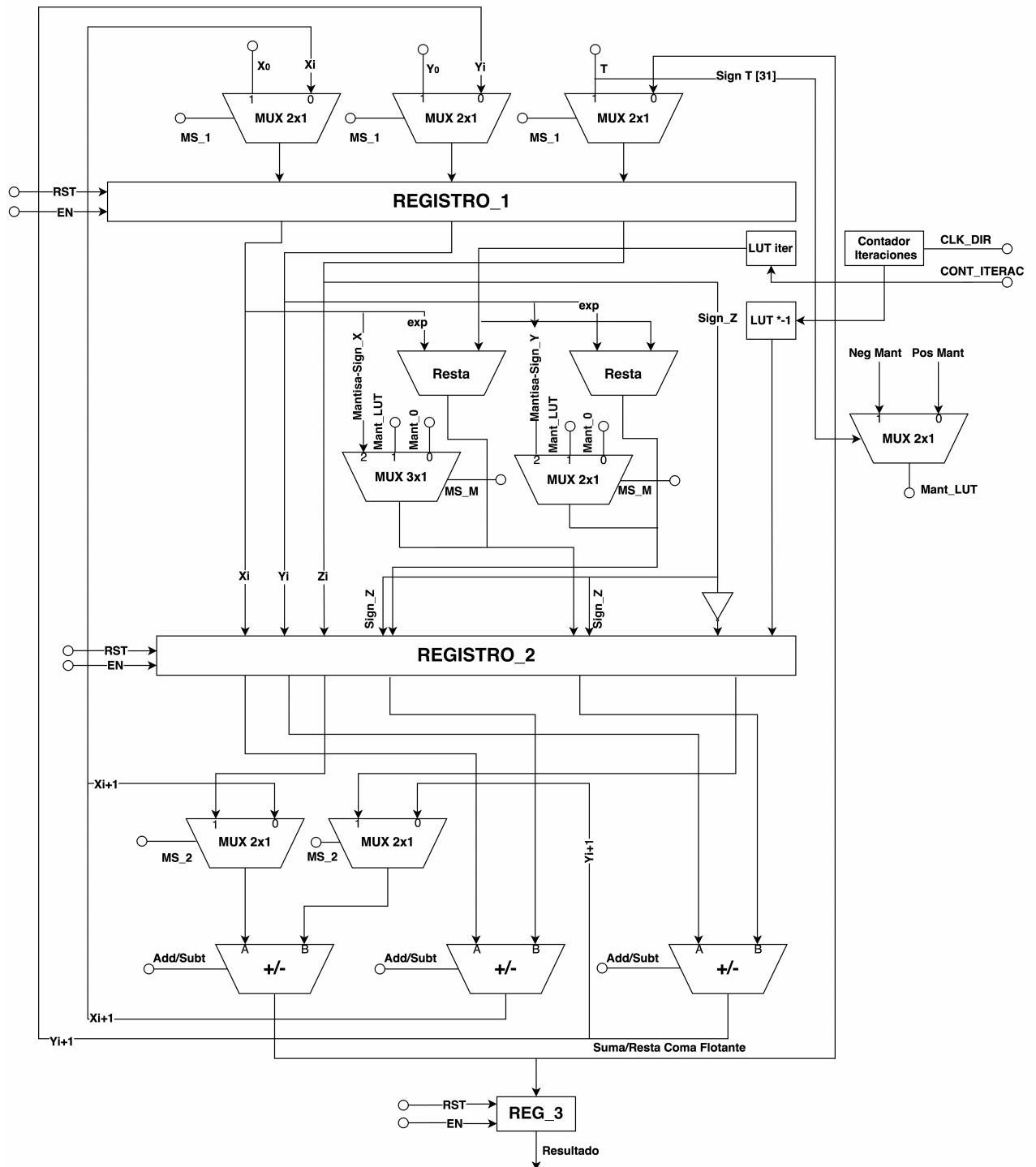


Figura 5.6: Coprocesador segmentado para el cálculo de una función exponencial con el algoritmo de CORDIC en hardware.

Para el desarrollo de la arquitectura segmentada de la figura 5.6 se siguió un enfoque similar al encontrado en [11], dónde por medio del aprovechamiento del paralelismo de la misma se implementaron diversas etapas que requirieron de un diseño adecuado del control de la secuencia de datos. Así se procesan de manera adecuada los mismos los cuales se encuentran en formato *IEEE754*, se introduce en la arquitectura tres unidades de suma/resta coma flotante de manera que aceleren la velocidad de procesamiento del sistema, ya que la utilización de un único sumador/restador vuelve muy lento el cálculo consumiendo una gran cantidad de tiempo de ejecución, como se demostró en [11].

La deslinealización inicia con la precarga de los valores iniciales de X_0 y Y_0 que a su vez poseen una constante fija al inicio del cálculo. Z_0 tiene como valor T, que es el argumento de la función exponencial (e^T). Una vez que se tienen los tres valores se cargan los mismos en el *Registro_1* en la primera etapa de segmentación.

El cálculo cruzado del valor siguiente X_i necesita determinar Y_0 con un desplazamiento, lo que se realiza mediante una resta al valor del exponente y un determinado cambio de signo δ_i . El mismo comportamiento se establece para Y_i e incluso se aprovecha la simetría del cálculo, como se aprecia en las ecuaciones de la tabla 5.1 y 5.2. Para la determinación de los valores a desplazar se hace uso de una tabla de consulta conocida como Look-Up Table (LUT) donde se almacenan los valores dados por la ecuación 2.52 para todos los índices, ya sean negativos y positivos.

La variable Z_i determina el signo δ de las operaciones en el algoritmo de CORDIC hiperbólico, como se puede observar en la figura 5.3. Se puede utilizar un circuito de comparación para demostrar el comportamiento entre los signos de X_i , Y_i y Z_i contra el signo de Z_i . La tabla 5.3 describe el funcionamiento del circuito diseñado.

Tabla 5.3: Signo δ de la iteración siguiente para las variables X_{i+1} , Y_{i+1} y Z_{i+1} , comparando el signo de las variables X_i , Y_i y Z_i contra el signo de Z_i

Sign X_0	Sign Z_0	Sign Y_0	Sign Z_0	Sign X_i	Sign Z_i	Sign Y_i
0	0	0	0	0	1	0
0	0	1	0	0	1	0
0	1	0	1	1	0	1
0	1	1	1	1	0	1
1	0	0	0	0	1	0
1	0	1	0	0	1	0
1	1	0	1	1	0	1
1	1	1	1	1	0	1

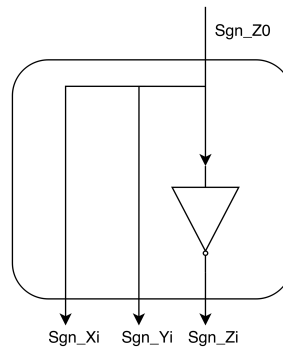


Figura 5.7: Circuito de comparación de signo actual δ , para las variables X_{i+1} , Y_{i+1} y Z_{i+1} de la iteración siguiente, utilizando la tabla 5.3.

El circuito de comparación es simple y sólo es conformado por un inversor para determinar los signos de los valores siguientes necesarios, lo que es fácilmente deducible de la tabla 5.3.

El deslinealizador necesita precargar valores de constantes para cada iteración que realiza, lo que se genera mediante la incorporación de dos LUTs en base a un par de memorias de sólo lectura o ROMs, de manera que puedan accederse cuando un contador de iteraciones indique la dirección del dato que se necesite en el momento exacto, tanto para calcular X_{i+1} como también Y_{i+1} . Además, ambas memorias están sincronizadas por número de iteración.

Las memorias de sólo lectura creadas para el coprocesador son las siguientes:

- *LUT_Z*: almacena los valores de $\operatorname{arctanh}(1 - 2^{-i-2})$ (iterac. negativas) y $\operatorname{arctanh}(2^{-i})$ (iterac. positivas) para cada iteración.
- *LUT_ITER*: almacena los desplazamientos que se deben realizar para cada iteración, se modifican los dos primeros valores por 0 debido al análisis realizado de la figura 5.3 para calcular las iteraciones negativas. Las iteraciones positivas 4, 13, 40, ... k, 3k+1 repiten desplazamientos.

Una vez realizados los desplazamientos en exponente y debida asignación del signo, el valor siguiente es almacenado en el *Registro_2*, se realiza la suma $X_i + \delta \cdot Y_{\text{desplazado},i}$ y se almacena el resultado en el *Registro_1X*; igualmente, se realiza el mismo procedimiento para Y_{i+1} (se utiliza otro sumador punto flotante), $Y_i + \delta \cdot X_{\text{desplazado},i}$, que se almacena en el *Registro_1Y*. Por último se calcula el valor de Z_{i+1} , con el restante sumador para $Z_i + LUT_i$, lo que se almacena en el *Registro_1Z*. Este proceso se realiza un número finito de iteraciones.

Una vez que se ha alcanzado el número de iteraciones previamente definido, se hace una suma de los valores X_{i+1} con Y_{i+1} finales, y se almacenan en el *Registro_3* que contiene el valor final.

5.4 Ajuste de rango para unidad de deslinealización

Cómo se vió en el capítulo 3, los valores de $\theta_2 = \ln(I_s)$ se encuentran en un rango de convergencia de $[-13, -11.59503504]$, muy distante al planteado en el presente capítulo ($[-3.44515, 3.44515]$). Para corregir esto se hace uso de una de las propiedades generales del exponencial:

$$e^{\alpha+\beta} = e^{\alpha} * e^{\beta} \quad (5.1)$$

Para adecuar la ecuación 5.1 al rango de convergencia del parámetro se plantea realizar una suma de un valor de 10 a todos los términos en el rango de θ_2 , lo que hará que los nuevos valores estén situados en $[-3, -1.59503504]$ con ello el sistema de deslinealización será capaz de procesarlos, pero necesariamente se debe cancelar el término de suma introducido de la siguiente forma:

$$e^{\alpha+10} * \frac{1}{e^{10}} = e^{\alpha} * e^{10} * \frac{1}{e^{10}} = e^{\alpha} \quad (5.2)$$

Para la cancelación, se necesita de una multiplicación del resultado obtenido por una constante de $\frac{1}{e^{10}}$, para lo que se hace uso de una unidad de multiplicación de coma flotante [13] capaz de devolver el valor de la función exponencial para el rango adecuado del parámetro.

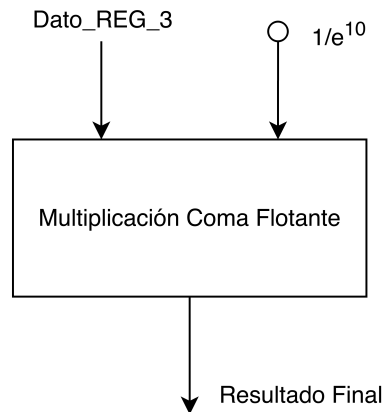


Figura 5.8: Esquema ideado para la multiplicación para determinar el cálculo de la función exponencial.

Una vez determinados los cambios se implementan en el coprocesador, se elimina el multiplexor Z_0 y Z_{i+1} para la primera etapa, se reemplazan los multiplexores 2x1 por multiplexores 3x1 (para el aprovechamiento del hardware presente) después de la segunda etapa de segmentación, y se agrega la multiplicación y un registro inmediatamente después del *Registro_3*, como se muestra en la figura 5.9.

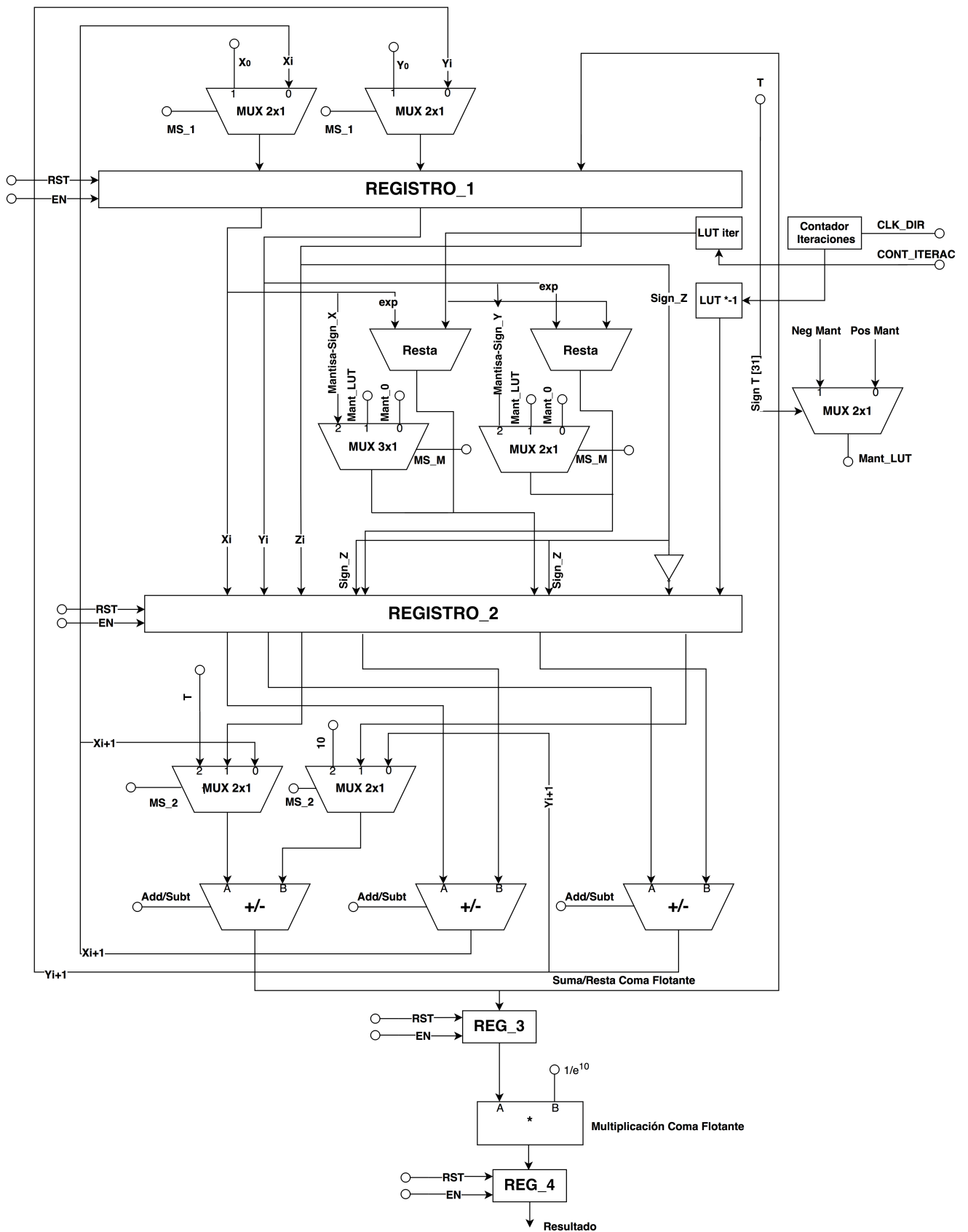


Figura 5.9: Coprocesador segmentado modificado para el cálculo de una función exponencial con el algoritmo de CORDIC para el parámetro $\theta_2 = \ln(I_s)$.

5.5 Sistema de control para la unidad de coprocesamiento CORDIC por medio de un máquina de estados finita (FSM)

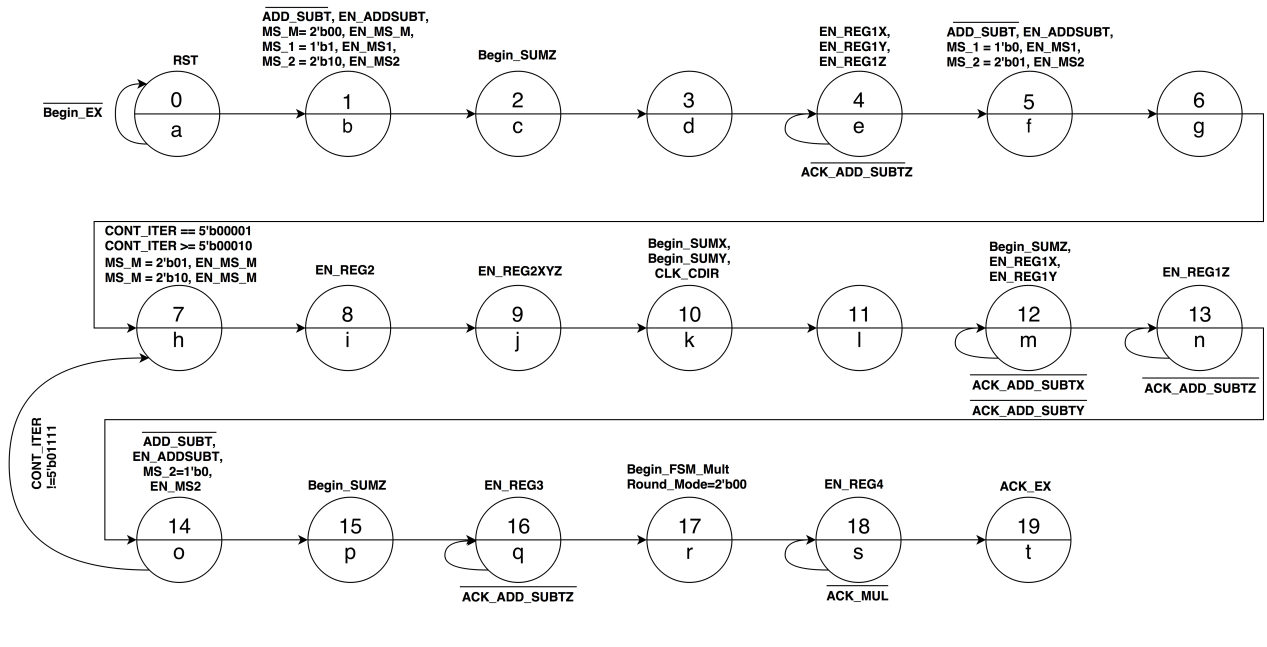


Figura 5.10: Máquina de estados finitos para la unidad de coprocesamiento CORDIC.

La máquina de estados finita se diseña primeramente para calcular el valor del dato T más 10 ($T+10$) que se definió en la sección 5.4, seguidamente se habilitan los registros de la primera etapa para X_0 , Y_0 y Z_0 , se sincronizan las LUT de desplazamiento e iteración, se calculan los desplazamientos, se habilita el valor de mantisa según la iteración y se habilitan las señales de EN_REG2 y $EN_REG2XYZ$.

Una vez calculados los nuevos valores desplazados se suman para determinar el valor de X_{i+1} , Y_{i+1} y Z_{i+1} respectivamente. Este ciclo se repite un número finito de veces y cada vez que se recorre esta secuencia se realizará una cuenta de iteración. Cuando se termina el cálculo, se habilita el EN_REG3 para que luego se ejecute la multiplicación del dato por la constante $\frac{1}{e^{10}}$ y finalmente habilite EN_REG4 .

5.6 Reducción de hardware en la arquitectura CORDIC

En la arquitectura planteada anteriormente, se denota una gran utilización de recursos debido al ajuste para el rango planteado. En este caso se pretende crear una manera de reducir los mismos, mediante un nuevo coprocesador con el fin de comparar sus tiempos de ejecución y porcentajes de error.

Para esto se reemplaza la lógica combinatorial de desplazamiento y suma que se encuentra entre la primera y segunda etapa de segmentación, por dos multiplicadores coma flotante completamente combinatoriales, los valores de la *LUT_ITER* deben de modificarse en este caso para incluir el valor de desplazamiento tanto de iteraciones negativas como positivas, de la forma $1 - 2^{-i-2}$ (iterac. negativas) y 2^{-i} (iterac. positivas) para cada iteración.

No se alteran ninguno de los principios del algoritmo, únicamente el cálculo cruzado es reemplazado por un método incluso más eficiente capaz de eliminar dependencias de hardware y expandir fácilmente los límites del dominio de las funciones hiperbólicas sin realizar análisis similares al de la figura 5.3.

La máquina de estados que controla esta nueva arquitectura, será incluso más reducida ya que no se deben de habilitar los multiplexores que se encuentran en la primera y segunda etapa, dichos estados se reemplazan por un estado de espera para que los multiplicadores sean capaces de calcular los valores requeridos por cada iteración.

```
import math

def CordicEx(T):

    #Condiciones iniciales
    Z_ant=T
    C_ant=11273.4
    S_ant=11273.4

    #LUT (Valores Modificados, iteraciones negativas)
    Bm=[3.118184795, 2.770631773, 2.422093543, 2.071567363, 1.716993602,
    i=0

    #Cantidad de desplazamientos por iteracion
    Em=[(1-2**-8), (1-2**-7), (1-2**-6), (1-2**-5), (1-2**-4), (1-2**-3),

    while i<15:

        Dm= Z_ant/(abs(Z_ant))
        Z_act= Z_ant + Dm*-1*Bm[i]
        C_act= C_ant + Em[i]*Dm*S_ant
        S_act= S_ant + Em[i]*Dm*C_ant
        Z_ant = Z_act
        S_ant = S_act
        C_ant = C_act
        i = i + 1

        ex=C_act+S_act #C_act = X_ant // S_act = Y_ant

    #Resultado
    return ex

>>> CordicEx(-13)
2.258749165346715e-06
>>> CordicEx(13)
443705.83492763847
```

Figura 5.11: Algoritmo de CORDIC en Python con expansión del rango de convergencia con siete iteraciones negativas.

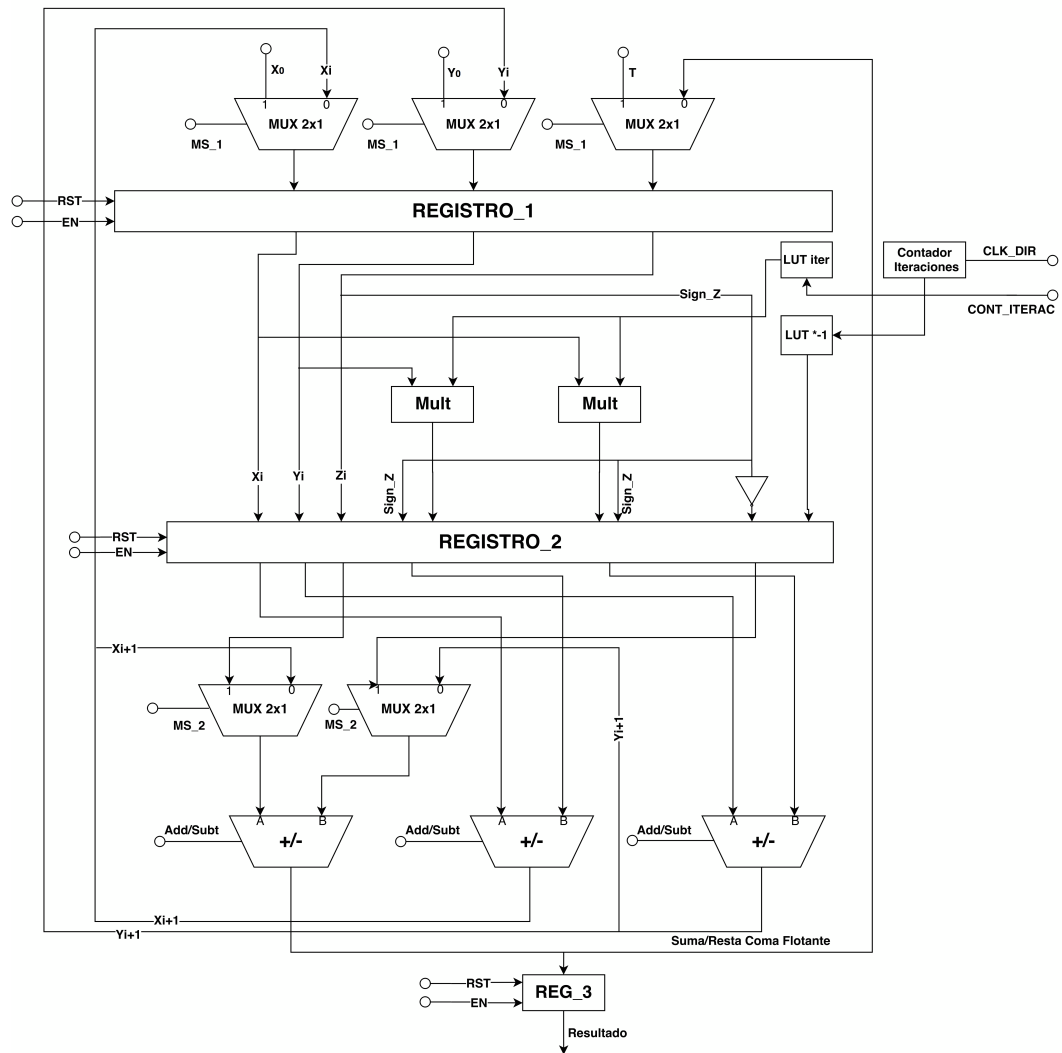


Figura 5.12: Coprocesador segmentado con reducción de hardware para el cálculo de una función exponencial con el algoritmo de CORDIC.

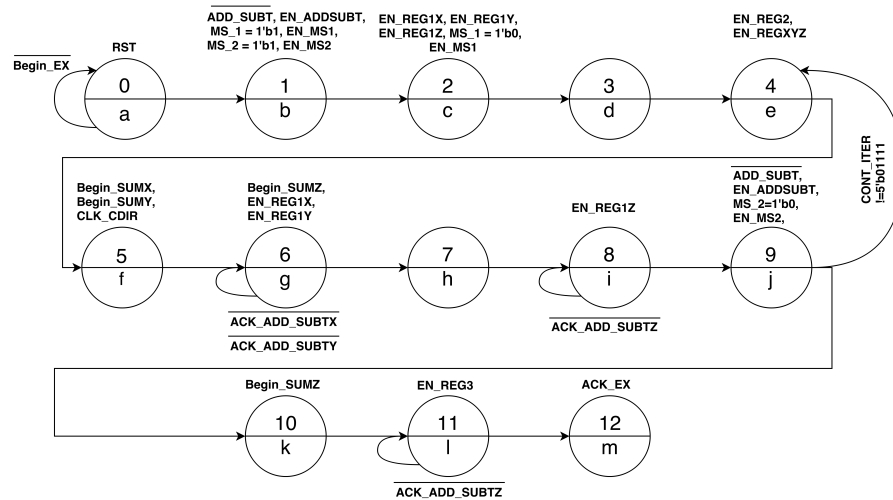


Figura 5.13: Máquina de estados finitos para la unidad de coprocesamiento CORDIC.

5.7 Verificación y resultados del sistema de deslinealización mediante Verilog HDL

Debido a la importancia de comprobar el error de la unidad bajo prueba, se implementa en Verilog una prueba de banco (*testbench*) con al menos 1000 valores provenientes del sistema de conversión-desnormalización para $\ln I_s$. El esquema a seguir para la comprobación de los datos se presenta en la figura 5.14

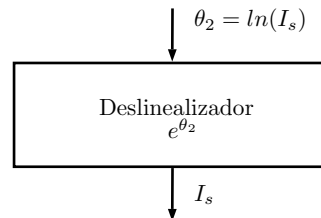


Figura 5.14: Esquema general para la comprobación de datos del sistema de deslinealización.

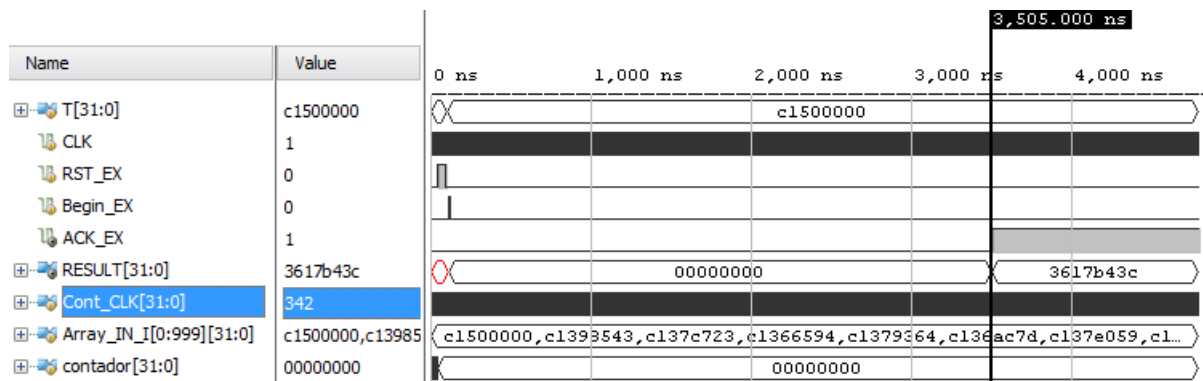


Figura 5.15: Simulación de la arquitectura modificada implementada en Verilog, ingresando un archivo de texto, con 1000 valores del intervalo de convergencia para θ_2 .

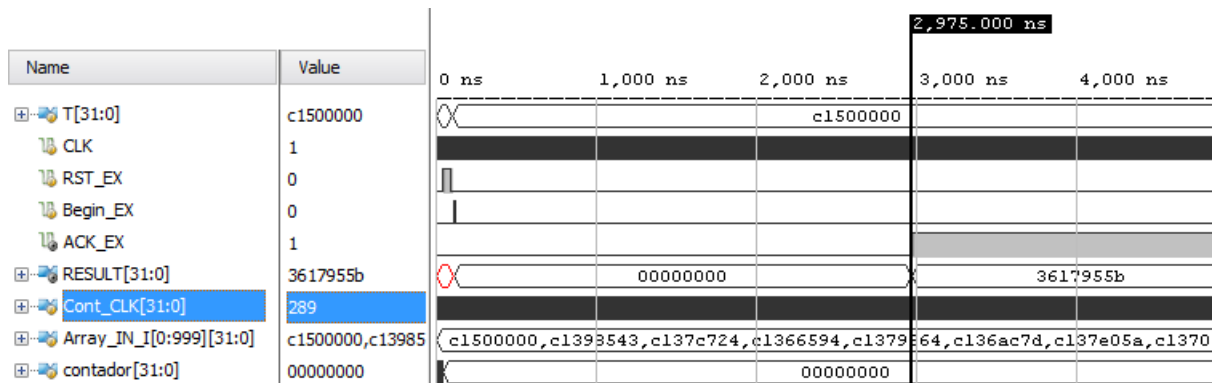


Figura 5.16: Simulación de la arquitectura reducida implementada en Verilog, ingresando un archivo de texto, con 1000 valores del intervalo de convergencia para θ_2 .

5.7.1 Resultados de la simulación del rango de convergencia del circuito deslinealizador CORDIC

Es de suma importancia verificar que el diseño creado mediante el algoritmo de CORDIC funcione de manera adecuada, para esto se genera una prueba de banco (*testbench*) con mil valores de entrada. En este caso se realizaron 15 iteraciones en ambas máquinas de estados del CORDIC para obtener su error, esto con el fin de medir el tiempo de ejecución. En la tabla 5.4 se puede observar el resumen de los resultados más relevantes para las simulaciones del rango de convergencia utilizando el algoritmo de CORDIC.

Tabla 5.4: Resultados experimentales obtenidos por medio de una simulación post-implementación para I_s .

	$I_s(modif)$	$I_s(reduc)$
Error máximo (%)	0,053582	0,49983
Error promedio (%)	0,026664	0,25481
Desviación estándar	0,011943	0,11209
Número de ciclos	342	289
Tiempo de ejecución (<i>us</i>)	3,42	2,89

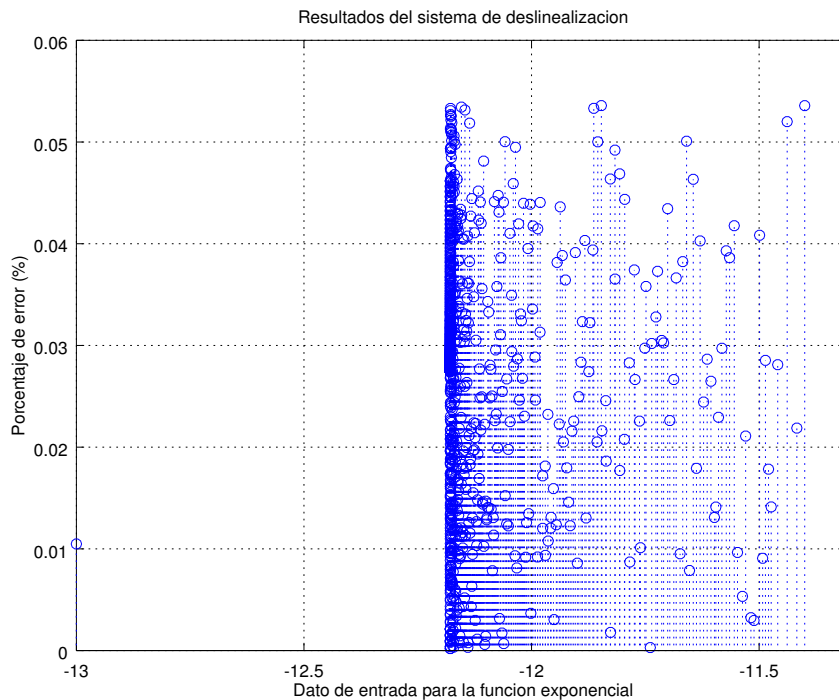


Figura 5.17: Porcentaje de error para los datos de la simulación post-implementación del deslinealizador, tomando mil valores de entrada dentro del rango de convergencia y 15 iteraciones para el algoritmo de CORDIC.

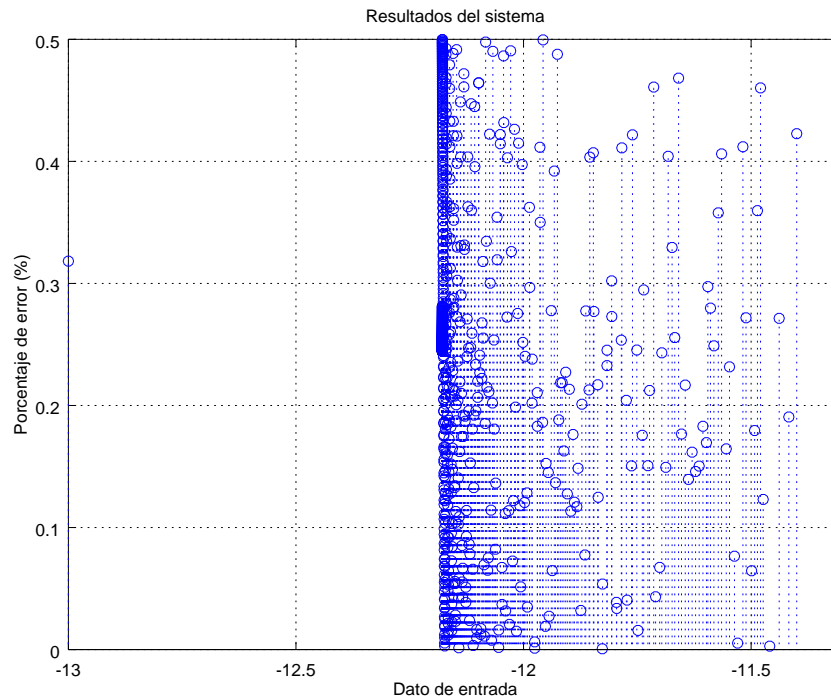


Figura 5.18: Porcentaje de error para los datos de la simulación post-implementación de la arquitectura reducida para el algoritmo de CORDIC.

El error máximo para ambas arquitecturas no supera ni tan siquiera un 1%. Sin embargo, la segunda posee un mayor error (0,49983%) comparada con la modificada (0,053582%) debido a que los multiplicadores introducidos no poseen una etapa de redondeo. Los resultados de ambas comprueban que el número de iteraciones es aceptable para obtener tanto un adecuado tiempo de procesamiento como una buena precisión.

Debido a que la segunda arquitectura posee menor tiempo de ejecución y menor uso de recursos se decide utilizarla como parte del sistema completo, esto a pesar de que se incrementa el error en los datos obtenidos, en esta se pueden calcular números en el rango $[-15.54462, 15.54462]$ sin ningún tipo de limitación.

Capítulo 6

Prueba completa de la unidad sobre una placa de desarrollo Nexys 4

En este capítulo se explica la prueba global de las unidades implementadas en un dispositivo programable (FPGA) y posteriormente se discuten los resultados obtenidos. Para esto primeramente se detalla el bloque general que funcionará como unidad bajo prueba (UUT), simulación de la unidad completa y se discute el desarrollo del ambiente para la verificación.

6.1 Descripción de la unidad completa bajo prueba

Inicialmente se debe detallar la unidad como un solo conjunto que contiene en este caso:

- *Parámetro θ_1* : los datos del mismo se emplean de forma lineal, debido a que el comportamiento no depende de ninguna otra función. Se trabaja únicamente con el sistema de conversión como fija a coma flotante y la desnormalización.
- *Parámetro θ_2* : este requiere tanto del sistema de conversión coma fija a coma flotante y desnormalización como también de una deslinealización.

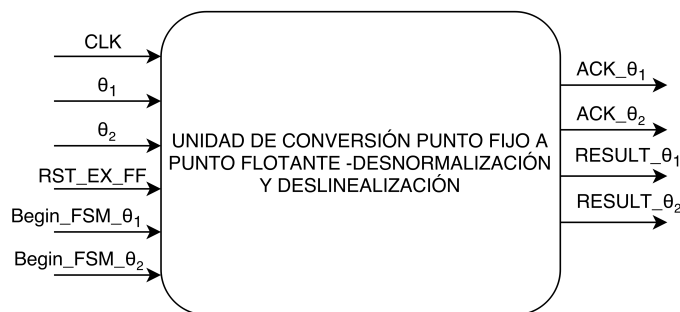


Figura 6.1: Unidad general del sistema conversión - desnormalización y deslinealización de parámetros.

La unidad de la figura 6.1 posee 6 entradas y 4 salidas, se detallan a continuación:

Entradas:

- CLK : reloj del sistema. Coordina el funcionamiento de múltiples circuitos.
- θ_1 : dato generado por el modelo del estimador de parámetros, relación directa con V_{oc} (tensión de circuito abierto del panel).
- θ_2 : dato generado por el modelo del estimador de parámetros, relación directa con la corriente de saturación del panel.
- RST_EX_FF : encargada de reestablecer los valores iniciales de la unidad.
- $Begin_FSM_01$ y $Begin_FSM_02$: inician la máquina de estados del convertidor coma fija a coma flotante.

Salidas:

- ACK_01 : indica que el resultado para α se encuentra listo.
- ACK_02 : indica que el resultado para I_s se encuentra listo.
- $RESULT_01$: resultado de α en coma flotante.
- $RESULT_02$: resultado de I_s en formato coma flotante.

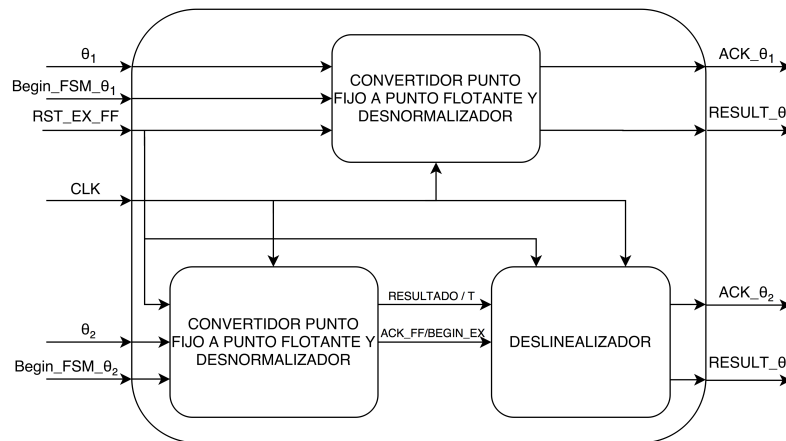


Figura 6.2: Unidad general completa que contiene los sistemas de conversión coma fija a coma flotante y deslinealización.

En la figura 6.2 se muestra la conformación de la unidad y se detalla cada uno de los bloques funcionales implementados, que se encuentran sincronizados por una única señal de reloj (CLK). Para α se tienen las señales $Begin_FSM_01$, RST_EX_FF y el dato de entrada, donde únicamente la conversión-desnormalización es llevada a cabo por la unidad por lo que sólo presenta la ejecución del mismo en la ruta de datos para este parámetro.

Para I_s se tienen las señales $Begin_FSM_02$, RST_EX_FF y el dato de entrada. En este caso la secuencia comienza cuando se realiza la conversión-desnormalización. Una vez finalizada se acciona una bandera de fin de cálculo para habilitar el sistema de deslinealización. Ambos sistemas accionan un par de señales que tienen como fin indicar que han finalizado y los datos han sido procesados.

6.2 Simulación generada para la unidad completa

Una vez definido el esquema general por realizar, se implementó cada uno de los bloques diseñados y se procedió a su consecuente simulación.

Una vez más, se sigue el procedimiento de utilizar 1000 valores que describen el comportamiento esperado para cada uno de los valores generados por el modelo teórico del estimador. Se toma referencia de cada uno de los valores estimados para α y $\ln(I_s)$, generados por las ecuaciones 6.1 y 6.2.

$$\dot{\theta}_1 = (\gamma_{11}z + \gamma_{12})(y - \hat{y}) \quad (6.1)$$

$$\dot{\theta}_2 = (\gamma_{21}z + \gamma_{22})(y - \hat{y}) \quad (6.2)$$

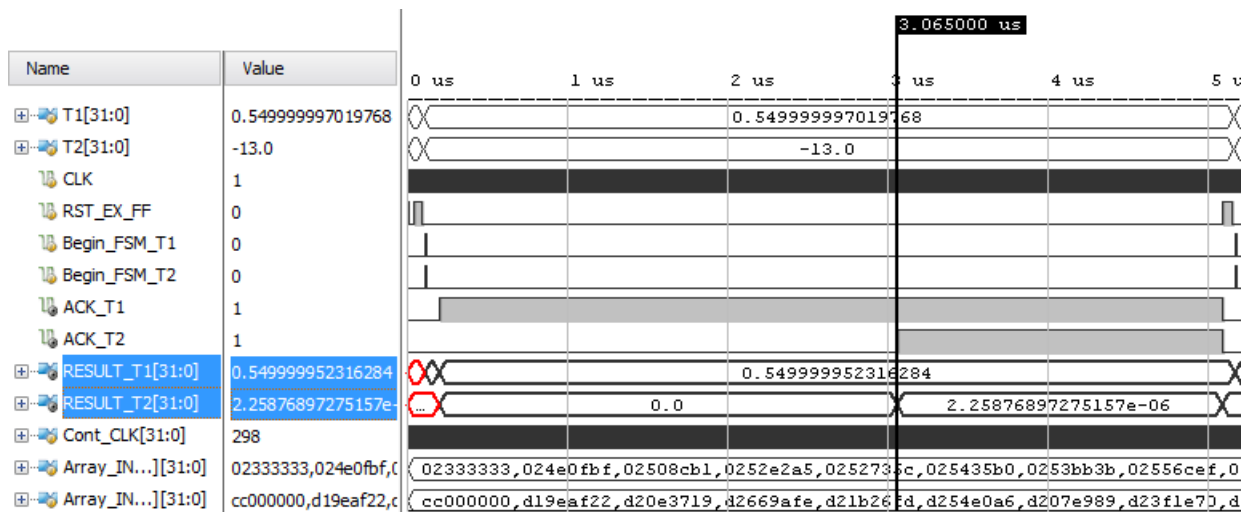


Figura 6.3: Simulación de la unidad completa para un modelo de estimación de parámetros.

Las simulaciones para el comportamiento del sistema a nivel de transferencia de registros (RTL Level) presentan resultados a nivel de verificación de sintaxis. Sin embargo, para efectos de verificación de un diseño que cumpla las especificaciones de síntesis, ruteo-posicionamiento y tiempos adecuados a transferir en un determinado dispositivo programable, es necesario generar al menos un par de pasos adicionales de simulaciones Post-Synthesis y Post Place & Route, esto con el fin de tener una referencia clara de cuál será su comportamiento una vez que el diseño sea finalizado. Es por esto que en la figura 6.3 se muestra una simulación de tiempos posterior a la implementación, lo que permite tener ya una buena certeza de que el sistema funcionará en la FPGA seleccionada [42].

6.3 Sistema para realización de pruebas en una placa de desarrollo Nexys 4

Una vez que el diseño y sus correspondientes simulaciones han sido concretadas es necesario desarrollar pruebas en la FPGA. Esta verificación final asegura un funcionamiento correcto. En la figura 6.4 se muestra un diagrama general para la verificación del diseño.

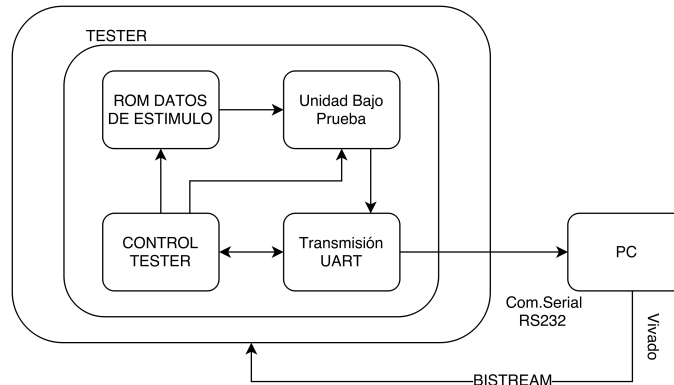


Figura 6.4: Diagrama de flujo general para el ambiente de verificación utilizado en la FPGA, utilizando un tester con una memoria ROM para los vectores de entrada y una UART para la transmisión de los resultados hacia el computador.

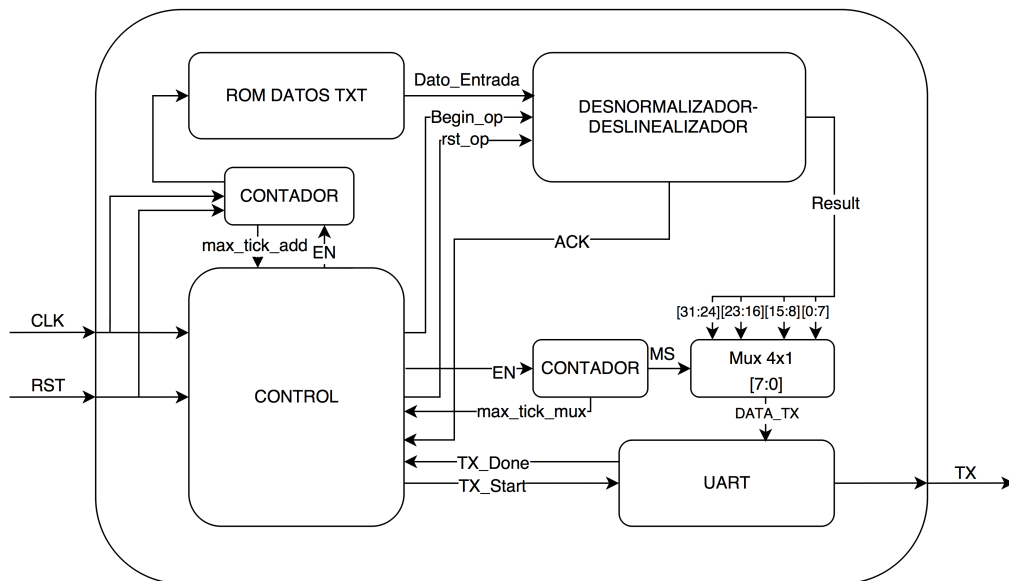


Figura 6.5: Detalle del ambiente de verificación ingresando los datos de estímulo por medio de una memoria ROM y enviando los datos de salida por medio de transmisión serial (UART) desde la Nexys 4 hacia un computador.

En este caso, para poder obtener resultados de la FPGA, fue necesario diseñar un circuito capaz de hacer una prueba con un vector de datos a la entrada de la unidad completa, con el fin de enviar estos por un medio de acceso sencillo, en este caso haciendo uso de transmisión serial (UART) hacia un computador donde puedan ser procesados de manera adecuada.

Procedimiento utilizado:

- Se ingresan 1000 valores por medio de una memoria de sólo lectura (ROM). En este caso, para direccionar los datos de la misma se utiliza un contador de 10 bits (2^{10}).
- Debido a que la transmisión serial envía paquetes de 1 byte, se descompone el dato de 32 bits por enviar en 4 bytes, se realiza mediante una multiplexación y cuenta. El multiplexor contiene el resultado de la división de 4 bytes y por medio del contador se envían uno a uno hasta formar el dato.
- Una máquina de estados actúa como control de los estados del tester. Primero se crea una carga en el contador de dirección de dato en la ROM, se envía el dato a la unidad general (deslinealización-desnormalización), se inicia la operación con la señal *Begin_op* hasta que la unidad procesa el dato y activa la bandera *ACK=High* y se comienza en este momento la transmisión.
- Una vez enviada la secuencia de 4 bytes se activa la bandera *tx_done*. El contador del multiplexor debe continuar para seguir procesando los datos en la memoria, lo que se logra enviando la señal *max_tick_mux*, una vez que el contador termina con el envío de todos los datos se activa *max_tick_add = 1* dando por terminada la transmisión por vía serial.
- La recepción de datos debe de estar habilitada en el computador, mediante un script en Octave, para almacenar los datos codificados en hexadecimal en un fichero en formato *.txt*.

6.4 Resultados de las pruebas de la unidad general en una placa de desarrollo Nexys 4

En los capítulos 4 y 5 se describieron las simulaciones de cada uno de los bloques de la unidad general independientemente. Sin embargo, una simulación completa con todo el sistema es requerida para comprobar que la unidad es capaz de procesar datos de manera adecuada, con una correcta sincronía y sin la presencia de errores. Esta simulación brinda una mejor perspectiva acerca de los porcentajes de error obtenidos una vez se dé la interconexión de los sistemas de conversión-desnormalización y deslinealización respectivamente.

En el capítulo 4 se demostró efectivamente el funcionamiento de la conversión coma fija a coma flotante con un error ínfimo, lo que demuestra que el sistema es capaz de convertir datos con una precisión muy alta y en un tiempo de ejecución muy bajo.

Por su parte, en el capítulo 5 se demostró el funcionamiento del sistema de deslinealización para el rango de convergencia del algoritmo de CORDIC con datos provenientes del modelo del estimador, comprobando que efectivamente se pueden calcular datos fuera de rango realizando un ajuste al hardware presente sin modificar los datos por calcular.

Para la comprobación de la unidad, se debe de tomar en consideración el error que introduce la desnormalización en el cálculo de I_s , ya que se genera la conexión de los dos bloques en la ruta de datos, para α no se tiene tal consideración ya que como se demostró en el diagrama de la figura 6.2 este sólo depende de un único sistema. Una vez obtenidos los datos se pueden analizar aspectos relevantes al diseño como frecuencia de ejecución total, utilización de recursos, reporte de tiempos y consumo de potencia en la placa de desarrollo.

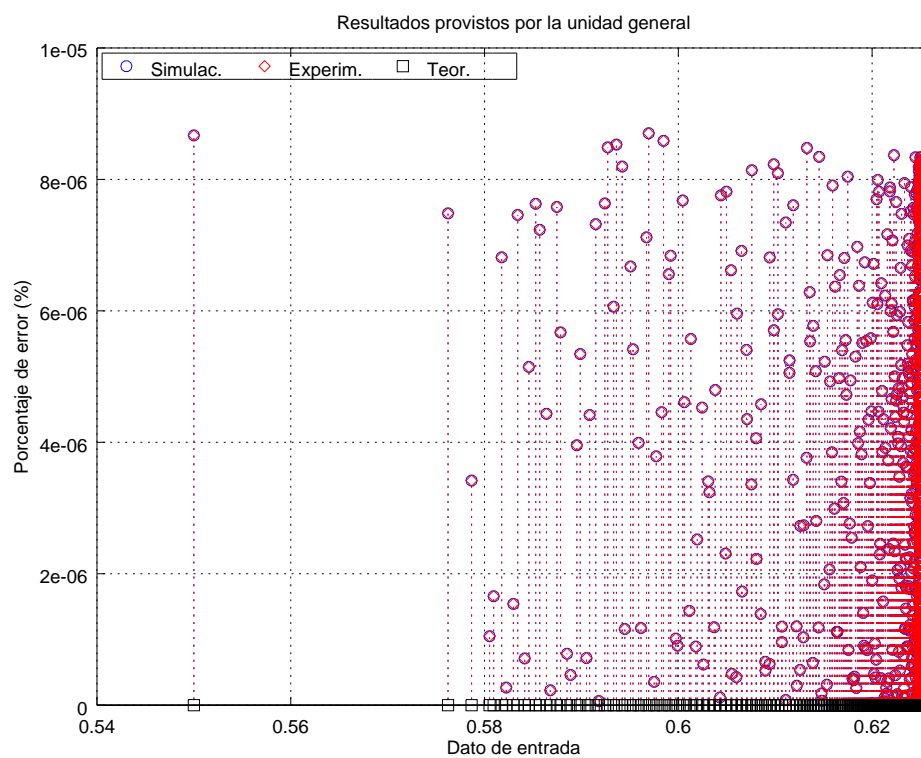


Figura 6.6: Porcentaje de error para la conversión-desnormalización de α obtenido mediante el sistema de verificación.

Efectuada la comprobación se puede observar que los resultados obtenidos para el error experimental no varían con respecto a los obtenidos en las simulaciones realizadas, lo que demuestra que el diseño desarrollado en la herramienta de descripción de hardware Verilog posee un comportamiento prácticamente igual al del diseño sintetizado en la Nexys 4. En la tabla 6.1 se muestra que los datos poseen el mismo valor de error máximo, promedio y tiempo de ejecución.

Tabla 6.1: Comparación de resultados de simulación vs experimentales de α obtenidos por el sistema de verificación implementado en una placa de desarrollo Nexys 4. El funcionamiento en la FPGA es por tanto correcto al 100% contra el modelo.

	<i>Simulac.</i>	<i>Experim.</i>
Error máximo ($\times 10^{-6}$)	8,7021	8,7021
Error promedio ($\times 10^{-6}$)	3,7409	3,7409
Desviación estándar ($\times 10^{-6}$)	2,5751	2,5751
Número de ciclos	9	9
Tiempo de ejecución (<i>ns</i>)	90	90

Para I_s se presentan las figuras 6.7 y 6.8, que demuestran el porcentaje de error incluyendo los dos sistemas. Al igual que en el caso de la comparación realizada para α , se puede notar que el error se mantiene inalterable. Debido a que se produjo la unión de dos sistemas para determinar el cálculo de los datos, el número de ciclos y el tiempo de ejecución aumentan pero no considerablemente, esto se indica mediante la tabla 6.2 en la que se muestran los cambios producidos.

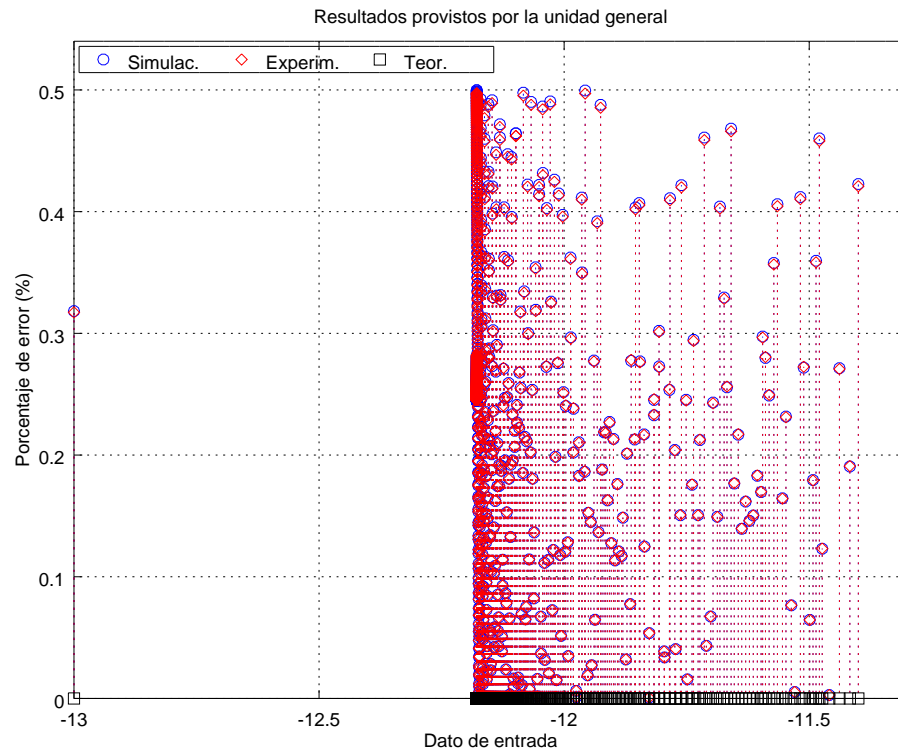


Figura 6.7: Porcentaje de error para la conversión-desnormalización/deslinealización de I_s obtenido mediante el sistema de verificación.

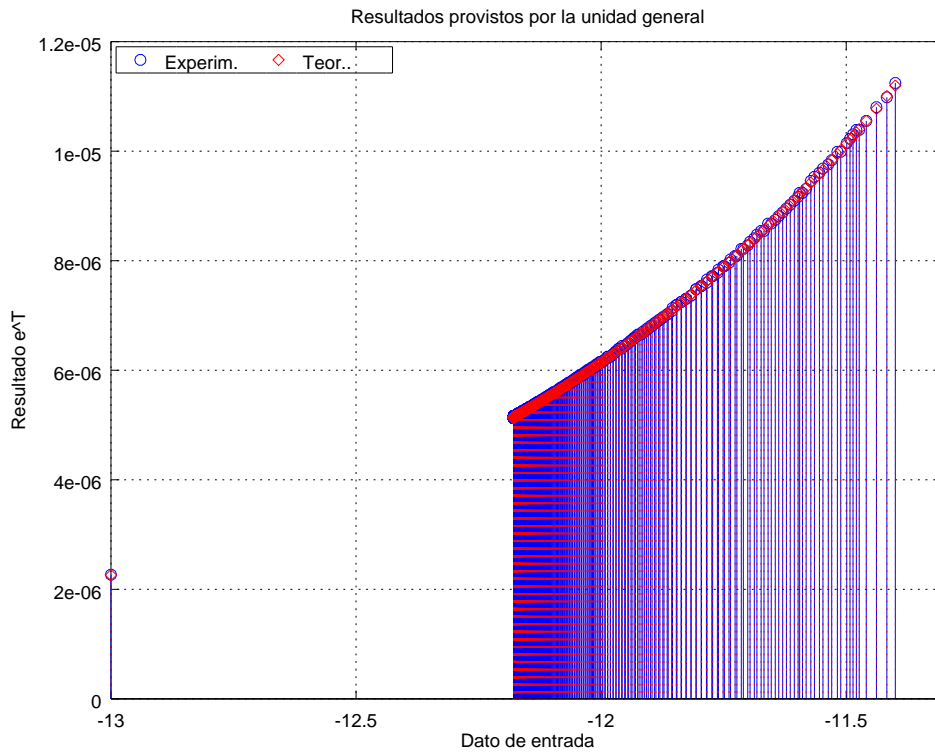


Figura 6.8: Resultado de la conversión-desnormalización/deslinealización para I_s obtenido mediante el sistema de verificación.

Tabla 6.2: Comparación de resultados de simulación vs experimentales de I_s obtenidos por el sistema de verificación implementado en una placa de desarrollo Nexys 4. El funcionamiento en la FPGA es por tanto correcto al 100% contra el modelo.

	<i>Simulac.</i>	<i>Experim.</i>
Error máximo (%)	0,49983	0,499801
Error promedio (%)	0,25481	0,25484
Desviación estándar	0,11209	0,11208
Número de ciclos	298	298
Tiempo de ejecución (<i>us</i>)	2,98	2,98

Dado que la frecuencia de operación del panel fotovoltaico es de 100 kHz, se debe de tomar en consideración que la unidad general implementada necesita operar a una velocidad aún mayor para generar los cálculos necesarios. De la tabla 6.2 se tiene que el tiempo de ejecución total es de 2,98 *us*. Esto representa una frecuencia máxima de 335,57 kHz, se debe de tomar en consideración que el muestreo en la FPGA debe ejecutarse a 1 MHz, lo que representa una limitación que debe corregirse.

6.5 Recursos utilizados

Tabla 6.3: Resumen del reporte post implementación del uso de dispositivos generado por la herramienta Vivado.

Recurso	Utilizados	Disponibles	Utilizados%
LUT	1685	63400	2.66
FF	1327	126800	1.05
BRAM	0.50	135	0.37
DSP	2	240	0.83
IO	3	210	1.43
BUFG	1	32	3.13

En la tabla 6.3 se resume el reporte generado por la herramienta para la implementación de la unidad completa incluyendo en este caso el tester, es por esto que el uso de I/O es de solo un 1,43% sobre el total de entradas disponibles, únicamente se utilizan 3 señales (CLK, RST, TX), si se removiera el tester la utilización de los puertos I/O se incrementaría considerablemente debido a que cada entrada para θ_1 y θ_2 y salida de datos requiere de 32 bits.

La mayor utilización de recursos se encuentra en los recursos de BUFG, LUTs y FFs, con un 3,12%, 2,66% y un 1,05% respectivamente, que son recursos utilizados principalmente como buffers de reloj global, lógica y registros en dicho orden.

6.6 Reporte de tiempos

Es necesario analizar la viabilidad del diseño en términos de tiempo para conocer si este es capaz de alcanzar todas las restricciones establecidas por el usuario. Para esto se hace uso del reporte de tiempos provisto por la herramienta Vivado, en el que se analiza cada una de las rutas de mayor retraso o críticas, de acuerdo a esto se analiza el slack.

Este se define como la diferencia entre el tiempo actual y el tiempo deseado para una ruta de tiempos y define si un circuito puede o no operar a una cierta velocidad o frecuencia específica. Se divide en dos tipos: setup y hold slack.

$$\text{Setup Slack} = \text{Tiempo de setup requerido por el dato} - \text{Tiempo de arribo del dato} \quad (6.3)$$

$$\text{Hold Slack} = \text{Tiempo de arribo de dato} - \text{Tiempo de hold requerido por el dato} \quad (6.4)$$

- Un slack positivo en ambos, indica que el diseño funciona a la frecuencia especificada por las restricciones y que posee un margen disponible.
- Si ambos slack son cero el diseño funciona a la frecuencia especificada pero no tiene un margen disponible, en este caso puede fallar por falta de sincronía.
- Si se tiene un valor negativo, no se alcanzan las restricciones impuestas para el diseño lo que impide que este funcione correctamente, se deben de acelerar las rutas problemáticas.

Se presentan los datos generados por la herramienta para el diseño de la unidad general.

Tabla 6.4: Resumen del reporte post implementación de tiempos del circuito completo, a partir de la herramienta Vivado. CLK=100 MHz

Peor slack de todas las rutas	Tiempo (ns)
Setup	1,578
Hold	0,139
Pulse Width	4,5

6.7 Consumo de potencia

Es necesario conocer el consumo de potencia generado por el circuito una vez este es sintetizado en el dispositivo programable, al igual que en el reporte de tiempos, se hace uso de la herramienta Vivado para determinar el consumo tanto estático como dinámico del diseño. Se muestran en la tabla 6.5 los resultados obtenidos.

Tabla 6.5: Resumen del reporte post implementación de la potencia estática, dinámica y total, de la herramienta Vivado para la unidad general.

Potencia	Consumo de potencia (mW)
Dinámica	23
Estática	97
Total	120

La potencia dinámica en este caso está determinada por cada una de las transiciones generadas por el circuito. Una mayor frecuencia generará transiciones más frecuentes lo que se reflejará en disipación de potencia, el diseño del circuito está generado en base a un reloj de 100 MHz. A continuación se presenta un resumen de la potencia consumida a través del diseño.

Tabla 6.6: Resumen del reporte generado por la herramienta Vivado que indica el consumo de potencia de diversos elementos.

On-chip (mW)	Consumo de potencia (mW)
Clocks	3
Slice Logic	8
Signals	7
Block RAM	2
DSPs	2
I/O	1
Static Power	97
Total Power	120

Capítulo 7

Conclusiones y recomendaciones

7.1 Conclusiones

- El modelo teórico de estimación determinó que es posible predecir los valores y convergencia de los parámetros θ_1 y θ_2 con una alta precisión a partir de datos de simulación con valores reales provistos por el fabricante del panel. Se demostró la dinámica variante en el tiempo de estos como parte de una solución matemática, la cual contribuirá como modelo de referencia dorada para la comparación del comportamiento esperado del mismo una vez este sea implementado en un lenguaje de descripción de hardware, específicamente en Verilog.
- El error del estimador generalmente será mayor si los valores iniciales del algoritmo se encuentran muy distanciados del valor de convergencia.
- Se demostró que es posible obtener un porcentaje de error menor a 5% para la conversión-desnormalización de los valores de los parámetros θ_1 y θ_2 , esto con un error máximo de $8,7021 \cdot 10^{-6} \%$ y $7,9589 \cdot 10^{-6} \%$, para cada uno de ellos respectivamente.
- Mediante la aplicación de una función exponencial a los datos del parámetro θ_2 se verificó que es posible obtener I_s como valor deslinealizado con un error máximo de 0,49983%, determinando la validez del sistema con la expansión y ajuste al rango de convergencia del algoritmo de CORDIC.
- A partir de la verificación funcional de la unidad general se comprobó experimentalmente que los porcentajes de error una vez sintetizados en la placa de desarrollo, tienen un error mucho menor al 5%, con valores máximos de $8,7021 \cdot 10^{-6} \%$ y 0,499801% para α e I_s .
- Se concluyó mediante los tiempos de ejecución de los sistemas que el diseño propuesto con 15 iteraciones tiene un error mínimo y puede ser utilizado como parte del sistema general de aumento de eficiencia de paneles fotovoltaicos ya que puede ejecutar los cálculos a una frecuencia mayor a la de operación del panel.

7.2 Recomendaciones

- Se sugiere agregar una etapa de redondeo a los multiplicadores de la arquitectura reducida para el cálculo del algoritmo de CORDIC, de esta manera se podría reducir aún más el error obtenido hasta incluso igualar el de la primera arquitectura sugerida en que el error máximo obtenido fue de 0,053553%.
- Debido a que solamente se realizó la implementación de la función exponencial (CORDIC) con 15 iteraciones, quizás se pueda tomar en consideración un número mayor de iteraciones para reducir aún más el error en este desarrollo y presentar una comparación detallada de las mismas.
- Si se integra el sistema completo se debe de realizar un adecuado *floorplanning* del diseño, específicamente en la etapa de implementación, debido a que este mejorará los tiempos de *slack* en el circuito, el timing closure y el flujo de datos.

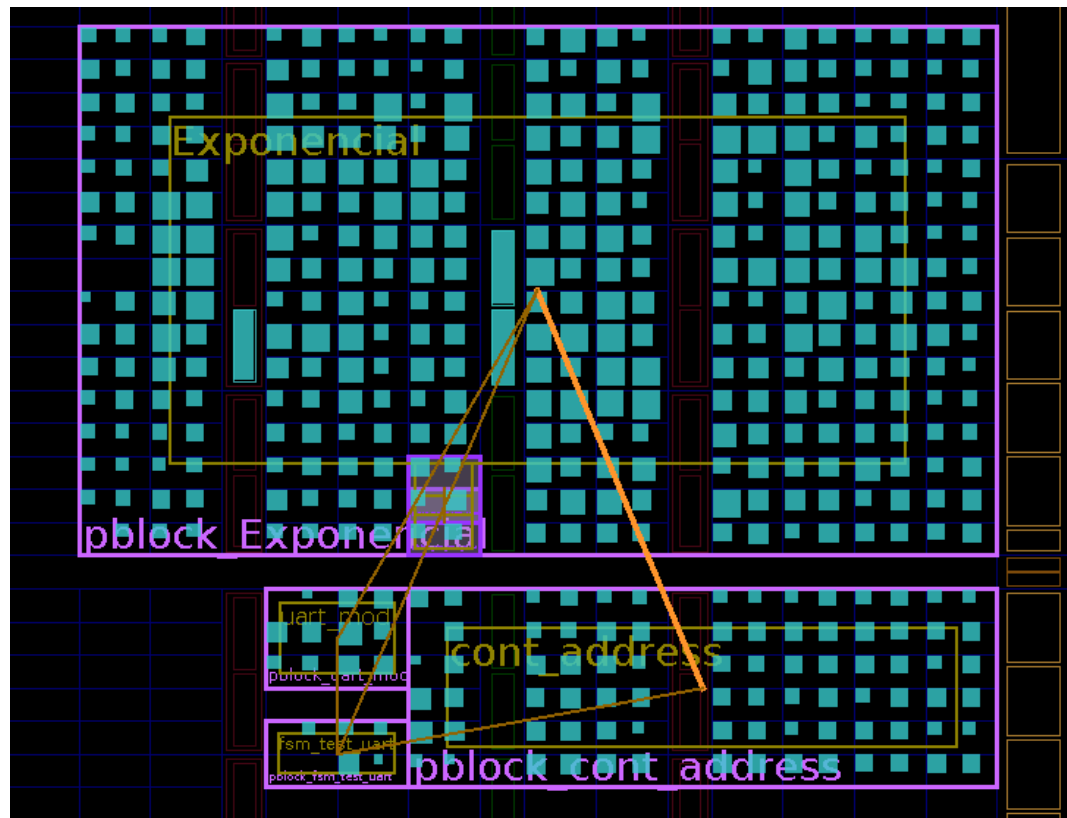


Figura 7.1: Floorplanning realizado para ambas unidades.

- Si se utiliza otro modelo de panel se debe igualmente realizar un análisis del comportamiento mediante el modelo teórico del estimador realizado con la simulación en Python, con el motivo de establecer la mejor aproximación del valor esperado, de manera que se reduzca el tiempo de convergencia de los parámetros.
- Se debe de buscar una alternativa para acelerar el sistema, esto debido a que los cálculos que se realizan sobre las señales se pueden ejecutar a un máximo de frecuencia de 10 kHz.

Bibliografía

- [1] “Field Applications for I-V Curve Tracers.” [En línea]. Disponible en: <http://solarprofessional.com/articles/design-installation/field-applications-for-i-v-curve-tracers> 7
- [2] R. J. Schweers, “Descripción en VHDL de arquitecturas para implementar el algoritmo CORDIC,” Tesis, Facultad de Informática, 2002. [En línea]. Disponible en: <http://hdl.handle.net/10915/3835> 13
- [3] “CORDIC Pseudo-rotation.” [En línea]. Disponible en: https://upload.wikimedia.org/wikipedia/commons/5/5a/CORDIC_Pseudo-rotation.svg 13
- [4] J. S. Walther, “A unified algorithm for elementary functions,” in *Proceedings of the May 18-20, 1971, spring joint computer conference*. ACM, 1971, pp. 379–385. 16
- [5] K. Solar. (2016) High efficiency multicrystal photovoltaic module. [En línea]. Disponible en: <http://www.kyocera.com.sg/products/solar/pdf/kc65t.pdf> 2, 21, 34
- [6] G. E. L. America. (2016) Energía Solar - Fotovoltaica. [En línea]. Disponible en: <http://www.greenenergy-latinamerica.com/es/energia-solar-solar-fotovoltaica-197> 1
- [7] “Potencial de la Energía Solar Fotovoltaica | Recope.” [En línea]. Disponible en: <https://www.recope.go.cr/potencial-de-la-energia-solar-fotovoltaica/> 1
- [8] “Solar Electricity - Photovoltaic Systems.” [En línea]. Disponible en: <http://www.solardirect.com/pv/systems/systems.htm> 1
- [9] “Photovoltaic Systems.” [En línea]. Disponible en: <http://www.renewableenergyworld.com/solar-energy/tech/solarpv.html> 1
- [10] J. Goldemberg, *Energy: What Everyone Needs to Know*. OUP USA, 2012. 1
- [11] A. Cervantes Segura, “Unidad de linealización y normalización para un estimador de parámetros de uso en un sistema de optimización de energía en paneles fotovoltaicos,” Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, Junio 2016. 2, 7, 8, 31, 34, 39, 40, 47
- [12] D. Rodríguez Valverde, “Diseño e implementación de una unidad aritmético-lógica de coma flotante para un procesador de aplicación específica.” Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, Noviembre 2015. 2

- [13] F. López Montero, “Diseño e implementación de hardware para optimizar la unidad aritmética de coma flotante de un procesador de aplicación específica,” Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, Junio 2016. 2, 49
- [14] Z. Sen, *Solar Energy Fundamentals and Modeling Techniques: Atmosphere, Environment, Climate Change and Renewable Energy*. Springer Science & Business Media, Mar. 2008, google-Books-ID: 244OdSz4dNQC. 6
- [15] S. I. Ao and International Association of Engineers, Eds., *World Congress on Engineering and Computer Science, WCECS 2008, San Francisco, USA, 22 - 24 October, 2008*, ser. Lecture notes in engineering and computer science. Hong Kong: IAENG International Association of Engineers, 2008, oCLC: 551898616. 6
- [16] F. M. González-Longatt, “Model of photovoltaic module in Matlab,” *II CIBELEC*, vol. 2005, pp. 1–5, 2005. [En línea]. Disponible en: http://www.academia.edu/download/31994216/Model_of_Photovoltaic_Module_in_Matlab.pdf 7, 21
- [17] “Power Curves & Characteristics for Solar Cells | Samlex Solar.” [En línea]. Disponible en: <http://www.samlexsolar.com/learning-center/solar-panels-characteristics.aspx> 7
- [18] J. F. Ferichola, “Caracterización de módulos fotovoltaicos con dispositivo portátil,” *Universidad Carlos III. Madrid. España*, 2009. [En línea]. Disponible en: https://e-archivo.uc3m.es/bitstream/handle/10016/6037/PFC_Julio_Fernandez_Ferichola.pdf?sequence=1&isAllowed=y 7
- [19] C. Meza and R. Ortega, “Control and estimation scheme for PV central inverters,” in *Information, Communication and Automation Technologies (ICAT), 2013 XXIV International Symposium on*. IEEE, 2013, pp. 1–6. [En línea]. Disponible en: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6684052 9, 10, 20, 23
- [20] J. A. Gow and C. D. Manning, “Development of a photovoltaic array model for use in power-electronics simulation studies,” *IEE Proceedings - Electric Power Applications*, vol. 146, no. 2, pp. 193–200, Mar. 1999. 9
- [21] U. Boke, “A simple model of photovoltaic module electric characteristics.” IEEE, 2007, pp. 1–8. [En línea]. Disponible en: <http://ieeexplore.ieee.org/document/4417572/> 9
- [22] L. Shengyi and R. Dougal, “Dynamic multiphysics model for solar array,” *IEEE Transactions on Energy Conversion*, vol. 17, no. 2, pp. 285–294, Jun. 2002. [En línea]. Disponible en: <http://ieeexplore.ieee.org/document/1009482/> 9
- [23] C. Meza and R. Ortega, “On-line estimation of the temperature dependent parameters of photovoltaic generators,” *IFAC Proceedings Volumes*, vol. 46, no. 11, pp. 653–658, 2013. [En línea]. Disponible en: <http://linkinghub.elsevier.com/retrieve/pii/S147466701633018X> 10, 20, 23

- [24] N. Whitehead and A. Fit-Florea, “Precision & performance: Floating point and IEEE 754 compliance for NVIDIA GPUs,” *rn (A + B)*, vol. 21, pp. 1–1 874 919 424, 2011. [En línea]. Disponible en: http://people.maths.ox.ac.uk/gilesm/cuda/doc/Floating_Point_on_NVIDIA_GPU_White_Paper.pdf 11
- [25] “IEEE Standard for Floating-Point Arithmetic,” *IEEE Std 754-2008*, pp. 1–70, Aug. 2008. 11
- [26] J. J. Raygoza, S. O, M. A. Carrazco, and A. Pedroza, “Implementación En Hardware De Un Sumador De Punto Flotante Basado En El Estándar Ieee 754-2008,” 2009. [En línea]. Disponible en: <http://www.redalyc.org/articulo.oa?id=73012215012> 11
- [27] “IEEE Standard 754 Floating-Point.” [En línea]. Disponible en: <http://steve.hollasch.net/cgindex/coding/ieeefloat.html> 11
- [28] “Introduction to Fixed Point Number Representation.” [En línea]. Disponible en: <http://www-inst.eecs.berkeley.edu/~cs61c/sp06/handout/fixedpt.html> 11, 12
- [29] J. Sudha, M. Hanumantharaju, V. Venkateswarulu, and J. H, “A Novel Method for Computing Exponential Function Using CORDIC Algorithm,” *Procedia Engineering*, vol. 30, pp. 519–528, 2012. [En línea]. Disponible en: <http://linkinghub.elsevier.com/retrieve/pii/S1877705812009034> 12, 17, 42
- [30] I. Janiszewski, H. Meuth, and B. Hoppe, “FPGA-Efficient Hybrid LUT/CORDIC Architecture,” in *Field Programmable Logic and Application*, ser. Lecture Notes in Computer Science, J. Becker, M. Platzner, and S. Vernalde, Eds. Springer Berlin Heidelberg, Aug. 2004, no. 3203, pp. 933–937, doi: 10.1007/978-3-540-30117-2_102. [En línea]. Disponible en: http://link.springer.com/chapter/10.1007/978-3-540-30117-2_102 12
- [31] J. E. Volder, “The CORDIC Trigonometric Computing Technique,” *IRE Transactions on Electronic Computers*, vol. EC-8, no. 3, pp. 330–334, Sep. 1959. 12
- [32] T. Adiono and R. S. Purba, “Scalable pipelined CORDIC architecture design and implementation in FPGA,” in *2009 International Conference on Electrical Engineering and Informatics*, vol. 02, Aug. 2009, pp. 646–649. 12, 13, 15
- [33] D. R. Llamocca-Obregón and C. P. Agurto-Ríos, “A fixed-point implementation of the expanded hyperbolic CORDIC algorithm,” *Latin American applied research*, vol. 37, no. 1, pp. 83–91, Jan. 2007. [En línea]. Disponible en: http://www.scielo.org.ar/scielo.php?script=sci_abstract&pid=S0327-07932007000100016&lng=es&nrm=iso&tlng=en 17, 42
- [34] I. Mansour and O. Bataineh, “A Dual Fixed Point implementation of Expanded Hyperbolic Cordic Algorithm.” [En línea]. Disponible en: <http://www.secs.oakland.edu/~llamocca/Courses/ECE495/FinalProject/Group3.hypcordic.dfx.pdf> 17

- [35] D. Llamocca and C. Agurto, “A fixed-point implementation of the natural logarithm based on a expanded hyperbolic CORDIC algorithm,” in *ResearchGate*, Mar. 2006. [En línea]. Disponible en: https://www.researchgate.net/publication/230668515_A_fixed-point_implementation_of_the_natural_logarithm_based_on_a_expanded_hyperbolic_CORDIC_algorithm 18, 42
- [36] X. Hu, R. G. Harber, and S. C. Bass, “Expanding the range of convergence of the CORDIC algorithm,” *IEEE Transactions on Computers*, vol. 40, no. 1, pp. 13–21, Jan. 1991. 18
- [37] Y. Mahmoud, M. Abdelwahed, and E. F. El-Saadany, “An Enhanced MPPT Method Combining Model-Based and Heuristic Techniques,” *IEEE Transactions on Sustainable Energy*, vol. 7, no. 2, pp. 576–585, Apr. 2016. 20
- [38] T. Logeswaran and A. SenthilKumar, “A Review of Maximum Power Point Tracking Algorithms for Photovoltaic Systems under Uniform and Non-uniform Irradiances,” *Energy Procedia*, vol. 54, pp. 228–235, Jan. 2014. [En línea]. Disponible en: <http://www.sciencedirect.com/science/article/pii/S1876610214011436> 20
- [39] G. Fernández García, “Estructuras espacio-temporales en sistemas de reacción-difusión-advención,” 2008. [En línea]. Disponible en: http://www.usc.es/export/sites/default/en/investigacion/grupos/gfnl/documents/thesis/tesis_Guillermo.pdf 24
- [40] “Qualitative Analysis.” [En línea]. Disponible en: <http://www.sosmath.com/diffeq/system/qualitative/qualitative.html> 24
- [41] E. Duman, “FPGA Modules for Conversions between Fixed and Floating-point in Quartus-II Environment,” *ResearchGate*, vol. 10, no. 06, Jan. 2010. [En línea]. Disponible en: https://www.researchgate.net/publication/270157113_FPGA_Modules_for_Conversions_between_Fixed_and_Floating-point_in_Quartus-II_Environment 33
- [42] “NexysTM FPGA Board Reference Manual.” [En línea]. Disponible en: https://reference.digilentinc.com/_media/nexys:nexys4:nexys4.rm.pdf 59