# Post-silicon Validation Procedure for a PWL ASIC Microprocessor Architecture

O. Lifschitz, J. A. Rodríguez, P. Julián, *Senior Member, IEEE* and O. Agamennoni

***Abstract*— In this paper, we present the environment set for validation and testing a particular ASIC that implements a piecewise linear (PWL) architecture. Description for a package debug propose is included. Methodologies for power consumption and maximum operation frequency estimation, based on laboratory measurements, are described.**

***Keywords*— Piecewise linear, Validation, ASIC.**

## I. Introducción

LAS FUNCIONES lineales a tramos son abstracciones matemáticas muy usadas en teoría de circuitos, gráficos por computadora y en sistemas de identificación y control [1]. La evaluación de este tipo de funciones se ha abordado de diferentes maneras por diversos algoritmos tales como: *simplicial paths* [2], la arquitectura de un comparador [3], y más recientemente redes neuronales [4]. Un microprocesador dedicado, denominado PWLR6, fue implementado utilizando un flujo de diseño EDA (*Electronic Design Automation*) basado en las herramientas de *Synopsys* utilizando la biblioteca estándar de celdas AMI 05 OSU [5]. El PWLR6 fue diseñado para ejecutar el cálculo de una función PWL de $R^6$ con un alto grado de la flexibilidad [6]. Una unidad de control microprogramada permite el arreglo de configuraciones diferentes utilizando el ISA (*Instruction Set Architecture*) PWLR6. Saltos absolutos y relativos, ALU (*Arithmetic Logic Unit*), escritura y lectura de memoria e instrucciones de acceso a registros, proporcionan un ambiente rico para explotar las funcionalidades PWLR6.

En este artículo se presenta un conjunto de funciones de depuración y metodologías para proceder a una validación y pruebas para *post-silicon*, proporcionando el medio ambiente adecuado para hacer frente a la complejidad del PWLR6.

En la etapa inicial de diseño del ASIC se introdujeron diseños específicos (DFT: *Design For Testability*) de testeo y observabilidad. Estos diseños de DFT se ejecutaron con el flujo de verificación de todo el conjunto del chip PWLR6. Con el fin de crear un entorno de pruebas eficientes se logró una sincronización entre los resultados del chip y los datos de simulación, con una resolución de un período de reloj.

Para garantizar una amplia cobertura y así poder asegurar con mayor certeza una correcta funcionalidad del chip se utilizó un procedimiento aleatorio. Además, un conjunto conocido de pruebas de pre-silicio estaban dispuestas para evaluar sistemáticamente cada uno de los bloques del PWLR6 por separado en caso de un error o fallo ocurriera.

Se desarrollaron métodos de medición de potencia y de frecuencia máxima de funcionamiento. Los métodos y los resultados se incluyen en el trabajo.

## II. Breve Descripción de la Arquitectura PWLR6

En esta sección se presenta una breve descripción de una arquitectura digital que implementa una PWL de $R^6$. La idea es explicar el nivel de complejidad de la evaluación del ASIC. La arquitectura propuesta se basa en un microprocesador [7] cuyo esquema incluye una ALU, un archivo de registro y una unidad de control [8] .

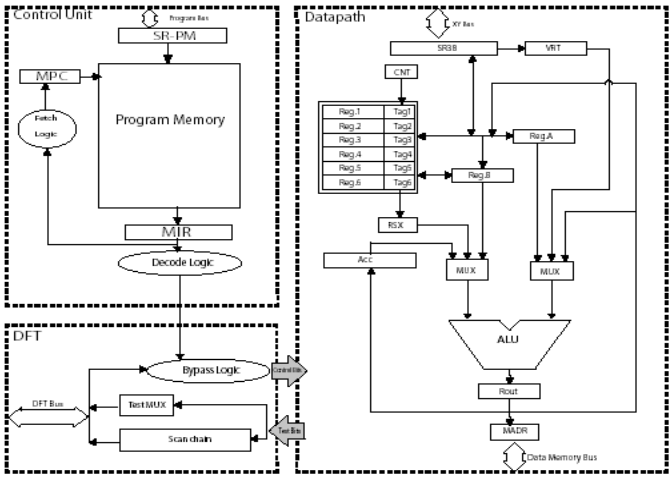


Figura 1. Diagrama en bloques del PWLR6 chip.

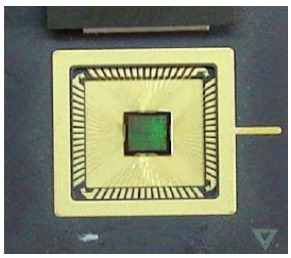

Figura 2. Foto del PWLR6 chip.

## III. Entorno De Testeo

### A. Entorno de Hardware

En esta sección se presenta una breve descripción del entorno de hardware (HW). El HW consiste en una configuración maestro-esclavo donde el PWLR6 está en el papel de esclavo y el amo se desarrolla en un *test board* de Xilinx con una FPGA Virtex5. El PWLR6 utiliza un dispositivo de memoria DDR2. El controlador y adaptador de

protocolos (MTH: *Memory Transfer Hub*) se encuentran realizados en la Virtex5. El MTH es responsable de adaptar el protocolo del controlador de memoria con el protocolo de memoria del PWLR6 chip. Además, permite la funcionalidad de estos sistemas en diferentes frecuencias. Las frecuencias de los sistemas son: Maestro@100MHz, controlador de memoria @160MHz y PWLR6 @ [6,25 MHz - 50 MHz]. El maestro se interconecta con una PC, a través de RS232, usando el Matlab como interfase. El chip PWLR6 se coloco en un PCB (*Printed Circuit Board*) de dos capas. Este PCB incluye algunos conectores de pines para el analizador lógico, osciloscopio y para facilitar mediciones de corriente y tensión durante la validación.

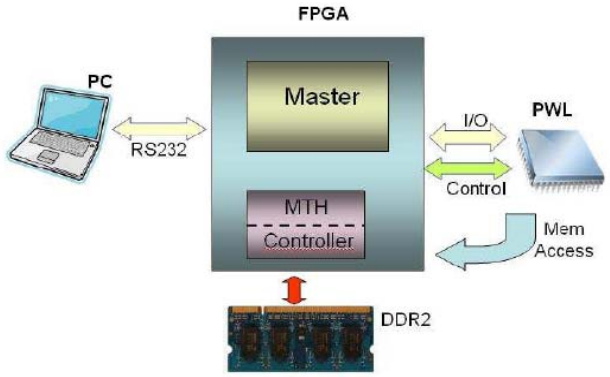


Figura 3. Esquema del entorno de HW.

### *B. Entorno de Software*

Existen varios niveles de software en el sistema, cada uno de ellos se ejecutan en una parte diferente del entorno de hardware:

**PWLR6 *Assembler***. - Este código corre en el PWLR6 y es para: recibir los datos de entrada de N-dimensiones (Xi) que son envidados por el maestro, calcula o evalúa la función PWL y envía los resultados de regreso al maestro. Además de eso, otros códigos se desarrollaron para propósitos de validación permitiendo activar diferentes partes dentro del chip PWLR6 o en el caso del consumo para activar la ALU con carga máxima.

El PWLR6 admite 256 palabras de código de programa de 20 bits que son alojadas en una ROM de instrucciones. Los códigos de assembler son programados en esta ROM por el maestro cuando el PWLR6 se encuentra en modo de programación. Estos códigos se encuentran almacenados en la FPGA. Los modos de programación y ejecución son fijados por el maestro.

**Maestro SW** - En el maestro se encuentran cuatro máquinas de estado y un PicoBlaze. El PicoBlaze es microcontrolador de 8 bits definido en HDL (*Hardware Definition Languaje*) y provisto por Xilinx como un *Softcore*. Sus funciones son: operar la RS232 y activar las máquinas de estado en función de los comandos recibidos por la RS232 desde el Matlab.

Las máquinas de estado son:

**Carga ASM**. – Programa el código assembler del PWLR6 dentro de la ROM. Esto se realizó mediante dos señales de asincrónicas: datos y reloj. Ambas señales son generadas por el maestro.

**Chip_clk_gen**. – Se encarga de generar el reloj para el PWLR6. Esta máquina se encarga de detener y activar el reloj (*stop_clock* y *Do_1 _clk*). (Véase la parte DFT)

**Exe_calc_func**. – Activa el PWLR6. Esta máquina pone el PWLR6 trabajar mediante el envío de las entradas en serie de las variables Xi ($Xi \varepsilon R^6$). Luego espera a que el PWLR6 envíe el resultado de regreso. Esta maquina enviará los resultados finales a la interfaz del Matlab.

**DFT_block**. - Genera la activación y control de los bloques de validación y observabilidad: Scan, Bypass-Scan y Bypass. (Véase la parte DFT). Esta maquina es controlada por el PicoBlaze dependiendo de los comandos enviados desde interfaz humana. Esta máquina, tambien, realiza el formateo de datos antes de llegar a la interfase de la PC.

**PC SW**. - Es una interfase de Matlab que se ocupa del envío de comandos al PicoBlaze y el formateo de los datos para la interfase humana. Además, crea los casos al azar y compara los resultados del chip y de una función PWL que se representó y evaluó en Matlab.

## IV. Diseño De Testabilidad (DFT)

La idea principal de las estructuras de testeo es la de colaborar con el diseñador en las tareas de validación y verificación. Estas estructuras se diseñaron durante el desarrollo del PWLR6 y su funcionalidad fue validada en simulaciones durante el pre-silicio.

El PWLR6 contiene 6 estructuras:

**Stop_clk**. –Algunos eventos, ya sean internos o externos, pueden hacer que el reloj del PWLR6 se detenga parando la ejecución del PWLR6 instantáneamente. Estos eventos, internos, pueden ser sincronizados con el simulador para una eventual correlación temporal entre el simulador y el PWLR6 chip. Esta sincronización permite una fácil verificación "*clock by clock*" a nivel de registro. La sincronización es garantizada utilizando un contador de periodos de reloj en el simulador y en el chip. La opción para detener el reloj externa esta contemplada en una señal de la interfase entre PC y el Chip pero la correlación temporal no esta garantizada.

**Do_1_clk**. –Genera un pulso aislado de reloj en el PWLR6 chip. Esto nos permite correr el PWLR6 chip "*clock by clock*" o paso a paso como se conoce en la literatura. La activación de esta señal puede ser interna o externa. La aplicación interna es cuando es utilizada por otro DFT. La opción externa esta contemplada en la interfase entre PC y el PWLR6 chip.

Los dos DFT que siguen están relacionados con la observabilidad del chip. La idea es sacar del interior del chip las señales o estados hacia el exterior de manera de contar desde el entorno de trabajo con señales para verificación. Estas señales son elegidas del *data path* y del *control path*.

**MUX**. – Este es un multlipexor interior al PWLR6 que es activado desde el exterior. Este multiplexor nos permite sacar hacia el exterior distintas señales internas. La salida de esas señales es en paralelo. La relación entre el número de bits y el número de registros "observables" es una relación de compromiso y depende de las consideraciones de diseño. Este

DFT tiene una activación "on the fly", es decir que su activación se puede hacer con el chip en funcionamiento y las señales elegidas que son sacadas al exterior irán, eventualmente, cambiando en cada ciclo de reloj. Cabe mencionar que este es un DFT de costo elevado debido a la cantidad de I/O pines que utiliza. Por otro lado, es el DFT más sencillo de diseñar. Este DFT permite la creación una "signature" de las señales que fácilmente son llevadas a un analizador lógico permitiendo la comparación con el simulador.

**Scan**. – Este DFT saca hacia el exterior un número diferente de bits que provienen de diferentes bloques del PWLR6. Este DFT es serial en un formato de configuración "*daisy chain*".
En oposición al MUX, el Scan tiene que ser activado cuando el PWLR6 se encuentra en un evento de "stop_clock" para garantizar que todos los bits encadenados pertenezcan al mismo periodo de reloj. En cuanto al numero de I/O pins, este DFT es barato ya que utiliza un pin de data y otro de reloj. El reloj de este DFT es ingresado externamente y es asincrónico. En este caso, el maestro genera este reloj.

**Bypass-Scan**. – Este DFT es similar en su funcionamiento al Scan pero solo actúa en el "*Control Register*". Debido a su complejidad e importancia el Registro de control tiene un DFT dedicado. El registro de control tiene las instrucciones de *assembler* que son de 20 bits provenientes de la ROM.

**Bypass**. – Este DFT permite la escritura del registro de control desde el exterior, dejando de lado la conexión con la ROM. La sincronización de este DFT es extremadamente necesaria para asegurar el funcionamiento correcto. Los DFT Bypass-scan y Bypass fueron introducidos en el chip para una eventual falla de la ROM ya sea por un mal diseño ó una falla de manufactura. La idea era facilitar el procedimiento "*onion peeling*", es decir, si la ROM fallaba se podía seguir verificando el chip para evaluar otros posibles "*bugs*" antes de un nuevo diseño.

## V. Post-Silicon

### A. Verificación Funcional De Bloques

La verificación funcional de bloques requiere que cada bloque sea testeado independientemente. Asegurar el correcto funcionamiento individual permite crear un proceso sólido de manera de aislar los problemas cuando aparecen. Esta verificación se realizó utilizando un compilador, desarrollado explícitamente, para crear las distintas situaciones necesarias para la activación de cada bloque. Los seis bloques principales incluidos en esta verificación son:

**DFT**. – Muchas de estas estructuras fueron testeadas durante el pre-silicio usando un analizador lógico. Debido a que el PWLR6 era el primer silicio que contaba con este paquete de validación, se tuvo que dedicar una ventana de testeo para estos DFT. Estos bloques no son parte de la evaluación del PWLR6. Idealmente, estos bloques formarían parte de una infraestructura de testeo que se utilizaría en todos los chips a futuro.

**Maquinas de estado en la FPGA**. – Estas maquinas de estado fueron comprobadas utilizando un analizador lógico. Los protocolos y los tiempos fueron validados utilizando los resultados del analizador y las simulaciones correspondientes.

**Memoria ROM**. – Esta fue validada utilizando los DFT para verificar la correcta escritura y lectura de la ROM. Este bloque era crítico para el funcionamiento del PWLR6 y no se había podido simular en las etapas de diseño.

**Protocolo de las Xi**. – Esta validación incluye todos los protocolos asincrónicos entre el maestro y esclavo. Los DFT y el analizador lógico sirvieron para garantizar la correcta comunicación entre de las Xi. Donde las Xi son las variables de R^6.

**Clasificación (*sorting*)**. – Debido a la fuerte dependencia con el valor n-dimensional se testearon diferentes valores de variables de entrada y utilizando los DFT de observabilidad se verificó el valor correcto de este bloque comparando con los resultados del simulador.

**Acceso externo a memoria**. – Debido al I/O bidirecional se puso especial atención en esta parte de la validación. Un *assembler* dedicado se utilizo para leer y escribir a memoria diferentes direcciones. Los datos fueron verificados usando el DFT y un analizador lógico. Además, se puso énfasis en el hecho que el PWLR6, MTH y la memoria operaban a distintas frecuencias.

**ALU**. – Además de los cálculos de PWL explícitamente, se testearon diferentes operaciones matemáticas y los resultados fueron chequeados utilizando los DFT y el analizador lógico.

### B. Verificación Funcional

En esta parte de la validación se utilizaron dos conjuntos de datos:
A) Casos de simulación
B) Casos *Aleatorios.*

Un "caso" es cuando se entregan 6 variables al PWLR6 y el PWLR6 devuelve un resultado que es la evaluación de $R^6$ a $R^1$ PWL.

Los casos de simulación fueron lo que se heredaron de las simulaciones pre-silicio. Usando en la evaluación de los DFT y correlaciones con el simulador.

Los casos aleatorios fueron generados aleatoriamente con Matlab usando como función a aproximar una función lineal de manera de evitar errores en la aproximación PWL.

La principal diferencia entre estos dos conjuntos de datos es la velocidad. Es decir, en la simulación los datos a testear son reducidos y la memoria utilizada es mínima sin embargo todos los DFT son activados y los valores obtenidos son chequeados comparando con el simulador. Los valores de memoria son establecidos con el simulador ya que son reducidos en cantidad. Por el otro lado, los casos aleatorios son verificados con Matlab utilizando un PWL creado en el entorno Matlab.

En este caso se utilizaron los 16MB de memoria. La comparación correcta de estos casos fueron la base para el testeo de máxima frecuencia de operación de la sección siguiente.

La Fig. 4 muestra algunos de los casos aleatorios. El error entre el PWLR6 chip y el Matlab es cero como se esperaba.

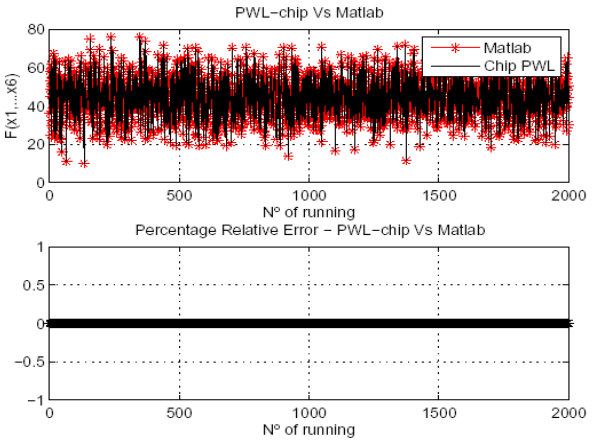


Figura 4. Resultado de algunos casos aleatorios.

## VI. Medidas De Frecuencia

La idea, en esta parte de la validación, era verificar la máxima frecuencia alcanzable por el PWLR6 chip. El plano de potencia del *I/O* y del *core,* en el PWLR6, están unidos. Esta unión limita el máximo voltaje que se puede aplicar al PWLR6 para no activar los *clamping* de los *I/O* de la FGPA.

A los efectos de superar esta limitación se procedió a definir un experimento para extrapolar la máxima frecuencia sin cambiar el voltaje de alimentación del PWLR6 por fuera del rango aceptable de los *I/O pins clamping*. Este ensayo ejercita el camino crítico obtenido de las simulaciones. Este camino esta relacionado con la ejecución de la ALU.

El ensayo consistía en reducir el voltaje de alimentación del PWLR6, manteniendo una frecuencia de operación, hasta obtener un error funcional, entendiéndose por error funcional a aquel resultado que difiere del correcto, pero que el chip continúa operativo, es decir los protocolos entre el maestro y esclavo continúan funcionando.

El éxito ó el fracaso de este ensayo era nuestro criterio para el grafico de frecuencia vs. PWLR6 *core voltaje*.

La frecuencia del PWLR6 se cambio usando el bloque DCM (*Digital Clock Manager*) propietario de la FPGA.
La Fig. 5 ejemplifica el esquema de cambio de frecuencias.

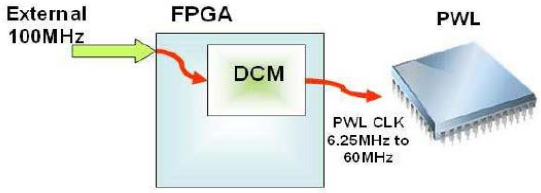


Figura 5. Cambios de Frecuencias en el PWLR6

La Fig. 6 muestra la frecuencia máxima de operación para cada voltaje de core ilustrando dos niveles distintos de cobertura.

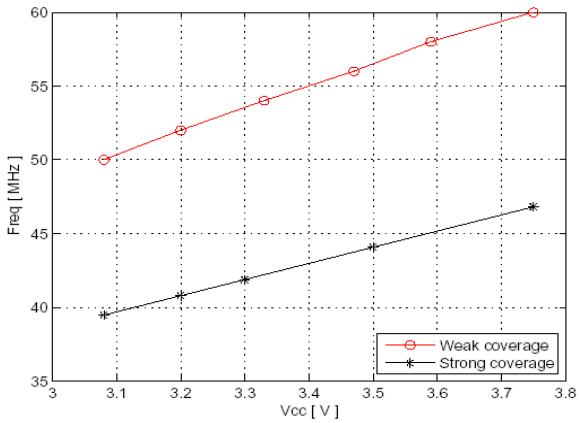


Figura 6. Frecuencia Vs. Voltaje de PWLR6 core.

La diferencia en cobertura (*coverage*) esta relacionada con los dos casos mencionados anteriormente: Casos de simulación y Casos aleatorios. A pesar de que ambos conjuntos de testeo apuntan a la ejecución de la ALU, el caso aleatorios tiene mas de 40K casos mientras que el otro son solamente 9 casos. Es por eso la diferencia que se obtuvo en el grafico de frecuencia Vs. voltaje.

## VII. Medidas De Potencia

El consumo de potencias tiene 3 componentes destacados: Consumo estático, árbol de reloj y consumo dinámico. El procedimiento para obtener estos componentes esta resumido en la siguiente tabla:

| Items de Consumo | Reloj | Reset | Ejecutando |
|---|---|---|---|
| Estático | off | on | off |
| Árbol de reloj | on | on | off |
| Dinámico | on | off | on |

La condición ejecutando significa que el PWLR6 se encuentra procesando las instrucciones de *assembler*.

El consumo de potencia se midió utilizando una resistencia en serie con la alimentación del PWLR6. El valor de esta resistencia se calculó de manera de mantener el voltaje en esta resistencia dentro del rango [600mV, 1500mV]. El valor utilizado fue de 100 Ohms.

Las mediciones se hicieron utilizando 2 dispositivos: Un osciloscopio digital (Agilent DSO3062A) y un multimetro Hewlett Packard (HP34401A). La idea era correlacionar las mediciones con ambos elementos de medición.

### *A. Figura De Consumos En Tiempo*

Para tener una idea del consumo del PWLR6 durante la ejecución, se procedió a tomar una figura en el osciloscopio.

Fig. 7 muestra el consumo del PWLR6 durante la ejecución de un caso.

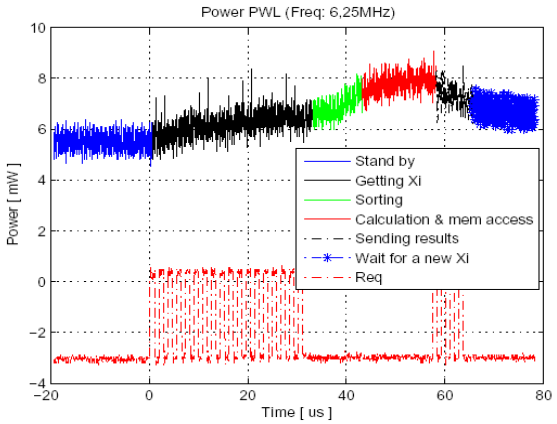


Figura 7. Consumo del PWLR6 durante ejecución.

En la Fig. 7, la señal "Req" no esta en la escala correcta, se muestra como referencia del protocolo asincrónico.

La grafica del consumo durante la ejecución fue la base para la creación del un "*power virus*". El peor caso de consumo fue durante el "*calculation and memory access*". Basado en este hecho, se obtuvo un *assembler* que activaba de manera constante a la ALU y los *I/O* de salida. A este *assembler* se lo denomino "*Power virus*" ya que es la máxima potencia que el PWLR6 puede disipar. Este assembler consistía en poner a la ALU en un ciclo infinito haciendo cálculos y saliendo a memoria. Este Power Virus se usó para calcular el consumo en diferentes puntos de frecuencia de operación para un dado voltaje de alimentación. Además, este power virus da una idea de la disipación en el caso que se necesite diseñar un disipador adecuado para el PWLR6 (*power envelope*).

### *B. Medidas De Potencia*

La potencia estática estaba alrededor de 160nW. La potencia estática corresponde al caso en el cual el reloj esta apagado y la señal de reset esta activada, como se menciona anteriormente. Usando el power virus se midió el consumo para distintos puntos de frecuencia y un voltaje fijo de 3.3V. La siguiente tabla a continuación muestra los valores obtenidos:

| PWLR6 | Consumo [mW] Vcc@3,3V | | |
|---|---|---|---|
| Frec. [MHz] | Árbol de Reloj | P. Virus | P. Virus + I/O |
| 25 | 22,7 | 34,91 | 49,68 |
| 12,5 | 11,42 | 17,6 | 24,87 |
| 6,25 | 5,69 | 8,75 | 12,4 |

Se puede comprobar que la potencia varía en forma lineal con la frecuencia y cuadrática con la tensión de alimentación.

## VIII. Conclusiones.

En este trabajo presentamos metodologías de validación y testeo de post-silicio. Tres importantes conclusiones se desprenden:

I) El trabajo de preparación del entorno, como: DFT, compilador de assembler y los diferentes códigos de assembler para los distintos bloques.

II)Definición de las metodologías para la medición de potencias (*Power virus*), la estimación de las frecuencias de operación y la capacidad de validación de los distintos bloques.

III) Definición de la idea de cobertura (*coverage*) en este tipo de chips, en especial, cuando la validación pre-silicio es muy lenta ó difícil de obtener por la complejidad de los casos a verificar.

## IX. Agradecimientos

Queremos agradecer la colaboración del Ingeniero Ariel Arelovich por la creación del compilador del PWLR6 que fue de gran ayuda durante la verificación.

## Referencias

[1] L. Castro, J. Figueroa, O. Agamennoni, "BIBO stability for NOE model structure using HL CPWL functions", *in Proc. of Modelling, Identification, and Control*, 2005.

[2] M. Chien and E.Kuh, "Solving nonlinear resistive networks using piecewise-linear analysis and simplicial subdivision", *IEEE Transactions on Circuits and Systems*, Vol.24, pp. 305-317, 1977

[3] P. Mandolesi, P. Julián, and A. Andreou, "A scalable and programmable simplicial CNN digital pixel processor architecture", *IEEE Transactions on Circuits and Systems-I: Regular papers*, Vol.51, pp. 988-996, 2004.

[4] Xusheng Sun, Shuning Wang, "A Special Kind of Neural Networks: Continuous Piecewise Linear Functions", *LNCS Advances y Neural network*, pp: 375-379. 2005.

[5] J. E. Stine, J. Grad, I. Castellanos, J. Blank, V. Dave, M. Prakash, N. Iliev, and N. Jachimiec, "A Framework for High-Level Synthesis of Systemon- Chip Designs", *International Conference on Microelectronic Systems Education, IEEE Computer Society*, pp. 11-12, 2005

[6] V. M. Jiménez-Fernandez, J. A. Rodríguez, P. M. Julián, O. Agamennoni, O. Lifschitz, "VLSI Microprocessor Architecture for a Simplicial PWL Function Evaluation Core", *in Proc. Arg. School of Micro Nanoelectronics*, pp. 50-60, 2008.

[7] Intel Co., *Microprocessor and Peripheral Handbook, Volume 1- Microprocessors*, Intel, 1987.

[8] J. Agustín Rodríguez, Omar D. Lifschitz, Víctor M. Jiménez-Fernandez, Pedro Julián, Osvaldo E. Agamennoni "Application-Specific Processor for Piecewise Linear Functions Computation", *IEEE Transactions on Circuits and Systems*, Vol: 58 Pag. 971-981, 2011.

**Omar D. Lifschitz**. Se recibió de Ingeniero Electrónico en el Instituto Tecnológico de Buenos Aires (ITBA) en el 1998. Actualmente se encuentra estudiando su doctorado en Control de Sistemas en la Universidad Nacional del Sur (UNS). Bahía Blanca Argentina. Desde el 1999 hasta el 2007 se desempeño como ingeniero de HW en el grupo de Validación Analógica en Intel Israel. Se especializo en Integridad de Señales.

**J. Agustín Rodríguez.** Se recibió de Ingeniero en Sistemas en la Universidad Nacional del Sur (UNS) Bahía Blanca Argentina en el 2007. Actualmente se encuentra estudiando el doctorado en Ciencias de la computación. Sus tópicos de investigación son: Arquitecturas de alto desempeño, Síntesis de alto nivel de arquitecturas multi-core, análisis de tiempos estáticos y problemas de ruteo.

**Pedro Julián** (S'93-M'99-SM'05) Recibió el Titulo de Ingeniero Electrónico en 1994 y el Doctorado en "Control de Sistemas" en 1999, ambos en Universidad Nacional del Sur (UNS). Fue becado en la Universidad de California Berkeley (2000 a 2002), en la Universidad Johns Hopkins (2002 a 2003) y obtuvo una beca Fulbright (2009) en la Universidad Johns Hopkins. Es Profesor Asociado en el Departamento de Ingeniería Eléctrica y de Computadoras (DIEC) en UNS e Investigador Independiente en el Consejo Nacional de Investigación de Argentina (CONICET). Es un miembro fundador del Consorcio de América Latina de Servicios Integrados (LACIS) y la Escuela Argentina de Microelectrónica (EAMTA). Se desempeñó como Vice Presidente de la Región 9 en la Sociedad de Circuitos y Sistemas (CASS) entre 2004 - 2007. Es investigador responsable del Proyecto Tecnópolis del Sur.

**Osvaldo E. Agamennoni** Se recibió de Ingeniero Electricista en la Universidad Nacional del Sur (UNS) (Bahía Blanca) en 1979 y obtuvo el grado de Doctor en Control de Sistemas en la misma Universidad en el año 1991. Desde el año 1980 al 1983 trabajó en el diseño automatizado de circuitos en el Departamento de Ingeniería Eléctrica de la UNS. Desde el año 1983 al 1991 realizó diversas tareas de investigación en el área de control de procesos en PLAPIQUI (Planta Piloto de Ingeniería Química - UNS CONICET). Desde el año 1992 al 1994 obtuvo una posición postdoctoral en la University of Sydney realizando trabajos de investigación aplicada en el área de inteligencia artificial para la impresa ICI (Imperial Chemical Industries). Desde el año 1986 pertenece a la CIC (Comisión de Investigaciones Científicas de la Provincia de Buenos Aires) teniendo actualmente el cargo de Investigador Principal de dicho organismo. Actualmente es Profesor Titular en el Departamento de Ingeniería Eléctrica y de Computadoras de la UNS, teniendo a su cargo cursos de fundamento de control, modelado e identificación de sistemas dinámicos. Sus áreas de interés incluyen modelado, identificación y control de sistemas no lineales y sistemas cognitivos.