Evaluación de un Algoritmo de Locomoción de Robots Ápodos en Diferentes Procesadores Embebidos en FPGA

J. Gonzalez-Gomez, I. Gonzalez , F. J. Gomez-Arribas y E. Boemo

Escuela Politécnica Superior, Universidad Autónoma de Madrid

{juan.gonzalez,ivan.gonzalez,francisco.gomez,eduardo.boemo}@uam.es

Resumen

La locomoción de robots ápodos modulares se realiza mediante la propagación de ondas que recorren su cuerpo desde la cola hasta la cabeza. En este artículo se describe la implementación de un algoritmo de locomoción para un robot ápodo de 8 módulos en tres plataformas con procesadores empotrados: MicroBlaze, PowerPC y LEON2. El objetivo es diseñar una unidad de control para un robot autónomo que funcione en tiempo real. El parámetro clave a optimizar es el tiempo para generar una nueva secuencia de locomoción, que es función del número de articulaciones del robot. Los resultados muestran que una unidad de punto flotante es necesaria para garantizar que este tiempo esté por debajo de 2 segundos. El rendimiento logrado usando un LEON2 con una unidad de punto flotante es 40 veces mejor que sin ella, empleando sólo un 6 % más de recursos.

1. Introducción

Los robots modulares y reconfigurables ofrecen la promesa de una mayor versatilidad, robustez y bajo coste[1]. Están compuestos de módulos capaces de unirse y separarse entre ellos, cambiando la forma del robot. Conviene aclarar que en este contexto, la palabra "reconfigurable" se refiere a la habilidad del robot para cambiar su forma. En los últimos años, el número de robots que siguen este enfoque ha crecido considerablemente[2][3][4][5].

Uno de los más avanzados es Polybot[1][6], diseñado en el PARC (Palo Alto Research Center). Puede adoptar múltiples formas y moverse de diferentes maneras. Por ejemplo, puede desplazarse como una rueda, despúes transformarse en una serpiente, con movimiento sinusoidal y finalmente convertirse en una araña de cuatro patas. Actualmente están diseñando la tercera generación de módulos[7]. Cada uno de ellos tiene su propio procesador PowerPC 555 en un chip.

Un paso más en la versatilidad es el uso de una FPGA en vez de un chip convencional. Ofrece al diseñador la posibilidad de implementar nuevas arquitecturas, algoritmos de control más rápidos o incluso modificar dinámicamente el *hardware* para adaptarlo a una nueva situación. En resumen, los robots modulares y reconfigurables controlados por FPGA no sólo pueden cambiar su forma, sino también su *hardware*, con lo que aumentan todavía más su versatilidad.

Una implementación de la locomoción de un robot ápodo usando FPGA se realizó con éxito[8]. Se utilizó el procesador MicroBlaze[9] para la ejecución del algoritmo y se añadió un *hardware* específico para posicionar los servos.

En este artículo se evalúa el algoritmo de locomoción para un robot ápodo de 8 módulos (figura 1) en diferentes procesadores empotrados en FPGA: MicroBlaze, PowerPC[10] y LEON2[11]. El tiempo que tarda el algoritmo en completar la generación del movimiento se calcula en función del número de nodos del robot, obteniéndose información sobre la escalabilidad. Estos resultados experimentales se utilizarán en los trabajos futuros para seleccionar las arquitecturas que mejor se ajusten a una determinada aplicación.

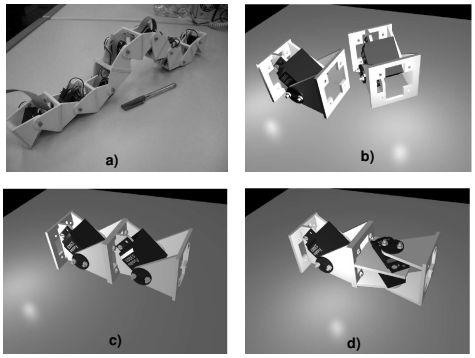

Figura 1: **a)** El robot ápodo "Cube Revolutions", formado por 8 módulos iguales, conectados en fase. **b)** Modelo 3D de dos módulos Y1. **c)** Dos módulos Y1 conectados en fase. **d)** Dos módulos Y1 conectados en desfase. Uno rota paralelamente al suelo y el otro perpendicular.

La organización del artículo es la siguiente. Primero se introduce brevemente el robot ápodo. Después se describen los detalles del algoritmo para la generación de movimiento y finalmente se muestran los resultados de la implementación.

2. El robot ápodo "Cube Revolutions"

El prototipo del robot ápodo, *Cube Revolutions*, se muestra en la figura 1a. Está compuesto por 8 módulos iguales, conectados en fase, por lo que sólo se desplaza en línea recta, hacia adelante o hacia atrás. La primera generación de los módulos creados, llamados módulos Y1[12], están hechos de PVC y tienen un único grado de libertad, actuado por un servo.

Los módulos Y1 se pueden conectar con dos orientaciones diferentes. Una es la conexión en fase, en la cual los dos módulos tienen la misma orientación, perpendicular al plano de apoyo (figura 1c). La otra es la conexión desfasada, en la que un módulo rota paralelo al suelo y el otro perpendicular (figura 1d).

El algoritmo de locomoción evaluado en este artículo ha sido diseñado para robots con conexión en fase como "Cube Revolutions", descrito en más detalle en [8]. La conexión desfasada se utiliza para construir robots capaces de moverse en cualquier dirección sobre el suelo y será estudiada en trabajos futuros.

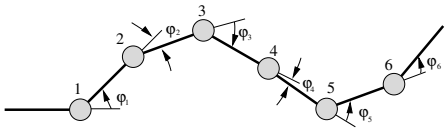

Figura 2: Vector de posición angular en el instante t_i para un robot compuesto por 6 articulaciones: $\overrightarrow{\varphi(t_i)} = (\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6)$

3. Algoritmo de locomoción

Se emplea un modelo de propagación de ondas para los cálculos, construyéndose las tablas de control de movimiento (*gait control tables*) a partir de los parámetros de la onda: amplitud, forma de la onda, longitud de onda y frecuencia. Estas tablas almacenan la posición de las articulaciones en los diferentes instantes. Cada fila de la tabla contiene la posición instantánea de las articulaciones y determina, por tanto, la forma del robot en ese instante. La matriz completa especifica la evolución en el tiempo de la forma que va adoptando el robot.

Sólo un subconjunto de todas las matrices posibles hacen que el robot se desplace. Mediante el algoritmo de locomoción, se generan tablas de control correctas, que permiten que el robot se mueva hacia adelante y hacia atrás.

La forma del robot en un instante t_i está determinada por su vector de posición angular $\overrightarrow{\varphi(t_i)} = (\varphi_1, \varphi_2, ..., \varphi_n)$, donde φ_j con $j \in \{1...n\}$, es el ángulo entre los dos segmentos de la articulación jth y n es el número total de módulos que componen el robot.

La figura 2 muestra un robot de seis articulaciones y su vector de posición angular para el instante t_i. La tabla de control es una matriz de mxn, cuyas filas son los vectores de posición angular en diferentes instantes: $\overrightarrow{\varphi(t_0)}, \overrightarrow{\varphi(t_1)}, ..., \overrightarrow{\varphi(t_m)}$ y m es el número total

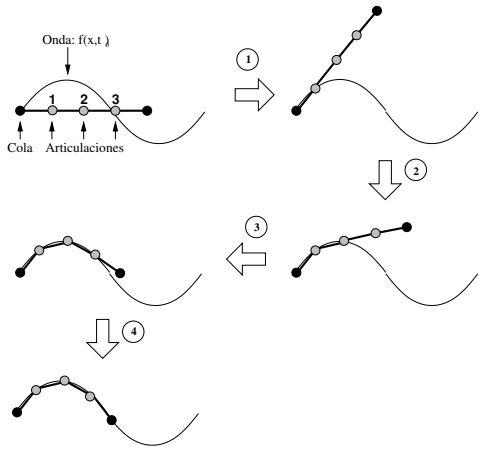

Figura 3: Pasos realizados para el cálculo del vector de posición angular $\overrightarrow{\varphi(t_0)} = (\varphi_1, \varphi_2, \varphi_3)$ a partir de una onda sinusoidal, para un robot de tres articulaciones.

de instantes en la evolución del movimiento. El valor de m depende de la resolución de tiempo necesaria para la aplicación. Un valor típico, usado en los experimentos de este artículo es de 20 (el movimiento del robot se describe con 20 instantes).

El algoritmo tiene dos partes. En la primera se calcula $\overrightarrow{\varphi(t_i)}$ a partir de la onda y en la segunda se crea la tabla.

3.1. Cálculo del vector de posición angular

Dada una onda en el instante t_i, $f(x, t_i)$, el problema es calcular el vector $\overrightarrow{\varphi(t_i)}$ que hace que la articulación cumpla la ecuación de la onda. Si (x_j, y_j) son las coordenadas de la articulación j, el algoritmo debe encontrar el vector $\overrightarrow{\varphi(t_i)}$ que satisfaga la ecuación $y_j = f(x_j, t_i)\ \forall j \in \{1...n\}$, esto es, que todas las articulaciones se encuentren sobre la onda.

El algoritmo utiliza un enfoque geométrico, basado en la rotación de puntos 2D. Un ejemplo aplicado a un robot de tres articulaciones se muestra en la figura 3. Inicialmente, $\overrightarrow{\varphi(t_0)} = (0, 0, 0)$ y la cola del robot está situada en el origen. El primer paso es la rotación del robot respecto de la cola hasta que la articulación 1 esté sobre la onda ($y_1 = f(x_1, t_0)$). Esto se realiza iterativamente, rotando un incremento angular ($\triangle\varphi$) y evaluando el error (ϵ). Después, se rota la articulación 1 hasta que la articulación 2 satisfaga la ecuación de la onda. Transcurridas cuatro iteraciones, se puede decir que el "robot se ajusta a la onda". Todas las articulaciones satisfacen la ecuación de la onda, con un error ϵ. En general, si el robot tiene n articulaciones, serán necesarias $n + 1$ iteraciones. La parte central de este algoritmo es la rotación de puntos en 2D, por tanto, se emplean las funciones seno, coseno y arco tangente.

3.2. Generación de tablas de control del movimiento

Para generar la tabla de control de movimiento, son necesarios el periodo T de la onda y el parámetro m. Se toma una muestra de la onda cada $\triangle t = \frac{T}{m}$ unidades de tiempo, en los instantes $t_0 = 0$, $t_1 = \frac{T}{m}$, $t_2 = \frac{T}{m}$, ..., $t_{m-1} = \frac{(m-1)T}{m}$.

La figura 4 muestra un ejemplo de los pasos necesarios para obtener las dos primeras filas de la tabla: $\overrightarrow{\varphi(t_0)}$ y $\overrightarrow{\varphi(t_1)}$, aplicado a un robot de seis articulaciones. Comenzando con el robot situado sobre el eje x y una onda sinusoidal en el instante t_0, el vector $\overrightarrow{\varphi(t_0)}$ se calcula "ajustando el robot a la onda", como se explicó en el apartado 3.1. Después se incrementa el tiempo obteniéndose una nueva onda desplazada y finalmente se ajusta el robot a esta nueva onda, obteniéndose $\overrightarrow{\varphi(t_1)}$. Repitiendo sucesivamente los pasos 1 y 2, se obtienen los m vectores de posición angular que componen la tabla.

El pseudo código se muestra en el algoritmo 1. Los parámetros de entrada son:

- $f(x, t)$: Ecuación de la onda
- n : Número total de articulaciones
- m: Número de muestras temporales
- T: Periodo de la onda
- ε: Máximo error permitido en la aproximación

Algorithm 1 Una versión simplificada del algoritmo de ajuste

Entradas: f,n,m,T,ε
Salidas: GCM (Matriz m x n de control del movimiento)

Ajustar_gusano_a_la_onda(f,n,m,T,ε)
Begin
  // Iterar sobre el tiempo
  for i=0 to m-1
    $t_i = i\frac{T}{m}$
    // Iterar sobre las articulaciones del robot
    for j=0 to n
      // Rotar la articulación j hacia la onda, hasta que $y_{j+1} = f(x_{j+1}, t_i) \pm \varepsilon$
      Do
        // Rotar la articulación j un ángulo $\triangle\varphi$
        Rotar(j,$\triangle\varphi$)
        $\varphi_j = \varphi_j + \triangle\varphi$
        $Error = |f(x_{j+1}, t_i) - y_{j+1}|$
      while (Error>ε)
      GCM[i][j]=φ_j
    Next j
  Next i
End

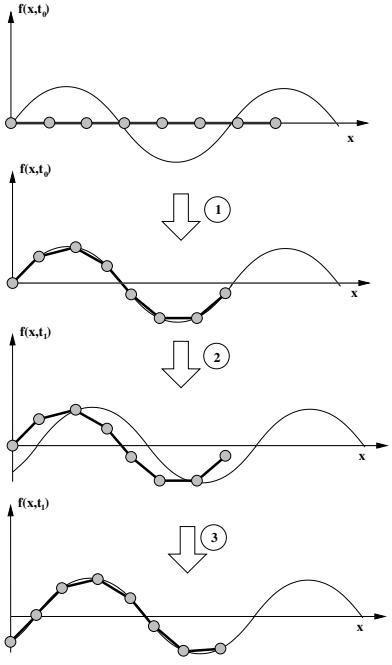

Figura 4: Ejemplo del algoritmo usado para generar las tablas de control. Se calculan las dos primeras filas de la tabla.

Punto flotante		Enteros		Otras	
Mult:	25.7 %	Mult:	20 %	atan	1.0 %
Sum.:	23 %	Restas:	1.23 %	sin	0.79 %
Div.:	22 %			sqrt	0.68 %
				cos	0.49 %
Total	71.4 %	Total	21.23 %	Total:	2.96 %

Cuadro 1: Análisis (*profile*) de la ejecución del algoritmo en MicroBlaze

Arq.	Procesador.	Frec.	FPGA
1	LEON	25 Mhz	Virtex XC2000E
2	LEON+FPU		
3	MicroBlaze	50Mhz	
4a	PowerPC		Virtex II Pro
4b		100Mhz	

Cuadro 2: Arquitecturas usadas para la evaluación del algoritmo

Procesador	Slices	BRAM
MicroBlaze	1321 (6 %)	74 (46 %)
LEON	4883 (25 %)	43 (26 %)
LEON+Meiko FPU	6064 (31 %)	40 (25 %)

Cuadro 3: Resultados de la implementación para las arquitecturas 1,2 y 3.

La salida es la matriz *mxn* de control del movimiento. El algoritmo se puede emplear para cualquier tipo de onda, sin embargo, para las pruebas de locomoción se han empleado ondas sinusoidales.

4. Implementación en procesadores embebidos en FPGA

4.1. Análisis de las operaciones del algoritmo

El algoritmo completo se ha implementado en C usando operaciones en coma flotante de doble precisión (*double*). Esto permite portarlo directamente a las diferentes arquitecturas sin tener que modificar el código fuente. El análisis (*profile*) del algoritmo en MicroBlaze muestra que el 71,4 % del tiempo de ejecución se gasta en operaciones de coma flotante (cuadro 1). El 21,23 % se dedica a operaciones con enteros y el 7,37 % restante en otras operaciones, incluidas las trigonométricas.

El *profile* sugiere el uso de una unidad en coma flotante (FPU) para mejorar el tiempo de ejecución.

4.2. Ejecución del algoritmo sobre diferentes arquitecturas

El cuadro 2 muestra las cuatro arquitecturas empleadas para evaluar el algoritmo. Se ejecuta el código en tres procesadores en FPGA: LEON2, MicroBlaze y el *core* PowerPC embebido en la FPGA Xilinx Virtex II Pro. (El PowerPC es el procesador empleado en PolyBot G3, el robot modular reconfigurable más avanzado desarrollado en el PARC). Los procesadores *software* (*Soft Core Processors*, SCPs) han sido implementados usando arquitecturas similares: sin unidades *hardware* de multiplicación y división y con similares cachés para instrucciones y datos. En primer lugar, la arquitectura 1 incluye un procesador LEON2. En la arquitectura 2 se añade la unidad en coma flotante Meiko [11]. La tercera incluye el procesador Xilinx MicroBlaze y finalmente, la arquitectura 4 consiste en un procesador embebido PowerPC. Las arquitecturas de la 1 a la 3 han sido implementadas en *hardware* sobre la plataforma RC1000 de Celoxica que incluye una FPGA Xilinx Virtex-E. Para la arquitectura 4 se ha utilizado la plataforma Alpha Data ADM-XPL que dispone de una Virtex II Pro.

5. Resultados

5.1. Resultados de la síntesis

Las herramientas empleadas para la síntesis han sido XST de Xilinx para Microblaze y Synplify Pro para LEON2. La implementación del PowerPC se realizado con el *Embedded Development Kit* (EDK) de Xilix.

Los resultados se muestran en el cuadro 3. El procesador MicroBlaze está optimizado pa-

	Tiempo de ejecución del algoritmo (segundos)				
n	**Arq 1**	**Arq 2**	**Arq 3**	**Arq 4a**	**Arq 4b**
4	7,980	0,199	5,865	2,552	1,276
8	16,116	0,392	12,065	5,105	2,553
12	26,794	0,632	20,346	8,445	4,225
16	40,069	0,941	30,652	12,586	6,297
20	55,866	1,295	43,070	17,504	8,758
24	74,341	1,707	57,677	23,259	11,639
28	95,585	2,179	74,452	29,871	14,948
32	119,404	2,706	93,293	37,277	18,655

Cuadro 4: Tiempo de cálculo del movimiento, GRT, como función del número de articulaciones, medido para las cuatro arquitecturas. El parámetro m se ha fijado a 20.

ra las FPGAs de Xilinx, por lo que ocupa casi un 20 % menos de área que el LEON2. La velocidad máxima es de 50MHz, el doble que la del LEON2, como se muestra en el cuadro 2. Sin embargo, la adición de una unidad de coma flotante al LEON2 sólo incrementa en un 6 % el área pero disminuye drásticamente el tiempo de ejecución del algoritmo.

5.2. Medida del tiempo de ejecución

La ejecución del algoritmo determina el tiempo que el robot necesita para generar una nueva secuencia de movimiento. Este tiempo se denomina GRT (*Gait Recalculation Time).* Para que el robot pueda cambiar rápidamente de un tipo de movimiento a otro, se requiere un GRT bajo, del orden de 2 segundos.

La tabla 4 muestra el GRT para las cuatro arquitecturas evaluadas en función del número de articulaciones. Estos resultados se representan en la figura 5a y el rendimiento de las arquitecturas en la figura 5b, suponiendo que todas funcionasen a 50MHz.

Como era de esperar, el GRT se incrementa con el número de articulaciones del robot. A mayor número de articulaciones, mayor el tiempo que tarda el robot en recalcular la nueva secuencia de movimiento. El PowerPC obtiene unos resultados mucho mejores que el MicroBlaze y LEON2 (Arquitecturas 1 y 3), ya que al tratarse de un *hard core*, dispone de

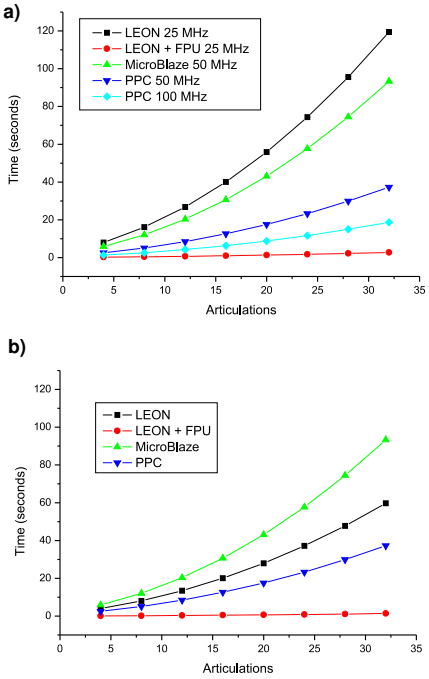

Figura 5: **a)** Comparación del GRT para las cuatro arquitecturas evaluadas, como función del número total de articulaciones. **b)** Resultados normalizados suponiendo una frecuencia de reloj de 50MHz para todas las arquitecturas.

unidades *hardware* específicas para algunas de las operaciones. El resultado más significativo se obtiene con la arquitectura 2 (LEON2 + FPU). Para un robot gusano de ocho articulaciones, el rendimiento obtenido por esta arquitectura es 40 veces superior al de LEON2 sin FPU, empleando sólo un 6 % más de recursos. Este rendimiento es 6,5 veces superior al obtenido por el PowerPC a 100 MHz.

De todas las arquitecturas evaluadas, sólo con la segunda (LEON2 + FPU) se consigue un GRT menor a 2 segundos.

6. Conclusiones y trabajos futuros

La locomoción de un robot gusano puede ser realizada por medio de la propagación de ondas a través del cuerpo del robot. El algoritmo empleado, genera las tablas de control para la onda elegida. El GRT es la clave para obtener un robot autónomo con capacidad de reacción en tiempo real.

El algoritmo ha sido satisfactoriamente evaluado y ejecutado en tres procesadores embebidos en FPGA: LEON2, MicroBlaze y PowerPC. El GRT se ha medido para cuatro arquitecturas, en función del número de articulaciones totales. Los resultados muestras que para lograr que esté por debajo de 2 segundos es necesario el empleo de una unidad en coma flotante. Un procesador LEON2 a 25 MHz con una FPU es hasta un orden de magnitud más rápido que un procesador PowerPC a 100 MHz. Esta es una de las ventajas de las FPGAs frente a los procesadores tradicionales: los diseñadores e investigadores pueden mejorar el robot por medio de cambios en la arquitectura y añadir *cores hardware* a medida.

El procesador LEON2 con una FPU es una buena opción cuando se requiere un GRT bajo. En aplicaciones no críticas el uso del procesador MicroBlaze permite disponer de un 75 % más de área, dejando este porcentaje libre para la implementación de nuevos *cores hardware*. La nueva versión de MicroBlaze incluye una FPU, que será evaluada en trabajos futuros.

El prototipo actual de robot gusano, *Cube Revolutions*, se mueve únicamente en línea recta. El movimiento en un plano será estudiado en trabajos posteriores. El mismo algoritmo de locomoción puede ser usado, calculando las tablas de control para dos ondas diferentes: una para las articulaciones en el plano paralelo al suelo y la otra para las articulaciones en el plano perpendicular. La locomoción se consigue como composición de las dos ondas. Además, se empleará una nueva alternativa para la generación de las tablas de control mediante el uso de algoritmos genéticos. El empleo de FPGAs permitirá el diseño de unidades *hardware* específicas para mejorar el rendimiento, e incrementará notablemente la versatilidad de los robots modulares y reconfigurables.

Referencias

[1] Mark Yim, Ying Zhang & David Duff, Xerox Palo Alto Research Center (PARC), "Modular Robots". IEEE Spectrum Magazine. Febrero 2002.

[2] 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.

[3] 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.

[4] 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.

[5] 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, pp.432-439, May 1998.

[6] D. Duff, M. Yim, K. Roufas,"Evolution of PolyBot: A Modular Reconfigurable Robot", Proc. of the Harmonic Drive Intl. Symposium, Nagano, Japan, Nov.

2001, and Proc. of COE/Super-Mechano-Systems Workshop, Tokyo, Japan, Nov. 2001.

[7] 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.

[8] González-Gómez J., Aguayo E., Boemo E.,"Locomotion of a Modular Worm-like Robot using a FPGA-based embedded MicroBlaze Soft-processor".Proc. of 7th International Conference on Climbing and Walking Robots, Madrid, Spain, Sep. 2004.

[9] Xilinx Inc. "MicroBlaze Processor Reference Guide, Embedded Development Kit. Version 6.2"

[10] Xilinx Inc. "PowerPC Processor Reference Guide, Embedded Development Kit. Version 6.2"

[11] Gaisler Research, "http://www.gaisler.com". (Consulta: Mayo-2005)

[12] Módulos Y1. Página web. http://www.iearobotics.com/personal/juan/doctorado/Modulos-Y1/modulos-y1.html. (Consulta: Mayo-2005)