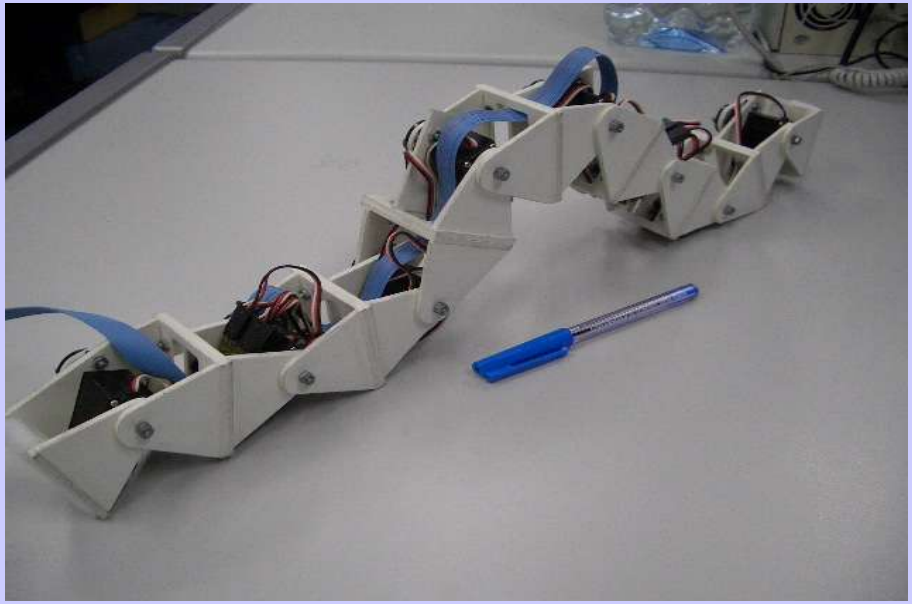
Evaluación de un Algoritmo de Locomoción de Robots Ápodos en Diferentes Procesadores Embebidos en una FPGA



Juan González-Gómez, Ivan González,
Francisco J. Gómez-Arribas y Eduardo Boemo

Escuela Politécnica Superior
Universidad Autónoma de Madrid

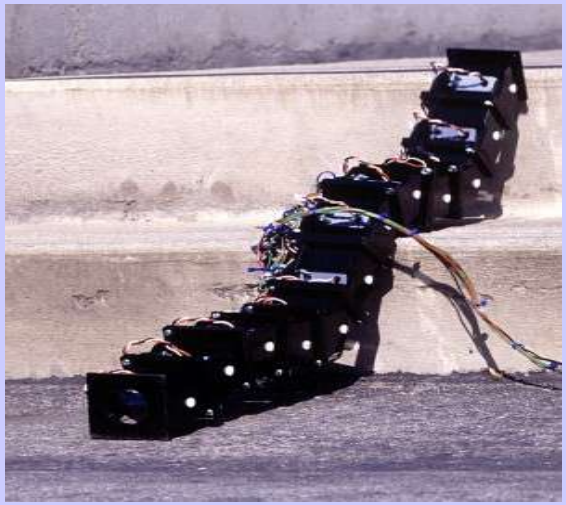
V Jornadas sobre Computación Reconfigurable y Aplicaciones, JCRA.
Granada, Septiembre 2005

Índice

Introducción (I)

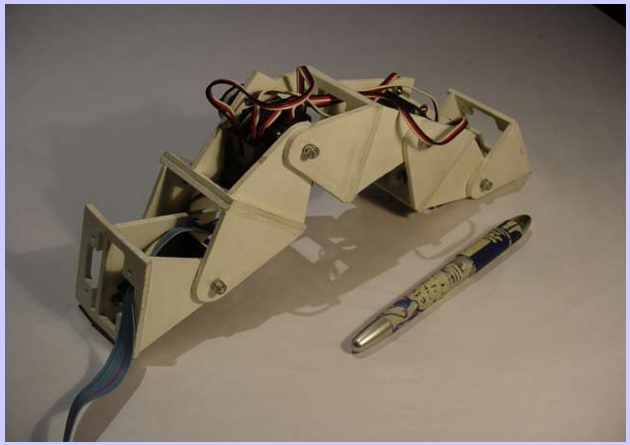
- **Robótica Modular Reconfigurable:**
  - Construcción de robots a partir de módulos sencillos
  - Los robots pueden "reconfigurarse físicamente", cambiando su forma para adaptarse a diferentes terrenos.
  - El robot más avanzado de este tipo es **PolyBot**, desarrollado en el **PARC** (Palo Alto Research Center)
  - Cada módulo dispone de un **PowerPC 555**

Una de nuestras **líneas de investigación** es estudiar la viabilidad de utilizar FPGA's en el campo de la robótica modular reconfigurable, sustituyendo a los procesadores convencionales y aplicar las técnicas de **codiseño hardware / software** y **reconfiguración dinámica**



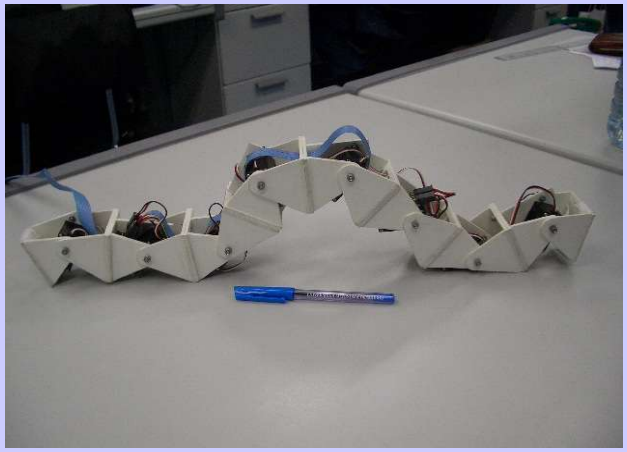
Introducción (II)
Trabajos previos

- Robot ápodo **Cube Reloaded**



- Construido a partir de 4 módulos en fase
- Estudiadas diferentes alternativas para el hardware de control:
  - Utilizando microcontroladores (6811, PIC16F876)
  - Utilizando FPGAs (**JCRA 2003**)

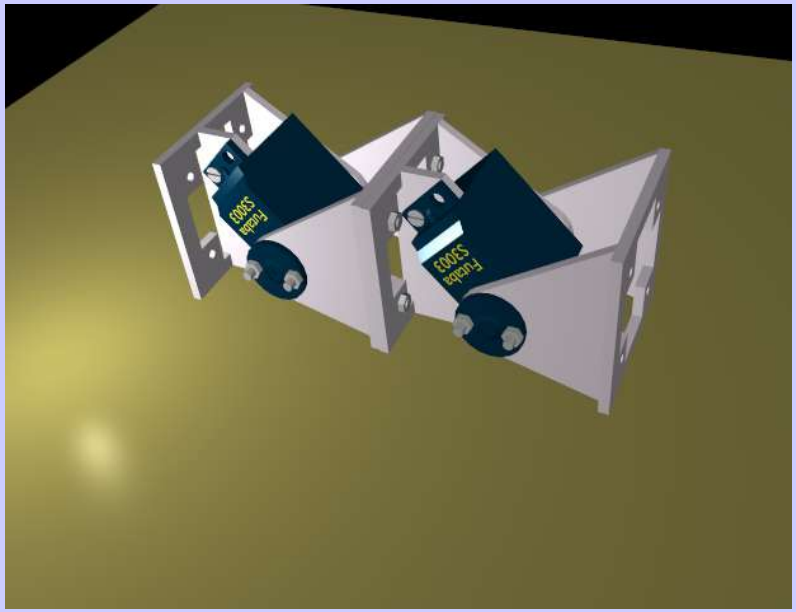
- Robot ápodo **Cube Revolutions**

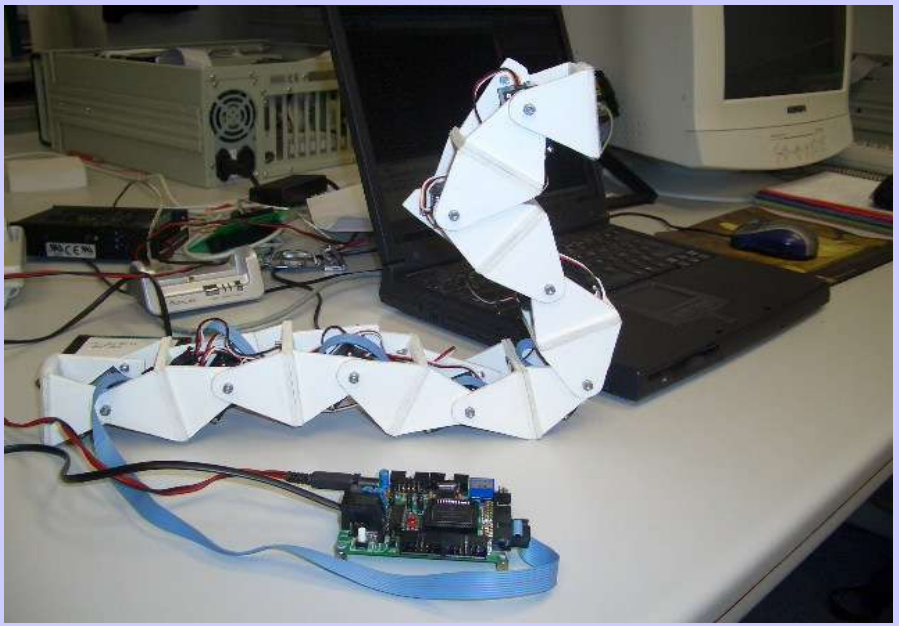


- 8 módulos en fase
- Restringido a locomoción en 1 Dimensión
- Nuevas formas de locomoción
- Controlado con el MicroBlaze (**JCRA 2004**)

Introducción (III)
Objetivos de este trabajo

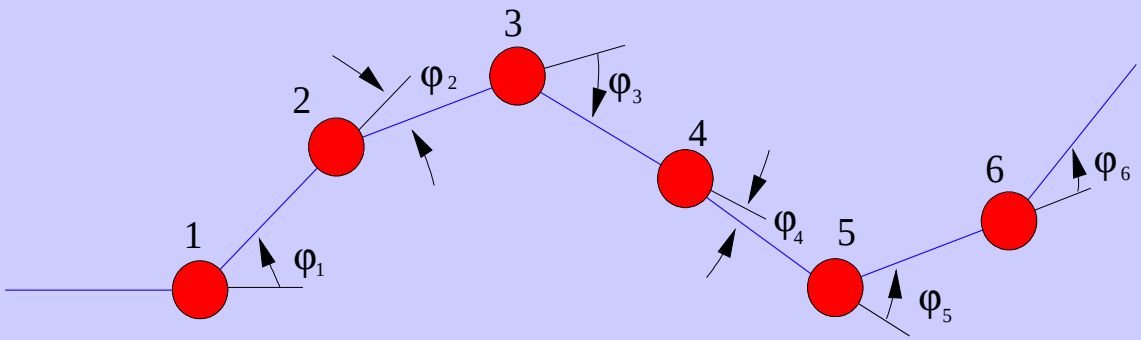
- Evaluar otras alternativas para la implementación del **algoritmo del locomoción** de Cube Revolutions y compararlas con los resultados de obtenidos con el MicroBlaze.
- Determinar cual es la mejor plataforma para el desarrollo de las siguientes versiones del robot ápodo.





Conceptos previos (I)
Vector de posición angular

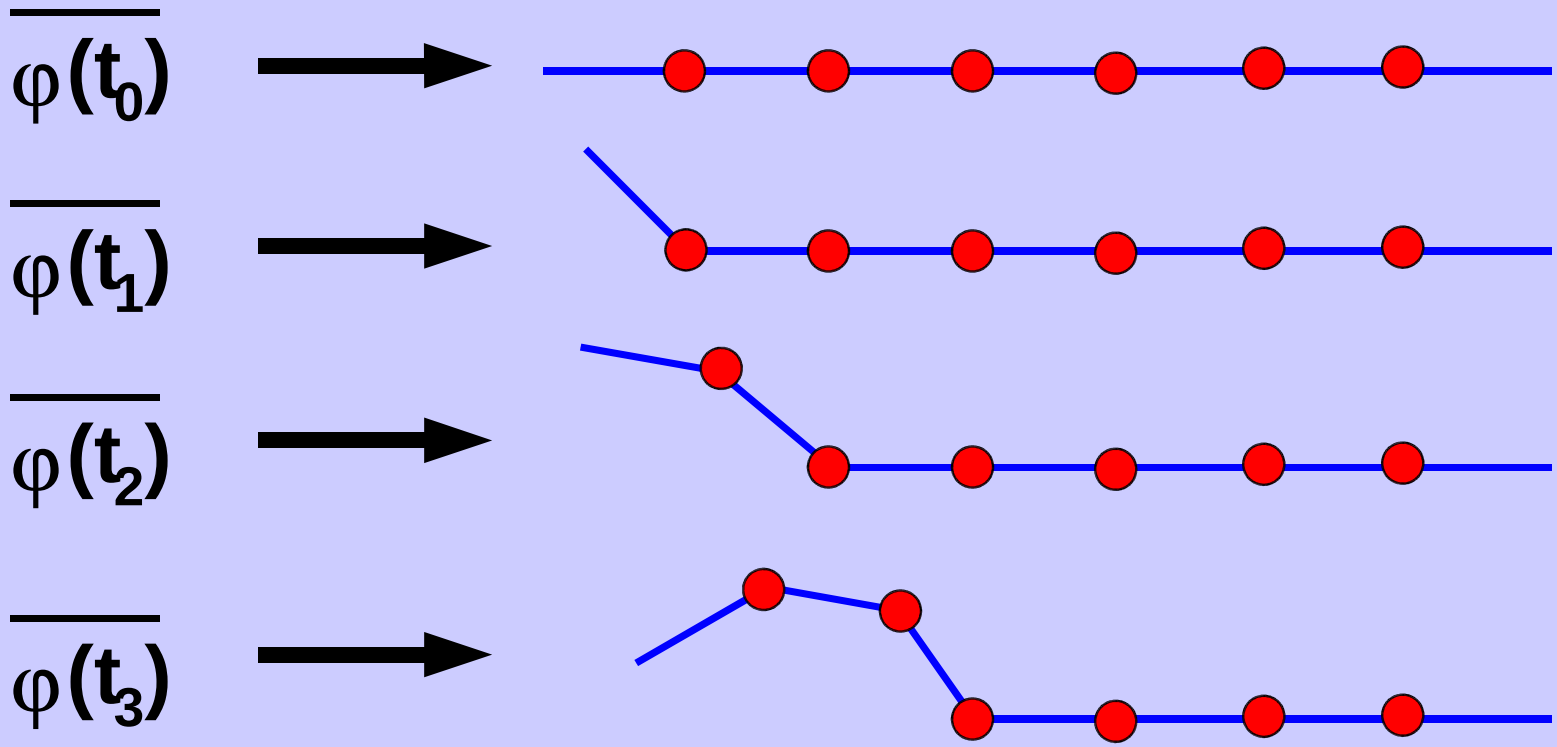
- La forma del robot en un instante está determinada por los ángulos de cada articulación.
- El **vector de posición angular** contiene estos ángulos
- Ej. Para un robot gusano de 6 articulaciones:



- Vector de posición angular: $\overline{\varphi(t_i)} = [\ \varphi_1\ \varphi_2\ \varphi_3\ \varphi_4\ \varphi_5\ \varphi_6\ ]$

Conceptos previos (II)
Matriz de movimiento

- Secuencia de movimiento: Sucesión de vectores de posición angular



- Se puede representar como una **matriz de mxn**, donde cada fila es un vector de posición angular

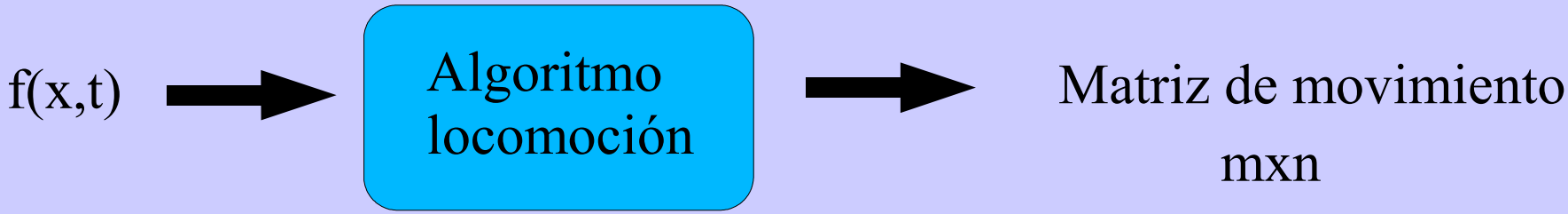
Conceptos previos (III)
Matriz de movimiento

- **n** es el número de articulaciones del robot
- **m** es la cantidad de posiciones intermedias que definen la secuencia (el número de "fotogramas")
- Ejemplo de una matriz de movimiento para un gusano de 8 articulaciones:

1	2	3	4	5	6	7	8
[-19	-9	24	-4	-22	17	12	-18]
[-17	-14	21	4	-25	11	19	-15]
[-12	-20	18	11	-25	3	23	-7]
			.				
			.				
[21	0	-23	12	17	-21	-6	22]
[19	9	-24	4	22	-17	-12	18]

Algoritmo de locomoción (I)

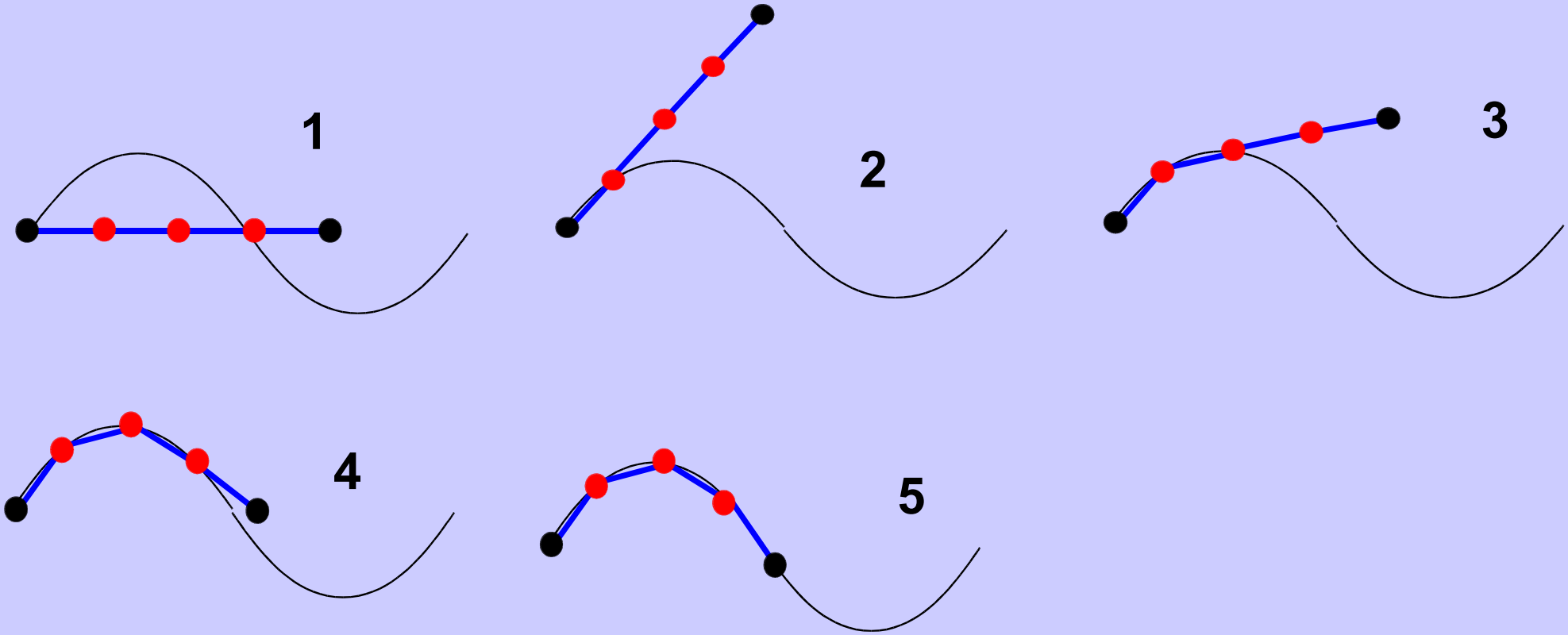
- El algoritmo de locomoción empleado calcula las matrices de movimiento a partir de la ecuación de una onda *f(x,t)*



- Este algoritmo tiene un enfoque geométrico
- El movimiento del gusano se consigue propagando las ondas desde la cola hasta la cabeza

Algoritmo de locomoción (II)

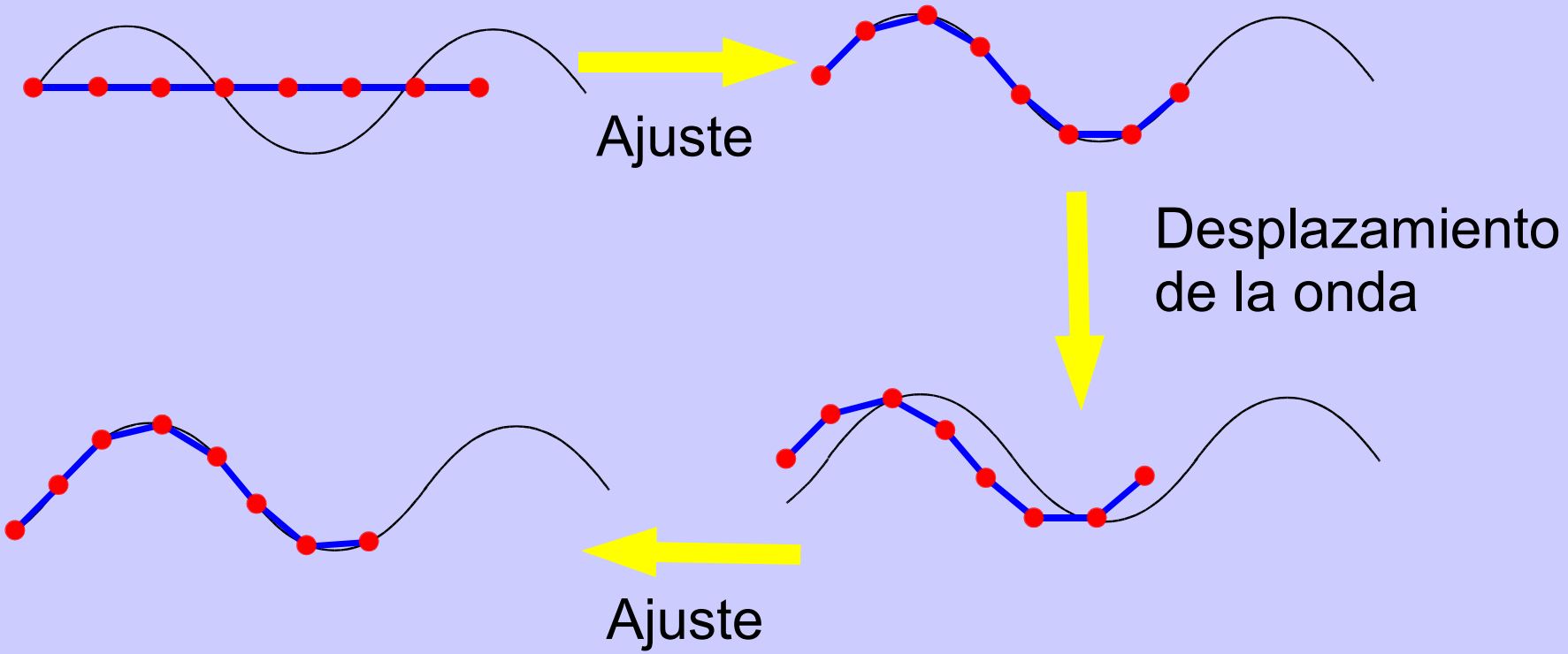
- Para obtener el vector de posición angular en un instante, se van rotando las articulaciones hasta que cumplan la ecuación de la onda:



- Decimos que el gusano se "ajusta" a la onda

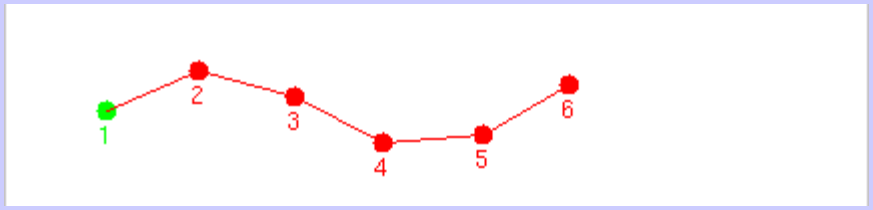
Algoritmo de locomoción (III)

- Para obtener la matriz de movimiento completa se va desplazando la onda y se va "ajustando" el gusano a ella. En cada "ajuste" obtenemos una fila de la matriz.



Algoritmo de locomoción (IV)

- **RESULTADO:** Tras iterar m veces se obtiene la matriz de movimiento
- Si se visualizase una matriz de movimiento en la pantalla se obtendría algo como esto:



- Si se utiliza otra onda diferente, se obtiene un tipo de locomoción distinto.
- El algoritmo es independiente de la onda empleada

Implementación en procesadores embebidos en FPGA

- Algoritmo de locomoción está implementado en C
- Coma flotante de doble precisión (tipo double)
- Análisis de las operaciones (profile):

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

- El 71.4% del tiempo se emplea en operaciones de punto flotante.

Implementación en procesadores embebidos en FPGA

- Arquitecturas empleadas para evaluar el algoritmo:

<table>
<tr><th>Arquitectura</th><th>Procesador</th><th>Frecuencia</th><th>FPGA</th></tr>
<tr><td>1</td><td>LEON</td><td rowspan="2">25Mhz</td><td rowspan="3">Virtex XC2000E</td></tr>
<tr><td>2</td><td>LEON + FPU</td></tr>
<tr><td>3</td><td>MicroBlaze</td><td rowspan="2">50Mhz</td></tr>
<tr><td>4a</td><td rowspan="2">PowerPC</td><td rowspan="2">Virtex II Pro</td></tr>
<tr><td>4b</td><td>100Mhz</td></tr>
</table>

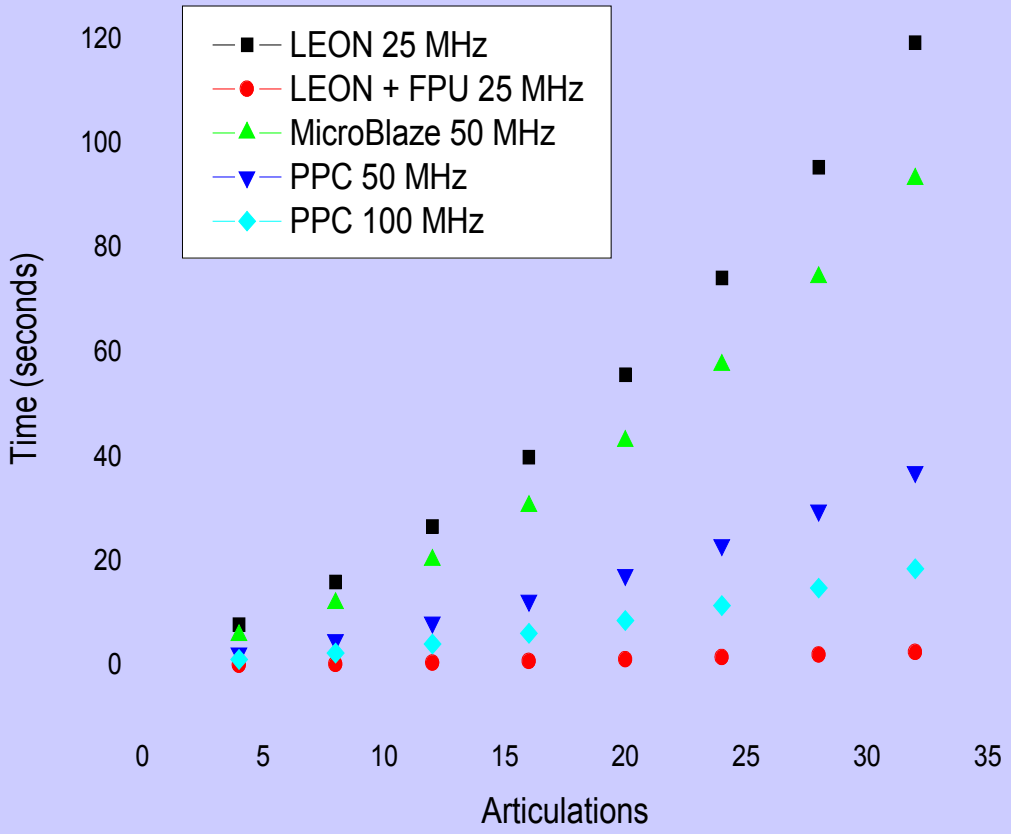
Resultados (I)
Síntesis

- Resultados para arquitecturas 1, 2 y 3
- Herramientas: XST, EDK de Xilinx y Symplify Pro

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

- MicroBlaze necesita un 20% menos de área que LEON2
- Con MicroBlaze la velocidad es de 50Mhz mientras que con LEON2 de 25Mhz

Resultados (II)
Tiempo de ejecución

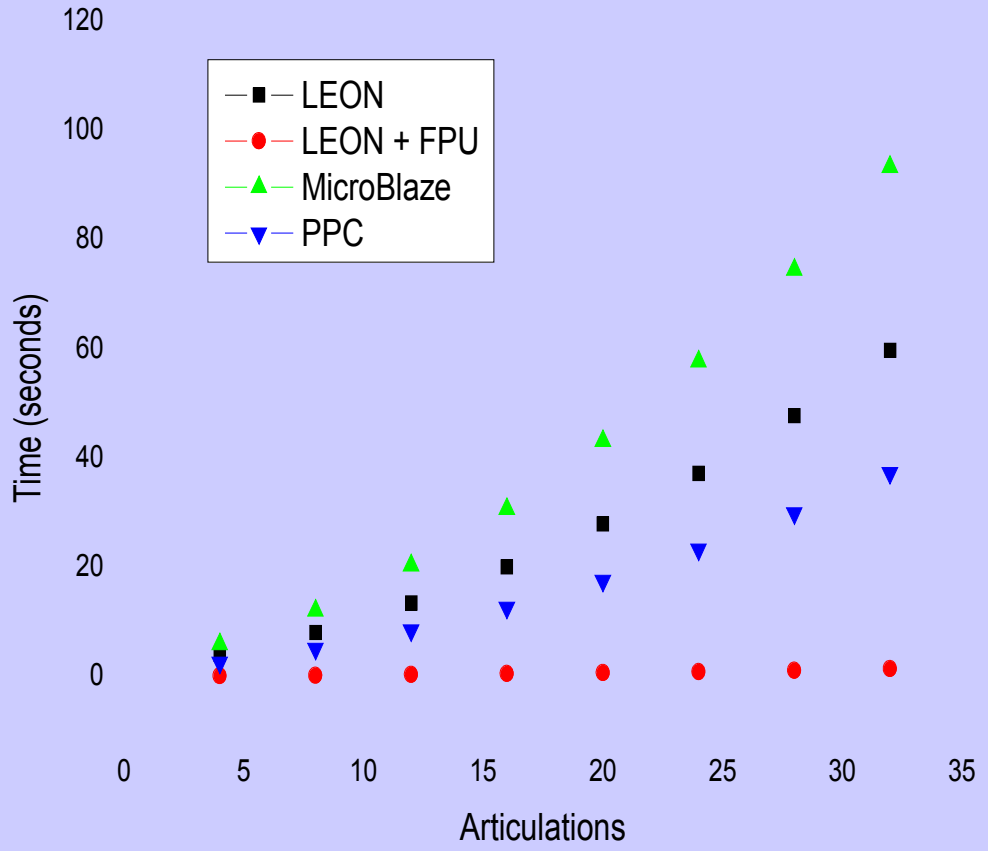


- Representación en función del número de articulaciones
- El tiempo crece con el número de articulaciones (como era de esperar)
- Power PC ofrece mejores resultados que LEON2 y MicroBlaze (está hard cored)

- Sin embargo, al añadir una FPU al LEON2, sólo se incrementa el área en un 6% sin embargo los tiempos se reducen drásticamente.

Resultados (II)
Tiempo de ejecución

- Trendimiento de las arquitecturas (normalizado a 50Mhz):



- El peor rendimiento se obtiene con MicroBlaze
- El rendimiento del LEON con FPU es muy superior al del resto

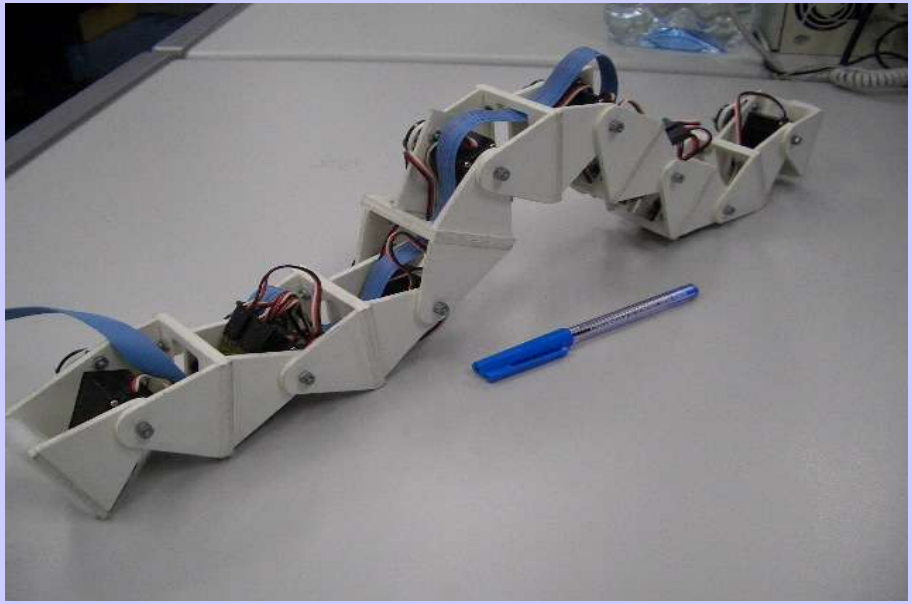
Conclusiones

- El algoritmo de locomoción se ha ejecutado en los procesadores: **MicroBlaze**, **LEON2** y **PowerPC** en FPGA
- Para tener un tiempo de ejecución por debajo de 2 segundos, la única alternativa es utilizar una **unidad FPU**
- Un LEON2 a 25MHz con FPU es un orden de magnitud más rápido que un PowerPC a 100MHz.
- Esta es una de las ventajas de la utilización de las FPGA's para el diseño de robots modulares, en vez de procesadores convencionales: Modificando la arquitectura se pueden conseguir grandes aumentos del rendimiento.

Trabajo futuro

- Aplicación al movimiento del gusano en un plano. Se podrá utilizar el mismo algoritmo, calculando dos matrices, una para las articulaciones que se mueven paralelamente al suelo y otra para las que lo hacen perpendicularmente.
- Implementación de un core para el cálculo de las matrices de movimiento
- Generación de las matrices de movimiento óptimas mediante algoritmos genéticos.

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



Juan González-Gómez, Ivan González,
Francisco J. Gómez-Arribas y Eduardo Boemo

Escuela Politécnica Superior
Universidad Autónoma de Madrid

V Jornadas sobre Computación Reconfigurable y Aplicaciones, JCRA.
Granada, Septiembre 2005