



**Repositorio Institucional de la Universidad Autónoma de Madrid**

<https://repositorio.uam.es>

Esta es la **versión de autor** de la comunicación de congreso publicada en:  
This is an **author produced version** of a paper published in:

Reconfigurable Computing: Architectures and Applications: Second International Workshop, ARC 2006, Delft, The Netherlands, March 1-3, 2006, Revised Selected Papers. Lecture Notes in Computer Science, Volumen 3985. Springer, 2006. 24-29.

**DOI:** [http://dx.doi.org/10.1007/11802839\\_4](http://dx.doi.org/10.1007/11802839_4)

**Copyright:** © 2006 Springer-Verlag

El acceso a la versión del editor puede requerir la suscripción del recurso  
Access to the published version may require subscription

# Evaluation of a locomotion algorithm for worm-like robots on FPGA-embedded processors

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

Computer Engineering School, Universidad Autonoma de Madrid, Spain  
{Juan.Gonzalez, Ivan.Gonzalez, Francisco.Gomez, Eduardo.Boemo}@uam.es

**Abstract.** The locomotion of worm-like robots, composed of a chain of equal linked modules, is achieved by means of wave propagation that traverse the body of the worm. In this paper, a locomotion algorithm designed for an eight modules worm-like robot has been successfully tested on three different FPGA-embedded processors: MicroBlaze, PowerPC and LEON2. The time the robot needs to generate a new motion wave, the gait recalculation time, is the key to achieve an autonomous robot with real-time reactions. Algorithm execution time for four different architectures, as a function of the total number of articulations of the robot, are presented. The results show that a huge improvement of the gait recalculation time can be achieved by using a float point unit. The performance achieved using the LEON2 with FPU is 40 times better than LEON2 without FPU, using only 6% of additional resources.

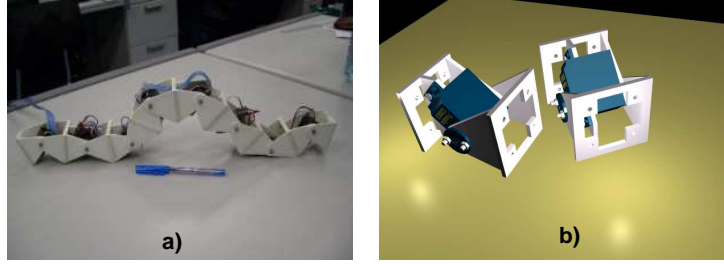
## 1 Introduction

Modular self-reconfigurable robots offer the promise of more versatility, robustness and low cost[1]. They are composed of modules, capable of attach and detach one to each other, changing the shape of the robot. In this context, the word “reconfigurable” means the ability of the robot to change its form, not a hardware reconfigurable system. In the last years, the number of robot following this approach has growth substantially[2].

One of the most advanced systems is Polybot[1][3], designed at Palo Alto Research Center (PARC). This robot has the capability to achieve different reconfigurations, such as moving as a wheel, using a rolling gait, then transforming itself into a snake and finally becoming a spider. Currently, the third generation of modules (G3) is being developed[4]. Each module has its own embedded PowerPC 555 processor with a traditional processor architecture.

An additional step on versatility is the use of FPGA technology instead of a conventional microprocessor chip. It gives the designer the possibility of implementing new architectures, faster control algorithms, or dynamically modify the hardware to adapt it to a new situation. In summary, modular reconfigurable robot controlled by a FPGA are not just able to change their shapes, but also their hardware, therefore, complete versatility can be achieved.

An implementation of a FPGA-based worm-like robot locomotion was successfully realized[5]. The Xilinx MicroBlaze[6] soft-processor was used for the algorithm execution and custom cores were added for servo positioning.



**Fig. 1. a)** “Cube revolutions” worm-like robot, composed of eight similar linked modules, connected in phase. **b)** A CAD rendering of two unconnected Y1 modules.

In this paper the locomotion algorithm for an eight modules worm-like robot (figure 1a) is evaluated in different FPGA embedded processors: MicroBlaze, PowerPC[7] and LEON2[8]. The time this algorithm takes to complete the movement generation is calculated as a function of the number of nodes of the robot, giving information about the scalability. This experimental results will be used in future work to select the architectures that fit best a particular application.

## 2 The worm-like robot

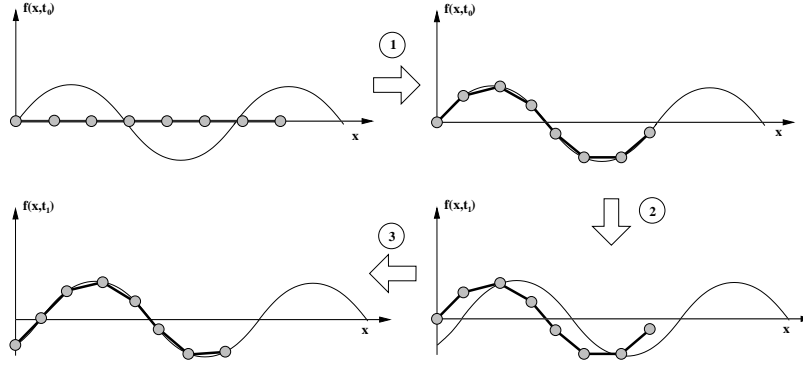
The prototype of the worm-like robot, called “Cube Revolutions”, is shown in figure 1a. It is composed of 8 similar linked modules, connected in phase, so that it can only move in a straight line, forward and backward. The first generation of the modules created, named Y1 (1b), were made of PVC and contains only one degree of freedom, actuated by a RC servo. More technical information can be found in [5].

## 3 Locomotion algorithm

The locomotion of the robot is achieved by means of gait control tables[1], pre-calculated data matrix that store the position of all the articulations at different time slots.

Each row of the table contains the position of the articulations at instant  $t_i$ , that is, the shape of the robot at  $t_i$ . The whole matrix determines the evolution of the shape of the robot in time. If it is well calculated, the robot will move correctly. In order to achieve locomotion, the controller reads the table, row by row, producing the PWM signals that actuate the servos.

The locomotion algorithm generates well-constructed gait control tables that allow the robot to move forward and backward. A wave propagation model is used for its calculation, building the tables from the parameters of the wave: amplitude, waveform, wavelength and frequency.



**Fig. 2.** An example of the algorithm used to generate the control tables. The first two rows of the gait control table are calculated

Figure 2 shows an example of how the algorithm calculate the first and second rows of the gait control table. It comprises two parts. First, the angles of the articulations are calculated by “fitting the worm to the wave”. Then, the wave is sifted (that is, the time is incremented) and the robot is fitted to the wave again. These steps are repeated until the wave has move a distance equal to the wavelength.

The algorithm takes a geometric approach and is based on rotations of 2D points, therefore, sine, cosine and arctan function are widely used.

## 4 Implementation on embedded FPGA processors

### 4.1 Algorithm operation analysis

The whole algorithm has been implemented in C language, using double precision floating point. The profiling analysis of the algorithm shows that the 71.4% of the execution time is spent in float point operations. The 21.23% is used for integers operations and the final 7.37% for other operations, including trigonometric ones.

The profile suggests the use of a float point unit (FPU) for improving the execution time.

### 4.2 Target architectures

Table 1 shows the four architectures used for the evaluation of the algorithm. Three FPGA-embedded processor has been tested: LEON2, Xilinx MicroBlaze and a PowerPC core embedded in the Xilinx Virtex II Pro FPGA. The PowerPC is the processor employed in PolyBot G3, the most advanced modular reconfigurable robots designed at PARC.

The soft core processors (SCP) have been implemented using similar architectural features: without hardware multiplier/divisor units and with similar data

Target architectures	1	2	3	4a	4b
Processor	LEON	LEON+ Meiko FPU	MicroBlaze	PowerPC	
Frequency	25 Mhz		50MHz	50Mhz	100Mhz
FPGA	Virtex XC2000E			Virtex II Pro	

**Table 1.** Architectures used for the evaluation of the algorithm

and instruction caches. Architectures 1 comprises only one LEON2 soft core processor (SCP). Architecture 2 adds the Meiko FPU[8]. The third architecture is a Xilinx MicroBlaze SCP. The final architecture consists of an embedded PowerPC core

Architectures 1 to 3 have been evaluated in hardware on the RC1000 development board from Celoxica that includes a Xilinx Virtex E FPGA. The architecture 4 has been implemented on a Alpha Data ADM-XPL board in a Virtex II Pro.

## 5 Results

Xilinx XST is used for the synthesis of MicroBlaze. Simplify Pro is used for the synthesis of the LEON2 processor. The reason why Simplify Pro is not used for the synthesis of the MicroBlaze processor is the fact that MicroBlaze processor is distributed as a parametrizable netlist, i.e. it is already synthesized. PowerPC implementation have been developed using the Xilinx Embedded Development Kit.

### 5.1 Synthesis results

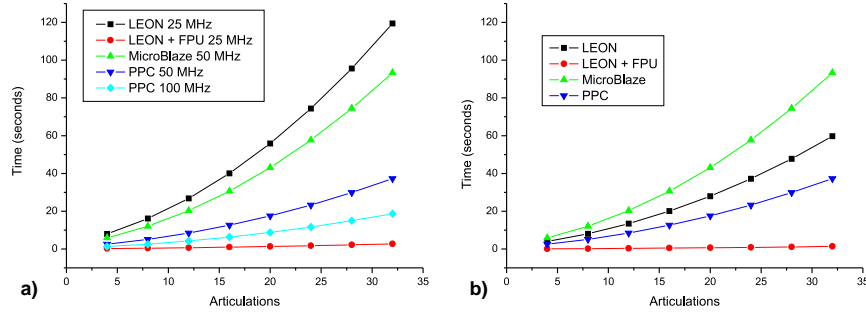
Processor	Slices	BRAM
MicroBlaze	1321 (6%)	74 (46%)
LEON	4883 (25%)	43 (26%)
LEON+Meiko FPU	6064 (31%)	40 (25%)

**Table 2.** Implementation results for architectures 1,2 and 3.

The results are shown in table 2. Since MicroBlaze processor is highly optimized for Xilinx FPGA circuit the resources used are lower than the resources used of the LEON processor. LEON2 are written not only for FPGA circuit so it is very difficult for a synthesis tool to synthesize LEON2 with the same low FPGA resource optimization as MicroBlaze. Also, as can be seen in table 1 the maximum clock frequency achieved for MicroBlaze is 50Mhz, whereas LEON2 runs at 25Mhz in the selected FPGA device.

The improved architecture LEON with FPU unit only suppose a 6% of additional resources

## 5.2 Execution time



**Fig. 3. a)** GRT comparison for the four architectures evaluated, as a function of the number of articulations. **b)** Normalized results supposing a 50MHz system clock frequency for all architectures.

The algorithm has been compiled for the different architectures and it is loaded from external memory and executed. The execution of the algorithm determines the time the robot needs to generate a new kind of movement. This time is called gait recalculation time (GRT). If a worm-like robot capable of having a fast reaction (a fast change in its kind of movement) is needed for a particular applications, a low GRT is required.

Figure 3a shows the GRT for the four architectures, as a function of the numbers of articulations. Figure 3b compares the different architectures working at 50MHz. Due to the limitations of the chosen architecture FPGA device, the 50Mhz data for the LEON2 processor has been estimated supposing a half cycle time.

As it was expected, the GRT increases with the number of articulation of the robot. Also, the PowerPC reaches a significantly better result than the other two processors, because it is a hard core with specific hardware functional units.

The most outstanding result is obtained with architecture 2 (LEON + FPU). For an 8 articulations worm-like robot, the performance achieved using the LEON2 with FPU is 40 times better than LEON2 without FPU, using a 6% additional resources. This performance is 6.5 times better than the obtained with the 100Mhz PowerPC implementation.

## 6 Conclusion and further work

The worm like-robot locomotion can be realized by means of the propagation of waves through the body of the robot. The algorithm used generates the gait control tables from the wave applied. The gait recalculation time is the key in order to achieve an autonomous robot with real-time reactions.

The algorithm has been successfully implemented and execute on three different embedded processors in FPGA: LEON2, MicroBlaze and PowerPC. The GRT has been measured in four architectures, as a function of  $n$ , the number of total articulations. Results show that it can be drastically improved by means of the use of an FPU unit. A 25MHz LEON with an Meiko FPU is almost one order of magnitude faster than an PowerPC working at 100Mhz. This is one of the advantages of the use of a FPGA instead of a traditional processor: designers and researchers can improve the robot by introduction architectural changes and adding custom hardware cores.

The LEON2 with an FPU is a very good option when a low GRT is required. In not critical applications the use of the MicroBlaze saves about the 75% of the area, leaving this percentage free for the implementation of new hardware cores.

The current worm like-robot prototype, "Cube Revolutions", can only move on a straight line. The movement on a plane will be studied in further works. The same locomotion algorithm will be used, but calculating two gait control tables from two different waves: one for the articulations in the plane parallel to the ground and the other in the perpendicular plane. The final locomotion will be generated as a composition of the two waves.

## References

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. 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.
4. 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.
5. 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.
6. Xilinx Inc. "MicroBlaze Processor Reference Guide, Embedded Development Kit. Version 6.2"
7. Xilinx Inc. "PowerPC Processor Reference Guide, Embedded Development Kit. Version 6.2"
8. Gaisler Research, "<http://www.gaisler.com>". (March, 2005)