Diseño de un Sistema Digital de Medición de Impedancia a Tiempo Real Utilizando la Metodología de Codiseño Hardware/Software

Guillermo G. Gastaldi
guillegastaldi@hotmail.com

José A. Rapallini
josrap@ing.unlp.edu.ar

Antonio A. Quijano
quijano@ing.unlp.edu.ar

Codiseño Hardware/Software – Centro de Técnicas Analógicas y Digitales (CeTAD)
Departamento de Electrotecnia, Facultad de Ingeniería.
Universidad Nacional de La Plata.

ABSTRACT

In this paper we describe the design of a real time impedance measurement system, using the Hardware / Software Codesign Methodology. After an introduction on the application of impedance measurement for electric system protection, we include the principles of HW / SW Codesign. Next we consider the descrption and functional simulation of our system, which are the design stages covered up to the present time, and at last we describe a prototype implemented with a data acquisition subsystem connected to a PC bus. Some experience was gained through the prototype development and the functional simulation, allowing us to carry on the codesign for an embedded system to be realized in a near future with FPGA or an ASIC.

RESUMEN

En este trabajo se presenta un sistema de medición de impedancia diseñado utilizando la metodología del Codiseño Hardware/Software. En primer lugar se introduce dicha metodología. Luego se presentan las etapas de la metodología desarrolladas hasta el momento (descripción y simulación funcional), referidas al diseño del sistema. Por último se describe la implementación de un prototipo utilizando una placa de adquisición conectada al bus del sistema de computo personal (PC). La experiencia del desarrollo del prototipo junto a la simulación funcional, da los fundamentos para continuar con los pasos del codiseño en la implementación del sistema embebido, que sera realizado en la siguiente etapa del proyecto con lógica programada (FPGA) o en un circuito integrado de aplicación específica (ASIC).

Diseño de un Sistema Digital de Medición de Impedancia a Tiempo Real Utilizando la Metodología de Codiseño Hardware/Software

Guillermo G. Gastaldi
guillegastaldi@hotmail.com

José A. Rapallini
josrap@ing.unlp.edu.ar

Antonio A. Quijano
quijano@ing.unlp.edu.ar

Codiseño Hardware/Software – Centro de Técnicas Analógicas y Digitales (CeTAD)
Departamento de Electrotecnia, Facultad de Ingeniería.
Universidad Nacional de La Plata.

ABSTRACT

In this paper we describe the design of a real time impedance measurement system, using the Hardware / Software Codesign Methodology. After an introduction on the application of impedance measurement for electric system protection, we include the principles of HW / SW Codesign. Next we consider the descrption and functional simulation of our system, which are the design stages covered up to the present time, and at last we describe a prototype implemented with a data acquisition subsystem connected to a PC bus. Some experience was gained through the prototype development and the functional simulation, allowing us to carry on the codesign for an embedded system to be realized in a near future with FPGA or an ASIC.

RESUMEN

En este trabajo se presenta un sistema de medición de impedancia diseñado utilizando la metodología del Codiseño Hardware/Software. En primer lugar se introduce dicha metodología. Luego se presentan las etapas de la metodología desarrolladas hasta el momento (descripción y simulación funcional), referidas al diseño del sistema. Por último se describe la implementación de un prototipo utilizando una placa de adquisición conectada al bus del sistema de computo personal (PC). La experiencia del desarrollo del prototipo junto a la simulación funcional, da los fundamentos para continuar con los pasos del codiseño en la implementación del sistema embebido, que sera realizado en la siguiente etapa del proyecto con lógica programada (FPGA) o en un circuito integrado de aplicación específica (ASIC).

Palabras Claves: Codiseño Hardware/Software, sistemas embebidos, medición de impedancia, filtrado digital, sistemas de tiempo real.

1. INTRODUCCIÓN

Una aplicación típica de los sistemas de medición de impedancia en tiempo real, es en el campo de las protecciones de sistemas eléctricos [1]. Dichas protecciones tienen como objeto detectar las fallas de cortocircuitos y separar a través de los interruptores correspondientes la parte del sistema fallado. En particular, los llamados relevadores de impedancia o de distancia digitales se basan en esta clase de sistemas.

El principio de funcionamiento de los relevadores de impedancia o distancia es el siguiente: calculan una impedancia aparente, basado en el cociente de los fasores de tensión y corriente a la frecuencia de red. En función del valor calculado de impedancia, el sistema determina si hubo falla o no y envía una señal lógica para comandar el interruptor.

Antes de realizar el cálculo fasorial, es necesario filtrar las señales de entrada debido al ruido que se presenta en las redes eléctricas, en particular al momento de una falla. Entre las más importantes causas de ruido se encuentran las cargas alineales conectadas a la red que generan componentes armónicas de la frecuencia fundamental y la característica inductiva de las líneas que genera componentes exponenciales cuando se produce un cortocircuito en la misma [5].

En la actualidad, es muy común el uso de filtros de Fourier, Coseno o Seno, para realizar el filtrado en sistemas de medición de impedancia aplicados a protecciones eléctricas [1], [5]. El motivo principal es la posibilidad de eliminar los armónicos de la fundamental eligiendo sus parámetros de forma adecuada. En este trabajo se utiliza el filtro Coseno.

El proceso de diseño llevado a cabo se basó en la filosofía de Codiseño Hardware/Software, en la cual el sistema es concebido como un todo, a diferencia del proceso del diseño digital tradicional, donde generalmente se desarrolla el hardware y posteriormente se adapta el software a ese diseño, lo cual en muchas oportunidades resulta ineficiente. En esta primera etapa del trabajo se desarrollaron las fases de descripción y simulación funcional del sistema, debido a la imposibilidad de contar con herramientas de particionamiento y síntesis. Como herramienta de diseño se ha utilizado el software comercial Simulink® y MATLAB® de MathWorks Inc y los paquetes adicionales Real Time Workshop® y Real Time Windows Target® de la misma empresa.

Una vez cumplida la simulación funcional, se implementó un prototipo del sistema en desarrollo basándose en una computadora PC compatible con un procesador Intel y una placa de adquisición. Para ello se desarrolló un programa escrito en C apoyándose en la descripción realizada mediante los diagramas en bloques de Simulink®.

En una próxima etapa del diseño del prototipo, se pasarán con la ayuda de otras herramientas de CAD (Diseño Asistido por Computadora) los códigos generados hasta el momento, a componentes de lógica programada o bien ASIC.

A continuación se define la metodología de Codiseño Hardware/Software, para luego comenzar con la descripción del sistema y las fases desarrolladas.

2. METODOLOGÍA DEL CODISEÑO HARDWARE/SOFTWARE

En la actualidad el diseño de sistemas digitales complejos, por ejemplo los sistemas embebidos, se basa en arquitecturas con uno o varios microprocesadores, microcontroladores o DSPs dedicados a ejecutar programas (Software) y lógica "hecha a medida" (custom) implementada con FPGAs, PLAs o ASICs (Hardware) que desempeñan tareas específicas.

El diseño entonces es una ardua tarea, pues se deben combinar elementos heterogéneos para satisfacer una funcionalidad que generalmente es llevada a cabo en tiempo real. También dicho diseño se ve restringido debido a variables como costo, potencia consumida, peso, etc.

La complejidad de dichos sistemas generó la necesidad de contar con una metodología ordenada que fuese capaz de guiar al proceso de diseño partiendo de una idea de sistema y llegando a la implementación del mismo satisfaciendo las especificaciones en el menor tiempo de diseño posible. Está metodología es llamada Codiseño Hardware/Software (HW/SW).

Una definición formal de Codiseño HW / SW se da en [2] "... es el proceso de diseño de un sistema que combina las perspectivas hardware y software desde los estados primarios, para aprovechar la flexibilidad del diseño y la localización eficiente de las funciones."

En [3] se propone la metodología general del Codiseño HW/SW, que se muestra a continuación:

Las fases del proceso completo son:

– Definición del problema a solucionar: el usuario o cliente, en el sentido más amplio, plantea la necesidad de resolver un problema particular. Esta etapa se desarrolla a la par de la extracción de especificaciones, la diferencia entre ambas es el actor: mientras que el cliente define el problema, el diseñador conceptualiza el sistema y extrae las especificaciones.

Fig.1. Metodología de Codiseño Hardware/Software.

– Extracción de especificaciones: conocidas las necesidades del cliente, se extrae un conjunto de especificaciones que determinan la operación funcional, entradas, salidas e interfase del sistema con el medio de operación. Además se complementa con restricciones que limitan las posibilidades de diseño. Este tipo de restricciones corresponde a variables físicas y de operación real del sistema, tales como: velocidad, consumo de potencia, tiempos de diseño, costo, etc. El resultado final de esta etapa es un documento de requisitos donde se especifica completamente al sistema.
– Descripción formal del sistema: luego de conceptualizar el sistema en la fase anterior, se procede a elaborar un modelo formal del mismo, empleando técnicas de descripción o entrada de diseño sobre las herramientas disponibles, que describa la funcionalidad correcta del sistema propuesto. En lo posible dicha descripción deberá ser independiente de cualquier implementación particular.
– Simulación funcional: con herramientas que soporten los modelos y descripciones de la etapa anterior, se

realiza una verificación funcional, hasta obtener un modelo 100% acorde con los requerimientos.

- Partición HW/SW: con el modelo en mano se procede a la selección de una arquitectura y a la asignación óptima de tareas hacia cada uno de los recursos. Es la fase de diseño más problemática. En esta fase se seleccionan tecnologías y elementos microprocesadores a utilizar.
- Cosíntesis: consistirá en adaptar y transformar la descripción o modelo en las tecnologías seleccionadas. Para mantener un control global del sistema se deberá tener en cuenta la implementación de otros módulos con los que cada módulo interactúa en el sistema, además de generar los elementos necesarios para la interfaz entre módulos.
- Cosimulación: se valida la transformación realizada, comprobando que se ajuste a las especificaciones y restricciones iniciales.
- Implementación: adaptación de las descripciones en dispositivos configurables y/o fabricación de los circuitos integrados necesarios. Para la parte software, consiste en la programación de las tareas sobre los recursos de memoria de los elementos microprocesadores de la arquitectura.
- Verificación: corresponde a una integración real de los módulos hardware y de los módulos software para validar su operación y generar un prototipo.

Existen en la actualidad varios grupos de investigación abocados al área del codiseño HW/SW, que han desarrollado herramientas que soportan diferentes aspectos del problema. También varios entornos de software comerciales como por ejemplo Simulink® de MathWorks Inc. permiten el desarrollo de algunas o todas las etapas involucradas en la metodología de codiseño.

Simulink® es un paquete de software, complemento de MATLAB® que permite modelar, simular y analizar sistemas dinámicos, esto es sistemas cuyas salidas y estados internos cambien con el tiempo. Es un entorno grafico, donde se crea un modelo en bloques del sistema, utilizando librerías de bloques estándar y un editor que permite interconectar los mismos. El modelo representa gráficamente las relaciones matemáticas dependientes del tiempo a través de las entradas, estados y salidas del sistema. Simulink® permite implementar modelos en tiempo continuo, en tiempo discreto, híbrido (discreto y continuo), maquinas de estado finito (con jerarquía y concurrencia) y flowcharts. Los sistemas pueden contener distintas partes que son muestradas o actualizadas a diferentes tasas. Además los modelos poseen jerarquía funcional, es decir que se puede crear subsistemas dentro del sistema. Con el agregado de paquetes adicionales, se puede generar código C para diferentes plataformas o código ADA, a partir del modelo del sistema en Simulink®, y simular a tiempo real el sistema.

3. DISEÑO DEL SISTEMA DE MEDICIÓN DE IMPEDANCIA A TIEMPO REAL

- Documento de Requisitos:

A continuación se presenta el documento de requisitos que resume las especificaciones del sistema:

1. Funcionalidad: El sistema deberá adquirir una señal de corriente y una de tensión, filtrarlas para eliminar posibles componentes de ruido y obtener solo la componente de frecuencia deseada (50 Hz), para luego realizar el cálculo de los fasores de tensión y corriente a dicha frecuencia y con ellos el de impedancia. Es deseable también, que el sistema rechace componentes exponenciales de las señales de entrada las cuales ocurren durante una falla (cortocircuito) en un sistema eléctrico. Esta funcionalidad se realiza a tiempo real.
2. Tiempo de respuesta: debido a que el sistema será usado en protecciones deberá responder en el menor tiempo posible. El tiempo de respuesta no deberá superar los 20ms.

- Descripción Jerárquica del Sistema Digital de Medición de Impedancia.

La forma de describir un sistema en Simulink® es mediante un diagrama en bloques. El mismo consiste de un conjunto de símbolos, llamados bloques, interconectados por líneas. Cada bloque representa un sistema dinámico elemental que produce una salida continua (la relación entre entrada y salida se representa a través de ecuaciones diferenciales continuas pero su solución se realiza en pasos discretos utilizando diferentes métodos de integración numéricos) o en puntos específicos en el tiempo (bloque discreto). Las líneas representan conexiones de salidas de bloques a entradas de otros bloques. El tipo de bloque determina la relación entre su entrada y su salida, estados y el tiempo.

En la descripción del sistema se tuvo en cuenta lo expresado en el documento de requisitos. En particular se observan tres tareas principales a desarrollar: filtrar la señal, calcular los fasores representativos de las señales de entrada y calcular la impedancia.

La primera de estas tareas es llevada a cabo por filtros Coseno (uno para la señal de tensión y uno para la señal de corriente) de 32 muestras por ciclo de la fundamental (50 Hz) y ventana de datos de un ciclo [1]. Los coeficientes que determinan la respuesta al impulso de dicho filtro vienen dados por la ecuación:

$$h[n] = \frac{1}{16} \cdot \cos\left(\frac{2 \cdot \pi \cdot n}{32}\right) \quad \text{con} \quad 0 \le n \le 31 \qquad (1)$$

Las 32 muestras por ciclo determinan la frecuencia de muestreo f_s utilizada por el sistema, teniendo en cuenta que un ciclo de la fundamental es de f_0=50Hz y entonces será f_s=1600 Hz.

La elección de este filtro se basó en el estudio realizado en un trabajo anterior [1] donde se evaluó la

respuesta en frecuencia del filtro Coseno, Seno y de Fourier con diferentes valores de muestras por ciclo y tamaños de ventana, y se concluyó que ventanas de datos 1 o 2 ciclos son apropiadas para la eliminación de las componentes armónicas de frecuencia fundamental, en particular la ventana de 1 ciclo es apropiada respecto al tiempo de respuesta.

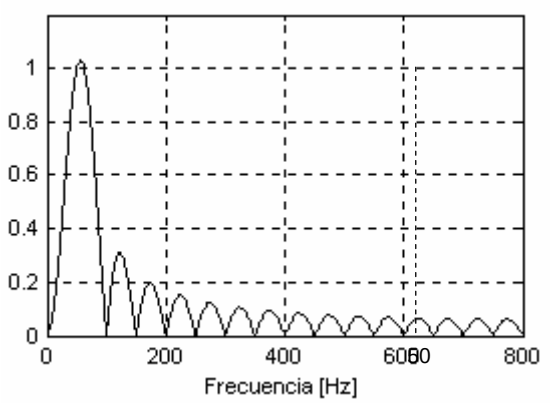

Fig. 2. Módulo de la respuesta en frecuencia de un filtro coseno con 32 muestras por ciclo y ventana de datos de un ciclo.

Se observa que para la frecuencia fundamental (50Hz) el módulo de la respuesta en frecuencia vale uno, mientras que para todos los armónicos de la fundamental la respuesta vale cero, es decir los filtra. La atenuación para el mayor lóbulo lateral, ubicado entre los 100 y 150 Hz, es del orden de los 10dB. Se observa también que el filtro es capaz de rechazar componentes de continua, las cuales están presentes en señales de carácter exponencial.

El filtro se describe mediante una realización en cascada. La misma se obtiene factorizando la transferencia H(z), en el plano z, según sus raíces. H(z) es un polinomio de grado 31 en z^{-1} y cuyos coeficientes coinciden con los de la respuesta al impulso h[n]. De este modo H(z) posee 31 ceros. Los mismos satisfacen condiciones de simetría, que permiten la simplificación de la forma de algunos de los factores, como se muestra a continuación. Se observa de la figura 2 que muchos de estos ceros caen sobre el círculo unitario. En la tabla I se muestran los ceros de H(z) para el filtro coseno con N=32 y ventana de un ciclo.

Operando sobre las expresiones de los ceros de H(z) en la Tabla I se pueden obtener las condiciones de simetría que satisfacen algunos de ellos:

(I): $Z_k = Z^*_{(32-k)}$ con k = 2,3, …, 15

(II): $Z_{(n+8)} = -Z^*_{(8-n)}$ con n = 1,2, ... ,6 y n = 8

Ceros de H(z)			
Z_0	1	Z_{16}	-1
Z_1	0,981	Z_{17}	$e^{j17\pi/16}$
Z_2	$e^{j\pi/8}$	Z_{18}	$e^{j9\pi/8}$
Z_3	$e^{j3\pi/16}$	Z_{19}	$e^{j19\pi/16}$
Z_4	$e^{j\pi/4}$	Z_{20}	$e^{j5\pi/4}$
Z_5	$e^{j5\pi/16}$	Z_{21}	$e^{j21\pi/16}$
Z_6	$e^{j3\pi/8}$	Z_{22}	$e^{j11\pi/8}$
Z_7	$e^{j7\pi/16}$	Z_{23}	$e^{j23\pi/16}$
Z_8	j	Z_{24}	-j
Z_9	$e^{j9\pi/16}$	Z_{25}	$e^{j25\pi/16}$
Z_{10}	$e^{j5\pi/8}$	Z_{26}	$e^{j13\pi/8}$
Z_{11}	$e^{j11\pi/16}$	Z_{27}	$e^{j27\pi/16}$
Z_{12}	$e^{j3\pi/4}$	Z_{28}	$e^{j7\pi/4}$
Z_{13}	$e^{j13\pi/16}$	Z_{29}	$e^{j29\pi/16}$
Z_{14}	$e^{j7\pi/8}$	Z_{30}	$e^{j15\pi/8}$
Z_{15}	$e^{j15\pi/16}$		

Tabla I. Ceros de H(z) para el filtro coseno con 32 muestras x ciclo y ventana de 1 ciclo.

La condición (I) determina que hay 14 pares de ceros complejos conjugados con lo cual estos podrán expresarse como factores de la forma:

$$(1 - 2\cdot Re\{Z_i\}\cdot z^{-1} + z^{-2}) \quad \text{con } i=2,3, \ldots 15 \qquad (2)$$

donde $Re\{Z_i\}$ es la parte real del cero correspondiente. La forma particular del polinomio se debe a la ubicación de los ceros, en el círculo unitario . En particular para i=8 el polinomio queda de la forma: $(1+z^{-2})$. A su vez por la condición (II), de estos 14 pares hay 6 de la forma $(1 - 2\cdot Re\{Z_i\}\cdot z^{-1} + z^{-2})$ y 6 de la forma $(1 + 2\cdot Re\{Z_i\}\cdot z^{-1} + z^{-2})$ con i=2,3, ..., 6. De este modo se pueden multiplicar estos factores para dar 6 de la forma:

$$(1 + [2- 4\cdot(Re\{Zi\})^2]\cdot z^{-2} + z^{-4}) \quad \text{con i=2,3, ..., 7} \qquad (3)$$

conteniendo a 24 ceros de H(z). Además queda uno de la forma (2) para i=15. Los factores para Z_0 Z_{16} Z_8 y Z_{24} se juntan para formar :

$$(1-z^{-1})(1+z^{-1})(1+z^{-2}) = (1-z^{-4}) \qquad (4)$$

Por último queda un factor $(1-0{,}981z^{-1})$ correspondiente al único cero de H(z) que no está sobre el círculo unitario. Para resumir en la Tabla II se presenta la factorización propuesta para H(z), diferenciando los tipos de factores y la cantidad de operaciones que ellos involucran (sumas, multiplicaciones y retardos). Las sumas indicadas en la Tabla II se consideran de a dos elementos.

Forma de los factores	Cantidad de factores	Operaciones por factor		
		sumas	multiplicaciones	retardos
$(1 + [2- 4(Re\{Zi\})^2]\cdot z^{-2} + z^{-4})$	6	2	1	4
$(1 - 2\cdot Re\{Z_{15}\}\cdot z^{-1} + z^{-2})$	1	2	1	2
$(1-z^{-4})$	1	1	0	4
$(1-0{,}9808z^{-1})$	1	1	1	1

TOTAL	9	16	8	31

Tabla II. Resumen de cantidad de operaciones con factoreo propuesto para H(z)

A todos estos factores debe agregarse una etapa de atenuación de 0,0625 para obtener el filtro descripto por (1). El diagrama en bloques que representa esta realización es el mostrado en la figura 3.

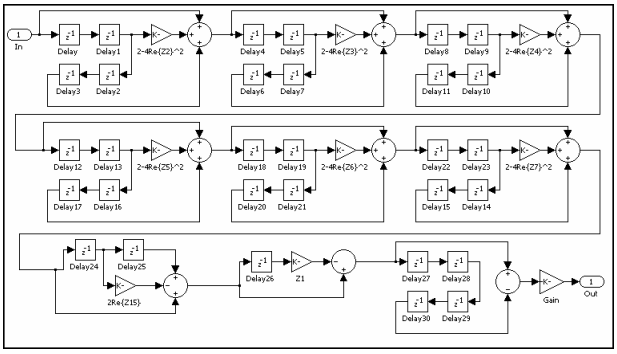

Fig. 3. Diagrama en bloques de realización en cascada del filtro con la factorización propuesta.

Comparando con la realización directa del filtro la realización en cascada posee la ventaja en cuanto a la disminución de la cantidad de operaciones a realizar.

Para observar esta situación se analiza brevemente la realización directa considerando la simetría de los coeficientes de la respuesta al impulso (coincidentes con los de la ecuación en diferencias). Se cumple que h[n+16]=h[16-n] para n=0,1, ..., 15. Se observa también que h[8]=h[24]=0 con lo cual no será necesario efectuar dichas multiplicaciones. Además h[8-n] = -h[8+n] con n=1, ...,7. Todo esto permite describir a la realización directa utilizando 8 multiplicaciones, 29 sumas (o restas) de a dos elementos y se utilizan 31 retardos. De este modo la realización en cascada reduce la cantidad de sumas de dos elementos a realizar. Otra de las ventajas de la cascada respecto a la realización directa es que la primera minimiza los efectos de los errores de cuantificación de los coeficientes, debido a la longitud de palabra finita con la que se cuenta en las implementaciones [6].

Una vez filtradas ambas señales, se procede al cálculo de los fasores correspondientes. Para ello, se retarda ¼ de onda la señal de salida del filtro (la componente de la señal de entrada de frecuencia fundamental) para obtener la componente en cuadratura del fasor como se muestra en la figura 4. El bloque Cos Filter32 en dicha figura representa al filtro coseno mostrado en la figura 3. Luego la señal sin retardo representa la parte real del fasor y la señal retardada la parte imaginaria del fasor. Como un ciclo corresponde a 32 muestras, ¼ de onda corresponderá a 8 muestras.

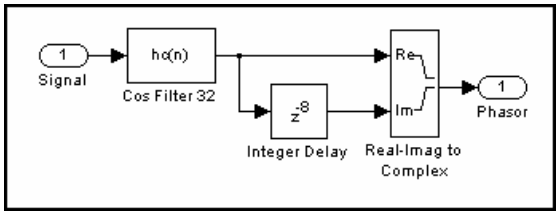

Fig.4. Formación del fasor a partir de la señal de salida del filtro.

Con la parte real e imaginaria del fasor se forma la señal fasorial como una señal compleja. Por último, con los fasores de tensión U y de corriente I se calcula el de impedancia Z, realizando un cociente de números complejos.

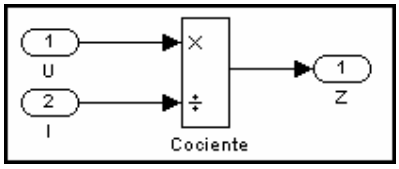

Fig. 5. Cálculo del fasor de impedancia.

El sistema completo se describe entonces, a través del diagrama en bloques que se muestra a continuación:

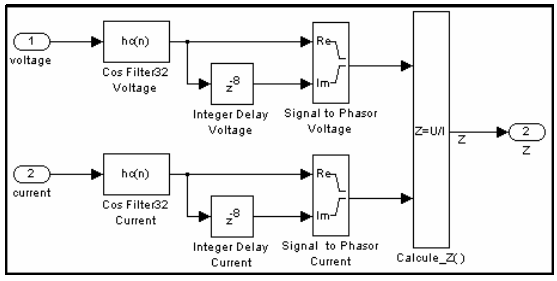

Fig.6. Diagrama en bloques del sistema completo

Los bloques Signal to Phasor convierten dos señales reales interpretadas como componente real e imaginaria en una señal compleja, mientras que el bloque Calcule_Z corresponde al diagrama mostrado en la figura 5.

– Simulación funcional.

En esta sección se presentan algunos resultados de simulaciones realizadas en Simulink®, para comprobar la correcta funcionalidad del sistema descripto en la etapa anterior. En esta fase del Codiseño HW/SW se comprueba que la descripción satisfaga la funcionalidad propuesta en el documento de requisitos, y no las restricciones de tiempo de respuesta, consumo, etc., ya que estas últimas dependerán del tipo de implementación, siendo entonces tópicos importantes para etapas posteriores.

Se simularon casos típicos de señales de falla en sistemas eléctricos, con presencia de componentes exponenciales y armónicos de la frecuencia fundamental. En la figura 7, se muestra una de estas simulaciones. Se presentan las señales de tensión y corriente de entrada y los valores de salida del módulo y

la fase del fasor de impedancia calculado por el sistema. La falla ocurre en el tiempo t=60ms correspondiendo a la muestra n=96 (32 muestras por ciclo). Los valores de las magnitudes están dados por unidad (p.u.). La señal de corriente presenta una característica exponencial luego de la falla y además se le agregaron componentes de las 2ª, 3ª y 5ª armónicas [5]. La señal de tensión también presenta las mismas componentes armónicas luego de la falla.

Inicialmente el valor de impedancia es de $10{,}09+j\cdot 0{,}99 = 10{,}149\cdot e^{j5{,}654^{o}}$ [p.u.], mientras que luego de la falla el valor es de $0{,}1+j\cdot 1 = 1{,}005\cdot e^{j84{,}28^{o}}$ [p.u.]. Se observa en la figura 7 que el sistema obtiene el nuevo valor de impedancia después de un ciclo de la frecuencia fundamental de ocurrida la falla. Esto último es una característica de los filtros con ventana de un ciclo, ya para obtener el valor actual procesan las últimas N muestras de las señales de entrada que entran en un ciclo de la fundamental.

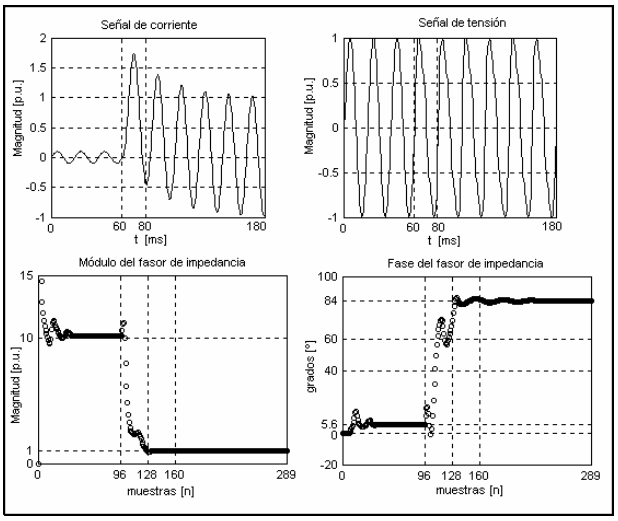

Fig. 7. Señales de entrada y salida obtenidas de la simulación del sistema en Simulink®.

Una vez realizada la simulación se comprobó la correcta funcionalidad del modelo de la figura 6.

4. IMPLEMENTACIÓN DE UN PROTOTIPO DEL SISTEMA.

Para llegar a un sistema experimental, en esta etapa del proyecto, se consideró la utilización de una arquitectura PC con procesador Intel Pentium, bus PCI y una placa de adquisición PCI-1200 de National Instruments, que permitiera el ingreso de las variables de entrada (tensión y corriente) en el código generado desde simulación funcional.

Como la placa posee un único conversor y se deben muestrear dos señales, el muestreo de las señales de tensión y corriente no son simultáneos sino mas bien secuenciales. De este modo la placa ha sido programada para trabajar en modo de adquisición con intervalo de exploración de dos canales [7], esto es la placa adquiere una muestra de un canal y después de un tiempo llamado intervalo de muestra adquiere una muestra del otro canal. Esta secuencia se repite cada intervalo de tiempo determinado por el llamado intervalo de exploración.

El programa se ha diseñado de forma tal que se satisfagan los requerimientos de tiempo real que impone dicho sistema. Para asegurar esto último se trabajó con una de las características de la placa PCI-1200, que permite generar un pedido de interrupción una vez realizada la conversión. Así al ser las conversiones periódicas se tendrá una interrupción también periódica que sirva como reloj para manejar al sistema de tiempo real. Se generan dos interrupciones en cada intervalo de exploración. La primera realiza el servicio del canal 1 de la placa (CH1-señal de tensión), mientras que la segunda hace lo propio con el canal 0 (CH0-señal de corriente). Como el intervalo de exploración representa el tiempo transcurrido entre muestras sucesivas de cada canal, se ha fijado su valor en 625μs correspondiendo a una frecuencia de muestreo de 1600 Hz (la elegida para los filtros) por canal.

El intervalo de muestra, que representa el tiempo trascurrido entre el muestreo de la señal de tensión y la de corriente, se tomó de 10 μs. Esto significa que entre la adquisición de la señal de tensión y la adquisición de la señal de corriente existirá un retardo de tiempo de dicho orden. Obviamente se introducirá un error en el cálculo de la fase de la impedancia, pues la misma se calcula como la diferencia entre la fase del fasor de tensión y la del fasor de corriente. De todos modos este retardo de 10μs representa solo 0,18º para una señal de 50 Hz con lo cual el error introducido en el cálculo de la impedancia es prácticamente nulo. Por este motivo no existe necesidad de emplear un Sample & Hold (S/H) para retener la señal de corriente mientras se muestrea la de tensión. Más aún, el error de fase es siempre el mismo y de este modo puede ser desafectado. En la figura 7 se muestra un diagrama de tiempos que simboliza la ejecución de las interrupciones.

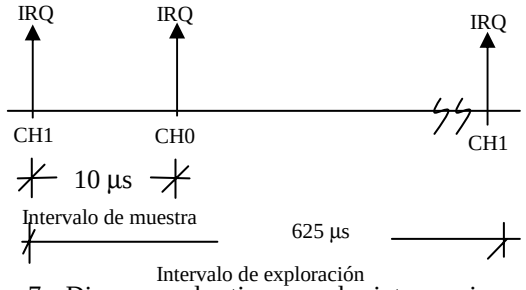

Fig. 7. Diagrama de tiempos de interrupciones del sistema

Para lograr el sincronismo se introduce en la rutina de servicio del CH1 una variable que mantenga información acerca de la ejecución del sistema. La misma determinará si ha concluido la ejecución del sistema o si siguen realizándose los cómputos cuando se atiende la interrupción, generando esto último una falla del sistema de tiempo real. Se observa que se debe ejecutar el sistema antes de los 625 μs.

El código principal del programa, correspondiendo a la función main() presenta la siguiente forma:

```
int main( ){
        User_Setup( );
        Config_Board( );
        Init_ISR( );
        Kernel( );
        Return_Old_ISR( );
        return 0;
}
```

Cada una de estas tareas realiza las funciones de más alto nivel del sistema. User_Setup() realiza la interfase con el usuario para aceptar los parámetros del sistema definidos por éste. Config_Board() configura la placa de adquisición en función de las necesidades del sistema de medición. Init_ISR() instala en la tabla de vectores de interrupción, al vector que representa la dirección de la función Read_Sample(), encargada del adquirir las muestras desde la placa. Por último Kernel() controla el despacho de las diferentes tareas del sistema de medición de impedancia. Cuando una falla se sucede, termina la ejecución de la función Kernel() y se invoca a Return_old_ISR() para retornar los vectores de interrupción originales.

Kernel() invoca en primer lugar a la función Init_System() que inicializa las variables, punteros y arreglos (arrays) del sistema de medición o sea System(), y luego envía un comando a la placa PCI-1200 para comenzar a adquirir. Enseguida ingresa en un bucle infinito el cual ejecutará al sistema hasta que una falla en la red se suceda, momento en el cual se registrarán los últimos valores de la impedancia (Z) y se accionará el interruptor. Cuando ingresa al bucle infinito el programa se mantiene idle hasta que wake_up=1, momento en el cual comienza la ejecución de System(). La variable wake_up es controlada por Read_Sample() y vale cero hasta que se obtiene la muestra de tensión momento en el cual pasa a valer 1. Siguiendo con el flujo de control, una vez ejecutado System() se invoca a Show_Results() para mostrar en pantalla al usuario los valores de impedancia. Antes de finalizar, el proceso System() hace wake_up=0. Si Read_Sample() comienza a servir nuevamente al CH1 y wake_up vale 1 entonces ocurrirá una falla del sistema puesto que esto significa que el mismo aún no ha terminado de ejecutarse.

La función System() invoca en primer lugar a FIR_Filter()para filtrar la señal de tensión y obtener el fasor correspondiente. El algoritmo de filtrado se implementa a través de una suma de convolución de los productos parciales entre los coeficientes del filtro coseno guardados en un arreglo de 32 elementos y las últimas 32 muestras de la señal de entrada (tensión o corriente según corresponda) almacenadas en una cola circular también de 32 elementos. El uso de esta cola circular permite actualizar las muestras con las sucesivas conversiones desechando la más antigua y aceptando la más nueva de manera dinámica. La cola circular para las muestras de tensión (corriente) se realiza con un arreglo de 32 elementos voltage (current) y con una variable puntero first_u (first_i) que indica la posición dentro del arreglo donde está almacenado el elemento más antiguo. Se cumple que si la muestra más antigua (la primera en entrar a la cola) está en la posición apuntada por first_u entonces la última en entrar estará en first_u–1. Para el caso de first_u=0 la última estará en la posición 31.

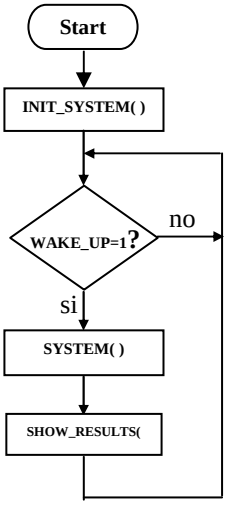

Fig.8. Flowchart del proceso Kernel().

En la línea 4 de System() se espera hasta que Read_Sample() haya obtenido el valor de corriente, momento en el cual asignara el valor 0 a channel. Siguiendo con el flujo, la línea 5 calcula el valor del fasor de corriente I con la función FIR_Filter(). Una vez obtenidos ambos fasores se procede al cálculo de la impedancia con Calcule_Z() y luego en función de este valor se decide si llamar a Switch_Control() o no. Antes de retornar el control a Kernel() se hace wake_up=0 para esperar la nueva conversión.

Código de System.

```
void System( )
{
        U=FIR_Filter(voltage,first_u);
        while(channel);
        I=FIR_Filter(current,first_i);
        Z=Calcule_Z(U,I);
        wake_up=0;
        return;
}
```

5. Conclusiones y trabajo futuro.

El Codiseño hardware/software permite optimizar un diseño particionando el sistema en componentes hardware y algoritmos software, considerando determinados factores de calidad. Sin embargo, surgen comúnmente limitaciones debido a indisponibilidad de herramientas adecuadas para el desarrollo de las

distintas etapas del Codiseño en particular la de particionamiento que se realiza en forma manual.

En esta primera etapa del trabajo, la herramienta Simulink® y Real Time Workshop®, han sido muy útiles para el desarrollo de la etapa de especificación y simulación funcional del sistema, y para la exploración del espacio de diseño.

Por otro lado, el prototipo desarrollado, usando una placa de adquisición de datos instalada en el bus PCI de una PC Pentium, permitió corroborar la funcionalidad a tiempo real del sistema.

En la realización de la segunda parte del proyecto, se abordarán las restantes etapas del Codiseño HW/SW, hasta llegar al sistema dedicado que se utilice en la práctica; y se tendrá en cuenta el uso de otras herramientas de diseño asistido por computadora disponibles (VHDL (VeriBest), Max Plus II de ALTERA, Optimizing C Compiler for TMS320, etc). Así también, para ampliar el espectro de posibilidades de diseño en nuestro centro, actualmente se estudia el entorno de Codiseño Ptolemy 0.7.1, desarrollado por el grupo de trabajo que lleva el mismo nombre y perteneciente a la universidad de Berkeley, USA. Este entorno permitirá, por ejemplo, generar códigos VHDL y C para ser empleados en la cosíntesis, coverificación e implementación de microcircuitos específicos para el sistema dedicado.

6. Referencias

[1]. H.O. Pascual, J.A. Rapallini , A.A. Quijano, "Implementación de un sistema de medida de impedancia para redes eléctricas en tiempo real." VII Workshop IBERCHIP IWS'2001, Montevideo, Uruguay, Marzo de 2001.

[2]. D.W. Franke y M. Purvis, "Hardware/Software Codesign: A Perspective." In Proceedings of the 13th Conference on Software Engineering, Austin-Texas (USA), Mayo de 1991.

[3]. Garcia Rozo, Sistemas Digitales. Elementos para un diseño a alto nivel. Colombia: CYTED, 1999.

[4]. B. Porat: A Course in Digital Signal Processing. New York: John Wiley & Sons, Inc., 1997.

[5]. E.O.Schweitzer, Daqing Hou, "Filtering for protective relays". 47th Annual Georgia Tech Protective Relaying Conference, Atlanta Georgia, April 1993.

[6]. J.G.Proakis y D.G. Manolakis, Tratamiento Digital de Señales. Principios, algoritmos y aplicaciones. 3ª Ed. España: Prentice Hall, 1998

[7]. PCI-1200 User Manual. National Instrumets Corporation, 1997.

[8]. G.G. Gastaldi, J.A. Rapallini, H.O. Pascual: Codiseño hardware/software para la implementación de un sistema de medida de impedancia. IX Jornadas de Jóvenes Investigadores de la A.U.G.M., MI-05,Univ. De Rosario, Argentina, 2001.

[9]. R. Niemann, Hardware/Software Co-Design for Data Flow Dominated Embedded Systems. Kuwler Academic,1998.

[10]. Informe Técnico CeTAD -CoHS03/00: Real Time Workshop 4. Real Time Windows Target2. Departamento de Electrotecnia, Fac. de Ingenieria, UNLP, 2000.

[11]. G.G. Gastaldi, J.A. Rapallini, H.O. Pascual, "Evaluación de programas de cálculo en ingeniería como herramienta de desarrollo para Codiseño Hardware / Software". IWS2002 VIII Workshop IBERCHIP , Guadalajara, México, Abril 2002.