

Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica



**Unidad de linealización y normalización para un estimador de
parámetros de uso en un sistema de optimización de energía en
paneles fotovoltaicos**

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

Adrián Ignacio Cervantes Segura

Borrador de 10 de mayo de 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.

Adrián Ignacio Cervantes Segura

Cartago, 10 de mayo de 2016

Céd: 1-1508-0317

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Proyecto de Graduación
Tribunal Evaluador

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

M.Sc. Leonardo Rivas Arce
Profesora Lector

Ing. Leonardo Sandoval
Profesor Lector

Dr. Alfonso Chacón Rodríguez
Profesor Asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica.

Cartago, 25 de marzo de 2016

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Proyecto de Graduación
Tribunal Evaluador
Acta de Evaluación

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Estudiante: Adrián Ignacio Cervantes Segura

Nombre del Proyecto: *Unidad de linealización y normalización para un estimador de parámetros de uso en un sistema de optimización de energía en paneles fotovoltaicos*

Miembros del Tribunal

M.Sc. Leonardo Rivas Arce
Profesora Lector

Ing. Leonardo Sandoval
Profesor Lector

Dr. Alfonso Chacón Rodríguez
Profesor Asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica.

Nota final del Proyecto de Graduación: _____

Cartago, 25 de marzo de 2016

Resumen

En esta tesis se presenta el diseño de un linealizador de corriente descrito en lenguaje VHDL, basado en el estándar para aritmética de punto flotante de IEEE 754 para microprocesadores, para el cual se utiliza el formato binario de precisión simple de 32 bits, utilizando el algoritmo de CORDIC para realizar la operación logaritmo natural. Por otro lado se realiza un convertidor de formato IEEE 754 punto flotante a punto fijo, ambos en 32 bits, por último se normaliza el dato en punto fijo, esta normalización se aplica tanto en corriente como en tensión. Estos circuitos serán utilizados en un sistema de aumento de eficiencia para suministros de paneles fotovoltaicos. Este linealizador cuenta con la capacidad para realizar operaciones aritméticas de punto flotante, lo cual es fundamental para una mejor precisión y desempeño del sistema en el procesamiento de los datos, ha sido diseñado considerando los parámetros de velocidad, área utilizada dentro de la FPGA y consumo de potencia estimada, además el circuito ha sido sintetizado y simulado sobre la FPGA Virtex® 7 de la familia Xilinx®. El circuito se interconectó en un diagrama esquemático principal para mayor facilidad de incorporación de los bloques de control, entradas y salidas.

Palabras clave: Aritmética Binaria, Formato IEEE 754, Punto fijo, FPGA, VHDL.

Abstract

This thesis presents the design of a current linearizer described in VHDL language based on the standard for Arithmetic floating Point IEEE-754 for microprocessors and the binary format used is single-precision 32-bit, using CORDIC algorithm to perform the operation. On the other hand a converter IEEE 754 floating point to fixed point, both in 32 bits, finally the data is normalized in fixed point, this normalization is applied to both current and voltage. These circuits will be used in a system for increased efficiency of supplies photovoltaic panels. This linearizer has the ability to perform arithmetic floating point operations, which is critical for better accuracy and performance of the system in data processing, it has been designed considering the parameters of speed, area used within the FPGA and consumption estimated power addition the circuit has been synthesized and simulated on FPGA Xilinx® Virtex® 7 family. The circuit has been interconnected in a main schematic diagram for easy incorporation of control blocks, inputs, outputs.

Keywords: Binary arithmetic, IEEE 754 Format, Fixed point, FPGA, VHDL.

a mis queridos padres

Agradecimientos

En primer lugar a Dios, quién me ha dado sabiduría y ha sido un pilar importante a lo largo de esta carrera.

Al Ing. Alfonso Chacón Rodríguez, quién ha sido un gran apoyo en el desarrollo del proyecto y en cursos a lo largo de la carrera, por los consejos, paciencia, conocimiento compartido y su nivel de profesionalismo.

A mi familia, por apoyarme a lo largo de toda mi educación, por la ayuda y por los grandes sacrificios que ha hecho para poder llegar donde estoy hoy, por los valores y por toda la ayuda que me han dado en todos estos años.

Finalmente, agradezco a las personas que de una u otra forma me han apoyado a lo largo del proceso.

Adrián Ignacio Cervantes Segura

Cartago, 10 de mayo de 2016

Índice general

Índice de figuras	iii
Índice de tablas	v
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 Enfoque de la solución	2
1.5 Meta	4
1.6 Objetivos y estructura	4
1.6.1 Objetivo general	4
1.6.2 Objetivos específicos	4
1.6.3 Estructura	5
2 Marco teórico	7
2.1 Descripción	7
2.2 Panel Fotovoltaico	7
2.2.1 Curvas Corriente-Tensión(I-V) para un PV	8
2.2.2 Modelos del panel fotovoltaico	8
2.3 Algoritmo de CORDIC	11
2.3.1 Sistema de coordenadas hiperbólico	12
2.3.2 Logaritmo natural utilizando el algoritmo hiperbólico de CORDIC	12
2.3.3 Exponencial utilizando el algoritmo hiperbólico de CORDIC	13
2.4 Punto flotante	13
2.5 Punto fijo	14
3 Sistema de linealización	17
3.1 Algoritmo de CORDIC en software	17
3.2 Sistema linealizador con el algoritmo de CORDIC	19
3.3 Coprocesador CORDIC	20
3.4 Sistema de control para el coprocesador CORDIC (FSM)	24
3.5 Algoritmo de CORDIC en Verilog	25
3.6 Simulación del circuito linealizador con algoritmo de CORDIC	25

3.6.1	Resultados de la simulación del rango de convergencia del circuito linealizador CORDIC	26
4	Sistema de conversión punto flotante a punto fijo y normalización	31
4.1	Sistema de conversión y normalización	31
4.2	Convertidor punto flotante - punto fijo y normalizador	33
4.3	Control para el convertidor punto flotante - punto fijo y normalizador	36
4.4	Sistema de conversión-normalización en verilog	37
4.5	Resultados de la simulación del circuito convertidor-normalizador	37
5	Sistema de linealización, conversión punto flotante a punto fijo y nor- malización	41
5.1	Circuito para realizar las pruebas en Nexys-4	43
5.2	Simulación del sistema de linealización, conversión punto flotante a punto fijo y normalización implementado en hardware	44
5.3	Resultados del sistema de linealización, conversión punto flotante a punto fijo y normalización implementado y verificado en hardware por medio de una FPGA nexys-4	45
5.3.1	Sistema de linealización, conversión y normalización para la corri- ente i_{pv} con 8 iteraciones implementado en una FPGA nexys-4 . .	45
5.3.2	Sistema de linealización, conversión y normalización para la corri- ente i_{pv} con 12 iteraciones implementado en una FPGA nexys-4 . .	47
5.3.3	Sistema de linealización, conversión y normalización para la corri- ente i_{pv} con 15 iteraciones implementado en una FPGA nexys-4 . .	48
5.4	Recursos utilizados	50
5.5	Reporte de tiempos	50
5.6	Consumo de potencia	51
6	Conclusiones y recomendaciones	53
6.1	Conclusiones	53
6.2	Recomendaciones	53
7	bibliográficas	55

Índice de figuras

1.1	Diagrama de solución para el sistema de aumento de eficiencia de paneles fotovoltaicos	3
1.2	Diagrama de solución para el sistema de linealización-normalización.	3
2.1	Curva Corriente(A)-Tensión(V)	8
2.2	Modelo simple ideal para un PV	9
2.3	Modelo con perdidas para un PV	9
2.4	Formato IEEE 754 punto flotante	14
2.5	Formato para un numero en punto fijo	14
3.1	Algoritmo de CORDIC en Python	18
3.2	Bloque principal: algoritmo de CORDIC en hardware	19
3.3	Coprocesador CORDIC y Control	19
3.4	Coprocesador segmentado para el cálculo de una función logarítmica con el algoritmo de CORDIC en hardware	21
3.5	Signo δ para cada iteración componente X_i , Y_i , Z_i	22
3.6	Maquina de estados finitos para la arquitectura de CORDIC	24
3.7	Simulación del linealizador en implementado en Verilog	25
3.8	Datos de la simulación post-síntesis del linealizador para un rango de convergencia CORDIC con 8 iteraciones	27
3.9	Porcentaje de error rango de convergencia circuito CORDIC con 8 iteraciones simulación post-síntesis	27
3.10	Datos de la simulacion post-síntesis del linealizador para un rango de convergencia CORDIC con 12 iteraciones	28
3.11	Porcentaje de error rango de convergencia circuito CORDIC con 12 iteraciones simulación post-síntesis	28
3.12	Datos de la simulación post-síntesis del linealizador para un rango de convergencia CORDIC con 15 iteraciones	29
3.13	Porcentaje de error rango de convergencia circuito CORDIC con 15 iteraciones simulación post-síntesis	29
4.1	Bloque general del convertidor punto flotante a punto fijo y normalizador .	31
4.2	Sistema de conversión, normalización, señales de datos y control	32
4.3	Proceso general de conversión y normalización	33
4.4	Circuito de conversión punto flotante a punto fijo y normalización	34

4.5	Maquina de estados finita para el convetidor-normalizador	36
4.6	Simulación del circuito de conversión y normalización	37
4.7	Comparación entre la conversión-normalización de corriente i_{pv} teórica y la simulación post-síntesis del circuito	38
4.8	Porcentaje de Error entre la conversión-normalización de corriente i_{pv} teórica y del circuito	38
4.9	Comparación entre la conversión-normalización de tensión V_{pv} teórica y la simulación post-síntesis del circuito	39
4.10	Porcentaje de error entre la conversión-normalización de tensión V_{pv} teórica y del circuito	39
5.1	Sistema de linealización, conversión y normalización de corriente y tensión para un panel fotovoltaico	41
5.2	Sistema de linealización-conversión-normalización de corriente y sistema de conversión- normalización de tensión para un panel fotovoltaico	42
5.3	Circuito de prueba para enviar datos por medio de UART desde la nexys-4 hacia un computador	43
5.4	Simulación del sistema de linealización-conversión-normalización de corriente y sistema de conversión- normalización de tensión para un panel fotovoltaico	44
5.5	Comparación entre el valor teórico y el valor obtenido del circuito de linealización-conversión-normalización de corriente con 8 iteraciones en el algoritmo de CORDIC del linealizador	46
5.6	Porcentaje de error entre el valor teórico y el valor obtenido del circuito de linealización-conversión-normalización de corriente con 8 iteraciones en el algoritmo de CORDIC del linealizador	46
5.7	Comparación entre el valor teórico y el valor obtenido del circuito de linealización-conversión-normalización de corriente con 12 iteraciones en el algoritmo de CORDIC del linealizador	47
5.8	Porcentaje de error entre el valor teórico y el valor obtenido del circuito de linealización-conversión-normalización de corriente con 12 iteraciones en el algoritmo de CORDIC del linealizador	48
5.9	Comparación entre el valor teórico y el valor obtenido del circuito de linealización-conversión-normalización de corriente con 15 iteraciones en el algoritmo de CORDIC del linealizador	49
5.10	Porcentaje de error entre el valor teórico y el valor obtenido del circuito de linealización-conversión-normalización de corriente con 15 iteraciones en el algoritmo de CORDIC del linealizador	49

Índice de tablas

2.1	Modelos para un PV	10
2.2	Sistema de coordenadas unificado (CORDIC)	12
3.1	Tabla de signo δ para cada iteración componente X_i , Y_i , Z_i	22
5.1	Resumen del reporte post implementación del uso de dispositivos generado por la herramienta Vivado.	50
5.2	Resumen del reporte post implementación de tiempos del circuito completo, a partir de la herramienta Vivado.	51
5.3	Resumen del reporte post implementación de la potencia estática, dinámica y total, de la herramienta Vivado.	51
5.4	Resumen del reporte generado por la herramienta Vivado que indica el consumo de potencia de diversos elementos.	51

Capítulo 1

Introducción

1.1 Entorno del proyecto

Hoy en día es cada vez más común el tema de las energías limpias, dentro de estas: eólica y solar. La instalación de suministros con paneles fotovoltaicos ha llegado a ser una tendencia en Costa Rica, estos son sistemas de autoconsumo de energía, y a su vez se utilizan para vender energía a otras empresas.

Un sistema de abastecimiento de energía solar, requiere de paneles solares, acumuladores de energía, inversores (conversión de corriente continua en corriente alterna) y reguladores, sin embargo actualmente las redes de suministros no cuentan con un sistema que les regule la tensión que se necesita para ubicarse en el punto de operación de potencia máxima, esto debido a que la corriente y la tensión del panel están variando constantemente 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 varía, lo que se busca es un punto de tensión donde se obtenga la máxima potencia.

Debido a la importancia de la eficiencia energética en el campo de la electrónica, se desarrolló parte de un sistema en donde se puede aprovechar de una mejor manera la energía, este tema es de suma importancia en la producción de energía, principalmente en los paneles fotovoltaicos, se debe aprovechar las mejores condiciones ambientales y poder acoplar el sistema para una máxima producción de energía, el desarrollo de este proyecto se basa en aumentar la eficiencia del sistema completo para un panel previamente escogido, se utilizará un panel modelo KS10T de la empresa KYOCERA SOLAR, para esto se realiza una realimentación con un dispositivo que regula la tensión máxima que debe tener el panel.

El proyecto linealizador-normalizador se realizó en el Instituto Tecnológico de Costa Rica, Escuela de Ingeniería Electrónica, con el coordinador del SESLab Dr. Carlos Meza, y el coordinador del DCILab Dr. Alfonso Chacón estos laboratorios se encargan de presentar propuestas de sistemas electrónicos de gran utilidad para el desarrollo tecnológico y sostenibilidad, de manera que los recursos sean aprovechados de la mejor forma, brindan

soluciones innovadoras con energías limpias, enfatizándose en el uso de paneles solares, motores eléctricos, circuitos integrados, diseño digital, entre otros.

1.2 Descripción del problema y justificación

Anteriormente se realizó un estimador de parámetros por parte de Clevis Lozano estudiante del Instituto Tecnológico de Costa Rica, sin embargo este recibe en la entrada cuantificaciones lineales para calcular los parámetros requeridos, la curva característica $i_{pv} - V_{pv}$ de una celda solar no tiene un comportamiento lineal, de manera que si se requiere estimar parámetros a partir de la corriente y tensión de este, se deben linealizar-normalizar las entradas y desnormalizar-deslinealizar las salidas. Para el modelo del panel se tienen cuatro tipos de configuraciones desde la más simple a la más compleja tomando en cuenta las perdidas resistivas de las celdas, paralelas y serie.

El coprocesador numérico a desarrollar, debía satisfacer los siguientes requerimientos:

- Se debe basar en el formato IEEE 754, el cual es un estándar en coma flotante, para realizar operaciones aritméticas.
- Utilizar una arquitectura de 32 bits.
- Utilizar Verilog como lenguaje de descripción de hardware.
- Optimizar las unidades para requerir la menor cantidad de recursos.

1.3 Síntesis del problema

¿Cómo realizar la linealización y normalización en formato IEEE 754 para el sistema de optimización de paneles fotovoltaicos?

1.4 Enfoque de la solución

Primeramente se realizará un proceso de recopilación de información dentro de temas como: curvas y modelo de un PV, ecuaciones características de los PV, algoritmo de CORDIC, estándar IEEE 754 y operaciones en punto fijo. Posteriormente, se utilizará este estándar para iniciar el diseño del linealizador y el normalizador del sistema general del panel fotovoltaico, este se puede observar en la figura 1.1.

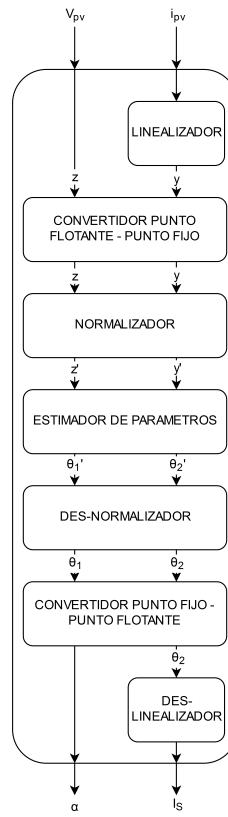


Figura 1.1: Diagrama de solución para el sistema de aumento de eficiencia de paneles fotovoltaicos

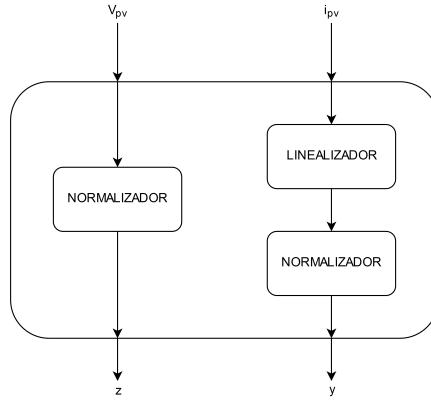


Figura 1.2: Diagrama de solución para el sistema de de linealización-normalización.

Para el diseño del los circuitos linealización-normalización de la figura 1.2 se deberá utilizar el diseño modular ya que se brinda una mejor perspectiva de los que se debe realizar, si se podrá asignar las entradas y salidas que requiere cada unidad. Una vez ejecutada la primera etapa de diseño se descompone en pequeños bloques funcionales, que se deberán diseñar en etapas e interconectarlas.

Este diseño modular, se implementará utilizando el lenguaje HDL Verilog, y se verificarán los resultados utilizando un modelo en alto nivel realizado en Python, contra los resultados obtenidos en las simulaciones Post Place & Route.

Por último, se realizarán las pruebas en una FPGA artix-7 para comprobar el funcionamiento del hardware y comparar con las simulaciones.

1.5 Meta

La meta de este proyecto es poder aprovechar de la mejor forma energía solar, de manera que se aumente la eficiencia del sistema de generación fotovoltaica de los paneles solares, así poder seguir impulsando la utilización de energías limpias que contribuyan a reducir las que son producidas por medio de hidrocarburos, que incluso de ser más caras, son mucho más dañinas para el ambiente, así poder autosatisfacer el consumo en los hogares, empresa u otros.

1.6 Objetivos y estructura

1.6.1 Objetivo general

Desarrollar una unidad de linealización y normalización para un estimador de parámetros Corriente-Tensión de un panel fotovoltaico.

1.6.2 Objetivos específicos

- Crear un circuito en formato IEEE 754, que linealice la corriente del modelo de un panel fotovoltaico, por medio de una operación logarítmica, con una corriente exponencial como parámetro de entrada y una corriente lineal ‘y’ en la salida.

Indicador: verificar mediante un programa de alto nivel la precisión del algoritmo implementado en hardware con un error menor al 5%

- Crear un circuito que convierta de punto flotante a punto fijo, la corriente lineal ‘y’ en la salida del linealizador.

Indicador: verificar mediante un programa de alto nivel la precisión del convertidor implementado en hardware con un error menor al 5%

- Crear un circuito que normalice los parámetros de corriente lineal ‘y’ y tensión lineal ‘z’, ambos en punto fijo, para las entradas del estimador.

Indicador: verificar mediante un programa de alto nivel la precisión del normalizador implementado en hardware con un error menor al 5%

1.6.3 Estructura

El capítulo 2 se describen los conceptos teóricos que se utilizaros a través del desarrollo del proyecto. En el capitulo 3 se detalla el diseño del algoritmo y control, implementación y los resultados obtenidos para el circuito de linealización. En el capítulo 4 se presentan el diseño del algoritmo y control, implementación y los resultados obtenidos para el circuito de normalización. El capitulo 5 muestra el sistema completo, junto con sus pruebas y resultados. El capítulo 6 se ofrecen conclusiones del proyecto, y recomendaciones. Finalmente, en el capitulo 7 se puede observar la bibliografía utilizada.

Capítulo 2

Marco teórico

2.1 Descripción

En un sistema de paneles fotovoltaicos es de suma importancia la eficiencia energética, para esto se debe estudiar el funcionamiento, curvas características, parámetros para poder caracterizar un panel, el modelo matemático tanto el ideal como la aproximación real, ecuaciones características obtenidas a partir de cada modelo, características del sistema que se requiere optimizar, algoritmos de calculo para realizar operaciones que se requieren en la solución, entre otros.

2.2 Panel Fotovoltaico

Existen muchos tipos de celdas solares [11], el más común se pueden describir como una junta $p-n$, por medio de la radiación electromagnética proveniente del sol que incide sobre la capa de semiconductor, esta se transforma en electricidad por un efecto fotovoltaico, en donde los fotones al tener mayor energía que la banda prohibida del semiconductor crean un par electrón-hueco, el campo eléctrico que se ejerce en la junta $p-n$ mueve los electrones (*portadores*) lo que produce una photocorriente, esta es directamente proporcional a la radiación del sol [1]. Debido al proceso de fabricación del panel, posee un comportamiento exponencial y no lineal muy similar al de un diodo [9].

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

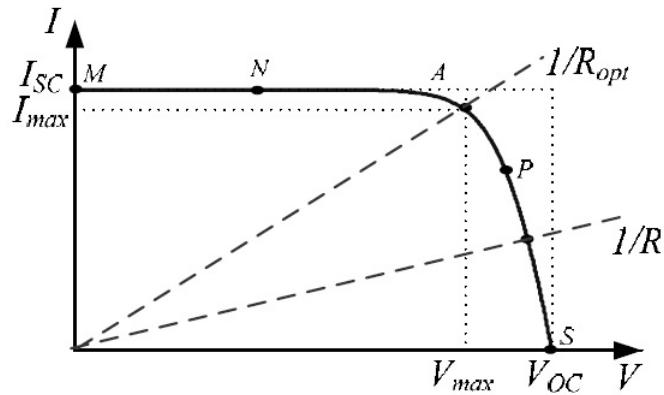


Figura 2.1: Curva Corriente(A)-Tensión(V)

La curva característica de un PV se puede obtener manteniendo fijos los parámetros de irradiancia(S) y temperatura(T) esto bajo condiciones controladas, si se tiene una carga en las terminales de salida, la potencia entregada solo dependerá del valor de la carga, de manera que si la carga es pequeña (puntos M-N) el panel se comportará como una fuente de corriente, pero si la carga es grande(puntos PS) se comportará como una fuente de tension[2].

Dentro de la caracterización de la celda se realizan varias pruebas:

- *Corriente de corto circuito I_{sc}* : Se define como el valor máximo de la corriente generada por el panel, en condiciones de cortocircuito $V = 0$.
- *Tensión de circuito abierto V_{oc}* : Se define como el valor que se tiene en la junta $p-n$ cuando se tiene una corriente generada $I = 0$.
- *Punto de máxima potencia*: se puede observar en el punto A(V_{max}, I_{max}) de la figura 2.1 donde la potencia máxima de la carga resistiva es $P_{max} = V_{max}I_{max}$

2.2.2 Modelos del panel fotovoltaico

Un panel fotovoltaico se puede modelar de manera simple (ideal), utilizando una fuente de corriente en paralelo con un diodo, la corriente de salida sera proporcional al radiación sobre la celda (foto-corriente)

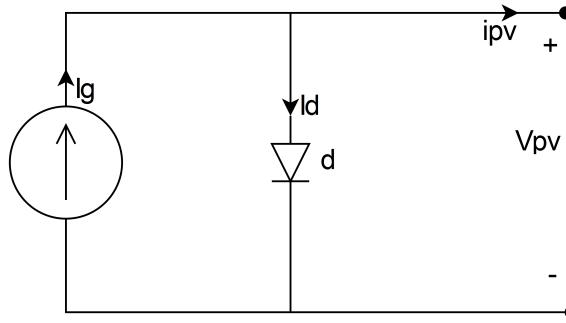


Figura 2.2: Modelo simple ideal para un PV

La figura 2.2 muestra el modelo básico, sin embargo este se puede realizar de una manera mas compleja, agregando variables para las características del panel:

- Dependencia de la Temperatura, la corriente de saturación del diodo (I_s) y la foto corriente (I_g)
- Perdidas debidas al flujo de corriente (R_s) y perdidas con referencia a tierra (R_p).
- Un parámetro n que será el numero de celdas en análisis.

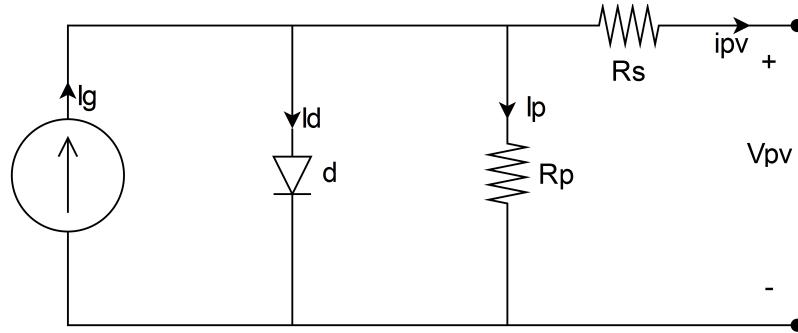


Figura 2.3: Modelo con perdidas para un PV

A partir del modelo con perdidas se puede deducir la ecuación que describe las corrientes, como sigue a continuación[4]:

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

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

Despejando i_{pv}

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

En general, la corriente que fluye por las terminales de un generador fotovoltaico está determinado por tres funciones de corriente:

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

Para obtener un modelo del comportamiento estático del generador fotovoltaico se supondrá lo siguiente:

- 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}

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

Según se definan las funciones i_{pv} e i_d se obtendrán modelos con complejidad y precisiones distintas, como los siguientes casos:

Tabla 2.1: Modelos para un PV

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:

$$i_{pv}(V_{pv}) = G_p V_{pv} + G_p i_{pv} R_s \quad (2.5)$$

$$i_d(V_{pv}) = I_s(T) e^{\frac{V_{pv}}{v_t}} e^{\frac{i_{pv} R_s}{v_t}} - I_s(T) \quad (2.6)$$

De esta forma el modelo general del comportamiento estático de un generador FV también se puede representar de la siguiente manera:

$$i_{pv} = KS - G_p V_{pv} + I_s(T) - G_p i_{pv} R_s - I_s(T) e^{\frac{V_{pv}}{v_t}} e^{\frac{i_{pv} R_s}{v_t}} \quad (2.7)$$

$$I_s(T) e^{\frac{V_{pv}}{v_t}} e^{\frac{i_{pv} R_s}{v_t}} = KS - G_p V_{pv} + I_s(T) - G_p i_{pv} R_s - i_{pv} \quad (2.8)$$

La ecuación 2.8 es no lineal, aplicando una linealización, se tiene [3]:

$$y = \ln(KS - G_p V_{pv} + I_s(T) - G_p i_{pv} R_s - i_{pv}) \quad (2.9)$$

si $I_g = KS \gg I_s$

$$y = \ln(KS - G_p V_{pv} - G_p i_{pv} R_s - i_{pv}) \quad (2.10)$$

$$z = V_{pv} + i_{pv} R_s \quad (2.11)$$

posteriormente a un proceso de calculo de parámetros se tiene:

$$\theta_1 = \ln(I_s(T)) \quad (2.12)$$

$$\theta_2 = \alpha \quad (2.13)$$

2.3 Algoritmo de CORDIC

El algoritmo *Coordinate Rotational Digital Computer* (CORDIC) es un método numérico en donde se realiza cierto numero de iteraciones para encontrar el valor deseado según sea la función en calculo, este algoritmo es utilizado para implementar funciones trigonométricas, logarítmicas y exponenciales. La facilidad de implementación, hace que sea uno de los algoritmos mas utilizados en el ámbito de la electrónica digital, CORDIC utiliza desplazamientos, sumas, restas y tablas look-up con valores previamente precargados en una memoria ROM, estos valores dependerán de la operación en calculo, se puede utilizar el método circular, lineal e hiperbólico.

Las ecuaciones generales para el algoritmo de CORDIC se definen como:

$$X_{i+1} = X_i - m d_i 2^{-i} Y_i \quad (2.14)$$

$$Y_{i+1} = Y_i - d_i 2^{-i} X_i \quad (2.15)$$

$$Z_{i+1} = Z_i - d_i e(i) \quad (2.16)$$

donde $e(i)$ se muestra en la tabla 2.2 [5] según sea el caso:

Tabla 2.2: Sistema de coordenadas unificado (CORDIC)

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

2.3.1 Sistema de coordenadas hiperbólico

Para el calculo de algunas funciones que no son tan directas con el algoritmo, se utilizan identidades [6] para el calculo como sigue:

$$\tan z = \frac{\sin z}{\cos z} \quad (2.17)$$

$$\tanh z = \frac{\sinh z}{\cosh z} \quad (2.18)$$

$$\exp z = \sinh z + \cosh z \quad (2.19)$$

$$\ln \omega = 2 \tanh^{-1} \left(\frac{y}{x} \right) \quad (2.20)$$

donde:

$$x = \omega + 1 \quad (2.21)$$

$$y = \omega - 1 \quad (2.22)$$

2.3.2 Logaritmo natural utilizando el algoritmo hiperbólico de CORDIC

Para un $\ln(\omega)$ con el algoritmo de CORDIC, se utiliza el modo hiperbolico [12] para esto se debe calcular primeramente la $\tanh^{-1} \left(\frac{y}{x} \right)$ con las siguientes ecuaciones:

$$X_{i+1} = X_i + d_i 2^{-i} Y_i \quad (2.23)$$

$$Y_{i+1} = Y_i + d_i 2^{-i} X_i \quad (2.24)$$

$$Z_{i+1} = Z_i - d_i \tanh^{-1}(2^{-i}) \quad (2.25)$$

donde i es el indice de cada iteración, las iteraciones 4, 13, 40, ..., k , $3k+1$ se deberán repetir para garantizar la convergencia. d_i es el signo de Y_i invertido, es decir el cuando el signo de Y_i es negativo, d_i será positivo y viceversa.

Utilizando la ecuación 2.20 se definen los valores de entrada para las ecuaciones anteriores $X_0 = \omega + 1$, $Y_0 = \omega - 1$ y $Z_0 = 0$.

Cabe destacar que el rango de convergencia para este algoritmo [7] se puede definir como:

$$0.106843 \leq \omega \leq 9.35947 \quad (2.26)$$

donde ω es el valor del argumento del logaritmo natural.

El resultado final de Z_i contiene el valor de $\tanh^{-1}\left(\frac{y}{x}\right)$, sin embargo se debe multiplicar por un factor de 2 para completar el calculo del logaritmo natural, segun la identidad de la ecuación 2.20

2.3.3 Exponencial utilizando el algoritmo hiperbólico de CORDIC

Para una función $e^{(\omega)}$ con el algoritmo de CORDIC, se debe utilizar las ecuaciones 2.23, 2.24 y 2.25 de manera iterativa, donde el valor final de X y Y , son el resultado para $\cosh(\omega)$ y $\sinh(\omega)$ respectivamente. se debe tomar en cuenta la repeticion de las iteraciones (i) 4, 13, 40, ..., k , $3k+1$ para garantizar la convergencia dando una mejor precision en el calculo.

Los valores iniciales se definen como *constantes* $X_0 = 1.20753406$, $Y_0 = 0$ y $Z_0 = \omega$ donde ω es el valor del argumento que se desea calcular y d_i es el signo de Z_i .

Cabe destacar que el rango de convergencia para este algoritmo se puede definir como:

$$0 \leq \omega \leq 1 \quad (2.27)$$

El valor final del calculo se obtiene mediante una suma con la identidad de la ecuación 2.19.

2.4 Punto flotante

La codificación para el formato punto flotante se realiza mediante el estándar IEEE 754, este requiere de tres campos en la palabra [10]:

- Signo
- Exponente
- Mantisa

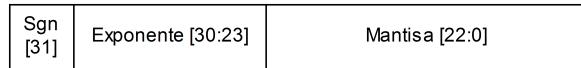


Figura 2.4: Formato IEEE 754 punto flotante

para el formato *IEEE 754* figura 2.4 de 32-bits [8] se asigna:

- 1 Bit para signo
- 8 Bits para exponente
- 23 Bits para mantisa

Donde el bit de signo representa un numero positivo si este es 0, de manera contraria si es 1 representa un numero negativo.

El exponente puede representar 256 (8-bits) valores sin embargo se tiene:

- $[0 - 127]$ exponentes negativos
- $[128 - 255]$ exponentes positivos

De esta manera para convertir el un exponente positivo en el valor correspondiente en punto flotante se debe sumar el valor del exponente como sigue $exp_{float} = 127 + exp$, por otro lado si se desea pasar de un valor decimal a punto flotante se deben realizar los siguientes pasos:

- Representar el signo con su debido bit
- Conversión decimal a binario punto fijo
- Conversión binario a notación científica
- Agrupar en signo, exponente y mantisa

2.5 Punto fijo

La representación de un numero en punto fijo cuenta con tres partes fundamentales:

- Signo
- Parte entera
- Parte fraccionaria



Figura 2.5: Formato para un numero en punto fijo

En la aritmética simple se utiliza se utilizan los signo +/- para saber si un numero es positivo o negativo, sin embargo por las limitaciones de una computadora esto no se puede hacer, debido a que esta procesan todos los datos representados por palabras compuestas por ceros '0' y unos '1', en consecuencia para representar el signo se utiliza un bit de signo, si el numero es positivo el bit de signo sera un 0 y si el numero es negativo sera un 1 [13].

La parte entera es un conjunto de bits que puede representar un numero en un rango entre 0 y 2^n , donde n es el numero de bits asignado.

La parte fraccionaria se compone de una serie de bits, 2^{-n} , en esta el numero de bits aumenta si se requiere de mucha precisión.

Si se requiere de una representación de un numero negativo en punto fijo, se debe utilizar el complemento a 2 [14], este toma el numero positivo y le aplica un inversión a todos los bits, finalmente se le suma 1 a la inversión, para obtener el numero negativo, así utilizando el bit de signo descrito anteriormente.

Capítulo 3

Sistema de linealización

Para realizar la linealización de la expresión exponencial de la corriente i_{pv} del modelo del panel, se implementa una función logarítmica, dentro de los algoritmos de cálculo se tiene el de CORDIC, este permite realizar una aproximación de la función de manera recursiva mediante cierta cantidad de iteraciones, para el caso del linealizador se utiliza el método hiperbólico para realizar el cálculo de la operación logaritmo natural, este algoritmo requiere de una tabla (Look-up table) con valores pre-cargados.

El rango de convergencia para este algoritmo es de $0.106843 < T < 9.35947$ donde T es el argumento del logaritmo natural, el valor máximo de T para el panel previamente escogido de $0.58A$, esta es la corriente en condiciones máximas para el panel, debido a esto se puede desplazar el intervalo que se tiene para los argumentos de la función, este se divide entre una constante $C = 16$ y se desplaza $0.00667769 < T < 0.58496687$, esto se realizó debido a que se pueden dar valores de corriente mas bajos que $0.106843A$. Esta constante C se debe compensar en el logaritmo, y se logra con la siguiente igualdad:

$$\ln(T) = \ln(16T) - \ln(16) \quad (3.1)$$

3.1 Algoritmo de CORDIC en software

Para comprobar el debido funcionamiento del este algoritmo se crea un programa de alto nivel en Python.

```

def CordicLn(T): #Algoritmo de cordic para resolver un Ln

    #Condiciones iniciales
    Z_ant=0
    X_ant=T+1
    Y_ant=T-1

    #Look-up table
    Bm=[0.54930614, 0.25541281, 0.12565721, 0.06258157, 0.06258157, 0.03126
    i=0

    #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**-11,2**-12,2**-13,2**-14,2**-15,2**-16,2**-17,2**-18,2**-19,2**-20,2**-21,2**-22,2**-23]

    while i<23:

        Dm= -Y_ant/(abs(Y_ant))      # Signo de Y
        Z_act= Z_ant + Dm*-2*Bm[i]   # Calcula el valor de Z actual
        X_act= X_ant + Em[i]*Dm*Y_ant # Calcula el valor X actual
        Y_act= Y_ant + Em[i]*Dm*X_ant # Calcula el valor 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
        ln=Z_act          # Resultado para cada iteracion

    return ln #resultado final del logaritmo natural

>>>
>>> T=0.2
>>> CordicLn(16*T) - math.log(16)
-1.6094370822397814
>>> math.log(0.2)
-1.3862943611198906
>>>

```

Figura 3.1: Algoritmo de CORDIC en Python

En la figura 3.1 se observa el programa realizado para la verificación del algoritmo, para este se utilizan las ecuaciones del marco teórico descritas con anterioridad, también se comprueba el cambio en el rango de cálculo con el escalado del argumento por 16.

Para simplificar el diseño del hardware se realizan cambios en las ecuaciones originales, se cambia la resta del valor actual de Z por una suma, y se incluye el signo en la LUT de Z, el resultado final del algoritmo se debe multiplicar por 2, sin embargo este escalado se puede realizar en la LUT, esto para evitar la multiplicación, esto se comprueba en el programa de la figura 3.1.

3.2 Sistema linealizador con el algoritmo de CORDIC



Figura 3.2: Bloque principal: Algoritmo de CORDIC en hardware

La figura 3.2 contiene el bloque general del algoritmo de CORDIC, poseen 4 entradas: CLK , T , Begin.LN , RST.LN y 4 salidas: ACK.LN , RESULT , U.F , O.F

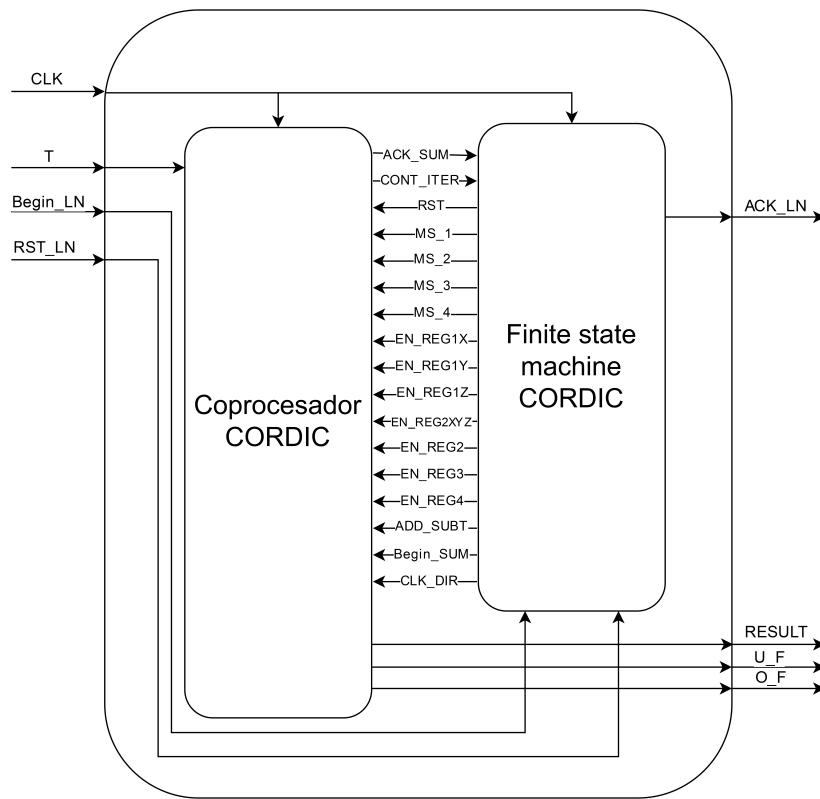


Figura 3.3: Coprocesador CORDIC y Control

El sistema de linealización de la figura 3.3 cuenta con dos módulos principales:

- *Coprocessador Cordic:* En este realizan todas las operaciones requeridas por el algoritmo, se encarga del manejo de los datos en el cálculo.

- *Control*: Este se encarga de proveer las señales de control requeridas por el coprocesador, según las condiciones que se tenga en cada estado.

Señales de datos:

- *T*: Dato de entrada, argumento del logaritmo natural.
- *RESULT*: Resultado de la operación logaritmo natural.

Señales de control:

- *CLK*: Reloj del sistema.
- *Begin_LN*: Esta se encarga de dar inicio a la operación logaritmo natural.
- *RST_LN*: Realiza un reset a la unidad CORDIC tanto para el coprocesador como para la máquina de estados.
- *ACK_LN*: Indica que el cálculo ya fue realizado.
- *Begin_SUM*: Esta se encarga de dar inicio a la unidad de suma-resta punto flotante.
- *ACK_SUM*: Indica que esta listo el calculo realizado en la unidad de suma-resta punto flotante.
- *O_F*: Indica si la suma-resta flotante realizada tiene un over-flow.
- *U_F*: Indica si la suma-resta flotante realizada tiene un under-flow.
- *CLK_DIR*: Activa el enable del contador de iteraciones.
- *CONT_ITER*: Indica el numero de iteración, para que la máquina de estados pueda detenerse en el número que se le asigne.
- *RST*: Realiza el reset de todos los registros de la unidad.
- *MS_1 , MS_2 , MS_3 , MS_4*: Realizan la selección de cada multiplexor, Mux1, Mux2, Mux3, Mux's4 respectivamente.
- *EN_REG1X , EN_REG1Y , EN_REG1Z*: Activa los enable de los registros de la primera etapa REG1X, REG1Y, REG1Z respectivamente, para almacenar datos.
- *EN_REG2XYZ*: Activa el enable del registro REG2XYZ de la segunda etapa.
- *EN_REG2*: Activa el enable del registro REG2 de la segunda etapa.
- *EN_REG3*: Activa el enable del registro REG3 del dato inicial.
- *EN_REG4*: Activa el enable del registro REG4 del dato final.

3.3 Coprocesador CORDIC

El diseño de este algoritmo se basa en una arquitectura segmentada, de manera que se almacenan y procesan varios datos a la vez, sin embargo se debe tener buena sincronización para evitar datos erróneos a través del proceso de cálculo. Por otro lado se utiliza el formato IEEE 754 con 32Bits, esto debido a que se requiere una adecuada precisión en el calculo y un menor número de iteraciones, repercutiendo en la velocidad del sistema.

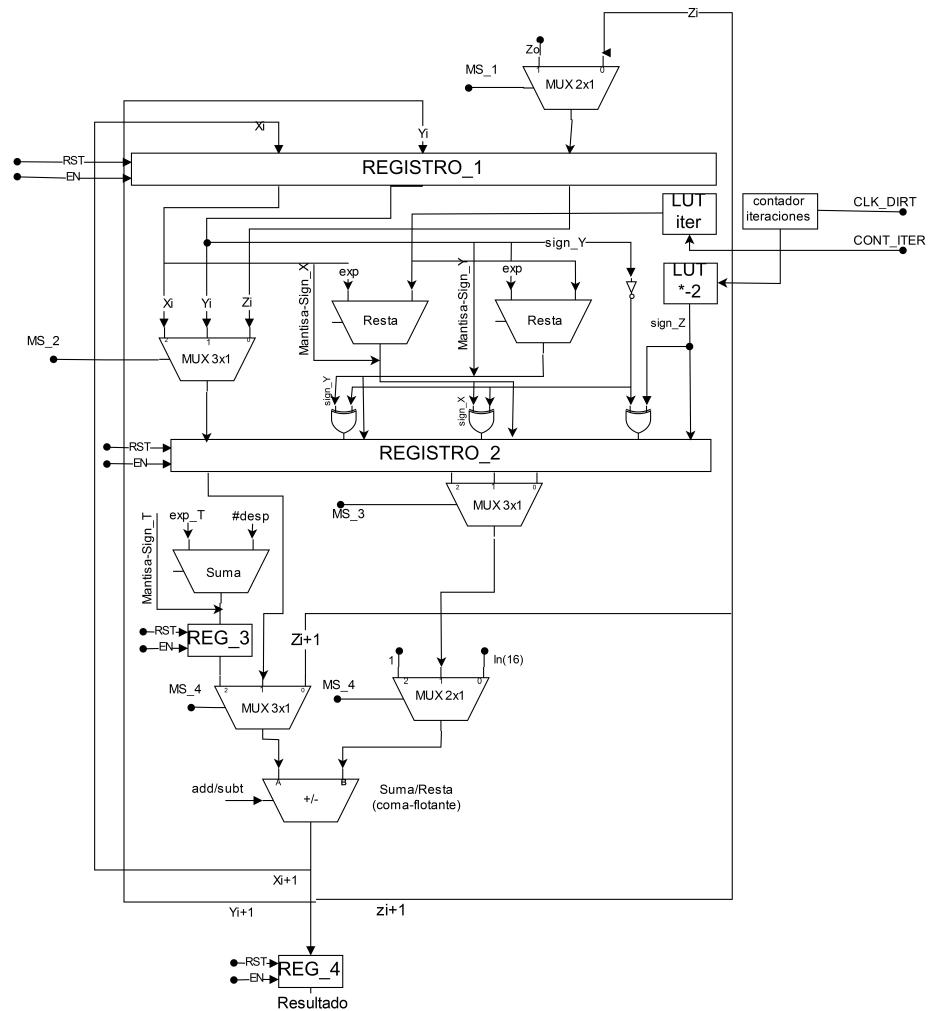


Figura 3.4: Coprocesador segmentado para el cálculo de una función logarítmica con el algoritmo de CORDIC en hardware

En la figura 3.4 se puede observar la arquitectura diseñada para el algoritmo de CORDIC, con el formato IEEE 754 en 32-Bits, se trabaja en punto flotante, este reduce el numero de iteraciones, y produce una mejor precisión tanto en las operaciones como el resultado final.

Esta etapa inicia con la carga de las condiciones iniciales, donde Z_0 contiene un valor inicial cero, para el valor inicial de X_0 y Y_0 primeramente se aplica el escalado $16*T$, este escalado se puede representar como un desplazamiento 2^{-4} , por lo tanto este movimiento en punto flotante se traduce como una suma de 4 en el exponente de T , posteriormente se realizan las siguientes operaciones en punto flotante, $X_0 = T + 1$ y $Y_0 = T - 1$, estas tres constantes dan inicio al proceso de cálculo de manera iterativa, por lo que se requiere almacenarlas en un registro en la primera etapa de segmentación (*Registro1*), los nuevos estados se deben calcular uno por uno, esto debido a que el circuito para el sumador-restador en punto flotante requiere de mucha área, este método (CORDIC) es un calculo cruzado es decir, para el próximo valor de X_i se requiere un valor de Y_0 con

un desplazamiento y un cambio de signo, y para Y_i se aplica un concepto similar con valores de X_0 , por lo tanto para el cálculo de X_i se requiere un restador en punto fijo para desplazar el valor del exponente de Y_0 , para el nuevo valor de Y_i se utiliza un restador punto fijo para el valor del exponente de X_0 y para Z_i se requiere una ROM con valores previamente cargados (*LUT*). Estas operaciones " X_i, Y_i " involucran el signo de " Y_0 " invertido.

Para el signo de las operaciones CORDIC se utiliza un circuito de comparación como se observa en la siguiente tabla:

Tabla 3.1: Tabla de signo δ para cada iteración componente X_i, Y_i, Z_i

Sign X_0	Sign Z_0	Sign Y_0	Sign $\sim Y_0$	Sign X_i	Sign Z_i	Sign Y_i
0	0	0	1	1	1	1
0	0	1	0	0	0	1
0	1	0	1	1	0	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	0	1	0	1	0	1
1	1	0	1	0	0	1
1	1	1	0	1	1	1

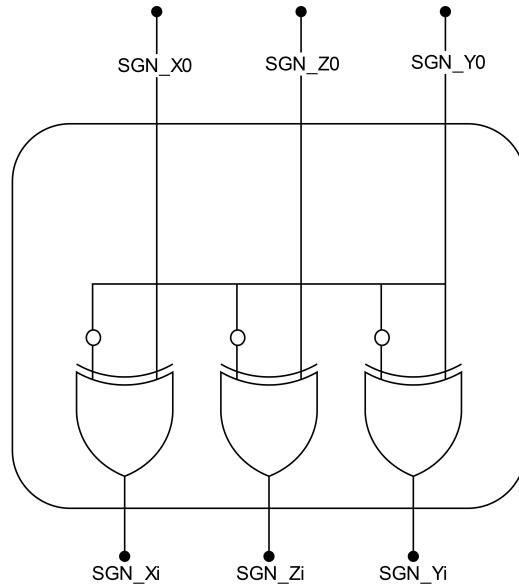


Figura 3.5: Signo δ para cada iteración componente X_i, Y_i, Z_i

A partir de la tabla 3.1 se extrae el circuito de comparación de signo de la figura 3.5, este es diseñado con compuerta *XOR* que poseen el mismo comportamiento de la tabla.

Se requiere de dos Look-up tables "LUT", los datos de cada tabla se almacenan en me-

morias ROM's, de manera que puedan ser accesados en cualquier momento que sean requeridos por medio de la dirección, dentro de las ROM's se dispone:

- *LUT_Z*: Contiene almacenados los valores de $-2\operatorname{arctanh}(2^{-i})$, para cada iteración.
- *LUT_ITER*: Esta contiene almacenados los desplazamientos que se deben realizar para cada iteración, esto debido a que las iteraciones 4 y 13 repiten desplazamientos como se menciona en el marco teórico.

El acceso a cada valor de la tablas se realiza mediante un contador de iteraciones, este indica a cada tabla la dirección que debe desplegar según el numero de iteración.

El proceso para el cálculo de las variables no se puede realizar de manera simultanea, ya que solo se cuenta con un sumador punto flotante, para esto se cuenta con las variables iniciales del registro 1 y se almacenan las variables modificadas en el *Registro 2*, la secuencia de cálculo toma primeramente los valores que se necesitan para X_i , posteriormente se realiza el cálculo y se almacena el resultado en el *Registro 1*, seguidamente se procede con el cálculo de Y_i y se almacena en el *Registro 1*, por ultimo se calcula el valor de Z_i , concluida la suma se almacena en el *Registro 1*, este proceso se hace de manera iterativa, una vez finalizada la cuenta de N iteraciones según se defina (N=numero entero), se rea-liza la resta $Z_i - \ln(16)$ para contrarrestar el efecto del escalado aplicado al argumento (T) al inicio del cálculo del logaritmo natural, por último el resultado final de Z_i contiene el valor de $\ln(T)$, este se almacena en el *Registro 4*.

3.4 Sistema de control para el coprocesador CORDIC (FSM)

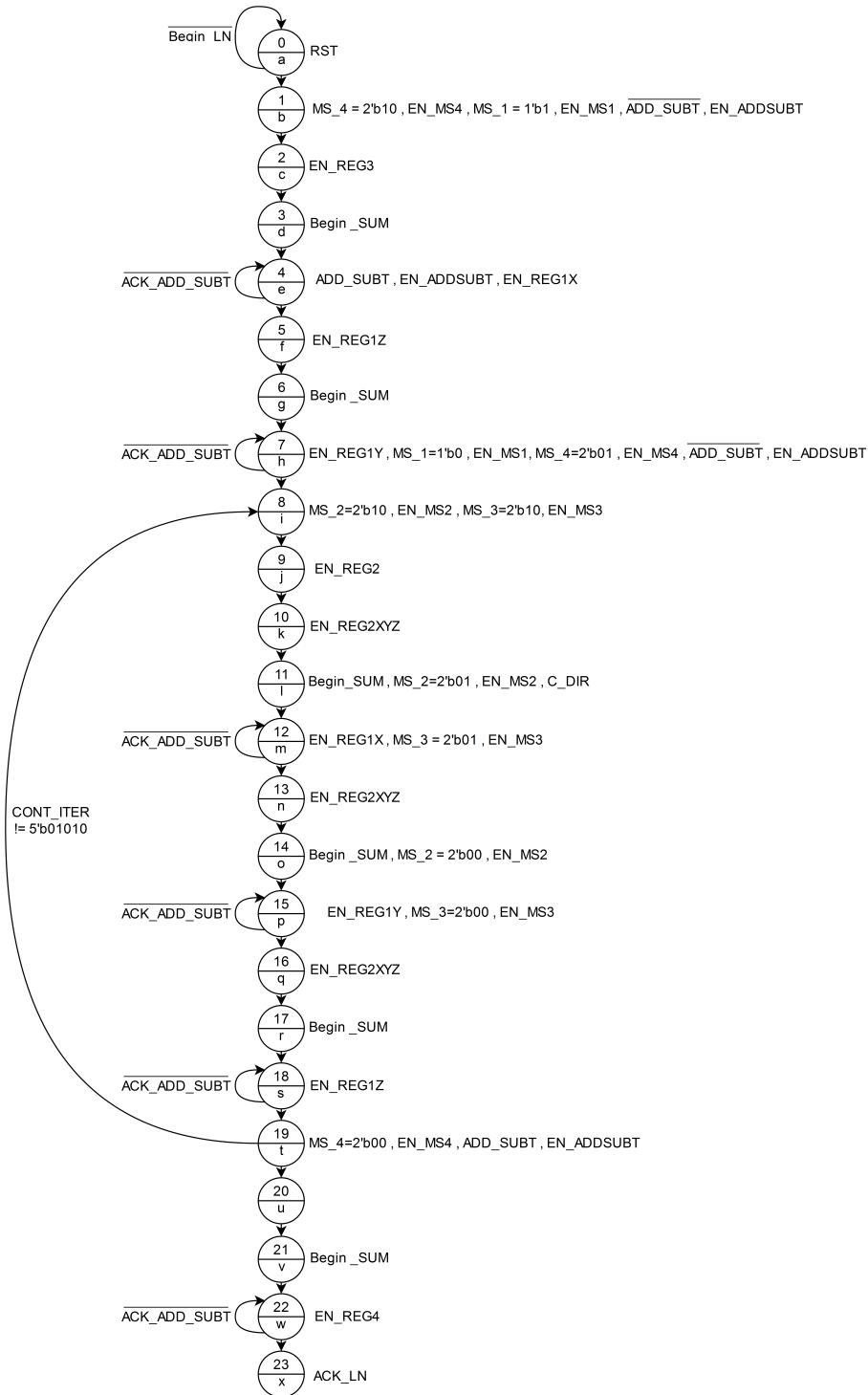


Figura 3.6: Maquina de estados finitos para la arquitectura de CORDIC

El sistema de control requiere de mucha sincronía, ya que los datos deben ser almacenados de manera correcta y estar listos cuando se requiere por otro segmento del coprocesador CORDIC, para esto se diseñó una máquina de estados finitos, donde inicialmente se calculan los valores iniciales de X_0 , Y_0 y Z_0 , posteriormente la máquina brinda las señales de control requeridas con la secuencia de cálculo de X_i , Y_i y Z_i respectivamente realizando una cuenta de iteración, se cuenta con una variable de entrada para que este control pueda saber el número de iteración en el que se encuentra, de manera que cuando se llega a la iteración N definida con anterioridad, se finaliza el cálculo.

3.5 Algoritmo de CORDIC en Verilog

Este algoritmo se implementó por medio de el lenguaje de descripción de hardware "Verilog", inicialmente se realizaron pequeños bloques pertenecientes a cada elemento requerido por la arquitectura diseñada, se vio la necesidad, por cuestión de orden, de desarrollar el coprocesador CORDIC y la unidad de control en bloques separados, de manera que se pudieran realizar pruebas sin dependencia de los bloques entre si, para una mejor depuración de errores y re-diseño. Finalmente se realizan las simulaciones al bloque completo en la figura 3.7.

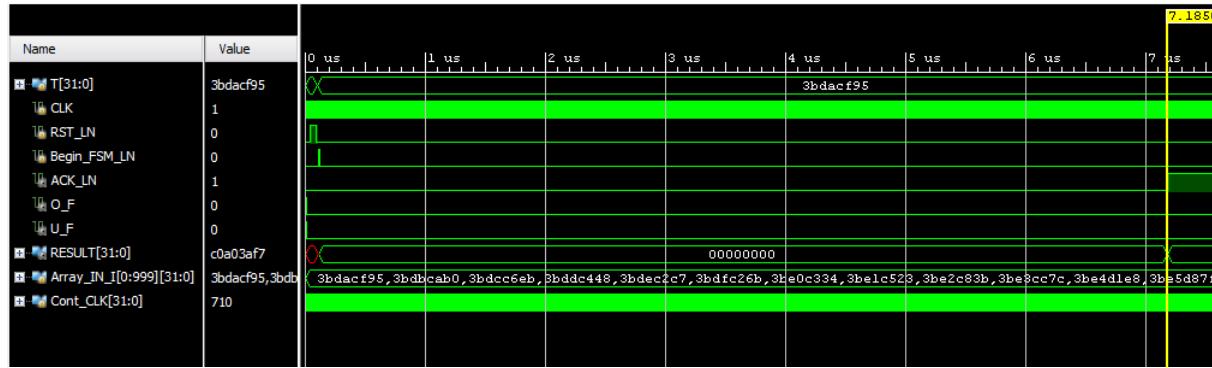


Figura 3.7: Simulación del linealizador en implementado en Verilog

3.6 Simulación del circuito linealizador con algoritmo de CORDIC

Las simulaciones de la implementación del algoritmo, se realizaron mediante mil valores de entrada, obteniendo así mil valores de salida, los cuales se comparan con los valores reales, para dicha comparación se realizaron aproximaciones con 8, 12 y 15 iteraciones.

Primeramente se realiza una simulación que comprueba el funcionamiento en el rango de operación $0.00667769 < T < 0.58496687$ y la linealización esta prueba se realiza con la siguiente función:

$$\ln(T) \quad (3.2)$$

donde el argumento T es:

$$T = e^{-x} \quad (3.3)$$

El intervalo $0,536 < x < 5,009$ se divide en mil valores, de manera que la función 4.2 devuelve mil valores, estos se almacenan en un archivo .txt de manera que se insertan en el circuito CORDIC para obtener mil valores del cálculo.

$$V_{pv} = V_{cte} + 0.3 * V_{cte} * \sin(2 * \pi * 100 * t) \quad (3.4)$$

$$i_{pv} = Ig - \frac{V_{pv}}{R_p} - Is * \left(e^{\frac{V_{pv} * \alpha}{2}} \right) \quad (3.5)$$

Seguidamente se realiza una prueba con mil valores simulando el comportamiento que tiene un PV, utilizando el modelo del panel, donde se utiliza el valor de V_{pv} para cada valor de tiempo, en la ecuación 3.5 obteniendo valores de corriente de entrada i_{pv} para el linealizador, así poder observar si se realiza la linealización en la salida.

3.6.1 Resultados de la simulación del rango de convergencia del circuito linealizador CORDIC

En la implementación de un algoritmo en hardware, es de suma importancia verificar que este funcione de manera adecuada en el rango de convergencia definido. Para la comprobación del algoritmo de la unidad del linealizador CORDIC, se realizaron una serie de pruebas, simulando con cierta cantidad de iteraciones y así poder observar cual es la mas adecuada para el cálculo requerido y su debido resultado, para esto se programó un simulación("testbench") en donde se corre una prueba con mil valores de entrada, ingresados por medio de un archivo de texto previamente editado con los datos de entrada con la función exponencial anteriormente descrita en la ecuación 4.3.

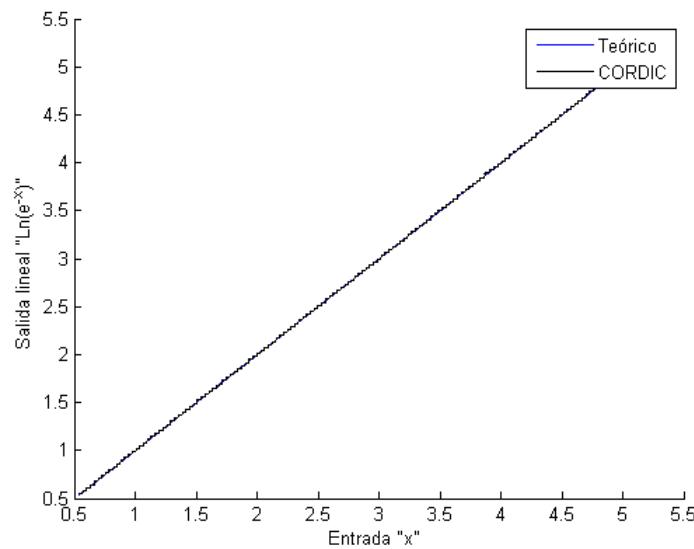


Figura 3.8: Datos de la simulación post-síntesis del linealizador para un rango de convergencia CORDIC con 8 iteraciones

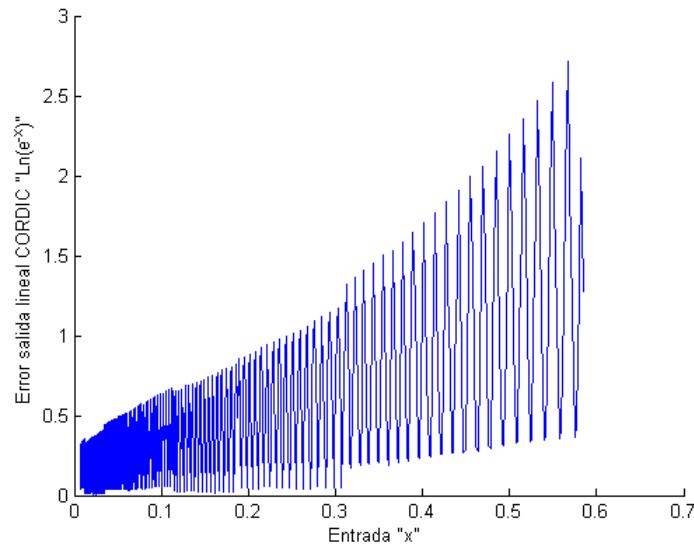


Figura 3.9: Porcentaje de error rango de convergencia circuito CORDIC con 8 iteraciones simulación post-síntesis

Primeramente se realizó una simulación con 8 iteraciones, como se muestra en la figura 3.8, para el cálculo de cada valor se requieren 460 ciclos de reloj desde el momento en se activa la señal Begin_LN hasta que se recibe la señal ACK_LN, que es donde se indica que se ha completado el cálculo, es de suma importancia realizar la comparación entre el valor teórico y el valor calculado obtenido. Para cada valor se calculó el error, estos se

pueden observar en la figura 3.9, donde el porcentaje de error máximo es de 2,72% y el porcentaje de error promedio es de 0,40%.

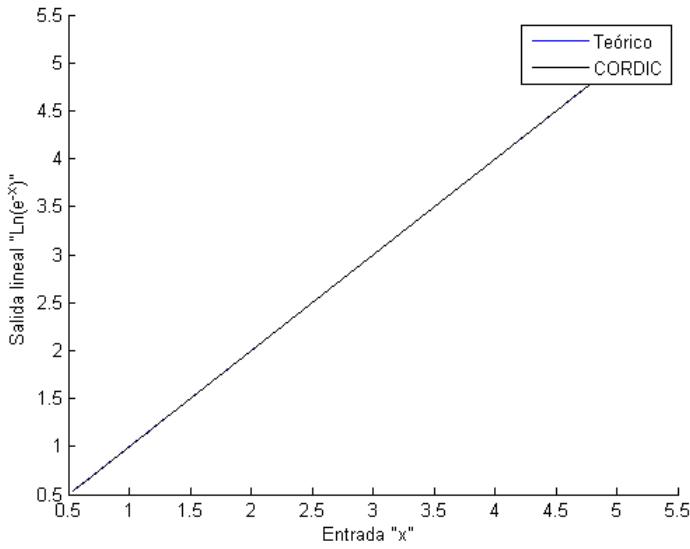


Figura 3.10: Datos de la simulación post-síntesis del linealizador para un rango de convergencia CORDIC con 12 iteraciones

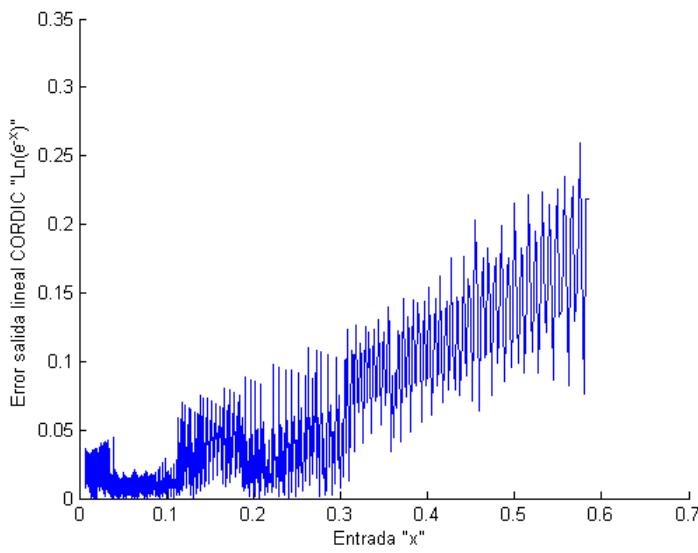


Figura 3.11: Porcentaje de error rango de convergencia circuito CORDIC con 12 iteraciones simulación post-síntesis

Una mejor aproximación se puede lograr utilizando 12 iteraciones en el cálculo del logaritmo natural, en la figura 3.10 se pueden observar los resultados obtenidos, se requieren

660 ciclos de reloj para ejecutar el cálculo completo. La figura 3.11 muestra el error en cada cálculo, donde el porcentaje de error máximo es de 0,259% y el porcentaje de error promedio es de 0,0351%.

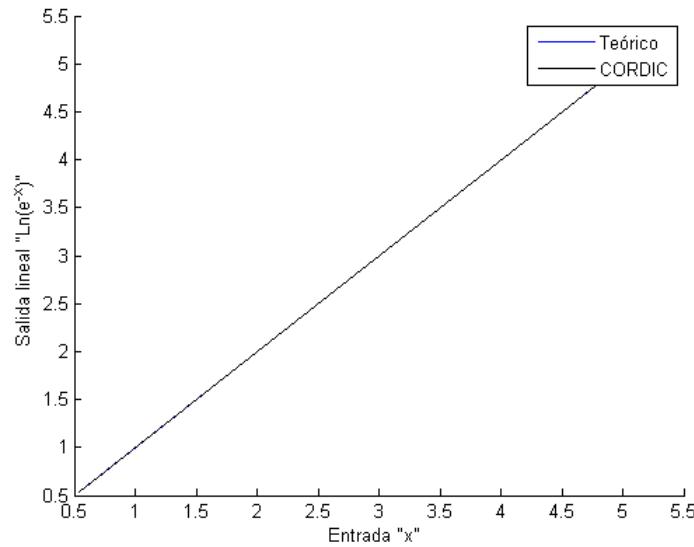


Figura 3.12: Datos de la simulación post-síntesis del linealizador para un rango de convergencia CORDIC con 15 iteraciones

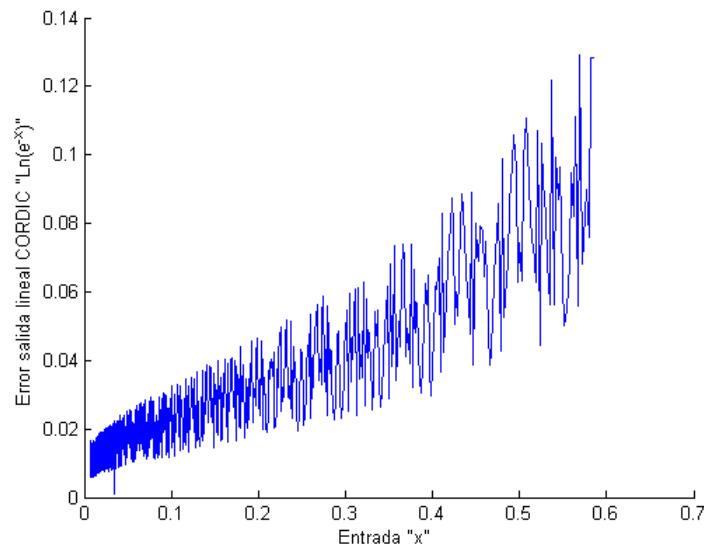


Figura 3.13: Porcentaje de error rango de convergencia circuito CORDIC con 15 iteraciones simulación post-síntesis

Utilizando 15 iteraciones en el cálculo del logaritmo natural se logra la mayor aproximación, sin embargo se requieren 818 ciclos de reloj y se vuelve mas lento el proceso, en la

figura 3.12 se pueden observar los resultados obtenidos. La figura 3.13 muestra el error en cada cálculo, donde el porcentaje de error máximo es de 0,129% y el porcentaje de error promedio es de 0,0257%.

En las figuras de error 3.9, 3.11 y 3.13 se puede observar que el comportamiento del algoritmo es mas exacto cuando los valores de corriente de entrada del argumento de linealizador "T" son pequeños, a medida que este argumento se vuelve mayor el porcentaje de error también incrementa.

Capítulo 4

Sistema de conversión punto flotante a punto fijo y normalización

El circuito realizado para la linealización se basa en el formato IEEE 754, sin embargo el estimador de parámetros que se tiene, esta basado en un formato punto fijo, de manera que se debe considerar una conversión entre ambos formatos, para esto se estudió como pasar de punto flotante a punto fijo, ambos en 32-bits.

4.1 Sistema de conversión y normalización

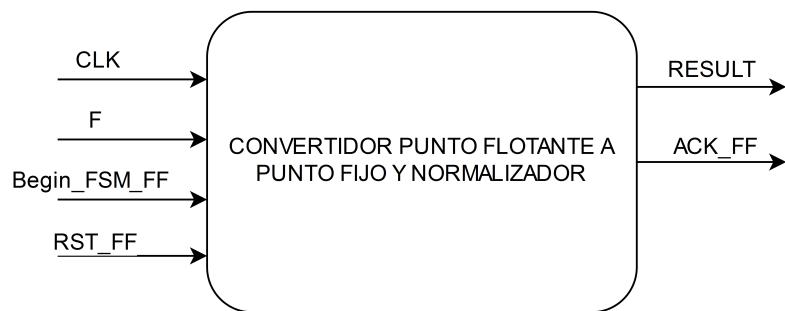


Figura 4.1: Bloque general del convertidor punto flotante a punto fijo y normalizador.

La figura 4.1 contiene el bloque general del sistema de conversión y normalización , este posee 4 entradas: CLK , F , Begin_FF , RST_FF y 2 salidas: ACK_FF , RESULT.

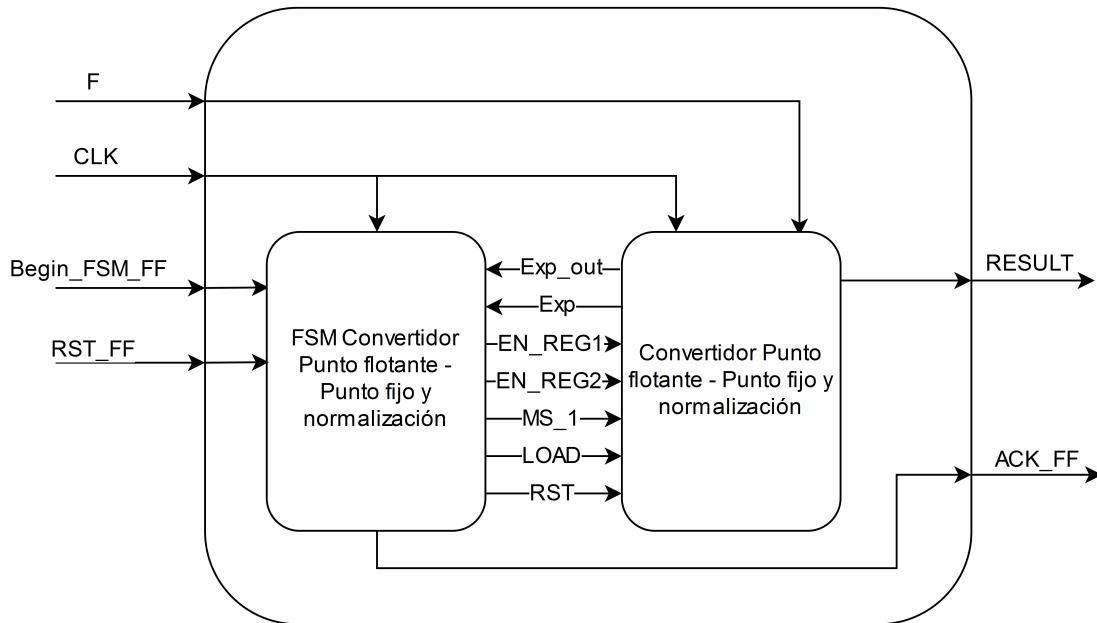


Figura 4.2: Sistema de conversión, normalización, señales de datos y control.

El sistema de conversión y normalización de la figura 4.2 cuenta con dos módulos principales:

- *Convertidor-Normalizador*: En este realizan todas las operaciones requeridas para la conversión del formato punto flotante-punto fijo y de la normalización, este se encarga del manejo de los datos en el cálculo.
- *Control*: Este se encarga de proveer las señales de control requeridas por el convertidor-normalizador, según las condiciones que se tenga y se requiera en cada estado.

Señales de datos:

- *F*: Dato de entrada, este posee un valor en formato IEEE 754.
- *RESULT*: Dato de salida convertido de punto flotante a punto fijo.

Señales de control:

- *CLK*: Reloj del sistema, este ejecuta ciclos de reloj con una frecuencia preestablecida.
- *Begin_FF*: Este se encarga de iniciar la la unidad, indica a la máquina de estados que inicia la secuencia.
- *RST_FF*: Este se encarga restablecer los valores iniciales del sistema de conversión y normalización.
- *ACK_FF*: Indica cuando la conversión y la normalización han sido realizadas.

4.2 Convertidor punto flotante - punto fijo y normalizador

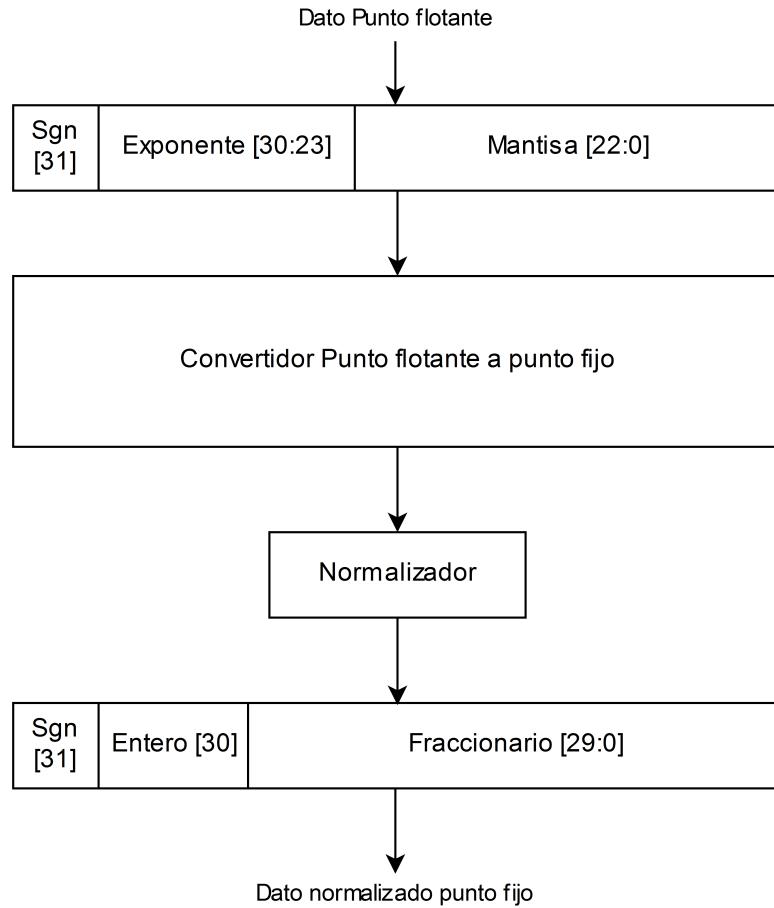


Figura 4.3: Proceso general de conversión y normalización

En la figura 4.3 se puede observar el diagrama de solución que se utilizó para el desarrollo del convertidor-normalizador , primeramente ingresa el número en formato IEEE 754 32-bits, posteriormente se efectúa la conversión a punto fijo, en donde se asigna un bit de signo, 5 bits de parte entera y 26 bits para la parte fraccionaria, este dato se procesa en una etapa de normalización y se obtiene el resultado, este valor final esta normalizado para la corriente y tensión del panel, $V = [0, 1]$ e $i = [-1, 1]$, para el formato punto fijo solo se requiere, un bit de signo, un bit en la parte entera y 30 para la parte fraccionaria, como se muestra en la figura 4.3, el aumento de bits en la parte fraccionaria indica una mejor precisión en el resultado.

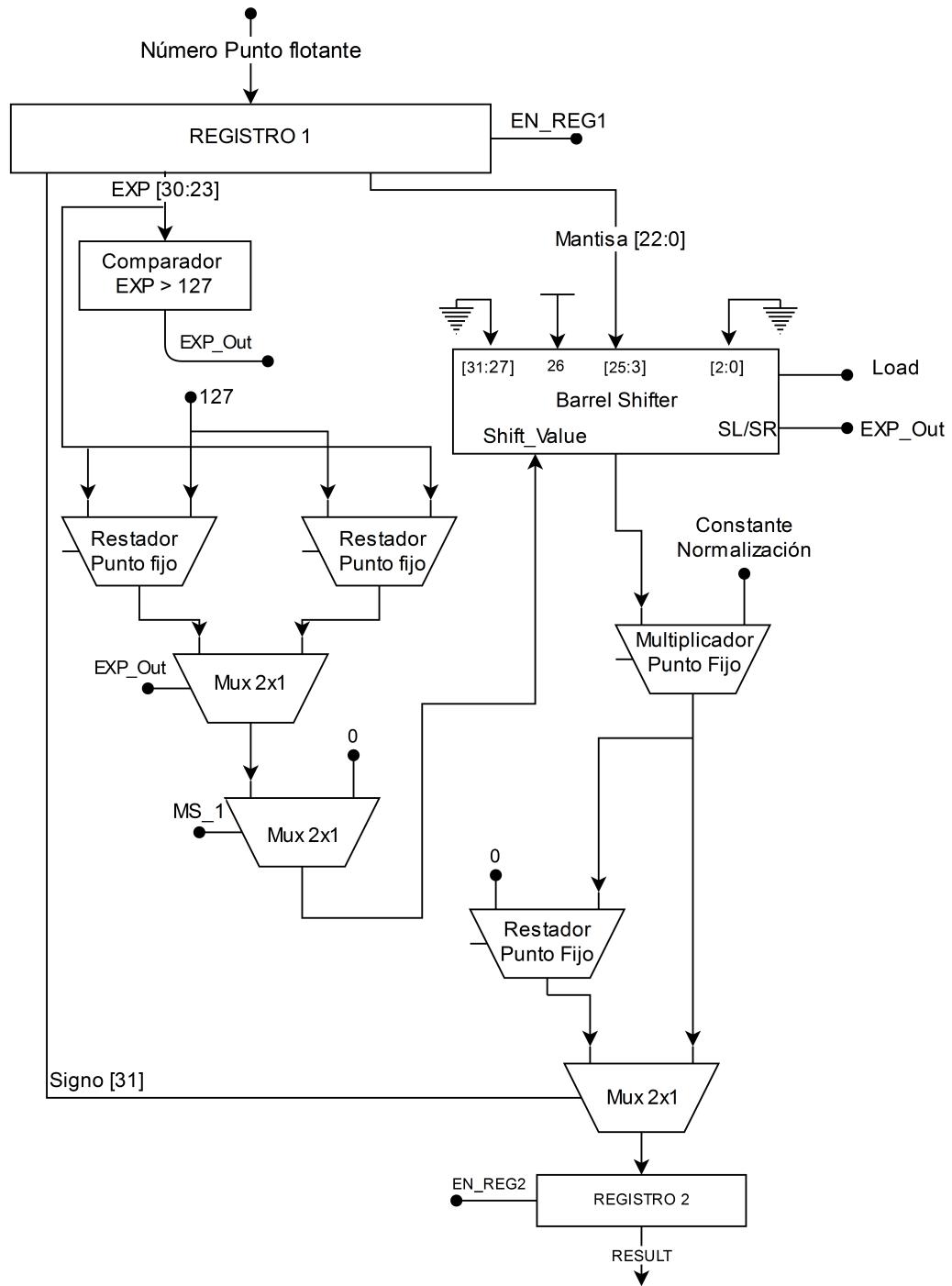


Figura 4.4: Circuito de conversión punto flotante a punto fijo y normalización

Como se observa en la figura 4.4, la etapa de conversión de formatos contiene un comparador, este se utiliza para saber si el número en formato IEEE 754, contiene un exponente mayor o menor que 127, si el número es igual a 127, indica un exponente de 0, si es mayor la bandera de salida del comparador es igual a 1, $EXP_Out = 1$, si se da esta condición, se debe realizar la operación $EXP - 127$, si el exponente es menor que 127, la bandera

EXP_Out es 0 y la operación es $127 - EXP$, ambas operaciones indican la cantidad de desplazamientos que se deben realizar en el Barrel-shifter, este se utiliza debido a que es puramente combinacional, por lo que no requiere ciclos de reloj para funcionar, esto disminuye el tiempo de cálculo en la etapa de conversión, la dirección de los desplazamientos se puede controlar mediante una señal de control que posee, esta es conectada a la bandera del comparador *EXP_Out*, el dato de entrada para el barrel-shifter esta compuesto por un bit mas significativo fijo en alto y los 23 bits de la mantisa del dato de entrada en punto flotante, este dato compuesto del barrel-shifter se le aplican los desplazamientos para obtener el resultado en punto fijo, este resultado siempre es positivo, debido a que en esta etapa no se contempla el signo, se retomará en otra etapa.

Un dato normalizado requiere una división entre el máximo valor que se puede procesar, sin embargo en un circuito digital las divisiones se tornan complicadas y requieren de mucha área, por lo que utiliza una multiplicación por una constante, esta etapa de normalización posee un multiplicador en punto fijo, las entradas de este contienen el valor convertido en punto fijo y una constante de normalización, esta constante se calcula en la siguiente ecuación:

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

La etapa de conversión y normalización se utiliza tanto para la corriente i_{pv} como para la tensión V_{pv} del panel, esta constante de normalización varía para cada circuito:

$$Cv_{norm} = \frac{1}{18.1} = 0.055248618 \quad (4.2)$$

$$Ci_{norm} = \frac{1}{Ln(0.00667769)} = 0.199641045 \quad (4.3)$$

Donde Cv_{norm} es la constante de normalización de la tensión y Ci_{norm} la constante de normalización de la corriente.

Posteriormente a la normalización, se debe tomar en cuenta el signo del valor inicial punto flotante, el formato IEEE 754 contiene el signo en el bit mas significativo (bit 32), en la unidad de conversión-normalización se realiza la resta $0 - Dato_Punto_fijo$, y el multiplexor 2x1 del resultado selecciona el valor final en punto fijo, si el bit 32 del valor inicial punto flotante es cero, el valor final en punto fijo es positivo, de lo contrario el valor final en punto fijo es negativo.

4.3 Control para el convertidor punto flotante - punto fijo y normalizador

La arquitectura diseñada para el convertidor-normalizador en su mayoría es combinacional sin embargo requiere un control, que detecta si se deben realizar desplazamientos, y cuando se debe almacenar datos en registros.

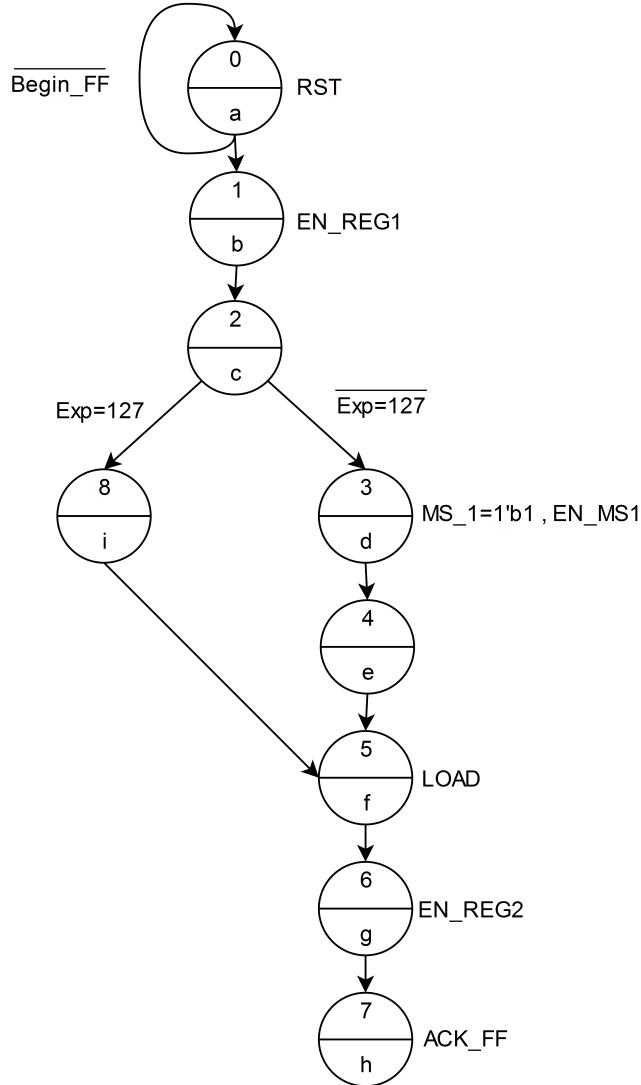


Figura 4.5: Maquina de estados finita para el convertidor-normalizador

El control de esta arquitectura se realiza por medio de una máquina de estado finita bastante sencilla, donde básicamente se cuenta con cuatro acciones principales, el primer estado (a) espera que la unidad sea iniciada mediante la señal *Begin_FF* y se ejecuta un reset en los registros, el estado (b) guarda el dato en el *Registro 1*, el estado (c) verifica la condición $EXP = 127$ con esta se determina si se deben realizar desplazamientos, el estado (f) almacena en el barrel-shifter el dato convertido, el estado (g) almacena el

resultado final en el *Registro 2* y en el estado (h) se indica mediante la bandera *ACK_FF* que el dato ya fue convertido y normalizado.

4.4 Sistema de conversión-normalización en verilog

Este circuito se implementó por medio de el lenguaje de descripción de hardware "Verilog", inicialmente se realizaron pequeños bloques pertenecientes a cada elemento requerido por la arquitectura diseñada, se implementa la unidad de conversión-normalización y la unidad de control en bloques separados, de manera que se pudieran realizar pruebas sin dependencia de los bloques entre si, para una mejor depuración de errores y re-diseño, finalmente se realizan las simulaciones al bloque completo en la figura 4.6.

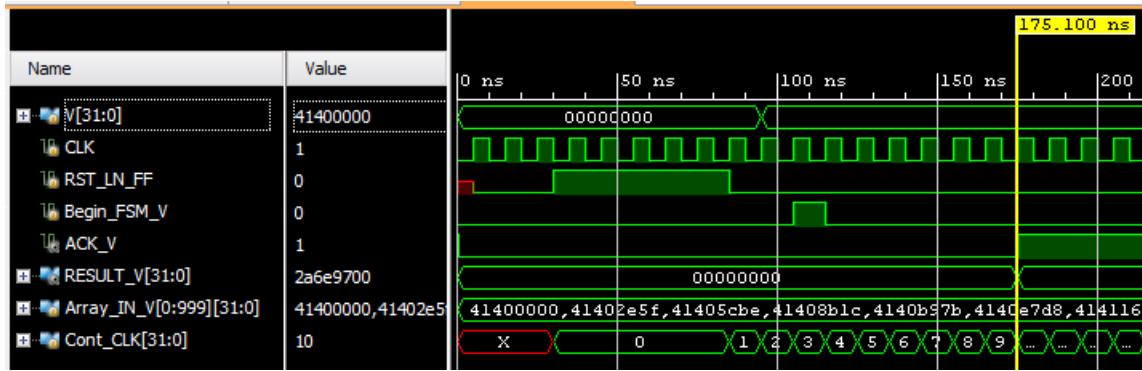


Figura 4.6: Simulación del circuito de conversión y normalización

4.5 Resultados de la simulación del circuito convertidor-normalizador

En la implementación de un diseño en hardware, es de suma importancia simular y verificar que este funcione de manera adecuada al comportamiento esperado teóricamente. Para la comprobación de la unidad de conversión-normalización, se realizaron una serie de pruebas en donde se programó una simulación ("testbench") que contiene una prueba con mil valores de entrada, ingresados por medio de un archivo de texto previamente editado con los datos de entrada del comportamiento según el modelo del panel, las pruebas de esta unidad se realizaron con valores de corriente i_{pv} y valores de tensión V_{pv} .

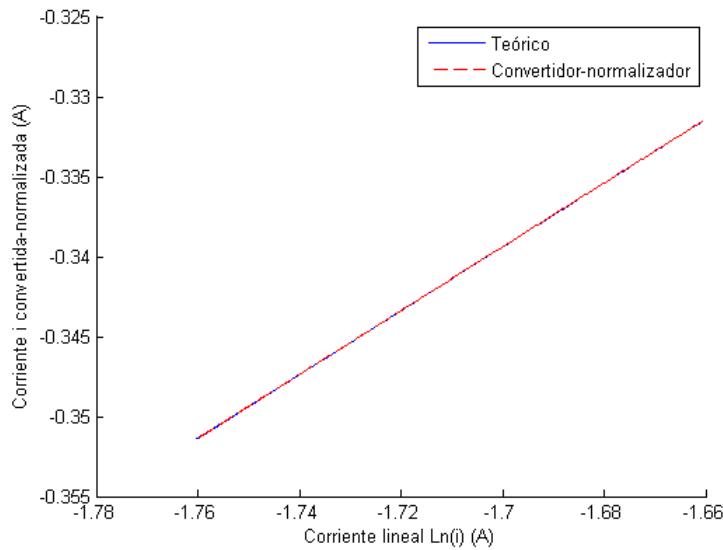


Figura 4.7: Comparación entre la conversión-normalización de corriente i_{pv} teórica y la simulación post-síntesis del circuito

En la figura 4.7 se presenta la comparación entre los resultados obtenidos teóricamente y experimentalmente, tomando como datos de entrada valores de corriente en formato punto flotante y retornando en la salida valores de corriente normalizados y en formato punto fijo. Estos resultados se pueden comparar por medio del porcentaje de error.

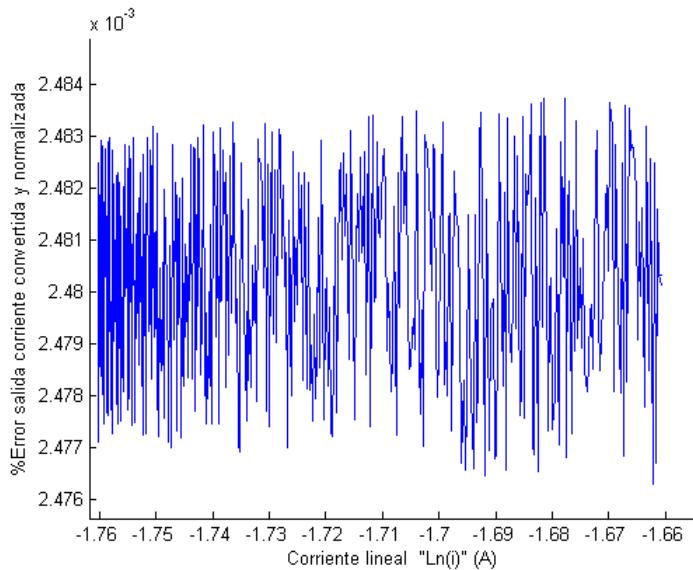


Figura 4.8: Porcentaje de error entre la conversión-normalización de corriente i_{pv} teórica y del circuito

En la figura 4.8 se puede observar el error entre la conversión de la corriente normali-

zada teórica y experimental, con un error porcentual máximo de 0,0024837% y un error promedio de 0,002478%

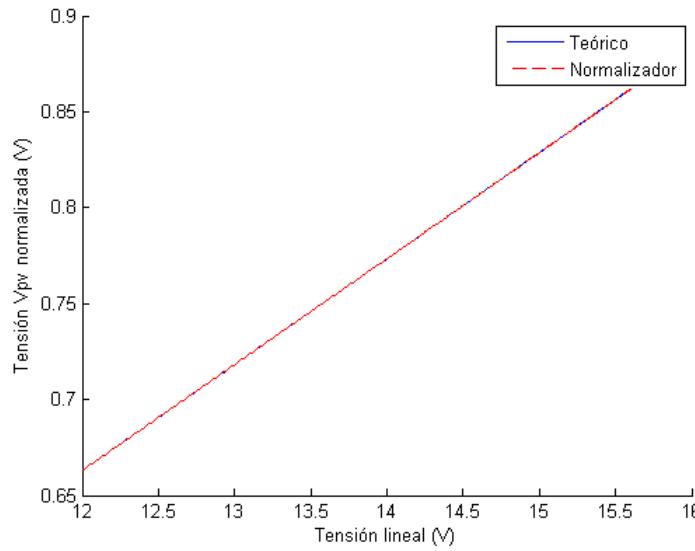


Figura 4.9: Comparación entre la conversión-normalización de tensión V_{pv} teórica y la simulación post-síntesis del circuito

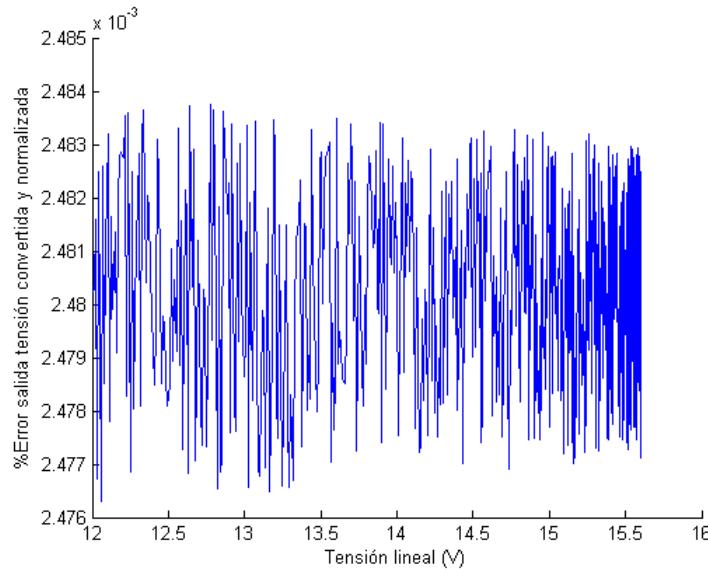


Figura 4.10: Porcentaje de error entre la conversión-normalización de tensión V_{pv} teórica y del circuito

Un análisis similar al que se realizó para la conversión-normalización de la corriente se utiliza para los resultados de la tensión, en la figura 4.9 se muestran los resultados obtenidos con una tensión de entrada y su normalización tanto teórico como experimental,

de la misma manera se puede hacer el cálculo del error entre ambas, en la figura 4.10 se puede observar el error máximo de 0,0024837% y un error promedio de 0,002478%.

Esta unidad contiene muchos bloques combinacionales, por lo que se requieren pocos ciclos de reloj en la máquina de estados para realizar la conversión y la normalización, la máquina de estados finita indica que se requieren 8 ciclos de reloj para que el resultado sea concluido, con las simulaciones efectuadas al circuito se comprueba como se mostró en la figura 4.6

Capítulo 5

Sistema de linealización, conversión punto flotante a punto fijo y normalización

En la implementación del circuito completo se utilizan dos etapas, una para corriente del panel i_{pv} y otra para la tensión del panel V_{pv} .

- *Corriente i_{pv} :* El bloque para la corriente requiere las etapa de linealización, conversión de la salida del linealizador de punto flotante a punto fijo, y un normalizador que da una salida de corriente entre el rango [-1,1].
- *Tensión V_{pv} :* La tensión del panel se trabaja de manera lineal, de manera que solo se requiere de un normalizador que contenga la salida entre el intervalo [0,1].

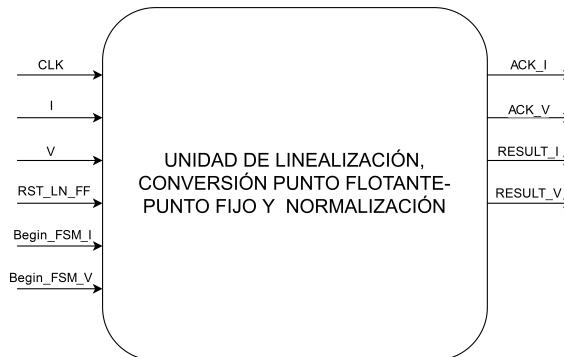


Figura 5.1: Sistema de linealización, conversión y normalización de corriente y tensión para un panel fotovoltaico

El sistema de la figura 5.1 cuenta con 6 entradas y 4 salidas, estas se describen a continuación.

Entradas:

- CLK : Reloj del sistema.
- I : Dato de corriente en formato IEEE 754
- V : Dato de tensión en formato IEEE 754
- RST_LN_FF : Reset para las unidades de corriente y tensión.
- $Begin_FSM_I$: Inicia la máquina de estados del linealizador de corriente.
- $Begin_FSM_V$: Inicia la máquina de estados de el convertidor punto flotante-punto fijo para la tensión.

Salidas:

- ACK_I : Esta bandera indica que el resultado de las operaciones para la corriente se encuentra listo.
- ACK_V : Esta bandera indica que el resultado de las operaciones para la tensión se encuentra listo.
- $RESULT_I$: Resultado de corriente lineal y normalizado en formato punto fijo.
- $RESULT_V$: Resultado de tensión normalizado en formato punto fijo.

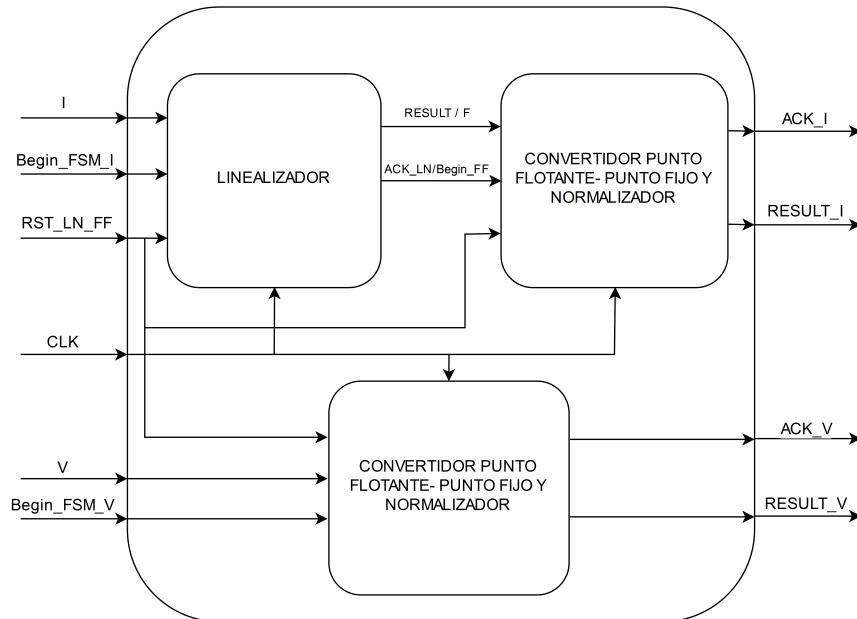


Figura 5.2: Sistema de linealización-conversión-normalización de corriente y sistema de conversión- normalización de tensión para un panel fotovoltaico

La figura 5.2 muestra la distribución de bloques funcionales para la corriente y tensión que proviene del panel fotovoltaico, todos los bloques se encuentran sincronizados con un mismo reloj (CLK) para la sección de corriente se realiza primeramente la linealización, iniciando con las señales RST_LN_FF y $Begin_FSM_I$ y el dato de entrada I , una vez ejecutada la operación se envía una señal indicando que el dato esta listo ACK_LN , esta se conecta a la señal de entrada e inicio del convertidor-normalizador de corriente $Begin_FF$, justamente un ciclo de reloj antes de que esta señal de control se envíe, el resultado de la linealización ($RESULT$) esta listo, este está conectado a la entrada F del convertidor-

normalizador, la ejecución de este se comprueba con la señal ACK_I en donde se indica que la conversión-normalización fue realizada y el resultado se encuentra en RESULT_I.

El bloque de tensión funciona de manera similar al de corriente, iniciando con las señales RST_LN_FF y Begin_FSM_V y el dato de entrada V, una vez ejecutada la operación se envía una señal indicando que el dato esta listo ACK_V, el resultado se despliega en RESULT_V

5.1 Circuito para realizar las pruebas en Nexys-4

Para desarrollar las pruebas dentro de la FPGA, se requirió diseñar e implementar un circuito para insertar datos a la entrada del linealizador-normalizador, sean procesados y posteriormente enviados por medio de transmisión serial hacia un computador.

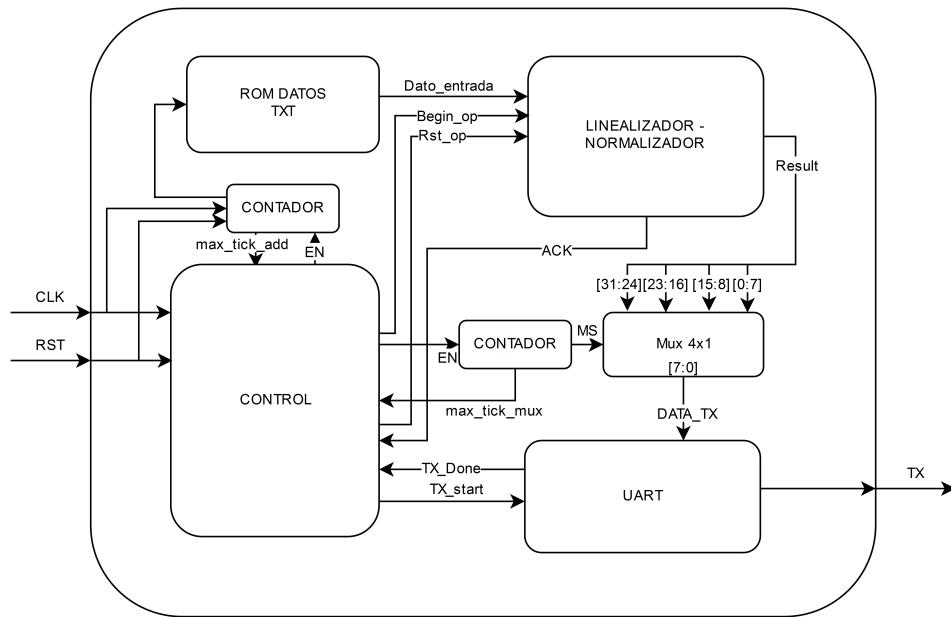


Figura 5.3: Circuito de prueba para enviar datos por medio de UART desde la nexys-4 hacia un computador

El diagrama de la figura 5.3 muestra el diseño del circuito utilizado, en donde se tiene una sección para ingresar 1024 datos, estos son almacenados en una memoria ROM, esta se carga por medio de un archivo de texto que contiene los datos de entrada para el linealizador-normalizador, esta memoria posee un contador para direccionar los 1024 datos. La sección de transmisión posee una unidad UART, esta se utiliza para enviar los resultados de la unidad de linealización-normalización hacia la PC, estos resultados tienen un formato punto fijo de 32 bits, el uart envia únicamente paquetes de 8 bits, por lo que se requiere enviar 4 paquetes por cada resultado, esto se logra por medio de una configuración de un multiplexor 4x1 y un contador, las entradas del multiplexor contienen

el resultado dividido en paquetes de 8 bits y el contador selecciona el multiplexor segun sea el paquete que se desea enviar. Por ultimo se cuenta con un control para el circuito, este se encarga de llevar la sincronía de los datos, primeramente se realiza una carga en el contador de la memoria ROM, este carga la primera posición de la ROM, posteriormente se inicia el procesamiento por parte de la unidad de linealización-normalización con la señal *Begin_op*, el control espera a que el resultado este listo por medio de la señal ACK, cuando la condición ACK=High, el sistema se encuentra listo para transmitir, el contador del multiplexor selecciona el primer paquete y lo envía, el siguiente paquete se puede transmitir hasta que el modulo UART envié la señal *TX_DONE*, así sucesivamente hasta que se envíen los 4 paquetes, cuando la selección del multiplexor esta en 2b'11 (contador del multiplexor), la señal *max_tick_mux* indica al control que debe realizar una cuenta para la dirección de la ROM. Este proceso se repite hasta que la cuenta de las direcciones de la ROM es 1023 y la señal *max_tick_add = 1*

5.2 Simulación del sistema de linealización, conversión punto flotante a punto fijo y normalización implementado en hardware

Este circuito se implementó por medio de la unión de los bloques anteriormente diseñados, implementados y simulados.

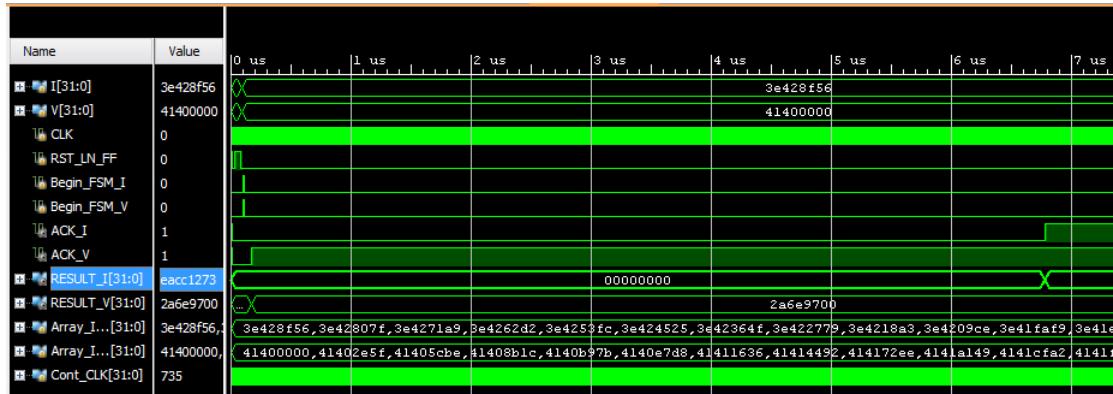


Figura 5.4: Simulación del sistema de linealización-conversión-normalización de corriente y sistema de conversión- normalización de tensión para un panel fotovoltaico

La simulación básica ”behavioral” efectuada por Vivado, verifica el funcionamiento adecuado del circuito a manera de software y de lógica sin embargo no es suficiente ya que se debe realizar la síntesis e implementación en verilog (Hardware), para esto se realizan las simulaciones post-synthesis y post-implementation, tanto del funcionamiento como de los retardos del sistema. La figura 5.4 muestra la simulación de tiempos posterior a la implementación.

5.3 Resultados del sistema de linealización, conversión punto flotante a punto fijo y normalización implementado y verificado en hardware por medio de una FPGA nexys-4

Las simulaciones para el circuito completo brindan una mejor información acerca del porcentaje de error total obtenido dentro de la conexión de las tres etapas linealización, conversión y normalización. Se efectuaron las simulaciones de temporización y funcionalidad posterior a la síntesis, obteniendo resultados experimentales para realizar comparación contra los teóricos esperados.

En el capítulo 3 se demostró el funcionamiento del linealizador con 8,12 y 15 iteraciones para el rango de convergencia del algoritmo de CORDIC, en este capítulo se utiliza el mismo principio de verificación, sin embargo se debe contemplar el error del normalizador, por lo que se realizan pruebas y comparaciones similares utilizando como datos de entrada un barrido de 1000 valores ingresados en el modelo teórico del panel fotovoltaico, esto con el fin de observar un comportamiento similar al real, estos valores de corriente son pequeños debido a la diferencia de corrientes y perdidas en el panel fotovoltaico, por lo que los porcentajes de error serán menores, ya que el circuito posee una mejor aproximación en el extremo inferior del intervalo de convergencia del algoritmo CORDIC. La visualización de los resultados obtenidos se realiza de una mejor manera mediante gráficos que indican como se comporta el circuito ante datos de entrada, salida, y porcentaje de error, apartir de los resultados de las simulaciones, se puede analizar la frecuencia de ejecución a la que se obtiene un resultado linealizado-normalizado, esta depende del numero de iteraciones que se utilice, por otro lado se deben sumar los ciclos de cada uno de los bloques funcionales que componen la ruta de ejecución de la corriente este circuito, se detallará de mejor manera en la siguiente sección. Si tomamos el bloque para la tensión, este solo cuenta con la conversión-normalización y no depende del numero de iteraciones por lo que se requieren de 8 ciclos de reloj por cada dato.

5.3.1 Sistema de linealización, conversión y normalización para la corriente i_{pv} con 8 iteraciones implementado en una FPGA nexys-4

A partir de las simulaciones post-síntesis efectuadas por medio de la herramienta Vivado con una aproximación de 8 iteraciones, se puede observar que se requiere de 468 ciclos de reloj para completar la ejecución de un dato, este se compone de 460 ciclos de reloj para el linealizador y 8 ciclos de reloj para el convertidor-normalizador, con una donde la velocidad de ejecución es de 217kHz con un reloj de sistema de 100MHz.

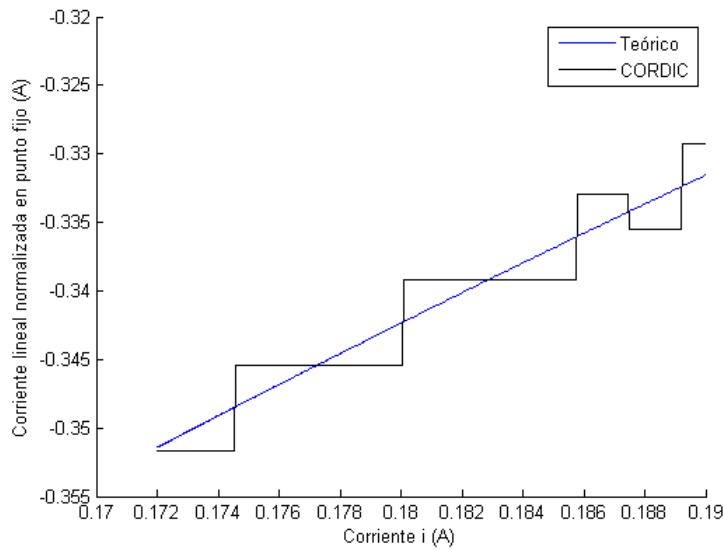


Figura 5.5: Comparación entre el valor teórico y el valor obtenido del circuito de linealización-conversión-normalización de corriente con 8 iteraciones en el algoritmo de CORDIC del linealizador

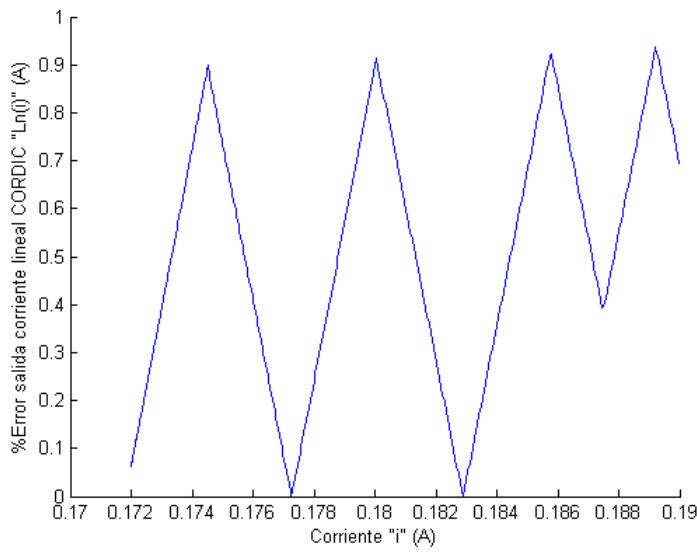


Figura 5.6: Porcentaje de error entre el valor teórico y el valor obtenido del circuito de linealización-conversión-normalización de corriente con 8 iteraciones en el algoritmo de CORDIC del linealizador

En la figura 5.5 se puede observar la comparación entre los datos obtenidos del circuito implementado en una FPGA nexys-4 contra los datos teóricos al realizar la misma función del circuito, debido a que son solo 8 iteraciones el circuito posee un resultados aceptables

pero poco exactos, el gráfico muestra que el algoritmo trata de mantenerse cerca de los valores reales, sin embargo posee un comportamiento escalonado, en donde para cierta cantidad de valores de entrada posee un mismo valor de salida constante cercano al valor real. La figura 5.6 muestra el porcentaje de error asociado a cada valor de entrada linealizado y normalizado en formato punto fijo, donde el porcentaje de error máximo es de 0,922% y un porcentaje de error promedio de 0,455% esto para valores de corriente derivados a partir del modelo teórico del panel fotovoltaico.

5.3.2 Sistema de linealización, conversión y normalización para la corriente i_{pv} con 12 iteraciones implementado en una FPGA nexys-4

Según las simulaciones post-síntesis del circuito con una configuración del algoritmo de CORDIC con 12 iteraciones, se requiere de 668 ciclos de reloj para realizar el procesamiento completo de un dato de entrada, 660 ciclos de reloj son utilizados por el linealizador y 8 ciclos de reloj para el convertidor-normalizador, con una velocidad de ejecución de 151kHz, utilizando un reloj de sistema de 100MHz.

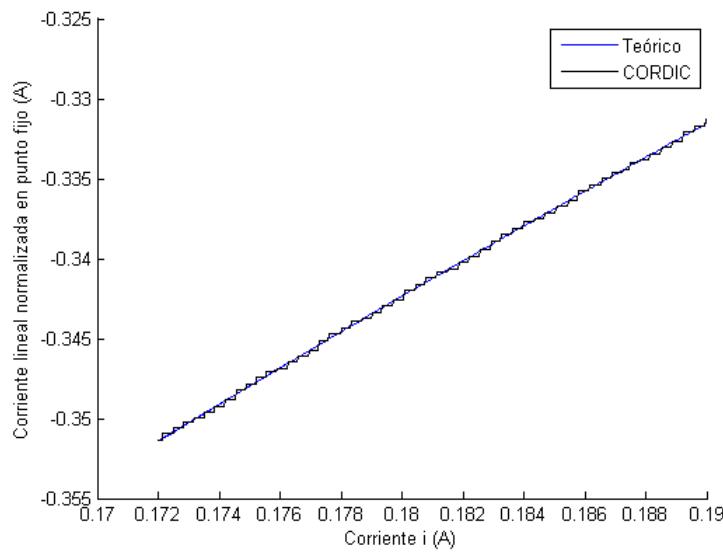


Figura 5.7: Comparación entre el valor teórico y el valor obtenido del circuito de linealización-conversión-normalización de corriente con 12 iteraciones en el algoritmo de CORDIC del linealizador

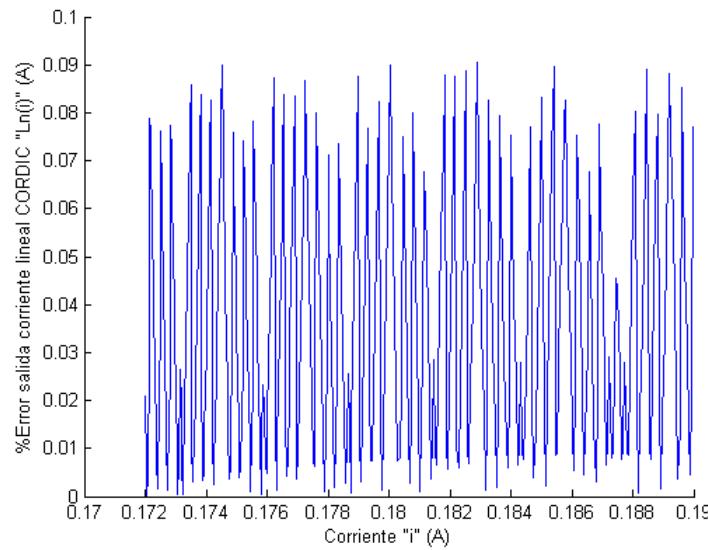


Figura 5.8: Porcentaje de error entre el valor teórico y el valor obtenido del circuito de linealización-conversión-normalización de corriente con 12 iteraciones en el algoritmo de CORDIC del linealizador

En la figura 5.7 se puede observar la comparación entre los datos obtenidos experimentalmente contra los datos teóricos al realizar la misma función del circuito, el uso de 12 iteraciones brinda una mejor exactitud contra 8 iteraciones, se puede observar que se presenta una mayor suavidad en la curva, sin embargo siempre se presenta un comportamiento escalonado pero con un valor mas cercano a la curva real. La figura 5.8 muestra el porcentaje de error asociado a cada valor de entrada linealizado con 12 iteraciones y normalizado en formato punto fijo, donde el porcentaje de error máximo es de 0,0897% y un porcentaje de error promedio de 0,0345% esto para valores de corriente derivados a partir del modelo teórico del panel fotovoltaico.

5.3.3 Sistema de linealización, conversión y normalización para la corriente i_{pv} con 15 iteraciones implementado en una FPGA nexys-4

En el sistema del panel es de suma importancia el tiempo de muestreo según sea la frecuencia del panel, esto implica velocidad de ejecución dentro del cálculo, se logró determinar que con 15 iteraciones se obtienen resultados con muy buena exactitud,sin embargo el tiempo de ejecución aumenta a 826 ciclos de reloj, tomando en cuenta que 818 ciclos son requeridos por el linealizador y 8 ciclos de reloj para la conversión de formato IEEE 754 a punto fijo y normalización, en donde la velocidad de ejecución es de 121kHz con un reloj de sistema de 100MHz.

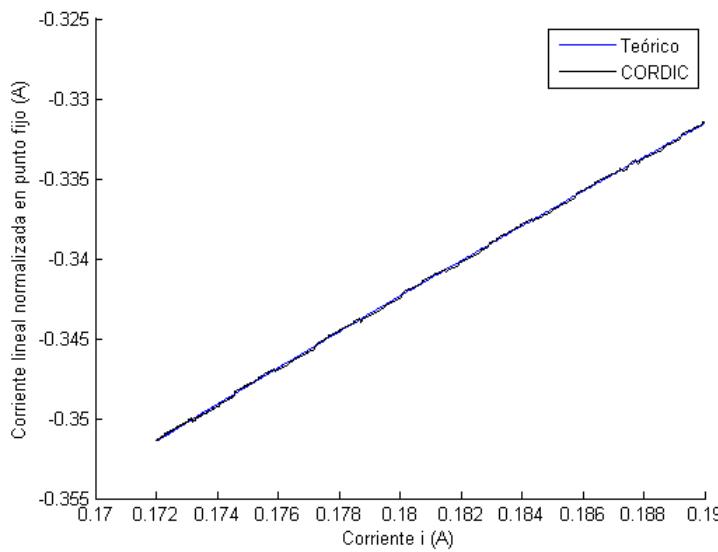


Figura 5.9: Comparación entre el valor teórico y el valor obtenido del circuito de linealización-conversión-normalización de corriente con 15 iteraciones en el algoritmo de CORDIC del linealizador

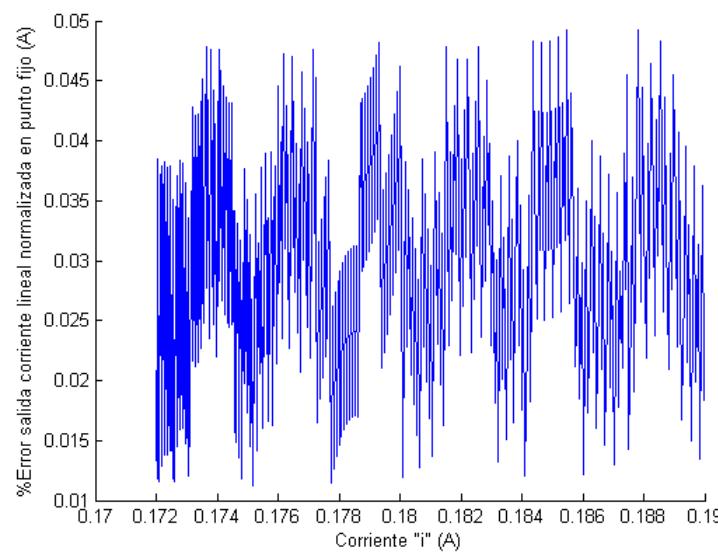


Figura 5.10: Porcentaje de error entre el valor teórico y el valor obtenido del circuito de linealización-conversión-normalización de corriente con 15 iteraciones en el algoritmo de CORDIC del linealizador

Con 15 iteraciones el valor experimental es muy similar al valor teórico, como se observa en la figura 5.9, y se puede comprobar en la figura 5.10 donde se muestra que el porcentaje de error es muy cercano a cero, con un valor máximo de 0,0483% y un error promedio

de 0,0292% esto para valores de corriente derivados a partir del modelo teórico del panel fotovoltaico.

El sistema de optimización se utilizará en un panel fotovoltaico que posee una frecuencia de muestreo de 100kHz, es decir que el sistema de optimización debe realizar un cálculo completo a una mayor velocidad que el panel, aproximadamente el sistema de linealización y normalización toma un 40% del total del tiempo de ejecución del sistema de optimización. Según lo anterior para realizar el cálculo en el tiempo indicado, el linealizador deberá utilizar 8 iteraciones.

5.4 Recursos utilizados

Tabla 5.1: Resumen del reporte post implementación del uso de dispositivos generado por la herramienta Vivado.

Recurso	Utilizados	Disponibles	Utilizados%
LUT	1017	63400	1.60
FF	912	126800	0.72
DSP	4	240	1.67
IO	134	210	63.81
BUFG	1	32	3.12

En la tabla 5.1 se muestra el uso de recursos para la implementación el circuito completo en la Nexys4, el máximo recurso utilizado son los puertos I/O debido a que se requieren 32 bits en cada entrada I,V y 32 bits en cada salida RESULT_I , RESULT_V, sin embargo el uso de los demás recursos es sumamente bajo, el uso de los I/O se reducirá debido a que este circuito se deben acoplar a otro bloques del sistema de optimización.

5.5 Reporte de tiempos

Dentro del reporte de tiempos se analizan las peores rutas "ruta critica", para un buen diseño se requiere de slacks positivos, en el caso de "setup" un slack positivo indica que el dato llega a tiempo antes de ser utilizado, en el caso del "hold" indica que se mantiene por el tiempo adecuado antes de ser procesado por otro bloque, si ambos slacks son cero, el diseño esta en el límite y puede ser un poco arriesgado en sincronización. En la tabla 5.2 se muestran los valores obtenidos para los slacks del circuito del linealizador-normalizador.

Tabla 5.2: Resumen del reporte post implementación de tiempos del circuito completo, a partir de la herramienta Vivado.

Peor slack de todas las rutas	tiempo (ns)
Setup	0.459
Hold	0.109
Pulse width	4.5

5.6 Consumo de potencia

Con la ayuda de las herramientas de análisis de potencia de Vivado, se pudo obtener los resultados que pertenecen al consumo de potencia, tanto estática como dinámica. En la tabla 5.3 se muestra el valor para cada consumo, y el total.

Tabla 5.3: Resumen del reporte post implementación de la potencia estática, dinámica y total, de la herramienta Vivado.

Potencia	Consumo de potencia (mW)
Dinámica	11
Estática	91
Total	102

Para los recursos utilizados se puede realizar un estudio de consumo de potencia dinámica, estos resultados se muestran en la tabla 5.4 donde el mayor consumo de potencia se presenta por parte del reloj del sistema y señales, las señales que poseen un valor de cero no se refieren a un consumo nulo, si no que es relativamente pequeño en comparación con las señales mas criticas.

Tabla 5.4: 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	4
Signals	4
Logic	3
DSP	0
I/O	0

Capítulo 6

Conclusiones y recomendaciones

6.1 Conclusiones

- Se demostró que es posible sintetizar la unidad de linealización-normalización con precisión aceptable utilizando versiones reducidas del estándar IEEE 754 (32bits).
- Se logró demostrar en el circuito linealizador que a mayor numero de iteraciones mayor exactitud presenta el resultado, sin embargo requiere un mayor tiempo de ejecución.
- Se demostró que para el circuito linealizador, el porcentaje de error máximo fue de 2.74% con 8 iteraciones en el rango de convergencia del algoritmo de CORDIC.
- Se demostró que para el circuito convertidor-normalizador tanto de corriente como de tensión, el porcentaje de error máximo fue de 0,0024%.
- Se concluyó que se debe utilizar 8 iteraciones para cumplir con el tiempo de ejecución del linealizador-normalizador en el sistema de optimización para paneles solares.
- Se determinó que el módulo de la corriente consume más recursos que el módulo de tensión, dado que la corriente requiere de un modulo de linealización y a su vez mayor cantidad de hardware.

6.2 Recomendaciones

- Se utilizó un único sumador punto flotante para reducir el área del circuito sin embargo el espacio utilizado en la FPGA es muy poco, por lo tanto se podría realizar modificaciones para reducir el número de ciclos de cada iteración de la máquina de estados, implementando una arquitectura con cálculo de X_i , Y_i y Z_i de manera paralela, es decir utilizar un sumador de punto flotante para cada variable de manera que se realicen los cálculos simultáneos, esta variación implica un aumento en el área sin embargo le aumentaría alrededor de 3 veces más la velocidad de ejecución.
- Si se utiliza algún otro tipo de panel con el sistema y se requiere un rango de linealización más amplio se puede utilizar un algoritmo de CORDIC hiperbólico

con una extensión de las iteraciones, agregando dos iteraciones negativas se puede extender, de manera que no sea tan limitado, sin embargo se requiere de mayor cantidad de recursos para la implementación de estas dos iteraciones y puede ser mas lento en ejecución.

Capítulo 7

bibliográficas

- [1] Suskis, Pavels, and Ilya Galkin. "Enhanced photovoltaic panel model for MATLAB-simulink environment considering solar cell junction capacitance." Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE. IEEE, 2013.
- [2] González-Longatt, Francisco M. "Model of photovoltaic module in Matlab." II CIBE-LEC 2005 (2005): 1-5.
- [3] C. Meza, R. Ortega, "Control and estimation scheme for PV central inventers", in 24th International Conference on information, Communication and Automation Technologies, Nov, 2013
- [4] Chiang, Ching-Tsan, Tung-Sheng Chiang, and Hou-Sheng Huang. "Modeling a photovoltaic power system by CMAC-GBF." Photovoltaic Energy Conversion, 2003. Proceedings of 3rd World Conference on. Vol. 3. IEEE, 2003.
- [5] Ibrahim, Muhammad Nasir, et al. "Hardware Implementation of Math Module Based on CORDIC Algorithm Using FPGA." Parallel and Distributed Systems (ICPADS), 2013 International Conference on. IEEE, 2013.
- [6] Walther, John S. "A unified algorithm for elementary functions." Proceedings of the May 18-20, 1971, spring joint computer conference. ACM, 1971.
- [7] Llamocca-Obregón, Daniel R., and Carla P. Agurto-Ríos. "A fixed-point implementation of the expanded hyperbolic CORDIC algorithm." Latin American applied research 37.1 (2007): 83-91.
- [8] Whitehead, Nathan, and Alex Fit-Florea. "Precision and performance: Floating point and IEEE 754 compliance for NVIDIA GPUs." rm (A+ B) 21 (2011): 1-1874919424.
- [9] Bello, C., et al. "Relevador portátil de curvas IV de paneles fotovoltaicos como herramienta de diagnostico in situ de sistemas de generación fotovoltaica." Avances en Energías Renovables y Medio Ambiente 13 (2009): 77-83.
- [10] Raygoza, Juan José, et al. "Implementación en hardware de un sumador de punto flotante basado en el estándar IEEE 754-2008." e-Gnosis 7 (2009).

- [11] Bube, Richard. Fundamentals of solar cells: photovoltaic solar energy conversion. Elsevier, 2012.
- [12] Boudabous, Anis, et al. "Implementation of hyperbolic functions using CORDIC algorithm." Microelectronics, 2004. ICM 2004 Proceedings. The 16th International Conference on. IEEE, 2004.
- [13] Mano, M. Morris. Arquitectura de computadoras. Pearson Educación, 1994.
- [14] Floyd, Thomas L. Fundamentos de sistemas digitales. Vol. 7. Prentice Hall, 2006.