Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Sistema de simulación de microrredes eléctricas aisladas basado en FPGA

Documento de tesis sometido a consideración para optar por el grado académico de Maestría en Electrónica con Énfasis en Sistemas Empotrados

Edgar Mauricio Brenes Gutiérrez

13 de marzo de 2018

Declaro que el presente documento de tesis ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos y resultados experimentales 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 tesis realizado y por el contenido del presente documento.

Edgar Mauricio Brenes Gutiérrez

Cartago, 13 de marzo de 2018

Céd: 1-0981-0660

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Tesis de Maestría
Tribunal Evaluador

Tesis de maestría defendida ante el presente Tribunal Evaluador como requisito para optar por el grado académico de maestría, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Dr. Alfonso Chacón Rodríguez Profesor Lector	MSc. Luis Diego Murillo Profesor Lector

Dr. Carlos Meza Benavides
Profesor Asesor

Los miembros de este Tribunal dan fe de que la presente tesis de maestría ha sido aprobada y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica.

Cartago, 13 de marzo de 2018

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Tesis de Maestría
Tribunal evaluador
Acta de evaluación

Tesis de maestría defendida ante el presente Tribunal Evaluador como requisito para optar por el grado académico de maestría, del Instituto Tecnológico de Costa Rica.

Estudiante: **Ing. Edgar Brenes Gutiérrez**

Nombre del Proyecto:
"Sistema de simulación de microrredes eléctricas aisladas basado en FPGA"

Miembros del Tribunal

Dr.-Ing. Alfonso Chacón Rodríguez
Profesor lector

M.Sc. Luis Diego Murillo Soto
Profesor lector

Dr. Carlos Meza Benavides
Director de Tesis

Los miembros de este Tribunal dan fe de que la presente Tesis de Maestría para optar por el grado académico de Máster en Electrónica con Énfasis en Sistemas Empotrados y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica.

Nota Final de Tesis: 95

Cartago, 12 de abril de 2018

Resumen

La necesidad de disminuir las emisiones de dióxido de carbono para alcanzar carbono neutralidad y el incremento en la demanda de energía eléctrica hacen necesario la utilización masiva de tecnologías de generación eléctrica de bajas emisiones e impacto ambiental. Las tecnologías de generación que tienen un menor impacto ambiental son aquellas que dependen de flujos de energía renovable (radiación solar y viento), son de pequeño tamaño y son utilizadas en el mismo sitio en donde se requiere que la energía sea consumida. El uso de este tipo de tecnología ocasiona cambios en la forma en que la energía eléctrica se gestiona y produce. Una de las formas de gestión más eficiente y flexible de fuentes de generación distribuida es por medio de microrredes. Una microrred es una colección de elementos de generación, consumo y almacenamiento eléctrico que puede operar de forma independiente o coordinada con otra red eléctrica.

Con la incorporación de dichos elementos en la red eléctrica se crea la necesidad de tener herramientas que permitan evaluar el diseño y los mecanismos de control necesarios para asegurar la calidad de la energía producida en las microrredes.

En este trabajo se propone y describe una arquitectura para implementar un circuito digital que permita simular una microrred compuesta por diferentes generadores solares fotovoltaicos y una carga resistiva. La arquitectura aprovecha la posibilidad que da un FPGA de diseñar un circuito que ejecute la mayoría de operaciones matemáticas de forma paralela para así obtener el mínimo tiempo de ejecución en la simulación. Se muestran los resultados obtenidos en diferentes pruebas realizadas al circuito implementado, donde se simulan microrredes con uno, cuatro y diez paneles solares.

Palabras clave: microrred, simulación, FPGA.

Abstract

The need to reduce carbon footprint in order to achieve carbon neutrality along with the increasing demand for electric enegy make it necessary to massively use low emission and low environmental impact electric generation technologies. The generation technologies that have a lower environmental impact are those that depend on renewable energy flows (solar radiation and wind), they are small in size and are used in the same place where energy is required to be consumed. The use of this type of technology changes the way in which electrical energy is managed and produced. One of the most efficient and flexible forms of management of distributed generation sources is through microgrids. A microgrid is a collection of electrical generation, consumption and storage elements that can operate independently or in coordination with another electrical network.

With the incorporation of these elements in the electrical network, the need is created to have tools that allow evaluating the design and control mechanisms necessary to ensure the quality of the energy produced in the microgrids.

In this work an architecture is proposed and described to implement a digital circuit that allows to simulate a micro-grid composed of different photovoltaic solar generators and a resistive load. The architecture takes advantage of the possibility given by an FPGA to design a circuit that executes the majority of mathematical operations in parallel in order to obtain the minimum execution time in the simulation. The results obtained in different tests executed in the implemented circuit are shown, where microgrids with one, four and ten solar panels were simulated.

Keywords: microgrid, simulation, FPGA.

a mi hijo Diego y mi esposa Shirley

Agradecimientos

A mi familia por el apoyo que me han dado. Y al profesor Carlos Meza por haberme propuesto un tema tan interesante para la tesis y por las recomendaciones a lo largo de la investigación y desarrollo de este trabajo.

Edgar Mauricio Brenes Gutiérrez

Cartago, 13 de marzo de 2018

Índice general

Índice de figuras

Índice de tablas

1. Introducción.

En la actualidad la mayoría de las redes eléctricas están compuestas por un conjunto de pocos generadores de energía centralizados y por múltiples consumidores pasivos. En estas redes eléctricas el flujo de energía es unidireccional, fluyendo desde el generador hasta los consumidores.

Debido a la creciente demanda de energía eléctrica sumada a la necesidad de disminuir las emisiones de dióxido de carbono se espera que en un futuro cercano las redes eléctricas estarán compuestas por un generador de energía principal junto con múltiples fuentes de energía renovable como paneles solares, generadores eólicos y otros junto con dispositivos de almacenamiento de energía del lado de los consumidores. Cuando estos consumidores activos generen más energía que la que consumen podrán inyectar energía a la red eléctrica haciendo que el flujo de energía sea multidireccional.

A este mecanismo de generación se le denomina generación distribuida para autoconsumo (GDA). Una de la forma más práctica de implementar y gestionar la GDA es por medio de microrredes, que son sistemas que consisten en cargas interconectadas con diferentes generadores de energía distribuidos y que pueden operar en paralelo con la red eléctrica o de forma aislada [1].

La Comisión Europea [2] define una microrred como un pequeño sistema de distribución de energía que conecta múltiples consumidores a múltiples fuentes distribuidas de generación y almacenamiento. Una microrred puede típicamente proveer energía a una comunidad de hasta 500 hogares a un bajo nivel de voltaje.

La figura 1.1 muestra el diagrama de una microred típica, compuesta de varias fuentes de energía y almacenamiento distribuidas, varios consumidores y la conexión a un sistema de distribución.

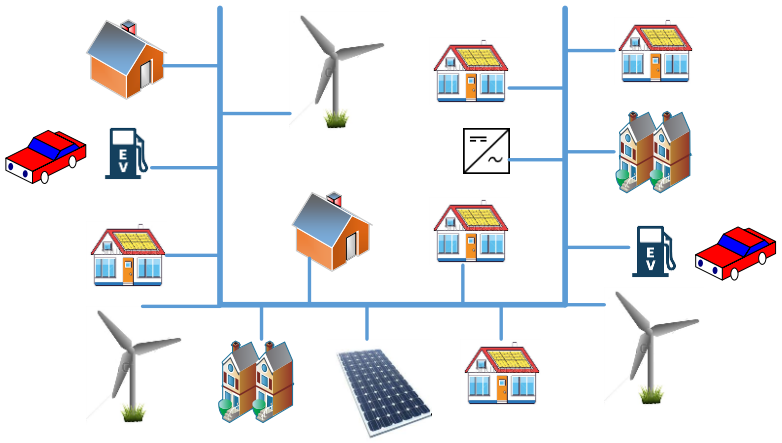

Figura 1.1 Diagrama de microred típica.

La inherente intermitencia y variabilidad de los dispositivos generadores de energías renovables complican la operación de las microrredes principalmente en el esfuerzo por asegurar la calidad y confiabilidad en el suministro de energía [3]. En una microrred se deben implementar mecanismos de control que permitan continuar con el suministro de energía y su calidad ante los cambios atmosféricos que la puedan afectar como lo sería un aumento en la nubosidad que haría que un panel solar genere menor energía, al igual que un cambio en la intensidad del viento va a afectar la cantidad de energía producida por un generador eólico. Estos mecanismos de control deberían por ejemplo limitar la cantidad de energía que se inyecta a la red eléctrica a pesar de un incremento en la intensidad del viento.

Este cambio de paradigma, pasar de un único generador de energía y múltiples consumidores pasivos a un generador principal y múltiples consumidores activos, trae nuevos retos en el modelado y simulación de estos nuevos sistemas de energía [4]. Se necesitan nuevas herramientas tanto para el diseño y prueba de microredes como para el desarrollo y verificación de estrategias de control de las microrredes.

1.1 Antecedentes

Las soluciones actuales para la simulación de microrredes se pueden agrupar en tres categorías: simuladores por software, emuladores de hardware y la tercera "*hardware en el lazo*".

1.1.1 Simuladores por software

En [4] se presenta una simulación implementada en Anylogic [5], una herramienta de modelado y simulación desarrollada por The Anylogic Company que se basa en tres paradigmas: dinámica de sistemas, eventos discretos y simulación basada en "agentes". En este caso el comportamiento complejo de un sistema completo se reproduce a través del comportamiento individual de los agentes y del análisis de los efectos en el sistema interconectado. Con este método modelaron el sistema en una estructura de dos capas representando la interacción física y lógica entre los componentes de la microrred. La capa lógica representa la comunicación de los agentes en tiempo real y la capa física representa los resultados técnicos de los cálculos del flujo de potencia.

Una simulación utilizando el programa FICO Xpress-Optimizer [13] se presenta en [6]. Esta herramienta provee una variedad de algoritmos para resolver problemas lineales de gran escala y problemas cuadráticos. En este caso se simuló una microrred compuesta por un generador fotovoltaico, un generador térmico, un sistema de almacenamiento de energía y una estación de carga de vehículos eléctricos. El objetivo de la simulación fue obtener la máxima utilización de electricidad producida por el generador fotovoltaico y el generador térmico y disminuir los efectos de la carga generados por el cargador de vehículos eléctricos. Con este método se simuló un día de verano y un día de invierno en intervalos de 15 minutos.

Ha habido también muchas soluciones basadas en Matlab. En [7] se simula una microrred utilizando dos tipos de simulación: transiente y estable. La simulación transiente se ejecuta utilizando Matlab/SimPowerSystems, el cual provee bibliotecas y herramientas para la simulación de sistemas de potencia eléctricos. La simulación estable se hace con DIgSILENT, un programa para el análisis de sistemas de potencia. La simulación de transitorios utiliza representación escalar y describe la dinámica con ecuaciones diferenciales, utiliza un método de modelado detallado y un nivel de simulación de microsegundos. La simulación estable utiliza representación vectorial y describe el sistema con ecuaciones algebraicas, utiliza un método de modelado simplificado y un nivel de simulación de milisegundos. Ambos métodos mostraron consistencia pero también mostraron las diferencias de los dos tipos de simulación.

En [8] se simula una microrred para demostrar la efectividad y precisión de un modelo estocástico de planificación de energía. En este caso la formulación del problema principal y

sus restricciones se implementaron en Matlab y se resolvieron por medio de IBM ILOG CPLEX Optimizer [9], el cual genera las variables de control óptimas, que son pasadas a OpenDSS para correr la simulación y chequear las restricciones, OpenDSS [14] es una herramienta de simulación de distribución de potencia utilizada en el análisis de integración de fuentes de potencia distribuidas y modernizaciones de las redes.

1.1.2 Emuladores

Un emulador es un dispositivo electrónico que representa el comportamiento eléctrico real de un dispositivo de una microrred. Un emulador se diseña para que se comporte eléctricamente como lo haría cuando se conecta a un generador, una carga o una batería real [10].

Una microrred basada en emuladores se presenta en [10], desarrollada en el Instituto de Investigación en Energía de Cataluña (IREC) en Barcelona, España. Esta microrred está compuesta de tres diferentes unidades configurables:

- Generador: emula diferentes tipos de generadores como generador eólico y generador solar, reproduciendo el comportamiento real y su naturaleza variable y dependencia de factores climatológicos externos.
- Carga: emula el comportamiento real de diferentes tipos de cargas utilizando diferentes perfiles.
- Almacenamiento de energía: emula un sistema de almacenamiento como batería, super capacitor, batería inercial o un vehículo eléctrico con capacidad "*Vehicule-to-grid*".

Con estas unidades configurables pueden reproducir diferentes escenarios sin tener que esperar por las condiciones climáticas propicias.

1.1.3 Hardware-in-the-loop

La simulación "*Hardware-in-the-loop*" (HIL) es una técnica con la cual se provee todo el estímulo necesario para ejercitar completamente la unidad bajo prueba y de esta forma se le hace creer que está conectada a un sistema real.

En [11] un sistema HIL se utiliza como un nuevo método para desarrollar y probar los algoritmos de control y estrategias de operación de una microrred. El sistema se compone de un simulador digital de tiempo real (RTDS) de la microrred, un prototipo del sistema de control bajo prueba y un emulador de comunicación para interfazar a ambos.

Para el RTDS se utilizan tres tipos de tarjetas: una tarjeta de salidas analógicas con dos convertidores digital a analógico de 16 bits, una tarjeta de entradas analógicas con dos convertidores analógico a digital de 16 bits, y una tarjeta de entradas y salidas digitales. Los resultados mostraron que el sistema HIL era útil y efectivo para probar los algoritmos de control y estrategias de operación de una microrred en tiempo real.

En [12] se presenta otro sistema HIL, esta vez basado en NI-PXI. El RTDS es un PXI (PCI eXtensions for Instrumentation), la cual es una plataforma de medición y automatización basada en PC y fabricado por National Instruments. En este caso también se demostró la viabilidad del sistema de simulación HIL y la estrategia de control propuesta [11].

1.2 Objetivos

El objetivo principal de este trabajo consiste en el diseño de la arquitectura de un sistema digital que permita simular el comportamiento de una microrred para su evaluación ante distintos escenarios con una precisión igual y un tiempo de simulación menor a un entorno de simulación ejecutado en un procesador de propósito general.

De esta forma, en este proyecto se diseñó un circuito que calcule la corriente generada por paneles solares en una microrred y que calcule el voltaje en la carga. Así mismo se busca implementar en un FPGA un circuito basado en la arquitectura propuesta para validar el diseño.

El circuito a implementar debe ser fácilmente reconfigurable para que se puedan simular diferentes microrredes donde se pueda activar uno o varios paneles solares, y debe permitir configurar los parámetros de cada panel solar individualmente, es decir cada panel solar puede tener diferentes valores en sus parámetros.

La arquitectura debe permitir incorporar en un futuro otros módulos de generadores sin tener que rediseñar el sistema. El sistema debe poder enviar los resultados a un PC para su análisis.

1.3 Relevancia del proyecto

La creciente preocupación por energías limpias favorece la integración de microrredes en la red eléctrica. Esta integración requiere de nuevas herramientas para su diseño y control. Con este trabajo de investigación se propone un sistema que permite simular microrredes para evaluar su desempeño ante distintos escenarios.

1.4 Contribuciones del proyecto

El presente proyecto permitió la implementación de un sistema de simulación de microrredes altamente configurable el cual determina las corrientes y voltajes en distintos puntos de una microrred compuesta por diferentes generadores fotovoltaicos y baterías. El sistema permite simular en forma paralela un sistema complejo cuyos fenómenos ocurren de forma concurrente.

Se diseñó e implementó un procesador de aplicación específica que se puede programar para simular diferentes microrredes. A este procesador se le pueden agregar de manera sencilla más submódulos que permitirán simular microrredes con mayor diversidad de componentes.

Se hicieron varios módulos que se pueden reutilizar para implementar modelos de otras fuentes de energías renovables que están fuera del alcance de este proyecto como generadores eólicos. Estos módulos son: multiplicador de punto flotante de doble precisión que ejecuta una multiplicación en 18 ciclos de reloj y sumador de punto flotante de doble precisión que ejecuta una suma en 3 ciclos de reloj.

1.5 Supuestos del proyecto

Para realizar este trabajo se establecieron los siguientes supuestos.

- La microrred a simular opera de forma aislada de la red eléctrica.
- La carga de la microrred es resistiva.
- Las baterías operan como una fuente ideal.

1.6 Restricciones del proyecto

Para el presente trabajo se tuvieron las siguientes restricciones.

- El sistema de simulación se implementará en una tarjeta Zedboard.
- La tarjeta tiene un FPGA Xilinx XC7Z020-CLG484.
- El FPGA cuenta con 220 módulos DSP48E1 para implementar multiplicadores de forma más eficiente.
- La comunicación entre el PC y la tarjeta de desarrollo para descargar el programa es a través de un puerto USB y el programa Vivado.
- Los resultados de la simulación se envían de la tarjeta a un PC a través de un puerto serial y utilizando un programa de comunicación como el Tera Term.

1.7 Estructura del documento

El presente trabajo de investigación está compuesto por seis capítulos. En el capítulo 2 se describe el caso de estudio utilizado en el presente trabajo. En el capítulo 3 se describe la solución propuesta que nace como resultado de la presente investigación. El cuarto capítulo muestra los resultados obtenidos al implementar el sistema propuesto en un FPGA. El capítulo cinco compara los resultados obtenidos en la implementación con respecto a los resultados obtenidos con un PC.

El capítulo 6 presenta las conclusiones obtenidas al realizar el presente trabajo.

2 Caso de estudio.

La metodología de diseño del procesador de aplicación específica se basó en una micro red compuesta por N paneles solares y una carga resistiva como la que se muestra en la figura 2.1.

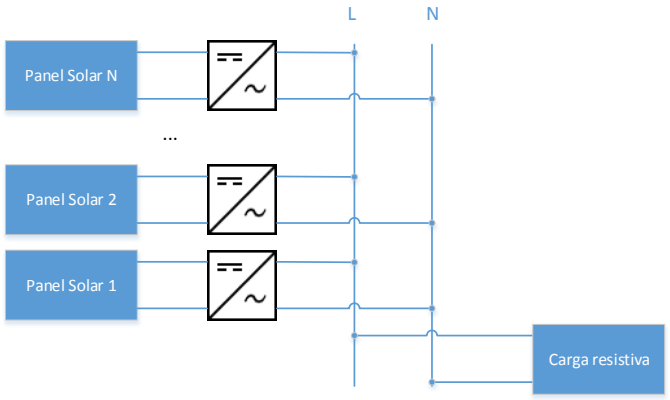

Figura 2.1 Caso de estudio.

Cada panel solar genera una corriente i_{pv} que se determina con la siguiente fórmula:

$$i_{pv} = i_g - i_s\left(e^{\frac{q(v+i_{pv}*R_S)}{kT}} - 1\right) - \frac{v+R_S i_{pv}}{R_p} \qquad (1)$$

En donde:

i_g = Corriente generada por la luz (fotocorriente)
i_s = Corriente de saturación inversa (corriente en oscuridad)
q = Carga del electrón = 1.6x10-19 As
k = Constante de Boltzmann = 1.38x10-23J/K
v = Voltaje
T = Temperatura en Kelvin
R_s = Resistencia serie
R_p = Resistencia paralelo

Para este trabajo se asume una resistencia serie R_S igual a cero y una resistencia en paralelo R_p infinita con los que la corriente generada por el panel solar se expresa con la siguiente ecuación.

$$i_{pv} = i_g - i_s(e^{\frac{qv}{kT}} - 1) \qquad (2)$$

Se utiliza la ecuación simplificada con el fin de calcular de una forma más sencilla la corriente en la microred. Sin embargo se puede cambiar el modelo y utilizar la ecuación (1) como una mejora futura al sistema sin tener que cambiar la arquitectura del mismo.

El circuito equivalente de cada panel solar con su inversor de voltaje se muestra en la figura 2.2.

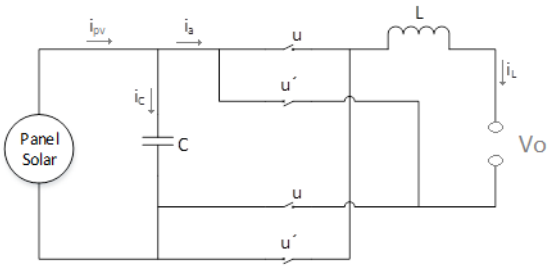

Figura 2.2 Circuito equivalente de panel solar e inversor

A partir de este circuito se obtuvieron las ecuaciones que representan la corriente y el voltaje en la carga R.

La corriente por el capacitor (i_c) es:

$$i_c = i_{pv} - i_a \quad (3)$$

i_{pv} es la corriente generada por la luz, i_a es la corriente por el inductor.

$$i_a = i_L\, u \quad (4)$$

en donde u puede tener valor de 1 o -1, dependiendo de la pareja de interruptores activos.

La corriente en el capacitor también se puede representar por:

$$i_c = C\frac{dvc}{dt} \quad (5)$$

Sustituyendo (5) y (4) en (3) se obtiene:

$$C\frac{dvc}{dt} = i_{pv} - ui_L \quad (6)$$

El voltaje en el capacitor está definido por:

$$uv_C = v_L + v_o \quad (7)$$

El voltaje en el inductor es:

$$v_L = L\frac{di_L}{dt} \quad (8)$$

El voltaje en la carga está dado por:

$$v_o = \sum_{j=1}^{j=n} i_{Lj}R \quad (9)$$

Sustituyendo (8) en (7) se obtiene:

$$L\frac{di_L}{dt} = uv_c - v_o \quad (10)$$

La ecuación (6) se puede expresar de la siguiente manera si se integran ambos lados de expresión:

$$v_C = \frac{1}{C}\int \left((i_g - i_s(\mathrm{e}^{\frac{qv}{kT}} - 1)) - ui_L\right) dt \quad (11)$$

La ecuación (10) se puede expresar de la siguiente manera si se integran ambos lados de la expresión:

$$i_L = \frac{1}{L}\int (uv_C - v_o)\, dt \quad (12)$$

A partir de las ecuaciones (9), (11) y (12) se diseñó el procesador de aplicación específica para la simulación de micro redes.

3 Propuesta de diseño de sistema de simulación de microrredes

Se propone un sistema de simulación de microrredes basado en un procesador de aplicación específica con un conjunto de instrucciones que permitan obtener los voltajes y corrientes en distintas microrredes.

En algunas aplicaciones el uso de procesadores de propósito general no es viable ya que puede que estos no ofrezcan el desempeño requerido para una aplicación o por su costo o consumo de energía [16].

Los procesadores de aplicación específica presentan varias ventajas con respecto a los procesadores de propósito general: algunas de las más importantes son el alto grado de paralelismo que se puede alcanzar ejecutando varias operaciones de manera concurrente e instrucciones que ejecuten varias transferencias de datos de manera simultánea.

Estas características motivan el diseño de un procesador de aplicación específica para la simulación de microrredes ya que esto involucra ejecutar una gran cantidad de operaciones matemáticas para resolver las ecuaciones que describen el funcionamiento de una microrred.

Para diseñar la arquitectura del sistema de simulación se tomaron en cuenta tres aspectos principales que se detallan a continuación:

- Que el sistema pudiera realizar la mayor cantidad de operaciones en forma concurrente para calcular los valores de la forma más rápida.
- Que el sistema fuera modular, esto para facilitar futuros cambios y actualizaciones en el sistema como nuevos generadores de energía.
- Que el sistema fuera altamente parametrizable para que no haya que realizar cambios importantes cuando se quiera aumentar las capacidades del sistema; por ejemplo, que permita incrementar el número de paneles solares soportados.

Según se mencionó en la sección anterior, se partirá del supuesto que una microrred aislada puede ser descrita por medio de un conjunto de ecuaciones diferenciales algebraicas. Más específicamente, el procesador debe resolver las integrales que se muestran en (11) y (12), y también debe calcular el voltaje en la carga realizando la sumatoria en (9).

3.1 Descripción general del sistema propuesto

La plataforma de simulación de microrredes consta de dos procesadores: un procesador ARM y un procesador de aplicación específica, y un módulo de almacenamiento temporal de datos. Este se presenta en la figura 3.1.

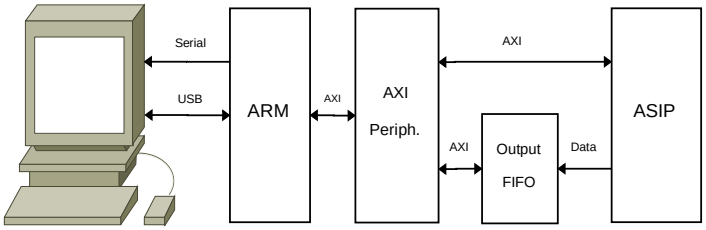

Figura 3.1 Diagrama general de sistema propuesto.

El procesador ARM se encarga de configurar el sistema, enviar las instrucciones a la memoria de programa del procesador de aplicación específica, controlar la ejecución del programa y leer los datos generados y enviarlos a un PC por puerto serie. El procesador de aplicación específica se encarga de ejecutar las operaciones matemáticas que permiten calcular el voltaje y la corriente en una microred. Este procesador envía los datos calculados al FIFO de salida en donde se almacenan mientras son enviados al PC por medio del módulo de transmisión de datos y del ARM.

La transmisión de datos entre el FIFO de salida y el PC limitan el performance del sistema, ya que como una primera propuesta se utiliza el puerto serial que tiene una velocidad de transmisión máxima de 115200 bits por segundo. Como una mejora futura al sistema se propone utilizar una interface Ethernet para la transmisión de datos entre el FIFO de salida y el PC. Además se propone que los datos no sean leídos por el procesador ARM sino que pasen directamente a la interface Ethernet por medio de un módulo DMA. El módulo DMA se encargaría de extraer los datos del FIFO y de enviarlos directamente al PC a través de un puerto Ethernet.

3.2 Descripción de los submódulos del sistema propuesto

3.2.1 Procesador de aplicación específica.

El procesador de aplicación específica es el módulo principal del sistema de simulación. Es un procesador que tiene una serie de bloques encargados de realizar las operaciones matemáticas necesarias para el cálculo de corriente y voltaje en una microred; las operaciones se ejecutan en punto flotante de doble precisión. Este procesador no tiene una unidad

aritmética lógica, sino que tiene varias instancias de un módulo que ejecuta el cálculo de las integrales descritas en (11) y (12) y así obtiene la corriente generada por cada uno de los paneles solares en la microrred. También el procesador tiene un módulo que calcula la corriente total en la microred (9) y otro que envía los resultados a un FIFO de salida para almacenamiento temporal de los datos generados. Las ecuaciones que describen el panel solar son resueltas por unidades distintas por lo que es posible cambiar de forma independiente cada ecuación sin reformular todo el sistema.

El procesador ejecuta una serie de instrucciones indicadas por el usuario por medio de las cuales se asignan valores a los parámetros generales del sistema y también a los parámetros de cada una de las instancias de paneles solares. Luego ejecuta las instrucciones para indicar a los módulos que realizan los cálculos cuándo iniciar un ciclo de cálculo, esperar resultados y enviarlos al FIFO de salida.

En la figura 3.2 se muestra el diagrama de bloques del procesador de aplicación específica.

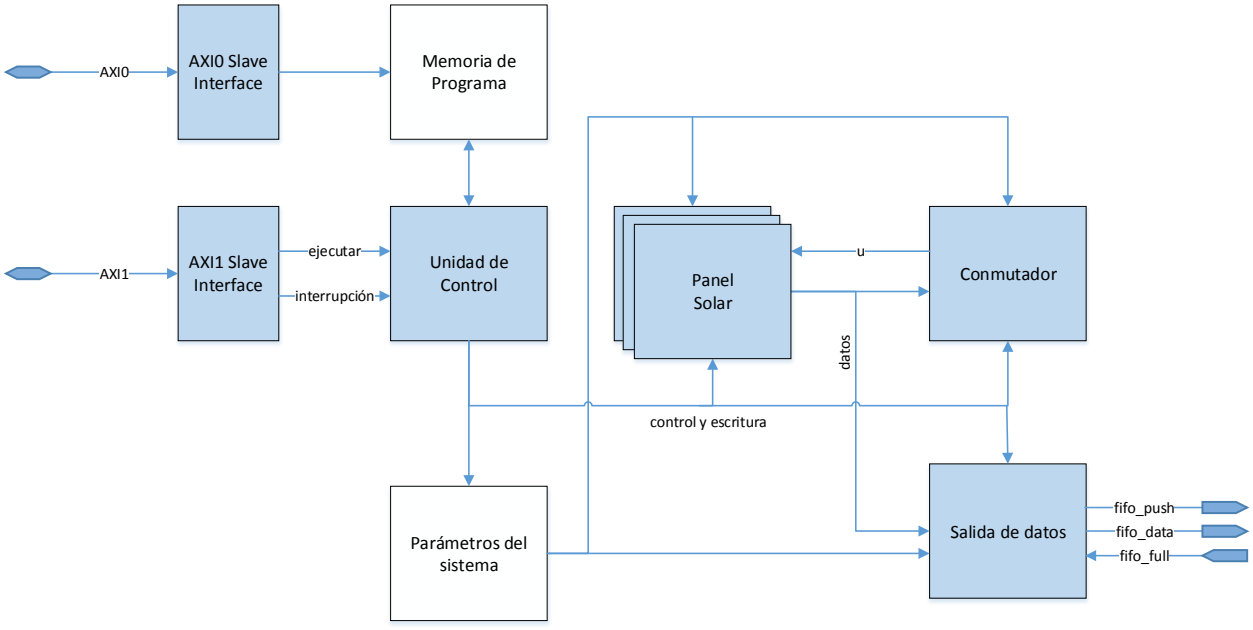

Figura 3.2 Diagrama de bloques del procesador de aplicación específica.

El módulo **Panel Solar** es el que se encarga directamente de resolver las integrales (11) y (12). Este se compone de dos submódulos: uno calcula la corriente de salida del panel solar y el otro calcula el voltaje v_c; ambos submódulos son independientes, pero comparten la información entre sí al final de un ciclo de cálculo para utilizar el valor generado por el otro submódulo en la siguiente iteración. En la figura 3.5 se muestra un diagrama de bloques de este módulo.

Cada uno de estos submódulos está compuesto por un multiplicador de punto flotante, un sumador de punto flotante y una máquina de estados con la respectiva secuencia para calcular la integral correspondiente. Para calcular v_c el submódulo debe obtener la función exponencial de qv/kT que se indica en la ecuación (11). Para esto se creó otro módulo que recibe dicho valor y utiliza una tabla para devolver el resultado de la función exponencial. Esta tabla contiene los valores de la función exponencial en el rango de valores de operación del panel solar.

En las operaciones de uso flotante no se implementa manejo de errores ya que el rango de las operaciones es conocido y en ningún caso, utilizando valores reales en los parámetros, se va a originar un overflow, underflow, valor inexacto ni se va a generar una división entre 0.

El módulo **Conmutador** calcula la sumatoria que se muestra en la ecuación (9). Este módulo recibe el valor de corriente generada por cada uno de los paneles solares activos y los suma para obtener el valor de corriente total en la carga y luego determina el voltaje en la carga.

En el módulo **Parámetros del Sistema** se almacenan los parámetros generales que comparten los diferentes módulos del procesador. Estos parámetros son: valor de resistencia de la carga, intervalo de tiempo para cálculo de integrales y período de muestreo.

El módulo **Salida de Datos** se encarga de enviar los valores indicados por la unidad de control a un FIFO fuera del procesador.

La **Unida de Control** se encarga de ejecutar las instrucciones almacenadas en la memoria de programa. Por medio de las instrucciones controla la ejecución de los otros módulos del procesador.

La figura 3.3 muestra un diagrama del procesador de simulación. Este procesador se comunica con el ARM por medio de dos interfaces AXI esclavas. Una de las interfaces se usa para enviar las instrucciones a ejecutar desde el ARM al procesador de simulación. La otra interface AXI se utiliza para controlar al procesador, indicando cuando comenzar a ejecutar el programa, cuando detenerse, cuando reiniciar o cuando ejecutar una interrupción, por medio de la cual el procesador ejecuta una serie de instrucciones adicionales enviadas por el ARM y luego se sigue con el flujo normal del programa.

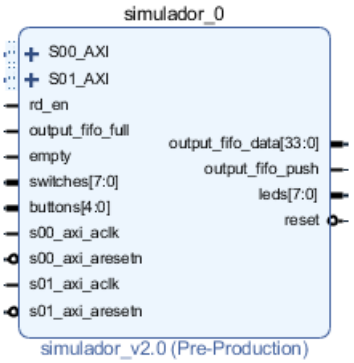

Figura 3.3 Diagrama de procesador de simulación.

El procesador se comunica también con el FIFO de salida, en el cual se almacenan temporalmente los resultados de los cálculos que se envía al PC. En la tabla 3.1 se muestran las interfaces del procesador de aplicación específica.

Tabla 3.1 Puertos del procesador de aplicación específica.

Puerto	Ancho	Dirección	Descripción
clk	1	entrada	Entrada de reloj.
srst	1	entrada	Señal de reset.
S00_AXI	32 bits bus de datos 10 bits bus direcciones	salida y entrada	Interface AXI slave para comunicación con el procesador ARM. Por medio de esta interface el ARM envía el código que va a ejecutar este procesador.
S01_AXI	32 bits bus de datos 7 bits bus direcciones	salida y entrada	Interface AXI slave para comunicación con el procesador ARM. Por medio de esta interface el ARM controla la ejecución del programa en este procesador.
output_fifo_data	34	salida	Datos a almacenar en el FIFO de salida.
output_fifo_push	1	salida	Inserta el dato en el FIFO de salida.
output_fifo_full	1	entrada	Indica que el FIFO de salida está lleno.

El procesador recibe las instrucciones a ejecutar por medio de la interface AXI0, y éstas se almacenan en la memoria de programa. Por medio de la interface AXI1 el usuario indica a través del ARM cuando comenzar a ejecutar el programa. Cuando la unidad de control recibe la señal para comenzar a ejecutar el programa el procesador lee la posición 0x00 de la memoria de programa, donde se encuentra almacenada la primera instrucción del programa o la primer palabra de la primera instrucción. La unidad de control ejecuta una a una las instrucciones almacenadas en la memoria de programa.

Cada módulo del procesador de simulación tiene un parámetro llamado MODULE_ID. A cada módulo se le asigna un valor diferente y se utiliza para direccionarlo.

Hay dos buses de comunicación entre la unidad de control y los demás módulos de procesador. Un bus se utiliza para escribir un valor en los distintos parámetros del sistema, es el bus de escritura. El otro bus se utiliza para indicar la ejecución de los cálculos matemáticos, es el bus de control.

El bus de escritura se muestra en la tabla 3.2

Tabla 3.2 Señales del bus de escritura.

Señal	Bits	Descripción
address	16	Dirección de módulo/parámetro a escribir.
data	64	Valor a escribir.
cmd	1	Indica cuando ejecutar la escritura.

La dirección consta de 16 bits: los 8 bits más significativos indican en cuál módulo se está haciendo la escritura; los 8 bits menos significativos indican la dirección del registro a escribir dentro del módulo. El formato de la dirección se detalla en la tabla 3.3:

Tabla 3.3 Bits de direccionamiento.

Bits de dirección															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Módulo								Registro							

En la tabla 3.4 se muestran los valores de MODULE_ID asignados a cada módulo dentro del procesador de simulación.

Tabla 3.4 Asignación de MODULE_ID para cada módulo en el procesador.

Módulo	MODULE_ID
Parámetros generales	0
Conmutador	1
Panel solar 0	2
Panel solar 1	3
Panel solar 2	4
Panel solar 3	5
Panel solar 4	6
Panel solar 5	7
Panel solar 6	8
Panel solar 7	9
Panel solar 8	10
Panel solar 9	11

Así, por ejemplo, para direccionar el registro 2 del panel solar 1 se utiliza la dirección 0x0302.

Con este formato de direccionamiento se pueden acceder 256 registros en 256 módulos diferentes; esta configuración es variable, no obstante. Se puede por ejemplo dar más bits al campo de módulo y menos bits al campo de registro o viceversa según corresponda.

El bus de datos se hizo de 64 bits para poder escribir un dato en formato punto flotante de doble precisión en un ciclo de reloj.

El bus de control, que se muestra en la tabla 3.5, se utiliza para indicar cuál o cuáles módulos del procesador deben ejecutar un cálculo y para saber cuáles módulos ya han terminado la ejecución.

En este caso, a diferencia del bus de escritura, se pueden direccionar varios módulos en una sola instrucción, lo que se hace utilizando un mapa de bits. La asignación de bits se hace también utilizando el parámetro MODULE_ID.

Tabla 3.5 Señales del bus de control.

Señal	Bits	Descripción
module_selected	*TOTAL_MOD*	Indica cuál o cuáles módulos se seleccionaron para ejecutar un comando.
module_ready	*TOTAL_MOD*	Indica cuales módulos han terminado de ejecutar una iteración de cálculo.
start	1	Indica a los módulos indicados por module_selected que deben iniciar una iteración de cálculo.

Por lo tanto, si se quiere ejecutar un ciclo de cálculos en el panel solar 0 y en el panel solar 1, se asigna a *module_selected* un valor de 0x000C.

Un programa a ser ejecutado por el procesador de aplicación específica consta primero por instrucciones para configurar los diferentes parámetros del sistema. Luego entra a un ciclo donde se indica a los módulos de generadores de energía que inicien un ciclo de cálculo, espera a que todos los módulos ejecuten los respectivos cálculos y una vez finalizados envía los datos generados al FIFO de salida. Este ciclo se puede repetir una cantidad determinada de veces.

En el diagrama de flujo de la figura 3.4 se describe un programa de simulación básico para ejecutar en el procesador de aplicación específica.

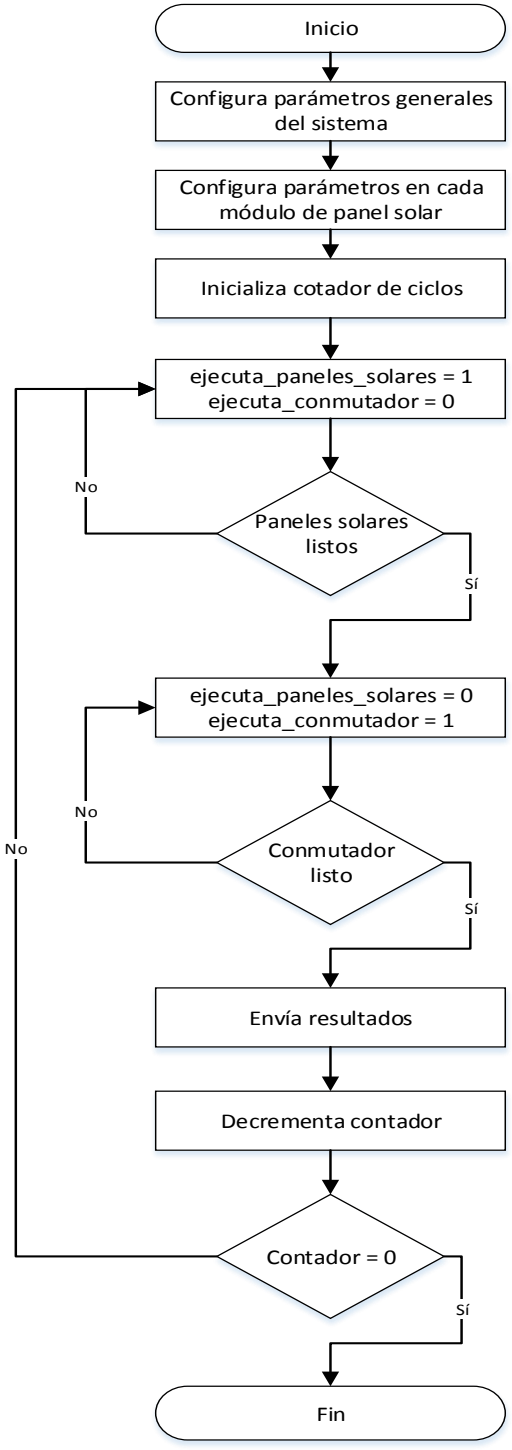

Figura 3.4 Diagrama de flujo de programa básico de simulación.

En las siguiente sub secciones se describe con mayor detalle los módulos del procesador de aplicación específica.

3.2.1.1 Unidad de control

La unidad de control se encarga de ejecutar las instrucciones almacenadas en la memoria de programa. Tiene una máquina de estados que controla el contador de programa, lee las instrucciones y realiza los pasos indicados por las mismas.

La unidad de control asigna valores a los parámetros generales del sistema y a los registros de los módulos de panel solar. También indica cuando comenzar a ejecutar un ciclo de cálculo y espera a que dichos cálculos finalicen. Además indica al módulo de salida de datos cuales valores enviar al FIFO de salida y controla el flujo del programa.

La unidad de control tiene dos buses para comunicarse con los otros módulos del procesador: un bus para acceder a los parámetros generales y a los parámetros de cada módulo de panel solar; y un bus de control y estado por medio del cual indica cuál o cuáles módulos deben iniciar un ciclo de cálculo y por el que cada módulo indica a la unidad de control cuándo finalizó la ejecución de un ciclo de cálculo.

En la tabla 3.6 se detallan las interfaces de la unidad de control con los otros módulos del procesador de simulación.

Tabla 3.6 Puertos de unidad de control.

Puerto	Ancho	Dirección	Descripción
clk	1	entrada	Entrada de reloj.
reset_L	1	entrada	Señal de reset.
program_counter	8	salida	Bus de dirección de la memoria de programa.
program_data	32	entrada	Bus de datos de la memoria de programa.
address	16	salida	Bus de dirección de la memoria de datos.
data	64	salida	Bus de datos de la memoria de datos.
rnw	1	salida	Indica si acceso a memoria de datos es para escritura o para lectura.
cmd	1	salida	Indica a la memoria de datos que debe ejecutar lectura o escritura.
module_selected	TOTAL_MOD	salida	Indica cual sub módulo se seleccionó para ejecutar un comando.

module_enabled	TOTAL_MOD	salida	Indica cuales sub módulos están habilitados para la simulación.
module_ready	TOTAL_MOD	entrada	Indica cuales módulos han terminado de ejecutar una iteración de cálculo.
start	1	salida	Indica a los sub módulos indicados por *module_selected* que deben iniciar una iteración de cálculo.
send	1	salida	Indica que se debe enviar un dato al FIFO de salida.
send_sel	sizeof(TOTAL_MOD*2)	salida	Indica cual dato se envía al FIFO de salida.
last_digit	1	salida	Indica cual carácter se va a enviar luego del último decinal.
print_ready	1	entrada	Indica que se puede enviar un dato al FIFO de salida.
execute	1	entrada	Indica a la unidad de control cuando ejecutar el programa.
execute_int	1	entrada	Indica a la unidad de control cuando ejecutar código de interrupción.
reinit	1	entrada	Indica a la unidad de control que reinicie la ejecución del programa.

La unidad de control es a su vez controlada por el ARM a través de la interface AXI1 del Procesador. Con una escritura del ARM a la dirección base de esta interfaz, que se muestra en la tabla 3.12, se controla la ejecución del programa.

La ejecución se controla de acuerdo a la siguiente tabla:

Tabla 3.7 Control de ejecución del programa.

Bit	Nombre	Descripción
0	*execute*	Habilita ejecución del programa. Se debe escribir este bit en 1 para que se ejecute el programa y en 0 para pausar la ejecución.
1	*execute_int*	Se utiliza para ejecutar código fuera del flujo normal del programa.
2	*reinit*	Se comienza a ejecutar el programa desde el inicio.

El flujo normal del programa se puede interrumpir temporalmente para ejecutar una serie de instrucciones y luego regresar al flujo normal del programa o para reiniciarlo. Esto con el fin de cambiar algunos valores de parámetros en tiempo de ejecución.

Si se escribe en 1 el bit *execute_int* el contador de programa salta a la posición 0xC0, donde previamente se habían escrito las instrucciones adicionales que se quieren ejecutar, además se almacena el valor del contador de programa en un registro. Luego, por medio de una instrucción el contador de programa, se asigna con el valor en el registro con lo que se carga con la dirección que tenía antes de la interrupción.

Otra forma de interrumpir el flujo normal del programa es por medio del bit *reinit*. Si se escribe este bit en 1 al contador de programa se le asigna el valor de 0x00 por lo que la siguiente instrucción a ejecutar es la primera instrucción del programa.

Las instrucciones se detallan en la sección 3.5

3.2.1.2 Memoria de programa

Es una memoria de lectura y escritura en donde se almacenan las instrucciones a ejecutar por el procesador de simulación.

Es una memoria de 1KB, las instrucciones son de 32 bits por lo que se pueden almacenar un máximo de 256 instrucciones.

Las últimas 64 palabras de la memoria se pueden utilizar para almacenar instrucciones a ejecutar a modo de interrupción. Cuando la unidad de control recibe la señal para ejecutar interrupción el contador de programa salta a la palabra 0xC0 de esta memoria.

El ARM escribe las instrucciones en esta memoria a través de la interface AXI00 del procesador; el acceso es de lectura y escritura. La unidad de control solo tiene acceso de lectura a esta memoria.

3.2.1.3 Parámetros del sistema

En este módulo se almacenan los parámetros generales del sistema que son utilizados por los demás módulos del procesador.

Los valores de los parámetros se almacenan en los registros que se muestran en la siguiente tabla:

Tabla 3.8 Parámetros del sistema.

Offset	Nombre	Ancho	Descripción
0x00	*resistor*	64	Valor en ohms de la carga de la microred. Se representa en punto flotante.
0x01	*delta t*	64	Este valor indica el incremento de tiempo en microsegundos entre cada ciclo de cálculo. Aquí se almacena en punto flotante.
0x02	*delta t int*	10	Este valor indica el incremento de tiempo en microsegundos entre cada ciclo de cálculo. Aquí se almacena como un entero, con un valor máximo de 1023 microsegundos.
0x03	*muestreo*	10	Indica cada cuantos ciclos se envía un dato al PC. Este valor se almacena como un entero con un valor máximo de 1023.

3.2.1.4 Panel solar

Este módulo es el que se encarga de calcular la corriente generada por un panel solar y el voltaje en el capacitor del mismo. Este módulo se instancia una cantidad de veces igual a la cantidad de paneles solares en la microred y todas las instancias pueden trabajar en paralelo.

La figura 3.5 muestra el diagrama de bloques del módulo de panel solar.

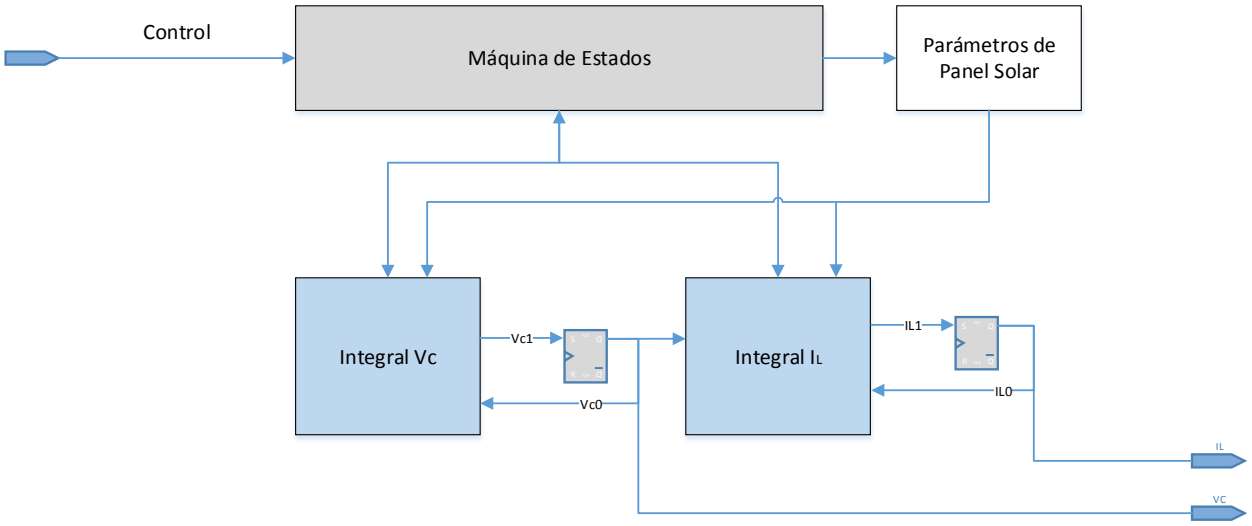

Figura 3.5 Diagrama de módulo de panel solar.

En la tabla 3.9 se muestran las interfaces de los módulos Panel Solar

Tabla 3.9 Puertos de módulo Panel Solar.

Puerto	Ancho	Dirección	Descripción
clk	1	entrada	Entrada de reloj.
reset_L	1	entrada	Señal de reset.
address	16	entrada	Bus de dirección.
cmd	1	entrada	Indica ejecución de comando de lectura o escritura.
rnw	1	entrada	Indica si el comando es de escritura o lectura.
data_in	64	entrada	Dato a escribir.
data_out	64	salida	Dato leído.
delta_t	64	entrada	Diferencial de tiempo para cálculo de integrales.
resistor	64	entrada	Valor de la carga.
u	1	entrada	Valor de *u* para cálculo de integrales.
vr	64	entrada	Voltaje en la carga.
vc	64	salida	Valor calculado de voltaje en el capacitor.

il	64	salida	Valor calculado de corriente de salida.
selected	1	entrada	Indica si el módulo debe ejecutar ciclo de cálculo.
start	1	entrada	Indica que se debe iniciar ciclo de cálculo.
ready	1	output	Indica que se ha completado ciclo de cálculo.
exp_req	1	salida	Solicita valor de exponencial.
exp_exponente	1	salida	Indica valor del exponente.
exp_ack	1	entrada	Indica respuesta válida.
exp_resultado	1	entrada	Valor de exponencial.

Cada instancia de este módulo tiene un conjunto de registros en donde se almacenan los parámetros particulares a cada una.

En éste módulo hay dos sub módulos que realizan los cálculos, un sub módulo calcula la integral que determina el valor de *Vc* y el otro sub módulo calcula la integral que determina el valor de la corriente generada por el panel solar.

Ambos sub módulos son controlados por una máquina de estados que se activa al recibir una señal de la unidad de control. Cuando los cálculos se finalizan se indica por medio de otra señal que los resultados están listos.

Los valores de *Vc* y de I_L calculados se almacenan en un registro y son utilizados como valores iniciales para el siguiente ciclo de cálculo.

La siguiente tabla detalla los registros de cada módulo, todos los valores se expresan en punto flotante de doble precisión.

Tabla 3.10 Registros de cada módulo de panel solar.

Offset	Nombre	Ancho	Descripción
0x00	Capacitor	64	Valor del capacitor de la celda solar.
0x01	Inductor	64	Valor del inductor del inversor a la salida del panel solar.
0x02	I_{S0}	64	Corriente de saturación inversa.
0x03	V_{C0}	64	Voltaje inicial en el capacitor.
0x04	I_{L0}	64	Corriente inicial en el inductor.
0x05	Celdas	64	Cantidad de celdas solares en el panel solar.

Todos estos valores se representan en punto flotante de 64 bits.

3.2.1.4.1 Integral Vc

Este módulo determina el valor del voltaje en el capacitor calculando la integral que se muestra en (11).

Para realizar los cálculos necesarios tiene un módulo multiplicador de punto flotante de doble precisión y un módulo sumador de punto flotante también de doble precisión.

En este módulo hay una máquina de estados que controla la ejecución de las operaciones de multiplicación y de suma, así como el movimiento de datos entre los dos módulos. También se encarga de invertir el signo de los datos en el caso que se requiera hacer una resta para poder utilizar el mismo sumador.

En la figura 3.6 se muestra el diagrama de bloques de este módulo.

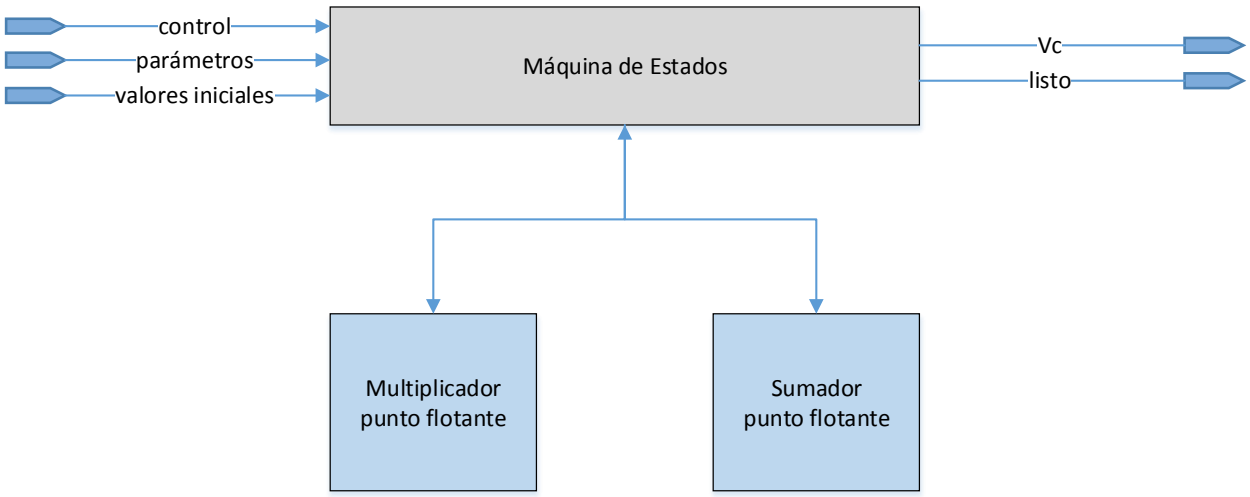

Figura 3.6 Diagrama de bloques de módulo integral Vc.

3.2.1.4.2 Integral I_L

Este módulo determina el valor de la corriente generada por un panel solar calculando la integral que se muestra en (12).

La figura 3.7 muestra el diagrama de bloques de este módulo. Es muy similar al anterior con la diferencia que la máquina de estados ejecuta otra secuencia de cálculos, en este caso para determinar la corriente.

Posee también un multiplicador y un sumador en punto flotante de doble precisión.

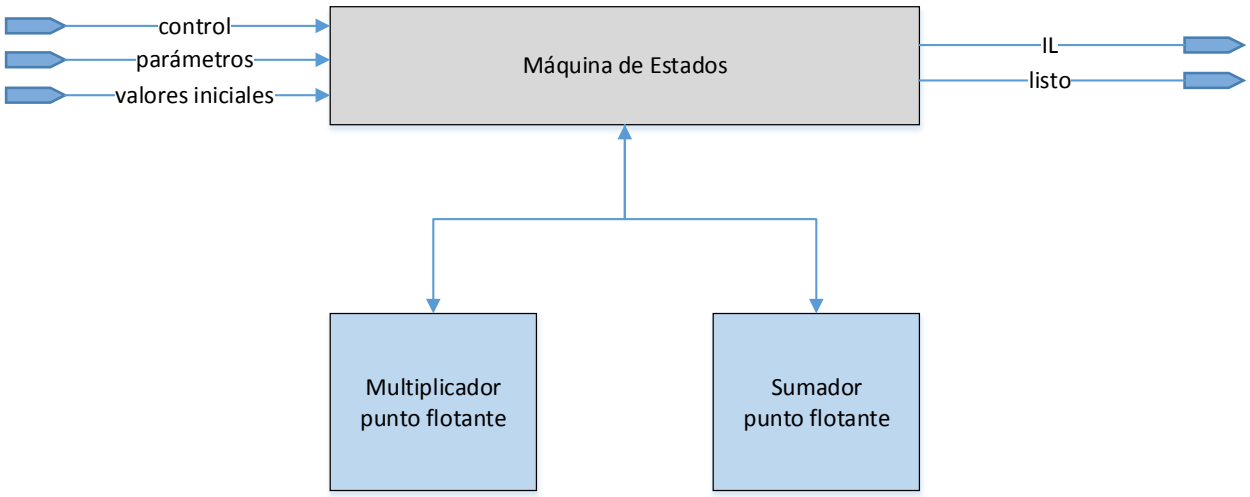

Figura 3.7 Diagrama de bloques de módulo integral IL.

3.2.1.5 Conmutador

Este módulo se implementó para calcular la sumatoria que se muestra en la ecuación (9). Recibe el valor de corriente generada por cada uno de los paneles solares activos y los suma para obtener el valor de corriente total en la carga. Luego multiplica ese valor de corriente por el valor de la carga para determinar el voltaje. Otra función de este módulo es determinar el valor de *u* que utilizan los módulos Panel Solar para el cálculo de las integrales. Para eso este módulo genera una señal senoidal de referencia que se utiliza para compararla con el voltaje en la carga. Si el voltaje en la carga es menor que el voltaje de referencia asigna un valor de 1 a la variable *u* y si el voltaje en la carga es mayor que el voltaje de referencia se asigna un valor de -1 a la variable *u*.

3.2.1.6 Multiplicador

Se diseñó un multiplicador de punto flotante de doble precisión que fuera eficiente en el uso de recursos del FPGA. Para eso se utilizó el multiplicador disponible en el DSP48E1, se creó un módulo que resuelve la multiplicación total haciendo varias multiplicaciones de 18 x 18 bits y sumando los resultados.

3.2.1.7 Exponencial

Para obtener la exponencial se utilizaron tablas implementadas en una memoria ROM. Se utilizan varias tablas para dar mayor precisión en el rango de operación propio de esta aplicación.

Se decidió utilizar una tabla para resolver la exponencial debido a la velocidad con la que se obtiene el resultado utilizando este método. Con una tabla se puede resolver un valor de exponencial cada ciclo de reloj.

Se creó un solo módulo de exponencial que es compartido por todos los módulos de panel solar, por lo que se implementó un árbitro tipo round-robin que va tomando cada una de las solicitudes de exponencial y envía el resultado al módulo respectivo.

Este módulo se implementó para calcular la parte de la ecuación (11) que calcula la corriente generada por un panel solar mostrada en (2). Suponiendo un valor de corriente de saturación inversa de 1μA y un valor de qV/kT = 17 el valor de i_g máximo es de 24.15A.

$$24{,}15\text{A} = 1\mu\text{A}(e^{17} - 1)$$

Si se soporta un valor de i_g máximo de 24.15A, para cualquier valor de exponencial de 17 en adelante la corriente generada por el panel solar es de 0A, ya que no se puede generar una corriente negativa. De este modo el módulo exponencial solo debe calcular la función exponencial para números entre 0 y 17. Para cualquier valor de exponencial mayor a 17 el valor de i_{pv} es 0A.

Para esto se implementaron cuatro tablas. La primera tabla contiene los valores de la función exponencial para un rango de 0 a 8 con una resolución de 0.25. La segunda tabla contiene los valores para un rango de 8 a 12 en incrementos de $1/2^8$, la tercera para un rango de 12 a 16 en incrementos de $1/2^{10}$ y la cuarta para valores de 16 a 17 en incrementos de $1/2^{11}$.

En la siguiente tabla se muestran los diferentes rangos para cada una de las tablas y el error máximo en cada caso.

Tabla 3.11. Rango de tablas i mplementadas para función exponencial.

Rango Mínimo	Rango Máximo	Límite Redondeo	Bits Entero	Bits Decimal	Total Bits	Datos	Diferencia Exp.	Diferencia Ipv (A)	% Error
0	7,75	7,875	3	2	5	32	350,2718	0,000350	0,00%
8	11,9961	11,9980	2	8	10	1024	317,5702	0,000318	0,00%
12	15,9990	15,9995	2	10	12	4096	4337,862	0,004338	0,04%
16	16,9995	16,9998	0	11	11	2048	5896,485	0,005896	0,06%

3.2.1.8 Salida de datos

Este módulo envía los datos indicados por la unidad de control al FIFO de salida.

Recibe los valores de voltaje en el capacitor y de corriente generada por cada panel solar así como el valor de la corriente y voltaje en la carga de la microred.

Por medio de la entrada *print_sel* la unidad de control indica cuál valor se envía al FIFO para luego ser enviado a la PC. Antes de enviar el dato seleccionado este módulo convierte el valor en punto flotante de doble precisión a punto fijo, en hexadecimal. La parte entera la representa con 16 bits y la parte decimal con 20 bits, lo que permite una resolución de 0.000001. Luego los valores hexadecimales los convierte a BCD y los almacena en el FIFO con este formato para facilitar así luego la conversión a ASCII.

Cuando el FIFO de salida está lleno este módulo indica a la unidad de control que no hay campo para enviar más datos. Si la unidad de control está ejecutando una instrucción para enviar un dato al PC se espera a que haya un espacio en el FIFO para poder seguir con la ejecución de las instrucciones.

Los datos se almacenan en el FIFO de salida con el siguiente formato:

Tabla 3.12 Formato datos de salida.

Bits																															
33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F	S	E1				E2				E3				D1				D2				D3				D4				D5			

donde:

E1 a E3 = Digitos de la parte entera.

D1 a D5 = Digitos de la parte decimal.

S = signo.

F = indica si es el último carácter de la línea.

De esta forma el valor máximo a representar es de 999.99999.

3.2.2 Sistema de procesamiento ZYNQ7 y periféricos.

Se utilizó el sistema de procesamiento ZYNQ7 que provee el FPGA y la herramienta de desarrollo Vivado para hacer uso del procesador ARM. Este sistema de procesamiento incluye un procesador ARM CortexA9 Dual Core, un controlador de memoria DDR3 y otra serie de componentes necesarios para el funcionamiento del ARM.

Junto con el ZYNQ7 el FPGA provee un módulo de interconexión entre el ARM y los demás componentes del sistema que se detallan más adelante, este módulo se encarga de re direccionar los accesos realizados por el ARM al sub módulo respectivo de acuerdo al mapa de memoria del sistema que se muestra en la tabla 3.12. La interconexión del ARM con los demás componentes se hace a través de interfaces AMBA AXI4.

Tabla 3.13 Mapa de memoria del sistema.

Dirección inicial	Dirección final	Interfaz	Descripción
0x43C0_0000	0x43C0_FFFF	S00AXI / Trasmisión	FIFO de salida
0x43C1_0000	0x43C1_FFFF	S00AXI / Procesador	Memoria de programa de procesador de simulación
0x43C2_0000	0x43C2_FFFF	S01AXI / Procesador	Control del sistema

3.2.3 FIFO de salida.

En este FIFO se almacenan los resultados de los cálculos realizados por el procesador de cálculo. Este es un FIFO con un ancho de palabra de 34 bits y una profundidad de 131072 palabras. Los resultados se almacenan en este FIFO temporalmente mientras se envían a un PC a través de un puerto serial. En la figura 3.8 se muestra el diagrama del FIFO de salida.

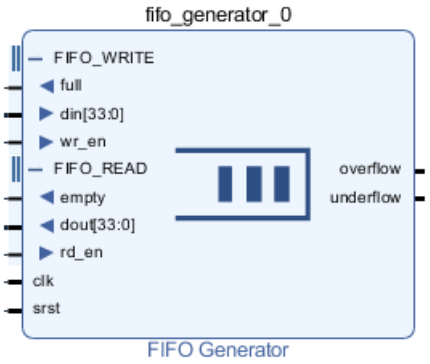

Figura 3.8 Diagrama del FIFO de salida.

El FIFO recibe los resultados del procesador de cálculo a través de su puertos *din[33:0]* y *wr_en. wr_en* indica cuando hay un dato válido en *din*. El FIFO indica cuando está lleno al procesador de cálculo a través de su salida full.

El FIFO indica al módulo *read_print_data* cuando tiene datos válidos a través de su salida *empty*. También indica a este módulo si hubo un error por overflow o underflow, con el propósito de almacenar esos estados en caso de que se detecte un mal funcionamiento cuando se está depurando el sistema.

Tabla 3.14 Puertos del FIFO de salida.

Puerto	Ancho	Dirección	Descripción
clk	1	entrada	Entrada de reloj.
srst	1	entrada	Señal de reset.
full	1	salida	Indica que el FIFO está lleno y no puede recibir más datos.
din	34	entrada	Dato a escribir en el FIFO.
wr_en	1	entrada	Inserta el dato en *din* en el FIFO.
empty	1	salida	Indica que no hay datos disponibles en el FIFO.
dout	34	salida	Siguiente dato en salir del FIFO.
rd_en	1	entrada	Saca el siguiente dato del FIFO.
overflow	1	salida	Señal de error que indica que se escribió un dato en el FIFO cuando estaba lleno.

underflow	1	salida	Señal de error que indica que se sacó un dato en el FIFO cuando estaba vacío.

3.2.4 Transmisión de resultados.

El módulo de transmisión de resultados, que se muestra en la figura 3.9, se encarga de extraer los datos del FIFO de salida y enviarlos al ARM para que este a su vez se los envíe a la PC y se genere un archivo de texto que se puede utilizar para procesar los resultados obtenidos.

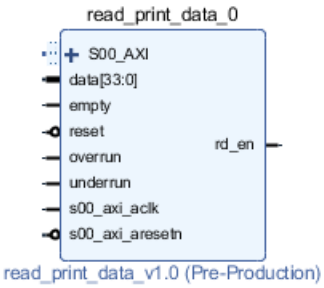

Figura 3.9 Módulo de transmisión de datos.

En la siguiente tabla se muestran los puertos de este módulo:

Tabla 3.15 Puertos del módulo de transmisión de datos.

Puerto	Ancho	Dirección	Descripción
s00_axi_clk	1	entrada	Entrada de reloj.
s00_axi_aresetn	1	entrada	Señal de reset.
S00_AXI	32 bits bus de datos 4 bits bus direcciones	salida y entrada	Interface AXI slave para comunicación con el procesador ARM.
data	34	entrada	Dato en el FIFO de salida.
empty	1	entrada	Indica si no hay un dato valido en data.
overflow	1	entrada	Indica que hubo un error por overflow en el FIFO de salida.

underflow	1	entrada	Indica que hubo un error por underrun en el FIFO de salida.
rd_en	1	salida	Saca un dato del FIFO de salida.

Este módulo tiene una interface AXI tipo esclava por medio de la cual se comunica con el ARM. El ARM utiliza esta interface para leer y extraer los datos del FIFO de salida.

Antes de enviar los datos al ARM este módulo convierte los datos extraídos del FIFO de salida de BCD a ASCII y forma un string de 12 caracteres que representan un número incluyendo el carácter de coma para separar la parte entera de la parte decimal, el signo del dato + o -, un carácter después del último decimal para indicar que ahí termina el número y finalmente el carácter *null* para indicar que ahí termina el string. Junto con esta información también se envía información al PC indicando si hay datos disponibles en el FIFO y las banderas de error de overrun o underrun.

La conversión de BCD a ASCII se realiza en éste módulo de manera combinacional para que el ARM no consuma ciclos de reloj haciendo la conversión antes de enviar los datos por el puerto serial.

Los datos son almacenados en el FIFO en formato BCD. En este módulo se convierten de ese formato a ASCII y se reordenan para ubicar en la dirección más baja de memoria el primer carácter a imprimir, en este caso el carácter de signo, y en la dirección más alta el carácter de *null* para que el programa que se ejecuta en el ARM sepa dónde termina la tira de caracteres a imprimir. El formato de la tira y la información del FIFO que se envían al ARM se muestran en la tabla 3.15.

Tabla 3.16 Formato de información a transmitir.

Offset de la dirección	Descripción
0x0000_0000	{E3, E2, E1, carácter de signo}
0x0000_0004	{D3, D2, D1, carácter de coma}
0x0000_0008	{L2, L1, D5, D4}
0x0000_000C	{overflow, underflow , dato válido, carácter null}

E1, E2 y E3 son los caracteres que representan la parte entera del dato, donde E1 es el más significativo y E3 el menos significativo. D1 a D5 son los caracteres que representan la parte decimal donde D1 es el más significativo y D5 el menos significativo. L1 y L2 son los últimos caracteres a desplegar o transmitir a la PC, estos caracteres dependen del bit de fin de línea. Si el bit de fin de línea es 0 se asigna a L1 el carácter de espacio y a L2 el de *null*. Si el bit de fin de línea es 1 se asigna a L1 el carácter para pasar a siguiente línea (LF) que tiene un valor de 0xA y a L2 el carácter para ir al inicio de línea (CR) con un valor de 0xD.

Para enviar los datos a la PC el programa que corre en el ARM crea un puntero tipo char que apunta a la dirección con *offset* 0x0000_0000 de éste módulo. Luego lee el *offset* 0x0000_000C de este módulo y extrae el bit de válido. Si este bit está en 1 se pasa el puntero a la función print. Esta función envía por puerto serie la cadena que se encuentra a partir de la dirección apuntada por el puntero que se pasó como argumento de la función. En este caso se envían los caracteres a partir del carácter de signo, que se encuentra en el *offset* 0x0000_0000 hasta el carácter null que se encuentra en el *offset* 0x0000_000D (L2) o en el *offset* 0x0000_000C según sea el caso.

Cuando este módulo recibe una lectura al *offset* 0x0000_0008 se activa la salida rd_en por un ciclo de reloj por lo que se extrae un dato del FIFO. En el siguiente ciclo de reloj hay un nuevo dato en la salida del FIFO si hay datos disponibles.

3.2.5 Instrucciones

Se implementaron instrucciones para movimiento de datos, ejecución de cálculos y control de flujo del programa.

Las instrucciones tienen un ancho de palabra de 32 bits y 1, 2 o 3 palabras por instrucción.

Las instrucciones siguen el formato descrito en la tabla 3.17.

Tabla 3.17 Formato de instrucciones.

Instrucción																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Código de operación								Argumento																							

Los 8 bits más significativos indican el código de operación y los otros 24 bits son el argumento, que dependen de cada instrucción. También el argumento puede continuar en la siguiente palabra o las siguientes dos palabras.

La tabla 3.16 resume las instrucciones disponibles.

Tabla 3.18 Instrucciones.

Instrucción	Código	Descripción.
WR_REG	0x01	Escribir en un registro del procesador.
RD_REG	0x02	Leer un registro del procesador.
SET_CYCLES	0x04	Escribir el contador de ciclos.
DEC_CYCLES	0x05	Decrementar el contador de ciclos y saltar si el contador no es 0x00.
EXEC_CALC	0x06	Ejecutar cálculos.
WAIT_CALC	0x07	Esperar a que los cálculos se hayan terminado.
PRINT	0x08	Enviar un dato al PC.
RFI	0x09	Regresar de código de interrupción.
EXEC_EN	0x0A	Habilitar módulos de panel solar.
REINIT	0x0B	Reiniciar ejecución del programa.
STOP	0xFF	Terminar ejecución de programa.

En el apéndice B se detallan las instrucciones soportadas por el procesador de aplicación específica.

4 Discusión y resultados

Para comprobar el correcto funcionamiento de la arquitectura propuesta se sintetizó el sistema de simulación en un FPGA Xilinx XC7Z020-CLG484. El FPGA utilizado incluye un procesador ARM Cortex A9 que se utilizó para enviar el programa a ejecutar en el procesador de simulación y también para extraer los datos generados y enviarlos a un PC a través de un puerto serial.

Se realizaron las siguientes pruebas:

- Respuesta al escalón unitario.
- Validación del sistema por simulación de RTL.
- Simulación de un panel solar con cambio de corriente fotogenerada.
- Simulación de un panel solar con diferente valor de inductor.
- Simulación de un panel solar con cambios en valor de la carga.
- Simulación de diez paneles solares.

4.1 Respuesta al escalón unitario

La respuesta al escalón unitario se obtuvo con el fin de comparar los resultados obtenidos con el sistema de simulación contra los resultados obtenidos por medio de un programa simulador de circuitos eléctricos corriendo en un computador. Se programó el procesador de aplicación específica de tal manera que permitiera generar los resultados de una respuesta a un escalón unitario con los siguientes parámetros: corriente generada por la luz de 1A, resistencia de carga de 1Ω, capacitancia de 1F e inductancia de 1H. Con estos valores se corrió una simulación de 10 segundos y se obtuvieron los datos de corriente generada por el panel solar y de voltaje en el capacitor. Con estos datos se generó el gráfico de la derecha en la figura 4.1.

Una vez obtenida esta información se simuló el circuito equivalente de la microred en estudio en una PC por medio del programa simulador Electronics Workbench utilizando los mismos valores. En este caso se obtuvo el gráfico de la izquierda en la figura 4.1.

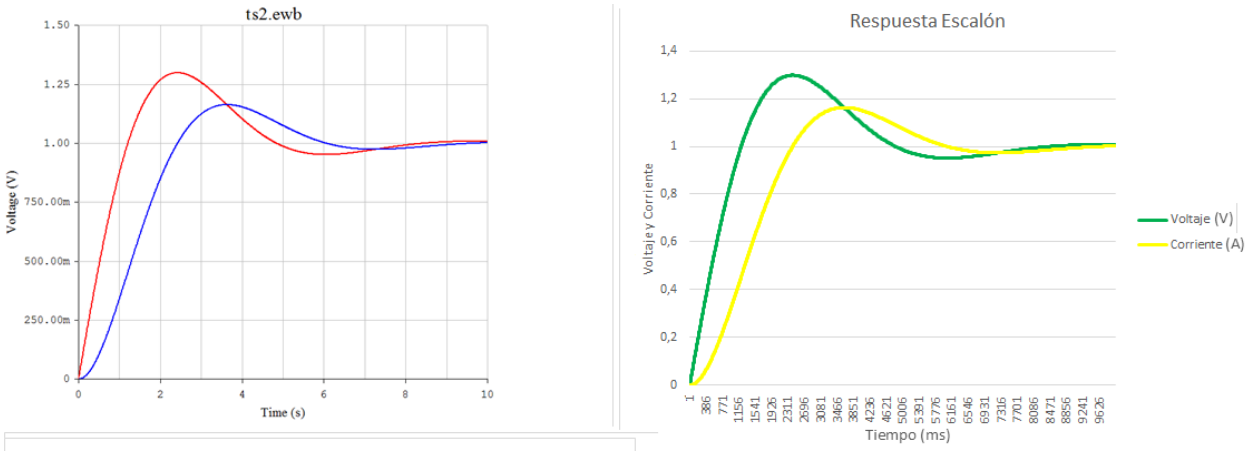

Figura 4.1 Comparación de resultados obtenidos con una PC y con el procesador de simulación.

Para comprobar que el gráfico generado con la plataforma de simulación fuera igual al gráfico generado por la simulación en una PC se colocaron los dos gráficos uno sobre el otro, el resultado se muestra en la figura 4.2. Aquí se puede observar que las señales generadas por el procesador de aplicación específica son iguales a las señales generador por un computador tanto en la amplitud como en el tiempo.

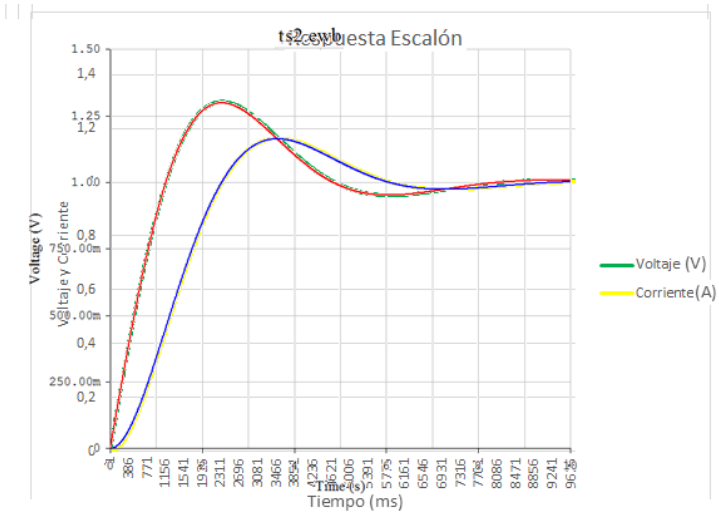

Figura 4.2 Superposición de gráficos de resultados.

4.2 Validación del sistema por simulación de RTL

Se creó un testbench para verificar el funcionamiento del procesador de simulación. Se hicieron diferentes pruebas para correr simulaciones y generar las señales de onda y así poder determinar que el procesador fuera capaz de ejecutar un programa que permitiera calcular el valor correcto del voltaje en el capacitor y la corriente en la carga en la microred en estudio. También para determinar el tiempo de ejecución de cada uno de los módulos que conforman el procesador así como el de un ciclo completo de cálculo. Las simulaciones se corrieron utilizando el simulador de la herramienta Vivado.

Se comprobó que la suma en punto flotante tarda 3 ciclos de reloj y que una multiplicación en punto flotante tarda 18 ciclos de reloj, como se observa en las figuras 4.3 y 4.4.

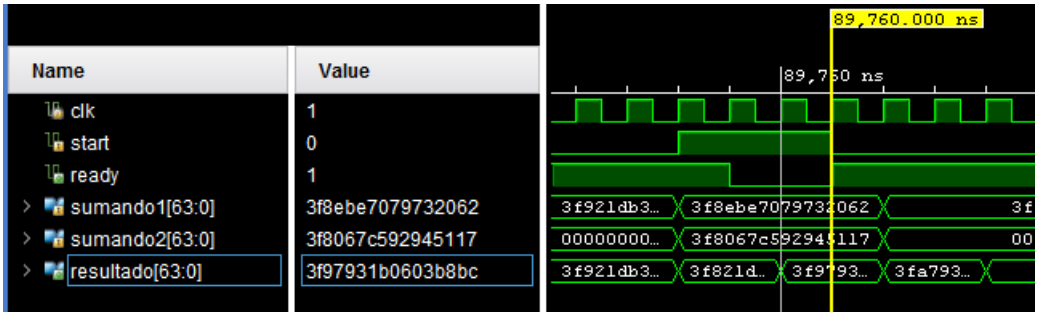

Figura 4.3 Cálculo de suma en punto flotante.

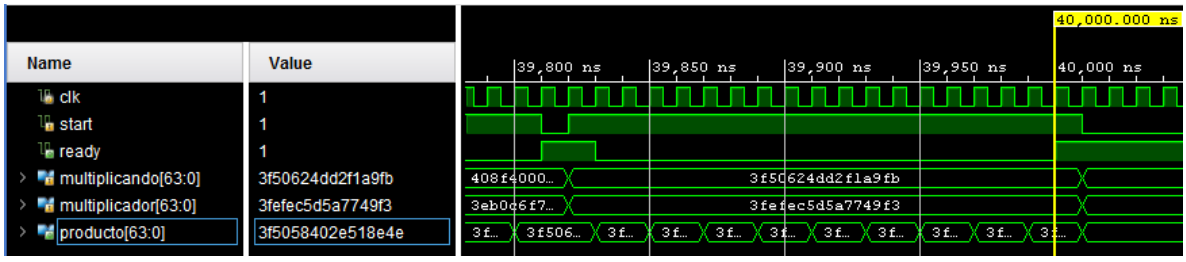

Figura 4.4 Cálculo de multiplicación en punto flotante.

Se determinó que el cálculo de la corriente por el inductor se realiza en 42 ciclos de reloj y el cálculo del voltaje en el capacitor en 85 ciclos de reloj. El módulo panel solar ejecuta estos dos cálculos en paralelo, el tiempo total desde que recibe la señal de inicio hasta que termina de calcular ambos valores es de 87 ciclos de reloj.

El conmutador tarda 19 ciclos de reloj en determinar el voltaje en la carga. Se requieren 3 ciclos más por cada módulo de panel solar adicional que esté activo en el sistema, que se necesitan para sumar la corriente generada por cada panel.

La latencia completa de cálculo que consiste en calcular la corriente en el inductor y el voltaje en el capacitor, calcular el voltaje en la carga y finalmente enviar los datos al FIFO de salida es de 125 ciclos de reloj.

El FPGA utilizado tiene un módulo de manejo de reloj (MMCM) que permite configurar el reloj a diferentes frecuencias. Si se configura el reloj a 125MHz se podría ejecutar un ciclo de cálculo en 1 microsegundos. Y si a la vez se configura el procesador de simulación para que en cada ciclo de cálculo el *delta t* sea de 1 microsegundo la plataforma de simulación trabajaría a tiempo real.

Si se configura el reloj a 500MHz un microsegundo de simulación se haría en 250 nanosegundos, por lo que la plataforma de simulación trabajaría a 4 veces la velocidad real. Por ejemplo una simulación de 1 minuto se obtendría en 15 segundos.

Sin embargo el desempeño del sistema está limitado por la velocidad de transmisión de datos a la PC. Una vez que se llena el FIFO de salida el procesador de simulación no continúa un cálculo hasta que haya un espacio disponible en el FIFO para almacenarlo. Esto limita la velocidad de ejecución a 1309 cálculos por segundo, porque cada resultado se expresa en 11 caracteres: un carácter de signo, 3 caracteres de la parte entera, la coma decimal, cinco caracteres de parte fraccionaria y un carácter delimitador, por lo que cada dato consta de 88 bits. La velocidad máxima de comunicación por puerto serie de la tarjeta Zedboard es de 115200 bits por segundo por lo que se pueden enviar 1309 resultados por segundo. Con la resolución máxima del sistema se pueden simular 1.301 milisegundos en un tiempo de 1 segundo, o de otra forma un segundo de simulación se ejecuta en 764 segundos. Se requiere de una interfaz de comunicación con una velocidad mínima de 88 megabits por segundo para que se puedan transmitir datos a tiempo real.

La otra opción es cambiar la tasa de muestreo que es configurable; si no se necesita un nivel de detalle de microsegundos en los gráficos a generar a partir de los resultados de los cálculos se puede disminuir la tasa de muestreo. Por ejemplo, si se quiere graficar con una resolución de 0.1ms entre cada punto se configura la plataforma de simulación para que envíe un resultado cada cien ciclos de cálculo. Con esto el tiempo de simulación de 1 segundo se ejecutaría en 7.64s.

Para que el sistema trabaje a tiempo real la tasa de muestreo se debe configurar en 764. Es decir el tiempo entre dos datos sería de 0.764ms. Un ciclo de la señal senoidal a 60Hz tendría 21 datos.

4.3 Simulación de un panel solar con cambio de corriente foto generada

La siguiente prueba que se realizó fue configurar el sistema activando un solo panel solar y cambiando el parámetro de corriente foto generada de 1A a 0.5A a los 1500ms de simulación, luego se continúa la simulación por 500ms más. Los otros parámetros que se utilizaron fueron:

- Capacitor: 0.1mF
- Inductor: 47mH
- Carga: 100Ω
- T: 300K
- Corriente de saturación inversa: 1pA
- Voltaje inicial en el capacitor: 0V
- Corriente inicial en el inductor: 0A
- Delta t: 1µs
- Muestreo: 10µs

Al ejecutar el programa se obtuvieron los datos que se muestran en la figura 4.5. Se puede observar que el voltaje en el capacitor alcanzó un valor de 338V a los 1137ms. Luego se puede observar que a los 1500ms el voltaje en el capacitor comienza a disminuir debido al decremento en la corriente foto generada cuando pasó de 1A a 0.5A, hasta llegar a los 308V a los 2000ms.
Se puede observar también cómo el voltaje en la carga está limitado por el voltaje en el capacitor, mientras el voltaje en el capacitor sea menor a 170V el voltaje en la carga se va a limitar al voltaje en el capacitor. Esto se puede observar con más detalle en la figura 4.6.

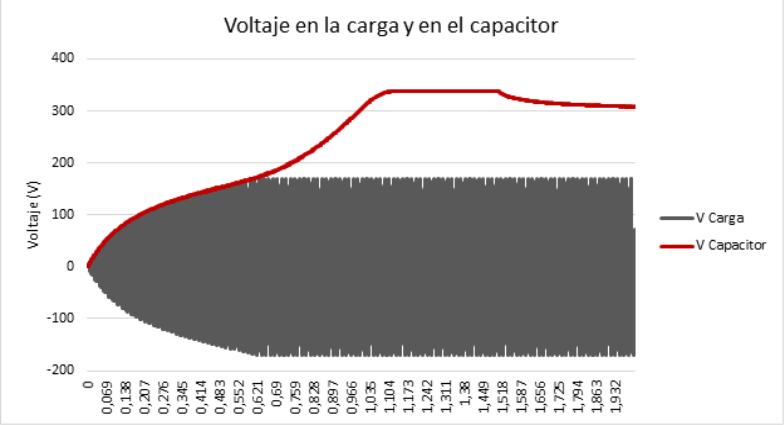

Figura 4.5 Resultado de simulación de un panel solar.

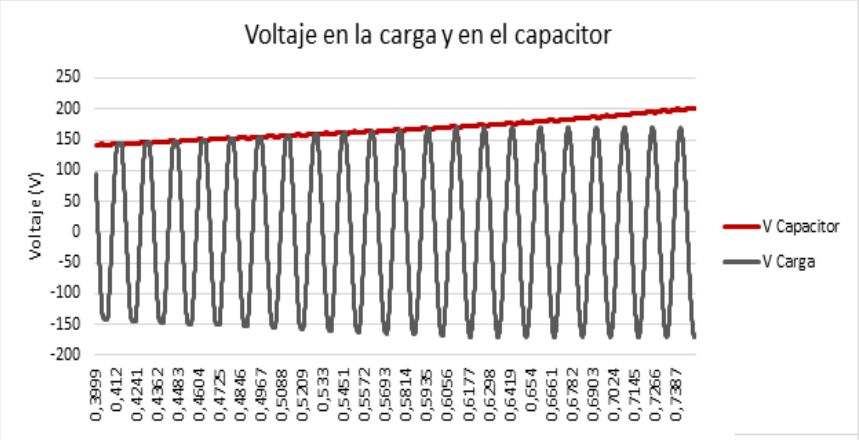

Figura 4.6 Detalle de señales generadas en la simulación.

4.4 Simulación de un panel solar con diferente valor de inductor

La cuarta prueba que se realizó fue cambiar el valor del inductor en el módulo de panel solar para ver la variación en el voltaje de rizo. Para esto se utilizaron los datos obtenidos en la prueba anterior y además se corrió otra simulación con un valor de inductor de 10mH. Todos los demás parámetros del sistema se dejaron con el mismo valor que en la prueba anterior.

Primero se graficó una sección de la señal senoidal obtenida con la prueba en la sección 4.3. Este gráfico se muestra en la figura 4.7. Luego se graficó la misma sección de señal senoidal en la carga esta vez con los datos de la simulación utilizando un valor de inductor de 10mH. Este gráfico se muestra en la figura 4.8.

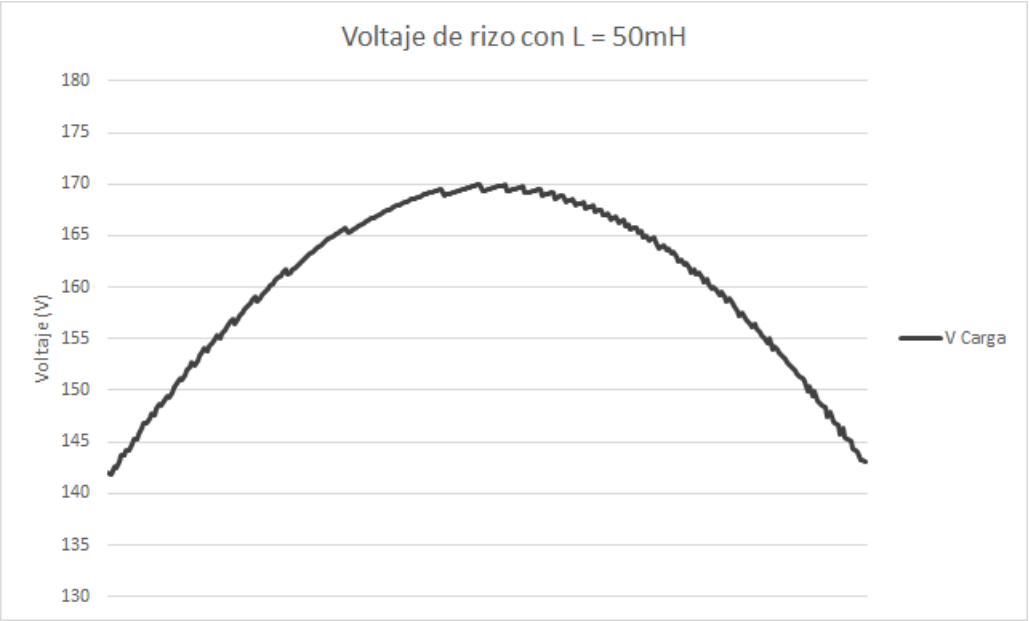

Figura 4.7 Voltaje de rizo con inductor de 50mH.

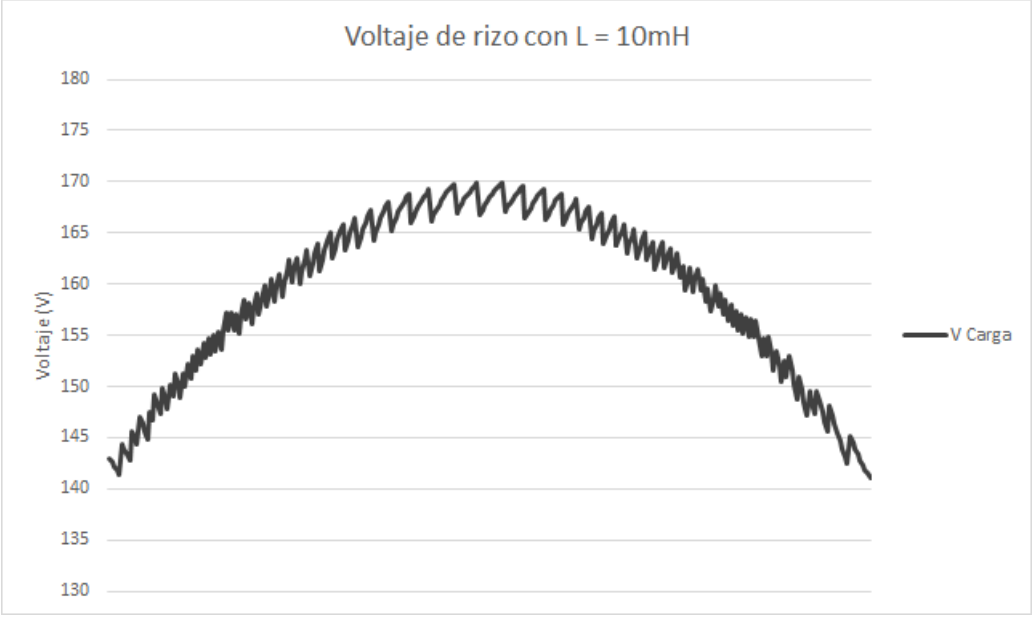

Figura 4.8 Voltaje de rizo con inductor de 10mH.

Con un inductor de 50mH se obtuvo un voltaje de rizo de 0.61 Vpp y con un inductor de 10mH se obtuvo un voltaje de rizo de 3.15Vpp.

4.5 Simulación de un panel solar con cambios en valor de la carga

La quinta prueba consistió en cambiar la carga del circuito. Primero se corrió una simulación aumentando la carga de 100Ω a 50Ω y a 25 Ω. Luego se corrió otra prueba disminuyendo la carga de 100Ω a 200Ωy a 400 Ω.

En esta simulación se utilizaron los mismos valores iniciales que en las dos pruebas anteriores. A los 1.5 segundos de aumenta la carga pasando de 100Ω a 50Ω y a los 2.5s se cambia nuevamente el valor de la carga esta vez a 25Ω.

Con una carga de 100Ω el voltaje en el capacitor alcanza un valor de 338.2V a los 1.1s, luego con una carga de 50Ω el voltaje en el capacitor cae a 314V y el voltaje en la carga continúa en un valor de 170V pico, pero a los 2.5s con una carga de 25Ω el voltaje en el capacitor cae a 45.6V haciendo que la señal de voltaje en la carga tenga un valor pico de 45.6V.

La figura 4.9 muestra las dos caídas en el voltaje en el capacitor y el voltaje en la carga al aumentar la carga en la microred dos veces.

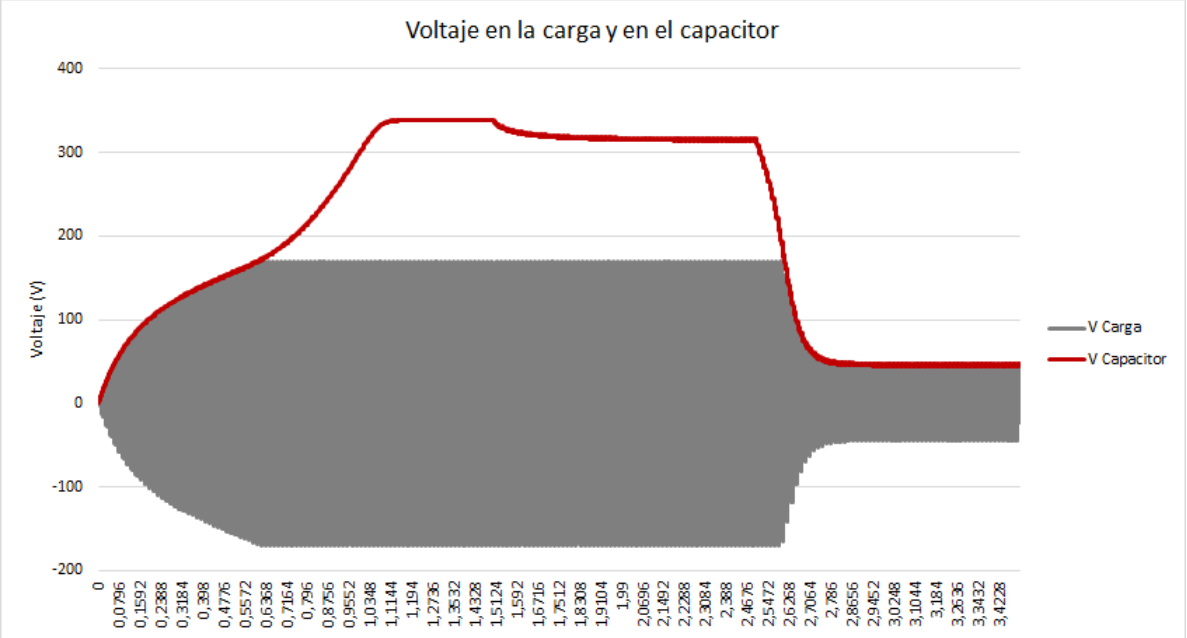

Figura 4.9 Cambio de voltaje al incrementar la carga.

En el otro caso a los 1.5s de simulación la carga se disminuyó pasando de 100Ω a 200Ω y luego a los 2.5s a 400Ω. El voltaje pico en la carga se mantuvo en 170V y el voltaje en el capacitor tuvo un leve aumento aumentó pasando de 338.2V a 343V con la carga de 200Ω y a 344.3V con la carga de 400Ω. Este aumento en el voltaje en el capacitor se puede apreciar en la figura 4.10.

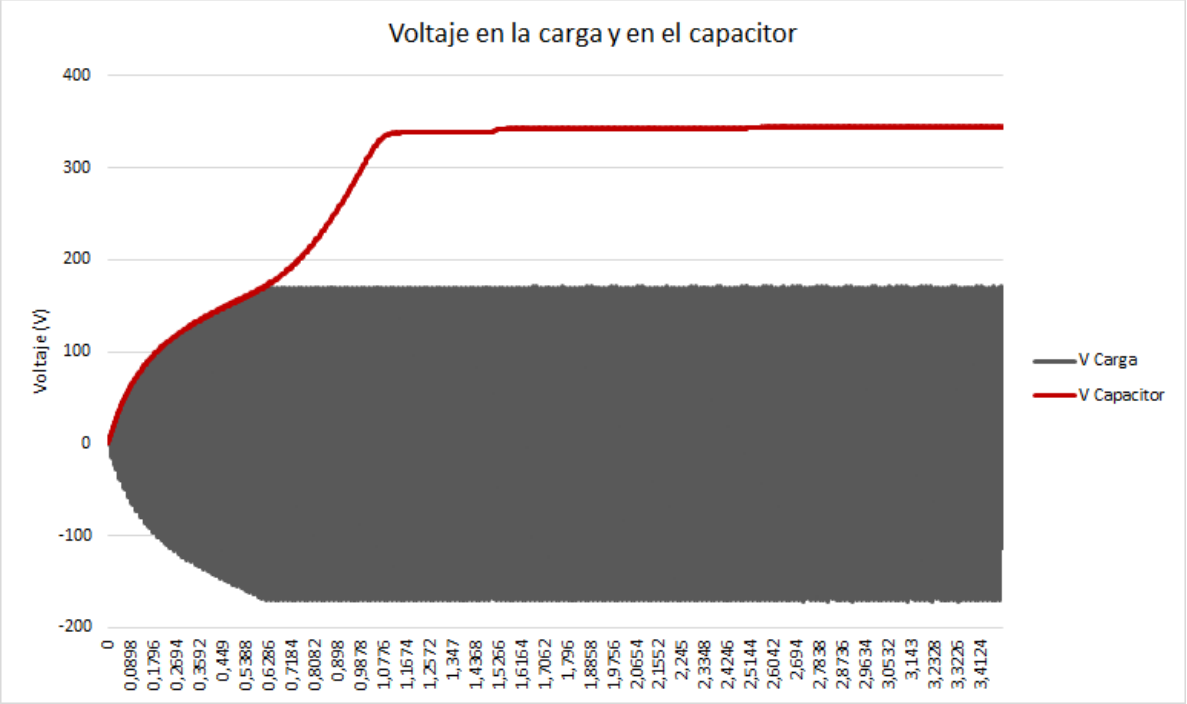

Figura 4.10 Cambio de voltaje al disminuir la carga.

4.6 Simulación de diez paneles solares

Por último se ejecutó una simulación con diez paneles solares activos: cada uno se configuró con una corriente foto generada diferente. Los valores de corriente foto generada utilizados en cada panel solar se muestran en la tabla 4.1. Todos los demás parámetros se configuraron con los mismos valores que en la simulación de un solo panel solar.

Tabla 4.4.1 Valores de corriente foto generada para simulación de 10 paneles solares.

Panel Solar	**Corriente foto generada (A)**
Panel 0	1.00
Panel 1	0.95
Panel 2	0.90
Panel 3	0.85
Panel 4	0.80
Panel 5	0.75
Panel 6	0.70
Panel 7	0.65
Panel 8	0.60
Panel 9	0.55

Con los resultados obtenidos se obtuvo el graficó 4.11 donde se muestra el voltaje en el capacitor de cada panel solar y el gráfico 4.12 donde se muestra la corriente generada por cada panel solar y la corriente total en la carga.

En el gráfico 4.11 se puede ver cómo el panel solar con mayor corriente foto generada alcanza su voltaje máximo en el capacitor en el menor tiempo y además este voltaje es mayor que el de los demás paneles solares. Mientras que el capacitor del panel solar con la menor corriente foto generada fue el último en cargar su capacitor al valor máximo, el cual fue el menor de todos los paneles solares.

En la figura 4.12 se muestran dos ciclos de las señales de corriente generados por los paneles solares y de la corriente total en la carga.

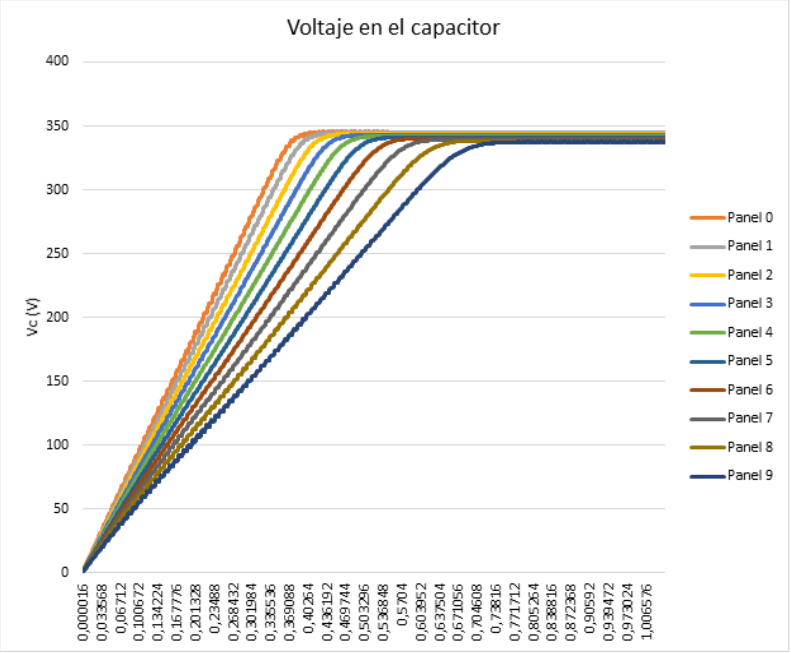

Figura 4.11 Resultados de simulación: Voltaje en capacitor de cada panel solar.

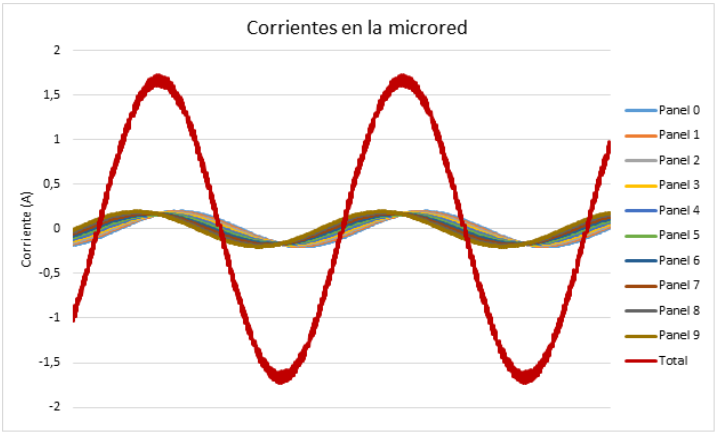

Figura 4.12 Resultados de simulación: Corriente generada por cada panel solar y corriente total en la carga.

4.7 Latencia y utilización de recursos del FPGA

A partir del reporte se utilización del FPGA se realizaron las siguientes tablas donde se muestra cada módulo del procesador de aplicación específica y la cantidad de recursos del FPGA que consume.

También a través de simulaciones de RTL se determinó la latencia de cada uno de los módulos implementados. En el apéndice C se muestran las formas de ondas obtenidas en la simulación de RTL donde se determina la latencia de cada módulo.

4.7.1 Unidad de Control

La unidad de control tiene una latencia de 1 a 3 ciclos de reloj dependiendo de la instrucción que se está ejecutando. La utilización d recursos del FPGA por parte de la unidad de control es baja.

Tabla 4.2 Utilización de recursos por parte de la unidad de control

Recursos	Utilización	
Slice LUTs	1444	2,71%
Slice Registers	232	0,22%
RAM Blocks	0	0,00%
DSPs	0	0,00%

4.7.2 Memoria de Programa

La memoria de programa utiliza una parte importante de los LUTs del FPGA, específicamente un 10% como se muestra en la siguiente tabla.

Tabla 4.3 Utilización de recursos por parte de la memoria de programa

Recursos	Utilización	
Slice LUTs	5451	10,24%
Slice Registers	8351	7,99%
RAM Blocks	0	0,00%
DSPs	0	0,00%

4.7.3 Parámetros del sistema

Este módulo tiene una baja utilización de recursos del FPGA como se muestra en la siguiente tabla. La latencia de este módulo es de un ciclo de reloj.

Tabla 4.4 Utilización de recursos por parte de los parámetros del sistema

Recursos	Utilización	
Slice LUTs	23	0,04%
Slice Registers	148	0,14%
RAM Blocks	0	0,00%
DSPs	0	0,00%

4.7.4 Panel Solar

Este módulo es uno de los que más recursos consume del FPGA. En la siguiente tabla se muestra la utilización de recursos por cada instancia de panel solar.

Tabla 4.5 Utilización de recursos por parte de cada módulo de panel solar,

Recursos	Utilización	
Slice LUTs	3583	6,73%
Slice Registers	1576	1,51%
RAM Blocks	0	0,00%
DSPs	2	0,89%

La latencia de estos módulos es de 87 ciclos de reloj.

4.7.5 Conmutador

La latencia del módulo conmutador es de 22 ciclos de reloj. Como se muestra en la siguiente tabla la mayoría de recursos que consume son los slice LUTs.

Tabla 4.6 Utilización de recursos por parte del conmutador

Recursos	Utilización	
Slice LUTs	1539	2,89%
Slice Registers	551	0,53%
RAM Blocks	2	0,39%
DSPs	1	0,45%

4.7.6 Exponencial

La función exponencial al estar implementada por medio de tablas tiene una latencia de 1 ciclo de reloj. La cantidad de recursos utilizados es baja como se muestra en la siguiente tabla.

Tabla 4.7 Utilización de recursos por parte del módulo exponencial

Recursos	Utilización	
Slice LUTs	271	0,51%
Slice Registers	63	0,06%
RAM Blocks	7,5	1,46%
DSPs	0	0,00%

4.7.7 Salida de datos

Este módulo tiene una latencia de 1 ciclo de reloj. En la siguiente tabla se muestra la utilización de recursos.

Tabla 4.8 Utilización de recursos por parte del módulo de salida de datos

Recursos	Utilización	
Slice LUTs	2765	5,19%
Slice Registers	205	0,20%
RAM Blocks	0	0,00%
DSPs	0	0,00%

5 Comparación con un entorno de simulación basado en un procesador de propósito general

En al artículo incluido al final de esta tesis, en el apéndice D, se desarrolló el estudio para comparar el sistema de simulación propuesto con un entorno de simulación basado en Python que se ejecuta en un procesador de propósito general.

Para el cálculo del tiempo de simulación se tomó en cuenta sólo el período de tiempo utilizado en los cálculos. En ambos casos se simularon microrredes con exactamente los mismos parámetros y se corrió una simulación de cien milisegundos.

En el caso de la simulación basada en Python se utilizó una función que imprime el tiempo antes y después de realizados los cálculos y así se obtiene el tiempo que le tomó al procesador correr los cien milisegundos de simulación. En el caso del sistema propuesto se implementó un contador que se incrementa cada ciclo de reloj, este contador se inicializa en 0 al inicio de la simulación y se lee al final, de esta forma se obtienen los ciclos de reloj utilizados y se traducen a tiempo de ejecución. La Tabla 5.1 muestra los resultados de simulación para ambos casos.

Tabla 5.1 Tiempo de ejecución de simulación de la microrred aislada para distinta cantidad de sistemas panel e inversor.

Cantidad de paneles e inversores en simulación	**Procesador Propósito General (ms)**	**Procesador de Aplicación Específica (ms)**
1	956	130
2	1179	142
4	1475	167
8	2242	219
16	3352	320

En el artículo adjunto se pueden apreciar más resultados entre ambos sistemas.

6 Conclusiones

La arquitectura propuesta para un sistema simulador de microredes permitió implementar un circuito capaz de resolver las ecuaciones que describen una microred y generar los valores de voltajes y corrientes en la misma.

La arquitectura permitió realizar la mayoría de cálculos en paralelo y que la variación en el tiempo de ejecución de un ciclo de cálculo fuera de solo cinco ciclos de reloj por cada módulo de panel solar que se habilite en una prueba.

Con las instrucciones para el procesador de simulación se pudieron escribir programas para realizar diferentes pruebas utilizando desde uno hasta diez paneles solares, hacer cambios en los parámetros del sistema durante la ejecución y enviar los resultados deseados a un PC por medio del puerto serial.

Con los resultados recibidos en el PC se pudieron analizar los datos para determinar valores máximos o mínimos, determinar tiempos de respuesta y para hacer gráficos de las señales de voltajes y corrientes en una microred.

Bibliografía

[1] Navigant Consulting. Microgrids Research Assessment: Phase 2 Final Report. Prepared for DOE's Office of Electricity Delivery and Energy Reliability. California, May 2006.

[2] (2014, Dic.) European Commission website on Microgrids [Online]. Available: http://www.microgrids.eu

[3] I. Strnad, D. Skrlec, and T. Tomisa, "A model for the efficient use of electricity produced from renewable energy sources for electric vehicle charging," in 2013 4th International Youth Conference on Energy (IYCE), 2013, pp. 1–8.

[4] E. Kremers, P. Viejo, O. Barambones, and J. M. González de Durana, "A Complex Systems Modelling Approach for Decentralised Simulation of Electrical Microgrids," in 2010 15th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), 2010, pp. 302–311.

[5] (2014, Dic.) AnyLogic website [Online]. Available: http://www.anylogic.com/

[6] I. Strnad, D. Skrlec, and T. Tomisa, "A model for the efficient use of electricity produced from renewable energy sources for electric vehicle charging," in 2013 4th International Youth Conference on Energy (IYCE), 2013, pp. 1–8.

[7] C. Sun, P. Li, C. Ding, G. Song, C. Wang, H. Huang, F. Peng, and H. Yin, "Dynamic time-domain simulation and analysis of medium-low voltage microgrid," in TENCON 2013 - 2013 IEEE Region 10 Conference (31194), 2013, pp. 1–4.

[8] W. Su, J. Wang, and J. Roh, "Stochastic Energy Scheduling in Microgrids With Intermittent Renewable Energy Resources," IEEE Transactions on Smart Grid, vol. 5, no. 4, pp. 1876–1883, Jul. 2014.

[9] IBM ILOG CPLEX Optimization Solver [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

[10] O. Gomis-Bellmunt, A. Sumper, A. Colet-Subirachs, A. Ruiz-Alvarez, F. Alvarez-Cuevas-Figuerola, and A. Sudria-Andreu, "A utility connected microgrid based on power emulators," in 2011 IEEE Power and Energy Society General Meeting, 2011, pp. 1–6.

[11] J.-H. Jeon, J.-Y. Kim, H.-M. Kim, S.-K. Kim, C. Cho, J.-M. Kim, J.-B. Ahn, and K.-Y. Nam, "Development of Hardware In-the-Loop Simulation System for Testing Operation and Control Functions of Microgrid," IEEE Transactions on Power Electronics, vol. 25, no. 12, pp. 2919–2929, Dec. 2010.

[12] Department of Electrical Engineering, Information Technology and Cybernetics, “Hardware-in-the-Loop Simulation”, Telemark University College.

[13] (2014, Dic.) FICO® Xpress Optimization Suite website [Online]. Available: http://www.a14a1.com/en/products/fico-xpress-optimization-suite/

[14] OpenDSS EPRI Distribution System Simulator [Online]. Available: http://sourceforge.net/projects/electricdss/

[15] Solar Cell Structure [Online]. Available: http://www.pveducation.org/pvcdrom/solar-cell-structure

[16] Margarida F. Jacome, Gustavo de Veciana. “Design Challenges for New Application-Specific Processors.” IEEE Design & Test of Computers. April–June 2000, pp 40-50.

Apéndice A Funcionalidad del DSP48E1

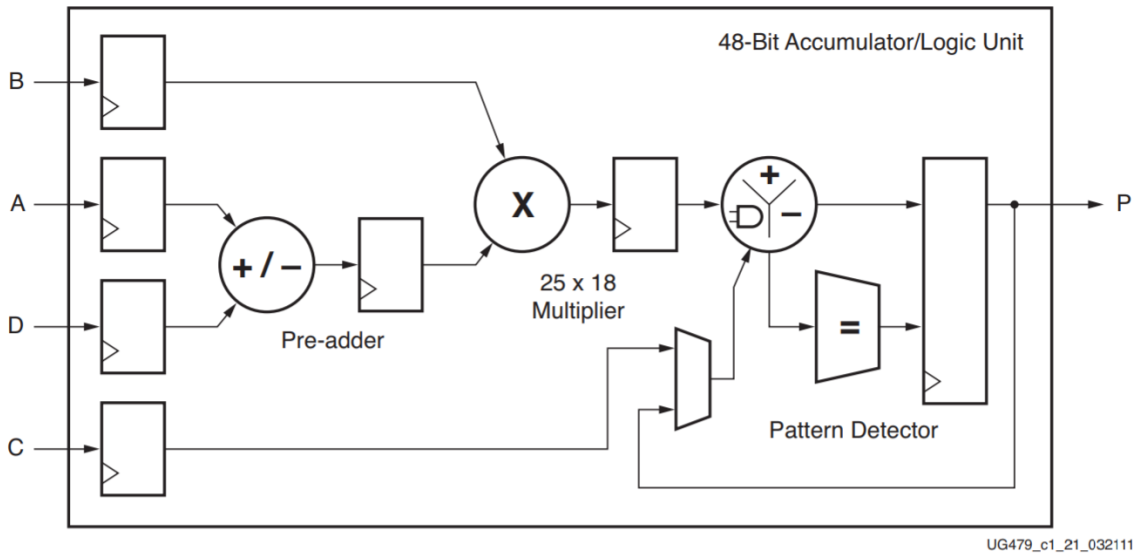

Figure 1-1: **Basic DSP48E1 Slice Functionality**

Apéndice B Instrucciones del procesador de aplicación específica

WR_REG

Escribe un valor de 64 bits en un registro del procesador, este registro puede estar en el módulo de parámetros generales o en alguno de los módulos de panel solar.

Palabra	Bits			
	31:24	23:16	15:8	7:0
0	0x01		Módulo destino	Registro destino
1	dato[31:0]			
2	dato[63:32]			

RD_REG

Lee un valor de 64 bits en un registro del procesador, este registro puede estar en el módulo de parámetros generales o en alguno de los módulos de panel solar.

Palabra	Bits			
	31:24	23:16	15:8	7:0
0	0x02		Módulo destino	Registro destino

SET_CYCLES

Esta instrucción se utiliza para escribir un valor en el contador que se utiliza para ejecutar ciclos en el programa.

Palabra	Bits																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0x04								Contador[23:0]																							

DEC_CYCLES

Decrementa en 1 el contador de ciclos y luego compara el valor con 0. Si el valor es diferente a 0 resta al contador de programa el offset que recibe como argumento de la función.

Palabra	Bits																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

0	0x05																							offset								

EXEC_CALC

Esta instrucción envía la señal de inicio de cálculo a los módulos indicados en el argumento.

Palabra	Bits																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0x06																															
1	module_selected[31:0]																															

WAIT_CALC

Esta instrucción espera a que los módulos indicados en el argumento hayan terminado de ejecutar un cálculo.

Palabra	Bits																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0x07																															
1	module_selected[31:0]																															

PRINT

Envía el resultado indicado por el campo índice al PC. Además utiliza el bit 16 para agregar un espacio al final del número o un carácter de retorno de línea.

Palabra	Bits																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0x01															F											índice					

El carácter o caracteres a enviar después del dato seleccionado depende del bit F:

F = 0: se envía carácter de espacio.

F = 1: se envían carácter de retorno y de salto de línea.

RFI

Esta instrucción se utiliza para regresar del código de interrupción.

Palabra	Bits																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0x09																															

Cuando se ejecuta esta instrucción el contador de programa se carga con la dirección que tenía antes de iniciar a ejecutar el código de interrupción.

REINIT

Carga el contador de programa con un valor de 0x00 por lo que se reinicia la ejecución del programa.

Palabra	Bits																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0x0B																															

STOP

Esta instrucción hace que el procesador termine de ejecutar instrucciones. Para volver a correr el programa se hace a través de la interface de control entre el ARM y el procesador.

Palabra	Bits																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0xFF																															

Apéndice C Formas de onda obtenidas en simulación

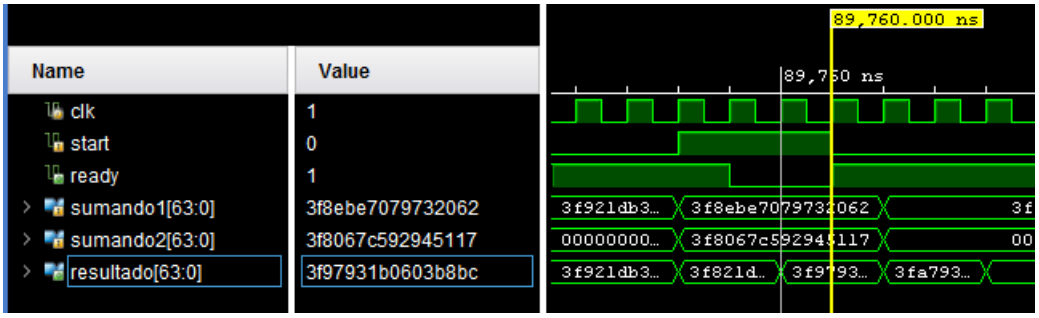

Figura C.1 Latencia del módulo de suma

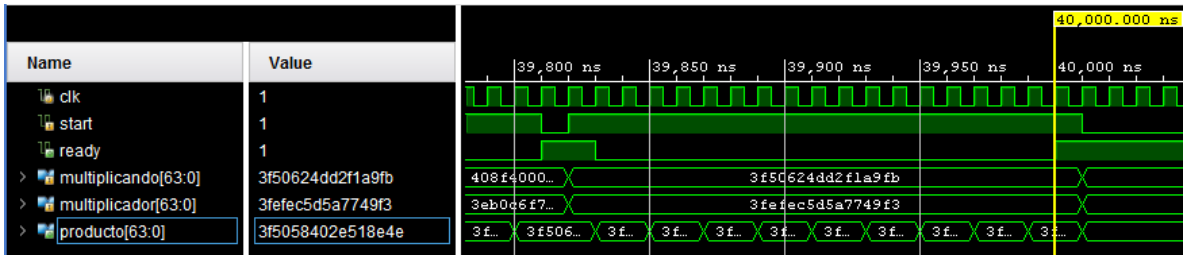

Figura C.2 Latencia del módulo de multiplicación

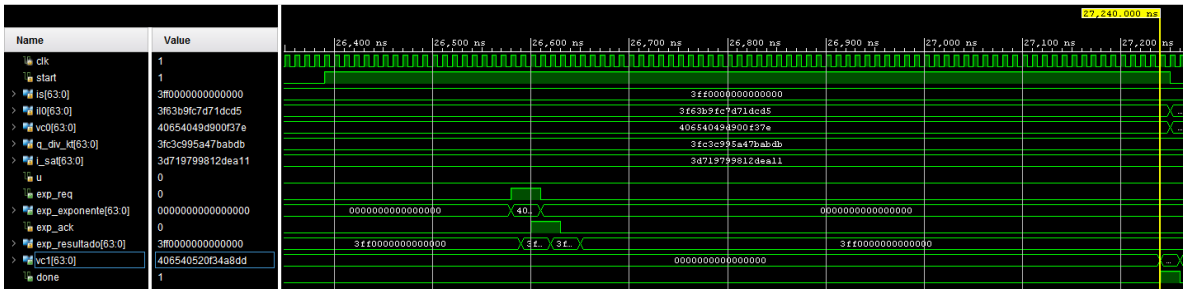

Figura C.3 Latencia del módulo de integral Vc

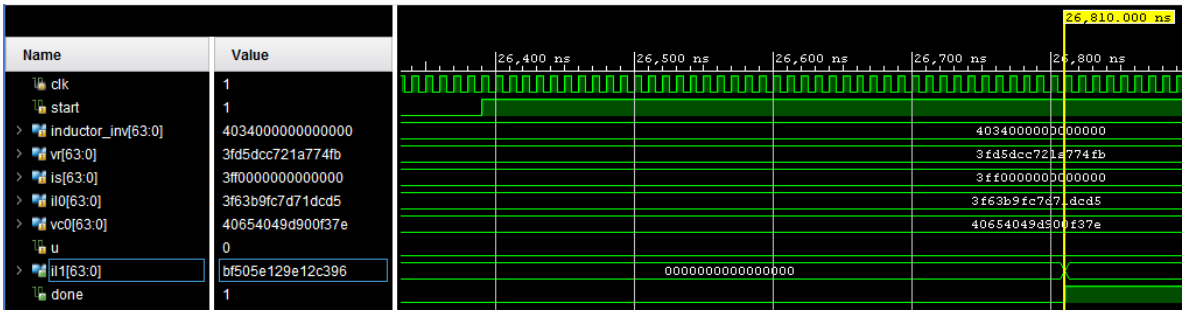

Figura C.4 Latencia del módulo de integral IL

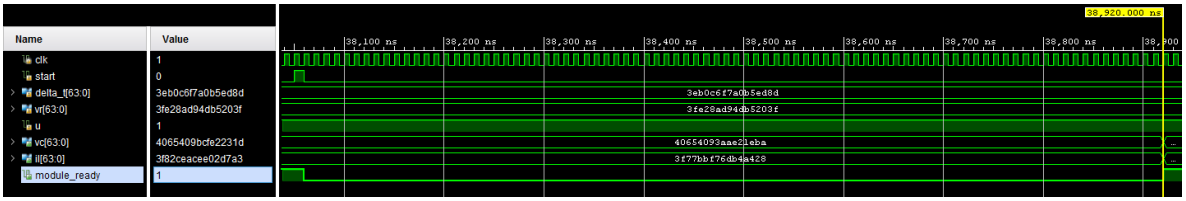

Figura C.5 Latencia del módulo de Panel Solar

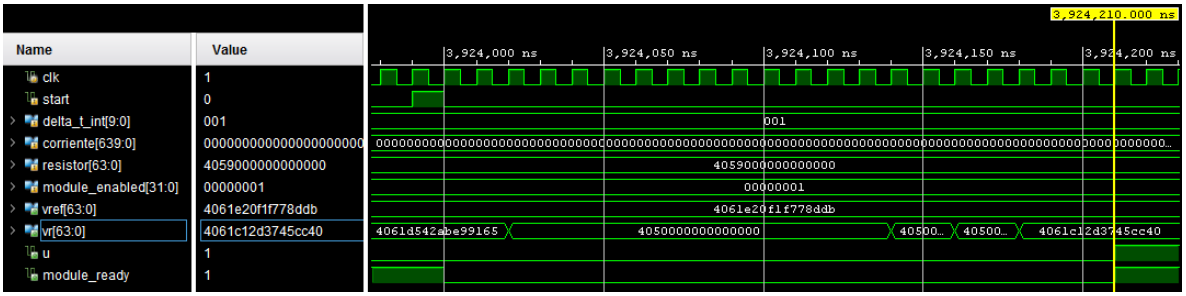

Figura C.6 Latencia del módulo conmutador

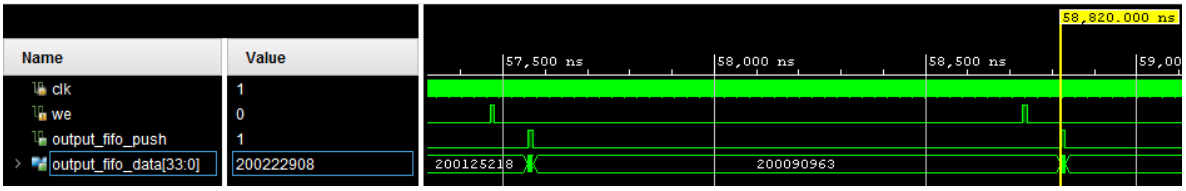

Figura C.7 Tiempo de ejecución de un ciclo de cálculo

Apéndice D Artículo Procesador de Aplicación Específica para Simulación de Microrredes

An Application-specific Instruction Set Processor for Microgrid Simulation

Edgar Mauricio Brenes
Aruba Networks
Hewlett Packard Enterprise
Heredia, Costa Rica
Email: edgar.brenes@hpe.com

Carlos Meza
Electronic Engineering School
Instituto Tecnologico de Costa Rica
Cartago, Costa Rica
Email: cmeza@itcr.ac.cr

***Abstract*—Microgrid represents a new paradigm in the power sector that offers more reliability and flexibility for electricity delivery. A microgrid consists of different types of power generation units, loads and energy storage systems that are controlled and coordinated. Smart power processing units play an important role in a microgrid. The present paper presents an application-specific instruction set processor for the study of isolated microgrid power converters. The proposed processor has been developed in order to solve differential algebraic equations that describe the dynamical behavior of microgrid power processor.**

I. Introduction

A microgrid is a subsystem of a larger electrical grid which consists of a collection of distributed power generators and loads located in a nearby area [1]. A typical diagram of a microgrid is depicted in Figure 1.

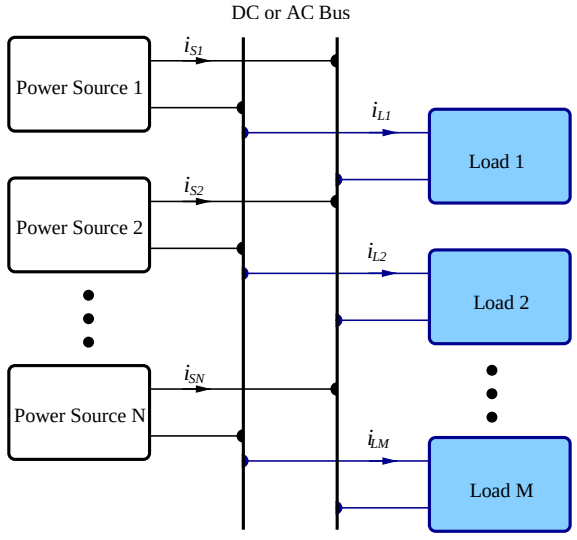

Figure 1. Diagram of a typical microgrid

One of the biggest challenges in the design and operation of microgrids is that they are inherently asymmetrical and heterogeneous [2], i.e., they present several different types microgenerators and single phase loads. Additionally, it is also desired that microgrid continue operating with loss of any component and any generator (plug-and-play capability) [1].

Another important challenge in plug-and-play microgrid is to have a dynamic fast-acting load balancing system. A microgrid can operate connected to the utility grid or as a stand-alone system. This paper deals only with stand-alone microgrids.

Smart power processing devices (e.g. power inverters and converters) can provide the necessary functionality to deal with the aforementioned challenges. Such power processing units are considered intertialess [2] and can potentially intervene to correct a failure or imbalance almost instantly. A multi-level hierarchical management and control system is able to operate the power processing units in a coordinated way [3].

Giving the importance of power processing units in microgrids it will be useful to study in detail their interaction in a microgrid. A simulation environment that is able to accomplish the aforementioned should make explicit the behavior of the system in a wide range of time scales, i.e., it should be able to simulate the dynamics of the system from miliseconds to hours. Large processing power is required in order to simulate the system in the required time scales considering also the inherently complexity and nonlinearity of the microgrid.

The cost reduction and performance increment of fieldprogrammable gate arrays (FPGA) in the last years [4] have make it possible to design and implement application-specific instruction set processor (ASIP) that can be used for a wide range of applications such as video processing [5], [6], advanced encryption [7], power monitoring [8] and monitor seismic activity [9]. The advantages offered by ASIPs is that they can be fine tunned to perform specific task more efficiently and faster than general purpose processors.

In this regards, the present work proposes an ASIP designed to be the power core of a simulation environment for microgrids. This proposed system is able to simulate concurrently several microgenerators and loads in a few miliseconds, solving several non-linear differential algebraic equations.

The rest of the papers is structured as follows: First, a simple stand-alone microgrid structure is presented. This microgrid topology will be used for design and evaluation of the ASIPbased simulation environment. In Section III-A the architecture of the ASIP is described. Next, a simple microgrid system is presented and simulated using the proposed simulation environment. Finally, section V presents the main conclusions and the future activities of this work.

II. A Stand Alone Microgrid Structure

The proposed ASIP and simulation environment have been designed based on a specific microgrid structure. The idea behind this approach is to validate the usefulness of an ASIP-based simulation environment for a simple microgrid configuration so that it can be later extended to more complex microgrid topologies.

The microgrid structure used in this work consists of *n* power sources connected in parallel to an alternate current (AC) bus, as depicted in Figure 2. Each power source (PS) injects current to the AC bus through a power unit (PU) that is controlled by a local current controller (LCC). The current reference for each LCC is provided by a global current controller (GCC) as seen in Figure 2.

Notice that by having the GCC define the current references of each LCC, (i^*_{Si}), all the currents generated by the power sources, i_{Si} will be in phase. More specifically, the GCC is in charge of:

- identifying the total current required by the load such that the AC bus voltage, v_{AC}, is equal to a reference value, $vAC*$,
- determining the power contribution of each power source by providing the desired value of the current (i_{Si}) that each power source needs to inject to the AC bus.

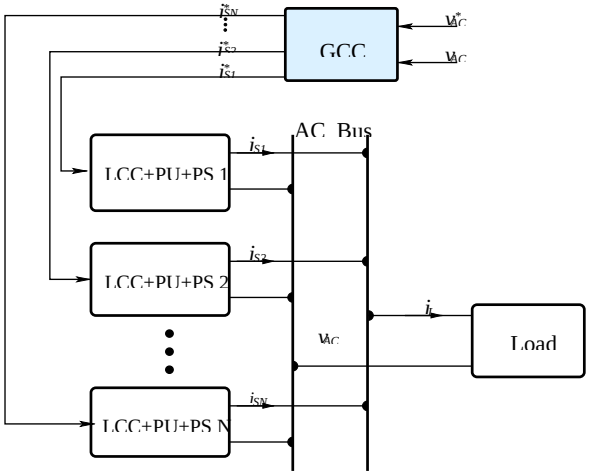

Figure 2. Structure of an Isolated Microgrid. GCC=Global Current Controller, LCC=Local current controller, PU=Power Unit, PS=Power source

As seen in Figure 2 each power source is associated to a power unit and a local current controller (LCC+PU+PS). A more detailed representation of this subsystem is shown in Figure 3, where the power source is a photovoltaic generator and the power unit is a full-bridge inverter. The LCC provides the required signals, μ_{1i}, μ_{2i}, μ_{3i}, μ_{4i}, to turn on the MOSFETS such that the full-bridge output current, i_{Si}, is equal to the reference value, i^*_{Si}. This subsystem can be modeled according to the following set of differential equations

$$C_i \frac{dv_{C_i}}{dt} = i_{\mathrm{pv}_i} - u_i i_{S_i} \quad (1)$$

$$L_i \frac{di_{S_i}}{dt} = u_i v_{C_i} - v_{AC} \quad (2)$$

where C_i, L_i represent the capacitance and inductance, respectively, of the $i-th$ PVG+Full bridge unit. $u_i \in 1,-1$ is the output of the LCC used to generate μ_{1i}, μ_{2i}, μ_{3i}, μ_{4i} of Fig. 3 in the following way

$$u_i = 1 \Leftrightarrow \mu_{1i} = \mu_{4i} = 1 \Leftrightarrow \mu_{2i} = \mu_{3i} = 0 \ (3)$$

$$u_i = -1 \Leftrightarrow \mu_{1i} = \mu_{4i} = 0 \Leftrightarrow \mu_{2i} = \mu_{3i} = 1 \ (4)$$

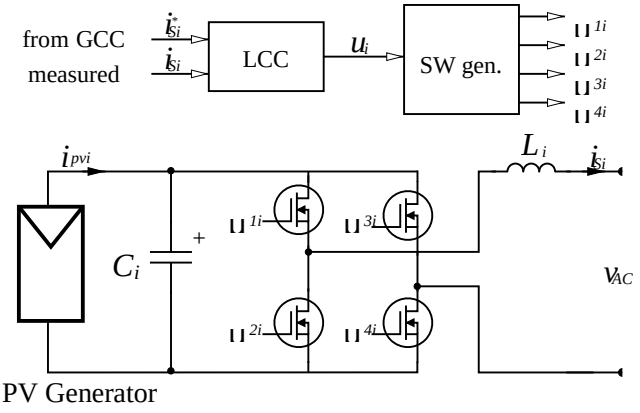

Figure 3. PV Power Source Diagram

Each photovoltaic generator (PVG) represents a string of *r* PV modules with the electrical characteristics of Fig.4 which can also be described as follows

$$i_{pvi} = I_{gi} - Isi(e^{\alpha i v_{pvi}} - 1) \quad (5)$$

where i_{pvi} and v_{pvi} are the current generated and the voltage at the *i*-th PV generator and I_{gi}, I_{si} and α_i are its parameters.

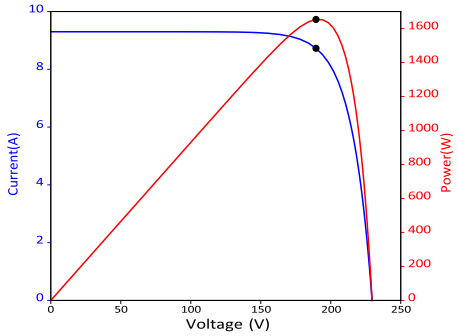

Figure 4. *i-v* and *P-v* curves of a PV Generator

As mentioned previously, the control scheme, i.e., the GCC and LCC, is in charge of making the current delivered to the grid such that the AC bus voltage is sinusoidal with constant frequency and amplitude. Moreover, the AC bus voltage can be defined as a function of the delivered current, i.e., $v_{AC} = f(i_L)$. In this regard, the control scheme must be designed considering $f(i_L)$. In this work, the load is modeled as a resistor, i.e.,

$$v_{AC} = R_L \sum_{i=1}^{i=n} i_{Si} \quad (6)$$

Nevertheless, modifying (6) and redesigning the controller it is possible to consider reactive and nonlinear loads.

Notice that the proposed case of study can be simulated solving a set of non-linear differential algebraic equations (DAE), i.e.,(1),(2),(5) and (6). In this case the order of the DAE will be $2n$, where n is the number of PVG+Full-bridges presented in the system.

III. ASIP-BASED SIMULATION ENVIRONMENT

The proposed simulation environment consists of a ASIP, and ARM processor and general purpose computer. The ASIP is controlled by an ARM processor, which is connected to a PC for interfacing with the user. A general diagram of the system is shown in figure Fig 5. Notice that the ARM processor and the ASIP can be easily integrated in a single silicon device, i.e., a field-programmable gate array (FPGA) or an application specific integrated circuit (ASIC). The ARM

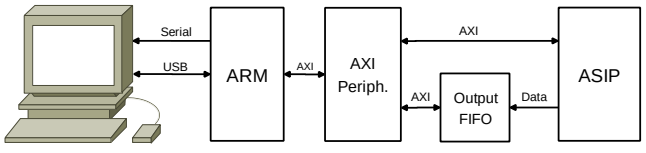

Figure 5. System Architecture

processor sends to the ASIP the simulation program to be executed and controls the execution of the program. The ASIP program memory and control registers are memory mapped in the ARM memory space. The ARM processor also polls the Output FIFO and if there is data available it pops the data from the FIFO and sends it to the computer. Communication between ARM processor and ASIP is through an AXI interface. The ASIP executes the program which has the instructions to generate the simulation data, it also sends the data to the Output FIFO when the print instruction is executed. The ASIP communicates with a general purpose computer to transmit all the simulated data for detailed data analysis and visualization. The communication is also required so that the computer can download the simulation program that will be executed in the ASIP. The computer program includes information about the configuration of the microgrid such as value of the load, number of active energy sources and photogenerated current in each solar panel.

A. The ASIP Architecture

The ASIP architecture is shown in figure Fig 6. The architecture allows execution of mathematical operations concurrently in order to solve equations (1),(2) and (6) in a short period of time. Operations are executed using IEEE 754 double precision floating point format. This number representation was preferred over fixed point because operands in (5) can be of order -12 and +12, needing at least 40 bits for integer value and 40 bits for fractional value.

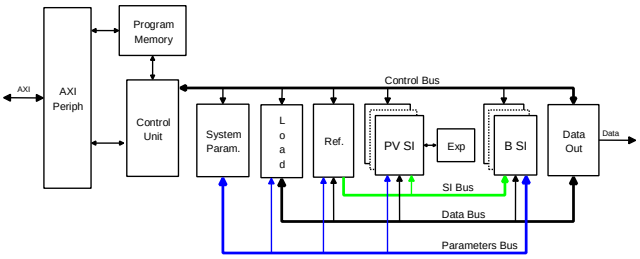

Figure 6. ASIP Architecture

The following subsections describe main modules of the ASIP.

1) Control unit: The control unit fetches instructions from the Program Memory and executes them. This module communicates with the other modules of the processor through a data bus and a control bus. Data bus is used to read and write parameters in each module. Control bus is used by the control unit to indicate a module when it has to execute an operation and to be notified when a module has complete it. There is one dedicated start signal for each module in the processor so the control unit can indicate to different modules to start an operation in the same clock cycle. The simulation program instructions are stored in the Program Memory by the ARM processor through an AXI interface. This memory is mapped in the ARM memory space so an access to an

address in that memory region will access the Program Memory directly.

2) PV Source Inverter and Controller: Photovoltaic Source Inverter module (PVSI) calculates the output current of a photovoltaic generator and the voltage on its internal capacitor by solving equations (1) and (2). This unit implements in the ASIP the LCC+PU+PS depicted in Figure 3. In Figure 7 the structure of the PVSI implemented in the ASIP is shown.

3)

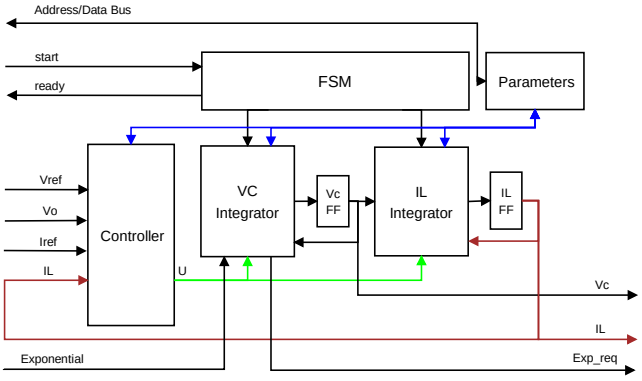

Figure 7. PVSI Block Diagram

Submodule VC Integral solves equation (1) and submodule IL Integral solves equation (2). Each of these two modules have a floating point multiplier, a floating point adder, and a finite state machine with the corresponding sequence of mathematical calculations and data movement to solve the equations.

PVSI has a finite states machine (FSM) to control the other submodules. When the ASIP Control Unit asserts the start signal the FSM triggers execution of mathematical operations to get values of output current and voltage capacitor in the photovoltaic generator, these two operations are executed in parallel. When both calculations are complete the FSM asserts the ready signal indicating to the ASIP Control Unit that a calculation cycle has complete.

Parameters module in PVSI store the values specific to a photovoltaic generator that are needed for the mathematical calculations, these values are: capacitance, inductance, reverse saturation current, initial voltage in capacitor, initial current in inductor, and number of solar cells in the photovoltaic generator. These values are written by the ASIP Control Unit through the data bus.

The controller in PVSI determines the value of *u* which is used to control the inverter switches as shown in Figure 3. It has two modes of operation: based on reference voltage or based on current, mode of operation is selected through a PVSI parameter. If control is based on reference voltage a comparator compares the value on the load against the value on the reference voltage, if voltage in the load is bigger than reference voltage then a value of -1 is assigned to *u*, otherwise *u* is assigned with +1. When control is based on current the value of *u* is determined comparing the current provided by the PVSI and the target current indicated by the reference unit, similarly if PVSI current is bigger than reference current *u* is assigned with -1, otherwise it is assigned with +1.

The number of PVSI modules in the ASIP is determined by a parameter, so it can be changed at synthesis time.

3) Battery Source Inverter and Controller: Battery Source Inverter and Controller unit (BSI) calculates the output current of a battery and an inverter. It works similar to a PVSI with the difference that it does not include VC Integrator submodule as the voltage is provided by the battery. It does include the IL Integrator module to calculate the current in the inductor of the inverter.

4) Exponential Unit: Exponential module receives a floating point value and returns its exponential value also in floating point format. Exponential value is needed by the VC Integrator in PVSI module, as shown in Fig 7, to calculate the output current (equation (5)). In order to reduce simulation time a set of lookup tables were implemented to generate the exponential value of a given number. There are different lookup tables for different operation ranges. When this unit receives a request it converts the operand to an integer and a fractional number, then depending on the values this unit extracts the exponential value from the corresponding lookup table. Given that this module takes a lot of resources there is only one exponential unit shared by all the PVSI units, so an arbitration method was implemented to return the corresponding exponential value to the different requesters.

5) Reference Unit: The Reference Unit generates the voltage of reference or the current of reference used by the PVSI and BSI units to calculate their corresponding value for *u*. This unit implements the GCC depicted in Figure 2. The generated voltage of reference is a sinusoidal waveform with a peak amplitude of 170V or 120V RMS. All PVSI and BSI units receive the same voltage of reference and each of them generates the corresponding value of *u* in order to follow this input voltage. This unit can also be configured to generate a different current of reference for each PVSI and BSI units. Based on the configured parameters this module calculates the current that each source have to inject

6) Load Unit: Load unit calculates voltage on the load by adding the current generated by all the energy sources and multiplying it by the value of the load as described by (6).

7) Instruction Set: The ASIP is controlled by instructions stored in the Program Memory. Instructions are one, two or three words long, where every word is 64 bits width. Every instruction is composed by an eight bits operation code and the arguments. Supported instructions are shown in Table I

Table I
ASIP INSTRUCTIONS

Instruction	Mnemonic	Arguments
Write parameter	WR	address, data
Read parameter	RD	address
Set cycles counter	SETC	number of cycles
Decrement and jump if not zero	DJNZ	address offset
Enable source	ENAB	modules
Execute calculation	EXEC	modules
Wait calculation	WAIT	modules
Print data	PRINT	index
Stop program execution	STOP	

IV. VALIDATION AND SIMULATION RESULTS

The ASIP and ARM processor were implemented in a FPGA Xilinx Zynq-7000 AP SoC XC7Z020-CLG484. Table II shows the resources utilized when synthesizing the ASIP with eight PVSI units and one BSI unit.

Table II
FPGA RESOURCES UTILIZATION

Resource	Utilization
Slice LUTs	43056 (81%)
Slice Registers	25088 (24%)
RAM Blocks	133 (26%)
DSPs	18 (8%)

A typical program that the ASIP processor runs is shown in Figure 8.

A. Validating the proposed simulation environment

In order to validate the developed ASIP-based simulation environment two cases have been tested. Such cases were simulated in the proposed environment and in a Python program running in a general purpose processor (GPP) of a typical personal computer.

The general purpose processor used was the Intel(R) Core(TM) i5-4200M CPU. The differential equations that describe the system were simulated in the GPP with Python 2.7 executed in a machine with Fedora Linux distribution.

The details of the cases simulated in the aforementioned platforms are shown in Table III. Each PV panel has the electrical characteristics shown in Fig. 4.

Figures 9 and 10 show the results of Case A. The system simulated in GPP solved the differential equations using the Runge-Kutta method of the Odespy Pythons library [10].

```
WR 0x00 , 0x00 , 0x4034000000000000 / / C i r c
 u i t load = 20 Ohms
WR 0x00 , 0x01 , 0x3eb0c6f7a0b5ed8d / / d e l t a
 t = 1us
WR 0x00 , 0x03 , 0x0000000000000009 / /
 sampling = 1 every 10 cycles
WR 0x02 , 0x00 , 0x4069000000000000 / / PVSI0
 Capacitor = 5mF (1/200)
WR 0x02 , 0x01 , 0x403546ce01000201
/ / PVSI0 Inductor = 47mH (1/21.2766)
WR 0x02 , 0x02 , 0x4022999999999a / / PVSI0
 Photocurrent = 9.3A
WR 0x02 , 0x03 , 0x4070e00000000000 / / PVSI0 I
 n i t i c a l Vc = 270V
WR 0x02 , 0x04 , 0x0000000000000000 / / PVSI0 I
 n i t i a l IL = 0A
WR 0x02 , 0x05 , 0x3eb21e908ed8f651 / / Dark
 sat . curr . = 1 ,08uA
WR 0x02 , 0x06 , 0x3faddddddddddde / / q / kTn
 = 0 ,05833
WR 0x0A , 0x01 , 0x403546ce01000201
/ / Bat0 Inductor = 47mH (1/21.2766)
WR 0x0A , 0x03 , 0x4070e00000000000 / / Bat0
 Vbat = 370V
WR 0x0A , 0x04 , 0x0000000000000000
/ / Bat0 I n i t i a l IL = 0A
ENAB 0x00000101 / / Enable PVSI0 and Bat0
SETC 0x0186A0 / / Cycles counter = 100.000
EXEC 0x0404 / / Calc PVSI0 & Bat0 IL
WAIT 0x0404 / / Wait c a l c u l a t i o n
EXEC 0x0002 / / Calculate load voltage
WAIT 0x0002 / / Wait c a l c u l a t i o n
PRINT 0x02 / / P r i n t V load
PRINT 0x03 / / P r i n t V reference
PRINT 0x04 / / P r i n t PVSI0 IL
PRINT 0x05 / / P r i n t PVSI0 Vc
PRINT 0x14 / / P r i n t Bat0 IL
PRINT 0x15 / / P r i n t Bat0 Vbat
DJNZ 0x0F / / Dec and jump i f not zero
STOP / / Stop simulation
```

Figure 8. ASIP Program for Simulation Case A (Table III)

Table III
PARAMETERS OF THE CASE SIMULATED

Case	Number of PV Generators	Composition of PVG	Load
A	1	1 string with 6 PV panels	20 Ω
B	4	2 strings with 6 PV panels and 2 strings with 5 PV panels	20 Ω

Figures 11 and 12 show the results of Case B. As mentioned previously, Case B represents a microgrid with 4 PV power sources, where each power source is associated with a fullbridge converter. The aforementioned figures shown that the proposed FPGA-based simulation environment generates identical responses as a conventional software based simulation.

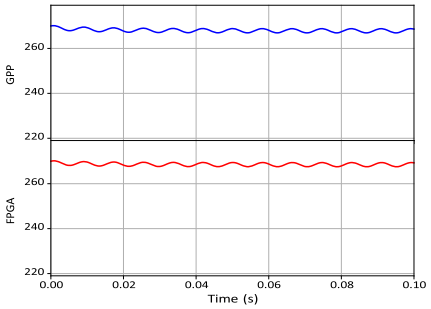

Figure 9. Simulation results: voltage in the PVG for Case A using a FPGAbased simulation environment and a conventional general purpose processor.

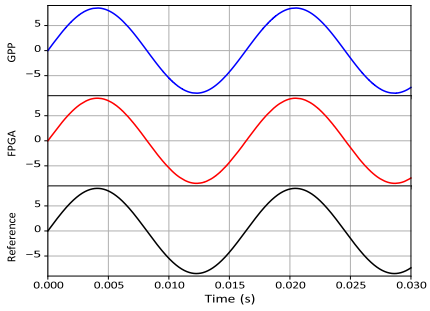

Figure 10. Simulation results: output current for Case A using a FPGA-based simulation environment and a conventional general purpose processor.

B. Comparing the simulation time

In order to compare the execution time of both simulation a microgrid with 1, 2, 4, 8 and 16 photovoltaic power sources were simulated. For these cases, both simulation systems solved the differential equations using the ForwardEuler method. The simulation time for each case are shown in Table IV, as expected the FPGA-based simulation environment is around 10 times faster than the GPP. Additionally, as shown in 13, for the FPGA-based system, the simulation time grows

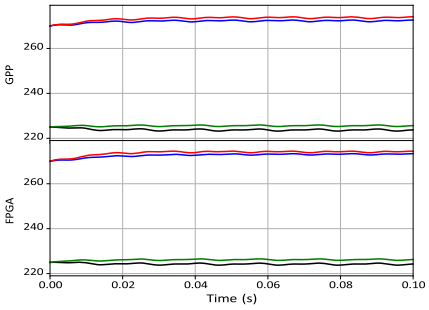

Figure 11. Simulation results: voltage in each PVG for Case B using a FPGAbased simulation environment and a conventional general purpose processor.

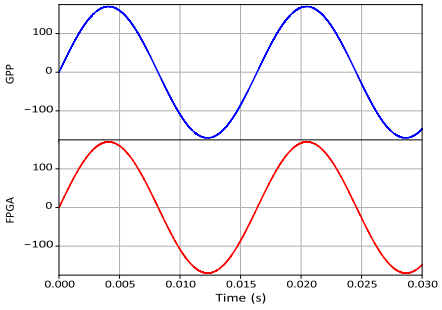

Figure 12. Simulation results: voltage at the load for Case B using a FPGAbased simulation environment and a conventional general purpose processor.

with respect to the quantity of PV generators with a smaller slope than the GPP environment.

Table IV
SIMULATION TIME

PVSIs	GPP (ms)	FPGA (ms)
1	956	130
2	1179	142
4	1475	167
8	2242	219
16	3352	320

V. CONCLUSIONS

A dynamical model useful for the analysis of power processing units that operates in microgrid has been presented. For a proof-of-concept system such model has been implemented in a FPGA-based simulation environment. The simulation results with the proposed FPGA-based systems have been proved to be identical than those obtained with a high level simulation language using a high-order numerical method. Finally, due to its concurrent nature, the FPGA-based system is able to execute the simulation in a much lower time than a General Purpose Process. The simulation time of the proposed system is less dependent on the amount of energy sources considered.

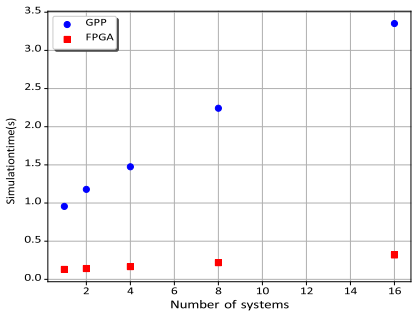

Figure 13. Simulation Circuit Block Diagram

The results of the present paper suggests that FPGA-based systems are suitable environments for the simulation of microgrid systems with several dozens of energy sources and loads of different types.

REFERENCES

[1] R. H. Lasseter and P. Paigi, "Microgrid: A conceptual solution," in *Power Electronics Specialists Conference, 2004. PESC 04. 2004 IEEE 35th Annual*, vol. 6. IEEE, 2004, pp. 4285–4290.

[2] A. S. Meliopoulos, "Challenges in simulation and design of/spl mu/grids," in *Power Engineering Society Winter Meeting, 2002. IEEE*, vol. 1. IEEE, 2002, pp. 309–314.

[3] A. Merabet, K. T. Ahmed, H. Ibrahim, R. Beguenane, and A. M. Ghias, "Energy management and control system for laboratory scale microgrid based wind-pv-battery," *IEEE Transactions on Sustainable Energy*, vol. 8, no. 1, pp. 145–154, 2017.

[4] L. Guan, "Fpga and digital signal processing," in *FPGA-based Digital Convolution for Wireless Applications*. Springer, 2017, pp. 5–23.

[5] H. Peters, R. Sethuraman, A. Beric, P. Meuwissen, S. Balakrishnan, C. A. A. Pinto, W. Kruijtzer, F. Ernst, G. Alkadi, J. Van Meerbergen *et al.*, "Application specific instruction-set processor template for motion estimation in video applications," *IEEE transactions on circuits and systems for video technology*, vol. 15, no. 4, pp. 508–527, 2005.

[6] J. Antikainen, P. Salmela, O. Silven, M. Juntti, J. Takala, and M. Myllyl´ a,¨ "Application-specific instruction set processor implementation of list sphere detector," *EURASIP Journal on Embedded Systems*, vol. 2007, no. 1, p. 054173, 2008.

[7] T. Good and M. Benaissa, "Very small fpga application-specific instruction processor for aes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 7, pp. 1477–1486, 2006.

[8] S. Vaas, M. Reichenbach, and D. Fey, "An application-specific instruction set processor for power quality monitoring," in *Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International*. IEEE, 2016, pp. 181–188.

[9] D. Bora, B. Thomas, S. Nandi, and G. Trivedi, "Application specific processor design implementation to monitor seismic activity," in *Accessibility to Digital World (ICADW), 2016 International Conference on*. IEEE, 2016, pp. 9–14.

[10] "The odespy software package," https://github.com/hplgit/odespy, accessed: 2018-01-14.