2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Morphing Bus: A rapid deployment computing architecture for high performance, resource-constrained robots

Colin D'Souza, Byung Hwa Kim, and Richard Voyles, *Senior Member, IEEE*

Abstract— **For certain applications, field robotic systems require small size for cost, weight, access, stealth or other reasons. Small size results in constraints on critical resources such as power, space (for sensors and actuators), and computing cycles, but these robots still must perform many of the challenging tasks of their larger brethren. The need for advanced capabilities such as machine vision, application-specific sensing, path planning, self localization, etc. is not reduced by small-scale applications, but needs may vary with the task. As a result, when resources are constrained, it is prudent to configure the robot for the task at hand; both hardware and software. We are developing a reconfigurable computing subsystem for resource-constrained robots that allows rapid deployment of statically configured hardware and software for a specific task. The use of a Field Programmable Gate Array (FPGA) provides flexibility in hardware for both sensor interfacing and hardware-accelerated computation. In this paper, we describe a static reconfiguration architecture we call the *Morphing Bus* that allows the rapid assembly of sensors and dedicated computation through reusable hardware and software modules. It is a novel sensor bus in the fact that no bus interface circuitry is required on the sensor side – the bus "morphs" to accommodate the signals of the sensor.**

I. Introduction

SMALL-SCALE robots fill specific applications needs, but are both size and power limited. Most such robots use microcontrollers to perform their control and feedback tasks in order to conserve space and power. However, heavy computational tasks such as vision, plume tracking, etc often require more computational power than conventional microcontrollers provide. An alternative to power-hungry, full-fledged CPUs for small-scale robots to achieve such heavy-duty tasks is the relative power efficiency of hardware acceleration. One way of providing this is to use Application Specific Integrated Circuits (ASIC) which are custom-designed for a specific task. These, however, are often too constraining for a general purpose robot.

A robot for emergency response might be used for a structural inspection at one moment and then to search for survivors the next. It might even be used with another agency to investigate a suspicious package. Each operation potentially requires a different suite of sensors and/or actuators. Due to size and power constraints it is often not possible (or even necessary) to carry all the sensors and actuators with it for all possible tasks. In this case, the robot could be equipped with limited sensors and actuators to carry out a bomb squad operation; and these would have to be changed for search and rescue in a collapsed building.

Manuscript received September 15, 2006. This work was supported in part by NSF Safety, Security and Rescue Research Center.

C. J. D'Souza is a Masters Student of Electrical Engineering at the University of Minnesota, Minneapolis MN 55455. (dsouz007@umn.edu)

B. H. Kim, is a PhD student of Electrical Engineering at the University of Minnesota, Minneapolis MN 55455. (bhkim@ece.umn.edu)

R. Voyles is with the Computer Engineering Department, University of Denver, Denver, CO 80208 USA. (Richard.Voyles@du.edu)

Robot customization thus makes sense for small scale robots that must make the most of its limited resources, but it causes a burden to the team deploying them. Users are typically not robotic experts and do not spend much training time learning to configure them prior to deployment. The robots should be able to be sent out immediately, but generally valuable time would be spent configuring such a system and customizing it for the task.

This paper aims to address these issues, by proposing a flexible, low cost, high performance system with easy configuration. The system is designed around an FPGA fabric that allows the user the flexibility of using different devices without having to be concerned with the interfacing details. It is based on the novel "morphing bus" concept [10]. The novelty of this architecture is the absence of interface logic on the side of the sensors and actuators with all the interface logic being moved into the FPGA. Every component that is connected to the morphing bus has a module in VHDL associated with it that performs the data processing and interfacing tasks for it. A tool that allows quick and easy configuration of the system prior to deployment was developed. It provides configuration management by abstracting away the details of the individual modules and creates a top level module. The FPGA fabric provides dedicated hardware. Every device is connected directly to the FPGA i.e. to the logic servicing it, which results in low latency. Throughput is increased due to simultaneous operation of all sensors. Besides as the data processing needs are being handled by the FPGA logic the processor is freed for task level functions such as determining the shortest part to target, or making decisions.

We built a prototyping system based around the Virtex-II^{TM} Pro FPGA for our project. The morphing bus uses static reconfiguration to interface devices prior to deployment. The various VHDL modules along with the order in which devices are to be connected are given to the tool developed, which generates the appropriate pin mapping and top level architecture to support that configuration.

The organization of the paper is as follows: section 2 reviews existing systems that use FPGAs either in robotics or for configurable I/O, section 3 explains the Morphing bus, section 4 deals with the tool developed to configure the morphing bus, section 5 demonstrates results using the tool and the new architecture and section 6 summarizes the ideas introduced.

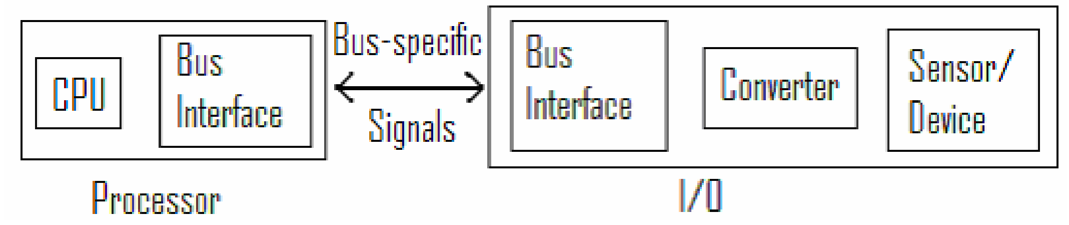

Fig. 1. Standard bus.

II. Literature Review

FPGAs are being increasingly used in robotics due to the very same reasons of flexibility achievable along with the performance of hardware.

A system is described in [1] that has flexible IO peripherals whose interfaces can be added and modified by reconfiguring the embedded FPGA. However the only extent that they went to in that system was to claim that every pin on the chip they fabricated was identical and could be controlled either as an input or output; either by the microcontroller or FPGA.

Rauma et al [2] proposed a system which takes parameters of bus width, number of modules, number of registers etc and a software tool generates the correct structure for the control applications. Templates were created for every module that described the interface to the bus. This bus however was internal to the FPGA.

Guéganno and D. Duhaut [3] use the FPGA as an I/O card but use external interface logic which our system seeks to eliminate completely.

The YaMoR robotic platform [4] can be reconfigured both electrically and mechanically. Reconfiguration of the electronics is achieved using the Spartan-3 FPGA with a Microblaze soft processor. They use the module based flow to reconfigure the FPGA. The module is defined in VHDL and synthesized by the user. They provide scripts for easily generating the corresponding configuration bitstreams for dynamic partial reconfiguration. These scripts however are only valid for their specific robotic controller and are not general.

Goncalves et al [5] presents a framework called ARCHITECT-R for hardware software co-design of FPGAs for mobile robotic applications. Their aim is to allow applications developed in CES (C for Embedded systems) to be translated into hardware/ software components, to be executed in a soft core microprocessor and FPGA hardware structures. They use an existing framework called NENYA [6] to implement a given computing structure in reconfigurable logic. NENYA can extract hardware images from sequential description to be executed by the reconfigurable logic and can even integrate temporal partitioning techniques in the compilation process if designs require more hardware resources than physically available. The goal of that paper was to help design a system; the aim of our paper is to speed up deployment while providing more performance than multiplexed bus architectures and more flexibility than fixed bus architectures.

III. Morphing Bus

Numerous bus protocols exist such as I2C, USB, PCI VMEBus, etc. Some like the USB achieve plug and play capability by storing interface logic on the device. Thus the protocol is able to query the device to gather interface information from it. Also logic is required for bus arbitration in case multiple devices need to be serviced at any given time. This standard bus is depicted in Fig. 1.

The morphing bus exploits the static reconfigurability of the FPGA to provide an interface to modular sensors and actuators without bus interface logic. As seen in Fig. 2 the morphing bus architecture gets rid of the need for interface and arbitration logic by providing a dedicated rather than a multiplexed bus for each device with the flexibility to swap the position of each device. The required data handshaking, data translation and signal processing is done on the FPGA.

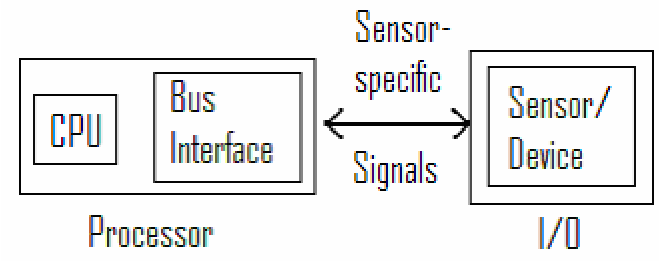

Fig. 2. Morphing bus.

The bus is made up of circuit boards (also called “cheese wedges” because of their shape: see fig 4) each of which is dedicated to only one or more sensors or actuators. The main emphasis is that the boards should be of low complexity and thus small size. Each board has electrical connectors at both ends. All the boards provide the same interface to the preceding and succeeding stages. Thus their position in the bus can be swapped. Each board uses as many bits of the bus as required to support the logic on that wedge and the remaining are fed to the next connector of the next stage which in turn does the same and so on.

Fig 3 shows an example of how the assignment of I/O of the FPGA takes place as boards with different functionalities are added. The input lines to a wedge are used as follows: few initial lines are dedicated to power and ground. These are common to all circuit boards and run through all of them. Starting from the next connection the wedge circuitry uses as many I/O pins as it requires. The remaining lines are shifted to the output connector such that the unused lines are now immediately after the power lines.

In fig 3(a) the bare FPGA base board is shown. This is the heart of the system and in our case consists of only the FPGA and supporting hardware along with a connector to start off the bus. Fig 3(b) shows how the bus starts off. A camera plugged into the FPGA uses required number of lines and the rest are transferred to the start of an output connector. The motor driver board uses 2 of these lines and passes the remaining in a similar fashion. Thus the FPGA pins are assigned sequentially in the same order that the devices are being plugged in. If the positions of two circuit boards in the chain are swapped, the pins of the FPGA connected to each device will differ but overall the same pins will be used.

The morphing bus is currently being designed for use in the TerminatorBot [12] and its structure is shown in Fig. 4. Because of the shapes of the wedges, when they are stacked up they take the form of a spiraling staircase. To provide support to this structure mechanical reinforcements are provided. Air is blown from the base upward, which follows the path along the spiral, cooling the ICs on every wedge. The whole structure is enclosed in wrap to maintain rigidity. If excessive cooling is required the wrap can be made from a conductive material and the various boards can be soldered to the wrap to provide additional conductive cooling.

The number of devices that can be connected in the morphing bus architecture is limited by the number of available pins routed from the FPGA through the wedges, since each board has a dedicated connection to an FPGA pin. This is ultimately determined by size of the connector

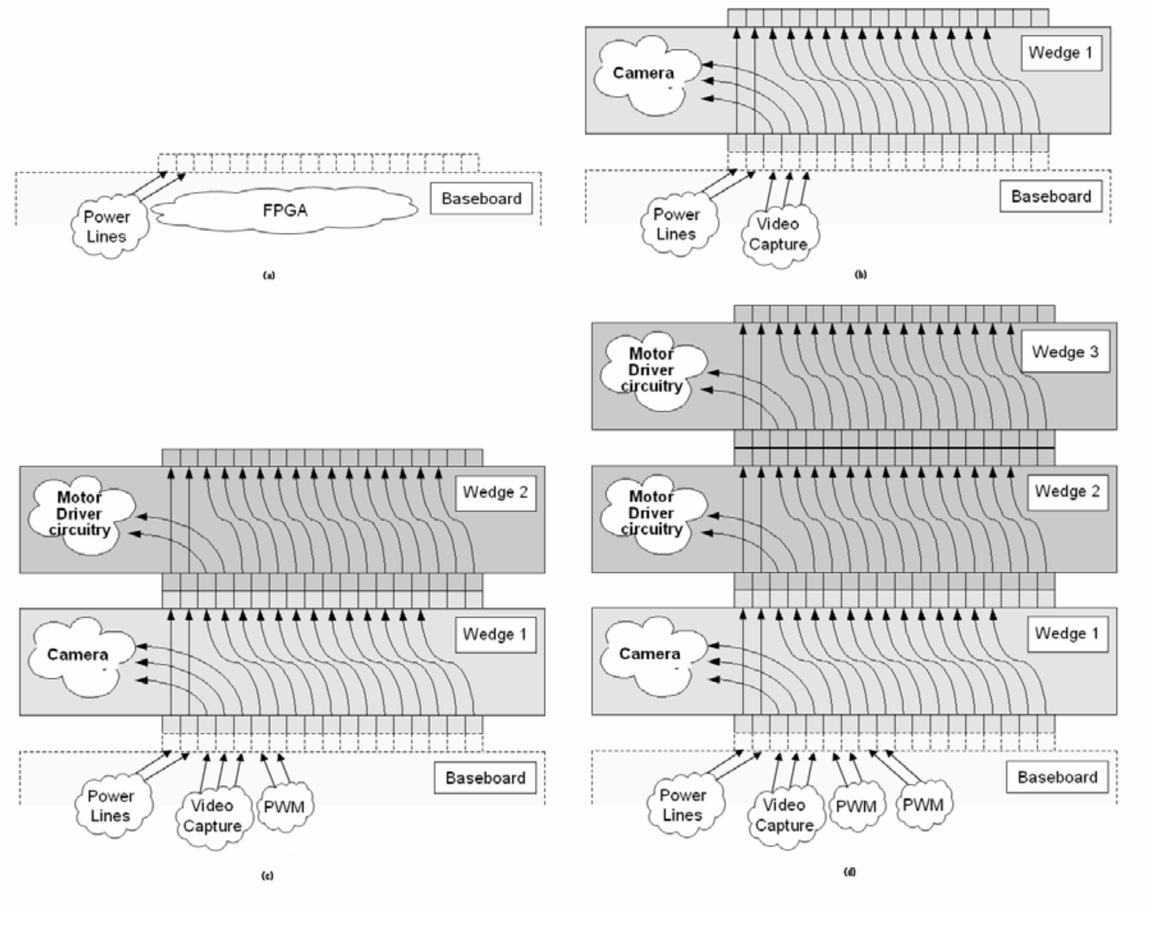

Fig. 3. Wedge diagram for morphing bus. (a) The FPGA base board. (b) When the first circuit board is plugged into the base board, it uses some pins for the component supported and the rest are routed through. (c - d) successive boards are plugged into previous ones, forming a chain and all having direct connections to the base board FPGA.

that can fit on each circuit board which in our case is limited by the size of the robot in which this system is being used. Also a large portion of the wedge is taken up by the pass through routing of the unused lines. However this is acceptable, since although this places an upper limit on the number of devices, we have the great advantage of being able to do without interface and arbitration hardware on the devices plugged in. Thus they can be very small, ideal for deployed field robots.

Another concern is that the boards are not hot swappable i.e. they have to be plugged in and the device has to be configured before the system is turned on. This leads to complexity of configuring the system prior to deployment, and dealing with module replacement at runtime. To simplify system configuration, a tools that takes in the order of the devices and HDL interface descriptions of each and automatically generating a top level file and a corresponding pin configuration file has been developed. These auto generated files can be used in the place and route process.

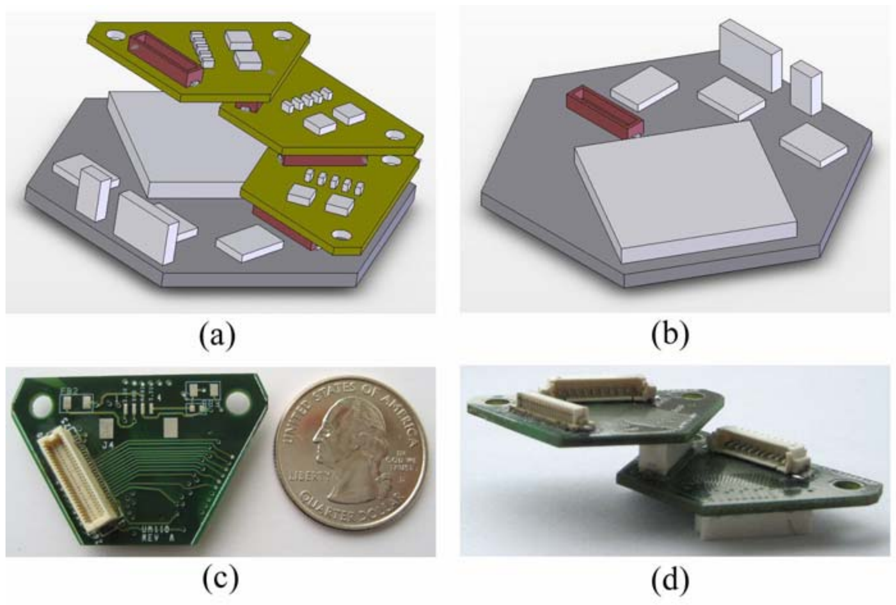

Fig. 4. TerminatorBot Morphing bus spiraling structure. (a) One wedge is connected to the base board, starting off a chain where every wedge is connected to the previous. (b) FPGA base board. (c) A single cheese wedge.

IV. Automating the Bus Configuration

Plug and play based peripherals allow easy and quick system setup. However the devices on the morphing bus do not use interface or arbitration logic, and thus do away with the extra circuitry that allows the host to query the device. Due to this the control program has no way of identifying the device type. This knowledge is essential as the FPGA routing and pin assignment depends on it, to ensure that the appropriate module is interfaced to the sensor/ actuator.

A software configuration tool has been developed to support the morphing bus concept. The tool has a database containing a library of modules in VHDL or netlist format which have been individually compiled and tested. The devices connected to the bus prior to deployment, and the order in which they are to be connected can be selected. In order to configure the FPGA an entire system design file along with the interfacing specification has to be provided to the Xilinx tools. The software takes the configuration assigned above and generates these top level configuration files. This makes it easy to use by operators who are typically not conversant with programming.

The tool is only concerned with the interface between the bus and the module instantiated in FPGA logic. Thus even propriety device cores can be included to handle processing requirements of a device if we know the core interfaces to the external system.

V. Experiments and Observations

In order to verify the functionality and the feasibility of the morphing bus, we built a prototype system as shown in Fig. 5. By its very design the morphing bus would allow direct connections to the FPGA bringing with it the associated advantages. Thus we had to prove that the morphing bus actually worked for different combinations of I/O. Secondly we had to prove that the morphing bus could be used easily by a person who could not write a program yet who would want to swap sensors and actuators.

A Xilinx ML310 development board served as the computing platform and base board. It contained a Virtex-II Pro^{TM} FPGA. For the prototype system we used a 20 bit wide morphing bus. The devices that we supported were a camera, motor with hardware PID position control, some LEDs and switches on boards to simulate other possible I/O combinations. One board consisted only of LEDs to simulate an O/P only board on the morphing bus, a board of only switches simulating an input only device, and one consisting of LEDs and switches for I/O. The motors had optical encoders, the outputs of which were fed into the FPGA, and after processing signals are sent to the motor to control its position. Thus we feel that we had a rigorous setup to test the functioning of the bus. The camera uses commands from the embedded PowerPC core to configure it, and then streams data into the FPGA which handles the data synchronization and saving to memory. Image processing algorithms could potentially be implemented.

We connected the modules in various permutations. Then using the automated tool we generated the top level VHDL module. Using this we programmed the FPGA and tested that all the devices in the bus were working as expected for the different device orderings. At no point did the user have to write a line of VHDL, as all the code was automatically generated.

The tool freed up a lot of time and effort that it would otherwise have taken to configure the bus and made it accessible to people who may not know coding but have to operate the robot in adverse conditions.

VI. Conclusions and Future Work

This paper talks about a powerful yet easy to configure computing platform, based on an FPGA with an embedded processor, and using the Morphing bus concept. This will be the backbone for the TerminatorBot and the heterogeneous wireless sensor network [13]. With the morphing bus, we have a quick and easy to deploy system. It has the flexibility of swapping and adding or removing devices prior to deployment along with dedicated connections to the computing platform without the use of bus interface or arbitration logic. A tool that configures the bus was developed and tested.

We are now looking at being able to add or remove devices at runtime based on the runtime reconfiguration [7], [8], [9] of the FPGA. Augmentation by the control via the PowerPC embedded in the chip, gives us a powerful platform. Also a study is being done towards task aware reconfiguration of the FPGA. Ultimately we hope to have a system capable of task aware reconfiguration.

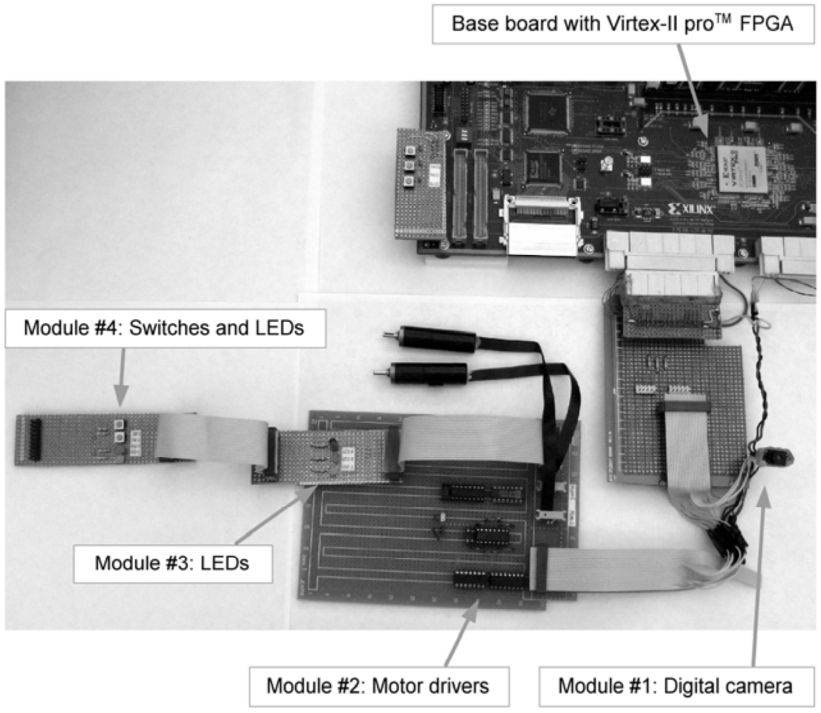
Base board with Virtex-II pro^{TM} FPGA
Module #4: Switches and LEDs
Module #3: LEDs
Module #2: Motor drivers
Module #1: Digital camera

Fig. 5. Experimental Setup - The prototype system showing the base board and “wedges”. The components used are labeled.

References

[1] M. Borgatti, F. Lertora, B. Foret, and L. Cali, “A Reconfigurable System Featuring Dynamically Extensible Embedded Microprocessor, FPGA, and Customizable I/O”, IEEE Journal of Solid-State Circuits, Vol. 38, Issue 3, March 2003, pp. 521 – 529.

[2] K. Rauma, O. Laakkonen, T. Harkonen, O. Pyrhonen, and J. Luukko, “New Bus Structure for Programmable Logic Devices Controlling Power Electronics”, 2005 IEEE 36th Conference on Power Electronics Specialists, June 12, 2005, pp. 2705 – 2708.

[3] C. Guéganno and D. Duhaut, “An hardware/software architecture for the control of self reconfigurable robots”, DARS-2004, France June 2004.

[4] A. Upegui, R. Moeckel, E. Dittrich, A. Ijspeert, and E. Sanchez, “An FPGA Dynamically Reconfigurable Framework for Modular Robotics”, Workshop Proceedings of the 18th International Conference on Architecture of Computing Systems 2005 (ARCS 2005), Berlin, Germany, pp. 83-89.

[5] R. A. Gonçalves, P.A. Moraes, J. M. P. Cardoso, D. F. Wolf, M. M. Fernandes, R. A. F. Romero, E. Marques, “ARCHITECT-R: A System for Reconfigurable Robots Design”, in ACM Symposium on Applied Computing (SAC 2003), March 9-12, Melbourne, Florida, pp. 679-683.

[6] Cardoso, J.M.P.; Neto, H.C.; "Fast hardware compilation of behaviors into an FPGA-based dynamic reconfigurable computing system" XII Symposium on Integrated Circuits and Systems Design, 1999. Proceedings. 29 Sept.-2 Oct. 1999 Page(s):150 - 153.

[7] Xilinx. Two flows for partial reconfiguration: Module based or difference based. Application Note 290, Xilinx, 2004. Xilinx. ISE 8.1i Documentation. Xilinx.

[8] Gregory Mermoud, “A Module-based dynamic partial reconfiguration tutorial”, Logic Systems Laboratory, EPFL, Nov. 2004.

[9] Creating a partially reconfigurable design with Xilinx EDK (Modular Design with EDK) http://wiki.ittc.ku.edu/rtrjvm/EDK_and_MD

[10] B.H. Kim, C. D'Souza, R.M. Voyles, J. Hesch, S. Roumeliotis, “A Reconfigurable Computing Platform for Plume Tracking with Mobile Sensor Networks”, Proceedings of the 2006 SPIE Defense and Security Symposium, Orlando, FL, April, 2006.

[11] W. Zhao, B.H. Kim, A.C. Larson and R.M. Voyles, ¡°FPGA Implementation of Closed-Loop Control System for a Small-Scale Robot¡±, in Proceedings of the 2005 International Conference on Advanced Robotics, Seattle, WA, 2005.

[12] Richard M. Voyles, “TerminatorBot: A Robot with Dual-Use Arms for Manipulation and Locomotion”, in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 2000, pp.61-66.

[13] Jaewook Bae, Amy Larson and Richard Voyles, “Wireless Video Sensor Networks over Bluetooth : High-Bandwidth Multi-Hop Networks for Resource-Constrained Robots” Submitted to ICRA 2007.