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 24 de abril 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, 24 de abril 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

El resumen es la síntesis de lo que aparecerá en el tesis. Tiene que ser lo suficientemente consiso y claro para que alguien que lo lea sepa qué esperar del resto de la tesis si la leyera completamente. Puede concluir con palabras clave, que son los temas principales tratados en el documento. El resumen queda fuera de la numeración del resto de secciones.

No se acostumbra utilizar referencias bibliográficas, tablas, o figuras en el resumen.

Palabras clave: palabras, clave, ...

Abstract

The same as before, but in English.

Keywords: word 1, word 2,

a mis queridos padres

Agradecimientos

El resultado de este trabajo no hubiese sido posible sin el apoyo de Thevenin, Norton, Einstein y mi querido amigo Ohm.

Adrián Ignacio Cervantes Segura

Cartago, 24 de abril 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 . . . 5
  - 1.6.1 Objetivo general . . . 5
  - 1.6.2 Objetivos específicos . . . 5
  - 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 . . . 14
- 2.5 Referencias bibliográficas . . . 14

3 Sistema de linealización 17
- 3.1 Algoritmo de CORDIC en software . . . 17
- 3.2 Linealizador con el algoritmo de CORDIC . . . 19
- 3.3 Coprocesador CORDIC . . . 20
- 3.4 Sistema de control para el coprocesador CORDIC . . . 24
- 3.5 Algoritmo de CORDIC en Verilog . . . 25
- 3.6 Simulación del algoritmo de CORDIC . . . 25

3.6.1 Simulacion del rango de convergencia del Linealizador CORDIC . . 26

4 Sistema de normalización **31**

4.1 Sistema de conversión y normalización 31
4.2 Convertidor punto flotante - punto fijo y normalizador 33
4.3 Control 35
4.4 Sistema de conversión-normalización en verilog 36
4.5 Simulación y verificación del convertidor-normalizador 37

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. . . . 4

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

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 Algoritmo de CORDIC en hardware . . . 21
3.5 Signo para cada iteración componente Xi , Yi , Zi . . . 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 Rango de convergencia circuito CORDIC con 8 iteraciones . . . 27
3.9 %Error rango de convergencia circuito CORDIC con 8 iteraciones . . . 27
3.10 Rango de convergencia circuito CORDIC con 12 iteraciones . . . 28
3.11 %Error rango de convergencia circuito CORDIC con 12 iteraciones . . . 28
3.12 Rango de convergencia circuito CORDIC con 15 iteraciones . . . 29
3.13 %Error rango de convergencia circuito CORDIC con 15 iteraciones . . . 29

4.1 Bloque general del convertidor puto flotante a punto fijo y normalizador . . . 31
4.2 Sistema de convesión, normalización, señales de datos y control . . . 32
4.3 conversión y normalización . . . 33
4.4 Normalizador . . . 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 del circuito . . . 37
4.8 %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 del circuito . . . 39
4.10 %Error entre la conversión-normalización de tensión V_{pv} teórica y del circuito 39

Índice de tablas

2.1 Modelos para un PV 10
2.2 Sistema de coordenadas unificado (CORDIC) 12

3.1 Tabla de signo para cada iteracion componente Xi , Yi , Zi 22

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 varia, 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 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, 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.

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.
- 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: algoritmo de CORDIC, estándar IEEE 754 y operaciones en punto fijo. Posteriormente, se utilizara 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.

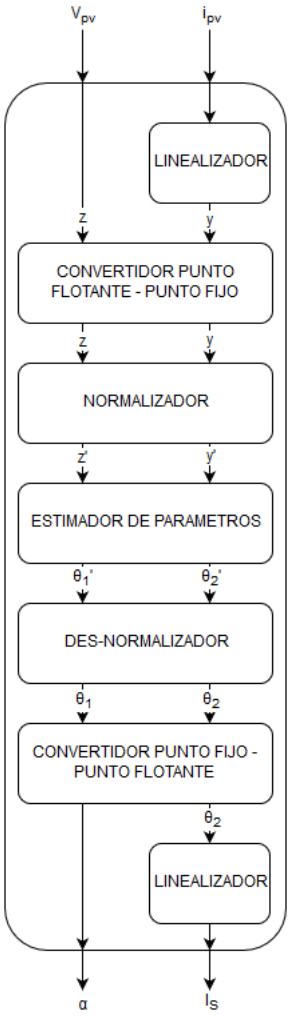

Figura 1.1: Diagrama de solución para el sistema de aumento de eficiencia de paneles fotovoltaicos

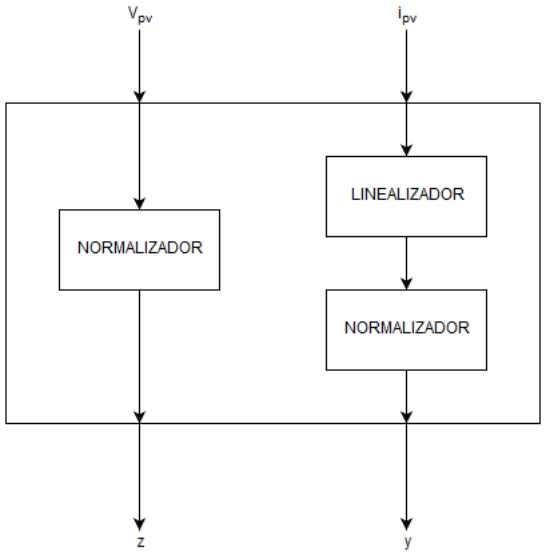

Figura 1.2: Diagrama de solución para el sistema de de Linealización-Normalización.

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 realizaran las pruebas en una FPGA para comprobar el funcionamiento del hardware.

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 utilizados 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. Finalmente, en el capítulo 6 se ofrecen conclusiones del proyecto, y recomendaciones.

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

Las celdas solares 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 el 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 fotocorriente, esta es directamente proporcional a la radiación del sol. Debido al proceso de fabricación del panel, posee un comportamiento exponencial y no lineal muy similar al de un diodo [1].

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

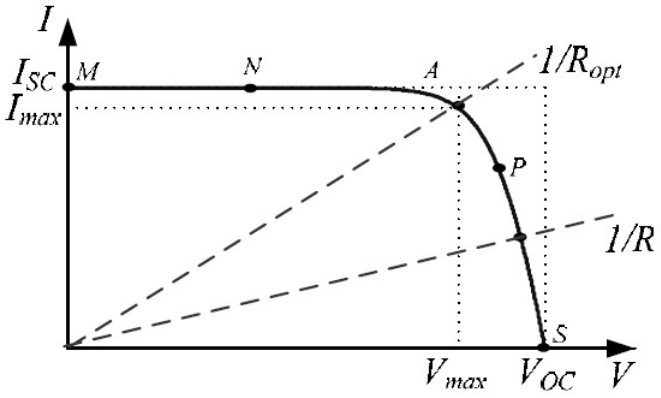

Figura 2.1: Curva Corriente(A)-Tensión(V)

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)

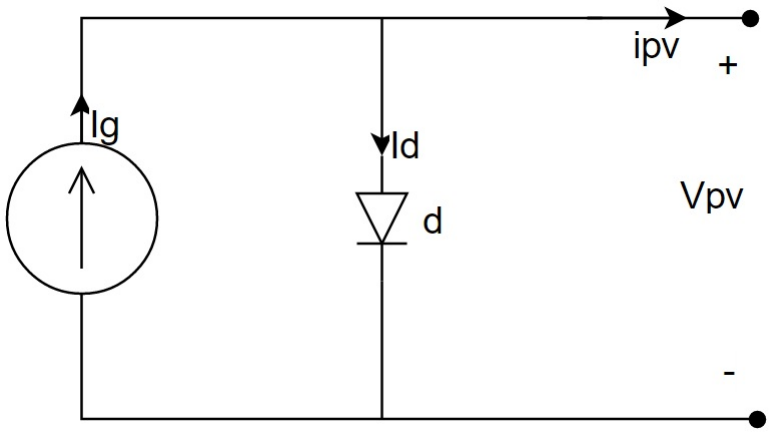

Figura 2.2: Modelo simple ideal para un PV

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 (Is) y la foto corriente (Ig)
- Perdidas debidas al flujo de corriente (Rs) y perdidas con referencia a tierra (Rp).
- Un parámetro n que será el numero de celdas en análisis.

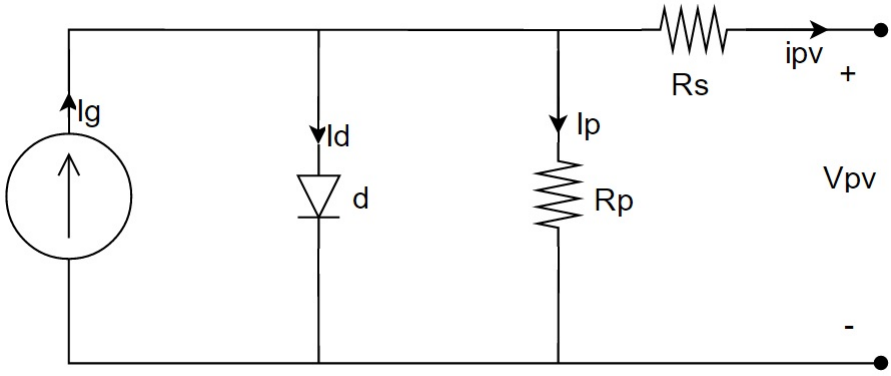

Figura 2.3: Modelo con perdidas para un PV

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

$$i_{pv} = I_s - i_d + i_p \tag{2.1}$$

$$I_g = 2i_{pv} + \frac{V_{pv} + i_{pv}R_s}{R_p} - I_s + I_s e^{\frac{V_{pv}+i_{pv}V_{pv}}{nv_t}} \tag{2.2}$$

Despejando i_{pv}

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

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

- *Ig*: Corriente generada debido al efecto fotoeléctrico
- *id*: Corriente de pérdida debido a la juntura p-n
- *ip*: Corriente de pérdida de naturaleza resistiva

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

- *Ig*: depende de la Irradiancia (S), pero no depende de la tensión en las terminales del generador fotovoltaico (*Vpv*)
- *ip* e *id*: dependen de la tensión *Vpv*
- *ip*: Depende de la temperatura (T)

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

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

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

De manera general se tiene:

$$i_{pv}\left(V_{pv}\right) = G_p V_{pv} + G_p i_{pv} R_s \tag{2.5}$$

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

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

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

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

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

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

si $I_g = KS >> I_s$

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

$$z = V_{pv} + i_{pv} R_s \tag{2.11}$$

posteriormente a un proceso de calculo de parámetros se tiene:

$$\theta_1 = ln\left(I_s(T)\right) \tag{2.12}$$

$$\theta_2 = \alpha \tag{2.13}$$

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:

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

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

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

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:

$$\tan z = \frac{\sin z}{\cos z} \tag{2.17}$$

$$\tanh z = \frac{\sinh z}{\cosh z} \tag{2.18}$$

$$\exp z = \sinh z + \cosh z \tag{2.19}$$

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

donde:

$$x = \omega + 1 \tag{2.21}$$

$$y = \omega - 1 \tag{2.22}$$

2.3.2 Logaritmo natural utilizando el algoritmo hiperbólico de CORDIC

Para un $Ln\,(\omega)$ con el algoritmo de CORDIC, se debe calcular primeramente la $\tanh^{-1}\left(\frac{y}{x}\right)$ con las siguientes ecuaciones:

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

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

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

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:

$$0.106843 \leq \omega \leq 9.35947 \tag{2.26}$$

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:

$$0 \leq \omega \leq 1 \tag{2.27}$$

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:

- Signo
- Exponente
- Mantisa

para el formato *IEEE 754* 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 Referencias 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 CIBELEC 2005 (2005): 1-5.

[3] C. Meza, R. Ortega, "Control and estimation scheme for PV central inventers", in 24th International Conference on information, Comunication 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. Procee-

dings 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." rn (A+ B) 21 (2011): 1-1874919424.

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 calculo 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 calculo 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, 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:

$$Ln\,(T) = Ln\,(16T) - Ln\,(16) \tag{3.1}$$

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**-


    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 #resulado final del logaritmo natural


>>>
>>> T=0.2
>>> CordicLn(16*T) - math.log(16)
-1.6094370822397814
>>> math.log(0.2)
-1.6094379124341003
>>>
```

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 calculo 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 Linealizador con el algoritmo de CORDIC

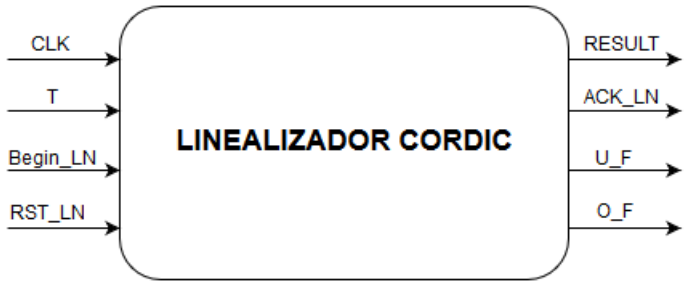

Figura 3.2: Bloque principal: Algoritmo de CORDIC en hardware

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

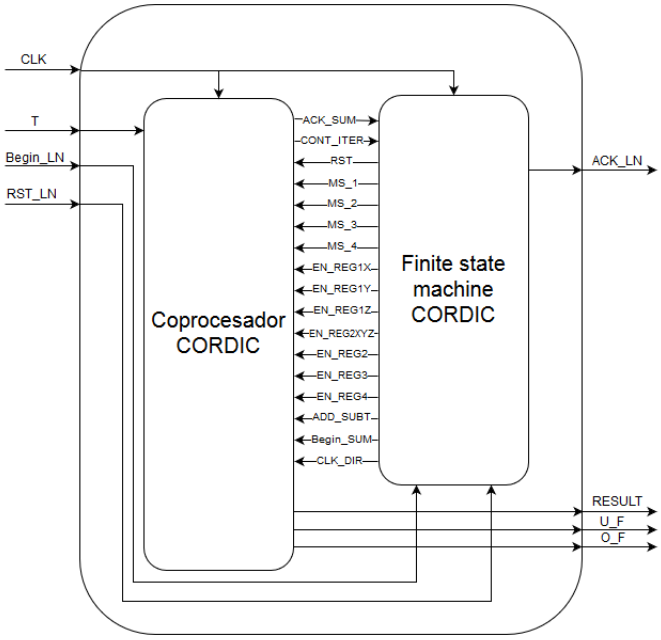

Figura 3.3: Coprocesador CORDIC y Control

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

- *Coprocesador Cordic*: En este realizan todas las operaciones requeridas por el algoritmo, se encarga del manejo de los datos en el calculo.
- *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 maquina de estados.
- *ACK_LN*: Indica que el calculo 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 iteracion, para que la maquina de estados pueda detenerse en el numero 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 selelccion 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 varios datos a la vez, sin embargo se debe tener buena sincronización para evitar datos erróneos a través del proceso de calculo. Por otro lado se utiliza el formato IEEE 754 con 32Bits, esto debido a que se requiere una adecuada precision en el calculo y un menor numero de iteraciones, repercutiendo en la velocidad del sistema.

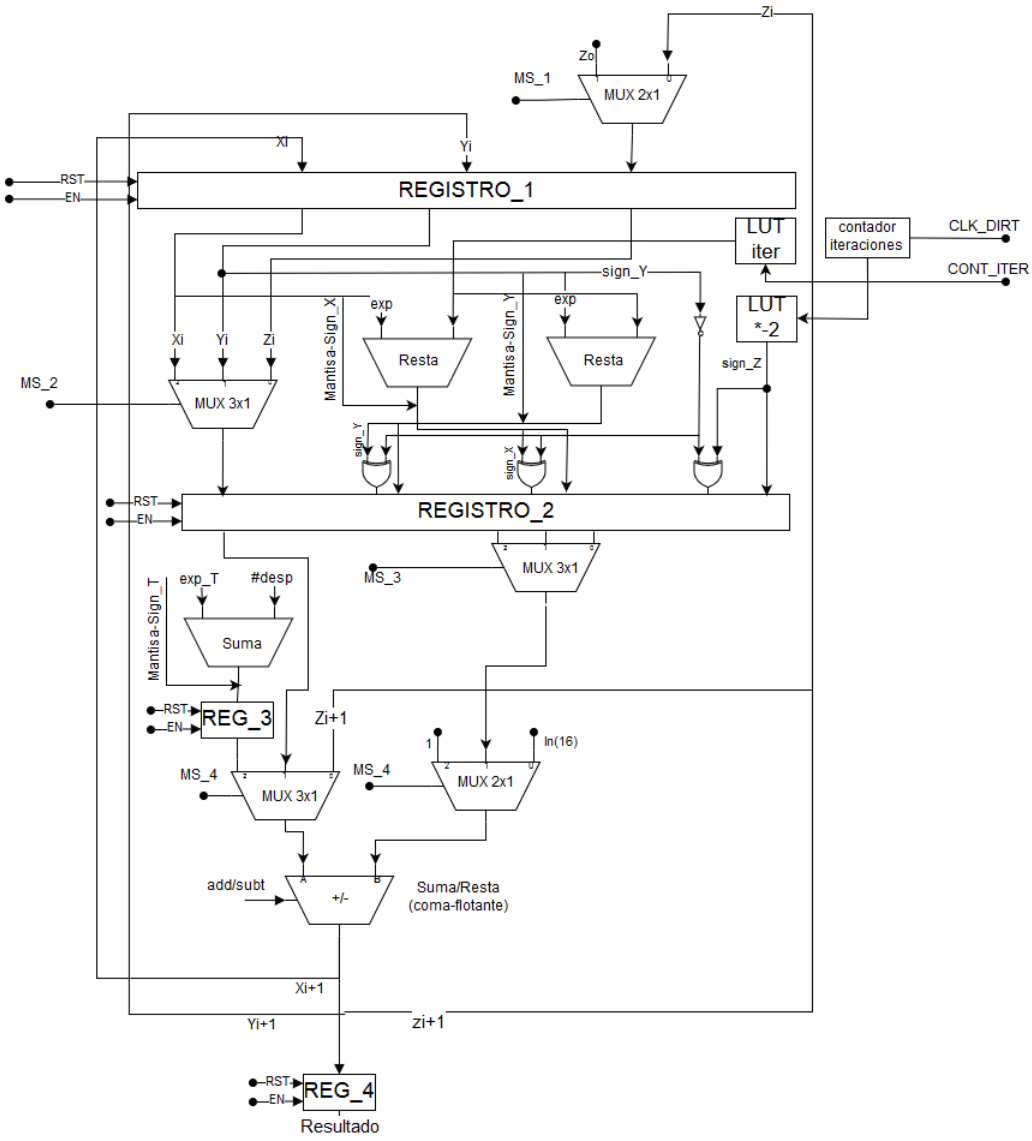

Figura 3.4: Algoritmo de CORDIC en hardware

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 ieraciones, 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 calculo 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 calculo 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 para cada iteracion componente Xi , Yi , Zi

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

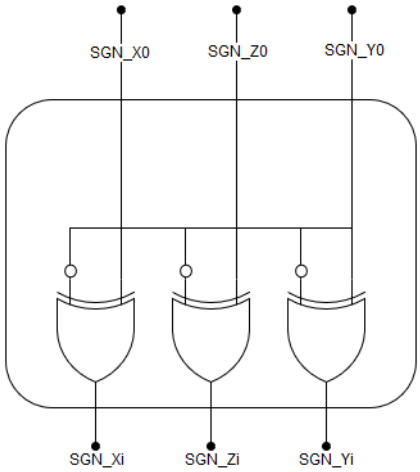

Figura 3.5: Signo para cada iteración componente Xi , Yi , Zi

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 memorias ROM's, de manera que puedan ser accesados en cualquier momento que sean

requeridos, dentro de las ROM's se dispone:

- LUT_Z: Contiene almacenados los valores de $-2arctanh\left(2^{-i}\right)$, 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 calculo 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 calculo toma primeramente los valores que se necesitan para el calculo de X_i, posteriormente se realiza el calculo y se almacena el resultado en el *Registro 1*, seguidamente se procede con el calculo 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 realiza la resta $Z_i - Ln\left(16\right)$ para contrarrestar el efecto del escalado aplicado al argumento (T) al inicio del calculo del logaritmo natural, por ultimo el resultado final de Z_i contiene el valor de $Ln\left(T\right)$, este se almacena en el *Registro 4*.

3.4 Sistema de control para el coprocesador CORDIC

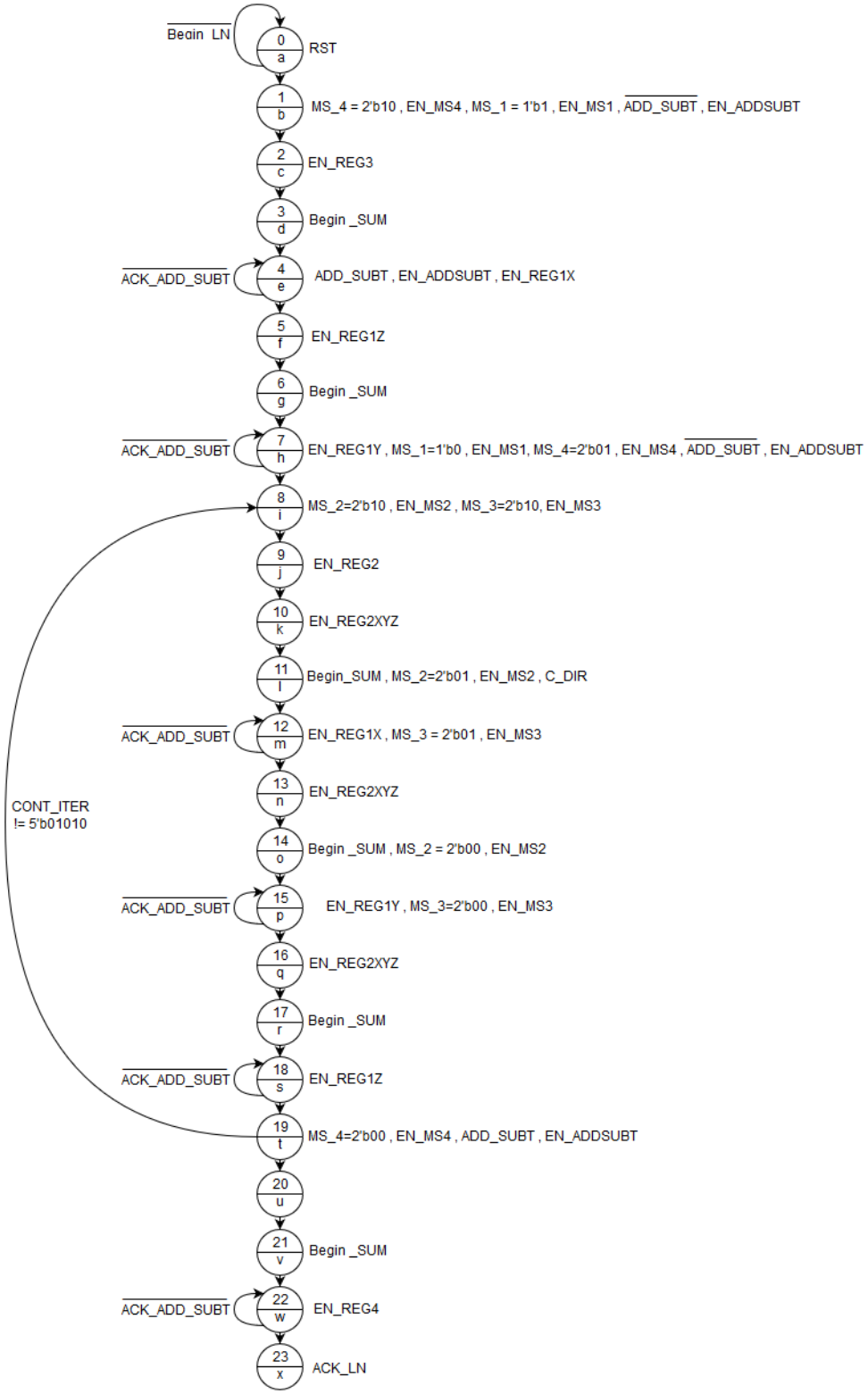

Figura 3.6: Maquina de estados finitos para la arquitectura de CORDIC

EL sistema de control requiere de mucha sincronía, ya que los datos debe ser almacenados de manera correcta y estar listos cuando se requiere por otro segmento del coprocesador CORDIC, para esto se diseñó una maquina de estados finitos, donde inicialmente se calculan los valores iniciales de X_0, Y_0 y Z_0, posteriormente la maquina brinda las señales de control requeridas con la secuencia de calculo 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 numero de iteración en el que se encuentra, de manera que cuando se llega a la iteración N definida con anterioridad, se finaliza el calculo.

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.

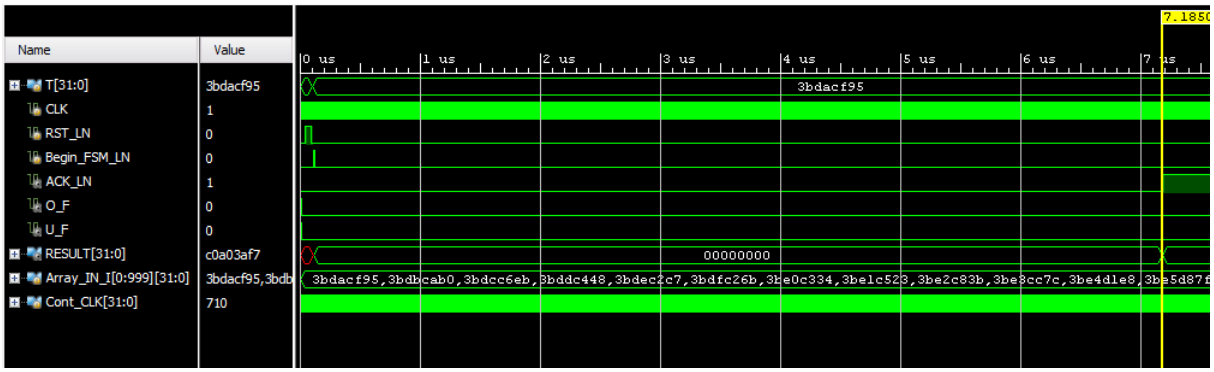

Figura 3.7: Simulación del linealizador en implementado en Verilog

3.6 Simulación del 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:

$$Ln\,(T) \tag{3.2}$$

donde el argumento T es:

$$T = e^{-x} \tag{3.3}$$

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.

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

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

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 Simulacion del rango de convergencia del 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 calculo 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.

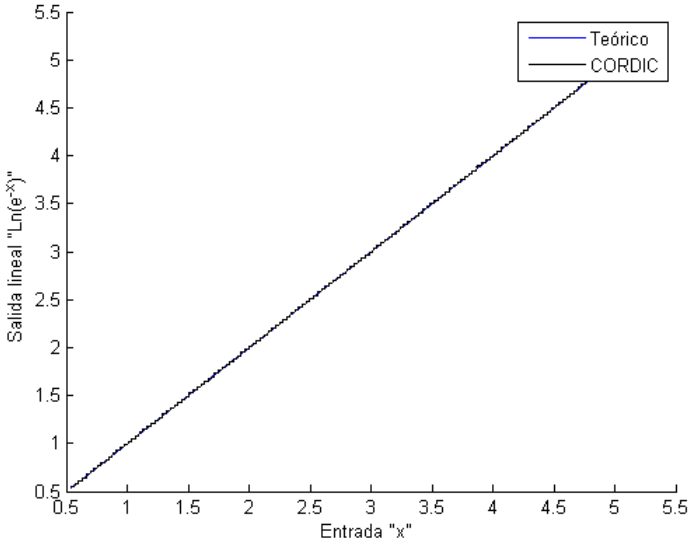

Figura 3.8: Rango de convergencia circuito CORDIC con 8 iteraciones

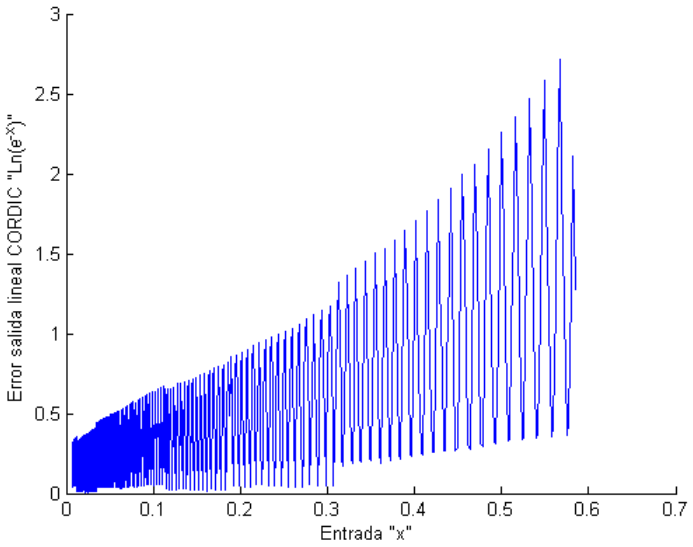

Figura 3.9: %Error rango de convergencia circuito CORDIC con 8 iteraciones

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%.

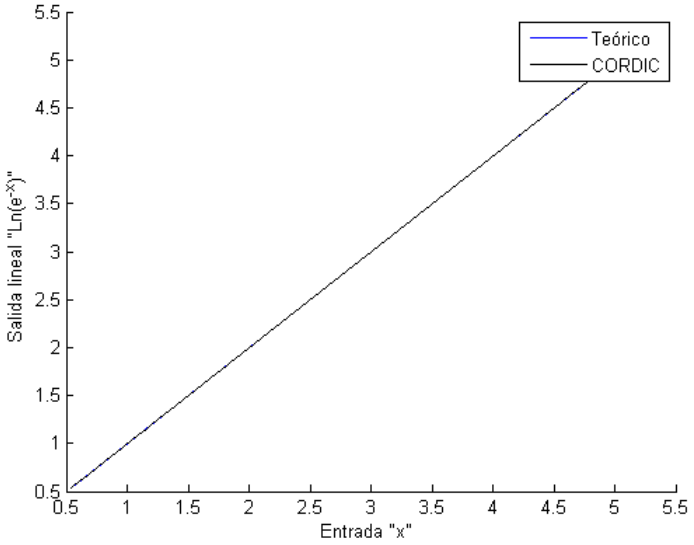

Figura 3.10: Rango de convergencia circuito CORDIC con 12 iteraciones

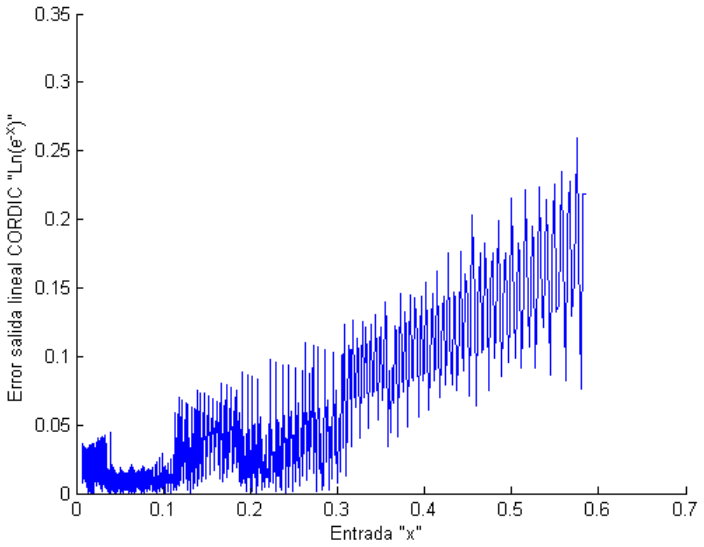

Figura 3.11: %Error rango de convergencia circuito CORDIC con 12 iteraciones

Una mejor aproximación se puede lograr utilizando 12 iteraciones en el calculo del logaritmo natural, en la figura 3.10 se pueden observar los resultados obtenidos, se requieren 675 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%.

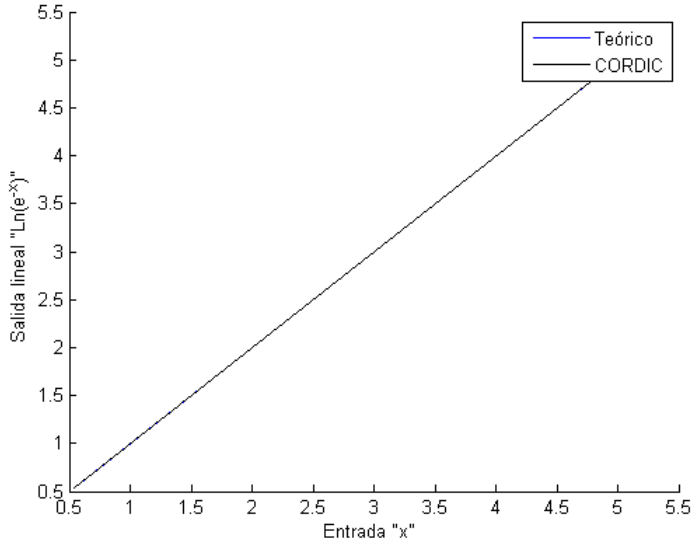

Figura 3.12: Rango de convergencia circuito CORDIC con 15 iteraciones

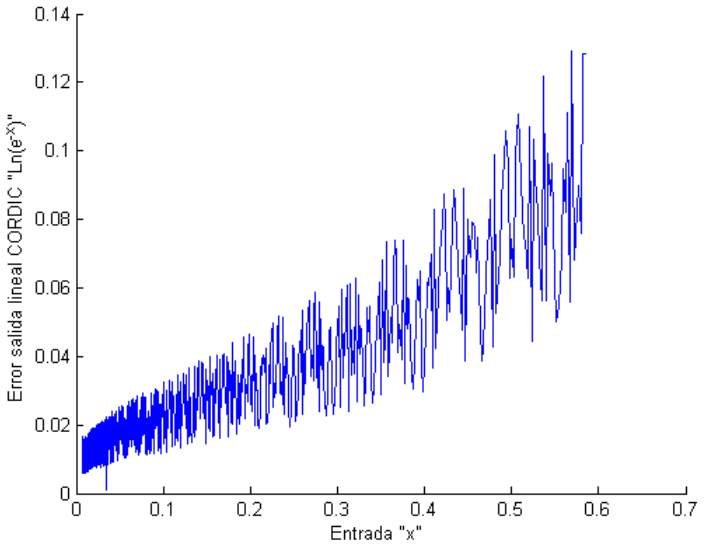

Figura 3.13: %Error rango de convergencia circuito CORDIC con 15 iteraciones

Utilizando 12 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%.

Capítulo 4

Sistema de 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

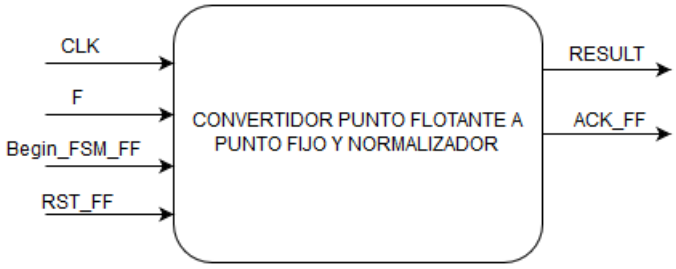

Figura 4.1: Bloque general del convertidor puto flotante a punto fijo y normalizador.

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.

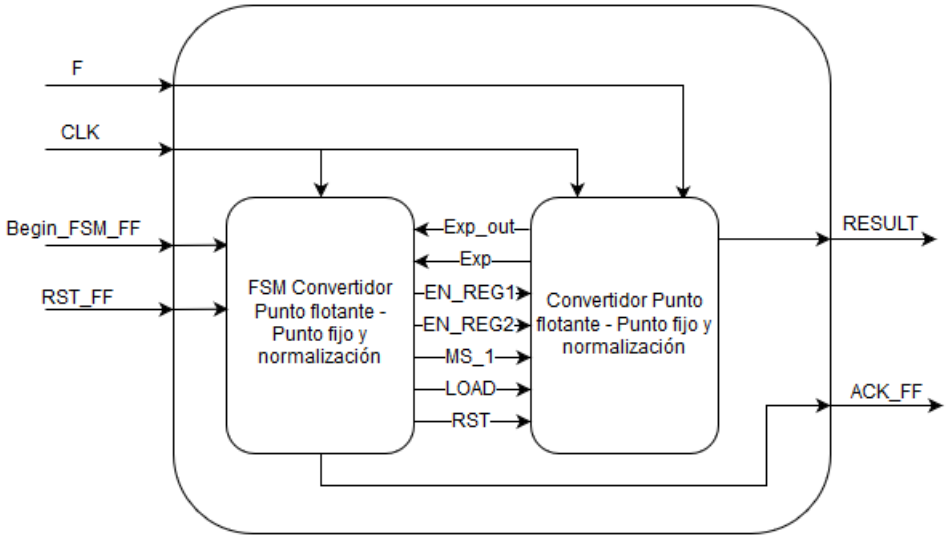

Figura 4.2: Sistema de conversión, normalización, señales de datos y control.

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 maquina de estados que inicia la secuencia.
- *RST_FF*: Este se encarga reestablecer 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

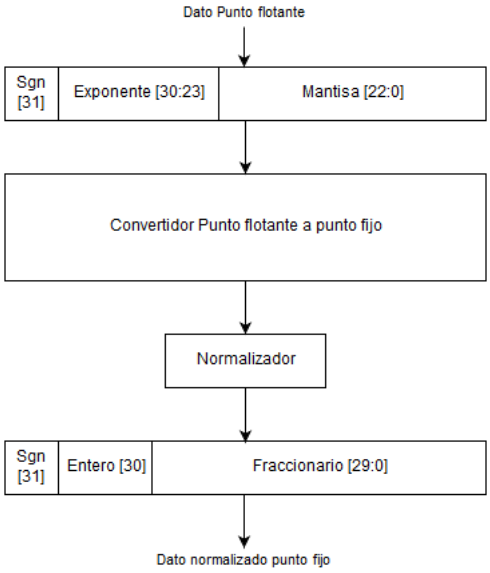

Figura 4.3: Proceso de conversión y normalización

En la figura 4.3 se puede observar el diagrama de solución que se utilizó para el desarrollo del convertidor-normalizador , primeramente ingresa el numero 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 tension 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.

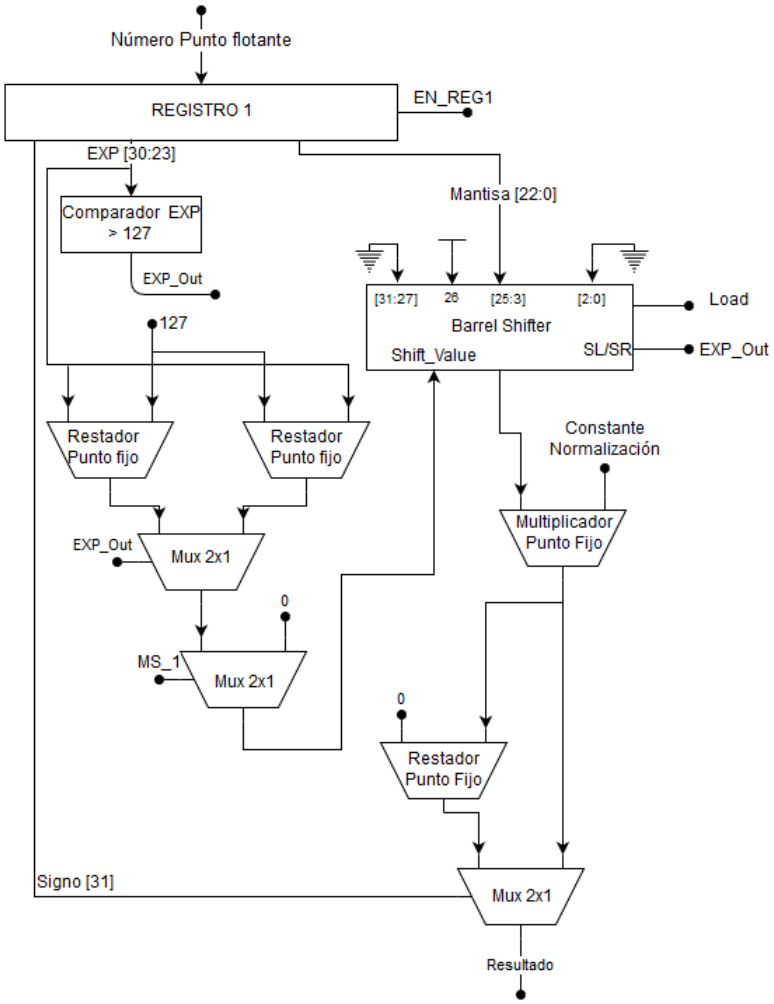

Figura 4.4: Conversión punto flotante a punto fijo y normalización

Como se observa en la figura 4.4, la etapa de conversión de formatos contiene un comparador, este se utiliza para saber si el numero en formato IEEE 754, contiene un exponente mayor o menor que 127, si el numero 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 calculo en la etapa de conversión, la direccion 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 retomara en otra etapa.

Un dato normalizado requiere una division entre el maximo valor que se puede procesar, sin embargo en un circuito digital las divisiones se tornar complicadas y requieren de mucha area, 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:

$$C_{norm} = \frac{1}{Valor_{MAX}} \tag{4.1}$$

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 varia para cada circuito:

$$Cv_{norm} = \frac{1}{18.1} = 0.055248618 \tag{4.2}$$

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

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 conversion-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

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.

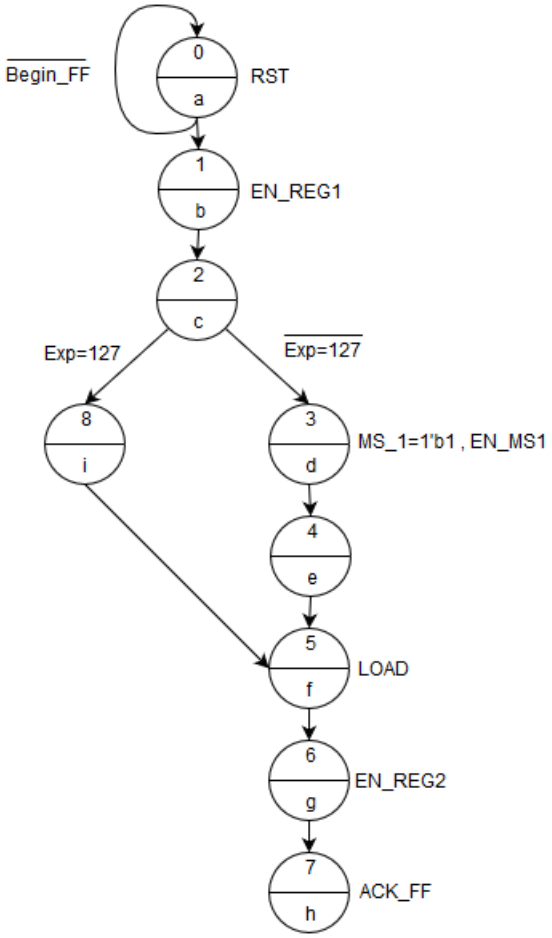

Figura 4.5: Maquina de estados finita para el convertidor-normalizador

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.

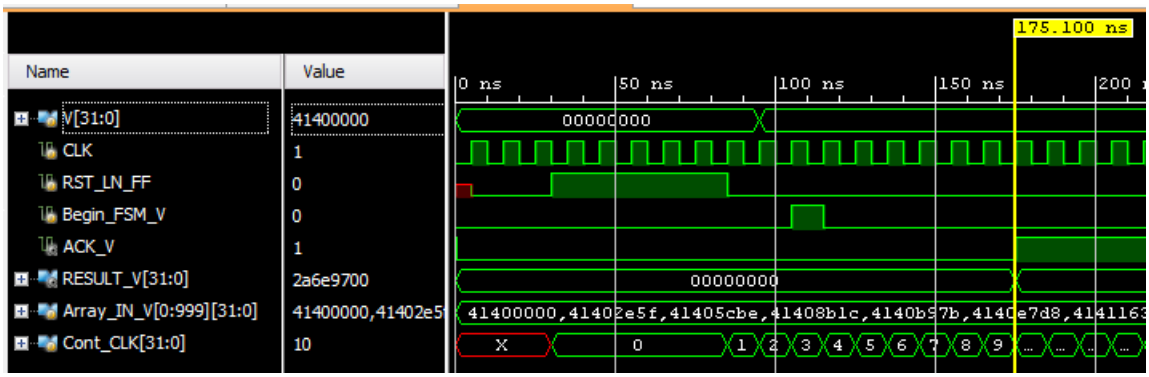

Figura 4.6: Simulación del circuito de conversión y normalización

4.5 Simulación y verificación del 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 segun el modelo del panel, las pruebas de esta unidad se realizaron con valores de corriente i_{pv} y valores de tension V_{pv}.

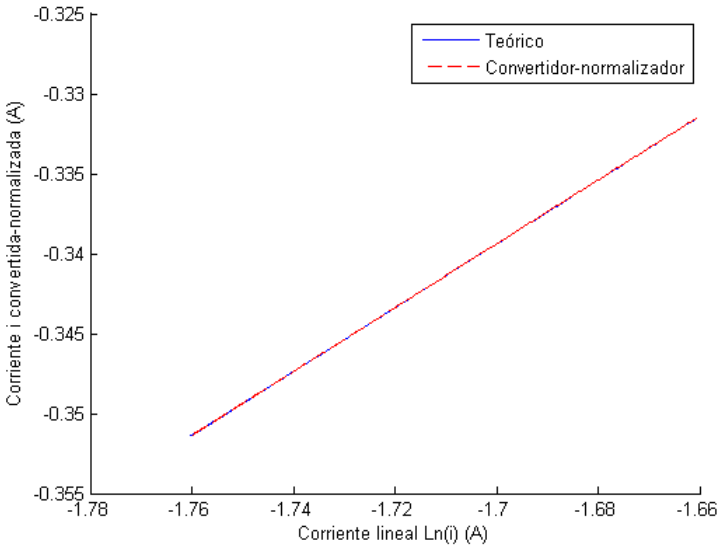

Figura 4.7: Comparación entre la conversión-normalización de corriente i_{pv} teórica y del circuito

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.

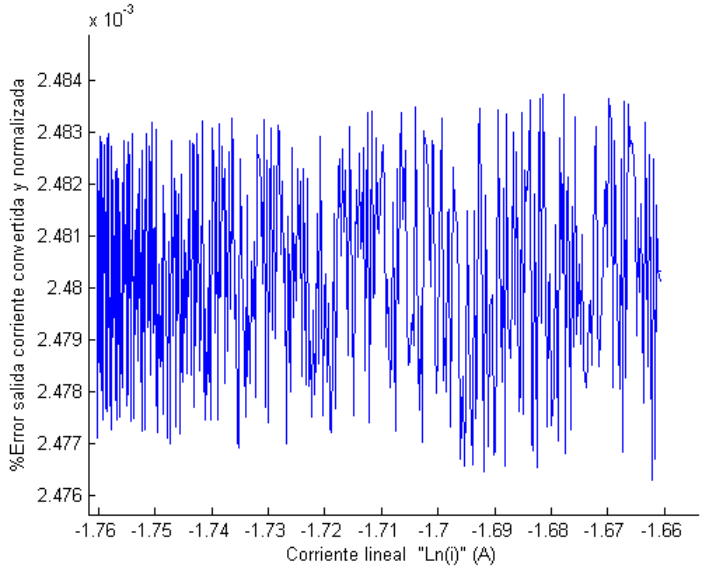

Figura 4.8: %Error entre la conversión-normalización de corriente i_{pv} teórica y del circuito

En la figura 4.8 se puede observar el error entre la conversión de la corriente normalizada teórica y experimental, con un error porcentual máximo de 0,0024837% y un error promedio de 0,002478%

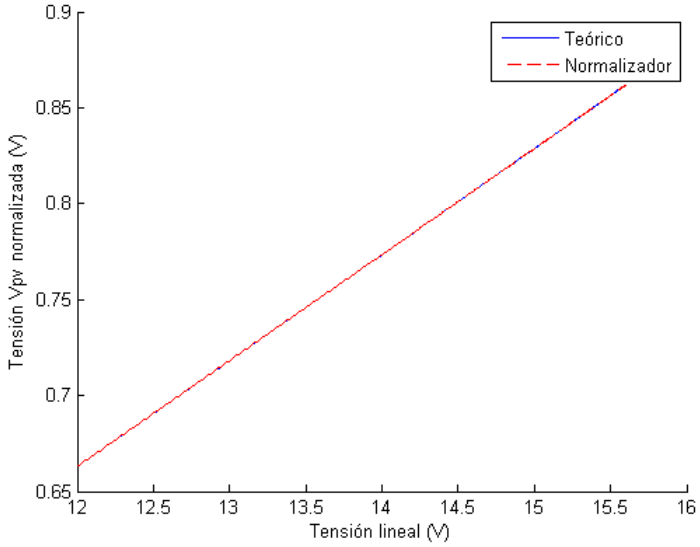

Figura 4.9: Comparación entre la conversión-normalización de tensión V_{pv} teórica y del circuito

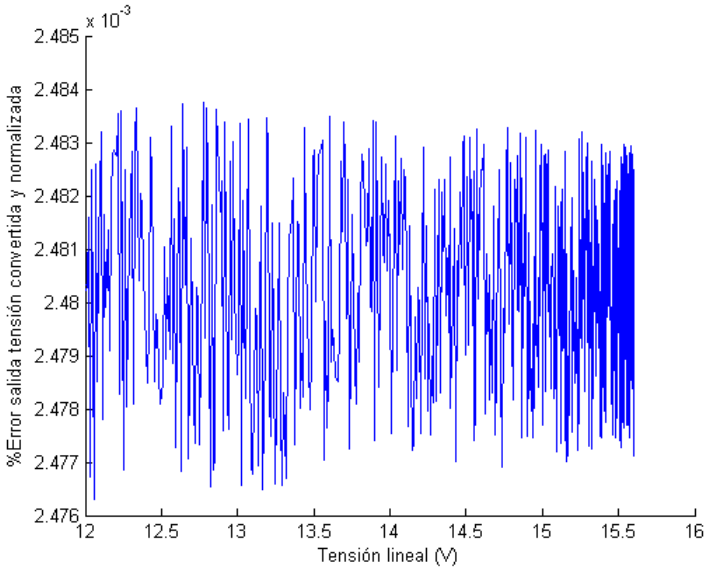

Figura 4.10: %Error entre la conversión-normalización de tensión V_{pv} teórica y del circuito

El análisis utilizado 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 tension de entrada y su normalizacion tanto teorico como experimental, de la misma manera se puede hacer el calculo 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 maquina de estados para realizar la conversión y la normalización, la maquina 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