Controlador tipo PID, sobre microcontrolador embebido en FPGA

David M. Caruso, Salvador E. Tropea
Instituto Nacional de Tecnología Industrial
Centro de Electrónica e Informática
Laboratorio de Desarrollo Electrónico con Software Libre
Email: {david,salvador}@inti.gob.ar

***Resumen*—El presente trabajo, muestra la aplicación de un algoritmo tipo PID, implementado sobre un microcontrolador compatible con un AVR, embebido en FPGA, dedicado al control de posición de un motor de corriente continua.**

Las características del algoritmo PID, lo hacen ideal para esta aplicación, brindando toda una variedad de formas de control del sistema. El diseño aquí presentado, se basa en la filosofía "hardware and software co-design", mostrando una alta eficacia en la aplicación y gran maleabilidad para mejoras futuras.

Este desarrollo incluye los siguientes periféricos: lector de encoder relativo, UART y PWM.

La interconexión entre el microcontrolador y dichos periféricos, se realiza bajo el estándar WISHBONE.

El diseño fue verificado usando simuladores y FPGAs de Xilinx (*Spartan* II y 3A), junto con la electrónica adecuada para el manejo del motor.

I. Introducción

Los controladores PID, son ampliamente usados desde hace varias décadas, en el control de todo tipo de sistemas lineales, dada su alta eficacia, simplicidad y su fácil aplicación. En este caso, se ve aplicado al control de posición de un motor de corriente continua, pero mediante ligeras modificaciones, puede aplicarse a cualquier tipo de sistema lineal. Es por esto que se seleccionó este tipo de algoritmo.

Por otro lado, nuestro laboratorio realiza aplicaciones con microcontroladores embebidos de forma frecuente. En este caso, se quiso llevar a la práctica, esta simple versión de control que involucra el trabajo en conjunto de hardware dedicado y software. Dicha filosofía de trabajo, aplicada sobre una FPGA, trae grandes beneficios, entre ellos, la fácil implementación e integración del sistema completo de control, la simplicidad con la que se añaden modificaciones al mismo, etc. La forma en la que se reparten las tareas entre el hardware y el software, logra encontrar un equilibrio que utiliza el mínimo de recursos, alcanzando la funcionalidad deseada.

El artículo está organizado de la siguiente forma. En la Sección II, se puede encontrar una breve descripción sobre como es el sistema a controlar, sumado a conocimientos generales sobre el algoritmo PID. La implementación, se resume en la Sección III, donde se describe parte por parte, los elementos constitutivos del sistema, tanto hardware como software. Dentro de la sección IV, se muestran los resultados de la síntesis de los elementos embebidos en la FPGA, y de la resolución adquirida en el control de posición. Las conclusiones, se ven enumeradas en la última sección de esta publicación.

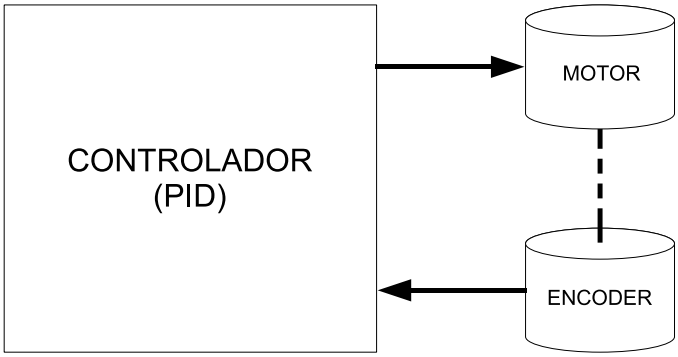

Figura 1. Sistema de control

II. Sistema de Control

A modo de esquematizar el problema aquí planteado, se incluyó una imagen que lo describe en forma de bloques (Fig. 1). Donde, el controlador toma (a través del encoder relativo) los datos de la posición actual del motor y computa el algoritmo PID. A su vez, el resultado del cálculo, se transforma en una señal modulada en ancho de pulso, que actúa sobre el motor dándole mayor o menor velocidad, logrando que el mismo tienda hacia la posición que se desea. En síntesis, puede decirse, que el controlador intenta controlar la posición a partir de una variación coherente de la velocidad del motor. Donde, dicha variación dependerá de como esté calibrado el PID, para obtener una respuesta dentro de los requerimientos de la aplicación. En III-D se muestran los métodos de calibración utilizados.

Notar, que la línea de trazos, representa el acople mecánico entre los ejes del motor y el encoder. El cual, puede verse como un camino de realimentación.

II-A. Algoritmo PID

El núcleo de este sistema de control, es el cálculo del algoritmo PID. La ecuación 1 muestra el cálculo del control PID a partir del error $e(t)$.

$$S_{pid} = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \qquad (1)$$

Donde K_p es la constante proporcional, K_i la integral y K_d la derivativa. Cada una de estas constantes tiene un impacto diferente en el control:

- **Término proporcional**: (K_p) es el encargado de indicar que tan lejos se ubica actualmente el sistema, con respecto al valor deseado. Se puede ver que mientras más alta sea la constante proporcional, menor será el tiempo que tarde en reducir el error del sistema, pero tendrá oscilaciones subamortiguadas alrededor del valor deseado. En cambio si la constante es pequeña, puede que el sistema no tenga oscilaciones, pero nunca alcanzará al valor deseado (error de offset).
- **Término integral**: (K_i) almacena información sobre las actividades pasadas del sistema. Esta información es de vital importancia, ya que le brinda al sistema de control, una memoria resumida, sobre como fue evolucionando el error del mismo. Es capaz de corregir el error de offset, dado por la parte proporcional, en un tiempo de actuación que es inversamente proporcional a la constante K_i. Si esta última, resulta muy grande (K_i>1), el sistema presentará oscilaciones antes de alcanzar el valor deseado.
- **Término derivativo**: (K_d) hace una comparación entre el error pasado y el actual, con lo que obtiene, una noción de como seguirá respondiendo el sistema. La misma, es utilizada, para contrarrestar variaciones abruptas que puedan ser nocivas, como las oscilaciones. El tiempo de anticipación del cálculo, depende de K_d, mientras mayor sea la misma, el sistema se anticipará más rápido.

III. Implementación

Se optó por una arquitectura, centralizada en el microcontrolador, el cual mantiene la comunicación con periféricos y la pc, computa el PID, mientras que los periféricos se encargan de las tareas simples en tiempo real.

La interconexión, entre el microcontrolador y los periféricos, es compatible con WISHBONE [1]. La misma, presenta una fácil adaptación a cualquier periférico, es decir, que adaptar un dispositivo a este estándar, es casi trivial. Además, el agregado de más periféricos al bus, es muy sencillo.

En la Fig. 2, se pueden ver la arquitectura implementada dentro de la FPGA.

III-A. Hardware embebido en la FPGA

Todas las descripciones, fueron transferidas a la FPGA Spartan 3A del kit de desarrollo de Avnet. Antes de ser llevados a hardware, cada uno de los módulos de VHDL descriptos anteriormente, fueron testeados usando bancos de prueba. Para dicha simulación se utilizó el compilador *GHDL* [2] 0.28, junto con otras herramientas recomendadas por el proyecto FPGALibre [3].

III-A1. Microcontrolador: Surge de una descripción en VHDL compatible con el AVR Atmega8 [4]. El mismo fue programado en lenguaje C, con el algoritmo PID, y responde a las tareas de comunicación con la PC como con los periféricos del sistema de control. Puede aceptar comandos como:

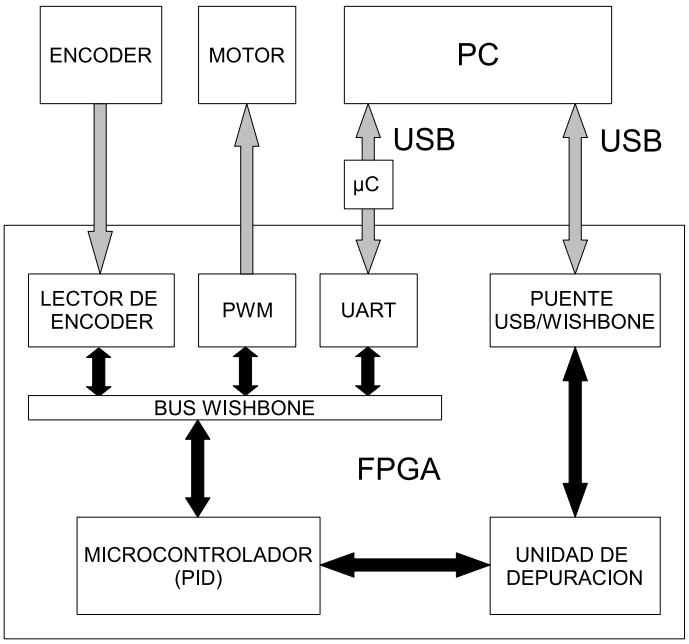

Figura 2. Arquitectura interna de cores embebidos

- Modificar los parámetros del PID (constantes proporcional, derivativa e integral).
- Indicar el ángulo de destino.
- Posición de referencia, en caso de no contar con la señal Z del encoder.
- Hacer girar en forma continua al motor, con una velocidad seleccionada.
- Seleccionar el sentido de giro.

Cada 10 milisegundos el microcontrolador computa el algoritmo PID, y de esta forma puede mantener un control de posición estable. Esto se logra a través del uso de interrupciones, generadas por un timer fijo.

III-A2. UART: Dicho core, fue mejorado en nuestro laboratorio, basado en una descripción de Open Cores [5]. Implementa una UART, con baudrate de 115200 fijo (configurable pre-síntesis), 8N1 y compatible con WISHBONE.

III-A3. PWM: Core modulador de ancho de pulsos, compatible con WISHBONE, que opera en una resolución de 16 bit y a una frecuencia de 30KHz.

III-A4. Decodificador de encoder relativo: Se encarga de tomar las señales del encoder físico, y traducirlas en cantidad de pasos. Opera, contando los cruces de las señales desfasadas 90° del encoder físico, de esta forma, incrementa en cuatro veces la resolución dada por el fabricante.

El mismo tiene una resolución de 16 bit, y posee un filtro para eliminar glitches que puedan producirse sobre las señales que entrega el encoder físico.

III-A5. Unidad de depuración: Permite depurar el programa del microcontrolador de una forma rápida. A través de la misma, se hace posible, cambiar el programa almacenado en los *block RAMs* (*BRAMs*) de la FPGA, sin tener que sintetizar una nueva descripción completa del sistema. Además, permite

la ejecución paso a paso del programa, uso de *breakpoints*, *watchpoints*, etc.

III-A6. Puente USB/WISHBONE: Permite controlar un bus WISHBONE desde el exterior de la FPGA, utilizando el puerto USB [6].

Es el medio de comunicación entre la unidad de depuración y el mundo exterior, permitiendo que a través de ella, fluyan los datos para reprogramar la memoria del microcontrolador, entre otras opciones.

III-B. Hardware externo a la FPGA

III-B1. Driver: Sobre una placa auxiliar, se montó el integrado L298, que es un doble puente H, para manejar las corrientes del motor, muy altas para la FPGA. Se utilizaron optoacopladores, para separar las señales que provenían de la FPGA de las de potencia. El L298, excita al motor con 12 V, mientras que desde la FPGA las señales actúan en 3,3 V. Los optoacopladores, transfieren las señales a 5 V para operar sobre el L298.

III-B2. Motor: El motor utilizado es el B138F-12-36, que posee una relación de reducción de 36:1, con una tensión de trabajo de 12V. El mismo gira a 73 rpm sin carga y a 53 rpm para su máximo torque.

III-B3. Encoder: El encoder relativo utilizado es el s5s-720, el cual cuenta con una resolución de 720 pasos por vuelta (llevados a 2880 por hardware) y una señal de cero (señal Z). Esto permite una resolución de un octavo de grado.

III-C. Software de control

III-C1. PID: La expresión 1, refleja como es un controlador PID que actua en forma continua. La aplicación de la misma en, un sistema digital, requiere que la ecuación se exprese en variable discreta (ver ecuación 2).

$$S_{pid}[n] = K_p e[n] + K_i \sum_{k=0}^{n-1} e[n-k] - K_d(e[n]-e[n-1]) \quad (2)$$

La expresión 2, para ser aplicada en un sistema computacional limitado, como lo es un microcontrolador, debió ser tratada término a término antes de poder ser calculada. Se enumeran, en forma resumida, dichos aspectos:

- **Prevención de desborde**: se entiende por desborde, al problema que surge cuando sobre registros de tamaño limitado, se pretenden realizar operaciones cuyo resultado es mayor al que puede retener el mismo. Por esto, todas las constantes del PID, son previamente evaluadas (una vez que son configuradas), para evitar que se produzca un desborde, por lo que su rango de valores posibles está limitado. A su vez, cada término del PID es analizado para ver si existieron desbordes, corroborando los signos entre los operandos y el resultado.
  Por otro lado, cada etapa del algoritmo, tiene limitado su valor máximo de participación en el resultado, es decir, poseen una saturación prefijada, para asegurar que no se produzca un desborde final.

Cuadro I
RESULTADOS DE LA SÍNTESIS

LUTs	FFs	Slices	BRAMs	Clock
2705	1019	1700	7	>24 MHz

- **Corrección cruces por cero**: el sistema, al tener libertad de giro, cada vez que completa un giro, la posición vuelve a ser referida a cero. Por esto, es que se debió implementar por software, un sistema que contemple esta opción, y filtre estas transiciones abruptas, de forma que, el cálculo derivativo no lo interprete como una acción inesperada del sistema.
- **Limitación de operación del cálculo integral**: los errores almacenados en el comienzo de la búsqueda de un nuevo punto requerido, son mayores a los que se dan cuando está cerca del valor solicitado. Como la porción integral del PID, posee una memoria, estos errores iniciales almacenados, necesitan ser contrarrestados, por lo que producirán oscilaciones en el sistema. A fin de evitar esto, se hizo que el cálculo integral sólo se aplicase si el error actual es menor a un parámetro.

III-C2. Software de comunicación (PC): Desde el lado de la PC, actúa un programa implementado en C++, el cual se encarga de realizar la calibración del PID, de forma tal de definir las constantes del mismo. La comunicación se mantiene vía USB. El kit de desarrollo utilizado, cuenta con un microcontrolador, que actúa como puente entre USB/RS-232, de forma que del lado de la FPGA se vea como RS-232. De esta forma, el microcontrolador embebido en la FPGA, utiliza una UART RS-232 como herramienta de comunicación, y se puede comunicar vía USB al mundo exterior.

III-D. Calibración PID

Todo sistema a controlar, tiene una ecuación que lo caracteriza, a la cual el equipo a controlar tiene que adecuarse, para así lograr la respuesta óptima. Modelizar el sistema, resulta un trabajo generalmente tedioso, por lo que se utilizaron métodos genéricos de calibración, tales como Ziegler-Nichols [7] y Harriot [8]. Estos métodos, a través de diversas excitaciones que le aplican al sistema, determinan cuales son las constantes del PID necesarias para que el sistema responda a un escalón con una forma de onda determinada. Son de fácil aplicación y computación.

En la Fig. 3, puede verse la oscilación presentada por el sistema cuando se lo calibró por el método de Ziegler-Nichols.

IV. RESULTADOS OBTENIDOS

Para la síntesis se utilizó el programa XST 10.1.02 K.37, obteniéndose los resultados expuestos en el Cuadro I.

Una vez llevado a hardware, el sistema, se verificó la comunicación entre la PC y el microcontrolador embebido, se pudo realizar el control de posición del motor con una precisión del orden de la resolución del encoder (0,25°). La respuesta del motor, puede verse representada en la Fig. 4. La misma, es la que se obtiene con el PID calibrado por el

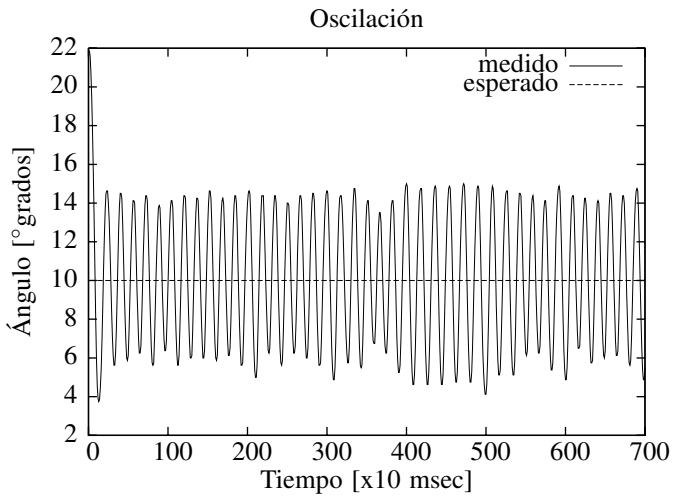

Figura 3. Oscilación - método Ziegler-Nichols

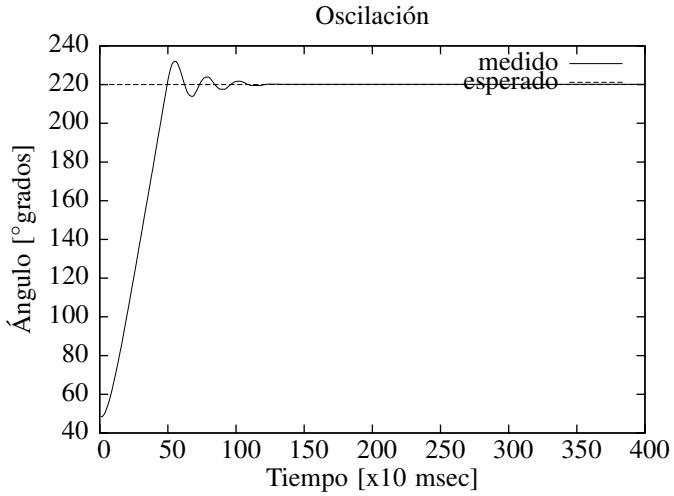

Figura 4. Respuesta al impulso - motor DC

método de Ziegler-Nichols, el cual, de antemano predice esta respuesta del sistema.

V. Conclusiones

La aplicación de la filosofía hardware and software codesign, permite hacer una repartición de tareas para alcanzar el punto de equilibrio en el que las dos partes poseen las mínimas dimensiones. En este caso, se puede ver que el hardware requerido es muy pequeño y se dedica a tareas de tiempo real, dejando que el microcontrolador se encargue del cálculo y el mantenimiento de la comunicación, logrando un software embebido sencillo y dedicado a tareas específicas. Con lo que, el microcontrolador embebido no es necesario que este totalmente completo como el dispositivo real en el que se basó, de forma que se implementa en hardware sólo las partes realmente necesarias del mismo.

El uso de la FPGA, permite que en versiones futuras se agreguen o quiten dispositivos, según la aplicación lo requiera. Como sería el caso de la unidad de depuración, la cual sólo es necesaria durante las pruebas.

El estándar de interconexión WISHBONE, brinda una facilidad en el agregado de nuevos periféricos, generando un sistema de control aún más sofisticado.

La utilización de las herramientas propuestas por el proyecto FPGALibre mostró ser adecuada para este desarrollo.

Referencias

[1] Silicore and OpenCores.Org. (2010, Jun.) WISHBONE System-on-Chip (SoC) interconnection architecture for portable IP cores. [Online]. Available: http://prdownloads.sf.net/fpgalibre/wbspec_b3-2.pdf?download

[2] T. Gingold. (2010, Jun.) A complete VHDL simulator. [Online]. Available: http://ghdl.free.fr/

[3] S. E. Tropea, D. J. Brengi, and J. P. D. Borgna, "FPGAlibre: Herramientas de software libre para diseño con FPGAs," in *FPGA Based Systems*. Mar del Plata: Surlabs Project, II SPL, 2006, pp. 173–180.

[4] S. E. Tropea and D. M. Caruso, "Microcontrolador compatible con AVR, interfaz de depuración y bus wishbone," in *Proceedings of the FPGA Designer Forum 2010*, Ipojuca, Brazil, 2010, pp. 1–6.

[5] (2010, Jun.) Open cores. [Online]. Available: http://www.opencores.org/

[6] R. A. Melo and S. E. Tropea, "IP core puente USB a WISHBONE," in *XV Workshop Iberchip*, vol. 2, Buenos Aires, 2009, pp. 531–533.

[7] J. M. G. D. Pernia and E. Luzardo, "Introducción a los controladores pid," Ph.D. dissertation, Facultad de Ingeniería Universidad de los Andes, 2000. [Online]. Available: http://www.ing.ula.ve/dpernia/pdfs/introd_pid.pdf

[8] P. Harriot, *Process Control*. McGraw Hill, 1964.