

Flexible, FPGA-Based Electronics for Modular Robots

D. Brandt, J.C. Larsen, D.J. Christensen, R.F.M. Garcia, D. Shaikh, U.P. Schultz, and K. Stoy

Abstract—In this paper we introduce electronics for the ATRON self-reconfigurable robot based on field programmable gate arrays (FPGAs). The immediate advantage of using FPGAs is that some of the module’s electronics can be moved into the FPGA, thereby the number of components can be reduced. In the case of the ATRON the number of components is reduced by 20%. Another advantage is that handling of low-level hardware, which is interrupt heavy, can be moved out of the main processor (also implemented on the FPGA) as we will exemplify with a simple FPGA-based communication system. Finally, we can reprogram the FPGA and therefore integrate task-specific electronics without physically changing the electronics or we can reconfigure the electronics for specific tasks. The disadvantages of an FPGA-based design include the cost of FPGAs, the extra layer of complexity in programming, and a limited increase in power consumption compared to micro-controllers. However, overall FPGAs make the electronics of modular robots more flexible and therefore may make them more suitable for real applications.

I. INTRODUCTION

Modular robots are a type of robots built from modules. Modules are mechatronic building blocks that can be connected in many different ways to create robots for a wide range of different tasks. The advantage of a modular approach is that the same set of modules can be reused across applications. The modular approach also improves reliability and robustness either through redundancy or simply because it is easy to replace broken modules. Finally, modular robots can be made relatively cheap compared to their complexity because they typically consist of a limited number of module types that can be mass-produced and thus take advantage of the cost-efficiency of large-scale production.

Research in modular robotics has mainly been focused on solving important, basic research questions related to distributed control and mechatronics design. However, in addition, researchers have recently started a push towards application of modular robots. One prime example of this trend is the ICRA Contingency Challenge initiative where researchers compete to create robotic solutions to tasks that are unknown before hand in a limited amount of time. Competitions like this put emphasis on developing robots and, in our case, modular robots that are practically useful and thus more application oriented in nature.

An insight that has been gained through this competition and by the general push towards application is that while

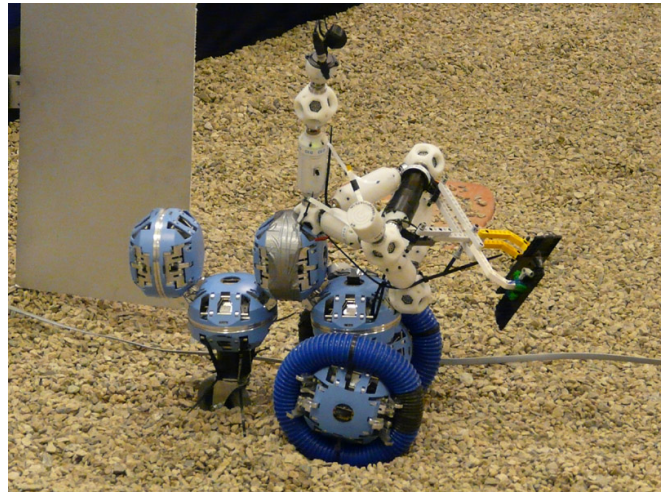


Fig. 1. A heterogeneous modular robot built from ATRON modules, ODIN modules, LEGO, a wireless camera, and various other pieces.

most modules can be reused across many tasks often task-specific modules or at least module adaptations are needed to successfully complete a specific task. An extreme example of this is the modular robot shown in Figure 1. To solve the given task we had to combine two modular robots and further adapt the resulting modular robot using available items such as LEGO and rubber tubes. In general, tasks like this force researchers to design and produce heterogeneous modular robots as opposed to the homogeneous ones we mainly have built until now. The implication of this insight is that we may have to design a wide range of different modules from scratch, which is problematic because it is time consuming and costly.

This problem can be reduced if we can make modules that are more flexible. One way to achieve this, at least at the level of electronics, is to use field programmable gate arrays (FPGAs) because their programmability allow us to integrate task-specific electronics into the module relatively easily. Another problem we hope to solve with an FPGA-based solution is that processors of modular robots spent a significant amount of processing time handling interrupts caused by hardware. In the ATRON self-reconfigurable robot, for instance, communication takes a significant amount of processing power because each processor of which there are two have five communication channels. However, the main problem may not be the processing power since the communication speed is modest, but writing reliable code that works even when continually interrupted. The FPGA may alleviate this problem by handling most of the hardware

The authors are with The Modular Robotics Lab, University of Southern Denmark. david.brandt@mmmi.sdu.dk, joela05@sduemail.sdu.dk, [david, franco, danish, ups, kaspers]@mmmim.sdu.dk

This work was supported by The Danish Council for Technology and Production.

interfacing outside the main processor core (a complete processor core can be implemented on an FPGA and works just like a traditional processor). In other words, the main processor can be dedicated to application specific code, and thus the task of writing controllers may be made easier. Finally, logical components that are normally put outside the micro-controller can be implemented directly in the FPGA and thus reduce the number of components used. The disadvantages of FPGAs compared to micro-controllers include their higher cost, slightly increased power consumption and an increased complexity of the whole system. However, we feel that FPGAs in the longer term have the potential to make heterogeneous modular robots more homogeneous from an electronics perspective and make it easier to integrate new electronics into already existing modules and, thereby, allow us to keep a significant part of the electronics homogeneous across otherwise heterogeneous, flexible modules.

In this paper we present FPGA-based electronics for the ATRON self-reconfigurable robot. We also describe a simple FPGA-based message routing system which is an example of how we can free the main processor from the load of handling communication and this also serves as an example of how we in general plan to use the FPGA.

II. MODULAR ROBOTS

The ATRON robot [8] belongs to a class of modular robots called self-reconfigurable robots. The modules of this type of robot are able to connect to and disconnect from neighbor modules and can, either by themselves or with the help of neighbor modules, move in a structure of other modules. This characteristic of the modules makes it possible for the robot as a whole to change its shape or in other words self-reconfigure. Also belonging to this class of robots are the M-TRAN robot [7], PolyBot [10], SuperBot [9], Molecube [13], and several others (see [11] for a nice overview of the field). In terms of electronics these robots all share a solution centered around a few central micro-controllers.

Reconfigurable modular robots do not have the ability to self-reconfigure and thus typically rely on a user to assemble them. One example of such a system is the ODIN robot that we have developed [4]. This robot is a heterogeneous modular system in which we use another approach to maintain reusability of electronics across modules. In this system each module is equipped with a general circuit board that takes care of functions such as processing, communication, and power distribution. In addition each type of module is equipped with a specific board that takes care of the functionality specific to each type of module. The advantage of this approach is that in order to design electronics for a new type of module only the specific board needs to be redesigned while the general board can be reused. Another example of a reconfigurable robot is the YaMoR robot [6] which interestingly enough is also based on an FPGA, but the potential of the FPGA has only been realized to a limited degree [5]. Other examples of heterogeneous modular robots include a simplified version of the Molecube [12] and the ckBot, but these systems are based on micro-controllers.

III. FPGA-BASED ELECTRONICS FOR ATRON

In this section we will first describe the design of the FPGA-based electronics for the ATRON modular robot and compare this design with that of the micro-controller-based electronics. Following this we will explain how we have implemented a simple system for routing messages in the robot to take some load of the main processor core (implemented in the FPGA).

A. ATRON

Let us just briefly introduce the ATRON modular robot. A more complete description can be found in [8].

The ATRON is a lattice-based modular self-reconfigurable robot system consisting of homogeneous modules. The basic module design is based on two hemispheres connected by a rotational joint. Further, each hemisphere is equipped with four connectors, two male and two female; the male/female connector design is chosen for mechanical reasons. The modules are equipped with a total of 8 infra red transmitters and receivers used for local communication between modules.

B. ATRON FPGA-based Electronics

The main goal of the FPGA-based ATRON electronics is to evaluate the usefulness of FPGAs in the context of modular robots. The immediate advantage of the FPGA is that it allows us to remove most of the external logic components from the circuit board and embed them in the FPGA instead. In our design we moved around 50 of the 250 components used in the micro-controller-based design, including complex components such as a brushless motor controller and serial to IrDA signal converters. The number of components needed for the center motor driver, for instance, is reduced from 34 to 12 components.

The overall design is much the same as in traditional systems based on micro-controllers. The central unit is the FPGA, which can contain a micro-controller. Since all the logic is embedded directly in the FPGA only the power drivers, A/D-converters and sensors are left outside. Besides the FPGA itself a boot flash is required as FPGAs are based on RAM technology. We chose the Xilinx Spartan3e-1200 FPGA due to its power efficiency and relatively low cost.

The design is currently being tested, but compared to the existing design based on ATmega128 micro-controller, the FPGA-based design will contain many advantages and several new features.

- **Processor** In the existing ATRON electronics the main processing is performed by two ATmega128 8-bit 16MHz micro-controllers. Producing a maximum of 16 MIPS each. The FPGA based design will most likely be using a MicroBlaze 32-bit soft-core as the main processing unit in each hemisphere. The MicroBlaze can run at speeds up to 100MHz in the Spartan3e FPGAs, producing up to 115 MIPS according to Xilinx. Further, it is possible to include caching, a floating point unit, and a memory management unit in the processor. Both commercial and open Linux ports for the MicroBlaze

exist. If needed each FPGA can implement multiple MicroBlaze cores.

- **Memory** The micro-controller-based electronics contains only the memory embedded in the ATmega128, that is 128Kbyte of flash, 4Kbyte of SRAM and 4Kbyte of EEPROM. The FPGA-based MicroBlaze can be configured with up to 64Kbyte of internal RAM. Furthermore, we have added 16Mb of external DDR-RAM. A boot flash is also used as program memory and can contain 8Mbyte of which 512Kbyte is reserved for booting the FPGA. Again these figures are for each hemisphere.
- **Wireless Communication** The FPGA-based design also includes wireless communication using the Zigbee protocol. Besides the possibility to communicate wirelessly with the modules, the possibility to enter and exit low-power sleep mode based on wireless communication has been implemented. Expected battery life in sleep mode is roughly one month.
- **Debug Communication** In the FPGA-based design serial debug output from each hemisphere through USB has been implemented running at speeds of up to 115.2Kbps. Drivers are native to all major operating systems: Windows, Linux and Mac OS X. The micro-controller-based design uses a software UART for debugging, which is limited to 2.4Kbps
- **Accelerometers** In the FPGA-based design each hemisphere contains a three-axis accelerometer. In the micro-controller-based design the accelerometers has only two axes each.
- **UARTs** The ATmega128 only contains two UARTs, which leads to multiplexing one UART between 4 infrared communication channels. In the FPGA-based design this can be avoided by implementing as many UARTs as needed for the design. The new design uses a total of 7 UARTs in one hemisphere and 6 in the other.
- **Proximity sensors** Previously the IR communication diodes were multiplexed with the proximity sensing. In the new design dedicated proximity sensors are implemented.
- **IO-ports** Due to the very large number of IO pins on the FPGA a total of 11 8-bit fully configurable, general purpose IO-ports have been implemented, as opposed to just one 8-bit port in the existing system.

The power consumption of the FPGA is the main disadvantage of the FPGA-based design. Depending on the implementation and the clock speed the FPGA can draw 100-300mA at 3.3V. This is several orders of magnitude more than the ATmega128. However, compared to the power consumption of the motors in order of 1-2A this is not a dramatic increase in the overall module power consumption.

The cost of a Spartan3e-1200 is roughly 35 USD compared to the 15 USD of the ATmega128. These numbers are for low quantities. The 320 pin FBGA (Fine pitch Ball Grid Array) package of the Spartan FPGA requires more than a soldering

iron and a two layer PCB to mount, which increases the prototyping cost since small series production is expensive.

The increased price due to the FPGA itself is somewhat counter-balanced by the reduced number of components. In order to further reduce the prototyping cost, we have chosen to mount the FPGA, boot flash and DDR RAM on a small add-on board, instead of the main ATRON board. The size of the FPGA board is 27 by 64 by 4.6 mm, including the board to board connectors. This allows us to use the exact same add-on board in both hemispheres. It removes the demand for manufacturing the large ATRON board in 8-layers which is necessary for the FPGA board. Moreover, it makes it possible to use the same FPGA board in other projects. The interface to the FPGA board is simply a 3.3V power supply, JTAG for programming and 100 completely configurable IO pins.

Besides all the extra features, and the reduced part count there is still one important advantage of using an FPGA: it allows us to redesign the hardware after the module is built, and to design complex logic circuits quite easily. These circuits can be used for taking load of the main processor core, by moving parts of the control from software to hardware. Let us look at a simple example and return to a more complex one in the following section. Disconnecting one of the four male connectors in the ATRON is done by a simple DC-motor. The basic operation is to apply current to the motor until a pin, which indicates that the connector is completely disconnected, goes high. In a micro-controller-based design, this would usually be done in one of two ways. The pin could be connected to an interrupt, such that when the disconnection was complete the processor would get interrupted and turn off the current. Alternatively, if an interrupt is not available the micro-controller must poll the pin at regular intervals. Using a logic circuit this task can be done completely by the hardware or a mix of hardware and software. Once the disconnection is complete the current is automatically turned off by dedicated logics. The processor core will not need to spend any processing time once the disconnection has been initiated.

Using an FPGA in the design adds a layer of complexity to the design process. The logic that has been removed from the board itself has to be implemented in the FPGA. This is done using a hardware description language (HDL), typically Verilog or VHDL. This is of course quite different from conventional hardware design, but since it is a matter of compiling and downloading the design, without any delay for soldering etc., the prototyping phase is much faster and more dynamic. For the Spartan series FPGAs, Xilinx provides a large library of logic components ready to integrate in the design. Besides the MicroBlaze core this includes components like UARTs, SPI controllers, interrupt controllers and brushless motor controllers among many other components.

C. FPGA-Based Communication Controller

Communication internally in a robot build from ATRON modules is based on completely distributed neighbor to neighbor communication. This implies that when distant modules need to communicate they need to route commu-

nication packages through other modules for which these packages are irrelevant.

In the micro-controller-based solution each micro-controller has to handle five communication channels, four infrared and one RS485. The last of which is used to communicate between the two hemispheres. The processors have to analyze all communication packages even if most of them are for another module in the robot. This can be quite processor intensive: in the worst case, where all five communication channels are active, the processor spends around 14% of its processing time just handling interrupts, but more importantly these frequent interrupts make programming of the modules challenging.

The introduction of FPGAs opens an opportunity to solve this problem, by placing a communication controller inside the FPGA, but outside the soft-core processor. The proposed new communication controller is inspired by several known techniques from old telephone systems using virtual circuit networks and switches used in modern PC networks (see for instance [2] p. 22 and 315, respectively). The idea behind the communication controller is that a sender module first sets up a communication line through the robot to the intended receiver module. Once a line is established the modules communicate without interrupting the intermediate modules. When the communication has been completed the line is disconnected and new communication lines can be formed as needed.

The communication controller is implemented in the FPGA, see Figure 2 for an overview and [3] for details. The heart of the controller is a *routing matrix* that routes messages from receiver to transmitter UARTs. The configuration of the routing matrix is stored in the *routing table* which is implemented as block RAM. In normal operation mode the communication controller runs by itself, and does not interrupt the MicroBlaze core (*MB*). The processor is only interrupted when a package for it is received or if the routing table needs to be reconfigured. Currently, only the MicroBlaze processor can update the routing table, but it could also be interesting to allow other modules to update the routing table directly.

Using this communication controller it is possible to implement both a local communication system and hybrid communication buses, inspired by the hybrid communication bus of the ODIN robot [1]. In order to implement this functionality we have introduced a one-bit header that indicates whether a packet is to be routed to the MicroBlaze or through the routing matrix to neighbor modules. If a module want to set up a communication line, it simply sends local packages between modules all the way through the chain of modules, to the recipient module which configures the routing matrix. When this is done, the communication line is open, and the sending module can transfer all its data to the recipient through the chain, without interrupting any processors along the way. The processors will then only be interrupted when setting up the line, but not by each package transmitted through the line. The line is finally disconnected using local messages. It takes approximately 17,37 μ s to set up the rout-

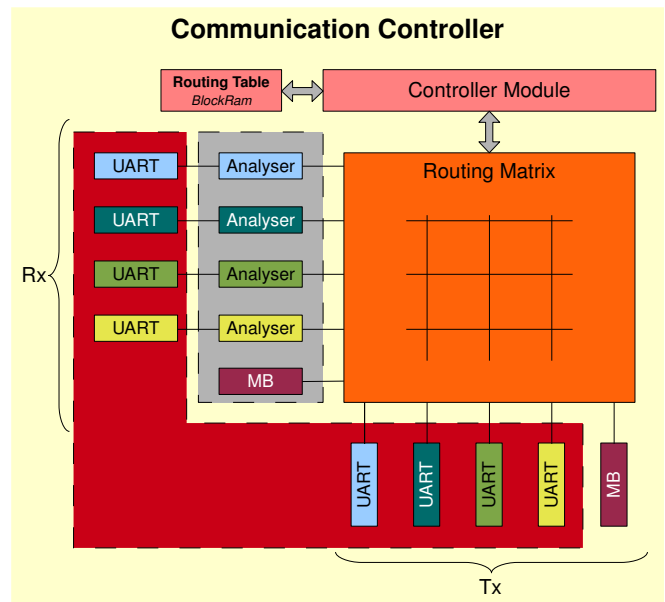


Fig. 2. An overview of the FPGA-based communication controller. See main text for details.

ing table and clear it afterwards, if the MB runs at 16MHz like the ATmega128. The communication controller has been successfully tested on three FPGA prototyping boards, but needs to be tested properly once the ATRON electronics is ready for use.

An interesting feature of this communication system we still have to explore is that it may reduce communication in the system because data messages are not just broadcasted, but can be sent down an isolated communication line. But the most important point may be that if this routing system does not work another one can be developed without changing the electronics.

IV. CONCLUSION AND FUTURE WORK

The preliminary work that we have done so far demonstrates that we can simplify the electronics of individual modules by incorporating FPGAs as well as making the electronics extensible. Furthermore, the parallel nature of an FPGA allows us to implement functionality, such as the demonstrated routing system, in the FPGA and thus free the main processor from such tasks. However, FPGAs also introduce more complexity, cost slightly more than micro-controllers, and maybe more importantly consume more power. Nevertheless, we hypothesize that FPGAs may be an effective way to increase the flexibility of modules by making it easier to adapt their electronics for specific applications.

V. ACKNOWLEDGEMENTS

This research is funded by the Danish Council for Technology and Production through the project “Morphing Production Lines”.

REFERENCES

- [1] R.F.M. Garcia, D.J. Christensen, K. Stoy, and A. Lyder. Hybrid approach: A self-reconfigurable communication network for modular robots. In *Proceedings, First international conference on robots and communication*, Athens, Greece, 2007.
- [2] J.F. Kurose and K.W. Ross. *Computer Networking - A Top-Down Approach Featuring The Internet*. Addison-Wesley, 3rd edition, 2004.
- [3] J.C. Larsen. Development of a hybrid communication structure for modular robot systems. Technical report, University of Southern Denmark, 2008. (<http://modular.mmmi.sdu.dk>).
- [4] A. Lyder, R.F.M. Garcia, and K. Stoy. Mechanical design of odin, an extendable heterogenous deformable modular robots. In *Proceedings, IEEE/RSJ international conference on intelligent robots and systems*, Nice, France, 2008. (to appear).
- [5] R. Moeckel, C. Jaquier, K. Drapel, E. Dittrich, A. Upegui, and A.J. Ijspeert. Exploring adaptive locomotion with YaMoR, a novel autonomous modular robot with Bluetooth interface. *Industrial Robot*, 33(4):285–290, 2006.
- [6] R. Moeckel, C. Jaquier, K. Drapel, A. Upegui, and A. Ijspeert. YaMoR and bluemove – an autonomous modular robot with bluetooth interface for exploring adaptive locomotion. In *Proceedings CLAWAR 2005*, pages 685–692, 2005.
- [7] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji. M-TRAN: self-reconfigurable modular robotic system. *IEEE/ASME Transactions on Mechatronics*, 7(4):432–441, 2002.
- [8] E.H. Østergaard, K. Kassow, R. Beck, and H.H. Lund. Design of the ATRON lattice-based self-reconfigurable robot. *Autonomous Robots*, 21(2):165–183, 2006.
- [9] B. Salemi, M. Moll, and W.-M. Shen. Superbot: A deployable, multi-functional, and modular self-reconfigurable robotic system. In *Proceedings, IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 3636–3641, Beijing, China, 2006.
- [10] M. Yim, D.G. Duff, and K.D. Roufas. PolyBot: A modular reconfigurable robot. In *Proc., IEEE Int. Conf. on Robotics & Automation*, pages 514–520, San Francisco, CA, USA, 2000.
- [11] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G.S. Chirikjian. Modular self-reconfigurable robot systems. In *IEEE Robotics & Automation Magazine*, pages 43–52, March 2007.
- [12] V. Zykov, A. Chan, and H. Lipson. Molecubes: An open-source modular robotics kit. In *Proceedings of the 2007 IEEE/RSJ Int. Conference on Robots and Systems, Self-Reconfigurable Robotics Workshop*, San Diego, CA, USA, 2007.
- [13] V. Zykov, E. Mytilinaios, B. Adams, and H. Lipson. Self-reproducing machines. In *Nature*, volume 435, pages 163–164, 2005.