Locomoción de un Robot Ápodo Modular con el Procesador MicroBlaze

González-Gómez J., Aguayo E. y Boemo E.

Escuela Politécnica Superior, Universidad Autónoma de Madrid, España,
{Juan.Gonzalez, Estanislao.Aguayo, Eduardo.Boemo}@ii.uam.es
http://www.eps.uam.es

Resumen. Los robots modulares reconfigurables prometen ofrecer mayor versatilidad, robustez y menor coste. Están construidos a partir de módulos pequeños y sencillos, capaces de unirse y separarse entre ellos. Cada módulo está controlado por un procesador convencional. En este artículo presentamos un prototipo de un robot ápodo modular (*cube revolutions*), constituido por la unión en cadena de 8 módulos iguales. Se desplaza en línea recta, por medio de ondas que recorren su cuerpo desde la cola hasta la cabeza. El robot calcula las posiciones de las articulaciones a partir de los parámetros de la onda: forma, amplitud y longitud de onda. Por flexibilidad, se ha utilizado el *soft-processor* Microblaze, empotrado en una FPGA Xilinx. Las FPGAs dotan a los robots Modulares de mayor versatilidad, al no depender de un procesador convencional concreto ni de una arquitectura hardware determinada.

1 Introducción

Los Robots modulares y reconfigurables ofrecen mayor versatilidad, robustez y menor coste[1]. Están constituidos por módulos capaces de separarse y unirse, cambiando la forma del robot. Se pueden desplazar por terrenos muy dispares y superar diversos obstáculos, logrando mayor versatilidad en la locomoción. Se está estudiando su empleo en aplicaciones espaciales[3] y de búsqueda y rescate en entornos urbanos[2].

Un robot modular que tiene N tipos diferentes de módulos se denomina N-modular. Se intenta reducir la heterogeneidad, disminuyendo la relación entre N y el número total de módulos. En los últimos años, se están desarrollando robots que siguen este enfoque[4][5][6][7].

Uno de los más avanzados es Polybot[4], de tipo 2-modular, capaz de realizar reconfiguraciones dinámicas de rueda a serpiente y de ésta a araña. Actualmente están trabajando en la tercera generación de módulos[8], los G3, que integran cada uno su propio procesador *powerPC* 555.

En este artículo estudiamos la viabilidad de utilizar FPGAs para el control de los módulos, en vez de procesadores convencionales. No disponen, a priori, de un procesador específico ni de una arquitectura concreta. El diseñador decide qué partes se realizarán en hardware y cuales en software.

En [9] estudiamos diferentes alternativas para la locomoción del robot ápodo *Cube Reloaded*[10], utilizando FPGAs. En la nueva versión, *Cube Revolutions* (figura 1), hemos diseñado un controlador de locomoción empleando el *soft-processor* Microblaze[11], junto con periféricos para el movimiento de los servos.

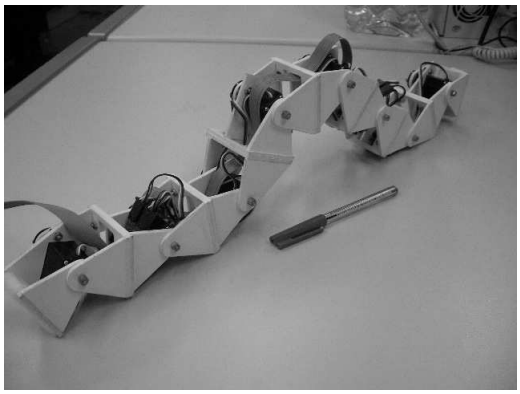

Fig. 1. El robot ápodo *Cube Revolutions*. Construido a partir de la unión en cadena de 8 módulos iguales, conectados en fase.

2 Mecánica

El prototipo construido, está formado por la unión en cadena de 8 módulos iguales, a los que llamamos módulos Y1. En la figura 2 se muestra el diseño en 3D. Tienen un único grado de libertad, actuado por un servomecanismo. El diseño de los módulos está inspirado en la generación G1 de *Polybot*.

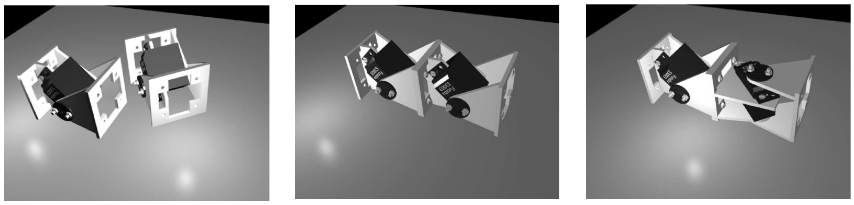

Fig. 2. Los módulos Y1. A la izquierda se pueden ver dos módulos sueltos. En la imagen central están conectados en fase, y en la derecha, con conexión desfasada.

Los módulos Y1 son sencillos y baratos. Permiten una rápida construcción de prototipos de robots ápodos. Se pueden conectar de dos maneras diferentes, como se muestra en la figura 2. Una es la conexión en fase, en la que dos módulos adyacentes tienen la misma orientación. Mediante esta encadenación, se construyen robots ápodos en los que todas las articulaciones permanecen siempre en el mismo plano, perpendicular al suelo. *Cube Revolutions* está constituido por 8 módulos Y1 conectados en fase, por lo que sólo puede desplazarse en línea recta.

Cada módulo, en su posición de reposo (ángulo de 0 grados), tiene unas dimensiones de 52x52x72 mm y un peso de 50gr. El material empleado es PVC expandido.

El rango de giro de está comprendido entre -90 y 90 grados. El robot tiene una longitud de 576mm y un peso total de 400gr. Tanto la electrónica como la alimentación se encuentran situadas en el exterior.

3 Locomoción

La locomoción se consigue aplicando ondas que recorren el cuerpo del gusano, desde la cola hasta la cabeza. Para simplificar la programación, se utilizan tablas de control[1] (*gait control tables*), descritas en el apartado 3.1. El controlador de locomoción (apartado 3.3) genera automáticamente estas tablas, a partir de las cuales se obtienen las señales PWM que posicionan los servos, haciendo que el robot se desplace.

3.1 Tablas de control

Cada articulación se caracteriza por el ángulo φ_i que forma un segmento con el anterior. El aspecto del gusano en un instante t viene determinado por el *vector de posición angular* $\overrightarrow{\varphi(t)} = (\varphi_1, \varphi_2, ..., \varphi_n)$. En la figura 3 se muestra este vector, en un instante dado, para un robot ápodo de 6 articulaciones.

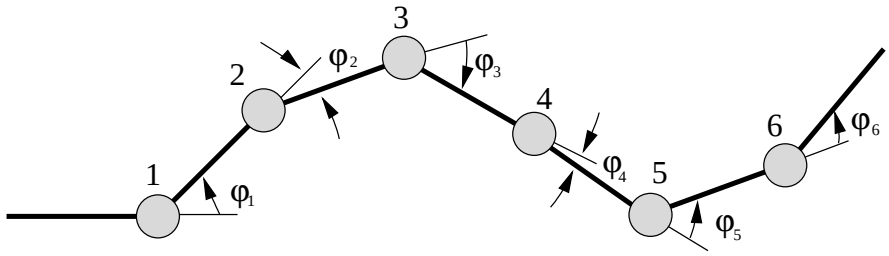

Fig. 3. Vector de posición angular para un ápodo de 6 articulaciones: $\overrightarrow{v} = (\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6)$

Para cada instante, existe un vectores de posición angular que determina la forma del gusano: $\overrightarrow{\varphi(t_0)}, \overrightarrow{\varphi(t_1)}, ..., \overrightarrow{\varphi(t_m)}$. La tabla de control es una matriz cuyas filas son los vectores de posición angular para los diferentes instantes. Para generar el movimiento, el controlador recorre la tabla, situando cada servo en la posición indicada.

En robots como *Polybot*, estas tablas están precalculadas, y se descargan en los módulos, consiguiéndose diferentes formas de locomoción (*gaits*). Es imposible tener calculadas o almacenadas todas las posibles tablas para todos los movimientos. En *Cube Revolutions*, se generan de forma automática.

3.2 Generación automática de tablas de control

En el prototipo desarrollado, las tablas de control se generan automáticamente, a partir de un modelo de propagación de onda. El algoritmo empleado se describe a continuación (figura 4). Partimos de una onda en el instante inicial, $f(x, t_0)$ (en el dibujo se utilizan ondas sinusoidales pero podrían tener cualquier otra forma) y de un modelo de gusano en el que todas sus articulaciones están sobre el eje x (estado inicial. Fig 4-1). Sean (x_i, y_i) las coordenadas de la articulación i-ésima, en ese instante. El vector de posición angular para ese instante, $\overrightarrow{\varphi(t_o)}$, se calcula haciendo que todas las articulaciones cumplan la función de onda $f(x, t_0)$, de manera que $y_i = f(x_i, t_0)$, siempre manteniéndose la restricción de que la distancia entre dos articulaciones sea *L*. Es decir, que el gusano se "ajusta" a la función de onda (4-2). A continuación se desplaza la onda (instante t_1. Figura 4-3) y se vuelve a realizar el "ajuste", obteniéndose $\overrightarrow{\varphi(t_1)}$ (Figura 4-4). Se repiten los puntos 3 y 4 hasta que la onda llegue a su fase inicial. Al cabo de *m* instantes de tiempo, se tienen todos los vectores que componen la tabla de control.

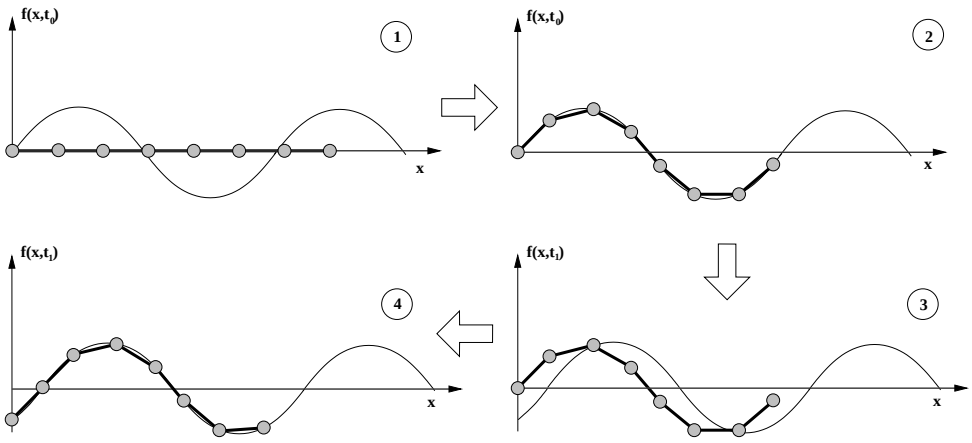

Fig. 4. Algoritmo empleado para generar automáticamente las tablas de control. 1) Estado inicial. 2) Las articulaciones cumplen la ecuación de la onda (el gusano se "ajusta" a la onda). 3) La onda se desplaza. 4) Se vuelve a "ajustar" el gusano a la onda

Mediante este algoritmo, se obtienen las tablas de control, con independencia de la forma de la onda usada. Se puede emplear para cualquier onda $f(x, t)$. Las pruebas de locomoción las hemos realizado utilizando ondas sinusoidales y semiondas (usando sólo la parte positiva de un periodo de una onda sinusoidal).

3.3 Controlador de locomoción

El controlador de locomoción genera las señales PWM para el posicionamiento de los servos a partir de los parámetros de la onda utilizada: forma, amplitud y longitud de

onda. Un sistema de control superior, podría mover el gusano sólo especificando estos parámetros. Se centraría en determinar qué ondas utilizar, y qué parámetros, en función del terreno por el que se vaya a desplazar. Por ejemplo, si el robot tiene que desplazarse por el interior de un tubo, se utilizaría una amplitud inferior a su diámetro. Si tiene que superar un escalón de una determinada altura, el controlador calcularía la amplitud necesaria.

La arquitectura se muestra en la figura 5. Está constituido por tres partes. El elemento principal es la *tabla de control*, donde se almacenan las posiciones de los servos y es la que caracteriza el movimiento (apartado 3.1). El *controlador de posición*, a partir de los valores de esta tabla, genera las señales PWM que se envían directamente a los servos.

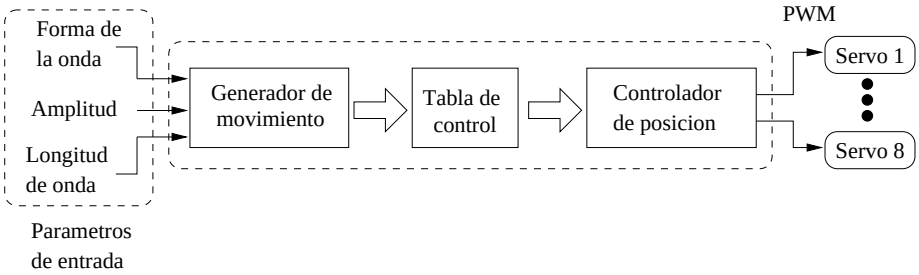

Fig. 5. Arquitectura del controlador del locomoción

El *generador de movimiento* obtiene la tabla de control a partir de los parámetros de entrada (forma de onda, amplitud y longitud de onda). Se implementa por *software*, según el algoritmo descrito en el apartado 3.2.

4 Implementación en FPGA

El controlador completo de locomoción, basado en *soft-processor* Microblaze se mapea en una FPGA SpartanIIE 400[12]. Tanto el generador de movimiento como las tablas de control están implementadas en *software*. Los algoritmos están programados en C. El compilador empleado es un *port* del GCC (GNU C Compiler), proporcionado por el fabricante de la FPGA.

El controlador de posición está descrito en VHDL. Se accede como un periférico de Microblaze, a través de puertos. El *software* sitúa las posiciones de los servos, y el controlador genera las señales PWM.

Una ventaja es que el sistema es muy escalable en relación al número de servos que se pueden llegar a controlar. La única limitación es la cantidad de área y el número de pines disponibles en la FPGA.

4.1 El procesador software Microblaze

El MicroBlaze[11] es un soft-procesor de 32 bits y arquitectura *Harvard*, diseñado por Xilinx. En la figura 6 se muestra el diseño cargado en la FPGA. Los *buses* siguen el estándar *Core Connect* de IBM[13]. También se ha incluido un módulo de depuración, para poder emplear la herramienta gdb de GNU[14].

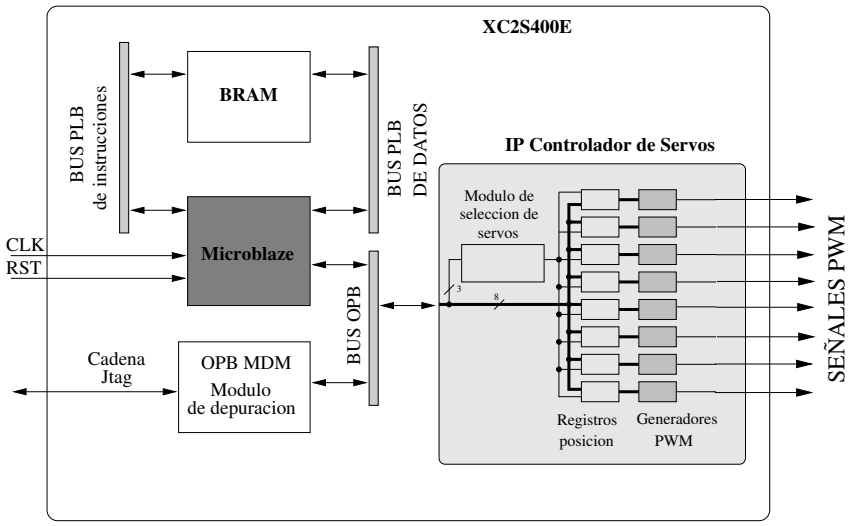

Fig. 6. Arquitectura del controlador de locomoción

El área ocupada por este procesador es de aproximadamente un 10 % en una Spartan IIE400, lo que deja un 90 % de espacio libre para añadir el *hardware* necesario.

4.2 Resultados obtenidos

La implementación del sistema de control se ha realizado con la herramienta ISE 6.1 de Xilinx y el EDK 6.1. La FPGA utilizada en *Cube Revolutions* es una SpartanIIE 400. Los resultados finales de la síntesis se muestran en la tabla 1

	Total	**Usado**	**Disponible**
BRAMs	14	8	6 (43 %)
Slices	2352	1312	1040 (44 %)
Pines de E/S	146	10	136 (93 %)
Frecuencia	–	50Mhz	—

Tabla 1. Resultados de la implementación del controlador de locomoción

Las 8 BRAM están configuradas para constituir una memoria de palabras de 32 bits. El controlador deja libre el 44 % del espacio y el 93 % de los *pines* de la FPGA, lo que permite reservar recursos para futuras mejoras. El reloj del sistema funciona a una frecuencia de operación de 50 MHz.

5 Conclusiones y trabajo futuro

Se ha construido un prototipo de un robot ápodo para estudiar la locomoción en línea recta. El controlador de locomoción usa tablas que se generan automáticamente a partir de la propagación de una onda a lo largo del gusano, que lo recorre desde la cola hasta la cabeza. Los tres parámetros de la onda (forma, amplitud y longitud de onda) determinan el movimiento. El software de más alto nivel sólo tiene que especificar estos parámetros para conseguir la locomoción.

El controlador de locomoción está implementando en una FPGA. Para la ejecución de los algoritmos, se utiliza el procesador MicroBlaze. Se han diseñado *cores hardware*, accesibles desde el mapa de memoria, para el posicionamiento de los servos.

La utilización de FPGAs permite diseñar robots más versátiles, donde se puede seleccionar la arquitectura más adecuada para el experimento a realizar. La única limitación son los recursos disponibles en la FPGA.

Dentro de los trabajos futuros, se estudiará la locomoción, analizando sus características en función de los parámetros de la onda aplicada y relacionándolos con la velocidad, estabilidad y consumo del robot. Uno de los enfoques será utilizar algoritmos genéticos, para determinar cuales son los parámetros óptimos, fijados unos requisitos de estabilidad, consumo y velocidad. Más adelante se abordará el movimiento en un plano, implementando los controladores en FPGA y finalmente se desarrollará una nueva generación de módulos, cada uno con su propia FPGA.

Agradecimientos

Este trabajo está financiado parcialmente por el Proyecto TIC2001-2688-C03-03 del Ministerio de Ciencia y Tecnología de España, y parcialmente por el Proyecto 07T/0052/2003-3 de la Consejería de Educación de la Comunidad de Madrid.

Referencias

1. Mark Yim, Ying Zhang & David Duff, Xerox Palo Alto Research Center (PARC), “Modular Robots”. IEEE Spectrum Magazine. Febrero 2002.
2. M. Yim, D.Duff, K.Roufas, “Modular Reconfigurable Robots, and Approach to Urban Search and Rescue,” Proc. of 1st Intl. WorkShop on Human-friendly welfare Robotic Systems (HWRS2000) Taejon, Korea, pp.69-76, Jan. 2000.
3. M. Yim, K. Roufas, D. Duff, Y. Zhang, C. Eldershaw, S. Homans, “Modular Reconfigurable Robots in Space Applications”, Autonomous Robot Journal, special issue for Robots in Space, Springer Verlag, 2003.
4. Mark Yim, David G. Duff, Kimon D. Roufas, “PolyBot: A Modular Reconfigurable Robot”, IEEE Intl. Conf. on Robotics and Automation (ICRA), San Francisco, CA, April 2000.

5. P. Will, A. Castano, W-M Shen, “Robot modularity for self-reconfiguration,” SPIE Intl. Symposium on Intelligent Sys. and Advanced Manufacturing, Proceeding Vol. 3839, pp.236-245, Sept. 1999.
6. K. Kotay, D.Rus, M.Vona, C.McGray, “The Self-reconfiguring Robotic Molecule,” Proc. of the IEEE International Conf. on Robotics and Automation, pp.424-431, May 1998.
7. S. Murata, H. Kurokawa, E. Yoshida, K. Tomita, S. kokaji, “A 3D self-Reconfigurable Structure,” Proc. of the IEEE International Conf. on Robotics and Automation, pp432-439, May 1998.
8. M. Yim, Y. Zhang, K. Roufas, D. Duff, C. Eldershaw, “Connecting and disconnecting for chain self-reconfiguration with PolyBot”, IEEE/ASME Transactions on mechatronics, special issue on Information Technology in Mechatronics, 2003.
9. J. González, I. González, E. Boemo, "Alternativas Hardware para la Locomoción de un Robot Ápodo”, III Jornadas sobre Computación Reconfigurable y Aplicaciones, JCRA03, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Septiembre 2003.
10. Robot ápodo Cube Reloaded. [En línea] http://www.iearobotics.com/personal/juan/doctorado/cube-reloaded/
11. Xilinx, inc, “Microblaze processor Reference Guide”. San Jose, California, Julio 2003.
12. Xilinx, inc, “Spartan-IIE 1.8V FPGA Family: Complete Data Sheet”. Julio 2003. [En línea] http://www.xilinx.com/bvdocs/publications/ds077.pdf
13. IBM inc, “On-Chip Peripheral Bus, architecure specifications”. Research Triangle Park, North Carolina. April 2001.
14. Proyecto GNU. [En línea] http://www.gnu.org